```
LLL                  IIIIIIIII     BBBBBBBBBBBB     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
LLL                  IIIIIIIII     BBBBBBBBBBBB     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
LLL                  IIIIIIIII     BBBBBBBBBBBB     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
LLL                     III        BBB         BBB  RRR         RRR        TTT        LLL
LLL                     III        BBB         BBB  RRR         RRR        TTT        LLL
LLL                     III        BBB         BBB  RRR         RRR        TTT        LLL
LLL                     III        BBB         BBB  RRR         RRR        TTT        LLL
LLL                     III        BBB         BBB  RRR         RRR        TTT        LLL
LLL                     III        BBB         BBB  RRR         RRR        TTT        LLL
LLL                     III        BBBBBBBBBBBB     RRRRRRRRRRRR           TTT        LLL
LLL                     III        BBBBBBBBBBBB     RRRRRRRRRRRR           TTT        LLL
LLL                     III        BBBBBBBBBBBB     RRRRRRRRRRRR           TTT        LLL
LLL                     III        BBB         BBB  RRR   RRR              TTT        LLL
LLL                     III        BBB         BBB  RRR   RRR              TTT        LLL
LLL                     III        BBB         BBB  RRR   RRR              TTT        LLL
LLL                     III        BBB         BBB  RRR      RRR           TTT        LLL
LLL                     III        BBB         BBB  RRR      RRR           TTT        LLL
LLL                     III        BBB         BBB  RRR      RRR           TTT        LLL
LLLLLLLLLLLLLLL      IIIIIIIII     BBBBBBBBBBBB     RRR         RRR        TTT        LLLLLLLLLLLLLLL
LLLLLLLLLLLLLLL      IIIIIIIII     BBBBBBBBBBBB     RRR         RRR        TTT        LLLLLLLLLLLLLLL
LLLLLLLLLLLLLLL      IIIIIIIII     BBBBBBBBBBBB     RRR         RRR        TTT        LLLLLLLLLLLLLLL
```

FILEID**STRMULTI


```
  SSSSSSSS  TTTTTTTTTT  RRRRRRRR    MM      MM  UU      UU  LL          TTTTTTTTTT    IIIIII
  SSSSSSSS  TTTTTTTTTT  RRRRRRRR    MM      MM  UU      UU  LL          TTTTTTTTTT    IIIIII
SS              TT      RR      RR  MMMM  MMMM  UU      UU  LL              TT          II
SS              TT      RR      RR  MMMM  MMMM  UU      UU  LL              TT          II
SS              TT      RR      RR  MM  MM  MM  UU      UU  LL              TT          II
SS              TT      RR      RR  MM  MM  MM  UU      UU  LL              TT          II
  SSSSSS        TT      RRRRRRRR    MM      MM  UU      UU  LL              TT          II
  SSSSSS        TT      RRRRRRRR    MM      MM  UU      UU  LL              TT          II
        SS      TT      RR  RR      MM      MM  UU      UU  LL              TT          II
        SS      TT      RR  RR      MM      MM  UU      UU  LL              TT          II
        SS      TT      RR    RR    MM      MM  UU      UU  LL              TT          II          ....
        SS      TT      RR    RR    MM      MM  UU      UU  LL              TT          II          ....
SSSSSSSS        TT      RR      RR  MM      MM  UUUUUUUUUU  LLLLLLLLLL      TT        IIIIII        ....
SSSSSSSS        TT      RR      RR  MM      MM  UUUUUUUUUU  LLLLLLLLLL      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
```

```
0 %TITLE 'STR$COMPARE_MULTI - Compare using Multinational Char Set'
0 MODULE STR$COMPARE_MULTI (                ! Compare using Multinational Char Set
0                 IDENT = '1-003'           ! File: STRMULTI.B32 Edit: DG1003
0                 ) =
1 BEGIN
1 !
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1 !
1
1 !++
1 ! FACILITY:     String Support Library
1 !
1 ! ABSTRACT:
1 !
1 !       This module performs character comparisons of 2 input strings
1 !       using the DEC Multinational Character Set (or foreign language
1 !       variations thereof).
1 !
1 ! ENVIRONMENT:  User mode - AST reentrant
1 !
1 ! AUTHOR: Linda Baillie, CREATION DATE: 10-Sept-1982
1 !
1 ! MODIFIED BY:
1 !
1 ! 1-001 - Original.  LGB 10-Sept-1982
1 ! 1-002 - Modified to make changes decided on by Design Review Board
1 !         on March 9, 1983.  DG 13-Sept-1983
1 !         Design notes for the tables and usage of them can be found on
1 !         TURTLE::RTL$:[RTL.NOTE]STRABMUL.MEM.
1 ! 1-003 - Fix so that case-blind comparisons of strings with different
1 !         lengths work.  DG 7-May-1984
1 !--
1
```

```
  0055   1 %SBTTL 'Declarations'
  0056   1 !
  0057   1 ! SWITCHES:
  0058   1 !
  0059   1
  0060   1 SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
  0061   1
  0062   1 !
  0063   1 ! LINKAGES:
  0064   1 !
  0065   1 REQUIRE 'RTLIN:STRLNK';                          ! Linkage to LIB$ANALYZE_DESC_R3
  0250   1 !                                                ! for $str$get_len_addr
  0251   1 !
  0252   1 ! TABLE OF CONTENTS:
  0253   1 !
  0254   1
  0255   1 FORWARD ROUTINE
  0256   1     STR$COMPARE_MULTI;                         ! Compare two strings made up of
  0257   1                                                ! the DEC Multinat'l Char Set
  0258   1 !
  0259   1 ! INCLUDE FILES:
  0260   1 !
  0261   1 REQUIRE 'RTLIN:RTLPSECT';                        ! Define PSECT declarations macros
  0356   1 REQUIRE 'RTLIN:STRMACROS';                       ! String macros
  1272   1 LIBRARY 'RTLSTARLE';                             ! System symbols, typically from SYS$LIBRARY:STARLET.L32
  1273   1
  1274   1 !
  1275   1 ! EQUATED SYMBOLS:
  1276   1 !
  1277   1 LITERAL
  1278   1     TRUE = 1,
  1279   1     FALSE = 0;
  1280   1
  1281   1
  1282   1 !
  1283   1 ! MACROS:
  1284   1 !
  1285   1
  1286   1 MACRO
  1287   1 !+
  1288   1 !   Convert lowercase letters to uppercase.
  1289   1 !   When converting lowercase to uppercase it is necessary to subtract %x'20'
  1290   1 !   from the lowercase table values to reach their uppercase equivalents.
  1291   1 !-
M 1292   1     UPCASE ( IN_BYTE ) =
M 1293   1         BEGIN
M 1294   1
M 1295   1         LOCAL
M 1296   1             TEMP_BYTE : BYTE;
M 1297   1
M 1298   1         TEMP_BYTE = CH$RCHAR( IN_BYTE );
M 1299   1
M 1300   1         IF ( .TEMP_BYTE GEQ %C'a'  AND  .TEMP_BYTE LEQ %C'z' )  OR     ! lowercase letters
M 1301   1            ( .TEMP_BYTE GEQ %X'E0'  AND  .TEMP_BYTE LEQ %X'FD'  AND    ! lc letters w/ diacritical marks
M 1302   1                 .TEMP_BYTE NEQ %X'F0' )
M 1303   1         THEN
M 1304   1             CH$WCHAR( .TEMP_BYTE - %X'20', IN_BYTE );
```

```

        END
    % ;

MACRO
!+
!  Set up generic names to reference STRING1.
!-
    SETUP_STRING1 =
        BEGIN

            STRX_ADDR = .STR1_ADDR;                        ! use generic names
            STRX_LEN  = .STR1_LEN;                         ! for common code
            MULTIX    = .MULTI1;
            CHARX     = .CHAR1;
            ARRAYX    = ARRAY1;

        END
    % ;

MACRO
!+
!  Set up generic names to reference STRING2.
!-
    SETUP_STRING2 =
        BEGIN

            STRX_ADDR = .STR2_ADDR;                        ! use generic names
            STRX_LEN  = .STR2_LEN;                         ! for common code
            MULTIX    = .MULTI2;
            CHARX     = .CHAR2;
            ARRAYX    = ARRAY2;

        END
    % ;

MACRO
!+
!  Search list of special chars.  This is the case where one char is
!  represented by a two-letter sequence.  For example, the German small
!  sharp 's' is represented by the sequence 'ss'.
!  SPEC_CHAR holds the list of special chars.
!  SPEC_SEQ holds the corresponding two-letter sequence.
!-
    SEARCH_SPEC_LIST =
        BEGIN

            LOCAL
                FOUND : INITIAL (FALSE);!  = TRUE if a special character
                                        !  has been found in SPEC_CHAR table

            INCR K FROM 0 TO 5 DO
                BEGIN                                   ! begin loop

                IF .CHARX EQL .SPEC_CHAR[.K]
                THEN
                    BEGIN
```

```
                          !+
                          !  Special character found.
                          !  Put the two-letter sequence that represents the
                          !  special character in ARRAYX.
                          !  SPEC_SEQ [.K*2] points to the first letter of the
                          !  two-letter sequence.
                          !  SPEC_SEQ [(.K*2)+1] points to the second letter of
                          !  the two-letter sequence.
                          !  For every one entry in the SPEC_CHAR table, there
                          !  are two corresponding entries in the SPEC_SEQ table.
                          !-
                          ARRAYX[.COUNT]   = .SPEC_SEQ[.K*2 ];
                          ARRAYX[.COUNT+1] = .SPEC_SEQ[(.K*2)+1];
                          FOUND = TRUE;                          ! spec char found
                          EXITLOOP;

                          END;

                      END;                                       !  end loop

                  !+
                  !  The special char is not in the list of special chars - Error
                  !-
                  IF .FOUND EQL FALSE
                  THEN
                      LIBSSIGNAL ( LIB$_INVARG );

              END                                                ! end of macro
          % ;

      MACRO
      !+
      !  Search list of pairs.  This is the case when two chars appear together
      !  they are sorted in a 'special' way.  For example, the Spanish 'CH' pair
      !  is sorted between 'CZ ' and 'DA'.
      !  SPEC_PAIR holds the list of spec pairs and their ordering values when they
      !  appear together.
      !-
          SEARCH_SPEC_PAIR =
              BEGIN

                  LOCAL
                      FOUND_FIRST  : INITIAL (FALSE),  !  = TRUE if 1st letter of pair
                                                       !  is found in SPEC_PAIR table
                      FOUND_SECOND : INITIAL (FALSE),  !  = TRUE if 2nd letter of pair
                                                       !  is found in SPEC_PAIR table
                      INDEX,                           !  holds # of pairs
                      SAVE_FIRST_LETTER;               !  holds 1st letter of pair for
                                                       !  comparison against all possible
                                                       !  2nd letter partners.

                  !+
                  !  Indices :
                  !  SPEC_PAIR table is set up such that each pair has 4 entries -
                  !    1 - first letter of pair    represented as .SPEC_PAIR[.R*4]
                  !    2 - ordering value for 1st  represented as .SPEC_PAIR[(.R*4)+1]
```

```
		!     3 - second letter of pair   represented as .SPEC_PAIR[(.R*4)+2]
		!     4 - ordering value for 2nd  represented as .SPEC_PAIR[(.R*4)+3]
		!
		!  While R is incremented in loop below, these indices allow all 4
		!  entries of the pair to be looked at in one pass through the loop.
		!-

		INDEX = .PAIR_LEN / 4;				! PAIR_LEN holds # of
								! entries in SPEC_PAIR table
		INCR R FROM 0 TO .INDEX - 1 DO
		    BEGIN					! begin search

		    AGAIN_PAIR_MACRO = FALSE;
		    CALL_SPEC_LIST = FALSE;

		    !+
		    !  Look for first letter of pair
		    !-

		    IF .CHARX EQL .SPEC_PAIR[.R*4]
		    THEN
			BEGIN					! 1st char of possible
								! pair was found
			FOUND_FIRST = TRUE;
			SAVE_FIRST_LETTER = .CHARX;

			!+
			!  Store ordering value of 1st char in ARRAYX
			!-

			ARRAYX[.COUNT] = .SPEC_PAIR[(.R*4)+1];

			!+
			!  Read next character - look for second letter of pair.
			!  If the first letter of the possible pair is the last
			!  letter of the input string, NO_PAIR was set to TRUE in
			!  the routine.
			!-

			IF .NO_PAIR EQL FALSE
			THEN
			    BEGIN				! begin no_pair = FALSE

			    CHARX = CH$RCHAR_A( STRX_ADDR );
			    L = .L + 1;				! increment loop counter
			    IF .JJ EQL 1
			    THEN
				UPCASE( CHARX );

			    !+
			    !  Look for 2nd letter of pair
			    !  It is possible to have a choice of '2nd letters'.
			    !  For example, CH and Ch.
			    !  The following loop handles this situation.
			    !  CHARX would hold the letter from the input string
			    !  that followed the letter C - H or h.
			    !  SPEC_PAIR[(.S*4)+2] would look at H the first time
```

```
        !   through the loop, then h in the subsequent pass.
        !   (always looking at the 2nd letters of the pairs in
        !   the table)
        !   SPEC_PAIR[.S*4] would look at C the first time
        !   through the loop, then different C's in subsequent
        !   passes. (always looking at the 1st letters of the
        !   pairs in the table)
        !   SAVE_FIRST_LETTER remembers what the first letter
        !   of the pair in the input string was.
        !-

        INCR S FROM .R TO .INDEX - 1 DO
            BEGIN                           ! begin 2nd char loop

            IF ( .CHARX EQL ( .SPEC_PAIR[(.S*4)+2] ) )  AND
                ( ( .SPEC_PAIR[.S*4] ) EQL .SAVE_FIRST_LETTER )
            THEN
                BEGIN
                !+
                ! 2nd char found, store in ARRAYX
                !-
                FOUND_SECOND = TRUE;
                ARRAYX[.COUNT+1] = .SPEC_PAIR[(.S*4)+3];
                EXITLOOP;

                END;

            END;                            ! end 2nd char loop


        !+
        !   2nd letter not part of a pair - ok (not an error),
        !   do THAT_TABLE lookup
        !-

        IF .FOUND_SECOND EQL FALSE
        THEN
            BEGIN                           ! begin f_s = FALSE

            MULTIX = .THAT_TABLE[.CHARX];

            !+
            !   FC in THAT_TABLE indicates the first letter of
            !   a possible pair.  Here we have an FC case
            !   following an FC case.  This means the
            !   first FC was not the beginning of a pair, so
            !   check the 2nd FC for a possible pair.
            !   For example, if 'CH' were the only pair in
            !   SPEC_PAIR, the word ACCEPT would get you into
            !   this code.
            !-

            IF .MULTIX EQL %X'FC'
            THEN
                BEGIN
                !+
                !   Acknowledge the previous char being placed
```

```
                                     !  in ARRAYX.  Set flag to call this macro again
                                     !-
                                     COUNT = .COUNT + 1;
                                     AGAIN_PAIR_MACRO = TRUE;

                                     END

                                 !+
                                 !  FD case following FC case.  This means the
                                 !  first FC was not the beginning of a pair,
                                 !  and the char following the FC char is a
                                 !  different kind of spec char.
                                 !  FD in THAT_TABLE indicates a special char that
                                 !  is represented by a two-letter sequence.
                                 !-

                              ELSE
                                  IF .MULTIX EQL %X'FD'
                                  THEN
                                      BEGIN
                                      !+
                                      !  Acknowledge the previous char being
                                      !  placed in ARRAYX.  Set flag to call
                                      !  the appropriate macro for the FD case
                                      !-
                                      COUNT = .COUNT + 1;
                                      CALL_SPEC_LIST = TRUE;

                                      END
                                  ELSE
                                      BEGIN
                                      !+
                                      !  2nd char is not part of pair, nor a
                                      !  special char, store in ARRAYX
                                      !-
                                      ARRAYX[.COUNT+1] = .MULTIX;

                                      END;

                              END;                            ! end f_s = FALSE

                          END                                 ! end no_pair = FALSE
                      ELSE
                          !+
                          !  There is no pair because the first letter of possible
                          !  pair was the last letter of the input string.
                          !  Offset count in routine.
                          !-
                          COUNT = .COUNT - 1;

                      END;                                    ! end 1st char of
                                                              ! possible pair found
                  IF .FOUND_FIRST EQL TRUE
                  THEN
                      EXITLOOP;

                  END;                                        ! end search
```

```
                        !+
                        !  Error - 1st letter of pair is not in pair table SPEC_PAIR
                        !-

                        IF .FOUND_FIRST EQL FALSE
                        THEN
                            LIBSSIGNAL ( LIB$_INVARG );

                    END
                %  ;

        !
        ! FIELDS:
        !
                NONE
        !
        ! PSECTS:
        !
        DECLARE_PSECTS (STR);                               ! Declare PSECTs for STR$ facility
        !
        ! OWN STORAGE:
        !
        !
        ! EXTERNAL REFERENCES:
        !
        EXTERNAL ROUTINE
                LIB$SIGNAL;

        EXTERNAL LITERAL
                LIB$_INVARG;

        EXTERNAL
                STR$AB_MULTI,
                STR$AB_MULTI_SPEC_CHAR,
                STR$AB_MULTI_SPEC_SEQ,
                STR$AB_MULTI_CLASS,
                STR$AB_MULTI_CLASS_SPEC_SEQ,
                STR$AB_DAN_NOR,
                STR$AB_DAN_NOR_CLASS,
                STR$AB_FIN_SWED,
                STR$AB_FIN_SWED_CLASS,
                STR$AB_GERMAN,
                STR$AB_GERMAN_CLASS,
                STR$AB_SPANISH,
                STR$AB_SPANISH_SPEC_PAIR,
                STR$AB_SPANISH_CLASS,
                STR$AB_SPANISH_CLASS_SPEC_PAIR;
```

```
%SBTTL 'STR$COMPARE_MULTI - Compare using Multinational Char Set'
GLOBAL ROUTINE STR$COMPARE_MULTI (
        STRING1: REF $STR$DESCRIPTOR,   ! Pointer to first string descriptor
        STRING2: REF $STR$DESCRIPTOR,   ! Pointer to second string descriptor
        CASE_BLIND_FLAG,                ! Case-blind flag
        FOREIGN_LANG                    ! Choice of ordering table (language)
    ) =

!++
! FUNCTIONAL DESCRIPTION:
!
!       This module performs character comparisons of 2 input strings
!       using the DEC Multinational Character Set (or foreign language
!       variations thereof).
!
! CALLING SEQUENCE:
!
!       ret_status.wlc.v = STR$COMPARE_MULTI ( STRING1.rt.dx, STRING2.rt.dx.,
!               [CASE_BLIND_FLAG.rlu.v], [FOREIGN_LANG.rlu.v] )
!
! FORMAL PARAMETERS:
!
!       STRING1.rt.dx                   ! Pointer to first string descriptor.
!       STRING2.rt.dx                   ! Pointer to second string descriptor.
!       [CASE_BLIND_FLAG.rlu.v]         ! Case-blind flag bit.
!                                       ! bit 0  -  caseblind (equivalence
!                                       !              uppercase to lowercase)
!       [FOREIGN_LANG.rlu.v]            ! Choice of ordering table (language).
!                                       !          1  -  Multinational table
!                                       !          2  -  Danish table
!                                       !          3  -  Finnish/Swedish table
!                                       !          4  -  German table
!                                       !          5  -  Norwegian table
!                                       !          6  -  Spanish table
!
! IMPLICIT INPUTS:
!
!       NONE
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! ROUTINE VALUE:
!
!       COMPARE_STATUS.wl.v     -1  if string1  <  string2
!                                0  if both are the same with blank fill for shorter
!                                1  if string1  >  string2
!
! SIDE EFFECTS:
!
!       May signal STR$_ILLSTRCLA on bad string class
!--
```

```
BEGIN

LOCAL
    CASE_BLIND: INITIAL (FALSE),        ! constant to hold case-blind decision
    SAME: INITIAL (FALSE),              ! if same = TRUE, strings are of = len
    STR1_LEN,                           ! Length of STRING1
    STR2_LEN,                           ! Length of STRING2
    STR1_ADDR_SAV,                      ! Address of 1st data byte of STRING1
    STR2_ADDR_SAV,                      ! Address of 2st data byte of STRING2
    STR_END,                            ! Length of smaller string
    CHAR1: BYTE,                        ! a character of STRING1
    CHAR2: BYTE,                        ! a character of STRING2
    MULTI1: BYTE,                       ! ordering value of a char of STRING1 (from table)
    MULTI2: BYTE,                       ! ordering value of a char of STRING2 (from table)
    NULL_STRING: BYTE INITIAL (%C' '),
                                        ! Null string will be changed to one space
    SPEC_CHAR: REF VECTOR[,BYTE],       ! generic name - special character tables
    SPEC_SEQ: REF VECTOR[,BYTE],        ! generic name - two-letter sequence tables
    SPEC_PAIR: REF VECTOR[,BYTE],       ! generic name - pair of letters tables
    PAIR_LEN,                           ! generic name - # of entries in SPEC_PAIR
                                        !                tables (not # of pairs)
    CLASS_SPEC_SEQ: REF VECTOR[,BYTE],
                                        ! generic name - two-letter sequence tables
    CLASS_SPEC_PAIR: REF VECTOR[,BYTE],
                                        ! generic name - pair of letters tables
    CLASS_TABLE:  REF VECTOR [256,BYTE],
                                        ! generic name for character
                                        !  ordering tables
    THAT_TABLE:  REF VECTOR [256,BYTE];
                                        ! generic name for character
                                        !  ordering tables

LITERAL
    V_CASE_BLIND = 1;                   ! bit flag - equivalence uppercase to lowercase

LABEL
    DO_LOOP;

BUILTIN
    ACTUALCOUNT;
```

```
;  2      !+
;  2      !  Calculate length and starting address of both strings
;  2      !-
;  2      $STR$GET_LEN_ADDR ( STRING1, STR1_LEN, STR1_ADDR_SAV );
;  2      $STR$GET_LEN_ADDR ( STRING2, STR2_LEN, STR2_ADDR_SAV );
;  2
;  2      !+
;  2      !  Check for null strings.
;  2      !  If either is a null string, make it equal to a space.
;  2      !-
;  2      IF .STR1_LEN EQL 0
;  2      THEN
;  3          BEGIN
;  3
;  3          STR1_LEN = 1;
;  3          STR1_ADDR_SAV = NULL_STRING;
;  3
;  2          END;
;  2
;  2       IF .STR2_LEN EQL 0
;  2       THEN
;  3           BEGIN
;  3
;  3           STR2_LEN = 1;
;  3           STR2_ADDR_SAV = NULL_STRING;
;  3
;  2           END;
;  2
;  2      !+
;  2      !  Find string with smaller length to use for loop max value.
;  2      !  If strings are of equal length, set SAME to TRUE.
;  2      !-
;  2      IF .STR1_LEN  EQL  .STR2_LEN
;  2      THEN
;  2          SAME = TRUE;
;  2      STR_END = MINU ( .STR1_LEN, .STR2_LEN );
;  2
;  2      !+
;  2      !  Read Flag parameter ;  set CASE_BLIND appropriately.
;  2      !-
;  3      IF ( ACTUALCOUNT() GEQ 3 )
;  2      THEN
;  3          IF ( .CASE_BLIND_FLAG  AND  V_CASE_BLIND )
;  2          THEN
;  2              CASE_BLIND = TRUE;
;  2
;  2      !+
;  2      !  Set up appropiate ordering tables and special character information.
;  2      !  First set up defaults, then read Foreign_Lang parameter to find out
;  2      !  which ordering table (language) the user has requested (if any).
;  2      !
;  2      TRAT_TABLE = STR$AB_MULTI;                              ! Default - DEC Multi Char Set
;  2      SPEC_CHAR = STR$AB_MULTI_SPEC_CHAR;
;  2      SPEC_SEQ = STR$AB_MULTI_SPEC_SEQ;
;  2      CLASS_TABLE = STR$AB_MULTI_CLASS;
;  2      CLASS_SPEC_SEQ = STR$AB_MULTI_CLASS_SPEC_SEQ;
```

```
IF ( ACTUALCOUNT() EQL 4 )
THEN
    CASE .FOREIGN_LANG FROM 1 TO 6 OF
    SET

    [1]:    ;

    [2]:    BEGIN

            THAT_TABLE = STR$AB_DAN_NOR;                ! Danish Char Set
            CLASS_TABLE = STR$AB_DAN_NOR_CLASS;

            END;

    [3]:    BEGIN

            THAT_TABLE = STR$AB_FIN_SWED;               ! Finnish and Swedish
            CLASS_TABLE = STR$AB_FIN_SWED_CLASS;

            END;

    [4]:    BEGIN

            THAT_TABLE = STR$AB_GERMAN;                 ! German Char Set
            CLASS_TABLE = STR$AB_GERMAN_CLASS;

            END;

    [5]:    BEGIN

            THAT_TABLE = STR$AB_DAN_NOR;                ! Norwegian Char Set
            CLASS_TABLE = STR$AB_DAN_NOR_CLASS;

            END;

    [6]:    BEGIN

            THAT_TABLE = STR$AB_SPANISH;                ! Spanish Char Set
            PAIR_LEN = 32;
            SPEC_PAIR = STR$AB_SPANISH_SPEC_PAIR;
            CLASS_TABLE = STR$AB_SPANISH_CLASS;
            CLASS_SPEC_PAIR = STR$AB_SPANISH_CLASS_SPEC_PAIR;

            END;

    [OUTRANGE]:                                         ! error in Foreign_Lang
            LIB$SIGNAL ( LIB$_INVARG );                 ! parameter

    TES;                                                ! 1 =>.foreign_lang<= 6

!+
!  Compare strings.
!
!  This requires three passes over the strings:
!  1st - a diacritical-blind comparison is done (so that '<a^>a' comes
!        before 'ab').
!  2nd - a case-blind comparison is done (so that 'Aa' comes before 'ab')
```

```
                !  3rd - a non-case-blind comparison is done (so that 'a' comes before 'A').
                !
                !  Loop until end of shorter string.  STR_END holds the length of the
                !  shorter string.  (or length of both strings of string lengths are =).
                !  After all characters of the shorter string are compared against the
                !  first few characters of the longer string, and the strings are equal,
                !  the constant SAME is checked.  If SAME = TRUE, the strings are of equal
                !  length and a routine status of 0 is returned.  Otherwise, do some
                !  further checking on the longer string before returning a routine
                !  status of 0, -1, or 1.
                !  As soon as the comparison shows that the strings are unequal, a
                !  routine status of 1 or -1 is immediately returned.
                !-
               INCR JJ FROM 0 TO 2 DO
                   BEGIN                                                ! begin triple loop

                   LOCAL
                       ARRAY1: VECTOR[100,BYTE],        ! hold ordering values for STRING1 when a special
                                                        !  char is encountered
                       ARRAY2: VECTOR[100,BYTE],        ! hold ordering values for STRING2 when a special
                                                        !  char is encountered
                       ARR_MIN,                         ! indicates which array (ARRAY1 or ARRAY2) is smaller
                       ARR_SAME,                        ! = TRUE if ARRAY1 is same length as ARRAY2
                       USED_ARRAYS,                     ! = TRUE if ARRAY1/2 were used
                       AGAIN_PAIR_MACRO: INITIAL (FALSE),
                                                        ! = TRUE if macro SEARCH_SPEC_PAIR has
                                                        ! to be invoked a second time
                                                        ! immediately after the first time
                       CALL_SPEC_LIST: INITIAL (FALSE),
                                                        ! = TRUE if the macro SEARCH_SPEC_LIST
                                                        ! has to be invoked immediately after
                                                        ! the macro SEARCH_SPEC_PAIR
                       NO_PAIR: INITIAL (FALSE),        ! = TRUE if search for a possible pair
                                                        ! will prove futile
                       STRX_ADDR,                       !     \
                       STRX_LEN,                        !      \
                       MULTIX: BYTE,                    !        generic names for common code
                       CHARX: BYTE,                     !      /
                       ARRAYX: REF VECTOR[,BYTE],       !     /
                       COUNT,                           ! index to ARRAYX (used first for ARRAY1 then
                                                        !  for ARRAY2)
                       COUNT1,                          ! used to save length of ARRAY1 for later
                                                        !  comparison against COUNT of ARRAY2
                       CHARS_READ: INITIAL (0),         ! # of characters read in loop
                       STR1_ADDR,                       ! current ptr into STRING1
                       STR2_ADDR,                       ! current ptr into STRING2
                       SAVE_SPEC_SEQ : REF VECTOR [,BYTE], ! 3 SAVEs used to save info for
                       SAVE_SPEC_PAIR: REF VECTOR [,BYTE], !  1st time thru loop when
                       SAVE_TABLE: REF VECTOR [256,BYTE];  !  class-blind table is used.

                   ARR_SAME = FALSE;
                   USED_ARRAYS = FALSE;
                   STR1_ADDR = .STR1_ADDR_SAV;
                   STR2_ADDR = .STR2_ADDR_SAV;

                   CHAR1 = CH$RCHAR_A( STR1_ADDR );                     ! get char of STRING1
                   CHAR2 = CH$RCHAR_A( STR2_ADDR );                     ! get char of STRING2
```

```
3
3               IF .JJ EQL 0
3               THEN
4                   BEGIN
4                   !+
4                   ! Use class-blind table first time thru loop.
4                   !-
4                   SAVE_TABLE = .THAT_TABLE;
4                   THAT_TABLE = .CLASS_TABLE;
4                   SAVE_SPEC_SEQ = .SPEC_SEQ;
4                   SPEC_SEQ = .CLASS_SPEC_SEQ;
4                   SAVE_SPEC_PAIR = .SPEC_PAIR;
4                   SPEC_PAIR = .CLASS_SPEC_PAIR;
4
3                   END;
3
3               IF .JJ EQL 1
3               THEN
4                   BEGIN
4                   !+
4                   ! Case-blind comparison done second time thru loop.
4                   !-
4                   UPCASE( CHAR1 );
4                   UPCASE( CHAR2 );
4
3                   END;
3
3  DO_LOOP:
4               BEGIN
4
4               INCR J FROM 1 TO .STR_END DO
5                   BEGIN                                       ! begin do
5
5                   !+
5                   !  Get ordering values from table
5                   !-
5                   MULTI1 = .THAT_TABLE [.CHAR1];
5                   MULTI2 = .THAT_TABLE [.CHAR2];
5                   CHARS_READ = .CHARS_READ + 1;               ! # of characters read
5
5                   !+
5                   !     Does either string contain a special character ?
5                   !
5                   !     There are two kinds of special characters -
5                   !     1.  One char represented by a two-letter sequence, indicated in
5                   !         the ordering value table (THAT_TABLE) by the value %X'FD'.
5                   !         ex: German small sharp 's' is represented by the two-letter
5                   !         sequence 'ss'
5                   !     2.  A pair of chars with a "special" sorting order, indicated
5                   !         in the ordering value table (THAT_TABLE) by the value
5                   !         %X'FC'.
5                   !         ex: Spanish 'CH' pair is sorted between 'CZ' and 'DA'
5                   !
5                   !     When a special char is encountered, treat these special cases
5                   !     separately.  Store the special char and all remaining chars of
5                   !     the string in an array.
5                   !     Do this for both strings even if only one of the strings
```

```
!       contained a special char.  After all chars of both strings are
!       stored in an array, compare the arrays.
!-
IF .MULTI1 EQL %X'FD'  OR  .MULTI2 EQL %X'FD'  OR
    .MULTI1 EQL %X'FC'  OR  .MULTI2 EQL %X'FC'
THEN
    BEGIN                                       ! begin special case loop

    USED_ARRAYS = TRUE;

    INCR I FROM 1 TO 2 DO
        BEGIN                                   ! begin outer loop
        !+
        !   These macros allow generic names to be used for both
        !   strings
        !-
        IF .I EQL 1
        THEN
            SETUP_STRING1
        ELSE
            SETUP_STRING2;

        COUNT = 0;                              ! will hold length of ARRAYX
        NO_PAIR = FALSE;

        !+
        !   Create two arrays (one at a time) holding the ordering
        !   values of the remaining chars in both strings.  Start
        !   at the position where the spec char was encountered,
        !   and stop at the last position of the shorter string.
        !   If the first character of one of the strings is a special
        !   char then .CHARS_READ = 1.
        !   If the fourth character of one of the strings is a
        !   special char then .CHARS_READ = 4, etc.
        !-
        INCR L FROM .CHARS_READ TO .STRX_LEN DO
            BEGIN                               ! begin inner loop

            IF .MULTIX EQL %X'FD'
            THEN
                !+
                !   *** TWO-LETTER SEQ CASE ***
                !   ***       FD CASE       ***
                !-
                BEGIN
                !+
                ! Macro to store the ordering values of the
                ! two-letter seq in ARRAYX
                !-
                SEARCH_SPEC_LIST;
                COUNT = .COUNT + 2;             ! two chars stored in ARRAYX

                END
            ELSE
                !+
                !   *** PAIR CASE ***
                !   ***  FC CASE  ***
```

```
 8                                      !-
 9                                      BEGIN                           ! begin fc case
 9
 9                                      IF .MULTIX EQL %X'FC'
 9                                      THEN
 9                                          !+
 9                                          !  Char is possibly the first char of a pair
 9                                          !  (such as the Spanish 'CH' pair, where each
 9                                          !  time a 'C' is encountered a search will be
 9                                          !  made for the second letter of the special
 9                                          !  pair, here the 'H'.  However, not all Spanish
 9                                          !  'C's are followed by 'H's, therefore the
 9                                          !  search for the second letter of the pair may
 9                                          !  not result in success.
 9                                          !-
10                                          BEGIN                       ! begin call pair macro
10                                          !+
10                                          ! Macro to store the ordering values of the pair
10                                          ! of letters in ARRAYX.
10                                          ! If the first letter of a possible pair is the
10                                          ! last letter of the input string, there is no
10                                          ! pair.  The macro SEARCH_SPEC_PAIR is still
10                                          ! needed to store the ordering value of the
10                                          ! last character in ARRAYX.  Set NO_PAIR to TRUE
10                                          ! to flag macro SEARCH_SPEC_PAIR not to look
10                                          ! for the non existant second letter of the
10                                          ! possible pair.
10                                          !-
10                                          IF .L EQL .STRX_LEN
10                                          THEN
10                                              NO_PAIR = TRUE;
10                                          SEARCH_SPEC_PAIR;
10
10                                          !+
10                                          !  It is possible that a 'FD' case immediately
10                                          !  followed the first letter of a 'FC' case,
10                                          !  i.e. there was not a 'pair', but now the FD
10                                          !  case must be addressed.
10                                          !  Constant CALL_SPEC_LIST is set to TRUE in
10                                          !  macro SEARCH_SPEC_PAIR to indicate this
10                                          !  occurence.
10                                          !-
10                                          IF .CALL_SPEC_LIST EQL TRUE
10                                          THEN
10                                              SEARCH_SPEC_LIST;
10
10                                          !+
10                                          !  It might be necessary to invoke the macro
10                                          !  SEARCH_SPEC_PAIR again - for example,
10                                          !  if 'C' were the first letter of
10                                          !  a possible pair but 'CC' was not a pair,
10                                          !  then the word 'ACCEPT' would cause this
10                                          !  macro to be called twice, the first time
10                                          !  looking at 'CC' as a possible pair, then
10                                          !  a second time looking at 'CE' as a possible
10                                          !  pair.  Constant AGAIN_PAIR_MACRO is set to
10                                          !  TRUE in macro SEARCH_SPEC_PAIR to indicate
```

```
                                          !  this occurence.
                                          !-
                                          WHILE .AGAIN_PAIR_MACRO EQL TRUE DO
                                              BEGIN                 ! begin do while
                                              !+
                                              ! Increment loop counter (L) to point to
                                              ! the next FC Case (point to the second
                                              ! 'C' in ACCEPT)
                                              ! If the first letter of a possible pair is
                                              ! the last letter of the input ,
                                              ! there is no pair.  The macro
                                              ! SEARCH_SPEC_PAIR is still needed to store
                                              ! the ordering value of the last character
                                              ! in ARRAYX.  Set NO_PAIR to TRUE to flag
                                              ! macro SEARCH_SPEC_PAIR not to look for the
                                              ! non existant 2nd letter of the possible
                                              ! pair.
                                              !-
                                              L = .L + 1;
                                              IF .L EQL .STRX_LEN
                                              THEN
                                                  NO_PAIR = TRUE;
                                              SEARCH_SPEC_PAIR;

                                              !+
                                              !  Check for FD case following a FC case
                                              !-
                                              IF .CALL_SPEC_LIST EQL TRUE
                                              THEN
                                                  SEARCH_SPEC_LIST;

                                              END;                  ! end do while
                                          !+
                                          !  Acknowledge the storing of two ordering
                                          !  values in ARRAYX
                                          !-
                                          COUNT = .COUNT + 2;

                                          END                       ! end call pair macro
                                      ELSE
                                          !+
                                          !  *** CHAR IS NOT SPECIAL CHARACTER ***
                                          !  ***                               ***
                                          !
                                          !  Store ordering value in ARRAYX.
                                          !  Only one char stored in ARRAYX.
                                          !-
                                          BEGIN                     ! begin regular char case

                                          ARRAYX[.COUNT] = .MULTIX;
                                          COUNT = .COUNT + 1;

                                          END;                      ! end regular char case

                                      END;                          ! end fc case

                                  !+
```

```
                    !  Get next char and do a table lookup.
                    !  (L is the loop incr from above)
                    !-
                    IF .L NEQ .STRX_LEN
                    THEN
                        BEGIN

                        CHARX = CH$RCHAR_A( STRX_ADDR );
                        IF .JJ EQL 1
                        THEN
                            UPCASE( CHARX );

                        MULTIX = .THAT_TABLE[.CHARX];

                        END;

                    END;                                ! end inner loop

                !+
                ! Save length of ARRAY1 for later comparison
                !-
                IF .I EQL 1
                THEN
                    COUNT1 = .COUNT;

                END;                                    ! end outer loop

            !+
            !   Compare ARRAY1 against ARRAY2 and return status.
            !   COUNT1 holds the length of ARRAY1,
            !   COUNT holds the length of ARRAY2.
            !   If arrays are of equal length, set ARR_SAME to TRUE.
            !   ARR_MIN holds length of shorter array.
            !-
            IF .COUNT1 EQL .COUNT
            THEN
                ARR_SAME = TRUE;
            ARR_MIN = MINU ( .COUNT1, .COUNT );

            INCR M FROM 0 TO .ARR_MIN-1 DO
                BEGIN                                   ! begin array compare loop

                IF .ARRAY1[.M] NEQ .ARRAY2[.M]
                THEN
                    !+
                    !  If not equal, return appropriate status
                    !-
                    IF .ARRAY1[.M] LSS .ARRAY2[.M]
                    THEN
                        RETURN -1
                    ELSE
                        RETURN 1;

                END;                                    ! end array compare loop -
                                                        ! end of at least one array.
            !+
            !   The smaller array holds the same values as the larger
```

```
                !   array, but unequal array lengths indicate a mismatch -
                !   if what remains of the longer array is blanks, then
                !   they are equal, else compare remainder against blanks
                !   and return appropriate status.
                !   0 value is returned only on 3rd pass thru loop.
                !-
                IF .ARR_SAME EQL FALSE
                THEN
                    BEGIN                                   ! begin compare

                    LOCAL
                        COMP_VAL;

                    IF .COUNT1 LSS .COUNT
                    THEN
                        BEGIN                               ! begin ARRAY1 is shorter

                        COMP_VAL = CH$COMPARE ( 0, 0, .COUNT - .ARR_MIN,
                                                ARRAY2 + .COUNT1,
                                                STR$K_FILL_CHAR );
                        IF .COMP_VAL EQL 0
                        THEN
                            BEGIN

                            IF .JJ EQL 2
                            THEN
                                RETURN 0                    ! STRING1 (with blank fill) = STRING2
                            ELSE
                                LEAVE DO_LOOP;

                            END
                        ELSE
                            RETURN .COMP_VAL;

                        END                                 ! end ARRAY1 is shorter
                    ELSE
                        BEGIN                               ! begin ARRAY2 is shorter

                        COMP_VAL = CH$COMPARE ( 0, 0, .COUNT1 - .ARR_MIN,
                                                ARRAY1 + .COUNT,
                                                STR$K_FILL_CHAR );
                        IF .COMP_VAL EQL 0
                        THEN
                            BEGIN

                            IF .JJ EQL 2
                            THEN
                                RETURN 0                    ! STRING1 = STRING2 (with blank fill)
                            ELSE
                                LEAVE DO_LOOP;

                            END
                        ELSE
                            RETURN -.COMP_VAL;

                        END ;                               ! end ARRAY2 is shorter
```

```
6                END;                                           ! end compare
6
6            END                                                ! end special case loop
6
6
6
6
6
6       !+
6       ! Look at the 'regular' characters, are they = ?
6       !-
5       ELSE
6            BEGIN                                              ! begin no special chars
6
6            IF .MULTI1 NEQ .MULTI2
6            THEN
6                 !+
6                 ! Not equal, return appropriate status
6                 !-
6                 IF .MULTI1 GTR .MULTI2
6                 THEN
6                      RETURN 1
6                 ELSE
6                      RETURN -1;
6            !+
6            ! Get next char in strings for next time thru do loop.
6            !-
6            CHAR1 = CH$RCHAR_A( STR1_ADDR );                  ! get char of STRING1
6            CHAR2 = CH$RCHAR_A( STR2_ADDR );                  ! get char of STRING2
6            IF .JJ EQL 1
6            THEN
7                BEGIN
7                !+
7                ! UPCASE case equivalences lowercase to uppercase
7                !-
7                UPCASE( CHAR1 );
7                UPCASE( CHAR2 );
7
6                END;
6
6
5            END;                                               ! end of no special chars
5
4       END;                                                    ! end of do
4
3    END;                                                       ! end DO_LOOP
3
3  !+
3  ! If we get to this point the 2nd pass thru the loop, the strings are
3  ! equal up to the point of the shorter string's length.
3  ! The 2nd pass does a case-blind comparison, so if the CASE_BLIND flag
3  ! is set and the strings are of equal length, then return 0 now.
3  !-
4  IF ( .CASE_BLIND EQL TRUE ) AND ( .JJ NEQ 0 )
3  THEN
3      IF .SAME EQL TRUE
```

```
		    THEN
			RETURN 0;

		!+
		!   If we get to this point the 3nd pass thru the loop, the strings are
		!   equal up to the point of the shorter string's length.
		!   If the strings are of equal length, then return 0.
		!-
		IF ( .JJ EQL 2 )  AND  ( .SAME EQL TRUE )
		THEN
		    RETURN 0;

		!+
		! If their lengths are not equal, and what remains of the longer
		! is blanks, then they are equal, else compare remainder against
		! blanks and return appropriate status.
		!-
		IF ( .SAME EQL FALSE )  AND  ( .USED_ARRAYS EQL FALSE )
		THEN
		BEGIN						! begin compare

		LOCAL
		    COMP_VAL;

		IF .STR1_LEN LSSU .STR2_LEN
		THEN
		    BEGIN					! begin STRING1 is shorter

		    COMP_VAL = CH$COMPARE ( 0, 0, .STR2_LEN - .STR_END, .STR2_ADDR - 1,
				STR$K_FILL_CHAR );
		    IF .COMP_VAL EQL 0
		    THEN
			BEGIN

			IF ( .JJ EQL 2 )  OR  ( ( .CASE_BLIND EQL TRUE )  AND  ( .JJ NEQ 0 ) )

			THEN
			    RETURN 0;				! STRING1 (with blank fill) = STRING2
			END
		    ELSE
			RETURN .COMP_VAL;

		    END					! end STRING1 is shorter
		ELSE
		    BEGIN					! begin STRING2 is shorter

		    COMP_VAL = CH$COMPARE ( 0, 0, .STR1_LEN - .STR_END, .STR1_ADDR - 1,
				STR$K_FILL_CHAR );
		    IF .COMP_VAL EQL 0
		    THEN
			BEGIN

			IF ( .JJ EQL 2 )  OR  ( ( .CASE_BLIND EQL TRUE )  AND  ( .JJ NEQ 0 ) )
			THEN
			    RETURN 0;				! STRING1 = STRING2 (with blank fill)

			END
```

```
; 1170      2359  5          ELSE
; 1171      2360  5              RETURN -.COMP_VAL;
; 1172      2361  5
; 1173      2362  4          END ;                                       ! end STRING2 is shorter
; 1174      2363  4
; 1175      2364  3      END;                                            ! end compare
; 1176      2365  3
; 1177      2366  3      !+
; 1178      2367  3      !  Reset the tables for the 2nd and 3rd passes.
; 1179      2368  3      !-
; 1180      2369  3      IF .JJ EQL 0
; 1181      2370  3      THEN
; 1182      2371  4          BEGIN
; 1183      2372  4
; 1184      2373  4          THAT_TABLE = .SAVE_TABLE;
; 1185      2374  4          SPEC_SEQ = .SAVE_SPEC_SEQ;
; 1186      2375  4          SPEC_PAIR = .SAVE_SPEC_PAIR;
; 1187      2376  4
; 1188      2377  3          END;
; 1189      2378  3
; 1190      2379  2      END;                                            ! end of triple loop
; 1191      2380  2
; 1192      2381  2      RETURN 0;                                       ! Needed by BLISS
; 1193      2382  1      END;


                                                .TITLE  STR$COMPARE_MULTI STR$COMPARE_MULTI - Compare u
;                                                                           sing Multinational
                                                .IDENT  \1-003\

                                                .EXTRN  LIB$SIGNAL, LIB$_INVARG
                                                .EXTRN  STR$AB_MULTI, STR$AB_MULTI_SPEC_CHAR
                                                .EXTRN  STR$AB_MULTI_SPEC_SEQ
                                                .EXTRN  STR$AB_MULTI_CLASS
                                                .EXTRN  STR$AB_MULTI_CLASS_SPEC_SEQ
                                                .EXTRN  STR$AB_DAN_NOR, STR$AB_DAN_NOR_CLASS
                                                .EXTRN  STR$AB_FIN_SWED
                                                .EXTRN  STR$AB_FIN_SWED_CLASS
                                                .EXTRN  STR$AB_GERMAN, STR$AB_GERMAN_CLASS
                                                .EXTRN  STR$AB_SPANISH, STR$AB_SPANISH_SPEC_PAIR
                                                .EXTRN  STR$AB_SPANISH_CLASS
                                                .EXTRN  STR$AB_SPANISH_CLASS_SPEC_PAIR
                                                .EXTRN  STR$ANALYZE_SDESC_R1

                                                .PSECT  _STR$CODE,NOWRT,  SHR,  PIC,2

                                 0FFC 00000     .ENTRY  STR$COMPARE_MULTI, Save R2,R3,R4,R5,R6,R7,-  ; 1639
                                                        R8,R9,R10,RT1                                 ;
             5E      FE90   CE  9E 00002        MOVAB   -368(SP), SP                                  ;
                            7E  7C 00007        CLRQ    SAME                                          ; 1691
      00AC   CE             20  90 00009        MOVB    #32, NULL_STRING                              ;
             50        04   AC  D0 0000E        MOVL    STRING1, R0                                   ; 1735
             02        03   A0  91 00012        CMPB    3(R0), #2                                     ;
                            0C  1A 00016        BGTRU   1$                                            ;
        30   AE             60  3C 00018        MOVZWL  (R0), STR1_LEN                                ;
      0088   CE        04   A0  D0 0001C        MOVL    4(R0), STRT_ADDR_SAV                          ;
                            0F  11 00022        BRB     2$                                            ;
```

```
                           00000000G  00  16 00024 1$:   JSB     STR$ANALYZE_SDESC_R1                         :
                30  AE                50  D0 0002A       MOVL    R0, 48(SP)                                   :
              0088  CE                51  D0 0002E       MOVL    R1, 136(SP)                                  :
                50        08  AC  D0 00033 2$:           MOVL    STRING2, R0                                  : 1736
                02        03  A0  91 00037               CMPB    3(R0), #2                                    :
                              0C  1A 0003B               BGTRU   3$                                           :
                1C  AE            60  3C 0003D           MOVZWL  (R0), STR2_LEN                               :
              0084  CE        04  A0  D0 00041           MOVL    4(R0), STR2_ADDR_SAV                         :
                              0F  11 00047               BRB     4$                                           :
                           00000000G  00  16 00049 3$:   JSB     STR$ANALYZE_SDESC_R1                         :
                1C  AE                50  D0 0004F       MOVL    R0, 28(SP)                                   :
              0084  CE                51  D0 00053       MOVL    R1, 132(SP)                                  :
                          30  AE  D5 00058 4$:           TSTL    STR1_LEN                                     : 1742
                              0B  12 0005B               BNEQ    5$                                           :
                30  AE            01  D0 0005D           MOVL    #1, STR1_LEN                                 : 1746
              0088  CE      00AC  CE  9E 00061           MOVAB   NULL_STRING, STR1_ADDR_SAV                   : 1747
                          1C  AE  D5 00068 5$:           TSTL    STR2_LEN                                     : 1751
                              0B  12 0006B               BNEQ    6$                                           :
                1C  AE            01  D0 0006D           MOVL    #1, STR2_LEN                                 : 1755
              0084  CE      00AC  CE  9E 00071           MOVAB   NULL_STRING, STR2_ADDR_SAV                   : 1756
                1C  AE        30  AE  D1 00078 6$:       CMPL    STR1_LEN, STR2_LEN                           : 1764
                              03  12 0007D               BNEQ    7$                                           :
                    6E            01  D0 0007F           MOVL    #1, SAME                                     : 1766
                    50        30  AE  D0 00082 7$:       MOVL    STR1_LEN, R0                                 : 1767
                1C  AE            50  D1 00086           CMPL    R0, STR2_LEN                                 :
                              04  1B 0008A               BLEQU   8$                                           :
                    50        1C  AE  D0 0008C           MOVL    STR2_LEN, R0                                 :
                70  AE            50  D0 00090 8$:       MOVL    R0, STR_END                                  :
                    03            6C  91 00094           CMPB    (AP), #3                                     : 1772
                              08  1F 00097               BLSSU   9$                                           :
                04  AC        0C  E9 00099               BLBC    CASE_BLIND_FLAG, 9$                          : 1774
                04  AE            01  D0 0009D           MOVL    #1, CASE_BLIND                               : 1776
                    56 00000000G  00  9E 000A1 9$:       MOVAB   STR$AB_MULTI, THAT_TABLE                     : 1783
                    5A 00000000G  00  9E 000A8           MOVAB   STR$AB_MULTI_SPEC_CHAR, SPEC_CHAR            : 1784
                    59 00000000G  00  9E 000AF           MOVAB   STR$AB_MULTI_SPEC_SEQ, SPEC_SEQ              : 1785
                38  AE 00000000G  00  9E 000B6           MOVAB   STR$AB_MULTI_CLASS, CLASS_TABLE              : 1786
              00A4  CE 00000000G  00  9E 000BE           MOVAB   STR$AB_MULTI_CLASS_SPEC_SEQ, CLASS_SPEC_SEQ  : 1787
                    04            6C  91 000C7           CMPB    (AP), #4                                     : 1789
                                  77  12 000CA           BNEQ    15$                                          :
         05         01        10  AC  CF 000CC           CASEL   FOREIGN_LANG, #1, #5                         : 1791
 002C  001B       003D      0072         000D1 10$:      .WORD   15$-10$,-                                    :
                  004E      003D         000D9                   13$-10$,-                                    :
                                                                 11$-10$,-                                    :
                                                                 12$-10$,-                                    :
                                                                 13$-10$,-                                    :
                                                                 14$-10$                                      :
                           00000000G  8F  DD 000DD       PUSHL   #LIB$_INVARG                                 : 1835
           00000000G  00              01  FB 000E3       CALLS   #1, LIB$SIGNAL                               :
                                  57  11 000EA           BRB     15$                                          :
                    56 00000000G  00  9E 000EC 11$:      MOVAB   STR$AB_FIN_SWED, THAT_TABLE                  : 1805
                38  AE 00000000G  00  9E 000F3           MOVAB   STR$AB_FIN_SWED_CLASS, CLASS_TABLE           : 1806
                                  46  11 000FB           BRB     15$                                          : 1791
                    56 00000000G  00  9E 000FD 12$:      MOVAB   STR$AB_GERMAN, THAT_TABLE                    : 1812
                38  AE 00000000G  00  9E 00104           MOVAB   STR$AB_GERMAN_CLASS, CLASS_TABLE             : 1813
                                  35  11 0010C           BRB     15$                                          : 1791
                    56 00000000G  00  9E 0010E 13$:      MOVAB   STR$AB_DAN_NOR, THAT_TABLE                   : 1819
                38  AE 00000000G  00  9E 00115           MOVAB   STR$AB_DAN_NOR_CLASS, CLASS_TABLE            : 1820
```

```
                                   24  11 0011D        BRB     15$                                      ; 1791
               56 00000000G        00  9E 0011F 14$:   MOVAB   STR$AB_SPANISH, THAT_TABLE               ; 1826
          00A8 CE                  20  D0 00126        MOVL    #32, PAIR_LEN                            ; 1827
               57 00000000G        00  9E 0012B        MOVAB   STR$AB_SPANISH_SPEC_PAIR, SPEC_PAIR      ; 1828
            38 AE 00000000G        00  9E 00132        MOVAB   STR$AB_SPANISH_CLASS, CLASS_TABLE        ; 1829
          00A0 CE 00000000G        00  9E 0013A        MOVAB   STR$AB_SPANISH_CLASS_SPEC_PAIR, -        ; 1830
                                                               CLASS_SPEC_PAIR
                              14   AE  D4 00143 15$:   CLRL    JJ                                       ; 2319
                              34   AE  D4 00146 16$:   CLRL    CALL_SPEC_LIST                           ; 1860
                              4C   AE  7C 00149        CLRQ    NO_PAIR
                            0080   CE  D4 0014C        CLRL    ARR_SAME                                 ; 1896
                              78   AE  7C 00150        CLRQ    CHARS_READ                               ; 1860
            28   AE         0084   CE  7D 00153        MOVQ    STR2_ADDR_SAV, STR2_ADDR                 ; 1899
            64   AE           2C   BE  90 00159        MOVB    @STR1_ADDR, CHAR1                        ; 1901
                              2C   AE  D6 0015E        INCL    STR1_ADDR
            60   AE           28   BE  90 00161        MOVB    @STR2_ADDR, CHAR2                        ; 1902
                              28   AE  D6 00166        INCL    STR2_ADDR
                            0090   CE  D4 00169        CLRL    144(SP)                                  ; 1904
                              14   AE  D5 0016D        TSTL    JJ
                              1C   12     00170        BNEQ    17$
                            0090   CE  D6 00172        INCL    144(SP)
          0094   CE                56  7D 00176        MOVQ    THAT_TABLE, SAVE_TABLE                   ; 1910
                 56           38   AE  D0 0017B        MOVL    CLASS_TABLE, THAT_TABLE                  ; 1911
          009C   CE                59  D0 0017F        MOVL    SPEC_SEQ, SAVE_SPEC_SEQ                  ; 1912
                 59         00A4   CE  D0 00184        MOVL    CLASS_SPEC_SEQ, SPEC_SEQ                 ; 1913
                 57         00A0   CE  D0 00189        MOVL    CLASS_SPEC_PAIR, SPEC_PAIR               ; 1915
                            008C   CE  D4 0018E 17$:   CLRL    140(SP)                                  ; 1919
                 01           14   AE  D1 00192        CMPL    JJ, #1
                              52   12     00196        BNEQ    23$
                            008C   CE  D6 00198        INCL    140(SP)
                 50           64   AE  90 0019C        MOVB    CHAR1, TEMP_BYTE                         ; 1925
            61   8F                50  91 001A0        CMPB    TEMP_BYTE, #97
                                   06  1F 001A4        BLSSU   18$
            7A   8F                50  91 001A6        CMPB    TEMP_BYTE, #122
                                   12  1B 001AA        BLEQU   19$
            E0   8F                50  91 001AC 18$:   CMPB    TEMP_BYTE, #224
                                   11  1F 001B0        BLSSU   20$
            FD   8F                50  91 001B2        CMPB    TEMP_BYTE, #253
                                   0B  1A 001B6        BGTRU   20$
            F0   8F                50  91 001B8        CMPB    TEMP_BYTE, #240
                                   05  13 001BC        BEQL    20$
   64   AE       50                20  83 001BE 19$:   SUBB3   #32, TEMP_BYTE, CHAR1
                 50           60   AE  90 001C3 20$:   MOVB    CHAR2, TEMP_BYTE                         ; 1926
            61   8F                50  91 001C7        CMPB    TEMP_BYTE, #97
                                   06  1F 001CB        BLSSU   21$
            7A   8F                50  91 001CD        CMPB    TEMP_BYTE, #122
                                   12  1B 001D1        BLEQU   22$
            E0   8F                50  91 001D3 21$:   CMPB    TEMP_BYTE, #224
                                   11  1F 001D7        BLSSU   23$
            FD   8F                50  91 001D9        CMPB    TEMP_BYTE, #253
                                   0B  1A 001DD        BGTRU   23$
            F0   8F                50  91 001DF        CMPB    TEMP_BYTE, #240
                                   05  13 001E3        BEQL    23$
   60   AE       50                20  83 001E5 22$:   SUBB3   #32, TEMP_BYTE, CHAR2
                              74   AE  D4 001EA 23$:   CLRL    J                                        ; 1939
                            0484   31     001ED        BRW     109$
            44   AE           64   AE  9A 001F0 24$:   MOVZBL  CHAR1, 68(SP)                            ;
```

```
                58   AE      44 BE46  90 001F5        MOVB    @68(SP)[THAT_TABLE], MULTI1              ;
                40   AE      60   AE  9A 001FB        MOVZBL  CHAR2, 64(SP)                            ; 1940
                54   AE      40 BE46  90 00200        MOVB    @64(SP)[THAT_TABLE], MULTI2              ;
                             78   AE  D6 00206        INCL    CHARS_READ                               ; 1941
                FD   8F      58   AE  91 00209        CMPB    MULTI1, #253                             ; 1963
                                  18  13 0020E        BEQL    25$                                      ;
                FD   8F      54   AE  91 00210        CMPB    MULTI2, #253                             ;
                                  11  13 00215        BEQL    25$                                      ;
                FC   8F      58   AE  91 00217        CMPB    MULTI1, #252                             ; 1964
                                  0A  13 0021C        BEQL    25$                                      ;
                FC   8F      54   AE  91 0021E        CMPB    MULTI2, #252                             ;
                                  03  13 00223        BEQL    25$                                      ;
                                03D8  31 00225        BRW     101$                                     ;
                7C   AE           01  D0 00228 25$:   MOVL    #1, USED_ARRAYS                          ; 1968
                24   AE           01  D0 0022C        MOVL    #1, I                                    ; 1995
                             68   AE  D4 00230 26$:   CLRL    104(SP)                                  ; 1976
                     01      24   AE  D1 00233        CMPL    I, #1                                    ;
                                  1C  12 00237        BNEQ    27$                                      ;
                             68   AE  D6 00239        INCL    104(SP)                                  ;
                5C   AE      2C   AE  D0 0023C        MOVL    STR1_ADDR, STRX_ADDR                     ; 1977
                48   AE      30   AE  D0 00241        MOVL    STR1_LEN, STRX_LEN                       ;
                0C   AE      58   AE  90 00246        MOVB    MULTI1, MULTIX                           ;
                     5B      44   AE  90 0024B        MOVB    68(SP), CHARX                            ;
                     54      9C   AD  9E 0024F        MOVAB   ARRAY1, ARRAYX                           ;
                                  18  11 00253        BRB     28$                                      ; 1976
                5C   AE      28   AE  D0 00255 27$:   MOVL    STR2_ADDR, STRX_ADDR                     ; 1979
                48   AE      1C   AE  D0 0025A        MOVL    STR2_LEN, STRX_LEN                       ;
                0C   AE      54   AE  90 0025F        MOVB    MULTI2, MULTIX                           ;
                     5B      40   AE  90 00264        MOVB    64(SP), CHARX                            ;
                     54    00B0   CE  9E 00268        MOVAB   ARRAY2, ARRAYX                           ;
                                  55  D4 0026D 28$:   CLRL    COUNT                                    ; 1982
                             4C   AE  D4 0026F        CLRL    NO_PAIR                                  ; 1983
       52       78   AE           01  C3 00272        SUBL3   #1, CHARS_READ, L                        ; 1995
                                02D4  31 00277        BRW     88$                                      ;
                FD   8F      0C   AE  91 0027A 29$:   CMPB    MULTIX, #253                             ; 1998
                                  31  12 0027F        BNEQ    34$                                      ;
                                  50  7C 00281        CLRQ    K                                        ; 2004
                   604A           5B  91 00283 30$:   CMPB    CHARX, (K)[SPEC_CHAR]                    ;
                                  11  12 00287        BNEQ    31$                                      ;
                   6544         6940  33 00289        CVTWB   (SPEC_SEQ)[K], (COUNT)[ARRAYX]           ;
                01 A544      01 A940  33 0028E        CVTWB   1(SPEC_SEQ)[K], 1(COUNT)[ARRAYX]         ;
                     51           01  D0 00295        MOVL    #1, FOUND                                ;
                                  04  11 00298        BRB     32$                                      ;
       E5            50           05  F3 0029A 31$:   AOBLEQ  #5, K, 30$                               ;
                                  51  D5 0029E 32$:   TSTL    FOUND                                    ;
                                  0D  12 002A0        BNEQ    33$                                      ;
                        00000000G 8F  DD 002A2        PUSHL   #LIB$_INVARG                             ;
          00000000G 00            01  FB 002A8        CALLS   #1, LIB$SIGNAL                           ;
                                0253  31 002AF 33$:   BRW     82$                                      ; 2010
                FC   8F      0C   AE  91 002B2 34$:   CMPB    MULTIX, #252                             ; 2020
                                  03  13 002B7        BEQL    35$                                      ;
                                024E  31 002B9        BRW     83$                                      ;
                48   AE           52  D1 002BC 35$:   CMPL    L, STRX_LEN                              ; 2045
                                  04  12 002C0        BNEQ    36$                                      ;
                4C   AE           01  D0 002C2        MOVL    #1, NO_PAIR                              ; 2047
                             10   AE  D4 002C6 36$:   CLRL    FOUND_FIRST                              ;
                             18   AE  D4 002C9        CLRL    FOUND_SECOND                             ;
```

```
        3C   AE    00A8  CE            04  C7 002CC       DIVL3   #4, PAIR_LEN, 60(SP)
                         58      3C    AE  D0 002D3       MOVL    60(SP), INDEX
                         51            01  CE 002D7       MNEGL   #1, R
                                     00BF  31 002DA       BRW     50$
                                 50    AE  D4 002DD 37$:  CLRL    AGAIN_PAIR_MACRO
                                 34    AE  D4 002E0       CLRL    CALL_SPEC_LIST
                                     6741  DF 002E3       PUSHAL  (SPEC_PAIR)[R]
                         9E            5B  91 002E6       CMPB    CHARX, @(SP)+
                                       7C  12 002E9       BNEQ    45$
                   10    AE            01  D0 002EB       MOVL    #1, FOUND_FIRST
                   20    AE            5B  9A 002EF       MOVZBL  CHARX, SAVE_FIRST_LETTER
                       6544       01 A741  F6 002F3       CVTLB   1(SPEC_PAIR)[R], (COUNT)[ARRAYX]
                                 4C    AE  D5 002F9       TSTL    NO_PAIR
                                       03  13 002FC       BEQL    38$
                                     0093  31 002FE       BRW     48$
                         5B      5C    BE  90 00301 38$:  MOVB    @STRX_ADDR, CHARX
                                 5C    AE  D6 00305       INCL    STRX_ADDR
                                       52  D6 00308       INCL    L
                         25    008C    CE  E9 0030A       BLBC    140(SP), 41$
                         50            5B  90 0030F       MOVB    CHARX, TEMP_BYTE
                   61    8F            50  91 00312       CMPB    TEMP_BYTE, #97
                                       06  1F 00316       BLSSU   39$
                   7A    8F            50  91 00318       CMPB    TEMP_BYTE, #122
                                       12  1B 0031C       BLEQU   40$
                   E0    8F            50  91 0031E 39$:  CMPB    TEMP_BYTE, #224
                                       10  1F 00322       BLSSU   41$
                   FD    8F            50  91 00324       CMPB    TEMP_BYTE, #253
                                       0A  1A 00328       BGTRU   41$
                   F0    8F            50  91 0032A       CMPB    TEMP_BYTE, #240
                                       04  13 0032E       BEQL    41$
             5B          50            20  83 00330 40$:  SUBB3   #32, TEMP_BYTE, CHARX
                         53            5B  9A 00334 41$:  MOVZBL  CHARX, R3
                         50      FF    A1  9E 00337       MOVAB   -1(R1), S
                                       23  11 0033B       BRB     43$
                                  02 A740  DF 0033D 42$:  PUSHAL  2(SPEC_PAIR)[S]
   53        9E          08            00  ED 00341       CMPZV   #0, #8, @(SP)+, R3
                                       18  12 00346       BNEQ    43$
                                     6740  DF 00348       PUSHAL  (SPEC_PAIR)[S]
20 AE        9E          08            00  ED 0034B       CMPZV   #0, #8, @(SP)+, SAVE_FIRST_LETTER
                                       0D  12 00351       BNEQ    43$
                   18    AE            01  D0 00353       MOVL    #1, FOUND_SECOND
                   01  A544       03 A740  F6 00357       CVTLB   3(SPEC_PAIR)[S], 1(COUNT)[ARRAYX]
                                       04  11 0035E       BRB     44$
             D9          50            58  F2 00360 43$:  AOBLSS  INDEX, S, 42$
                                 18    AE  D5 00364 44$:  TSTL    FOUND_SECOND
                                       2D  12 00367 45$:  BNEQ    49$
                   0C    AE          6346  90 00369       MOVB    (R3)[THAT_TABLE], MULTIX
                   FC    8F      0C    AE  91 0036E       CMPB    MULTIX, #252
                                       08  12 00373       BNEQ    46$
                                       55  D6 00375       INCL    COUNT
                   50    AE            01  D0 00377       MOVL    #1, AGAIN_PAIR_MACRO
                                       19  11 0037B       BRB     49$
                   FD    8F      0C    AE  91 0037D 46$:  CMPB    MULTIX, #253
                                       08  12 00382       BNEQ    47$
                                       55  D6 00384       INCL    COUNT
                   34    AE            01  D0 00386       MOVL    #1, CALL_SPEC_LIST
                                       0A  11 0038A       BRB     49$
```

```
          01 A544        0C    AE  90 0038C 47$:    MOVB    MULTIX, 1(COUNT)[ARRAYX]
                               02  11 00392         BRB     49$
                               55  D7 00394 48$:    DECL    COUNT
                    01     10  AE  D1 00396 49$:    CMPL    FOUND_FIRST, #1
                               09  13 0039A         BEQL    52$
           02       51         58  F2 0039C 50$:    AOBLSS  INDEX, R, 51$
                               03  11 003A0         BRB     52$
                             FF38  31 003A2 51$:    BRW     37$
                           10  AE  D5 003A5 52$:    TSTL    FOUND_FIRST
                               0D  12 003A8         BNEQ    53$
                        00000000G  8F  DD 003AA     PUSHL   #LIB$_INVARG
              00000000G 00     01  FB 003B0         CALLS   #1, LIB$SIGNAL
                    01     34  AE  D1 003B7 53$:    CMPL    CALL_SPEC_LIST, #1                        ; 2059
                               20  12 003BB         BNEQ    57$
                               50  7C 003BD         CLRQ    K                                         ; 2060
                 604A          5B  91 003BF 54$:    CMPB    CHARX, (K)[SPEC_CHAR]
                               03  12 003C3         BNEQ    55$
                             0125  31 003C5         BRW     79$
           F3       50         05  F3 003C8 55$:    AOBLEQ  #5, K, 54$
                               51  D5 003CC 56$:    TSTL    FOUND
                               0D  12 003CE         BNEQ    57$
                        00000000G  8F  DD 003D0     PUSHL   #LIB$_INVARG
              00000000G 00     01  FB 003D6         CALLS   #1, LIB$SIGNAL
                    01     50  AE  D1 003DD 57$:    CMPL    AGAIN_PAIR_MACRO, #1                      ; 2076
                               03  13 003E1         BEQL    58$
                             011F  31 003E3         BRW     82$
                               52  D6 003E6 58$:    INCL    L                                         ; 2092
               48   AE         52  D1 003E8         CMPL    L, STRX_LEN                               ; 2093
                               04  12 003EC         BNEQ    59$
               4C   AE         01  D0 003EE         MOVL    #1, NO_PAIR                               ; 2095
                           10  AE  D4 003F2 59$:    CLRL    FOUND_FIRST
                           18  AE  D4 003F5         CLRL    FOUND_SECOND
               58          3C  AE  D0 003F8         MOVL    60(SP), INDEX
               51              01  CE 003FC         MNEGL   #1, R
                             00BF  31 003FF         BRW     73$
                           50  AE  D4 00402 60$:    CLRL    AGAIN_PAIR_MACRO
                           34  AE  D4 00405         CLRL    CALL_SPEC_LIST
                             6741  DF 00408         PUSHAL  (SPEC_PAIR)[R]
               9E              5B  91 0040B         CMPB    CHARX, @(SP)+
                               7C  12 0040E         BNEQ    68$
               10   AE         01  D0 00410         MOVL    #1, FOUND_FIRST
               20   AE         5B  9A 00414         MOVZBL  CHARX, SAVE_FIRST_LETTER
                 6544      01 A741  F6 00418        CVTLB   1(SPEC_PAIR)[R], (COUNT)[ARRAYX]
                           4C  AE  D5 0041E         TSTL    NO_PAIR
                               03  13 00421         BEQL    61$
                             0093  31 00423         BRW     71$
               5B          5C  BE  90 00426 61$:    MOVB    @STRX_ADDR, CHARX
                           5C  AE  D6 0042A         INCL    STRX_ADDR
                               52  D6 0042D         INCL    L
               25        008C  CE  E9 0042F         BLBC    140(SP), 64$
               50              5B  90 00434         MOVB    CHARX, TEMP_BYTE
               61   8F         50  91 00437         CMPB    TEMP_BYTE, #97
                               06  1F 0043B         BLSSU   62$
               7A   8F         50  91 0043D         CMPB    TEMP_BYTE, #122
                               12  1B 00441         BLEQU   63$
               E0   8F         50  91 00443 62$:    CMPB    TEMP_BYTE, #224
                               10  1F 00447         BLSSU   64$
```

```
                 FD    8F              50  91 00449        CMPB    TEMP_BYTE, #253
                                       0A  1A 0044D        BGTRU   64$
                 F0    8F              50  91 0044F        CMPB    TEMP_BYTE, #240
                                       04  13 00453        BEQL    64$
            5B         50              20  83 00455 63$:   SUBB3   #32, TEMP_BYTE, CHARX
                       53              5B  9A 00459 64$:   MOVZBL  CHARX, R3
                       50          FF  A1  9E 0045C        MOVAB   -1(R1), S
                                       23  11 00460        BRB     66$
                               02 A740 DF 00462 65$:       PUSHAL  2(SPEC_PAIR)[S]
      53        9E     08              00  ED 00466        CMPZV   #0, #8, @(SP)+, R3
                                       18  12 0046B        BNEQ    66$
                                  6740 DF 0046D            PUSHAL  (SPEC_PAIR)[S]
 20   AE        9E     08              00  ED 00470        CMPZV   #0, #8, @(SP)+, SAVE_FIRST_LETTER
                                       0D  12 00476        BNEQ    66$
                 18    AE              01  D0 00478        MOVL    #1, FOUND_SECOND
                 01 A544       03 A740 F6 0047C            CVTLB   3(SPEC_PAIR)[S], 1(COUNT)[ARRAYX]
                                       04  11 00483        BRB     67$
            D9         50              58  F2 00485 66$:   AOBLSS  INDEX, S, 65$
                                   18  AE  D5 00489 67$:   TSTL    FOUND_SECOND
                                       2D  12 0048C 68$:   BNEQ    72$
                 0C    AE            6346  90 0048E        MOVB    (R3)[THAT_TABLE], MULTIX
                 FC    8F          0C  AE  91 00493        CMPB    MULTIX, #252
                                       08  12 00498        BNEQ    69$
                                       55  D6 0049A        INCL    COUNT
                 50    AE              01  D0 0049C        MOVL    #1, AGAIN_PAIR_MACRO
                                       19  11 004A0        BRB     72$
                 FD    8F          0C  AE  91 004A2 69$:   CMPB    MULTIX, #253
                                       08  12 004A7        BNEQ    70$
                                       55  D6 004A9        INCL    COUNT
                 34    AE              01  D0 004AB        MOVL    #1, CALL_SPEC_LIST
                                       0A  11 004AF        BRB     72$
                 01 A544           0C  AE  90 004B1 70$:   MOVB    MULTIX, 1(COUNT)[ARRAYX]
                                       02  11 004B7        BRB     72$
                                       55  D7 004B9 71$:   DECL    COUNT
                       01          10  AE  D1 004BB 72$:   CMPL    FOUND_FIRST, #1
                                       09  13 004BF        BEQL    75$
            02         51              58  F2 004C1 73$:   AOBLSS  INDEX, R, 74$
                                       03  11 004C5        BRB     75$
                                     FF38  31 004C7 74$:   BRW     60$
                                   10  AE  D5 004CA 75$:   TSTL    FOUND_FIRST
                                       0D  12 004CD        BNEQ    76$
                             00000000G 8F  DD 004CF        PUSHL   #LIB$_INVARG
               00000000G  00           01  FB 004D5        CALLS   #1, LIB$SIGNAL
                       01          34  AE  D1 004DC 76$:   CMPL    CALL_SPEC_LIST, #1                    ; 2101
                                       03  13 004E0        BEQL    77$
                                     FEF8  31 004E2        BRW     57$
                                       50  7C 004E5 77$:   CLRQ    K                                     ; 2102
                    604A               5B  91 004E7 78$:   CMPB    CHARX, (K)[SPEC_CHAR]
                                       11  12 004EB        BNEQ    80$
                    6544             6940  33 004ED 79$:   CVTWB   (SPEC_SEQ)[K], (COUNT)[ARRAYX]
                 01 A544       01 A940 33 004F2            CVTWB   1(SPEC_SEQ)[K], 1(COUNT)[ARRAYX]
                       51              01  D0 004F9        MOVL    #1, FOUND
                                       04  11 004FC        BRB     81$
            E5         50              05  F3 004FE 80$:   AOBLEQ  #5, K, 78$
                                     FEC7  31 00502 81$:   BRW     56$
                       55              02  C0 00505 82$:   ADDL2   #2, COUNT                             ; 2110
                                       05  11 00508        BRB     84$                                   ; 2020
```

```
                         8544      0C   AE  90 0050A 83$:   MOVB    MULTIX, (COUNT)+[ARRAYX]                  ; 2123
                      48   AE           52  D1 0050F 84$:   CMPL    L, STRX_LEN                               ; 2134
                                        39  13 00513        BEQL    88$
                      5B        5C      BE  90 00515        MOVB    @STRX_ADDR, CHARX                         ; 2138
                                5C      AE  D6 00519        INCL    STRX_ADDR
                      25      008C      CE  E9 0051C        BLBC    140(SP), 87$                              ; 2139
                      50                5B  90 00521        MOVB    CHARX, TEMP_BYTE                          ; 2141
                 61   8F                50  91 00524        CMPB    TEMP_BYTE, #97
                                        06  1F 00528        BLSSU   85$
                 7A   8F                50  91 0052A        CMPB    TEMP_BYTE, #122
                                        12  1B 0052E        BLEQU   86$
                 E0   8F                50  91 00530 85$:   CMPB    TEMP_BYTE, #224
                                        10  1F 00534        BLSSU   87$
                 FD   8F                50  91 00536        CMPB    TEMP_BYTE, #253
                                        0A  1A 0053A        BGTRU   87$
                 F0   8F                50  91 0053C        CMPB    TEMP_BYTE, #240
                                        04  13 00540        BEQL    87$
             5B       50                20  83 00542 86$:   SUBB3   #32, TEMP_BYTE, CHARX
                      50                5B  9A 00546 87$:   MOVZBL  CHARX, R0                                 ; 2143
             0C       AE              6046  90 00549        MOVB    (R0)[THAT_TABLE], MULTIX
   FD25      52       01        48      AE  F1 0054E 88$:   ACBL    STRX_LEN, #1, L, 29$                      ; 1995
                      04        68      AE  E9 00555        BLBC    104(SP), 89$                              ; 2152
                      08   AE           55  D0 00559        MOVL    COUNT, COUNT1                             ; 2154
   FCCC      24  AE   01                02  F1 0055D 89$:   ACBL    #2, #1, I, 26$                            ; 1970
                      55        08      AE  D1 00564        CMPL    COUNT1, COUNT                             ; 2165
                      05                12     00568        BNEQ    90$
               0080   CE                01  D0 0056A        MOVL    #1, ARR_SAME                              ; 2167
                      50        08      AE  D0 0056F 90$:   MOVL    COUNT1, R0                                ; 2168
                      55                50  D1 00573        CMPL    R0, COUNT
                      03                1B     00576        BLEQU   91$
                      50                55  D0 00578        MOVL    COUNT, R0
                 6C   AE                50  D0 0057B 91$:   MOVL    R0, ARR_MIN
                      50                01  CE 0057F        MNEGL   #1, M                                     ; 2170
                                        18  11 00582        BRB     95$
          00B0 CE40          9C AD40    91 00584 92$:   CMPB    ARRAY1[M], ARRAY2[M]                      ; 2173
                                        0E  13 0058C        BEQL    95$
                                        05  1E 0058E        BGEQU   93$                                       ; 2178
                      51                01  CE 00590        MNEGL   #1, R1                                    ; 2182
                                        03  11 00593        BRB     94$
                      51                01  D0 00595 93$:   MOVL    #1, R1
                      50                51  D0 00598 94$:   MOVL    R1, R0
                                            04 0059B        RET
             E3       50        6C      AE  F2 0059C 95$:   AOBLSS  ARR_MIN, M, 92$                           ; 2170
                              0080      CE  D5 005A1        TSTL    ARR_SAME                                  ; 2194
                                        03  13 005A5        BEQL    96$
                                      00CA  31 005A7        BRW     109$
                      55        08      AE  D1 005AA 96$:   CMPL    COUNT1, COUNT                             ; 2201
                                        25  18 005AE        BGEQ    98$
             50       55        6C      AE  C3 005B0        SUBL3   ARR_MIN, COUNT, R0                        ; 2205
                      58                01  D0 005B5        MOVL    #1, R8
                              00B0      CE  DF 005B8        PUSHAL  ARRAY2
                      6E        0C      AE  C0 005BC        ADDL2   COUNT1, (SP)
    50       20 00000000 9F             00  2D 005C0        CMPC5   #0, @#^X00000000, #32, R0, @(SP)+
                                        9E     005C9
                                        03  1A 005CA        BGTRU   97$
                      58                01  D9 005CC        SBWC    #1, R8
                      50                58  D0 005CF 97$:   MOVL    R8, COMP_VAL
```

```
                                 23   13 005D2          BEQL    100$                                          ; 2208
                                      04 005D4          RET                                                   ; 2212
            50      08   AE   6C AE   C3 005D5  98$:    SUBL3   ARR_MIN, COUNT1, R0                           ; 2226
                         58      01   D0 005DB          MOVL    #1, R8
    50          20 00000000 9F   00   2D 005DE          CMPC5   #0, @#^X00000000, #32, R0, ARRAY1[COUNT]
                              9C AD45    005E7
                                 03   1A 005EA          BGTRU   99$
                         58      01   D9 005EC          SBWC    #1, R8
                         50      58   D0 005EF  99$:    MOVL    R8, COMP_VAL
                                 03   13 005F2          BEQL    100$                                          ; 2229
                               010A   31 005F4          BRW     118$
                    02   14      AE   D1 005F7  100$:   CMPL    JJ, #2                                        ; 2233
                                 7F   12 005FB          BNEQ    110$
                               011B   31 005FD          BRW     121$                                          ; 2235
                54  AE   58      AE   91 00600  101$:   CMPB    MULTI1, MULTI2                                ; 2259
                                 0A   13 00605          BEQL    103$
                                 04   1B 00607          BLEQU   102$                                          ; 2264
                         50      01   D0 00609          MOVL    #1, R0                                        ; 2268
                                      04 0060C          RET
                         50      01   CE 0060D  102$:   MNEGL   #1, R0
                                      04 00610          RET
                64  AE   2C      BE   90 00611  103$:   MOVB    @STR1_ADDR, CHAR1                             ; 2273
                         2C      AE   D6 00616          INCL    STR1_ADDR
                60  AE   28      BE   90 00619          MOVB    @STR2_ADDR, CHAR2                             ; 2274
                         28      AE   D6 0061E          INCL    STR2_ADDR
                    4E 008C      CE   E9 00621          BLBC    140(SP), 109$                                 ; 2275
                    50   64      AE   90 00626          MOVB    CHAR1, TEMP_BYTE                              ; 2281
                61  8F           50   91 0062A          CMPB    TEMP_BYTE, #97
                                 06   1F 0062E          BLSSU   104$
                7A  8F           50   91 00630          CMPB    TEMP_BYTE, #122
                                 12   1B 00634          BLEQU   105$
                E0  8F           50   91 00636  104$:   CMPB    TEMP_BYTE, #224
                                 11   1F 0063A          BLSSU   106$
                FD  8F           50   91 0063C          CMPB    TEMP_BYTE, #253
                                 0B   1A 00640          BGTRU   106$
                F0  8F           50   91 00642          CMPB    TEMP_BYTE, #240
                                 05   13 00646          BEQL    106$
          64  AE             50  20   83 00648  105$:   SUBB3   #32, TEMP_BYTE, CHAR1
                         50  60      AE   90 0064D  106$:   MOVB    CHAR2, TEMP_BYTE                          ; 2282
                61  8F           50   91 00651          CMPB    TEMP_BYTE, #97
                                 06   1F 00655          BLSSU   107$
                7A  8F           50   91 00657          CMPB    TEMP_BYTE, #122
                                 12   1B 0065B          BLEQU   108$
                E0  8F           50   91 0065D  107$:   CMPB    TEMP_BYTE, #224
                                 11   1F 00661          BLSSU   109$
                FD  8F           50   91 00663          CMPB    TEMP_BYTE, #253
                                 0B   1A 00667          BGTRU   109$
                F0  8F           50   91 00669          CMPB    TEMP_BYTE, #240
                                 05   13 0066D          BEQL    109$
          60  AE             50  20   83 0066F  108$:   SUBB3   #32, TEMP_BYTE, CHAR2
   FB74   74  AE         01  70  AE   F1 00674  109$:   ACBL    STR_END, #1, J, 24$                           ; 1933
                                 55   D4 0067C  110$:   CLRL    R5                                            ; 2299
                    01   04      AE   D1 0067E          CMPL    CASE_BLIND, #1
                                 0C   12 00682          BNEQ    111$
                                 55   D6 00684          INCL    R5
                         14      AE   D5 00686          TSTL    JJ
                                 05   13 00689          BEQL    111$
```

```
                                01              6E  D1 0068B           CMPL    SAME, #1                           ; 2301
                                                6F  13 0068E           BEQL    117$                               ;
                                                54  D4 00690 111$:     CLRL    R4                                 ; 2310
                                02        14    AE  D1 00692           CMPL    JJ, #2                             ;
                                                07  12 00696           BNEQ    112$                               ;
                                                54  D6 00698           INCL    R4                                 ;
                                01              6E  D1 0069A           CMPL    SAME, #1                           ;
                                                7C  13 0069D           BEQL    121$                               ;
                                                6E  D5 0069F 112$:     TSTL    SAME                               ; 2319
                                                62  12 006A1           BNEQ    119$                               ;
                                          7C    AE  D5 006A3           TSTL    USED_ARRAYS                        ;
                                                5D  12 006A6           BNEQ    119$                               ;
                           1C   AE        30    AE  D1 006A8           CMPL    STR1_LEN, STR2_LEN                 ; 2326
                                                23  1E 006AD           BGEQU   114$                               ;
                 50        1C   AE        70    AE  C3 006AF           SUBL3   STR_END, STR2_LEN, R0              ; 2330
                                5B              01  D0 006B5           MOVL    #1, R11                            ;
                 58        28   AE              01  C3 006B8           SUBL3   #1, STR2_ADDR, R8                  ;
      50         20 00000000    9F              00  2D 006BD           CMPC5   #0, @#^X00000000, #32, R0, (R8)    ;
                                                68     006C6                                                      ;
                                                03  1A 006C7           BGTRU   113$                               ;
                                5B              01  D9 006C9           SBWC    #1, R11                            ;
                                50              5B  D0 006CC 113$:     MOVL    R11, COMP_VAL                      ;
                                                23  13 006CF           BEQL    116$                               ; 2332
                                                    04 006D1           RET                                        ; 2336
                 50        30   AE        70    AE  C3 006D2 114$:     SUBL3   STR_END, STR1_LEN, R0              ; 2348
                                5B              01  D0 006D8           MOVL    #1, R11                            ;
                 58        2C   AE              01  C3 006DB           SUBL3   #1, STR1_ADDR, R8                  ;
      50         20 00000000    9F              00  2D 006E0           CMPC5   #0, @#^X00000000, #32, R0, (R8)    ;
                                                68     006E9                                                      ;
                                                03  1A 006EA           BGTRU   115$                               ;
                                5B              01  D9 006EC           SBWC    #1, R11                            ;
                                50              5B  D0 006EF 115$:     MOVL    R11, COMP_VAL                      ;
                                                0D  12 006F2           BNEQ    118$                               ; 2350
                                24              54  E8 006F4 116$:     BLBS    R4, 121$                           ; 2354
                                0B              55  E9 006F7           BLBC    R5, 119$                           ;
                                          14    AE  D5 006FA           TSTL    JJ                                 ;
                                                06  13 006FD           BEQL    119$                               ;
                                                1A  11 006FF 117$:     BRB     121$                               ; 2356
                                50              50  CE 00701 118$:     MNEGL   COMP_VAL, R0                       ; 2360
                                                    04 00704           RET                                        ;
                                0A      0090    CE  E9 00705 119$:     BLBC    144(SP), 120$                      ; 2369
                                59      009C    CE  D0 0070A           MOVL    SAVE_SPEC_SEQ, SPEC_SEQ            ; 2374
                                56      0094    CE  7D 0070F           MOVQ    SAVE_TABLE, THAT_TABLE             ; 2373
          FA2B        14    AE          01          02  F1 00714 120$:     ACBL    #2, #1, JJ, 16$                    ; 1859
                                                50  D4 0071B 121$:     CLRL    R0                                 ; 2382
                                                    04 0071D           RET                                        ;

; Routine Size:  1822 bytes,    Routine Base:  _STR$CODE + 0000


; 1194          2383  1
; 1195          2384  1        END                                   ! end of module
; 1196          2385  0        ELUDOM
```

```
;                          PSECT SUMMARY
;
;       Name                 Bytes                         Attributes
;
;   _STR$CODE                    1822  NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(2)


;                     Library Statistics
;
;                                           -------- Symbols --------        Pages      Processing
;       File                                Total   Loaded   Percent        Mapped     Time
;
;   _$255$DUA28:[SYSLIB]STARLET.L32;1        9776       4        0           581        00:00.7



;                            COMMAND QUALIFIERS

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

; Size:          1822 code + 0 data bytes
; Run Time:         00:33.6
; Elapsed Time:     02:09.7
; Lines/CPU Min:     4264
; Lexemes/CPU-Min: 15775
; Memory Used:  591 pages
; Compilation Complete
```

0214 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

STRDUPLCH
LIS

STRCOMPAR
LIS

STRFINDSB
LIS

STRMATCH
LIS

STRMSG
LIS

STRLENEXT
LIS

STRCOPY
LIS

STRFINDFI
LIS

STRLEFT
LIS

STRMULTI
LIS

STRCOMEQL
LIS

STRCONCAT
LIS

STRMOVQ
LIS

STRGETFRE
LIS

LINKER
LIS

STRPOSIT
LIS

STRUNWDEQ
LIS

STRPOSEXT
LIS

STRREPLAC
LIS

STRSRCHIN
LIS

STRRIGHT
LIS

LINKER

STRTRIM
LIS

LINK
MAP

PREFIX
REQ

ISDSORT
LIS

STRUPCASE
LIS

STRTRANSL
LIS

DATBAS
MDL

TIRAUX
REQ

ISGENC
REQ

STRPREFIX
LIS

DATBAS
LIS