

```

SSSSSSSSSSSS 00000000 RRRRRRRRRR TTTTTTTTTTTTTT 33333333 22222222
SSSSSSSSSSSS 00000000 RRRRRRRRRR TTTTTTTTTTTTTT 33333333 22222222
SSSSSSSSSSSS 00000000 RRRRRRRRRR TTTTTTTTTTTTTT 33333333 22222222
SSS          000      000 RRR          RRR TTT          333      222
SSS          000      000 RRR          RRR TTT          333      222
SSS          000      000 RRR          RRR TTT          333      222
SSS          000      000 RRR          RRR TTT          333      222
SSS          000      000 RRR          RRR TTT          333      222
SSS          000      000 RRR          RRR TTT          333      222
SSSSSSSSSS 000      000 RRRRRRRRRR TTT          333      222
SSSSSSSSSS 000      000 RRRRRRRRRR TTT          333      222
SSSSSSSSSS 000      000 RRRRRRRRRR TTT          333      222
SSS          000      000 RRR  RRR TTT          333      222
SSS          000      000 RRR  RRR TTT          333      222
SSS          000      000 RRR  RRR TTT          333      222
SSS          000      000 RRR  RRR TTT          333      222
SSS          000      000 RRR  RRR TTT          333      222
SSS          000      000 RRR  RRR TTT          333      222
SSS          000      000 RRR  RRR TTT          333      222
SSSSSSSSSS 00000000 RRR          RRR TTT          33333333 22222222
SSSSSSSSSS 00000000 RRR          RRR TTT          33333333 22222222
SSSSSSSSSS 00000000 RRR          RRR TTT          33333333 22222222

```

```

SSSSSSSS 000000 RRRRRRRR CCCCCCCC 000000 LL UU UU TTTTTTTTTT IIIIII
SSSSSSSS 000000 RRRRRRRR CCCCCCCC 000000 LL UU UU TTTTTTTTTT IIIIII
SS      00      00 RR      RR CC      00      00 LL UU UU TT      TT IIII
SS      00      00 RR      RR CC      00      00 LL UU UU TT      TT IIII
SS      00      00 RR      RR CC      00      00 LL UU UU TT      TT IIII
SSSSSS 00      00 RRRRRRRR CCCCCCCC 00      00 LL UU UU TT      TT IIII
SSSSSS 00      00 RRRRRRRR CCCCCCCC 00      00 LL UU UU TT      TT IIII
SS      00      00 RR      RR CC      00      00 LL UU UU TT      TT IIII
SS      00      00 RR      RR CC      00      00 LL UU UU TT      TT IIII
SS      00      00 RR      RR CC      00      00 LL UU UU TT      TT IIII
SSSSSSSS 000000 RRR      RR CCCCCCCC 000000 LLLLLLLLLL UUUUUUUUUU TT      TT IIIIII
SSSSSSSS 000000 RRR      RR CCCCCCCC 000000 LLLLLLLLLL UUUUUUUUUU TT      TT IIIIII

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

```

Vertical text on the right edge of the page.

```

1 0001 0 MODULE COLLSUTILITIES(
2 0002 0 IDENT = 'V04-000
3 0003 0 ) = ! File: SORCOLUTI.B32 Edit: PDG3014
4 0004 1 BEGIN
5 0005 1
6 0006 1 *****
7 0007 1 *
8 0008 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY *
9 0009 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *
10 0010 1 * ALL RIGHTS RESERVED. *
11 0011 1 *
12 0012 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
13 0013 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *
14 0014 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *
15 0015 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
16 0016 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *
17 0017 1 * TRANSFERRED. *
18 0018 1 *
19 0019 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *
20 0020 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *
21 0021 1 * CORPORATION. *
22 0022 1 *
23 0023 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *
24 0024 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *
25 0025 1 *
26 0026 1 *
27 0027 1 *****
28 0028 1
29 0029 1
30 0030 1 ++
31 0031 1
32 0032 1 FACILITY: VAX-11 SORT/MERGE
33 0033 1 PDP-11 SORT/MERGE
34 0034 1
35 0035 1 ABSTRACT:
36 0036 1
37 0037 1 This module contains routines that process a user-defined collating
38 0038 1 sequence.
39 0039 1
40 0040 1 ENVIRONMENT: VAX/VMS user mode
41 0041 1
42 0042 1 AUTHOR: Peter D Gilbert, CREATION DATE: 20-Jan-1983
43 0043 1
44 0044 1 MODIFIED BY:
45 0045 1
46 0046 1 T03-001 Original
47 0047 1 T03-002 Add a temporary fix to get a reasonable pad character if
48 0048 1 the pad character is ignored. PDG 26-Jan-1983
49 0049 1 T03-003 Support ignored pad characters. Set ADJ to zero if there are
50 0050 1 ignored characters. PDG 28-Jan-1983
51 0051 1 T03-004 Add COLL$FOLD. PDG 31-Jan-1983
52 0052 1 T03-005 Define CODE and PLIT psects. 1-Feb-1983
53 0053 1 T03-006 Change the interface to SOR$$COLLATE x. PDG 7-Mar-1983
54 0054 1 T03-007 Remove STATIC table stuff. Changes for PDP-11 compatability.
55 0055 1 PDG 5-Apr-1983
56 0056 1 T03-008 Changes to simplify zapping the upper table. PDG 12-Apr-1983
57 0057 1 T03-009 Store info in RES_REVERSE (not CS_REVERSE) in COLL$RESULT.

```

COLL\$UTILITIES
V04-000

C 11
16-Sep-1984 01:06:02 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 13:10:40 [SORT32.SRC]SORCOLUTI.B32;1

Page 2
(1)

:	58	0058	1	!	
:	59	0059	1	!	The ORDER parameter to TIE_BREAK is now required. Changed
:	60	0060	1	!	value of CS_K_REG for Bliss-11. PDG 15-Apr-1983
:	61	0061	1	!	T03-010 Defined error statuses for Bliss-11. PDG 21-Apr-1983
:	62	0062	1	!	T03-011 Add routine headers. PDG 5-Jul-1983
:	63	0063	1	!	T03-012 Allocate large structures in the work area, not on the stack.
:	64	0064	1	!	PDG 25-Apr-1983
:	65	0065	1	!	T03-013 Merge changes from Sort-11 and Sort-32 versions. 19-Sep-1983
:	66	0066	1	!	T03-014 Allocate on the stack for Sort-32. Specify a comparison
:	67	0067	1	!	routine that can be used after an initial CMPC for Sort-32
:	68	0068	1	!--	PDG 14-Oct-1983

```

70 0069 1
71 0070 1
72 0071 1
73 0072 1
74 0073 1
75 0074 1
76 0075 1
77 0076 1
78 0077 1
79 0078 1
80 0079 1
81 0080 1
82 0081 1
83 0082 1
84 0083 1
85 0084 1
86 0085 1
87 0086 1
88 0087 1
89 0088 1
90 0089 1
91 0090 1
92 0091 1
93 0092 1
94 0093 1
95 0094 1
96 0095 1
97 0096 1
98 0097 1
99 0098 1
100 0099 1
101 0100 1
102 0101 1
103 0102 1
104 0103 1
105 0104 1
106 0105 1
107 0106 1
108 0107 1
109 0108 1
110 0109 1
111 0110 1
112 0111 1
113 0112 1
114 0113 1
115 0114 1
116 0115 1
117 0116 1
118 0117 1
119 0118 1
120 0119 1
121 0120 1
122 0121 1
123 0122 1
124 0123 1
125 0124 1
126 0125 1

```

++

OVERVIEW:

The routines must be called in the following order:
INIT [BASE] [NEXT ! MODIFY ! FOLD]... RESULT
The routines PAD, TIE_BREAK and UPPER may be optionally called any time after the INIT and before the RESULT.

In all of these routines, the user passes a two element vector containing the length/address of a work area these routines can use. The call to RESULT returns the length that is needed to store the compressed version of the area. The user can then call the routine whose address is stored at the beginning of the area. This routine is passed the lengths/address of the strings, and returns:

```

-1 if String1 < String2
0  if String1 = String2
+1 if String1 > String2

```

All characters are passed to these routines as a word length followed by zero, one or two characters (4 bytes max). The routine INIT simply initializes all characters as ignored, the pad character as the null character, and no tie-breaking.

BASE defines a base collating sequence (via a 256 byte table). All 256 single-byte characters are given one-byte collating values, taken from the table.

NEXT specifies a character that is to get a single-byte collating value that collates larger than any other currently defined collating value.

OTHERS causes NEXT to be called for all currently ignored single-byte characters (similar to COBOL-style definitions).

MODIFY defines a character to collate just less than, equal to, or just greater than the (0,1,or 2 byte) collating value of a (0,1,or 2 byte) character string.

FOLD causes all lower case letters to be given the collating values of their upper case equivalents. If a double character that contain no lower case letters is defined, then lower case and mixed case double characters are defined to collate equal to this double character.

For example,

This	causes these definitions
'd\$'	'd\$'='d\$'
'\$d'	'\$d'='\$d'
'd\$'	none
'\$d'	none
'xy'	none
'xY'	none
'xY'	'xy'='xY', 'xY'='xY', 'xY'='xY'

PAD defines the (single byte) pad character.

UPPER specifies a simple (i.e., like BASE), secondary collating sequence that should be applied if the primary collating sequence collates two strings as equal.

```

127 0126 1
128 0127 1
129 0128 1
130 0129 1
131 0130 1
132 0131 1
133 0132 1
134 0133 1
135 0134 1
136 0135 1
137 0136 1
138 0137 1
139 0138 1
140 0139 1
141 0140 1
142 0141 1
143 0142 1
144 0143 1
145 0144 1
146 0145 1
147 0146 1
148 0147 1
149 0148 1
150 0149 1
151 0150 1
152 0151 1
153 0152 1
154 0153 1
155 0154 1
156 0155 1
157 0156 1
158 0157 1
159 0158 1
160 0159 1
161 0160 1
162 0161 1
163 0162 1
164 0163 1
165 0164 1
166 0165 1
167 0166 1
168 0167 1
169 0168 1
170 0169 1
171 0170 1
172 0171 1
173 0172 1
174 0173 1
175 0174 1
176 0175 1
177 0176 1
178 0177 1
179 0178 1
180 0179 1
181 0180 1
182 0181 1
183 0182 1

```

TIE_BREAK specifies that, if the primary and secondary collating sequences collate the strings as equal, a final comparison should be done which compares the unsigned binary values of the characters in the strings.

The linkage to the comparison routine is:
(for VAXen):

```

JSB(
REGISTER=0,      ! Length of String1
REGISTER=1,      ! Address of String1
REGISTER=2,      ! Length of String2
REGISTER=3,      ! Address of String2
REGISTER=5,      ! Address of table
REGISTER=0):     ! Result of the comparison (-1, 0, +1)
NOPRESERVE(4,5)
PRESERVE(9,10)
NOTUSED(6,7,8,11)

```

(for PDP-11s):

TBS

And the condition codes reflect the setting of R0.

IMPLEMENTATION:

During the definition of the collating sequence, collating values are represented by two word values: <x,y>.

```

<0,0> indicates an ignored character
<x,0> indicates a single-value collating value
<x,y> indicates a double-value collating value

```

The collating values for single characters are stored in a 256 element array. Double characters and their collating values are stored in a sequential list at the end of the other tables. (The UPPER table is always left in byte form).

The succinct tables generated by RESULT have the form:

```

RES$TB      A byte of flags for tie-break and upper
RES$REVERSE A byte of flags to reverse sense of tie-break CMPC
RES$PAD     A byte for the pad character
RES$PTAB    A 256 byte table, with a value of zero indicating
             that the STAB table must be consulted.
RES$UPPER   A 256 byte table.
RES$STAB    A list of entries for double characters and characters
             with double collating values. If a character with a
             value of 0 in PTAB is not found in this table, it is
             ignored.

```

The flags within TB are:

```

TB$NOTB    'XB'0100'    Don't do tie-breaking (the Z-bit)
TB$NOUPPER 'XB'0010'    Don't use upper table (the V-bit)

```

The flags within REVERSE are:

```

TB$REVERSE 'XB'0001'    Reverse tie-break CMPC (the C-bit)

```

An entry in STAB is four bytes in length:

```

<ch0,ch1,cv0,cv1>
The ch0,ch1 together form a single or double character.
The cv0,cv1 together form a single or double collating value.

```

```

184 0183 1 | with special forms mentioned above.
185 0184 1 | These entries are ordered in groups with equal ch0 values, in order
186 0185 1 | of increasing ch0 values. The groups are followed by two "trailer"
187 0186 1 | entries:
188 0187 1 |     <%X'FF',%X'FF',.....>
189 0188 1 |     <%X'00',%X'00',.....>
190 0189 1 | Each group has the form:
191 0190 1 |     <x,%X'FF', collating value of x>   One of these
192 0191 1 |     <x, y , collating value of xy>    0 or more, ordered by y
193 0192 1 |
194 0193 1 | The choice of this representation is succinct and allows for efficient
195 0194 1 | processing. See the support routines SOR$COLLATE_0, 1 and 2 for more
196 0195 1 | details.
197 0196 1 |
198 0197 1 | --
199 0198 1 | LIBRARY 'SYS$LIBRARY:XPORT';
200 0199 1 | %IF %BLISS(BLISS32) %THEN
201 0200 1 | PSECT
202 0201 1 |     CODE=          SOR$RO_CODE(PIC,SHARE),
203 0202 1 |     PLIT=          SOR$RO_CODE(PIC,SHARE);
204 0203 1 | %FI
205 0204 1 |
206 0205 1 | LITERAL
207 0206 1 |     CS_K_REG = %IF %BLISS(BLISS32) %THEN 10 %ELSE 3 %FI;
208 0207 1 | MACRO
209 0208 1 |     LNK_CALL = %IF %BLISS(BLISS32) %THEN CALL %ELSE JSR %FI %,
210 0209 1 |     LNK_SUBR = %IF %BLISS(BLISS32) %THEN JSB %ELSE JSR %FI %;
211 0210 1 | LINKAGE
212 0211 1 |     CS_LINK_0 = LNK_SUBR: GLOBAL(CS=CS_K_REG),
213 0212 1 |     CS_CALL_0 = LNK_CALL: GLOBAL(CS=CS_K_REG),
214 0213 1 |     CS_LINK_1 = LNK_SUBR(REGISTER=1): GLOBAL(CS=CS_K_REG),
215 0214 1 |     CS_LINK_2 = LNK_SUBR(REGISTER=1, REGISTER=2): GLOBAL(CS=CS_K_REG);
216 0215 1 |
217 0216 1 | FORWARD ROUTINE
218 0217 1 | D_LOOKUP: CS_LINK_1, | Look up a double character
219 0218 1 | D_NEW: CS_LINK_1, | Create a new secondary table entry
220 0219 1 | GIVE_COLL: CS_LINK_2, | Assign a collating value to a character
221 0220 1 | DO_BOMP: CS_LINK_1, | Increase collating values
222 0221 1 | COLL$INIT, | Initialize collating sequence
223 0222 1 | COLL$BASE, | Define the base collating sequence
224 0223 1 | COLL$NEXT, | Define the next character
225 0224 1 | ! COLL$OTHERS, | Define undefined single characters
226 0225 1 | COLL$MODIFY, | Make a modification
227 0226 1 | COLL$FOLD, | Fold upper/lower case characters
228 0227 1 | COLL$TIE_BREAK, | Indicate tie-breaking
229 0228 1 | COLL$PAD, | Indicate collating value of the pad character
230 0229 1 | COLL$UPPER, | Upper case comparison
231 0230 1 | COLL VALUE: CS_LINK_2, | Gets the collating value of a character
232 0231 1 | COMPRESS: CS_LINK_1, | Compress the range of collating values
233 0232 1 | COMPRESS M: CS_LINK_0, | Compress the tables and set attributes
234 0233 1 | COLL$RESULT; | Build the final tables
235 0234 1 |
236 0235 1 |
237 0236 1 | ! Define the error statuses returned by these routines
238 0237 1 |
239 0238 1 | %IF %BLISS(BLISS32) %THEN
240 0239 1 |

```

```

: 241      0240  1  EXTERNAL LITERAL
: 242      0241  1      SORS_COL_ADJ,
: 243      0242  1      SORS_COL_CMPLX,
: 244      0243  1      SORS_COL_CHAR,
: 245      0244  1      SORS_COL_PAD,
: 246      0245  1      SORS_COL_THREE;
: 247      0246
: 248      0247  1  BIND
: 249      0248  1      COLL$_ADJ = SORS_COL_ADJ,      ! Invalid ADJ parameter
: 250      0249  1      COLL$_CMPLX = SORS_COL_CMPLX,   ! Collating sequence is too complex
: 251      0250  1      COLL$_CHAR = SORS_COL_CHAR,     ! Invalid character definition
: 252      0251  1      COLL$_PAD = SORS_COL_PAD,       ! Invalid pad character
: 253      0252  1      COLL$_THREE = SORS_COL_THREE;    ! Cannot define 3-byte collating values
: 254      0253
: 255      0254  1  LITERAL
: 256      0255  1      TRUE = 1,
: 257      0256  1      FALSE = 0;
: 258      0257
: 259      U 0258  1  %ELSE
: 260      UU 0259
: 261      UU 0260  1  LIBRARY 'S11V3SRC:SMCOM';
: 262      UU 0261
: 263      UU 0262  1  BIND
: 264      UU 0263  1      COLL$_ADJ = SORS_SPCADJ,      ! Invalid ADJ parameter
: 265      UU 0264  1      COLL$_CMPLX = SORS_WKAREA,   ! Collating sequence is too complex
: 266      UU 0265  1      COLL$_CHAR = SORS_SPCCHR,    ! Invalid character definition
: 267      UU 0266  1      COLL$_PAD = SORS_SPCPAD,     ! Invalid pad character
: 268      UU 0267  1      COLL$_THREE = SORS_SPCTHR;   ! Cannot define 3-byte collating values
: 269      U 0268
: 270      0269  1  %FI
: 271      0270
: 272      0271
: 273      0272  1  ! Define the successful status returned by these routines
: 274      0273  1  !
: 275      0274  1  %IF NOT %DECLARED(SS$_NORMAL) %THEN LITERAL SS$_NORMAL = 1; %FI
: 276      0275
: 277      0276  1  MACRO
: 278      M 0277  1      IF_ERROR_( X ) = %IF %BLISS( BLISS16 ) %THEN IF X NEQ SS$_NORMAL
: 279      0278  1      %ELSE IF NOT X %FI %;
: 280      0279  1  MACRO
: 281      M 0280  1      CS_SETUP(PARAM) =
: 282      M 0281  1      %IF %NULL(PARAM)
: 283      M 0282  1      %THEN
: 284      M 0283  1      EXTERNAL REGISTER CS = CS_K_REG: REF CS_BLOCK
: 285      M 0284  1      %ELSE
: 286      M 0285  1      GLOBAL REGISTER CS = CS_K_REG: REF CS_BLOCK;
: 287      M 0286  1      CS = .PARAM[1]
: 288      0287  1      %FI %;
: 289      0288
: 290      0289  1  %IF %DECLARED(%QUOTE ELIF ) %THEN UNDECLARE %QUOTE ELIF ; %FI
: 291      0290  1  %IF %DECLARED(%QUOTE BASE_) %THEN UNDECLARE %QUOTE BASE_; %FI
: 292      0291  1  MACRO
: 293      0292  1      ELIF=          ELSE IF %,
: 294      0293  1      BASE_=        0, 0, 0, 0 %;
: 295      0294
: 296      0295  1  MACRO
: 297      M 0296  1      MOVE_COLL_ALL_(X,Y) =

```



```
.. 298 M 0297 1 BEGIN
.. 299 M 0298 1 %IF %FIELDEXPAND(COLL_ALL,2) NEQ 0
.. 300 M 0299 1 %THEN
.. 301 M 0300 1     BBLOCK[X,COLL_ALL] = .BBLOCK[Y,COLL_ALL];
.. 302 M 0301 1 %ELSE
.. 303 M 0302 1     BBLOCK[X,COLL_C0] = .BBLOCK[Y,COLL_C0];
.. 304 M 0303 1     BBLOCK[X,COLL_C1] = .BBLOCK[Y,COLL_C1];
.. 305 M 0304 1 %FI
.. 306 M 0305 1 END %;
.. 307 M 0306 1 !MACRO
.. 308 M 0307 1     MOVE32_(X,Y) =
.. 309 M 0308 1     %IF %BLISS(BLISS32)
.. 310 M 0309 1     %THEN X = .Y
.. 311 M 0310 1     %ELSE ((X) = .(Y); (X+2)=.(Y+2)) %FI %;
.. 312 M 0311 1 LITERAL
.. 313 M 0312 1     K_CHARS = 256;           ! Number of 1-byte characters
.. 314 M 0313 1
.. 315 M 0314 1 MACRO
.. 316 M 0315 1     XBYTE = %EXPAND $BITS(8) %,
.. 317 M 0316 1     XWORD = %EXPAND $BITS(16) %,
.. 318 M 0317 1     XLONG = %EXPAND $BITS(32) %,
.. 319 M 0318 1     XDESC = $$SUB_BLOCK(2) %,
.. 320 M 0319 1     XADDR = $ADDRESS %;
.. 321 M 0320 1 $SHOW(FIELDS)
.. 322 M 0321 1
.. 323 M 0322 1 STRUCTURE
.. 324 M 0323 1     BBLOCK[O,P,S,E;BS=0] = [BS](BBLOCK+O)<P,S,E>;
```

```

: 326      0324  1  !
: 327      0325  1  !
: 328      0326  1  ! A char is an elementary data structure representing a single or double
: 329      0327  1  ! character.
: 330      0328  1  !
: 331      0329  1  $UNIT_FIELD
: 332      0330  1  CHAR_FIELDS =
: 333      0331  1  SET
: 334      L 0332  1  CHAR_LEN= [XWORD],
: %PRINT:      [0,0,16,0] (+%X'0')
: 335      L 0333  1  CHAR_CO= [XBYTE],
: %PRINT:      [2,0,8,0] (+%X'2')
: 336      L 0334  1  CHAR_C1= [XBYTE],
: %PRINT:      [3,0,8,0] (+%X'3')
: 337      0335  1 $OVERLAY(CHAR_CO)
: 338      L 0336  1 CHAR_C01= [XWORD],
: %PRINT:      [2,0,16,0] (+%X'2')
: 339      0337  1 $OVERLAY(0,0,0,0)
: 340      L 0338  1 CHAR_ALL= [XLONG]
: %PRINT:      [0,0,32,0] (+%X'0')
: 341      0339  1 TES;
: 342      0340  1 LITERAL CHAR_K_SIZE= $FIELD SET UNITS; ! Size in bytes
: 343      0341  1 MACRO CHAR_BLOCK= BBLOCK[CHAR_K_SIZE] FIELD(CHAR_FIELDS) %;

```

```

: 345      0342 1  |
: 346      0343 1  |
: 347      0344 1  |
: 348      0345 1  |
: 349      0346 1  |
: 350      0347 1  |
: 351      0348 1  |
: 352      0349 1  |
: 353      0350 1  |
: 354      0351 1  |
: 355      0352 1  |
: 356      L 0353 1  |
: %PRINT:  |
: 357      L 0354 1  |
: %PRINT:  |
: 358      0355 1  |
: 359      L 0356 1  |
: %PRINT:  |
: 360      0357 1  |
: 361      0358 1  |
: 362      0359 1  |

```

C O L L _ B L O C K

A coll is an elementary data structure representing a single, double or ignored collating value.

<0,0> ignored
<x,0> single collating value (x ne 0)
<x,y> double collating value (x,y ne 0)

\$UNIT FIELD
COLL_FIELDS =
SET
COLL_C0= [XWORD],
[0,0,16,0] (+%x'0')

COLL_C1= [XWORD],
[2,0,16,0] (+%x'2')

\$OVERLAY(0,0,0,0)
COLL_ALL= [XLONG]
[0,0,32,0] (+%x'0')

TES;
LITERAL COLL_K_SIZE= \$FIELD SET UNITS; ! Size in bytes
MACRO COLL_BLOCK= BBLOCK[COLL_K_SIZE] FIELD(COLL_FIELDS) %;

```

: 364      0360  1  !
: 365      0361  1  !
: 366      0362  1  ! This data structure holds pertinent information between calls.
: 367      0363  1  !
: 368      0364  1  $UNIT FIELD
: 369      0365  1  CS_FIELDS =
: 370      0366  1  SET
: 371      L 0367  1  CS_SIZE= [XWORD], ! Size of this block
: %PRINT: [0,0,16,0] (+%X'0')
: 372      L 0368  1  CS_CURR_SIZE= [XWORD], ! Current size of this block
: %PRINT: [2,0,16,0] (+%X'2')
: 373      L 0369  1  CS_COLL_MAX= [XWORD], ! Largest collating value
: %PRINT: [4,0,16,0] (+%X'4')
: 374      L 0370  1  CS_DCHAR= [XWORD], ! Number of double characters
: %PRINT: [6,0,16,0] (+%X'6')
: 375      L 0371  1  CS_TB= [XBYTE], ! Tie-break / Upper bits
: %PRINT: [8,0,8,0] (+%X'8')
: 376      L 0372  1  CS_PAD= [XBYTE], ! Pad character
: %PRINT: [9,0,8,0] (+%X'9')
: 377      L 0373  1  CS_REVERSE= [XBYTE], ! Reverse sense of tie-break CMPC
: %PRINT: [10,0,8,0] (+%X'A')
: 378      L 0374  1  CS_MODS= [$BIT], ! Modifications were made
: %PRINT: [11,0,1,0] (+%X'B')
: 379      L 0375  1  CS_IGN= [$BIT], ! There are ignored characters
: %PRINT: [11,1,1,0] (+%X'B')
: 380      L 0376  1  CS_DCOLL= [$BIT], ! There are double collating values
: %PRINT: [11,2,1,0] (+%X'B')
: 381      0377  1  $ALIGN(WORD)
: 382      0378  1  CS_PSTATIC= [$ADDRESS], ! Address of static base table
: 383      0379  1  CS_USTATIC= [$ADDRESS], ! Address of static upper table
: 384      L 0380  1  CS_UPPER= [$BYTES(K_CHARS)], ! Secondary table
: %PRINT: [12,0,0,0] (+%X'C')
: 385      L 0381  1  CS_PTAB= [$BYTES(K_CHARS*COLL_K_SIZE)], ! Table of single chars
: %PRINT: [268,0,0,0] (+%X'10C')
: 386      L 0382  1  CS_STAB= [$BYTES(0)] ! Table of double chars
: %PRINT: [1292,0,0,0] (+%X'50C')
: 387      0383  1  TES;
: 388      0384  1  LITERAL CS_K_SIZE= $FIELD SET UNITS; ! Size in bytes
: 389      0385  1  MACRO CS_BLOCK= BBLOCK[CS_R_SIZE] FIELD(CS_FIELDS,RES_FIELDS) %;
: 390      M 0386  1  MACRO CS_PTAB_(X)= COLL_K_SIZE*(X)+%FIELDEXPAND(CS_PTAB,0),0,
: 391      M 0387  1  %IF COLL_K_SIZE+%BPUNIT LEQ %BPVAL
: 392      0388  1  %THEN COLL_K_SIZE+%BPUNIT %ELSE 0 %FI,0 %;

```

R E S - B L O C K

```

394      0389 1  |
395      0390 1  |
396      0391 1  | This data structure holds the compressed form of the tables.
397      0392 1  | For Bliss-11, it is defined in a library so that the structure can be known
398      0393 1  | to the comparison routines, which are in a different overlay.
399      0394 1  |
400      U 0395 1  %IF NOT %BLISS(BLISS32) %THEN
401      U 0396 1  |
402      U 0397 1  LIBRARY 'S11V3SRC:SORCOLUTI';
403      U 0398 1  |
404      0399 1  %ELSE
405      0400 1  |
406      0401 1  $UNIT FIELD
407      0402 1  RES_FIELDS =
408      0403 1  SET
409      L 0404 1  RES_RTN=      [$ADDRESS],
%PRINT:  L 0405 1  RES_RTN_A=  [0,0,32,0] (+%X'0')
410      0406 1  RES_TB=      [$ADDRESS],
%PRINT:  L 0406 1  RES_TB=      [4,0,32,0] (+%X'4')
411      0407 1  RES_PAD=      [$BYTE],
%PRINT:  L 0407 1  RES_PAD=      [8,0,8,0] (+%X'8')
412      0408 1  RES_REVERSE= [$BYTE],
%PRINT:  L 0408 1  RES_REVERSE= [9,0,8,0] (+%X'9')
413      0409 1  RES_REVERSE= [10,0,8,0] (+%X'A')
414      0410 1  $ALIGN(WORD)
415      L 0410 1  RES_PTAB=      [$BYTES(K_CHARS)],
%PRINT:  L 0411 1  RES_UPPER=  [12,0,0,0] (+%X'C')
416      0411 1  RES_UPPER=  [268,0,0,0] (+%X'10C')
%PRINT:  L 0412 1  RES_STAB=      [$BYTES(0)]
417      0412 1  RES_STAB=  [524,0,0,0] (+%X'20C')
%PRINT:
418      0413 1  RES;
419      0414 1  LITERAL RES_K_SIZE= $FIELD_SET_UNITS;      ! Size in bytes
420      0415 1  |
421      0416 1  %FI
422      0417 1  |
423      0418 1  %IF RES_K_SIZE GTR CS_K_SIZE %THEN %ERROR('Something terrible happened') %FI
424      0419 1  |
425      0420 1  ! These values must be known to the macro routine
426      0421 1  |
427      0422 1  GLOBAL LITERAL
428      0423 1  RES$RTN=      %FIELDEXPAND(RES_RTN,0),
429      0424 1  RES$TB=      %FIELDEXPAND(RES_TB,0),
430      0425 1  RES$REVERSE= %FIELDEXPAND(RES_REVERSE,0),
431      0426 1  RES$PAD=      %FIELDEXPAND(RES_PAD,0),
432      0427 1  RES$PTAB=     %FIELDEXPAND(RES_PTAB,0),
433      0428 1  RES$UPPER=    %FIELDEXPAND(RES_UPPER,0),
434      0429 1  RES$STAB=    %FIELDEXPAND(RES_STAB,0),
435      0430 1  TBS$NOTB =    %B'0100',      ! Don't do tie-breaking (the Z-bit)
436      0431 1  TBS$NOUPPER =   %B'0010',      ! Don't use upper table (the V-bit)
437      0432 1  TBS$REVERSE =  %B'0001',      ! Reverse tie-break CMPC (the C-bit)
438      0433 1  |

```

```

: 440      0434 1 |           S T - B L O C K
: 441      0435 1 |
: 442      0436 1 | A secondary table entry consists of:
: 443      0437 1 |     An indication whether the input character is one or two bytes.
: 444      0438 1 |     The one or two byte input character.
: 445      0439 1 |     The collating value.
: 446      0440 1 |     The offset to the next secondary table entry.
: 447      0441 1 |
: 448      0442 1 | $UNIT_FIELD
: 449      0443 1 |     ST_FIELDS =
: 450      0444 1 |     SET
: 451      L 0445 1 |     ST_CHAR=          [XWORD],
: 452      L 0446 1 |     ST_COLL=         [0,0,16,0] (+%X'0')
: 453      0447 1 | $OVERLAY(ST_CHAR)
: 454      L 0448 1 |     ST_CHAR_0=       [XBYTE],
: 455      L 0449 1 |     ST_CHAR_1=       [0,0,8,0] (+%X'0')
: 456      0450 1 | $CONTINUE
: 457      0451 1 |     TES;
: 458      0452 1 | LITERAL ST_K_SIZE=   $FIELD SET UNITS;      ! Size in bytes
: 459      0453 1 | MACRO   ST_BLOCK=    BBLOCK[ST_R_SIZE] FIELD(ST_FIELDS) %;

```

```

: 461 0454 1 :
: 462 0455 1 : When we are inserting another collating value, but have no available
: 463 0456 1 : single-byte collating values, we can:
: 464 0457 1 :   Find two adjacent one-byte collating values (x and x+1) that:
: 465 0458 1 :     are not used as the first byte of any two-byte collating values,
: 466 0459 1 :     and are not used as the second byte of any two-byte collating values
: 467 0460 1 :     for which the first byte is used as a one-byte collating value.
: 468 0461 1 :   Change the characters that collate to x and x+1 to collate to the two-byte
: 469 0462 1 :     collating values <x,0> and <x,1>, respectively.
: 470 0463 1 :   This frees the single-byte collating value x+1.
: 471 0464 1 :
: 472 0465 1 : Or (preferably) we can:
: 473 0466 1 :
: 474 0467 1 :   Find a collating value (x) that:
: 475 0468 1 :     is used only as the first byte of two-byte collating values
: 476 0469 1 :     for which not all 256 different second byte values are used,
: 477 0470 1 :     it has an adjacent value y (either x-1 or x+1) such that:
: 478 0471 1 :       it is not used as the first byte of any two-byte collating values,
: 479 0472 1 :       and is not used as the second byte of any two-byte collating values
: 480 0473 1 :       for which the first byte is used as a one-byte collating value.
: 481 0474 1 :   Add another second-byte collating value (z) to the two-byte collating
: 482 0475 1 :     values that have x as their first-byte collating value, such that z is
: 483 0476 1 :     less than (y=x-1), or greater than (y=x+1) all the other second-byte
: 484 0477 1 :     collating values.
: 485 0478 1 :   Change the characters that collate to y to collate to <x,z>.

```

```

487 0479 1 GLOBAL ROUTINE COLL$INIT(
488 0480 1     COLL_SEQ:      REF VECTOR[2]           ! Collating sequence
489 0481 1     ) =
490 0482 1     ++
491 0483 1
492 0484 1     FUNCTIONAL DESCRIPTION:
493 0485 1
494 0486 1         Initialize a collating sequence description.
495 0487 1         It is initialized to all ignored characters.
496 0488 1
497 0489 1     FORMAL PARAMETERS:
498 0490 1
499 0491 1         COLL_SEQ      a two-longword array specifying the length/address
500 0492 1         of storage to use for the collating sequence.
501 0493 1
502 0494 1     IMPLICIT INPUTS:
503 0495 1
504 0496 1         NONE
505 0497 1
506 0498 1     IMPLICIT OUTPUTS:
507 0499 1
508 0500 1         The memory specified by COLL_SEQ is initialized.
509 0501 1
510 0502 1     ROUTINE VALUE:
511 0503 1
512 0504 1         Status code
513 0505 1
514 0506 1     SIDE EFFECTS:
515 0507 1
516 0508 1         NONE
517 0509 1
518 0510 1     --
519 0511 2     BEGIN
520 0512 2
521 0513 2     CS_SETUP(COLL_SEQ);
522 0514 2
523 0515 2     IF .COLL_SEQ[0] LSSU CS_K SIZE THEN RETURN COLL$_CMLPX;
524 0516 2     CH$FILL(0, CS_K SIZE, CS[BASE ]);
525 0517 2     CS[CS_SIZE] = -MINU(.COLL_SEQ[0], 1^%FIELDEXPAND(CS_SIZE,2)-1);
526 0518 2     CS[CS_CURR_SIZE] = CS_K SIZE;
527 0519 2     CS[CS_TB] = TB$NOTB OR TB$NOUPPER;
528 0520 2
529 0521 2     RETURN S$$_NORMAL;
530 0522 1     END;

```

.TITLE COLLSUTILITIES
.IDENT \V04-000\

```

RES$RTN== 0
RES$TB== 8
RES$REVERSE== 10
RES$PAD== 9
RES$PTAB== 12
RES$UPPER== 268
RES$STAB== 524
TB$NOTB== 4

```



```

TB$NOUPPER==                2
TB$REVERSE==                1
      .EXTRN SOR$_COL_ADJ, SOR$_COL_CMPLX
      .EXTRN SOR$_COL_CHAR, SOR$_COL_PAD
      .EXTRN SOR$_COL_THREE

      .PSECT SOR$RO_CODE,NOWRT, SHR, PIC,2

      .ENTRY COLLS$INIT, Save R2,R3,R4,R5,R6,R10
047C 00000      MOVL     COLL_SEQ, R6      : 0479
56      04 AC D0 00002      : 0513
5A      04 A6 D0 00006      :
0000050C 8F 66 D1 0000A      :
      08 1E 00011      : 0515
50 00000000G 8F D0 00013      :
      04 0001A      :
050C 8F 00 2C 0001B 1$:      MOVC5  #0, (SP), #0, #1292, (CS)      : 0516
      6A 00022      :
50      66 D0 00023      :
0000FFFF 8F 50 D1 00026      MOVL     (R6), R0      : 0517
      05 1B 0002D      :
50      8F 3C 0002F      BLEQU  2$,
      6A 50 B0 00034 2$:      MOVZWL #65535, R0
02      AA 050C 8F B0 00037      MOVW   R0, (CS)
08      AA 06 90 0003D      MOVW   #1292, 2(CS)
50      01 D0 00041      MOVW   #6, 8(CS)
      04 00044      MOVL     #1, R0
      RET
  
```

; Routine Size: 69 bytes, Routine Base: SOR\$RO_CODE + 0000

```

532 0523 1 GLOBAL ROUTINE COLL$BASE(
533 0524 1     COLL_SEQ:      REF VECTOR[2],           ! Collating sequence
534 0525 1     BASE_SEQ:      REF VECTOR[K_CHARS, BYTE] ! Base sequence
535 0526 1     ;
536 0527 1     ) =
537 0528 1 ++
538 0529 1
539 0530 1 FUNCTIONAL DESCRIPTION:
540 0531 1
541 0532 1     Specify the base collating sequence.
542 0533 1
543 0534 1 FORMAL PARAMETERS:
544 0535 1
545 0536 1     COLL_SEQ      a two-longword array specifying the length/address
546 0537 1                of storage to use for the collating sequence.
547 0538 1
548 0539 1     BASE_SEQ      a 256-byte array giving the (single byte) collating
549 0540 1                value for each character.
550 0541 1
551 0542 1 IMPLICIT INPUTS:
552 0543 1
553 0544 1     INIT must have already been called.
554 0545 1
555 0546 1 IMPLICIT OUTPUTS:
556 0547 1
557 0548 1     NONE
558 0549 1
559 0550 1 ROUTINE VALUE:
560 0551 1
561 0552 1     Status code
562 0553 1
563 0554 1 SIDE EFFECTS:
564 0555 1
565 0556 1     NONE
566 0557 1
567 0558 1 --
568 0559 2 BEGIN
569 0560 2 LOCAL
570 0561 2     BS:      REF VECTOR[K_CHARS, BYTE];
571 0562 2 BUILTIN
572 0563 2     NULLPARAMETER;
573 0564 2
574 0565 2     CS_SETUP(COLL_SEQ);
575 0566 2
576 0567 2     BS = BASE_SEQ[0];
577 0568 2     DECR I FROM K_CHARS-1 TO 0 DO (CS[CS_PTAB_(.I)]) = .BS[.I] + 1;
578 0569 2     CS[CS_COLL_MAX] = K_CHARS;
579 0570 2 !     IF NOT NUL[PARAMETER(3)] THEN CS[CS_PSTATIC] = BASE_SEQ[0];
580 0571 2
581 0572 2     RETURN SSS_NORMAL;
582 0573 1     END;

```

0400 0000

.ENTRY COLL\$BASE, Save R10

: 0523

COLLSUTILITIES
V04-000

E 12
16-Sep-1984 01:06:02 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 13:10:40 [SORT32.SRC]SORCOLUTI.B32;1

Page 17
(10)

50	04	AC	7D	00002	MOVQ	COLL_SEQ, R0	:	0565	
5A	04	A0	D0	00006	MOVL	4(R0), CS	:		
50	FF	8F	9A	0000A	MOVZBL	#255, I	:	0568	
010C	CA40	6041	9A	0000E	MOVZBL	(I)[BS], 268(CS)[I]	:		
		010C	CA40	D6	00015	INCL	268(CS)[I]	:	
	F1		50	F4	0001A	SOBGEQ	I, 1\$:	0569
04	AA	0100	8F	B0	0001D	MOVW	#256, 4(CS)	:	0572
	50		01	D0	00023	MOVL	#1, R0	:	0573
			04	00026	RET		:		

; Routine Size: 39 bytes, Routine Base: SOR\$RO_CODE + 0045

```

584 0574 1 GLOBAL ROUTINE COLL$UPPER(
585 0575 1     COLL_SEQ:      REF VECTOR[2],           ! Collating sequence
586 0576 1     UPPER_SEQ:   REF VECTOR[K_CHARS, BYTE] ! Secondary sequence
587 0577 1     ;
588 0578 1     { STATIC
589 0579 1     ++
590 0580 1
591 0581 1     FUNCTIONAL DESCRIPTION:
592 0582 1
593 0583 1     Specify the secondary collating sequence.
594 0584 1     If two strings compare equal using the sequence specified with BASE,
595 0585 1     SEQUENCE, MODIFY and IGNORE, the collating sequence specified by this
596 0586 1     routine is then used.
597 0587 1
598 0588 1     FORMAL PARAMETERS:
599 0589 1
600 0590 1     COLL_SEQ      a two-longword array specifying the length/address
601 0591 1     of storage to use for the collating sequence.
602 0592 1
603 0593 1     UPPER_SEQ     a 256-byte array giving the (single byte) collating
604 0594 1     value for each character.
605 0595 1
606 0596 1     IMPLICIT INPUTS:
607 0597 1
608 0598 1     INIT must have already been called.
609 0599 1
610 0600 1     IMPLICIT OUTPUTS:
611 0601 1
612 0602 1     NONE
613 0603 1
614 0604 1     ROUTINE VALUE:
615 0605 1
616 0606 1     Status code
617 0607 1
618 0608 1     SIDE EFFECTS:
619 0609 1
620 0610 1     NONE
621 0611 1
622 0612 1     --
623 0613 2     BEGIN
624 0614 2     LOCAL
625 0615 2     BS:      REF VECTOR[K_CHARS, BYTE],
626 0616 2     X;
627 0617 2     BUILTIN
628 0618 2     NULLPARAMETER;
629 0619 2
630 0620 2     CS_SETUP(COLL_SEQ);
631 0621 2
632 0622 2     X = UPPER_SEQ[0];
633 0623 2     IF .X NEQ 0 THEN X = K_CHARS;
634 0624 2     CH$COPY(.X, UPPER_SEQ[0], 0, K_CHARS, CS[CS_UPPER]);
635 0625 2
636 0626 2     IF NOT NULLPARAMETER(3) THEN CS[CS_USTATIC] = UPPER_SEQ[0];
637 0627 2     CS[CS_TB] = .CS[CS_TB] AND NOT TB$NOUPPER;
638 0628 2
639 0629 2     RETURN SS$_NORMAL;
640 0630 1     END;

```

					043C 00000	.ENTRY	COLL\$UPPER, Save R2,R3,R4,R5,R10	:	0574
		50	04	AC	D0 00002	MOVL	COLL_SEQ, RC	:	0620
		5A	04	A0	D0 00006	MOVL	4(R0), C\$:	
		50	08	AC	D0 0000A	MOVL	UPPER_SEQ, X	:	0622
				05	13 0000E	BEQL	1\$:	0623
0100	8F			50	8F 3C 00010	MOVZWL	#256, X	:	
		00	08	BC	50 2C 00015 1\$:	MOVCS	X, @UPPER_SEQ, #0, #256, 12(CS)	:	0624
					0C	AA	0001D	:	
			08	AA	02 8A 0001F	BICB2	#2, 8(CS)	:	0627
				50	01 D0 00023	MOVL	#1, R0	:	0629
					04 00026	RET		:	0630

; Routine Size: 39 bytes, Routine Base: SOR\$RO_CODE + 006C

```

: 642 0631 1 GLOBAL ROUTINE COLL$NEXT(
: 643 0632 1     COLL_SEQ:      REF VECTOR[2],      ! Collating sequence
: 644 0633 1     CHART:      REF CHAR_BLOCK      ! Character being defined
: 645 0634 1     ) =
: 646 0635 1     +-+
: 647 0636 1
: 648 0637 1     FUNCTIONAL DESCRIPTION:
: 649 0638 1
: 650 0639 1         Define a character to collate greater than any currently defined
: 651 0640 1         character.
: 652 0641 1
: 653 0642 1     FORMAL PARAMETERS:
: 654 0643 1
: 655 0644 1         COLL_SEQ      a two-longword array specifying the length/address
: 656 0645 1         of storage to use for the collating sequence.
: 657 0646 1
: 658 0647 1         CHAR1        a character.
: 659 0648 1
: 660 0649 1     IMPLICIT INPUTS:
: 661 0650 1
: 662 0651 1         INIT must have already been called.
: 663 0652 1
: 664 0653 1     IMPLICIT OUTPUTS:
: 665 0654 1
: 666 0655 1         NONE
: 667 0656 1
: 668 0657 1     ROUTINE VALUE:
: 669 0658 1
: 670 0659 1         Status code
: 671 0660 1
: 672 0661 1     SIDE EFFECTS:
: 673 0662 1
: 674 0663 1         NONE
: 675 0664 1
: 676 0665 1     --
: 677 0666 2     BEGIN
: 678 0667 2     LOCAL
: 679 0668 2         COLL:  COLL_BLOCK;
: 680 0669 2
: 681 0670 2     CS_SETUP(COLL_SEQ);
: 682 0671 2
: 683 0672 2     CS[CS_MODS] = TRUE;
: 684 0673 2
: 685 0674 2     CS[CS_COLL_MAX] = .CS[CS_COLL_MAX] + 1;
: 686 0675 2     COLL[COLL_?0] = .CS[CS_COLL_MAX];
: 687 0676 2     COLL[COLL_?1] = 0;
: 688 0677 2     RETURN GIVE_COLL( CHAR1[CHAR_ALL], COLL[COLL_ALL] );
: 689 0678 1     END;

```

```

                                OFFC 00000      .ENTRY COLL$NEXT, Save R2,R3,R4,R5,R6,R7,R8,R9,- ; 0631
                                50      04      AC 7D 00002      MOVQ  COLL_SEQ, R0 ; 0670
                                5A      04      A0 D0 00006      MOVL  4(R0), CS ;

```

COLLSUTILITIES
V04-000

I 12
16-Sep-1984 01:06:02
14-Sep-1984 13:10:40

VAX-11 Bliss-32 V4.0-742
[SORT32.SRC]SORCOLUTI.B32;1

Page 21
(12)

0B	AA	01	88	0000A
		04	AA	B6 0000E
	7E	04	AA	3C 00011
	52		6E	9E 00015
		0000V	30	00018
			04	0001B

BISB2	#1, 11(CS)
INCW	4(CS)
MOVZWL	4(CS), COLL
MOVAB	COLL, R2
BSBW	GIVE_COLL
RET	

: 0672
: 0674
: 0675
: 0677
: 0678

: Routine Size: 28 bytes, Routine Base: SOR\$RO_CODE + 0093

```

691 C 0679 1 %(
692 C 0680 1 GLOBAL ROUTINE COLLSOTHERS(
693 C 0681 1     COLL_SEQ: REF VECTOR[2]      ' Collating sequence
694 C 0682 1     ) =
695 C 0683 1 --++
696 C 0684 1
697 C 0685 1 FUNCTIONAL DESCRIPTION:
698 C 0686 1
699 C 0687 1     Define all currently ignored (undefined) characters to collate larger
700 C 0688 1     than all the non-ignored (defined) characters, in order of the
701 C 0689 1     character codes.
702 C 0690 1
703 C 0691 1 FORMAL PARAMETERS:
704 C 0692 1
705 C 0693 1     COLL_SEQ      a two-longword array specifying the length/address
706 C 0694 1     of storage to use for the collating sequence.
707 C 0695 1
708 C 0696 1 IMPLICIT INPUTS:
709 C 0697 1
710 C 0698 1     INIT must have already been called.
711 C 0699 1
712 C 0700 1 IMPLICIT OUTPUTS:
713 C 0701 1
714 C 0702 1     NONE
715 C 0703 1
716 C 0704 1 ROUTINE VALUE:
717 C 0705 1
718 C 0706 1     Status code
719 C 0707 1
720 C 0708 1 SIDE EFFECTS:
721 C 0709 1
722 C 0710 1     NONE
723 C 0711 1
724 C 0712 1 --
725 C 0713 1 BEGIN
726 C 0714 1 LOCAL
727 C 0715 1     CHAR: CHAR_BLOCK,
728 C 0716 1     P: REF COLL_BLOCK,
729 C 0717 1     S:
730 C 0718 1
731 C 0719 1 CS_SETUP(COLL_SEQ);
732 C 0720 1
733 C 0721 1 CSECS_MODS] = TRUE;
734 C 0722 1
735 C 0723 1 CHAR[CHAR_LEN] = 1;
736 C 0724 1
737 C 0725 1 P = CSECS_PTAB];
738 C 0726 1 INCR I FROM 0 TO K_CHARS-1 DO
739 C 0727 1     BEGIN
740 C 0728 1     IF
741 C 0729 1         %IF %FIELDEXPAND(COLL_ALL,2) NEQ 0
742 C 0730 1         %THEN .P[COLL_ALL] EQL 0
743 C 0731 1         %ELSE .P[COLL_CO] EQL 0 AND .P[COLL_C1] EQL 0
744 C 0732 1         %FI
745 C 0733 1     THEN
746 C 0734 1     BEGIN
747 C 0735 1     CHAR[CHAR_CO] = .I;

```


COLLSUTILITIES
V04-000

K 12
16-Sep-1984 01:06:02
14-Sep-1984 13:10:40

VAX-11 Bliss-32 V4.0-742
[SORT32.SRC]SORCOLUTI.B32;1

Page 23
(13)

```
: 748      C 0736 1          S = COLLSNEXT(COLL_SEQ[0], CHAR(BASE_));  
: 749      C 0737 1          IF ERROR_(.S) THEN RETURN .S;  
: 750      C 0738 1          END;  
: 751      C 0739 1          P = .P + COLL_K_SIZE;  
: 752      C 0740 1          END;  
: 753      C 0741 1  
: 754      C 0742 1          RETURN SSS_NORMAL;  
: 755      C 0743 1          END;  
: 756      C 0744 1 )%
```

```

758      0745 1 MACRO
759      M 0746 1   FOR_ALL_COLL(X) =
760      M 0747 1   BEGIN
761      M 0748 1     LOCAL X: REF COLL_BLOCK;
762      M 0749 1     LOCAL STEP;
763      M 0750 1     X = CS[CS_PTAB];
764      M 0751 1     STEP = COLL_K_SIZE;
765      M 0752 1     DECR FIRST FROM 1 TO 0 DO
766      M 0753 1       BEGIN
767      M 0754 1         DECR I FROM (IF .FIRST THEN K_CHARS ELSE .CS[CS_DCHAR])-1 TO 0 DO
768      M 0755 1           BEGIN
769      M 0756 1             %
770      M 0757 1       END_ALL_COLL(X) =
771      M 0758 1         X = .X + .STEP;
772      M 0759 1         END;
773      M 0760 1         STEP = ST_K_SIZE;
774      M 0761 1         X = .X + %FIELDEXPAND(ST_COLL,0)
775      M 0762 1           - K_CHARS * COLL_K_SIZE
776      M 0763 1           - %FIELDEXPAND(CS_PTAB,0)
777      M 0764 1           + %FIELDEXPAND(CS_STAB,0);
778      M 0765 1         END;
779      M 0766 1       END %
780      M 0767 1   FOR_ALL_DCHARS(X) =
781      M 0768 1   BEGIN
782      M 0769 1     LOCAL X: REF ST_BLOCK;
783      M 0770 1     X = CS[CS_STAB];
784      M 0771 1     DECR I FROM .CS[CS_DCHAR]-1 TO 0 DO
785      M 0772 1       BEGIN
786      M 0773 1         %
787      M 0774 1       END_ALL_DCHARS(X) =
788      M 0775 1         X = .X + ST_K_SIZE;
789      M 0776 1         END;
790      M 0777 1       END %
791      M 0778 1   FOR_ALL_SCHARS(X) =
792      M 0779 1   BEGIN
793      M 0780 1     LOCAL X: REF COLL_BLOCK;
794      M 0781 1     X = CS[CS_PTAB];
795      M 0782 1     DECR I FROM K_CHARS-1 TO 0 DO
796      M 0783 1       BEGIN
797      M 0784 1         %
798      M 0785 1       END_ALL_SCHARS(X) =
799      M 0786 1         X = .X + COLL_K_SIZE;
800      M 0787 1         END;
801      M 0788 1       END %;

```

```

: 803 0789 1 GLOBAL ROUTINE COLLSMODIFY(
: 804 0790 1     COLL_SEQ:      REF VECTOR[2],      ! The collating sequence
: 805 0791 1     CHART:        REF CHAR_BLOCK,     ! Character being defined
: 806 0792 1     CHAR2:        REF CHAR_BLOCK,     ! In terms of this character
: 807 0793 1     ADJ           ! Adjustment
: 808 0794 1     ) =
: 809 0795 1 !++
: 810 0796 1
: 811 0797 1 FUNCTIONAL DESCRIPTION:
: 812 0798 1
: 813 0799 1     Modify the collating sequence.
: 814 0800 1     Based on the value of ADJ, define CHAR1 to collate just less than (-1),
: 815 0801 1     equal to (0), or just greater than (+1) CHAR2.
: 816 0802 1
: 817 0803 1 FORMAL PARAMETERS:
: 818 0804 1
: 819 0805 1     COLL_SEQ      a two-longword array specifying the length/address
: 820 0806 1     of storage to use for the collating sequence.
: 821 0807 1
: 822 0808 1     CHAR1         the character being defined.
: 823 0809 1
: 824 0810 1     CHAR2         the character used to define CHAR1.
: 825 0811 1
: 826 0812 1     ADJ           adjustment; either -1, 0 or +1.
: 827 0813 1
: 828 0814 1 IMPLICIT INPUTS:
: 829 0815 1
: 830 0816 1     INIT must have already been called.
: 831 0817 1
: 832 0818 1 IMPLICIT OUTPUTS:
: 833 0819 1
: 834 0820 1     NONE
: 835 0821 1
: 836 0822 1 ROUTINE VALUE:
: 837 0823 1
: 838 0824 1     Status code
: 839 0825 1
: 840 0826 1 SIDE EFFECTS:
: 841 0827 1
: 842 0828 1     NONE
: 843 0829 1
: 844 0830 1 !--
: 845 0831 2 BEGIN
: 846 0832 2 LOCAL
: 847 0833 2     COLL:  COLL_BLOCK,
: 848 0834 2     LADJ,      ! Local copy of adj
: 849 0835 2     S;        ! Status value
: 850 0836 2
: 851 0837 2     CS_SETUP(COLL_SEQ);
: 852 0838 2
: 853 0839 2     CS[CS_MODS] = TRUE;
: 854 0840 2
: 855 0841 2 !+
: 856 0842 2 Define CHAR1 to collate:
: 857 0843 2     (ADJ = -1) less than, (ADJ = 0) equal to, or (ADJ = +1) greater than
: 858 0844 2     the character CHAR2.
: 859 0845 2 !-
```

```

860 0846 2
861 0847 2
862 0848 2
863 0849 2
864 0850 2
865 0851 2
866 0852 2
867 0853 2
868 0854 2
869 0855 2
870 0856 2
871 0857 2
872 0858 2
873 0859 2
874 0860 2
875 0861 2
876 0862 2
877 0863 2
878 0864 2
879 0865 2
880 0866 2
881 0867 2
882 0868 2
883 0869 2
884 0870 2
885 0871 2
886 0872 2
887 0873 2
888 0874 2
889 0875 2
890 0876 2
891 0877 2
892 0878 2
893 0879 2
894 0880 2
895 0881 2
896 0882 2
897 0883 2
898 0884 2
899 0885 2
900 0886 2
901 0887 2
902 0888 2
903 0889 2
904 0890 2
905 0891 2
906 0892 2
907 0893 2
908 0894 2
909 0895 2
910 0896 2
911 0897 2
912 0898 2
913 0899 2
914 0900 2
915 0901 2
916 0902 2

! Check that ADJ = +1, 0, or -1
LADJ = .ADJ;
SELECTONE .LADJ OF SET [-1,0,+1]:0; [OTHERWISE]:RETURN COLLS_ADJ; TES;

! Set COLL to the current collating value of CHAR2
S = COLL_VALUE(CHAR2[CHAR ALL], COLL[COLL_ALL]);
IF_ERROR_( .S ) THEN RETURN .S;

! If COLL indicates an ignored character
Then
  Check that ADJ >= 0
  If ADJ > 0 then set COLL to the lowest character, and ADJ to -1
IF .COLL[COLL_C0] EQL 0
THEN
  BEGIN
  IF .LADJ LSS 0 THEN RETURN COLLS_ADJ;
  IF .LADJ GTR 0
  THEN
    BEGIN
    COLL[COLL_C0] = 1;           ! The smallest collating value
    COLL[COLL_C1] = 0;         ! No second character
    LADJ = -1;                 ! Create something even smaller
    END;
  END;

! Give CHAR1 the collating value COLL
S = GIVE_COLL( CHAR1[CHAR ALL], COLL[COLL_ALL] );
IF_ERROR_( .S ) THEN RETURN .S;

! If ADJ = 0 then we are done
IF .LADJ EQL 0 THEN RETURN SSS_NORMAL;

! Set COLL to the current collating value of CHAR1
S = COLL_VALUE(CHAR1[CHAR ALL], COLL[COLL_ALL]);
IF_ERROR_( .S ) THEN RETURN .S;

! Bump the collating values of everything greater than or equal to the
! new collating value we want to give CHAR1.
IF (S = .COLL[COLL_C1]) EQL 0 THEN S = .COLL[COLL_C0];
IF .LADJ GTR 0 THEN S = .S + 1;
S = DO_BUMP(.S);
IF_ERROR_( .S ) THEN RETURN .S;
CS[CS_COLL_MAX] = .CS[CS_COLL_MAX] + 1;
```

917 0903 2
918 0904 2
919 0905 2
920 0906 2
921 0907 2
922 0908 2
923 0909 2
924 0910 2
925 0911 2
926 0912 2
927 0913 2
928 0914 2
929 0915 2
930 0916 2
931 0917 2
932 0918 1

```
! Set COLL to the current collating value of CHAR1  
S = COLL_VALUE(CHAR1[CHAR_ALL], COLL[COLL_ALL]);  
IF_ERROR_( .S ) THEN RETURN .S;  
  
! Adjust the collating value COLL, and assign it to CHAR1  
IF .COLL[COLL_C1] NEQ 0  
THEN COLL[COLL_C1] = .COLL[COLL_C1] + .LADJ  
ELSE COLL[COLL_C0] = .COLL[COLL_C0] + .LADJ;  
RETURN GIVE_COLL( CHAR1[CHAR_ALL], COLL[COLL_ALL] );  
  
END;
```

	OFFC	00000	.ENTRY	COLL\$MODIFY, Save R2,R3,R4,R5,R6,R7,R8,R9,-	
54	0000V	CF 9E 00002	MOVAB	R10,R11	0789
5E		04 C2 00007	SUBL2	COLL_VALUE, R4	
50	04	AC D0 0000A	MOVL	#4, SP	
5A	04	A0 D0 0000E	MOVL	COLL_SEQ, R0	0837
OB AA		01 88 00012	MOVL	4(R0), CS	
53	10	AC D0 00016	BISB2	#1, 11(CS)	0839
FFFFFFFF 8F		53 D1 0001A	MOVL	ADJ, LADJ	0849
		19 19 00021	CMPL	LADJ, #-1	0850
01		53 D1 00023	BLSS	1\$	
		14 14 00026	CMPL	LADJ, #1	
52		6E 9E 00028	BGTR	1\$	
51	0C	AC D0 0002B	MOVAB	COLL, R2	0855
		64 16 0002F	MOVL	CHAR2, R1	
78		50 E9 00031	JSB	COLL_VALUE	
		6E B5 00034	BLBC	S, 9\$	0856
		14 12 00036	TSTW	COLL	0864
		53 D5 00038	BNEQ	3\$	
		08 18 0003A	TSTL	LADJ	0867
50	00000000G	8F D0 0003C 1\$:	BGEQ	2\$	
		04 00043	MOVL	#COLL\$_ADJ, R0	
		06 15 00044 2\$:	RET		
6E		01 D0 00046	BLEQ	3\$	0868
53		01 CE 00049	MOVL	#1, COLL	0871
52		6E 9E 0004C 3\$:	MNEGL	#1, LADJ	0873
51	08	AC D0 0004F	MOVAB	COLL, R2	0880
		0000V 30 00053	MOVL	CHAR1, R1	
53		50 E9 00056	BSBW	GIVE_COLL	
		53 D5 00059	BLBC	S, 9\$	0881
		04 12 0005B	TSTL	LADJ	0886
50		01 D0 0005D	BNEQ	4\$	
		04 00060	MOVL	#1, R0	
52		6E 9E 00061 4\$:	RET		
51	08	AC D0 00064	MOVAB	COLL, R2	0891
		64 16 00068	MOVL	CHAR1, R1	
			JSB	COLL_VALUE	

3F		50	E9	0006A	BLBC	S, 9\$:	0892
50	02	AE	3C	0006D	MOVZWL	COLL+2, S	:	0898
		03	12	00071	BNEQ	5\$:	
50		6E	3C	00073	MOVZWL	COLL, S	:	
		53	D5	00076	TSTL	LADJ	:	0899
		02	15	00078	BLEQ	6\$:	
		50	D6	0007A	INCL	S	:	
51		50	D0	0007C	MOVL	S, R1	:	0900
		0000V	30	0007F	BSBW	DO_BUMP	:	
27		50	E9	00082	BLBC	S, -9\$:	0901
	04	AA	B6	00085	INCW	4(CS)	:	0902
52		6E	9E	00088	MOVAB	COLL, R2	:	0907
51	08	AC	D0	0008B	MOVL	CHAR1, R1	:	
		64	16	0008F	JSB	COLL VALUE	:	
18		50	E9	00091	BLBC	S, 9\$:	0908
	02	AE	B5	00094	TSTW	COLL+2	:	0913
		06	13	00097	BEQL	7\$:	
02	AE	53	A0	00099	ADDW2	LADJ, COLL+2	:	0914
		03	11	0009D	BRB	8\$:	
6E		53	A0	0009F	ADDW2	LADJ, COLL	:	0915
52		6E	9E	000A2	MOVAB	COLL, R2	:	0916
51	08	AC	D0	000A5	MOVL	CHAR1, R1	:	
		0000V	30	000A9	BSBW	GIVE_COLL	:	
		04	000AC	9\$:	RET		:	0918

; Routine Size: 173 bytes, Routine Base: SOR\$RO_CODE + 00AF

```

: 934 0919 1 GLOBAL ROUTINE COLLS$FOLD(
: 935 0920 1     COLL_SEQ: REF VECTOR[2],      ! The collating sequence
: 936 0921 1     BV_L: REF BITVECTOR[K_CHARS], ! Lower case letters
: 937 0922 1     CC:                      ! Lower XOR .CC = Upper
: 938 0923 1     ) =
: 939 0924 1     ++
: 940 0925 1
: 941 0926 1     FUNCTIONAL DESCRIPTION:
: 942 0927 1
: 943 0928 1         Fold characters (this is a shorthand for several calls to MODIFY).
: 944 0929 1         For each character (X) in the set of characters specified by BV_L,
: 945 0930 1         define it to collate equal to its change-case form (X xor CC).
: 946 0931 1         Also, for all double characters for which neither character is in BV_L,
: 947 0932 1         define the change-case forms to equal it.
: 948 0933 1
: 949 0934 1     FORMAL PARAMETERS:
: 950 0935 1
: 951 0936 1         COLL_SEQ      a two-longword array specifying the length/address
: 952 0937 1         of storage to use for the collating sequence.
: 953 0938 1
: 954 0939 1         BV_L         the address of a 256-bit bitvector.
: 955 0940 1
: 956 0941 1         CC          change-case value to be xor-ed to give the other case.
: 957 0942 1
: 958 0943 1     IMPLICIT INPUTS:
: 959 0944 1
: 960 0945 1         INIT must have already been called.
: 961 0946 1
: 962 0947 1     IMPLICIT OUTPUTS:
: 963 0948 1
: 964 0949 1         NONE
: 965 0950 1
: 966 0951 1     ROUTINE VALUE:
: 967 0952 1
: 968 0953 1         Status code
: 969 0954 1
: 970 0955 1     SIDE EFFECTS:
: 971 0956 1
: 972 0957 1         NONE
: 973 0958 1
: 974 0959 1     --
: 975 0960 2     BEGIN
: 976 0961 2     LOCAL
: 977 0962 2         COLL: COLL_BLOCK,
: 978 0963 2         CHAR: CHAR_BLOCK,
: 979 0964 2         S;                               ! Status value
: 980 0965 2
: 981 0966 2     CS_SETUP(COLL_SEQ);
: 982 0967 2
: 983 0968 2     ! Define lower case letters to equal their upper case equivalents
: 984 0969 2     !
: 985 0970 2     CHAR[CHAR_LEN] = 1;
: 986 0971 2     DECR I FROM K_CHARS-1 TO 0 DO
: 987 0972 2         IF .BV_L[I]
: 988 0973 2         THEN
: 989 0974 2             BEGIN
: 990 0975 2                 CHAR[CHAR_CO] = .I;

```

```

: 991 0976 3
: 992 0977 3
: 993 0978 2
: 994 0979 2
: 995 0980 2
: 996 0981 2
: 997 0982 2
: 998 0983 2
: 999 0984 4
1000 0985 4
1001 0986 4
1002 0987 4
1003 0988 5
1004 0989 5
1005 0990 5
1006 0991 5
1007 0992 6
1008 0993 6
1009 0994 6
1010 0995 6
1011 0996 6
1012 0997 6
1013 0998 7
1014 0999 7
1015 1000 7
1016 1001 7
1017 1002 7
1018 1003 7
1019 1004 7
1020 1005 6
1021 1006 6
1022 1007 5
1023 1008 5
1024 1009 6
1025 1010 6
1026 1011 6
1027 1012 6
1028 1013 5
1029 1014 4
1030 1015 2
1031 1016 2
: 1032 1017 2
: 1033 1018 1

```

```

S = GIVE_COLL( CHAR[CHAR_ALL], CS[CS_PTAB_(.I XOR .CC)] );
IF_ERROR_( .S ) THEN RETURN .S;
END;

! For all double characters that contain no lower case letters,
! and that contain upper case letters, define lower case forms.
CHAR[CHAR_LEN] = 2;
FOR_ALL_DCHARS(ST)
  IF NOT .BV_L[.ST[ST_CHAR_0]] AND
  NOT .BV_L[.ST[ST_CHAR_1]]
  THEN
    BEGIN
      CHAR[CHAR_C0] = .ST[ST_CHAR];
      IF .BV_L[.ST[ST_CHAR_0] XOR .CC]
      THEN
        BEGIN
          CHAR[CHAR_C0] = .CHAR[CHAR_C0] XOR .CC;
          S = GIVE_COLL( CHAR[CHAR_ALL], ST[ST_COLL] );
          IF_ERROR_( .S ) THEN RETURN .S;
          IF .BV_L[.ST[ST_CHAR_1] XOR .CC]
          THEN
            BEGIN
              CHAR[CHAR_C1] = .CHAR[CHAR_C1] XOR .CC;
              S = GIVE_COLL( CHAR[CHAR_ALL], ST[ST_COLL] );
              IF_ERROR_( .S ) THEN RETURN .S;
              CHAR[CHAR_C0] = .CHAR[CHAR_C0] XOR .CC;
              S = GIVE_COLL( CHAR[CHAR_ALL], ST[ST_COLL] );
              IF_ERROR_( .S ) THEN RETURN .S;
            END;
          END
        END
      ELIF .BV_L[.ST[ST_CHAR_1] XOR .CC]
      THEN
        BEGIN
          CHAR[CHAR_C1] = .CHAR[CHAR_C1] XOR .CC;
          S = GIVE_COLL( CHAR[CHAR_ALL], ST[ST_COLL] );
          IF_ERROR_( .S ) THEN RETURN .S;
        END;
      END;
    END;
  END;
END_ALL_DCHARS(ST);
RETURN SSS_NORMAL;
END;

```

		OFFC 00000		.ENTRY COLL\$FOLD, Save R2,R3,R4,R5,R6,R7,R8,R9,-	: 0919
57	0000V	CF 9E 00002		MOVAB GIVE_COLL, R7	:
5E		04 C2 00007		SUBL2 #4, SP	:
50	04	AC D0 0000A		MOVL COLL_SEQ, R0	: 0966
5A	04	A0 D0 0000E		MOVL 4(R0), CS	:
6E		01 B0 000 2		MOVW #1, CHAR	: 0970
55	08	AC D0 00015		MOVL BV_L, R5	: 0972
53	FF	8F 9A 00019		MOVZBL #255, I	:

17		65		53	E1	0001D	1\$:	BBC	I, (R5), 2\$	
	02	AE		53	90	00021		MOVB	I, CHAR+2	0975
51		53	0C	AC	CD	00025		XORL3	CC, I, R1	0976
		52	010C	CA41	DE	0002A		MOVAL	268(CS)[R1], R2	
		51		6E	9E	00030		MOVAB	CHAR, R1	
				67	16	00033		JSB	GIVE COLL	
		7B		50	E9	00035		BLBC	S, 6\$	0977
		E2		53	F4	00038	2\$:	SOBGEQ	I, 1\$	0972
		6E		02	80	0003B		MOVW	#2, CHAR	0983
		53	050C	CA	9E	0003E		MOVAB	1292(R10), ST	0984
		56	06	AA	3C	00043		MOVZWL	6(CS), I	
				70	11	00047		BRB	8\$	
		51		63	9A	00049	3\$:	MOVZBL	(ST), R1	0985
66		65		51	E0	0004C		BBS	R1, (R5), 7\$	
		51	01	A3	9A	00050		MOVZBL	1(ST), R1	0986
5E		65		51	E0	00054		BBS	R1, (R5), 7\$	
	02	AE		63	80	00058		MOVW	(ST), CHAR+2	0989
		54	0C	AC	D0	0005C		MOVL	CC, R4	0990
		51		63	9A	00060		MOVZBL	(ST), R1	
		51		54	CC	00063		XORL2	R4, R1	
31		65		51	E1	00066		BBC	R1, (R5), 4\$	
	02	AE		54	8C	0006A		XORB2	R4, CHAR+2	0993
		52	02	A3	9E	0006E		MOVAB	2(ST), R2	0994
		51		6E	9E	00072		MOVAB	CHAR, R1	
				67	16	00075		JSB	GIVE COLL	
		45		50	E9	00077		BLBC	S, 9\$	0995
		51	01	A3	9A	0007A		MOVZBL	1(ST), R1	0996
		51		54	CC	0007E		XORL2	R4, R1	
31		65		51	E1	00081		BBC	R1, (R5), 7\$	
	03	AE		54	8C	00085		XORB2	R4, CHAR+3	0999
		52	02	A3	9E	00089		MOVAB	2(ST), R2	1000
		51		6E	9E	0008D		MOVAB	CHAR, R1	
				67	16	00090		JSB	GIVE COLL	
		2A		50	E9	00092		BLBC	S, 9\$	1001
	02	AE		54	8C	00095		XORB2	R4, CHAR+2	1002
				0F	11	00099		BRB	5\$	1003
		51	01	A3	9A	0009B	4\$:	MOVZBL	1(ST), R1	1007
		51		54	CC	0009F		XORL2	R4, R1	
10		65		51	E1	000A2		BBC	R1, (R5), 7\$	
	03	AE		54	8C	000A6		XORB2	R4, CHAR+3	1010
		52	02	A3	9E	000AA	5\$:	MOVAB	2(ST), R2	1011
		51		6E	9E	000AE		MOVAB	CHAR, R1	
				67	16	000B1		JSB	GIVE COLL	
		09		50	E9	000B3	6\$:	BLBC	S, 9\$	1012
		53		06	C0	000B6	7\$:	ADDL2	#6, ST	1015
		8D		56	F4	000B9	8\$:	SOBGEQ	I, 3\$	0984
		50		01	D0	000BC		MOVL	#1, R0	1017
				04	000BF	9\$:	RET			1018

; Routine Size: 192 bytes, Routine Base: SOR\$RO_CODE + 015C

```

: 1035      1019  1 ROUTINE GIVE_COLL(
: 1036      1020  1   CHAR:  REF CHAR_BLOCK,
: 1037      1021  1   COLL:  REF COLL_BLOCK
: 1038      1022  1   ):      CS_LINK_2 =
: 1039      1023  1   ++
: 1040      1024  1
: 1041      1025  1   FUNCTIONAL DESCRIPTION:
: 1042      1026  1
: 1043      1027  1       Set CHAR to the collating value COLL
: 1044      1028  1
: 1045      1029  1   FORMAL PARAMETERS:
: 1046      1030  1
: 1047      1031  1       CHAR          a character to be defined
: 1048      1032  1
: 1049      1033  1       COLL          a collating value to assign to CHAR
: 1050      1034  1
: 1051      1035  1   IMPLICIT INPUTS:
: 1052      1036  1
: 1053      1037  1       INIT must have already been called.
: 1054      1038  1       CS is specified as a global register.
: 1055      1039  1
: 1056      1040  1   IMPLICIT OUTPUTS:
: 1057      1041  1
: 1058      1042  1       NONE
: 1059      1043  1
: 1060      1044  1   ROUTINE VALUE:
: 1061      1045  1
: 1062      1046  1       Status code
: 1063      1047  1
: 1064      1048  1   SIDE EFFECTS:
: 1065      1049  1
: 1066      1050  1       NONE
: 1067      1051  1
: 1068      1052  1   --
: 1069      1053  2   BEGIN
: 1070      1054  2   LOCAL
: 1071      1055  2   TEMP:  REF COLL_BLOCK;
: 1072      1056  2
: 1073      1057  2   CS_SETUP();
: 1074      1058  2
: 1075      1059  2   CASE .CHAR[CHAR_LEN] FROM 1 TO 2 OF
: 1076      1060  2   SET
: 1077      1061  2   [1]:
: 1078      1062  3   BEGIN
: 1079      1063  3   MOVE_COLL_ALL_( C[CS_PTAB_(.CHAR[CHAR_C0])], COLL[COLL_ALL] );
: 1080      1064  2   END;
: 1081      1065  2   [2]:
: 1082      1066  3   BEGIN
: 1083      1067  3   TEMP = D_LOOKUP(.CHAR[CHAR_C01]);
: 1084      1068  3   IF .TEMP=EQL 0
: 1085      1069  3   THEN
: 1086      1070  3   TEMP = D_NEW(.CHAR[CHAR_C01]);
: 1087      1071  3   IF .TEMP EQL 0 THEN RETURN COLLS_CMLX;
: 1088      1072  3   MOVE_COLL_ALL_( TEMP[COLL_ALL], COLL[COLL_ALL] );
: 1089      1073  2   END;
: 1090      1074  2   [INRANGE, OUTRANGE]: RETURN COLLS_CHAR;
: 1091      1075  2   TES;

```

: 1092
: 1093
1076 2
1077 1
RETURN SSS_NORMAL;
END;

01	53	51	DD	0C000	GIVE_COLL:	PUSHL	R3	1019
	01	63	DO	00002		MOVL	R1, R3	
	0019	000D	AF	00005	1\$:	CASEW	(CHAR), #1, #1	1059
				00009		.WORD	2\$-1\$,-	
							3\$-1\$	
	50	8F	DO	0000D		MOVL	#COLLS_CHAR, R0	1074
		31	11	00014		BRB	7\$	
	50	A3	9A	00016	2\$:	MOVZBL	2(CHAR), R0	1063
010C	CA40	62	DO	0001A		MOVL	(COLL), 268(CS)[R0]	
		22	11	00020		BRB	6\$	1059
	51	A3	3C	00022	3\$:	MOVZWL	2(CHAR), R1	1067
		0000V	30	00026		BSBW	D_LOOKUP	
		50	D5	00029		TSTL	TEMP	1068
		07	12	0002B		BNEQ	4\$	
	51	A3	3C	0002D		MOVZWL	2(CHAR), R1	1070
		0000V	30	00031		BSBW	D_NEW	
		50	D5	00034	4\$:	TSTL	TEMP	1071
		09	12	00036		BNEQ	5\$	
	50	8F	DO	00038		MOVL	#COLLS_CMLX, R0	
		06	11	0003F		BRB	7\$	
	60	62	DO	00041	5\$:	MOVL	(COLL), (TEMP)	1072
	50	01	DO	00044	6\$:	MOVL	#1, R0	1076
		08	BA	00047	7\$:	POPR	#^M<R3>	1077
		05	00049			RSB		

: Routine Size: 74 bytes, Routine Base: SOR\$RO_CODE + 021C

```

1095 1078 1 ROUTINE COLL_VALUE(
1096 1079 1 CHAR:          REF CHAR_BLOCK,      ! Character to look up
1097 1080 1 COLL:         REF COLL_BLOCK      ! Collating value (output)
1098 1081 1 ): CS_LINK_2 =
1099 1082 1 ++
1100 1083 1
1101 1084 1 FUNCTIONAL DESCRIPTION:
1102 1085 1
1103 1086 1     Look up the collating value of a characetr.
1104 1087 1
1105 1088 1 FORMAL PARAMETERS:
1106 1089 1
1107 1090 1     CHAR          a character who's collating value is to be found
1108 1091 1
1109 1092 1     COLL          where CHAR's collating value is to be stored
1110 1093 1
1111 1094 1 IMPLICIT INPUTS:
1112 1095 1
1113 1096 1     INIT must have already been called.
1114 1097 1     CS is specified as a global register.
1115 1098 1
1116 1099 1 IMPLICIT OUTPUTS:
1117 1100 1
1118 1101 1     NONE
1119 1102 1
1120 1103 1 ROUTINE VALUE:
1121 1104 1
1122 1105 1     Status code
1123 1106 1
1124 1107 1 SIDE EFFECTS:
1125 1108 1
1126 1109 1     NONE
1127 1110 1
1128 1111 1 --
1129 1112 2 BEGIN
1130 1113 2 LOCAL
1131 1114 2 TEMP:  COLL_BLOCK,
1132 1115 2 P:     REF COLL_BLOCK;
1133 1116 2
1134 1117 2 CS_SETUP();
1135 1118 2
1136 1119 2 CASE .CHAR[CHAR_LEN] FROM 0 TO 2 OF
1137 1120 2 SET
1138 1121 2
1139 1122 2 [2]:
1140 1123 2 BEGIN
1141 1124 2
1142 1125 2     ! See whether this double character is defined
1143 1126 2     !
1144 1127 2     P = D_LOOKUP(.CHAR[CHAR_C01]);
1145 1128 2     IF .P-NEQ 0
1146 1129 2     THEN
1147 1130 2     BEGIN
1148 1131 2     MOVE COLL_ALL ( COLLE[COLL_ALL], P[COLL_ALL] );
1149 1132 2     RETURN SSS_NORMAL;
1150 1133 2     END;
1151 1134 2

```

```

1152      1135      ! Take the concatenation of the collating values of
1153      1136      ! the two single characters.
1154      1137      !
1155      1138      MOVE_COLL_ALL ( COLL[COLL_ALL], CSECS_PTAB (.CHAR[CHAR_CO])) );
1156      1139      MOVE_COLL_ALL ( TEMPE[COLL_ALL], CSECS_PTAB (.CHAR[CHAR_C1])) );
1157      1140      IF .COLL[COLL_CO] EQL 0
1158      1141      THEN
1159      1142      MOVE_COLL_ALL ( COLL[COLL_ALL], TEMPE[COLL_ALL] )
1160      1143      ELIF .COLL[COLL_CT] EQL 0
1161      1144      THEN
1162      1145      BEGIN
1163      1146      COLL[COLL_C1] = .TEMPE[COLL_CO];
1164      1147      IF .TEMPE[COLL_C1] NEQ 0 THEN RETURN COLLS_THREE;
1165      1148      END
1166      1149      ELSE
1167      1150      IF .TEMPE[COLL_CO] NEQ 0 THEN RETURN COLLS_THREE;
1168      1151      RETURN SSS_NORMAL;
1169      1152      END;
1170      1153      [1]:
1171      1154      BEGIN
1172      1155      MOVE_COLL_ALL ( COLL[COLL_ALL], CSECS_PTAB (.CHAR[CHAR_CO])) );
1173      1156      RETURN SSS_NORMAL;
1174      1157      END;
1175      1158      [0]:
1176      1159      BEGIN
1177      1160      COLL[COLL_CO] = 0;
1178      1161      COLL[COLL_C1] = 0;
1179      1162      RETURN SSS_NORMAL;
1180      1163      END;
1181      1164      [INRANGE,OUTRANGE]:
1182      1165      RETURN COLLS_CMLPX;
1183      1166
1184      1167      TES;
1185      1168
1186      1169      END;

```

		53	DD	00000	COLL_VALUE:				
	5E	04	C2	00002	PUSHL	R3			: 1078
	53	51	D0	00005	SUBL2	#4, SP			
02	00	63	AF	00008	MOVL	R1, R3			
000F	0057	0063		0000C	CASEW	(CHAR), #0, #2			: 1159
					.WORD	8\$-1\$,-			
						7\$-1\$,-			
						2\$-1\$			
	50	00000000G	8F	D0	00012	MOVL	#COLLS_CMLPX, R0		: 1165
			59	11	00019	BRB	10\$		
	51	02	A3	3C	0001B	MOVZWL	2(CHAR), R1		: 1127
			0000V	30	0001F	BSBW	D_LOOKUP		
			50	D5	00022	TSTL	P-		: 1128
			05	13	00024	BEQL	3\$		
	62		60	D0	00026	MOVL	(P), (COLL)		: 1131
			46	11	00029	BRB	9\$: 1132
	50	02	A3	9A	0002B	MOVZBL	2(CHAR), R0		: 1138

62	010C	CA40	D0	0002F		MOVL	268(CS)[R0], (COLL)	
50	03	A3	9A	00035		MOVZBL	3(CHAR) R0	1139
6E	010C	CA40	D0	00039		MOVL	268(CS)[R0], TEMP	
		62	B5	0003F		TSTW	(COLL)	1140
		05	12	00041		BNEQ	4\$	
62		6E	D0	00043		MOVL	TEMP, (COLL)	1142
		29	11	00046		BRB	9\$	1140
	02	A2	B5	00048	4\$:	TSTW	2(COLL)	1143
		09	12	0004B		BNEQ	5\$	
02	A2	6E	B0	0004D		MOVW	TEMP, 2(COLL)	1146
	02	AE	B5	00051		TSTW	TEMP+2	1147
		02	11	00054		BRB	6\$	
		6E	B5	00056	5\$:	TSTW	TEMP	1150
		17	13	00058	6\$:	BEQL	9\$	
50	00000000G	8F	D0	0005A		MOVL	#COLLS_THREE, R0	
		11	11	00061		BRB	10\$	
50	02	A3	9A	00063	7\$:	MOVZBL	2(CHAR) R0	1155
62	010C	CA40	D0	00067		MOVL	268(CS)[R0], (COLL)	
		02	11	0006D		BRB	9\$	1159
		62	D4	0006F	8\$:	CLRL	(COLL)	1160
50		01	D0	00071	9\$:	MOVL	#1, R0	1162
5E		04	C0	00074	10\$:	ADDL2	#4, SP	1169
		08	BA	00077		POPR	#^M<R3>	
		05	00079			RSB		

; Routine Size: 122 bytes, Routine Base: SOR\$RO_CODE + 0266

```

1188 1170 1 GLOBAL ROUTINE COLL$TIE_BREAK(           ! Indicate tie-breaking
1189 1171 1     COLL_SEQ: REF VECTOR[2],
1190 1172 1     ORDER                                     ! Ascending/Descending flag
1191 1173 1     ) =
1192 1174 1 --++
1193 1175 1
1194 1176 1     FUNCTIONAL DESCRIPTION:
1195 1177 1
1196 1178 1     Indicates that a tie-breaking comparison should be done to distinguish
1197 1179 1     records that compare equal with the primary and secondary comparisons.
1198 1180 1     This tie-breaking comparison is a simple binary string compare.
1199 1181 1
1200 1182 1     FORMAL PARAMETERS:
1201 1183 1
1202 1184 1     COLL_SEQ      a two-longword array specifying the length/address
1203 1185 1                   of storage to use for the collating sequence.
1204 1186 1
1205 1187 1     ORDER          indicates whether the simple comparison part of the
1206 1188 1                   tie-breaking should be:
1207 1189 1                   In the normal order      (ORDER = FALSE) or
1208 1190 1                   In the opposite order   (ORDER = TRUE).
1209 1191 1
1210 1192 1     This distinction is important for DEC STD 169, which places lower case
1211 1193 1     letters before their upper case equivalents. It is unrelated to whether
1212 1194 1     the keys are ascending or descending.
1213 1195 1
1214 1196 1
1215 1197 1     IMPLICIT INPUTS:
1216 1198 1
1217 1199 1     INIT must have already been called.
1218 1200 1
1219 1201 1     IMPLICIT OUTPUTS:
1220 1202 1
1221 1203 1     NONE
1222 1204 1
1223 1205 1     ROUTINE VALUE:
1224 1206 1
1225 1207 1     Status code
1226 1208 1
1227 1209 1     SIDE EFFECTS:
1228 1210 1
1229 1211 1     NONE
1230 1212 1
1231 1213 1 --
1232 1214 2     BEGIN
1233 1215 2
1234 1216 2     CS_SETUP(COLL_SEQ);
1235 1217 2
1236 1218 2     CS[CS_TB] = .CS[CS_TB] AND NOT TB$NOTB;
1237 1219 2
1238 1220 2     IF .ORDER
1239 1221 2     THEN
1240 1222 2         CS[CS_REVERSE] = .CS[CS_REVERSE] OR TB$REVERSE;
1241 1223 2
1242 1224 2     RETURN SSS_NORMAL;
1243 1225 1     END;

```

	50		0400 00000		.ENTRY	COLL\$TIE_BREAK, save R10	:	1170
	5A	04	AC D0 00002		MOVL	COLL_SEQ, R0	:	1216
	08 AA	04	A0 D0 00006		MOVL	4(R0), CS	:	
	04	08	04 8A 0000A		BICB2	#4, 8(CS)	:	1218
	0A AA		AC E9 0000E		BLBC	ORDER, 1\$:	1220
	50		01 88 00012		BISB2	#1, 10(CS)	:	1222
			01 D0 00016 1\$:		MOVL	#1, R0	:	1224
			04 00019		RET		:	1225

; Routine Size: 26 bytes. Routine Base: SOR\$RO_CODE + 02E0


```

1245 1226 1 GLOBAL ROUTINE COLL$PAD(
1246 1227 1     COLL_SEQ:      REF VECTOR[2],      ! The collating sequence
1247 1228 1     PAD:           REF CHAR_BLOCK     ! Pad character
1248 1229 1     ) =
1249 1230 1 !++
1250 1231 1
1251 1232 1     FUNCTIONAL DESCRIPTION:
1252 1233 1
1253 1234 1         Specifies a pad character to be used in comparisons of strings with
1254 1235 1         different lengths.
1255 1236 1
1256 1237 1     FORMAL PARAMETERS:
1257 1238 1
1258 1239 1         COLL_SEQ      a two-longword array specifying the length/address
1259 1240 1         of storage to use for the collating sequence.
1260 1241 1
1261 1242 1         PAD           the pad character.
1262 1243 1
1263 1244 1     IMPLICIT INPUTS:
1264 1245 1
1265 1246 1         INIT must have already been called.
1266 1247 1
1267 1248 1     IMPLICIT OUTPUTS:
1268 1249 1
1269 1250 1         NONE
1270 1251 1
1271 1252 1     ROUTINE VALUE:
1272 1253 1
1273 1254 1         Status code
1274 1255 1
1275 1256 1     SIDE EFFECTS:
1276 1257 1
1277 1258 1         NONE
1278 1259 1
1279 1260 1     NOTES:
1280 1261 1
1281 1262 1     Assertion:
1282 1263 1     The purpose of a pad character is to allow strings of different
1283 1264 1     lengths to compare equal.
1284 1265 1     Therefore:
1285 1266 1     There is no reason to have a different pad character for the
1286 1267 1     tie-break.
1287 1268 1     Also:
1288 1269 1     It is unreasonable for the pad character to be ignored.
1289 1270 1     It is unreasonable for the pad character to be the second
1290 1271 1     character of a double character.
1291 1272 1
1292 1273 1     -
1293 1274 2     BEGIN
1294 1275 2
1295 1276 2     CS_SETUP(COLL_SEQ);
1296 1277 2
1297 1278 2     IF .PAD[CHAR_LEN] NEQ 1 THEN RETURN COLL$_PAD;
1298 1279 2     CS[CS_PAD] = .PAD[CHAR_CO];
1299 1280 2     RETURN $$$_NORMAL;
1300 1281 2
1301 1282 1     END;

```

			0400	00000		.ENTRY	COLL\$PAD, Save R10	:	1226
50	04	AC	D0	00002		MOVL	COLL SEQ, R0	:	1276
5A	04	A0	D0	00006		MOVL	4(R0), C\$:	
50	08	AC	D0	0000A		MOVL	PAD, R0	:	1278
01		60	B1	0000E		CMPW	(R0), #1	:	
		08	13	00011		BEQL	1\$:	
50	00000000G	8F	D0	00013		MOVL	#COLL\$_PAD, R0	:	
			04	0001A		RET		:	
09	AA	02	A0	90 0001B	1\$:	MOVB	2(R0), 9(C\$)	:	1279
50		01	D0	00020		MOVL	#1, R0	:	1280
			04	00023		RET		:	1282

: Routine Size: 36 bytes, Routine Base: SOR\$RO_CODE + 02FA

```

: 1303 1283 1 ROUTINE DO_BUMP(X: WORD): CS_LINK_1 = ! Bump collating values >= X
: 1304 1284 1
: 1305 1285 1 ' +
: 1306 1286 1
: 1307 1287 1 FUNCTIONAL DESCRIPTION:
: 1308 1288 1
: 1309 1289 1 Create an unused collating value by increasing all collating values
: 1310 1290 1 that are greater than or equal to X.
: 1311 1291 1
: 1312 1292 1 FORMAL PARAMETERS:
: 1313 1293 1
: 1314 1294 1 X a (single) collating value, passed as a word.
: 1315 1295 1
: 1316 1296 1 IMPLICIT INPUTS:
: 1317 1297 1
: 1318 1298 1 INIT must have already been called.
: 1319 1299 1 CS is specified as a global register.
: 1320 1300 1
: 1321 1301 1 IMPLICIT OUTPUTS:
: 1322 1302 1
: 1323 1303 1 NONE
: 1324 1304 1
: 1325 1305 1 ROUTINE VALUE:
: 1326 1306 1
: 1327 1307 1 Status code
: 1328 1308 1
: 1329 1309 1 SIDE EFFECTS:
: 1330 1310 1
: 1331 1311 1 NONE
: 1332 1312 1
: 1333 1313 1 --
: 1334 1314 2 BEGIN
: 1335 1315 2 MACRO
: 1336 1316 2 BUMP_(Z) = IF .Z GEQ .X THEN Z = .Z + 1 ELSE 0 %;
: 1337 1317 2
: 1338 1318 2 CS_SETUP();
: 1339 1319 2
: 1340 1320 5 FOR_ALL COLLS(P)
: 1341 1321 5 -BUMP_(P[COLL_(0)]);
: 1342 1322 5 BUMP_(P[COLL_(1)]);
: 1343 1323 2 END_ALL COLLS(P);
: 1344 1324 2 RETURN $$$_NORMAL;
: 1345 1325 1 END;

```

```

52 010C 1C BB 0000 DO_BUMP: PUSHR #^M<R2,R3,R4> : 1283
54 010C CA 9E 0002 MOVAB 268(R10), P : 1320
53 010C 04 D0 0007 MOVL #4, STEP :
07 010C 01 D0 0000A MOVL #1, FIRST : 1321
50 0100 53 E9 0000D 1$: BLBC FIRST, 2$ : 1320
50 0100 8F 3C 00010 MOVZWL #256, R0 :
19 11 00015 BRB 6$ :
50 06 AA 3C 00017 2$: MOVZWL 6(CS) R0 :
13 11 0001B BRB 6$ : 1323

```



```

: 1347 1326 1 ROUTINE D_NEW(X: WORD): CS_LINK_1 = ! Get space for new double char
: 1348 1327 1
: 1349 1328 1 +-
: 1350 1329 1
: 1351 1330 1 FUNCTIONAL DESCRIPTION:
: 1352 1331 1
: 1353 1332 1 Get space for the new double character specified by X, and return the
: 1354 1333 1 address in which its collating value will be stored.
: 1355 1334 1
: 1356 1335 1 FORMAL PARAMETERS:
: 1357 1336 1
: 1358 1337 1 X a (double) character, passed as a word.
: 1359 1338 1
: 1360 1339 1 IMPLICIT INPUTS:
: 1361 1340 1
: 1362 1341 1 INIT must have already been called.
: 1363 1342 1 CS is specified as a global register.
: 1364 1343 1
: 1365 1344 1 IMPLICIT OUTPUTS:
: 1366 1345 1
: 1367 1346 1 NONE
: 1368 1347 1
: 1369 1348 1 ROUTINE VALUE:
: 1370 1349 1
: 1371 1350 1 The address in which the collating value will be stored,
: 1372 1351 1 Or zero if no more space is available.
: 1373 1352 1
: 1374 1353 1 SIDE EFFECTS:
: 1375 1354 1
: 1376 1355 1 NONE
: 1377 1356 1
: 1378 1357 1 --
: 1379 1358 2 BEGIN
: 1380 1359 2 LOCAL
: 1381 1360 2 P: REF ST_BLOCK;
: 1382 1361 2 CS_SETUP();
: 1383 1362 2 P = .CS[CS_CURR_SIZE] + ST_K_SIZE;
: 1384 1363 2 IF .P GTRU .CS[CS_SIZE] THEN RETURN 0; ! No more storage!
: 1385 1364 2 CS[CS_CURR_SIZE] = .P;
: 1386 1365 2 CS[CS_DCHAR] = .CS[CS_DCHAR] + 1;
: 1387 1366 2 P = .P + CS[BASE_] - ST_K_SIZE;
: 1388 1367 2 P[ST_CHAR] = .X;
: 1389 1368 2 RETURN P[ST_COLL];
: 1390 1369 1 END;

```

50	02	AA	3C	00000	D_NEW:	MOVZWL	2(CS), P	: 1362
50		06	C0	00004		ADDL2	#6, P	: 1363
10		00	ED	00007		CMPZV	#0, #16, (CS), P	: 1364
		10	1F	0000C		BLSSU	1\$: 1365
02	AA		50	B0	0000E	MOVW	P, 2(CS)	: 1366
		06	AA	B6	00012	INCW	6(CS)	: 1367
50	FA	AA40	9E	00015		MOVAB	-6(CS)[P], P	: 1368
80		51	B0	0001A		MOVW	X, (P)+	: 1369


```

: 1392      1370 1 ROUTINE D_LOOKUP(X: WORD): CS_LINK_1 =      ! Look up a double character
: 1393      1371 1
: 1394      1372 1 ++
: 1395      1373 1
: 1396      1374 1 FUNCTIONAL DESCRIPTION:
: 1397      1375 1
: 1398      1376 1     Get the collating value for a double character.
: 1399      1377 1
: 1400      1378 1 FORMAL PARAMETERS:
: 1401      1379 1
: 1402      1380 1     X          a (double) character, passed as a word.
: 1403      1381 1
: 1404      1382 1 IMPLICIT INPUTS:
: 1405      1383 1
: 1406      1384 1     INIT must have already been called.
: 1407      1385 1     CS is specified as a global register.
: 1408      1386 1
: 1409      1387 1 IMPLICIT OUTPUTS:
: 1410      1388 1
: 1411      1389 1     NONE
: 1412      1390 1
: 1413      1391 1 ROUTINE VALUE:
: 1414      1392 1
: 1415      1393 1     The address of the collating value,
: 1416      1394 1     Or zero if the double character is undefined.
: 1417      1395 1
: 1418      1396 1 SIDE EFFECTS:
: 1419      1397 1
: 1420      1398 1     NONE
: 1421      1399 1
: 1422      1400 1 --
: 1423      1401 2 BEGIN
: 1424      1402 2
: 1425      1403 2 CS_SETUP();
: 1426      1404 2
: 1427      1405 4 FOR_ALL_DCHARS(ST)
: 1428      1406 4     IF :ST[ST_CHAR] EQL .X THEN RETURN ST[ST_COLL];
: 1429      1407 2 END_ALL_DCHARS(ST);
: 1430      1408 2
: 1431      1409 2 RETURN 0;
: 1432      1410 1 END;

```

```

          OC  BB 00000 D_LOOKUP:
          50   050C CA 9E 00002      PUSHR  #^M<R2,R3>      : 1370
          53   06  AA 3C 00007      MOVAB  1292(R10), ST : 1405
          51   11  11 11 0000B      MOVZWL 6(CS), I     : 1406
          52   02  60 B1 0000D 1$:  BRB    3$
          50   08  09 12 00010      CMPW   (ST), X
          50   08  A0 9E 00012      BNEQ   2$
          50   08  52 D0 00016      MOVAB  2(R0), R2
          50   08  08 11 00019      MOVL   R2, R0
          50   06  C0 0001B 2$:  BRB    4$
          50   06  C0 0001B 2$:  ADDL2  #6, ST      : 1407

```

COLL\$UTILITIES
V04-000

H 14
16-Sep-1984 01:06:02 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 13:10:40 [SORT32.SRC]SORCOLUTI.B32;1

Page 46
(23)

EC

53	F4	0001E	3\$:	SOBGEQ	I	1\$
50	D4	00021		CLRL	R0	
0C	BA	00023	4\$:	POPR	#^M<R2,R3>	
05	00	0025		RSB		

: 1405
: 1409
: 1410
:

: Routine Size: 38 bytes, Routine Base: SOR\$RO_CODE + 0381


```
1434 1411 1 | To compress the range of collating values, we must determine what values
1435 1412 1 | are currently in use. In practice, we will determine which of the values
1436 1413 1 | 1..MAX_USED-1 are in use, and whether any larger values are in use.
1437 1414 1 | Unused values are freed, and larger values are decreased; repeat as needed.
1438 1415 1 | If more that 2*K_CHARS distinct values are in use, it would be almost
1439 1416 1 | impossible to "double-up" sufficient values to fit things in a byte, and
1440 1417 1 | certainly not by this code.
1441 1418 1 |
1442 1419 1 | LITERAL
1443 1420 1 |     MAX_USED = 2 * K_CHARS;
1444 1421 1 |
1445 1422 1 | ROUTINE COMPRESS
1446 1423 1 | (
1447 1424 1 |     S_BV: REF BITVECTOR[MAX_USED]
1448 1425 1 | ): CS_LINK_1 =
1449 1426 1 | ++
1450 1427 1 |
1451 1428 1 | FUNCTIONAL DESCRIPTION:
1452 1429 1 |
1453 1430 1 |     Reduce the range of collating values in use by simply accounting for
1454 1431 1 |     unused collating values.
1455 1432 1 |
1456 1433 1 | FORMAL PARAMETERS:
1457 1434 1 |
1458 1435 1 |     S_BV             bitvector (output parameter)
1459 1436 1 |                     each bit indicates whether the corresponding collating
1460 1437 1 |                     value is in use.
1461 1438 1 |
1462 1439 1 | IMPLICIT INPUTS:
1463 1440 1 |
1464 1441 1 |     INIT must have already been called.
1465 1442 1 |     CS is specified as a global register.
1466 1443 1 |
1467 1444 1 | IMPLICIT OUTPUTS:
1468 1445 1 |
1469 1446 1 |     NONE
1470 1447 1 |
1471 1448 1 | ROUTINE VALUE:
1472 1449 1 |
1473 1450 1 |     Status code
1474 1451 1 |
1475 1452 1 | SIDE EFFECTS:
1476 1453 1 |
1477 1454 1 |     NONE
1478 1455 1 |
1479 1456 1 | --
1480 1457 2 | BEGIN
1481 1458 2 | LOCAL
1482 1459 2 |     S_BV_0,
1483 1460 2 |     USED:  %BLISS16(REF) VECTOR[MAX_USED,WORD],
1484 1461 2 |     FREED;
1485 1462 2 |
1486 1463 2 | MACRO SET_BV_(VAL) = IF .VAL LSSU MAX_USED THEN S_BV[.VAL] = TRUE ELSE 0 %;
1487 M 1464 2 | MACRO DEC_BV_(VAL) = IF .VAL LSSU MAX_USED THEN VAL = .USED[.VAL]
1488 1465 2 |     ELSE (VAL = .VAL = .FREED; S_BV_0 = FALSE) %;
1489 1466 2 |
1490 1467 2 | CS_SETUP();
```

```

: 1491
: 1492
: 1493
: 1494
: 1495
: 1496
: 1497
: 1498
: 1499
: 1500
: 1501
: 1502
: 1503
: 1504
: 1505
: 1506
: 1507
: 1508
: 1509
: 1510
: 1511
: 1512
: 1513
: 1514
: 1515
: 1516
: 1517
: 1518
: 1519
: 1520
: 1521
: 1522
: 1523
: 1524
: 1525
: 1526
: 1527
: 1528
: 1529
: 1530
: 1531
: 1532
: 1533
: 1534
: 1535
: 1536
: 1537
: 1538
: 1539
: 1540
: 1541
: 1542
: 1543
: 1544
: 1545
: 1546
: 1547

```

```

1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547

```

```

: Allocate the USED vector in the work area
:
: IF %BLISS(BLISS16) %THEN
USED = .CS[CS_CURR_SIZE] + %SIZE(VECTOR[MAX_USED,WORD]);
IF .USED GTRU .CS[CS_SIZE] THEN RETURN COLLS_CMLPX;
USED = .USED + CS[BASE_] - %SIZE(VECTOR[MAX_USED,WORD]);
%FI

: Use as few distinct collating values as possible
: (without converting single to double or double to single).
:
USED[0] = 0;
WHILE TRUE DO
  BEGIN
    LOCAL
      VAL;

    : Determine which old collating values are being used
    :
    DECR I FROM MAX_USED/%BPVAL-1 TO 0 DO (S BV[0]+(.I*%UPVAL)) = 0;
    %IF (MAX_USED - MAX_USED/%BPVAL*%BPVAL) NEQ 0 %THEN %ERROR('') %FI
    FOR_ALL_COLL(S(P)
      SET_BV (P[COLL_C0]);
      SET_BV (P[COLL_C1]);
    END_ALL_COLL(S(P);

    : Compute the new collating values
    :
    FREED = 0;
    VAL = 0;
    INCR I FROM 1 TO MINU(MAX_USED-1, .CS[CS_COLL_MAX]) DO
      BEGIN
        IF .S BV[I]
          THEN USED[I] = VAL = .VAL + 1
          ELSE FREED = .FREED + 1;
        END;

    : Now convert to the new collating values
    :
    S BV_0 = TRUE;
    FOR_ALL_COLL(S(P)
      DEC_BV (P[COLL_C0]);
      DEC_BV (P[COLL_C1]);
    END_ALL_COLL(S(P);
    CS[CS_COLL_MAX] = .CS[CS_COLL_MAX] - .FREED;

    : Continue?
    :
    IF .S BV_0 THEN EXITLOOP;
    IF .FREED EQL 0 THEN RETURN COLLS_CMLPX;

    END;

RETURN SSS_NORMAL;
END;

```

```

! We got everything
! Can't do any more

```

Label	Address	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418	Op419	Op420	Op421	Op422	Op423	Op424	Op425	Op426	Op427	Op428	Op429	Op430	Op431	Op432	Op433	Op434	Op435	Op436	Op437	Op438	Op439	Op440	Op441	Op442	Op443	Op444	Op445	Op446	Op447	Op448	Op449	Op450	Op451	Op452	Op453	Op454	Op455	Op456	Op457	Op458	Op459	Op460	Op461	Op462	Op463	Op464	Op465	Op466	Op467	Op468	Op469	Op470	Op471	Op472	Op473	Op474	Op475	Op476	Op477	Op478	Op479	Op480	Op481	Op482	Op483	Op484	Op485	Op486	Op487	Op488	Op489	Op490	Op491	Op492	Op493	Op494	Op495	Op496	Op497	Op498	Op499	Op500	Op501	Op502	Op503	Op504	Op505	Op506	Op507	Op508	Op509	Op510	Op511	Op512	Op513	Op514	Op515	Op516	Op517	Op518	Op519	Op520	Op521	Op522	Op523	Op524	Op525	Op526	Op527	Op528	Op529	Op530	Op531	Op532	Op533	Op534	Op535	Op536	Op537	Op538	Op539	Op540	Op541	Op542	Op543	Op544	Op545	Op546	Op547	Op548	Op549	Op550	Op551	Op552	Op553	Op554	Op555	Op556	Op557	Op558	Op559	Op560	Op561	Op562	Op563	Op564	Op565	Op566	Op567	Op568	Op569	Op570	Op571	Op572	Op573	Op574	Op575	Op576	Op577	Op578	Op579	Op580	Op581	Op582	Op583	Op584	Op585	Op586	Op587	Op588	Op589	Op590	Op591	Op592	Op593	Op594	Op595	Op596	Op597	Op598	Op599	Op600	Op601	Op602	Op603	Op604	Op605	Op606	Op607	Op608	Op609	Op610	Op611	Op612	Op613	Op614	Op615	Op616	Op617	Op618	Op619	Op620	Op621	Op622	Op623	Op624	Op625	Op626	Op627	Op628	Op629	Op630	Op631	Op632	Op633	Op634	Op635	Op636	Op637	Op638	Op639	Op640	Op641	Op642	Op643	Op644	Op645	Op646	Op647	Op648	Op649	Op650	Op651	Op652	Op653	Op654	Op655	Op656	Op657	Op658	Op659	Op660	Op661	Op662	Op663	Op664	Op665	Op666	Op667	Op668	Op669	Op670	Op671	Op672	Op673	Op674	Op675	Op676	Op677	Op678	Op679	Op680	Op681	Op682	Op683	Op684	Op685	Op686	Op687	Op688	Op689	Op690	Op691	Op692	Op693	Op694	Op695	Op696	Op697	Op698	Op699	Op700	Op701	Op702	Op703	Op704	Op705	Op706	Op707	Op708	Op709	Op710	Op711	Op712	Op713	Op714	Op715	Op716	Op717	Op718	Op719	Op720	Op721	Op722	Op723	Op724	Op725	Op726	Op727	Op728	Op729	Op730	Op731	Op732	Op733	Op734	Op735	Op736	Op737	Op738	Op739	Op740	Op741	Op742	Op743	Op744	Op745	Op746	Op747	Op748	Op749	Op750	Op751	Op752	Op753	Op754	Op755	Op756	Op757	Op758	Op759	Op760	Op761	Op762	Op763	Op764	Op765	Op766	Op767	Op768	Op769	Op770	Op771	Op772	Op773	Op774	Op775	Op776	Op777	Op778	Op779	Op780	Op781	Op782	Op783	Op784	Op785	Op786	Op787	Op788	Op789	Op790	Op791	Op792	Op793	Op794	Op795	Op796	Op797	Op798	Op799	Op800	Op801	Op802	Op803	Op804	Op805	Op806	Op807	Op808	Op809	Op810	Op811	Op812	Op813	Op814	Op815	Op816	Op817	Op818	Op819	Op820	Op821	Op822	Op823	Op824	Op825	Op826	Op827	Op828	Op829	Op830	Op831	Op832	Op833	Op834	Op835	Op836	Op837	Op838	Op839	Op840	Op841	Op842	Op843	Op844	Op845	Op846	Op847	Op848	Op849	Op850	Op851	Op852	Op853	Op854	Op855	Op856	Op857	Op858	Op859	Op860	Op861	Op862	Op863	Op864	Op865	Op866	Op867	Op868	Op869	Op870	Op871	Op872	Op873	Op874	Op875	Op876	Op877	Op878	Op879	Op880	Op881	Op882	Op883	Op884	Op885	Op886	Op887	Op888	Op889	Op890	Op891	Op892	Op893	Op894	Op895	Op896	Op897	Op898	Op899	Op900	Op901	Op902	Op903	Op904	Op905	Op906	Op907	Op908	Op909	Op910	Op911	Op912	Op913	Op914	Op915	Op916	Op917	Op918	Op919	Op920	Op921	Op922	Op923	Op924	Op925	Op926	Op927	Op928	Op929	Op930	Op931	Op932	Op933	Op934	Op935	Op936	Op937	Op938	Op939	Op940	Op941	Op942	Op943	Op944	Op945	Op946	Op947	Op948	Op949	Op950	Op951	Op952	Op953	Op954	Op955	Op956	Op957	Op958	Op959	Op960	Op961	Op962	Op963	Op964	Op965	Op966	Op967	Op968	Op969	Op970	Op971	Op972	Op973	Op974	Op975	Op976	Op977	Op978	Op979	Op980	Op981	Op982	Op983	Op984	Op985	Op986	Op987	Op988	Op989	Op990	Op991	Op992	Op993	Op994	Op995	Op996	Op997	Op998	Op999	Op1000	Op1001	Op1002	Op1003	Op1004	Op1005	Op1006	Op1007	Op1008	Op1009	Op1010	Op1011	Op1012	Op1013	Op1014	Op1015	Op1016	Op1017	Op1018	Op1019	Op1020	Op1021	Op1022	Op1023	Op1024	Op1025	Op1026	Op1027	Op1028	Op1029	Op1030	Op1031	Op1032	Op1033	Op1034	Op1035	Op1036	Op1037	Op1038	Op1039	Op1040	Op1041	Op1042	Op1043	Op1044	Op1045	Op1046	Op1047	Op1048	Op1049	Op1050	Op1051	Op1052	Op1053	Op1054	Op1055	Op1056	Op1057	Op1058	Op1059	Op1060	Op1061	Op1062	Op1063	Op1064	Op1065	Op1066	Op1067	Op1068	Op1069	Op1070	Op1071	Op1072	Op1073	Op1074	Op1075	Op1076	Op1077	Op1078	Op1079	Op1080	Op1081	Op1082	Op1083	Op1084	Op1085	Op1086	Op1087	Op1088	Op1089	Op1090	Op1091	Op1092	Op1093	Op1094	Op1095	Op1096	Op1097	Op1098	Op1099	Op1100	Op1101	Op1102	Op1103	Op1104	Op1105	Op1106	Op1107	Op1108	Op1109	Op1110	Op1111	Op1112	Op1113	Op1114	Op1115	Op1116	Op1117	Op1118	Op1119	Op1120	Op1121	Op1122	Op1123	Op1124	Op1125	Op1126	Op1127	Op1128	Op1129	Op1130	Op1131	Op1132	Op1133	Op1134	Op1135	Op1136	Op1137	Op1138	Op1139	Op1140	Op1141	Op1142	Op1143	Op1144	Op1145	Op1146	Op1147	Op1148	Op1149	Op1150	Op1151	Op1152	Op1153	Op1154	Op1155	Op1156	Op1157	Op1158	Op1159	Op1160	Op1161	Op1162	Op1163	Op1164	Op1165	Op1166	Op1167	Op1168	Op1169	Op1170	Op1171	Op1172	Op1173	Op1174	Op1175	Op1176	Op1177	Op1178	Op1179	Op1180	Op1181	Op1182	Op1183	Op1184	Op1185	Op1186	Op1187	Op1188	Op1189	Op1190	Op1191	Op1192	Op1193	Op1194	Op1195	Op1196	Op1197	Op1198	Op1199	Op1200	Op1201	Op1202	Op1203	Op1204	Op1205	Op1206	Op1207	Op1208	Op1209	Op1210	Op1211	Op1212	Op1213	Op1214	Op1215	Op1216	Op1217	Op1218	Op1219	Op1220	Op1221	Op1222	Op1223	Op1224	Op1225	Op1226	Op1227	Op1228	Op1229	Op1230	Op1231	Op1232	Op1233	Op1234	Op1235	Op1236	Op1237	Op1238	Op1239	Op1240	Op1241	Op1242	Op1243	Op1244	Op1245	Op1246	Op1247	Op1248	Op1249	Op1250	Op1251	Op1252	Op1253	Op1254	Op1255	Op1256	Op1257	Op1258	Op1259	Op1260	Op1261	Op1262	Op1263	Op1264	Op1265	Op1266	Op1267	Op1268	Op1269	Op1270	Op1271	Op1272	Op1273	Op1274	Op1275	Op1276	Op1277	Op1278	Op1279	Op1280	Op1281	Op1282	Op1283	Op1284	Op1285	Op1286	Op1287	Op1288	Op1289	Op1290	Op1291	Op1292	Op1293	Op1294	Op1295	Op1296	Op1297	Op1298	Op1299	Op1300	Op1301	Op1302	Op1303	Op1304	Op1305	Op1306	Op1307	Op1308	Op1309	Op1310	Op1311	Op1312	Op1313	Op1314	Op1315	Op1316	Op1317	Op1318	Op1319	Op1320	Op1321
-------	---------	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------

0200	53		62	3C	000A5	15\$:	MOVZWL	(P), R3	1511
	8F		53	B1	000A8		CMPW	R3, #512	
			06	1E	000AD		BGEQU	16\$	
	62		6E43	B0	000AF		MOVW	USED[R3], (P)	
			05	11	000B3		BRB	17\$	
	62		56	A2	000B5	16\$:	SUBW2	FREED, (P)	
			57	D4	000B8		CLRL	S BV 0	
0200	53	02	A2	3C	000BA	17\$:	MOVZWL	2(P), R3	1512
	8F		53	B1	000BE		CMPW	R3, #512	
			07	1E	000C3		BGEQU	18\$	
02	A2		6E43	B0	000C5		MOVW	USED[R3], 2(P)	
			06	11	000CA		BRB	19\$	
02	A2		56	A2	000CC	18\$:	SUBW2	FREED, 2(P)	
			57	D4	000D0		CLRL	S BV 0	
	52		55	C0	000D2	19\$:	ADDL2	STEP P	1513
	CD		50	F4	000D5	20\$:	SOBGEQ	I, 15\$	1510
	55		06	D0	000D8		MOVL	#6, STEP	1513
	52		02	C0	000DB		ADDL2	#2, P	
	B4		54	F4	000DE		SOBGEQ	FIRST, 13\$	1510
04	AA		56	A2	000E1		SUBW2	FREED, 4(CS)	1514
	10		57	E8	000E5		BLBS	S BV 0, 22\$	1518
			56	D5	000E8		TSTL	FREED	1519
			03	13	000EA		BEQL	21\$	
			FF21	31	000EC		BRW	1\$	
	50	00000000G	8F	D0	000EF	21\$:	MOVL	#COLLS_CMPLX, R0	
			03	11	000F6		BRB	23\$	
	50		01	D0	000F8	22\$:	MOVL	#1, R0	1523
	5E	0400	CE	9E	000FB	23\$:	MOVAB	1024(SP), SP	1524
		01FC	8F	BA	00100		POPR	#^M<R2,R3,R4,R5,R6,R7,R8>	
			05	00104			RSB		

: Routine Size: 261 bytes, Routine Base: SOR\$RO_CODE + 03A7

```

: 1549      1525  1 ROUTINE COMPRESS_M: CS_LINK_0 =
: 1550      1526  1
: 1551      1527  1  ++
: 1552      1528  1
: 1553      1529  1  FUNCTIONAL DESCRIPTION:
: 1554      1530  1
: 1555      1531  1      Convert the collating sequence to the efficient and succinct form that's
: 1556      1532  1      used by the comparison routines.
: 1557      1533  1
: 1558      1534  1  FORMAL PARAMETERS:
: 1559      1535  1
: 1560      1536  1      NONE
: 1561      1537  1
: 1562      1538  1  IMPLICIT INPUTS:
: 1563      1539  1
: 1564      1540  1      INIT must have already been called.
: 1565      1541  1      CS is specified as a global register.
: 1566      1542  1
: 1567      1543  1  IMPLICIT OUTPUTS:
: 1568      1544  1
: 1569      1545  1      NONE
: 1570      1546  1
: 1571      1547  1  ROUTINE VALUE:
: 1572      1548  1
: 1573      1549  1      Status code
: 1574      1550  1
: 1575      1551  1  SIDE EFFECTS:
: 1576      1552  1
: 1577      1553  1      NONE
: 1578      1554  1
: 1579      1555  1  --
: 1580      1556  2  BEGIN
: 1581      1557  2  LOCAL
: 1582      1558  2      BV:      BITVECTOR[MAX_USED],
: 1583      1559  2      NEED,
: 1584      1560  2      S;                          ! Status value
: 1585      1561  2
: 1586      1562  2  CS_SETUP():
: 1587      1563  2
: 1588      1564  2  |+
: 1589      1565  2  | We are going to mash this collating sequence down to size.
: 1590      1566  2  | -
: 1591      1567  2
: 1592      1568  2  | First, check that the pad character isn't used in any double characters,
: 1593      1569  2  | and that it collates to a single byte collating value.
: 1594      1570  2  |
: 1595      1571  4  FOR_ALL_DCHARS(ST)
: 1596      1572  4      IF .ST[ST_CHAR_0] EQL .CS[CS_PAD]
: 1597      1573  4      OR .ST[ST_CHAR_1] EQL .CS[CS_PAD] THEN RETURN COLL$_PAD;
: 1598      1574  2  END_ALL_DCHARS(ST);
: 1599      1575  2  IF .BBLOCK[CS[CS_PTAB_(.CS[CS_PAD])],COLL_C1] NEQ 0 THEN RETURN COLL$_PAD;
: 1600      1576  2
: 1601      1577  2
: 1602      1578  2  ! Use fewer collating values
: 1603      1579  2  !
: 1604      1580  2  S = COMPRESS(BV[0]);
: 1605      1581  2  IF_ERROR_(.S) THEN RETURN .S;

```

```

: 1606 1582 2
: 1607 1583 2
: 1608 1584 2
: 1609 1585 2
: 1610 1586 2
: 1611 1587 2
: 1612 1588 5
: 1613 1589 5
: 1614 1590 5
: 1615 1591 2
: 1616 1592 2
: 1617 1593 2
: 1618 1594 2
: 1619 1595 2
: 1620 1596 2
: 1621 1597 2
: 1622 1598 2
: 1623 1599 2
: 1624 1600 2
: 1625 1601 2
: 1626 1602 2
: 1627 1603 2
: 1628 1604 2
: 1629 1605 2
: 1630 1606 2
: 1631 1607 2
: 1632 1608 2
: 1633 1609 2
: 1634 1610 2
: 1635 1611 2
: 1636 1612 2
: 1637 1613 2
: 1638 1614 2
: 1639 1615 2
: 1640 1616 2
: 1641 1617 2
: 1642 1618 2
: 1643 1619 2
: 1644 1620 2
: 1645 1621 2
: 1646 1622 2
: 1647 1623 2
: 1648 1624 2
: 1649 1625 2
: 1650 1626 2
: 1651 1627 2
: 1652 1628 2
: 1653 1629 2
: 1654 1630 2
: 1655 1631 2
: 1656 1632 6
: 1657 1633 6
: 1658 1634 6
: 1659 1635 7
: 1660 1636 7
: 1661 1637 7
: 1662 1638 6

```

```

: Determine some attributes.
: Are there any ignored collating values.
: Are there any double collating values.
FOR_ALL_COLL$S(P)
  IF .P[COLL_C0] EQL 0 THEN CS[CS_IGN] = TRUE;
  IF .P[COLL_C1] NEQ 0 THEN CS[CS_DCOLL] = TRUE;
END_ALL_COLL$S(P);

: A double character <i0,i1> with double collating value <c0,c1> can
: be deleted if:
: The collating value of <i0,0> is <c0,0>, and
: The collating value of <i1,0> is <c1,0>, and
: There are no double characters of the form: <i1,z>
0;

: Determine whether to convert single collating values to double,
: Or to convert double to single.
NEED = .CS[CS_COLL_MAX] - K_CHARS;

: If we already have double collating values or double characters,
: there's not much harm in creating one more to create a free collating
: value. This is advantageous in the comparison routine; also necessary,
: since 0 will be used to indicate a special character.
IF .CS[CS_DCOLL] OR .CS[CS_DCHAR] GTR 0 THEN NEED = .NEED + 1;

: Recall that, on entry to this block, bv[x] indicates that
: the collating value is in use.
IF .NEED GTR 0
THEN
  BEGIN
    : Convert single to double
    : Find a sequence of adjacent (single) collating values that
    : are not used in a double collating value.
    : Convert characters with these collating values to have double
    : collating values.
    LOCAL
      CHAR: CHAR_BLOCK,
      S, 0;
    FOR_ALL_COLL$S(P)
      IF .P[COLL_C1] NEQ 0
      THEN
        BEGIN
          BV[.P[COLL_C0]] = FALSE;
          BV[.P[COLL_C1]] = FALSE;
        END;

```

```

: 1663
: 1664
: 1665
: 1666
: 1667
: 1668
: 1669
: 1670
: 1671
: 1672
: 1673
: 1674
: 1675
: 1676
: 1677
: 1678
: 1679
: 1680
: 1681
: 1682
: 1683
: 1684
: 1685
: 1686
: 1687
: 1688
: 1689
: 1690
: 1691
: 1692
: 1693
: 1694
: 1695
: 1696
: 1697
: 1698
: 1699
: 1700
: 1701
: 1702
: 1703
: 1704
: 1705
: 1706
: 1707
: 1708
: 1709
: 1710
: 1711
: 1712
: 1713
: 1714
: 1715
: 1716
: 1717
: 1718
: 1719

```

```

1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719

```

```

END_ALL_COLL$S(P);
BV[.BLOCK[CS[CS_PTAB_(.CS[CS_PAD])],COLL_CO]] = FALSE;
BV[0] = FALSE;

: Now we know what single collating values are available
:
CHAR[CHAR_LEN] = 1;
Q = .CS[CS_COLL_MAX]+1;
WHILE .NEED GTR 0 DO
  BEGIN
    WHILE (Q=.Q-1) GEQ 0 DO IF .BV[Q] THEN EXITLOOP;
    IF (S=.Q) LEQ 0 THEN RETURN COLL$_CPLX;
    WHILE .BV[(Q=.Q-1)] DO 0;
    IF .S-.Q-1 GTR 0
    THEN
      BEGIN
        IF .S-.Q-1 GTR K_CHARS-1 THEN Q = .S-K_CHARS;
        NEED = .NEED - (.S-.Q-1);
        IF .NEED LSS 0 THEN Q = .Q - .NEED;

        FOR_ALL_COLL$S(P)
          IF
            %IF %FIELDEXPAND(COLL_ALL,2) NEQ 0
            %THEN .P[COLL_ALL] GTR .Q AND .P[COLL_ALL] LEQ .S
            %ELSE .P[COLL_C1] EQL 0 AND
            .P[COLL_CO] GTR .Q AND .P[COLL_CO] LEQ .S
            %FI
          THEN
            BEGIN
              P[COLL_C1] = .P[COLL_CO] - .Q;
              P[COLL_CO] = .S;
            END;
          END_ALL_COLL$S(P);
        END;
      END;
    END;
  S = COMPRESS(BV[0]);
  IF_ERROR_(.S) THEN RETURN .S;
END
ELSE
  BEGIN
    : Try converting double to single
    :
    : We can convert a double collating values <x,y> to a single collating
    : value if either:
    : There are no collating values of the form: <x,0> or <z,x>, or
    : There are no collating values of the form: <y,0> or <y,z>.
    : And (additionally), of double collating values of the form: <x,z>,
    : <x,y> has the y with the largest (or smallest) value.
    0;
  END;

```

```

: 1720      1696      2
: 1721      1697      2
: 1722      1698      2
: 1723      1699      2
: 1724      1700      2
: 1725      1701      2
: 1726      1702      1

```

```

! Check that the pad character is not the second character of a double
! character.
0;
RETURN SSS_NORMAL;
END;

```

		00FC	8F	BB	00000	COMPRESS	M:			
	SE	C0	AE	9E	00004		PUSHR	#^M<R2,R3,R4,R5,R6,R7>		1525
	50	050C	CA	9E	00008		MOVAB	-64(SP), SP		1571
	51	06	AA	3C	0000D		MOVZWL	1292(R10), ST		1572
			10	11	00011		BRB	6(CS), I		
09	AA		60	91	00013	1\$:	CMPB	2\$		
			1A	13	00017		BEQL	(ST), 9(CS)		
09	AA	01	A0	91	00019		CMPB	3\$		1573
			13	13	0001E		BEQL	1(ST), 9(CS)		
	50		06	C0	00020		ADDL2	3\$		1574
	ED		51	F4	00023	2\$:	SOBGEQ	#6, ST		1571
	50	09	AA	9A	00026		MOVZBL	I, 1\$		1575
		010E	CA40	DF	0002A		PUSHAL	9(CS), R0		
			9E	B5	0002F		TSTW	270(CS)[R0]		
			0A	13	00031		BEQL	@(SP)+		
	50	00000000G	8F	D0	00033	3\$:	MOVL	5\$		
			0157	31	0003A	4\$:	BRW	#COLLS_PAD, R0		
	51		6E	9E	0003D	5\$:	MOVAB	35\$		1580
			FEB8	30	00040		BSBW	BV, R1		
	F4		50	E9	00043		BLBC	COMPRESS		1581
	57	010C	CA	9E	00046		MOVAB	S, 4\$		1588
	51		57	D0	0004B		MOVL	268(R10), R7		
	53		04	D0	0004E		MOVL	R7, P		
	52		01	D0	00051		MOVL	#4, STEP		
	07		52	E9	00054	6\$:	BLBC	#1, FIRST		
	50	0100	8F	3C	00057		MOVZWL	FIRST, 7\$		
			1A	11	0005C		BRB	#256, R0		
	50	06	AA	3C	0005E	7\$:	MOVZWL	11\$		
			14	11	00062		BRB	6(CS), R0		1591
			61	B5	00064	8\$:	TSTW	11\$		1589
			04	12	00066		BNEQ	(P)		
0B	AA		02	88	00068		BISB2	9\$		
		02	A1	B5	0006C	9\$:	TSTW	#2, 11(CS)		1590
			04	13	0006F		BEQL	2(P)		
0B	AA		04	88	00071		BISB2	10\$		
	51		53	C0	00075	10\$:	ADDL2	#4, 11(CS)		1591
	E9		50	F4	00078	11\$:	SOBGEQ	STEP, P		1588
	53		06	D0	0007B		MOVL	I, 8\$		1591
	51		02	C0	0007E		ADDL2	#6, STEP		
	D0		52	F4	00081		SOBGEQ	#2, P		1588
	51	04	AA	3C	00084		MOVZWL	FIRST, 6\$		1606
	51	FF00	C1	9E	00088		MOVAB	4(CS), NEED		
05	0B		02	E0	0008D		BBS	-256(R1), NEED		1613
		06	AA	B5	00092		TSTW	#2, 11(CS), 12\$		
								6(CS)		

55		01	D0	00149	MOVL	#1, FIRST	1663
07		55	E9	0014C	BLBC	FIRST, 29\$	1660
50	0100	8F	3C	0014F	MOVZWL	#256, R0	
		1B	11	00154	BRB	32\$	
50	06	AA	3C	00156	MOVZWL	6(CS), R0	
		15	11	0015A	BRB	32\$	1672
53		62	D1	0015C	CMPL	(P), Q	1663
		0D	15	0015F	BLEQ	31\$	
54		62	D1	00161	CMPL	(P), S	
		08	14	00164	BGTR	31\$	
02	A2	62	53	A3	SUBW3	Q, (P), 2(P)	1669
		62	54	B0	MOVW	S, (P)	1670
		52	56	C0	ADDL2	STEP, P	1672
E8		50	F4	00171	SOBGEQ	I, 30\$	1660
56		06	D0	00174	MOVL	#6, STEP	1672
52		02	C0	00177	ADDL2	#2, P	
CF		55	F4	0017A	SOBGEQ	FIRST, 28\$	1660
		FF78	31	0017D	BRW	22\$	1647
51		6E	9E	00180	MOVAB	BV, R1	1677
		FD75	30	00183	BSBW	COMPRESS	
54		50	D0	00186	MOVL	R0, S	
05		54	E8	00189	BLBS	S, 34\$	1678
50		54	D0	0018C	MOVL	S, R0	
		03	11	0018F	BRB	35\$	
50		01	D0	00191	MOVL	#1, R0	1701
5E	40	AE	9E	00194	MVAB	64(SP), SP	1702
	00FC	8F	BA	00198	POPR	#^M<R2,R3,R4,R5,R6,R7>	
		05	0019C	RSB			

; Routine Size: 413 bytes, Routine Base: SOR\$RO_CODE + 04AC

```

: 1728      1703 1 ! Debugging routines
: 1729      1704 1 !
: 1730      L 1705 1 %IF %SWITCHES(DEBUG)
: 1731      U 1706 1 %THEN
: 1732      U 1707 1 LINKAGE
: 1733      U 1708 1     CALL = CALL;
: 1734      U 1709 1 %IF %BLISS(BLISS16) %THEN
: 1735      U 1710 1     MACRO
: 1736      U 1711 1         DELTA_BEGIN = DEL_BEGIN %,
: 1737      U 1712 1         DELTA_END   = DEL_END   %;
: 1738      U 1713 1 %FI
: 1739      U 1714 1 EXTERNAL ROUTINE
: 1740      U 1715 1     DELTA_BEGIN:      CALL,
: 1741      U 1716 1     DELTA:           CALL,
: 1742      U 1717 1     DELTA_END:       CALL,
: 1743      U 1718 1     SOR$$OUTPUT:    CALL;
: 1744      U 1719 1 MACRO
: 1745      U 1720 1     D (X) = UPLIT(%CHARCOUNT(X),UPLIT BYTE(X)) %,
: 1746      U 1721 1     OUT_(X)[ ] = SOR$$OUTPUT(D_(X) %IF %LENGTH GTR 1 %THEN ,%REMAINING %FI) %;
: 1747      U 1722 1
: 1748      U 1723 1 ROUTINE OUT_PT_1(I,CO,C1): NOVALUE =
: 1749      U 1724 1     OUT_('!XB(TAF) CO=!XW, C1=!XW',
: 1750      U 1725 1     .I, 1, .I, .CO, .C1);
: 1751      U 1726 1 ROUTINE OUT_PT_2: NOVALUE = OUT_(' ...');
: 1752      U 1727 1
: 1753      U 1728 1 GLOBAL ROUTINE COLL_DUMP(ADJ): CS_CALL_0 =
: 1754      U 1729 1
: 1755      U 1730 1 !++
: 1756      U 1731 1 !
: 1757      U 1732 1 ! FUNCTIONAL DESCRIPTION:
: 1758      U 1733 1 !
: 1759      U 1734 1 !     Dump the current (uncompressed) collating sequence definition.
: 1760      U 1735 1 !
: 1761      U 1736 1 ! FORMAL PARAMETERS:
: 1762      U 1737 1 !
: 1763      U 1738 1 !     ADJ     (optional) adjustment to be used when writing the '%X' form
: 1764      U 1739 1 !             of the primary table.  For collating sequences with no ignored
: 1765      U 1740 1 !             or double characters, this should be specified as -1, so that
: 1766      U 1741 1 !             the dump can be used in a compilation.
: 1767      U 1742 1 !
: 1768      U 1743 1 ! IMPLICIT INPUTS:
: 1769      U 1744 1 !
: 1770      U 1745 1 !     INIT must have already been called.
: 1771      U 1746 1 !     CS is specified as a global register.
: 1772      U 1747 1 !
: 1773      U 1748 1 ! IMPLICIT OUTPUTS:
: 1774      U 1749 1 !
: 1775      U 1750 1 !     NONE
: 1776      U 1751 1 !
: 1777      U 1752 1 ! ROUTINE VALUE:
: 1778      U 1753 1 !
: 1779      U 1754 1 !     Status code
: 1780      U 1755 1 !
: 1781      U 1756 1 ! SIDE EFFECTS:
: 1782      U 1757 1 !
: 1783      U 1758 1 !     NONE
: 1784      U 1759 1 !

```

```

: 1785      U 1760 1  !--
: 1786      U 1761 1  BEGIN
: 1787      U 1762 1
: 1788      U 1763 1  CS_SETUP();
: 1789      U 1764 1
: 1790      U 1765 1  OUT_(%STRING(
: 1791      U 1766 1  'SIZE=!XW, CURR_SIZE=!XW, COLL_MAX=!XW, TB=!UB, ',
: 1792      U 1767 1  'DCHAR=!XW, PAD=!XB'),
: 1793      U 1768 1  .CS[CS_SIZE],
: 1794      U 1769 1  .CS[CS_CURR_SIZE],
: 1795      U 1770 1  .CS[CS_COLL_MAX],
: 1796      U 1771 1  .CS[CS_TB],
: 1797      U 1772 1  .CS[CS_DCHAR],
: 1798      U 1773 1  .CS[CS_PAD]);
: 1799      U 1774 1  OUT_(%STRING(
: 1800      U 1775 1  'MODS=!UB, IGN=!UB, DCOLL=!UB!/PTAB:'),
: 1801      U 1776 1  .CS[CS_MODS],
: 1802      U 1777 1  .CS[CS_IGN],
: 1803      U 1778 1  .CS[CS_DCOLL]);
: 1804      U 1779 1  !
: 1805      U 1780 1  ! cs_pstatic=      [$address],      ! Address of static base table
: 1806      U 1781 1  ! cs_ustatic=      [$address],      ! Address of static upper table
: 1807      U 1782 1  ! cs_upper=      [$bytes(k_chars)], ! Secondary table
: 1808      U 1783 1  DELTA BEGIN(%B'1111',OUT_PT_1,OUT_PT_2);
: 1809      U 1784 1  INCR I FROM 0 TO K_CHARS-1 DO
: 1810      U 1785 1  BEGIN
: 1811      U 1786 1  LOCAL P: REF COLL_BLOCK;
: 1812      U 1787 1  P = CSE[CS_PTAB(.I)];
: 1813      U 1788 1  DELTA(.I, .P[COLL_CO], .P[COLL_C1]);
: 1814      U 1789 1  END;
: 1815      U 1790 1  DELTA_END();
: 1816      U 1791 1
: 1817      U 1792 1  OUT_('ST:');
: 1818      U 1793 1  FOR_ALL_DCHARS(ST)
: 1819      U 1794 1  OUT_('!XW(!AF) CO=!XW, C1=!XW',
: 1820      U 1795 1  .ST[ST_CHAR],
: 1821      U 1796 1  2, ST[ST_CHAR],
: 1822      U 1797 1  .BBLOCK[ST[ST_COLL], COLL_CO],
: 1823      U 1798 1  .BBLOCK[ST[ST_COLL], COLL_C1]);
: 1824      U 1799 1  END_ALL_DCHARS(ST);
: 1825      U 1800 1
: 1826      U 1801 1  INCR I FROM 0 TO K_CHARS/8-1 DO
: 1827      U 1802 1  BEGIN
: 1828      U 1803 1  STRUCTURE COLL_VECTOR[I] = [ ](COLL_VECTOR+I*COLL_K_SIZE)<0,%BPVAL,0>;
: 1829      U 1804 1  LOCAL P: REF COLL_VECTOR;
: 1830      U 1805 1  P = CSE[CS_PTAB(8*.I)];
: 1831      U 1806 1  OUT_(%STRING(
: 1832      U 1807 1  '%x'!XB'! , %x'!XB'! , %x'!XB'! , %x'!XB'! ,
: 1833      U 1808 1  '%x'!YB'! , %x'!XB'! , %x'!XB'! , %x'!XB'! ,
: 1834      U 1809 1  .P[0]+.ADJ, .P[1]+.ADJ, .P[2]+.ADJ, .P[3]+.ADJ,
: 1835      U 1810 1  .P[4]+.ADJ, .P[5]+.ADJ, .P[6]+.ADJ, .P[7]+.ADJ);
: 1836      U 1811 1  END;
: 1837      U 1812 1
: 1838      U 1813 1  RETURN SSS_NORMAL;
: 1839      U 1814 1  END;
: 1840      U 1815 1  %FI

```

```

: 1842      1816  1 GLOBAL ROUTINE COLL$RESULT(
: 1843      1817  1     COLL_SEQ:      REF VECTOR[2],           ! The collating sequence
: 1844      1818  1     RESLEN:       REF VECTOR[1]           ! Returned length
: 1845      1819  1     ) =
: 1846      1820  1
: 1847      1821  1 ++
: 1848      1822  1 FUNCTIONAL DESCRIPTION:
: 1849      1823  1
: 1850      1824  1     Compress the collating sequence for storage and use by the comparison
: 1851      1825  1     routines.
: 1852      1826  1
: 1853      1827  1 FORMAL PARAMETERS:
: 1854      1828  1
: 1855      1829  1     COLL_SEQ      a two-longword array specifying the length/address
: 1856      1830  1     of storage to use for the collating sequence.
: 1857      1831  1
: 1858      1832  1     RESLEN       a word (output parameter) into which the length of the
: 1859      1833  1     compressed collating sequence description is written.
: 1860      1834  1     Thus, only RESLEN bytes of the storage specified by
: 1861      1835  1     COLL_SEQ needs to be saved.
: 1862      1836  1
: 1863      1837  1 IMPLICIT INPUTS:
: 1864      1838  1
: 1865      1839  1     INIT must have already been called.
: 1866      1840  1
: 1867      1841  1 IMPLICIT OUTPUTS:
: 1868      1842  1
: 1869      1843  1     NONE
: 1870      1844  1
: 1871      1845  1 ROUTINE VALUE:
: 1872      1846  1
: 1873      1847  1     Status code
: 1874      1848  1
: 1875      1849  1 SIDE EFFECTS:
: 1876      1850  1
: 1877      1851  1     NONE
: 1878      1852  1
: 1879      1853  1 --
: 1880      1854  2 BEGIN
: 1881      1855  2 LOCAL
: 1882      1856  2     ADJ,
: 1883      1857  2     TAB:      %BLISS16(REF) VECTOR[K_CHARS, BYTE],
: 1884      1858  2     UPP:      %BLISS16(REF) VECTOR[K_CHARS, BYTE],
: 1885      1859  2     NEWS_P: REF VECTOR[,WORD];
: 1886      1860  2 MACRO
: 1887      1861  2     NEWS (X,Y) =
: 1888      1862  2     BEGIN
: 1889      1863  2     NEWS_P[0] = X; NEWS_P = NEWS_P[1];
: 1890      1864  2     %IF %NULL(Y)
: 1891      1865  2     %THEN
: 1892      1866  2     NEWS_P[0] = 0; NEWS_P = NEWS_P[1]
: 1893      1867  2     %ELSE
: 1894      1868  2     CH$WCHAR_A(.BBLOCK[Y, COLL_C0], NEWS_P);
: 1895      1869  2     CH$WCHAR_A(.BBLOCK[Y, COLL_C1], NEWS_P);
: 1896      1870  2     %FI
: 1897      1871  2     END %;
: 1898      1872  2 MACRO

```

```

: 1899      1873  2
: 1900      1874  2
: 1901      L 1875  2
: 1902      1876  2
: 1903      1877  2
: 1904      1878  2
: 1905      1879  2
: 1906      1880  2
: 1907      1881  2
: 1908      1882  2
: 1909      U 1883  2
: 1910      UU 1884  2
: 1911      UU 1885  2
: 1912      UU 1886  2
: 1913      UU 1887  2
: 1914      UU 1888  2
: 1915      UU 1889  2
: 1916      U 1890  2
: 1917      1891  2
: 1918      1892  2
: 1919      1893  2
: 1920      1894  2
: 1921      1895  2
: 1922      1896  2
: 1923      1897  2
: 1924      1898  3
: 1925      1899  3
: 1926      1900  2
: 1927      1901  2
: 1928      1902  2
: 1929      1903  2
: 1930      1904  2
: 1931      1905  2
: 1932      1906  2
: 1933      1907  2
: 1934      1908  2
: 1935      1909  2
: 1936      1910  2
: 1937      L 1911  2
: 1938      UU 1912  2
: 1939      U 1913  2
: 1940      1914  2
: 1941      1915  2
: 1942      1916  2
: 1943      1917  2
: 1944      U 1918  2
: 1945      UU 1919  2
: 1946      UU 1920  2
: 1947      UU 1921  2
: 1948      UU 1922  2
: 1949      U 1923  2
: 1950      1924  2
: 1951      1925  2
: 1952      1926  2
: 1953      1927  2
: 1954      1928  2
: 1955      1929  3

```

```

RES_STAB_TMP = CS_UPPER %;

%IF %BLISS(BLISS32)
%THEN
EXTERNAL ROUTINE
SOR$$COLLATE_0: ADDRESSING_MODE(LONG_RELATIVE),
SOR$$COLLATE_1: ADDRESSING_MODE(LONG_RELATIVE),
SOR$$COLLATE_2: ADDRESSING_MODE(LONG_RELATIVE),
SOR$$COLLATE_0_A: ADDRESSING_MODE(LONG_RELATIVE),
SOR$$COLLATE_1_A: ADDRESSING_MODE(LONG_RELATIVE)
%ELSE
Because of overlay structure, Sort-11 has to resolve the
addresses on the fly
BJND
SOR$$COLLATE_0 = 0,
SOR$$COLLATE_1 = 1,
SOR$$COLLATE_2 = 2
%FI;

CS_SETUP(COLL_SEQ);
! Compress the tables
BEGIN LOCAL STATUS;
STATUS = COMPRESS M();
IF ERROR_(.STATUS) THEN RETURN .STATUS;
END;

! Compute the adjustment
! This is 1, unless we have: double characters or double collating values
! or ignored characters, in which case it is zero.
! If the adjustment is zero, we will use a zero in the primary table to
! indicate that the secondary table must be used.
ADJ = 1;
IF .CS[CS_DCOLL] OR .CS[CS_IGN] OR .CS[CS_DCHAR] GTR 0 THEN ADJ = 0;

%IF %SWITCHES(DEBUG)
%THEN
COLL_DUMP(-.ADJ);
%FI

! Allocate the TAB and UPP tables in the work area
%IF %BLISS(BLISS16) %THEN
TAB = .CS[CS_CURR_SIZE] + 2 * %SIZE(VECTOR[K_CHARS, BYTE]);
IF .TAB GTRU .CS[CS_SIZE] THEN RETURN COLL$_CPLX;
CS[CS_CURR_SIZE] = .TAB;
TAB = .TAB + CS[BASE] - 2 * %SIZE(VECTOR[K_CHARS, BYTE]);
UPP = .TAB + %SIZE(VECTOR[K_CHARS, BYTE]);
%FI

! First, compute the primary table (into tab)
CH$FILL(0, K_CHARS, TAB[0]);
BEGIN

```

1956	1930	3
1957	1931	3
1958	1932	3
1959	1933	4
1960	1934	4
1961	1935	4
1962	1936	3
1963	1937	2
1964	1938	4
1965	1939	4
1966	1940	2
1967	1941	2
1968	1942	2
1969	1943	2
1970	1944	2
1971	1945	2
1972	1946	2
1973	1947	2
1974	1948	2
1975	1949	2
1976	1950	2
1977	1951	2
1978	1952	2
1979	1953	2
1980	1954	2
1981	1955	2
1982	1956	2
1983	1957	2
1984	1958	2
1985	1959	2
1986	1960	2
1987	1961	2
1988	1962	3
1989	1963	3
1990	1964	3
1991	1965	3
1992	1966	3
1993	1967	3
1994	1968	3
1995	1969	3
1996	1970	3
1997	1971	3
1998	1972	3
1999	1973	3
2000	1974	3
2001	1975	3
2002	1976	3
2003	1977	3
2004	1978	3
2005	1979	3
2006	1980	3
2007	1981	3
2008	1982	4
2009	1983	4
2010	1984	4
2011	1985	4
2012	1986	5

```

LOCAL P: REF COLL_BLOCK;
P = CS[CS_PTAB] + (K_CHARS-1) * COLL_K_SIZE;
DECR I FROM K_CHARS-T TO 0 DO
  BEGIN
    IF .P[COLL_C1] EQL 0 THEN TAB[I] = .P[COLL_CO] - .ADJ;
    P = .P - COLL_K_SIZE;
  END;
END;
FOR _ALL_DCHARS(ST)
  TAB[ST[ST_CHAR_0]] = 0;
END _ALL_DCHARS(ST);

! Copy the upper table
CH$MOVE(K_CHARS, CS[CS_UPPER], UPP[0]);

! Don't bother using silly upper tables.
IF CH$EQL(0, UPP[0], K_CHARS, UPP[0], .UPP[0])
THEN
  CS[CS_TB] = .CS[CS_TB] OR TB$NOUPPER;

! Order the entries in the cs_stab table by the character codes.
! This is needed if there are several double characters with the
! same first character. Note that the entry with the smallest value
! must be the first one accessed by the for_all_dchars macro.
! This code depends on the for_all_dchars macro accessing the entries
! in order from lower addresses to higher addresses.
BEGIN
MACRO
  SWAP_(X,Y) = (T = .X; X = .Y; Y = .T) %;
  SWAP_ST_(X,Y) =
    BEGIN
      LOCAL T;
      SWAP_(X[ST_CHAR], Y[ST_CHAR]);
      %IF %FIELDEXPAND(ST_COLL,2) NEQ 0
      %THEN
        SWAP_(X[ST_COLL], Y[ST_COLL]);
      %ELSE
        SWAP_(BBLOCK[X[ST_COLL],COLL_CO],BBLOCK[Y[ST_COLL],COLL_CO]);
        SWAP_(BBLOCK[X[ST_COLL],COLL_C1],BBLOCK[Y[ST_COLL],COLL_C1]);
      %FI
    END %;
LOCAL
  ST_MIN: REF ST_BLOCK,
  ST_1: REF ST_BLOCK,
  ST_2: REF ST_BLOCK;
ST_1 = CS[CS_STAB];
DECR I FROM CS[CS_DCHAR]-1 TO 1 DO
  BEGIN
    ST_MIN = ST_1[BASE_];
    ST_2 = ST_1[BASE_];
    DECR J FROM .I-1 TO 0 DO
      BEGIN

```

2013 1987 5
2014 1988 5
2015 1989 4
2016 1990 4
2017 1991 4
2018 1992 3
2019 1993 2
2020 1994 2
2021 1995 2
2022 1996 2
2023 1997 2
2024 1998 2
2025 1999 2
2026 2000 2
2027 2001 2
2028 2002 2
2029 2003 2
2030 2004 2
2031 2005 3
2032 2006 3
2033 2007 3
2034 2008 3
2035 2009 4
2036 2010 4
2037 2011 4
2038 2012 4
2039 2013 4
2040 2014 4
2041 2015 4
2042 2016 4
2043 2017 4
2044 2018 4
2045 2019 4
2046 2020 4
2047 2021 4
2048 2022 4
2049 2023 4
2050 2024 5
2051 2025 5
2052 2026 5
2053 2027 4
2054 2028 6
2055 2029 6
2056 2030 6
2057 2031 7
2058 2032 7
2059 2033 7
2060 2034 7
2061 2035 7
2062 2036 7
2063 2037 6
2064 2038 4
2065 2039 3
2066 2040 3
2067 2041 3
2068 2042 2
2069 2043 2

```
ST_2 = .ST_2 + ST_K_SIZE;  
IF .ST_2[ST_CHAR] LSSU .ST_MIN[ST_CHAR] THEN ST_MIN = ST_2[BASE_];  
END;  
SWAP ST (ST_MIN, ST_1);  
ST_1 = .ST_T + ST_K_SIZE;  
END;  
END;  
  
: Now compute the secondary table  
: We compute it to cover the cs_upper table.  
: It may extend far enough to cover the cs_ptab table, but we should  
: always be ahead, unless there are more than k_chars double characters.  
NEWS_P = CS[RES_STAB_TMP];  
IF .ADJ EQL 0  
THEN  
BEGIN  
: This must be an incr loop  
INCR PT_IDX FROM 0 TO K_CHARS-1 DO IF .TAB[PT_IDX] EQL 0 THEN  
BEGIN  
LOCAL  
ENTRY;  
ENTRY = FALSE;  
IF  
%IF %FIELDEXPAND(COLL_ALL,2) NEQ 0  
%THEN .CS[CS_PTAB_(.PT_IDX)] NEQ 0  
%ELSE  
BEGIN  
LOCAL P: REF COLL_BLOCK;  
P = CS[CS_PTAB_(.PT_IDX)];  
.P[COLL_C0] NEQ 0 OR .P[COLL_C1] NEQ 0  
END  
%FI  
THEN  
BEGIN  
NEWS (X'FF00'+.PT_IDX, CS[CS_PTAB_(.PT_IDX)]);  
ENTRY = TRUE;  
END;  
FOR_ALL_DCHARS(ST)  
IF .ST[ST_CHAR_0] EQL .PT_IDX<0,8,0>  
THEN  
BEGIN  
IF NOT .ENTRY  
THEN  
NEWS (X'FF00'+.PT_IDX, CS[CS_PTAB_(.PT_IDX)]);  
ENTRY = TRUE;  
NEWS_(.ST[ST_CHAR], ST[ST_COLL]);  
END;  
END_ALL_DCHARS(ST);  
END;  
NEWS_(X'FFFF');  
NEWS_(X'0000');  
END;
```


0100	8F		00	6E		00	2C	0002A	3\$:	MOVCS	#0, (SP), #0, #256, TAB	1928
				51	FF00	CD		00031				1931
				50	0508	CA	9E	00034		MOVAB	1288(R10), P	1932
					FF	8F	9A	00039		MOVZBL	#255, I	1934
					02	A1	B5	0003D	4\$:	TSTW	2(P)	
						07	12	00040		BNEQ	5\$	
			FF00 CD40	61		59	83	00042		SUBB3	ADJ, (P), TAB[I]	
				51		04	C2	00049	5\$:	SUBL2	#4, P	1935
				EE		50	F4	0004C		SOBGEQ	I, 4\$	1932
				57	050C	CA	9E	0004F		MOVAB	1292(R10), R7	1938
				52		57	D0	00054		MOVL	R7, ST	
				51		06	AA	00057		MOVZWL	6(CS), I	
						0B	11	0005B		BRB	7\$	
				50		82	9A	0005D	6\$:	MOVZBL	(ST)+, R0	1939
					FF00 CD40	94		00060		CLRB	TAB[R0]	
				52		05	C0	00065		ADDL2	#5, ST	1940
				F2		51	F4	00068	7\$:	SOBGEQ	I, 6\$	1938
				58	0C	AA	9E	0006B		MOVAB	12(CS), R8	1945
0100	8F	04	AE	68	0100	8F	28	0006F		MOVCS	#256, (R8), UPP	
		04	AE	AE		00	2D	0007F		CMPCS	#0, UPP, UPP, #256, UPP	1949
						04	AE	0007F				
						04	12	00081		BNEQ	8\$	
				08		02	88	00083		BISB2	#2, 8(CS)	1951
						57	D0	00087	8\$:	MOVL	R7, ST_1	1980
						51	06	AA	3C	0008A	6(CS), -1	1981
						2D	11	0008E		BRB	12\$	
				53		52	D0	00090	9\$:	MOVL	ST_1, ST_MIN	1983
				54		52	D0	00093		MOVL	ST_1, ST_2	1984
				50		51	D0	00096		MOVL	I, J	1985
						0B	11	00099		BRB	11\$	
				54		06	C0	0009B	10\$:	ADDL2	#6, ST_2	1987
				63		64	B1	0009E		CMPW	(ST_2), (ST_MIN)	1988
						03	1E	000A1		BGEQU	11\$	
				53		54	D0	000A3		MOVL	ST_2, ST_MIN	
				F2		50	F4	000A6	11\$:	SOBGEQ	J, 10\$	1985
				50		63	3C	000A9		MOVZWL	(ST_MIN), T	1990
				63		62	B0	000AC		MOVW	(ST_1), (ST_MIN)	
				82		50	B0	000AF		MOVW	T, (ST_1)+	
				50		02	A3	000B2		MOVL	2(ST_MIN), T	
				02		62	D0	000B6		MOVL	(ST, T), 2(ST_MIN)	
				82		50	D0	000BA		MOVL	T, (ST_1)+	
				D0		51	F5	000BD	12\$:	SOBGTR	I, 9\$	1981
				50		58	D0	000C0		MOVL	R8, NEWS_P	2001
						59	D5	000C3		TSTL	ADJ	2002
						69	12	000C5		BNEQ	20\$	
						51	D4	000C7		CLRL	PT_IDX	2008
					FF00 CD41	95		000C9	13\$:	TSTB	TAB[PT_IDX]	
						51	12	000CE		BNEQ	19\$	
				53		55	D4	000D0		CLRL	ENTRY	2012
					010C CA41	DE		000D2		MOVAL	268(CS)[PT_IDX], R3	2015
						63	D5	000D8		TSTL	(R3)	
						10	13	000DA		BEQL	14\$	
				80	FF00	8F	A1	000DC		ADDW3	#65280, PT_IDX, (NEWS_P)+	2025
				80		63	90	000E2		MOVW	(R3), (NEWS_P)+	
				80		02	A3	000E5		MOVW	2(R3), (NEWS_P)+	
				55		01	D0	000E9		MOVL	#1, ENTRY	2026
				54		57	D0	000EC	14\$:	MOVL	R7, ST	2028

			56	06	AA 3C 000EF	MOVZWL	6(CS), I	2029
			29		11 000F3	BRB	18\$	
			51		64 91 000F5	15\$: CMPB	(ST), PT_IDX	
			21		12 000F8	BNEQ	17\$	
			0D		55 E8 000FA	BLBS	ENTRY, 16\$	2032
		80	51	FF00	8F A1 000FD	ADDW3	#65280, PT_IDX, (NEWS_P)+	2034
			80		63 90 00103	MOVW	(R3), (NEWS_P)+	
			80	02	A3 90 00106	MOVW	2(R3), (NEWS_P)+	
			55		01 D0 0010A	16\$: MOVL	#1, ENTRY	2035
			80		64 B0 0010D	MOVW	(ST), (NEWS_P)+	2036
			52	02	A4 9E 00110	MOVAB	2(ST), R2	
			80		62 90 00114	MOVW	(R2), (NEWS_P)+	
			80	02	A2 90 00117	MOVW	2(R2), (NEWS_P)+	
			54		06 C0 0011B	17\$: ADDL2	#6, ST	2038
			D4		56 F4 0011E	18\$: SOBGEQ	I, 15\$	2028
		A0	51	000000FF	8F F3 00121	19\$: AOBLEQ	#255, PT_IDX, 13\$	2008
			80	FFFF	8F 3C 00129	MOVZWL	#65535, (NEWS_P)+	2040
			52		80 D4 0012E	CLRL	(NEWS_P)+	2041
			50		58 C3 00130	20\$: SUBL3	R8, NEWS_P, TMP	2051
		08	BC	020C	C2 9E 00134	MOVAB	524(R2), @RESLEN	2052
08	BC		10		00 ED 0013A	CMPZV	#0, #16, (CS), @RESLEN	2053
			50	00000000G	08 1E 00140	BGEQU	21\$	
			6E	08	AA B0 0014A	21\$: MOVW	8(CS), SAVE	2057
			02	0A	AA 90 0014E	MOVW	10(CS), SAVE+2	2059
			51		D4 00153	CLRL	R1	2061
			59		D5 00155	TSTL	ADJ	
			0B		13 00157	BEQL	22\$	
			51		D6 00159	INCL	R1	
			50	00000000G	EF 9E 0015B	MOVAB	SOR\$\$COLLATE_0, R0	
			15		11 00162	BRB	24\$	
			06		AA B5 00164	22\$: TSTW	6(CS)	2062
			50	00000000G	09 12 00167	BNEQ	23\$	
			50	00000000G	EF 9E 00169	MOVAB	SOR\$\$COLLATE_1, R0	2061
			07		11 00170	BRB	24\$	
			50	00000000G	EF 9E 00172	23\$: MOVAB	SOR\$\$COLLATE_2, R0	
			6A		50 D0 00179	24\$: MOVL	R0, (CS)	
			09		51 E9 0017C	BLBC	R1, 25\$	2066
			50	00000000G	EF 9E 0017F	MOVAB	SOR\$\$COLLATE_0_A, R0	
			10		11 00186	BRB	27\$	
			06		AA B5 00188	25\$: TSTW	6(CS)	2067
			09		12 0018B	BNEQ	26\$	
			50	00000000G	EF 9E 0018D	MOVAB	SOR\$\$COLLATE_1_A, R0	2066
			02		11 00194	BRB	27\$	
			50		D4 00196	26\$: CLRL	R0	
			04	AA	50 D0 00198	27\$: MOVL	R0, 4(CS)	
	020C	CA	68		52 28 0019C	MOVW	4(CS), 524(CS)	2070
	010C	CA	04	AE	0100 8F 28 001A2	MOVW	#256, UPP, 268(CS)	2071
		68	FF00	CD	0100 8F 28 001AB	MOVW	#256, TAB, (R8)	2072
			08	AA	6E B0 001B3	MOVW	SAVE, 8(CS)	2073
			0A	AA	02 AE 90 001B7	MOVW	SAVE+2, 10(CS)	2075
			50		01 D0 001BC	MOVL	#1, R0	2078
			04		04 001BF	RET		2079

; Routine Size: 448 bytes, Routine Base: SOR\$RO_CODE + 0649

: 2106 2080 1
: 2107 2081 1 END
: 2108 2082 0 ELUDOM

PSECT SUMMARY

Name	Bytes	Attributes
SOR\$RO_CODE	2057	NOVEC,NOWRT, RD, EXE, SHR, LCL, REL, CON, PIC,ALIGN(2)
. ABS .	0	NOVEC,NOWRT,NORD,NOEXE,NOSHR, LCL, ABS, CON,NOPIC,ALIGN(0)

Library Statistics

File	Total	Symbols Loaded	Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]XPORT.L32;1	590	36	6	252	00:00.1

COMMAND QUALIFIERS

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

: Size: 2057 code + 0 data bytes
: Run Time: 00:53.9
: Elapsed Time: 02:44.7
: Lines/CPU Min: 2317
: Lexemes/CPU-Min: 38807
: Memory Used: 244 pages
: Compilation Complete

