```
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        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
```

```
  000000    TTTTTTTTTT    SSSSSSSS    SSSSSSSS    CCCCCCCC    000000    PPPPPPPP    YY      YY
  000000    TTTTTTTTTT    SSSSSSSS    SSSSSSSS    CCCCCCCC    000000    PPPPPPPP    YY      YY
CO      OO      TT      SS          SS          CC          OO      OO  PP      PP  YY      YY
OO      OO      TT      SS          SS          CC          OO      OO  PP      PP  YY      YY
OO      OO      TT      SS          SS          CC          OO      OO  PP      PP    YY  YY
OO      OO      TT      SS          SS          CC          OO      OO  PP      PP    YY  YY
OO      OO      TT        SSSSSS      SSSSSS    CC          OO      OO  PPPPPPPP        YY
OO      OO      TT        SSSSSS      SSSSSS    CC          OO      OO  PPPPPPPP        YY
OO      OO      TT              SS          SS  CC          OO      OO  PP              YY
OO      OO      TT              SS          SS  CC          OO      OO  PP              YY
OO      OO      TT              SS          SS  CC          OO      OO  PP              YY          ....
OO      OO      TT              SS          SS  CC          OO      OO  PP              YY          ....
  000000        TT      SSSSSSSS    SSSSSSSS      CCCCCCCC    000000    PP              YY          ....
  000000        TT      SSSSSSSS    SSSSSSSS      CCCCCCCC    000000    PP              YY          ....

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
```

Table of contents

```
0000     1          .TITLE  OTS$SCOPY - String copying module
0000     2          .IDENT  /1-011/         ; File: OTSSCOPY.MAR Edit: SBL1011
0000     3
0000     4 ;
0000     5 ;*****************************************************************************
0000     6 ;*                                                                           *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9 ;*  ALL RIGHTS RESERVED.                                                     *
0000    10 ;*                                                                           *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    16 ;*  TRANSFERRED.                                                             *
0000    17 ;*                                                                           *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    19 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    20 ;*  CORPORATION.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24 ;*                                                                           *
0000    25 ;*                                                                           *
0000    26 ;*****************************************************************************
0000    27 ;
0000    28
0000    29 ; FACILITY: Language-independent support: string handling
0000    30 ;
0000    31 ; ABSTRACT:
0000    32 ;
0000    33 ;       This module contains routines to allocate and deallocate
0000    34 ;       strings.  These entry points were in VMS release 1, before
0000    35 ;       there was a separate string facility, and they are being
0000    36 ;       retained for compatabliity.  They are implemented by calling
0000    37 ;       LIB$SGET1_DD_R6, LIB$SFREE1_DD6 and LIB$SFREEN_DD6.
0000    38 ;       This module also contains the routines to do string copying
0000    39 ;       using OTS$ semantics.  They are implemented by calling
0000    40 ;       LIB$SCOPY_DXDX6 and LIB$SCOPY_R_DX6.
0000    41 ;
0000    42 ;
0000    43 ; ENVIRONMENT: VAX-11 User Mode
0000    44 ;
0000    45 ; AUTHOR: R. Reichert, CREATION DATE: 3-APR-1981
0000    46 ;
0000    47 ; MODIFIED BY:
0000    48 ;
0000    49 ; 1-001 - Original.  Based on Version 1-007 of OTSSCOPY.B32.  With the
0000    50 ;         addition of the code to accomodate additional classes of
0000    51 ;         descriptors, necessitating a call to LIB$ANALYZE_SDESC_R3
0000    52 ;         it became increasingly difficult to control the register
0000    53 ;         usage in OTS$SCOPY_DXDX6 and OTS$SCOPY_R_DX6.
0000    54 ;         (In fact the original .B32 didn't control them correctly.)
0000    55 ;         RKR 3-APR-1981
0000    56 ; 1-002 - Revise which error statuses get turned into signals in
0000    57 ;         CHECK_FOR_FATAL.  RKR 3-SEP-1981
```

```
0000     58 ; 1-008 - Original OTSSCOPY.B32 had a revision history that ran up
0000     59 ;         through 1-007.  To avoid confusion with module idents that
0000     60 ;         are out in the field, this module's ident must be at
0000     61 ;         least 1-008.  RKR 14-SEP-1981
0000     62 ; 1-009 - Add special-case code to process string descriptors that
0000     63 ;         "read" like fixed string descriptors.  RKR 7-OCT-1981.
0000     64 ; 1-010 - Redirect jsb's from LIB$ANALYZE_SDESC_R3 to
0000     65 ;         LIB$ANALYZE_SDESC_R2.  RKR 18-NOV-1981.
0000     66 ; 1-011 - Use general mode addressing.  SBL 30-Nov-1981
0000     67 ;
0000     68 ;--
```

```
0000     70          .SBTTL  DECLARATIONS
0000     71 ;
0000     72 ; LIBRARY MACRO CALLS:
0000     73 ;
0000     74          $SSDEF                  ; SS$_ symbols
0000     75          $DSCDEF                 ; DSC$ symbols
0000     76 ;
0000     77 ; EXTERNAL DECLARATIONS:
0000     78 ;
0000     79 ; Prevent undeclared symbols from being automatically global.
0000     80 ;
0000     81          .DSABL  GBL
0000     82 ; The condition codes and signals we deal with
0000     83
0000     84          .EXTRN  OTS$_FATINTERR          ; Fatal internal error
0000     85          .EXTRN  LIB$_FATERRLIB          ;
0000     86
0000     87          .EXTRN  OTS$_INVSTRDES          ; Invalid string descriptor
0000     88          .EXTRN  LIB$_INVSTRDES          ;
0000     89
0000     90          .EXTRN  OTS$_INSVIRMEM          ; Insufficient virtual memory
0000     91          .EXTRN  LIB$_INSVIRMEM          ;
0000     92
0000     93          .EXTRN  LIB$_INVARG             ; Invalid argument
0000     94
0000     95          .EXTRN  LIB$_WRONUMARG          ; Wrong number of arguments
0000     96          .EXTRN  OTS$_WRONUMARG          ;
0000     97
0000     98 ; The external routines we use
0000     99
0000    100          .EXTRN  LIB$STOP                ; Signal a fatal error
0000    101          .EXTRN  LIB$SGET1_DD_R6         ; Alloc. string by descr
0000    102          .EXTRN  LIB$SFREE1_DD6          ; Free 1 by descr.
0000    103          .EXTRN  LIB$SFREEN_DD6          ; Free N by descr.
0000    104          .EXTRN  LIB$SCOPY_DXDX6         ; Copy string by descr
0000    105          .EXTRN  LIB$SCOPY_R_DX6         ; Copy string by ref.
0000    106          .EXTRN  LIB$ANALYZE_SDESC_R2    ; Analyze desc to get length and
0000    107                                          ; address of data
0000    108 ;
0000    109 ; MACROS:
0000    110 ;
0000    111          .MACRO  SIGNAL_FATAL_ERR ?L1
0000    112 ;+
0000    113 ; This macro checks to see if the current status in R0 is a success.
0000    114 ; If so, it continues.  If it is not a success, it branches to
0000    115 ; CHECK_FOR_FATAL_ERROR for a closer look at the error code.  If it
0000    116 ; it is found to be one of a set of fatal errors of interest, the
0000    117 ; corresponding OTS$ error is signaled.  Else the supplied error code
0000    118 ; is signaled.
0000    119
0000    120          BLBS    R0, L1             ; If success code, bypass checks
0000    121          BRW     CHECK_FOR_FATAL_ERROR   ; see if it is one of interest
0000    122 L1:
0000    123          .ENDM   SIGNAL_FATAL_ERR
0000    124
0000    125 ;
0000    126 ; EQUATED SYMBOLS:
```

```
                        ;
                        ;       NONE
                        ;
                        ; OWN STORAGE:
                        ;
00000000                        .PSECT _OTS$DATA PIC, USR, CON, REL, LCL, NOSHR, -
                                       NOEXE, RD, WRT, LONG
                        ;
                        ;       NONE
                        ;
                        ; PSECT DECLARATIONS:
                        ;
                                .PSECT _OTS$CODE PIC, USR, CON, REL, LCL, SHR, -
00000000                               EXE, RD, NOWRT, LONG
```

```
                        .SBTTL  OTS$SGET1_DD    Allocate a dynamic string
                ;++
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       Allocate a string.  LEN bytes are allocated to DESCRIP, which
                ;       is assumed to be a dynamic descriptor.  If the descriptor
                ;       already has storage allocated to it, but not enough, the old
                ;       storage is deallocated.
                ;
                ; CALLING SEQUENCE:
                ;
                ;       status.wlc.v = OTS$SGET1_DD (LEN.rwu.v, DESCRIP.wqu.r)
                ;
                ; FORMAL PARAMETERS:
                ;
                ;       LEN.rwu.v       The number of bytes to allocate.
                ;       DESCRIP.wqu.r   The descriptor.  The DSC$B_DTYPE field is not
                ;                       touched.
                ;
                ; IMPLICIT INPUTS:
                ;
                ;       NONE
                ;
                ; IMPLICIT OUTPUTS:
                ;
                ;       NONE
                ;
                ; ROUTINE VALUE:
                ; COMPLETION CODES:
                ;
                ;       NONE
                ;
                ; SIDE EFFECTS:
                ;
                ;       May deallocate the descriptor's storage and allocate new
                ;       storage for it.
                ;       May signal OTS$_INSVIRMEM, OTS$_FATINTERR
                ;
                ;--

                ; Displacements from AP

00000004        LEN     =       4
00000008        DESCRIP =       8

                        .ENTRY  OTS$SGET1_DD, ^M<R2,R3,R4,R5,R6>        ; Entry point
50   04 AC   3C         MOVZWL  LEN(AP), R0                     ; length desired to R0
51   08 AC   D0         MOVL    DESCRIP(AP), R1                 ; descriptor address to R1
00000000'GF  16         JSB     G^LIB$SGET1_DD_R6               ; go allocate
                        SIGNAL_FATAL_ERR                        ; signal if a fatal error
             04         RET                                     ; to caller
```

```
                        0017   195          .SBTTL  OTS$SGET1_DD_R6 Allocate a synamic string
                        0017   196 ;++
                        0017   197 ; FUNCTIONAL DESCRIPTION:
                        0017   198 ;
                        0017   199 ;       Allocate a string.  LEN bytes are allocated to DESCRIP, which
                        0017   200 ;       is assumed to be a dynamic descriptor.  If the descriptor
                        0017   201 ;       already has storage allocated to it, but not enough, the old
                        0017   202 ;       storage is deallocated.
                        0017   203 ;
                        0017   204 ; CALLING SEQUENCE
                        0017   205 ;
                        0017   206 ;       status.wlc.v = JSB OTS$SGET1_DD_R6 (LEN.rwu.v, DESCRIP.wqu.r)
                        0017   207 ;
                        0017   208 ; FORMAL PARAMETERS:
                        0017   209 ;
                        0017   210 ;       LEN.rwu.v       In R0, the number of bytes to allocate.
                        0017   211 ;       DESCRIP.wqu.r   In R1, The descriptor.  The DSC$B_DTYPE field
                        0017   212 ;                       is not touched.
                        0017   213 ;
                        0017   214 ; IMPLICIT INPUTS:
                        0017   215 ;
                        0017   216 ;       NONE
                        0017   217 ;
                        0017   218 ; IMPLICIT OUTPUTS:
                        0017   219 ;
                        0017   220 ;       NONE
                        0017   221 ;
                        0017   222 ; ROUTINE VALUE:
                        0017   223 ; COMPLETION CODES:
                        0017   224 ;
                        0017   225 ;       NONE
                        0017   226 ;
                        0017   227 ; SIDE EFFECTS:
                        0017   228 ;
                        0017   229 ;       May deallocate the descriptor's storage and allocate new
                        0017   230 ;       storage for it.
                        0017   231 ;       May signal OTS$_INSVIRMEM or  OTS$_FATINTERR
                        0017   232 ;
                        0017   233 ;--
                        0017   234
                        0017   235 OTS$SGET1_DD_R6::
                        0017   236
      50    50    3C    0017   237          MOVZWL  R0, R0                  ; extract words worth of length
                        001A   238
                        001A   239                                          ; R1 already contains address of
                        001A   240                                          ; descriptor
                        001A   241
00000000'GF     16      001A   242          JSB     G^LIB$SGET1_DD_R6       ; go allocate
                        0020   243          SIGNAL_FATAL_ERR                ; signal error if a fatal one
                  05    0026   244          RSB                             ; return to our caller
```

```
                                                                .SBTTL  OTS$SFREE1_DD Deallocate a dynamic string
                                                        ;++
                                                        ; FUNCTIONAL DESCRIPTION:
                                                        ;
                                                        ;       Deallocate a string.  The string is assumed to be dynamic.
                                                        ;       If it isn't, LIB$SFREE1_DD6 will take care of it.
                                                        ;
                                                        ; CALLING SEQUENCE:
                                                        ;
                                                        ;       status.wlc.v = OTS$SFREE1_DD (DESCRIP.wqu.r)
                                                        ;
                                                        ; FORMAL PARAMETERS:
                                                        ;
                                                        ;       DESCRIP.wqu.r   The descriptor of the string to deallocate.
                                                        ;
                                                        ; IMPLICIT INPUTS:
                                                        ;
                                                        ;       NONE
                                                        ;
                                                        ; IMPLICIT OUTPUTS:
                                                        ;
                                                        ;       NONE
                                                        ;
                                                        ; ROUTINE VALUE:
                                                        ; COMPLETION CODES:
                                                        ;
                                                        ;       NONE
                                                        ;
                                                        ; SIDE EFFECTS:
                                                        ;
                                                        ;       May deallocate virtual storage.
                                                        ;       May signal OTS$_FATINTERR
                                                        ;
                                                        ;--
                                                        ;
                                                        ; Displacements from AP
                                                        ;
                                  00000004              DESCRIP =       4
                                                        ;
                                  007C                  .ENTRY  OTS$SFREE1_DD, ^M<R2,R3,R4,R5,R6>       ; Entry point
                       50   04 AC    D0                 MOVL    DESCRIP(AP), R0         ; address of descriptor to R0
                  00000000'GF    16                     JSB     G^LIB$SFREE1_DD6        ; go free string
                                                        SIGNAL_FATAL_ERR                ; signal if error is fatal
                                  04                    RET                             ; to caller
```

```
                     003A   291          .SBTTL  OTS$SFREE1_DD6 Deallocate a dynamic string
                     003A   292 ;++
                     003A   293 ; FUNCTIONAL DESCRIPTION:
                     003A   294 ;
                     003A   295 ;       Deallocate a string.  The string is assumed to be dynamic.
                     003A   296 ;       If it isn't, LIB$SFREE1_DD6 will take care of it.
                     003A   297 ;
                     003A   298 ; CALLING SEQUENCE:
                     003A   299 ;
                     003A   300 ;       status.wlc.v = JSB OTS$SFREE1_DD6 (DESCRIP.wqu.r)
                     003A   301 ;
                     003A   302 ; FORMAL PARAMETERS:
                     003A   303 ;
                     003A   304 ;       DESCRIP.wqu.r   In R0, the descriptor of the string to
                     003A   305 ;                       deallocate.
                     003A   306 ;
                     003A   307 ; IMPLICIT INPUTS:
                     003A   308 ;
                     003A   309 ;       NONE
                     003A   310 ;
                     003A   311 ; IMPLICIT OUTPUTS:
                     003A   312 ;
                     003A   313 ;       NONE
                     003A   314 ;
                     003A   315 ; ROUTINE VALUE:
                     003A   316 ; COMPLETION CODES:
                     003A   317 ;
                     003A   318 ;       NONE
                     003A   319 ;
                     003A   320 ; SIDE EFFECTS:
                     003A   321 ;
                     003A   322 ;       May deallocate virtual storage.
                     003A   323 ;       May signal OTS$_FATINTERR
                     003A   324 ;
                     003A   325 ;--
                     003A   326
                     003A   327 OTS$SFREE1_DD6::
                     003A   328                                          ; R0 already contains address of
                     003A   329                                          ; descriptor to be freed
00000000'GF    16    003A   330          JSB     G^LIB$SFREE1_DD6        ; go free string
                     0040   331          SIGNAL_FATAL_ERR                ; check for fatal error
               05    0046   332          RSB                             ; return to our caller
```

```
                     0047   334          .SBTTL  OTS$SFREEN_DD Deallocate N Dynamic Strings
                     0047   335 ;++
                     0047   336 ; FUNCTIONAL DESCRIPTION:
                     0047   337 ;
                     0047   338 ;       Deallocate a number of strings.  The strings are all assumed
                     0047   339 ;       to be dynamic.  If not, LIB$FREE1_DD6 will eventually take care
                     0047   340 ;       of them.
                     0047   341 ;
                     0047   342 ; CALLING SEQUENCE:
                     0047   343 ;
                     0047   344 ;       status.wlc.v = OTS$SFREEN_DD (NUM_DESC.rwu.v, DESC_PTR.wqu.r)
                     0047   345 ;
                     0047   346 ; FORMAL PARAMETERS:
                     0047   347 ;
                     0047   348 ;       NUM_DESC.rwu.v  The number of descriptors to deallocate.
                     0047   349 ;       DESC_PTR.wqu.r  The first of these descriptors.
                     0047   350 ;
                     0047   351 ; IMPLICIT INPUTS:
                     0047   352 ;
                     0047   353 ;       NONE
                     0047   354 ;
                     0047   355 ; IMPLICIT OUTPUTS:
                     0047   356 ;
                     0047   357 ;       NONE
                     0047   358 ;
                     0047   359 ; ROUTINE VALUE:
                     0047   360 ; COMPLETION CODES:
                     0047   361 ;
                     0047   362 ;       SS$_NORMAL
                     0047   363 ;
                     0047   364 ; SIDE EFFECTS:
                     0047   365 ;
                     0047   366 ;       May deallocate virtual storage.
                     0047   367 ;
                     0047   368 ;--
                     0047   369
                     0047   370
                     0047   371 ; Displacements from AP
                     0047   372
           00000004  0047   373 NUM_DESC         = 4
           00000008  0047   374 DESC_PTR         = 8
                     0047   375
               007C  0047   376          .ENTRY  OTS$SFREEN_DD, ^M<R2,R3,R4,R5,R6>       ; Entry point
    50   04 AC   7D  0049   377          MOVQ    NUM_DESC(AP), R0                        ; number of desc ==> R0
                     004D   378                                                          ; address of first desc ==>R1
00000000'GF    16  004D   379          JSB     G^LIB$SFREEN_DD6                        ; go free N descriptors
                     0053   380          SIGNAL_FATAL_ERR                                ; check for fatal error
                  04 0059   381          RET                                             ; to caller
```

```
                     005A   383          .SBTTL OTS$SFREEN_DD6 Deallocate N Dynamic Strings
                     005A   384 ;++
                     005A   385 ; FUNCTIONAL DESCRIPTION:
                     005A   386 ;
                     005A   387 ;       Deallocate a number of strings.  The strings are all assumed
                     005A   388 ;       to be dynamic.  If they aren't, eventually LIB$SFREE1_DD6 wil
                     005A   389 ;       take care of them.
                     005A   390 ;
                     005A   391 ; CALLING SEQUENCE:
                     005A   392 ;
                     005A   393 ;       status.wlc.v = JSB OTS$SFREEN_DD6 (NUM_DESC.rl.v, DESC_PTR.wqu.r)
                     005A   394 ;
                     005A   395 ; FORMAL PARAMETERS:
                     005A   396 ;
                     005A   397 ;       NUM_DESC.rl.v   In R0, the number of descriptors to deallocate.
                     005A   398 ;       DESC_PTR.wqu.r  In R1, the address of first of these descriptors
                     005A   399 ;
                     005A   400 ; IMPLICIT INPUTS:
                     005A   401 ;
                     005A   402 ;       NONE
                     005A   403 ;
                     005A   404 ; IMPLICIT OUTPUTS:
                     005A   405 ;
                     005A   406 ;       NONE
                     005A   407 ;
                     005A   408 ; ROUTINE VALUE:
                     005A   409 ; COMPLETION CODES:
                     005A   410 ;
                     005A   411 ;       SS$_NORMAL
                     005A   412 ;
                     005A   413 ; SIDE EFFECTS:
                     005A   414 ;
                     005A   415 ;       May deallocate virtual storage.
                     005A   416 ;       May signal OTS$_FATINTERR
                     005A   417 ;
                     005A   418 ;--
                     005A   419
                     005A   420 OTS$SFREEN_DD6::
00000000'GF    16    005A   421          JSB     G^LIB$SFREEN_DD6        ; let LIB$SFREEN_DD6 do it
                     0060   422          SIGNAL_FATAL_ERR                ; check for fatal error
               05    0066   423          RSB                             ; return to caller.
```

```
                     0067   425          .SBTTL OTS$SCOPY_DXDX Copy String by Descriptor
                     0067   426 ;++
                     0067   427 ; FUNCTIONAL DESCRIPTION:
                     0067   428 ;
                     0067   429 ;       Copy any supported class string passed by descriptor to any
                     0067   430 ; supported class string.
                     0067   431 ;
                     0067   432 ; CALLING SEQUENCE:
                     0067   433 ;
                     0067   434 ;       status.wlc.v = OTS$SCOPY_DXDX (SRC_DESC.rt.dx, DEST_DESC.wt.dx)
                     0067   435 ;
                     0067   436 ; FORMAL PARAMETERS:
                     0067   437 ;
                     0067   438 ;       SRC_DESC.rt.dx  The source descriptor.
                     0067   439 ;       DEST_DESC.wt.dx The destination descriptor.  The class and dtype
                     0067   440 ;                       fields are not disturbed.
                     0067   441 ;
                     0067   442 ; IMPLICIT INPUTS:
                     0067   443 ;
                     0067   444 ;       NONE
                     0067   445 ;
                     0067   446 ; IMPLICIT OUTPUTS:
                     0067   447 ;
                     0067   448 ;       NONE
                     0067   449 ;
                     0067   450 ; ROUTINE VALUE:
                     0067   451 ; COMPLETION CODES:
                     0067   452 ;
                     0067   453 ;       The number of bytes of the source not moved to the destination.
                     0067   454 ;
                     0067   455 ; SIDE EFFECTS:
                     0067   456 ;
                     0067   457 ;       May allocate and deallocate virtual storage.
                     0067   458 ;       May signal OTS$_INVSTRDES, OTS$_INSVIRMEM, or OTS$_FATINTERR.
                     0067   459 ;
                     0067   460 ;--
                     0067   461
                     0067   462 ; Displacements from AP
                     0067   463
            00000004 0067   464 SRC_DESC        = 4
            00000008 0067   465 DEST_DESC       = 8
                     0067   466
                 007C 0067   467          .ENTRY  OTS$SCOPY_DXDX, ^M<R2,R3,R4,R5,R6>      ; Entry point
                     0069   468 ;+
                     0069   469 ; Copy string using LIB$SCOPY_DXDX6
                     0069   470 ;-
      50    04 AC    7D 0069   471          MOVQ    SRC_DESC(AP), R0        ; load R0 and R1 with addresses
                     006D   472                                             ; source and destination
                     006D   473                                             ; descriptors
00000000'GF    16    006D   474          JSB     G^LIB$SCOPY_DXDX6       ; go copy string
                     0073   475          SIGNAL_FATAL_ERR                ; check for fatal error
                     0079   476
                     0079   477 ;+
                     0079   478 ; Compute length of source string and save it in R4
                     0079   479 ; (no need to check status after call to LIB$ANALYZE_SDESC_R2.  If
                     0079   480 ;  there was anything wrong with source descriptor, LIB$SCOPY_DXDX6
                     0079   481 ;  would already have complained about it.)
```

```
                          0079    482 ;-
                          0079    483         MOVL    SRC_DESC(AP), R0            ; address of source descr.
                          007D    484         CMPB    DSC$B_CLASS(R0), #DSC$K_CLASS_D ; read like fixed ?
                          0081    485         BGTRU   1$                          ; no
                          0083    486         MOVZWL  DSC$W_LENGTH(R0), R4        ; length - > R4
                          0086    487         BRB     2$                          ; join common flow
                          0088    488
                          0088    489 1$:     JSB     G^LIB$ANALYZE_SDESC_R2      ; extract length of source
                          008E    490         MOVZWL  R1, R4                      ; length of source string
                          0091    491
                          0091    492 ;+
                          0091    493 ; Compute length of destination string
                          0091    494 ; (no need to check status after call to LIB$ANALYZE_SDESC_R2.  If
                          0091    495 ;  there was anything wrong with destination descriptor, LIB$SCOPY_DXDX6
                          0091    496 ;  would already have complained about it.)
                          0091    497 ;-
                          0091    498 2$:     MOVL    DEST_DESC(AP), R0           ; address of destination descr
                          0095    499         CMPB    DSC$B_CLASS(R0), #DSC$K_CLASS_D ; read like fixed ?
                          0099    500         BGTRU   3$                          ; no
                          009B    501         MOVZWL  DSC$W_LENGTH(R0), R1        ; length -> R1
                          009E    502         MOVL    DSC$A_POINTER(R0), R2       ; address -> R2
                          00A2    503         BRB     4$                          ; join common flow
                          00A4    504
                          00A4    505 3$:     JSB     G^LIB$ANALYZE_SDESC_R2      ; extract length of destination
                          00AA    506 ;+
                          00AA    507 ; At this point, R1 is length of destination, R2 is address of
                          00AA    508 ; destination, and R4 is length of source.
                          00AA    509 ;
                          00AA    510 ; Compute MAX (0, source_length - destination_length).  This becomes
                          00AA    511 ; the number of unmoved bytes which must end up in R0.
                          00AA    512 ;-
50   54   51   C3          00AA    513 4$:     SUBL3   R1, R4, R0                  ; source len - destination len
          02   18          00AE    514         BGEQ    5$                          ; if positive
          50   D4          00B0    515         CLRL    R0                          ; else zero
                           00B2    516 5$:
               04          00B2    517         RET
```

```
                     00B3   519          .SBTTL  OTS$SCOPY_DXDX6 Copy String by Descriptor
                     00B3   520 ;++
                     00B3   521 ; FUNCTIONAL DESCRIPTION:
                     00B3   522 ;
                     00B3   523 ;       Copy any supported class string passed by descrip*or to any
                     00B3   524 ; supported class string.
                     00B3   525 ;
                     00B3   526 ; CALLING SEQUENCE:
                     00B3   527 ;
                     00B3   528 ;     status.wlc.v = JSB OTS$SCOPY_DXDX6 (SRC_DESC.rt.dx,
                     00B3   529 ;                                         DEST_DESC.wt.dx)
                     00B3   530 ;
                     00B3   531 ; FORMAL PARAMETERS:
                     00B3   532 ;
                     00B3   533 ;       SRC_DESC.rt.dx  The source descriptor, in R0.
                     00B3   534 ;       DEST_DESC.wt.dx The destination descriptor.  The class and dtype
                     00B3   535 ;                       fields are not disturbed.  This is in R1.
                     00B3   536 ;
                     00B3   537 ; IMPLICIT INPUTS:
                     00B3   538 ;
                     00B3   539 ;       None
                     00B3   540 ;
                     00B3   541 ; IMPLICIT OUTPUTS:
                     00B3   542 ;
                     00B3   543 ;       R0              Number of unmoved bytes remaining in source
                     00B3   544 ;                       string.
                     00B3   545 ;       R1              Address one byte beyond the last byte in the
                     00B3   546 ;                       source string that was moved.
                     00B3   547 ;       R2              0
                     00B3   548 ;       R3              Address one byte beyond the destination string
                     00B3   549 ;       R4              0
                     00B3   550 ;       R5              0
                     00B3   551 ;       PSL<N>          1 = Source length less than destination length
                     00B3   552 ;       PSL<Z>          1 = Source length equals destination length
                     00B3   553 ;       PSL<V>          0
                     00B3   554 ;       PSL<C>          1 = Source length LSSU destination length
                     00B3   555 ;
                     00B3   556 ; ROUTINE VALUE:
                     00B3   557 ; COMPLETION CODES:
                     00B3   558 ;
                     00B3   559 ;       See IMPLICIT OUTPUTS, above.
                     00B3   560 ;
                     00B3   561 ; SIDE EFFECTS:
                     00B3   562 ;
                     00B3   563 ;       May allocate and deallocate virtual storage.
                     00B3   564 ;       May signal OTS$_INVSTRDES, OTS$_INSVIRMEM, OTS$_FATINTERR
                     00B3   565 ;
                     00B3   566 ;--
                     00B3   567
                     00B3   568 ; Temp locations on stack
                     00B3   569
00000000             00B3   570 TEMP_SRC_ADDR   = 0
00000004             00B3   571 TEMP_DST_ADDR   = 4
00000008             00B3   572 TEMP_SRC_LEN    = 8
0000000C             00B3   573 TEMP_DST_LEN    = 12
00000010             00B3   574 STACK_SPACE     = 16
                     00B3   575
```

```
                                00B3   576 OTS$SCOPY_DXDX6::
                                00B3   577 ;+
                                00B3   578 ; Save R0 (source desc addr) and R1 (dest descr addr) on the stack.
                                00B3   579 ;-
      5E    10    C2    00B3   580          SUBL2   #STACK_SPACE, SP          ; make space on stack
      6E    50    D0    00B6   581          MOVL    R0, TEMP_SRC_ADDR(SP)     ; Save source descr address
04 AE       51    D0    00B9   582          MOVL    R1, TEMP_DST_ADDR(SP)     ; Save destination descr addr
                                00BD   583 ;+
                                00BD   584 ; Copy string using LIB$SCOPY_DXDX6
                                00BD   585 ;-
00000000'GF         16    00BD   586          JSB     G^LIB$SCOPY_DXDX6         ; go copy string
                                00C3   587          SIGNAL_FATAL_ERR          ; check for fatal error
                                00C9   588 ;+
                                00C9   589 ; Compute length and address of source string and save on stack
                                00C9   590 ; (no need to check status after call to LIB$ANALYZE_SDESC_R2.  If
                                00C9   591 ;  there was anything wrong with source descriptor, LIB$SCOPY_DXDX6
                                00C9   592 ;  would already have complained about it.)
                                00C9   593 ;-
      50    6E    D0    00C9   594          MOVL    TEMP_SRC_ADDR(SP), R0     ; address of source descr.
02    03 A0       91    00CC   595          CMPB    DSC$B_CLASS(R0), #DSC$K_CLASS_D ; read like fixed ?
            0A    1A    00D0   596          BGTRU   1$                        ; no
08 AE       60    3C    00D2   597          MOVZWL  DSC$W_LENGTH(R0), TEMP_SRC_LEN(SP)      ; length
6E    04 A0       D0    00D6   598          MOVL    DSC$A_POINTER(R0), TEMP_SRC_ADDR(SP)    ; address
            0D    11    00DA   599          BRB     2$                        ; join common flow
                                00DC   600
00000000'GF         16    00DC   601 1$:      JSB     G^LIB$ANALYZE_SDESC_R2    ; extract length of source
08 AE       51    D0    00E2   602          MOVL    R1, TEMP_SRC_LEN(SP)      ; length of source string
   6E       52    D0    00E6   603          MOVL    R2, TEMP_SRC_ADDR(SP)     ; addr of 1st byte of source
                                00E9   604
                                00E9   605 ;+
                                00E9   606 ; Compute length and address of destination string and save on stack
                                00E9   607 ; (no need to check status after call to LIB$ANALYZE_SDESC_R2.  If
                                00E9   608 ;  there was anything wrong with destination descriptor, LIB$SCOPY_DXDX6
                                00E9   609 ;  would already have complained about it.)
                                00E9   610 ;-
      50    04 AE       D0    00E9   611 2$:      MOVL    TEMP_DST_ADDR(SP), R0     ; address of destination descr
      02    03 A0       91    00ED   612          CMPB    DSC$B_CLASS(R0), #DSC$K_CLASS_D ; read like fixed ?
                  0B    1A    00F1   613          BGTRU   3$                        ; no
      0C AE       60    3C    00F3   614          MOVZWL  DSC$W_LENGTH(R0), TEMP_DST_LEN(SP)      ; length
04 AE    04 A0       D0    00F7   615          MOVL    DSC$A_POINTER(R0), TEMP_DST_ADDR(SP)    ; address
                  0E    11    00FC   616          BRB     4$                        ; join common flow
                                00FE   617
   00000000'GF         16    00FE   618 3$:      JSB     G^LIB$ANALYZE_SDESC_R2    ; extract length of destination
   0C AE       51    D0    0104   619          MOVL    R1, TEMP_DST_LEN(SP)      ; length of dest string
   04 AE       52    D0    0108   620          MOVL    R2, TEMP_DST_ADDR(SP)     ; address of 1st byte of dest.
                                010C   621
                                010C   622 ; Compute MAX (0, source_length - destination_length).  This becomes
                                010C   623 ; the number of unmoved bytes which must eventually end up in R0.
                                010C   624 ;-
56  08 AE    0C AE    C3    010C   625 4$:      SUBL3   TEMP_DST_LEN(SP), TEMP_SRC_LEN(SP), R6
                  02    18    0112   626          BGEQ    5$                        ; if positive
                  56    D4    0114   627          CLRL    R6                        ; else zero
                                0116   628 5$:
                                0116   629
                                0116   630 ;+
                                0116   631 ; Compute address of first unmoved source byte as
                                0116   632 ;       R1 = TEMP_SRC_ADDR + MIN (TEMP_SRC_LEN, TEMP_DST_LEN)
```

```
                                0116   633 ;-
          08 AE    0C AE   D1  0116   634         CMPL    TEMP_DST_LEN(SP), TEMP_SRC_LEN(SP)
                       07   18  0119   635         BGEQ    6$                      ; Destination length bigger
          51    0C AE   6E   C1  011D   636         ADDL3   TEMP_SRC_ADDR(SP), TEMP_DST_LEN(SP), R1 ;dst len smaller
                       05   11  0122   637         BRB     7$
          51    08 AE   6E   C1  0124   638 6$:     ADDL3   TEMP_SRC_ADDR(SP), TEMP_SRC_LEN(SP), R1 ;src len smaller
                                0129   639 7$:
                                0129   640
                                0129   641 ;+
                                0129   642 ; Compute address of one byte beyond last byte written to destination
                                0129   643 ; string as
                                0129   644 ;       R3 = TEMP_DST_ADDR + TEMP_DST_LEN
                                0129   645 ;-
     53    0C AE    04 AE   C1  0129   646         ADDL3   TEMP_DST_ADDR(SP), TEMP_DST_LEN(SP), R3
                                012F   647
                                012F   648 ;+
                                012F   649 ; Set up all remaining registers as specified under IMPLICIT OUTPUTS
                                012F   650 ; above.
                                012F   651 ;
                                012F   652 ; Final movement must be into R0 in order to set up the condition codes
                                012F   653 ; correctly.
                                012F   654 ;
                                012F   655 ; First, we restore the stack.
                                012F   656 ;-
               5E    10   C0  012F   657         ADDL2   #STACK_SPACE, SP        ; restore stack address
                                0132   658
                     54   7C  0132   659         CLRQ    R4                      ; R4 and R5 are zero
                                0134   660                                         ; R3 all set
                     52   D4  0134   661         CLRL    R2                      ; R2 is zero
                                0136   662                                         ; R1 all set
               50    56   D0  0136   663         MOVL    R6, R0                  ; Set R0 and set condition codes
                                0139   664                                         ; properly
                          05   0139   665         RSB                             ; return to caller
```

```
                        .SBTTL OTS$SCOPY_R_DX Copy String by Reference
                ;++
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       Copy any string passed by reference to any supported class
                ; string passed by descriptor.
                ;
                ; CALLING SEQUENCE:
                ;
                ;       status.wlc.v = CTS$SCOPY_R_DX (SRC_LEN.rwu.v, SRC_ADDR.rt.r,
                ;                                      DEST_DESC.wt.dx)
                ;
                ; FORMAL PARAMETERS:
                ;
                ;       SRC_LEN.rwu.v   The number of bytes of data in the source
                ;       SRC_ADDR.rt.r   The address of the first of those bytes.
                ;       DEST_DESC.wt.dx The destination descriptor.  The class and dtype
                ;                       fields are not disturbed.
                ;
                ; IMPLICIT INPUTS:
                ;
                ;       NONE
                ;
                ; IMPLICIT OUTPUTS:
                ;
                ;       NONE
                ;
                ; ROUTINE VALUE:
                ; COMPLETION CODES:
                ;
                ;       The number of unmoved source bytes, or 0 if there are no unmoved
                ;       source bytes.
                ;
                ; SIDE EFFECTS:
                ;
                ;       May allocate and deallocate virtual storage.
                ;       May signal OTS$_INVSTRDES, OTS$_INSVIRMEM, OTS$_FATINTERR
                ;
                ;--

                ; Displacements off AP

00000004        SRC_LEN         = 4
00000008        SRC_ADDR        = 8
0000000C        DEST_DESC       = 12

                        .ENTRY  OTS$SCOPY_R_DX, ^M<R2,R3,R4,R5,R6>      ; Entry point
                ;+
                ; Copy string using LIB$SCOPY_R_DX6
                ;-
50   04 AC   D0         MOVL    SRC_LEN(AP), R0          ; R0 is length of source
51   08 AC   7D         MOVQ    SRC_ADDR(AP), R1         ; R1 is addr of source
                                                         ; R2 is addr of dest desc
00000000'GF   16        JSB     G^LIB$SCOPY_R_DX6        ; copy the string
                        SIGNAL_FATAL_ERR                 ; check for fatal error

                ;+
```

```
                        0150   724 ; Compute length of destination string (into R1).
                        0150   725 ; (no need to check status after call to LIB$ANALYZE_SDESC_R2.  If
                        0150   726 ;  there was anything wrong with destination descriptor, LIB$SCOPY_DXDX6
                        0150   727 ;  would already have complained about it.)
                        0150   728 ;-
 50    0C AC    D0      0150   729          MOVL    DEST_DESC(AP), R0          ; address of dest descr
 02    03 A0    91      0154   730          CMPB    DSC$B_CLASS(R0), #DSC$K_CLASS_D ; rad like fixed ?
          05    1A      0158   731          BGTRU   1$                         ; no
    51    60    3C      015A   732          MOVZWL  DSC$W_LENGTH(R0), R1       ; length -> R1
          06    11      015D   733          BRB     2$                         ; join common flow
                        015F   734
00000000'GF     16      015F   735 1$:      JSB     G^LIB$ANALYZE_SDESC_R2     ; get length
                        0165   736 ;+
                        0165   737 ; Compute R0 = MAX (0, source_len - destination_length)
                        0165   738 ;-
 52    04 AC    3C      0165   739 2$:      MOVZWL  SRC_LEN(AP), R2            ; words worth of source length
50    52    51      C3  0169   740          SUBL3   R1, R2, R0                 ; R0 = dest len - src len
          02    18      016D   741          BGEQ    3$                         ; if positive
          50    D4      016F   742          CLRL    R0                         ; else zero
                        0171   743 3$:
                04      0171   744          RET                                ; to caller
```

```
                     ;  746          .SBTTL  OTS$SCOPY_R_DX6 Copy Strings by reference
                     ;  747 ;++
                     ;  748 ; FUNCTIONAL DESCRIPTION:
                     ;  749 ;
                     ;  750 ;       Copy any class string passed by reference to any supported
                     ;  751 ; class string passed by descriptor.
                     ;  752 ;
                     ;  753 ; CALLING SEQUENCE:
                     ;  754 ;
                     ;  755 ;       status.wlc.v = JSB OTS$SCOPY_R_DX6 (SRC_LEN.rwu.v, SRC_ADDR.rt.r,
                     ;  756 ;                                           DEST_DESC.wt.dx)
                     ;  757 ;
                     ;  758 ; FORMAL PARAMETERS:
                     ;  759 ;
                     ;  760 ;       SRC_LEN.rwu.v     The number of source bytes, in R0.
                     ;  761 ;       SRC_ADDR.rt.r   Address of the first of these bytes, in R1.
                     ;  762 ;       DEST_DESC.wt.dx  The destination descriptor.  The class and
                     ;  763 ;                         dtype fields are not disturbed.  This is in R2
                     ;  764 ;
                     ;  765 ; IMPLICIT INPUTS:
                     ;  766 ;
                     ;  767 ;       None
                     ;  768 ;
                     ;  769 ; IMPLICIT OUTPUTS:
                     ;  770 ;
                     ;  771 ;       R0              Number of unmoved bytes remaining in source
                     ;  772 ;                       string.
                     ;  773 ;       R1              Address one byte beyond the last byte in the
                     ;  774 ;                       source string that was moved.
                     ;  775 ;       R2              0
                     ;  776 ;       R3              Address one byte beyond the destination string
                     ;  777 ;       R4              0
                     ;  778 ;       R5              0
                     ;  779 ;       PSL<N>          1 = Source length less than destination length
                     ;  780 ;       PSL<Z>          1 = Source length equals destination length
                     ;  781 ;       PSL<V>          0
                     ;  782 ;       PSL<C>          1 = Source length LSSU destination length
                     ;  783 ;
                     ;  784 ; ROUTINE VALUE:
                     ;  785 ; COMPLETION CODES:
                     ;  786 ;
                     ;  787 ;       See IMPLICIT OUTPUTS, above.
                     ;  788 ;
                     ;  789 ; SIDE EFFECTS:
                     ;  790 ;
                     ;  791 ;       May allocate and deallocate virtual storage.
                     ;  792 ;       May signal OTS$_INVSTRDES, OTS$_INSVIRMEM, OTS$_FATINTERR.
                     ;  793 ;
                     ;  794 ;--
                     ;  795
                     ;  796 ; Temp locations on stack
                     ;  797
00000000             ;  798 TEMP_SRC_ADDR    = 0
00000004             ;  799 TEMP_DST_ADDR    = 4
00000008             ;  800 TEMP_SRC_LEN     = 8
0000000C             ;  801 TEMP_DST_LEN     = 12
00000010             ;  802 STACK_SPACE      = 16
```

```
                        0172   803
                        0172   804 OTS$SCOPY_R_DX6::
                        0172   805 ;+
                        0172   806 ; Save input arguments (in R0 - R2) onto stack
                        0172   807 ;-
      5E   10   C2      0172   808         SUBL2   #STACK_SPACE, SP          ; make space on stack
   08 AE   50   3C      0175   809         MOVZWL  R0, TEMP_SRC_LEN(SP)      ; Save source length
      6E   51   D0      0179   810         MOVL    R1, TEMP_SRC_ADDR(SP)     ; Save source descr addr
   04 AE   52   D0      017C   811         MOVL    R2, TEMP_DST_ADDR(SP)     ; Save destination desc addr
                        0180   812 ;+
                        0180   813 ; Copy string using LIB$SCOPY_R_DX6
                        0180   814 ;-
00000000'GF   16        0180   815         JSB     G^LIB$SCOPY_R_DX6         ; go copy string
                        0186   816         SIGNAL_FATAL_ERR                  ; check for fatal error
                        018C   817 ;+
                        018C   818 ; Compute length and address of destination string and save on stack
                        018C   819 ; (no need to check status after call to LIB$ANALYZE_SDESC_R2.  If
                        018C   820 ;  there was anything wrong with destination descriptor, LIB$SCOPY_DXDX6
                        018C   821 ;  would already have complained about it.)
                        018C   822 ;-
   50   04 AE   D0      018C   823         MOVL    TEMP_DST_ADDR(SP), R0     ; address of destination descr
   02   03 A0   91      0190   824         CMPB    DSC$B_CLASS(R0), #DSC$K_CLASS_D ; read like fixed ?
           0B   1A      0194   825         BGTRU   1$                        ; no
   0C AE   60   3C      0196   826         MOVZWL  DSC$W_LENGTH(R0), TEMP_DST_LEN(SP) ; length
04 AE   04 A0   D0      019A   827         MOVL    DSC$A_POINTER(R0), TEMP_DST_ADDR(SP)    ; length
           0E   11      019F   828         BRB     2$                        ; join common flow
                        01A1   829
00000000'GF   16        01A1   830 1$:      JSB     G^LIB$ANALYZE_SDESC_R2    ; extract length of destination
   0C AE   51   D0      01A7   831         MOVL    R1, TEMP_DST_LEN(SP)      ; length of dest string
   04 AE   52   D0      01AB   832         MOVL    R2, TEMP_DST_ADDR(SP)     ; address of 1st byte of dest.
                        01AF   833
                        01AF   834 ; Compute MAX (0, source_length - destination_length).  This becomes
                        01AF   835 ; the number of unmoved bytes which must eventually end up in R0.
                        01AF   836 ;-
56   08 AE   0C AE   C3 01AF   837 2$:      SUBL3   TEMP_DST_LEN(SP), TEMP_SRC_LEN(SP), R6
              02   18   01B5   838         BGEQ    3$                        ; If positive
              56   D4   01B7   839         CLRL    R6                        ; else zero
                        01B9   840 3$:
                        01B9   841
                        01B9   842 ;+
                        01B9   843 ; Compute address of first unmoved source byte as
                        01B9   844 ;       R1 = TEMP_SRC_ADDR + MIN (TEMP_SRC_LEN, TEMP_DST_LEN)
                        01B9   845 ;-
   08 AE   0C AE   D1   01B9   846         CMPL    TEMP_DST_LEN(SP), TEMP_SRC_LEN(SP)
              07   18   01BE   847         BGEQ    4$                        ; Destination length bigger
51   0C AE   6E   C1   01C0   848         ADDL3   TEMP_SRC_ADDR(SP), TEMP_DST_LEN(SP), R1 ;dst len smaller
              05   11   01C5   849         BRB     5$
51   08 AE   6E   C1   01C7   850 4$:      ADDL3   TEMP_SRC_ADDR(SP), TEMP_SRC_LEN(SP), R1 ;src len smaller
                        01CC   851 5$:
                        01CC   852
                        01CC   853 ;+
                        01CC   854 ; Compute address of one byte beyond last byte written to destination
                        01CC   855 ; string as
                        01CC   856 ;       R3 = TEMP_DST_ADDR + TEMP_DST_LEN
                        01CC   857 ;-
53   0C AE   04 AE   C1 01CC   858         ADDL3   TEMP_DST_ADDR(SP), TEMP_DST_LEN(SP), R3
                        01D2   859
```

```
                     01D2    860 ;+
                     01D2    861 ; Set up all remaining registers as specified under IMPLICIT OUTPUTS
                     01D2    862 ; above.
                     01D2    863 ;
                     01D2    864 ; Final movement must be into R0 in order to set up the condition codes
                     01D2    865 ; correctly.
                     01D2    866 ;
                     01D2    867 ; First, we restore the stack.
                     01D2    868 ;-
5E   10   C0         01D2    869         ADDL2   #STACK_SPACE, SP        ; restore stack address
                     01D5    870
     54   7C         01D5    871         CLRQ    R4                      ; R4 and R5 are zero
                     01D7    872                                         ; R3 all set
     52   D4         01D7    873         CLRL    R2                      ; R2 is zero
                     01D9    874                                         ; R1 all set
50   56   D0         01D9    875         MOVL    R6, R0                  ; Set R0 and set condition codes
                     01DC    876                                         ; properly
          05         01DC    877         RSB                             ; to caller
```

```
                                J 4
01DD    879          .SBTTL  CHECK_FOR_FATAL_ERROR
01DD    880 ;++
01DD    881 ; FUNCTIONAL DESCRIPTION:
01DD    882 ;
01DD    883 ; This routine looks at current status in R0 and if it finds one of the
01DD    884 ; fatal LIB$ errors, its causes the corresponding OTS$ error to be
01DD    885 ; signalled.
01DD    886 ;
01DD    887 ;       if it finds:            it signals:
01DD    888 ;       ------------            -----------
01DD    889 ;       LIB$_FATERRLIB          OTS$_FATINTERR
01DD    890 ;       LIB$_INVSTRDES          OTS$_INVSTRDES
01DD    891 ;       LIB$_INSVIRMEM          OTS$_INSVIRMEM
01DD    892 ;       LIB$_WRONUMARG          OTS$_WRONUMARG
01DD    893 ;       LIB$_INVARG             OTS$_INVSTRDES
01DD    894 ;
01DD    895 ;
01DD    896 ; CALLING SEQUENCE:
01DD    897 ;
01DD    898 ;     JMP CHECK_FOR_FATAL_ERROR  with INPUT_STATUS.rlc.v in R0
01DD    899 ;
01DD    900 ; FORMAL PARAMETERS:
01DD    901 ;
01DD    902 ;       INPUT_STATUS.rlc.v      In R0, the status to be checked.
01DD    903 ;                               On entry, we know it is not a success
01DD    904 ;                               status.
01DD    905 ;
01DD    906 ; IMPLICIT INPUTS:
01DD    907 ;
01DD    908 ;       None
01DD    909 ;
01DD    910 ; IMPLICIT OUTPUTS:
01DD    911 ;
01DD    912 ;       NONE
01DD    913 ;
01DD    914 ; ROUTINE VALUE:
01DD    915 ; COMPLETION CODES:
01DD    916 ;
01DD    917 ;       Never returns
01DD    918 ;
01DD    919 ; SIDE EFFECTS:
01DD    920 ;
01DD    921 ;       Will signal some error.
01DD    922 ;
01DD    923 ;--
01DD    924
01DD    925 CHECK_FOR_FATAL_ERROR:
01DD    926
00000000'8F  50  D1  01DD    927          CMPL    R0, #LIB$_INVSTRDES
             08  12  01E4    928          BNEQ    1$
     00000000'8F  DD  01E6    929          PUSHL   #OTS$_INVSTRDES
             46  11  01EC    930          BRB     FATAL
                 01EE    931 1$:
00000000'8F  50  D1  01EE    932          CMPL    R0, #LIB$_INSVIRMEM
             08  12  01F5    933          BNEQ    2$
     00000000'8F  DD  01F7    934          PUSHL   #OTS$_INSVIRMEM
             35  11  01FD    935          BRB     FATAL
```

```
                        01FF    936 2$:
00000000'8F    50   D1  01FF    937             CMPL    R0, #LIB$_FATERRLIB
               08   12  0206    938             BNEQ    3$
     00000000'8F    DD  0208    939             PUSHL   #OTS$_FATINTERR
               24   11  020E    940             BRB     FATAL
                        0210    941 3$:
00000000'8F    50   D1  0210    942             CMPL    R0, #LIB$_INVARG
               08   12  0217    943             BNEQ    4$
     00000000'8F    DD  0219    944             PUSHL   #OTS$_INVSTRDES
               13   11  021F    945             BRB     FATAL
                        0221    946 4$:
00000000'8F    50   D1  0221    947             CMPL    R0, #LIB$_WRONUMARG
               08   12  0228    948             BNEQ    5$
     00000000'8F    DD  022A    949             PUSHL   #OTS$_WRONUMARG
               02   11  0230    950             BRB     FATAL
                        0232    951 5$:
               50   DD  0232    952             PUSHL   R0                      ; prepare to signal incoming
                        0234    953                                             ; error
00000000'GF    01   FB  0234    954 FATAL:      CALLS   #1, G^LIB$STOP          ; to never return
                        023B    955
                        023B    956
                        023B    957
                        023B    958             .END                            ; End of module OTS$SCOPY
```

```
CHECK_FOR_FATAL_ERROR           000001DD R      03
DESCRIP                       = 00000004
DEST_DESC                     = 0000000C
DSC$A_POINTER                 = 00000004
DSC$B_CLASS                   = 00000003
DSC$K_CLASS_D                 = 00000002
DSC$W_LENGTH                  = 00000000
FATAL                           00000234 R      03
LEN                           = 00000004
LIB$ANALYZE_SDESC_R2            ********   X    00
LIB$SCOPY_DXDX6                 ********   X    00
LIB$SCOPY_R_DX6                 ********   X    00
LIB$SFREE1_DD6                  ********   X    00
LIB$SFREEN_DD6                  ********   X    00
LIB$SGET1_DD_R6                 ********   X    00
LIB$STOP                        ********   X    00
LIB$_FATERRLIB                  ********   X    00
LIB$_INSVIRMEM                  ********   X    00
LIB$_INVARG                     ********   X    00
LIB$_INVSTRDES                  ********   X    00
LIB$_WRONUMARG                  ********   X    00
NUM_DESC                      = 00000004
OTS$SCOPY_DXDX                  00000067 RG     03
OTS$SCOPY_DXDX6                 000000B3 RG     03
OTS$SCOPY_R_DX                  0000013A RG     03
OTS$SCOPY_R_DX6                 00000172 RG     03
OTS$SFREE1_DD                   00000027 RG     03
OTS$SFREE1_DD6                  0000003A RG     03
OTS$SFREEN_DD                   00000047 RG     03
OTS$SFREEN_DD6                  0000005A RG     03
OTS$SGET1_DD                    00000000 RG     03
OTS$SGET1_DD_R6                 00000017 RG     03
OTS$_FATINTERR                  ********   X    00
OTS$_INSVIRMEM                  ********   X    00
OTS$_INVSTRDES                  ********   X    00
OTS$_WRONUMARG                  ********   X    00
SRC_ADDR                      = 00000008
SRC_DESC                      = 00000004
SRC_LEN                       = 00000004
STACK_SPACE                   = 00000010
TEMP_DST_ADDR                 = 00000004
TEMP_DST_LEN                  = 0000000C
TEMP_SRC_ADDR                 = 00000000
TEMP_SRC_LEN                  = 00000008
```

+----------------+
! Psect synopsis !
+----------------+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
_OTS$DATA	00000000 (0.)	02 (2.)	PIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
_OTS$CODE	0000023B (571.)	03 (3.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	34	00:00:00.03	00:00:02.48
Command processing	116	00:00:00.33	00:00:02.20
Pass 1	245	00:00:04.06	00:00:15.93
Symbol table sort	0	00:00:00.58	00:00:02.99
Pass 2	163	00:00:01.55	00:00:06.24
Symbol table output	6	00:00:00.03	00:00:00.03
Psect synopsis output	3	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	569	00:00:06.61	00:00:29.90

The working set limit was 1350 pages.
37480 bytes (74 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 579 non-local and 35 local symbols.
958 source lines were read in Pass 1, producing 28 object records in Pass 2.
11 pages of virtual memory were used to define 9 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SYSLIB]STARLET.MLB;2	5

604 GETS were required to define 5 macros.

There were no errors, warnings or information messages.

```
MACRO/ENABLE=SUPPRESSION/DISABLE=(GLOBAL,TRACEBACK)/LIS=LIS$:OTSSCOPY/OBJ=OBJ$:OTSSCOPY MSRC$:OTSSCOPY/UPDATE=(ENH$:OTSSCOPY)
```

OTSPKDIVL
LIS

OTSPKDIVS
LIS

STRABMUL
LIS

STRCHESTA
LIS

RTLLIB
LIS

STRAPPEND
LIS

STRARITH
LIS

STRALLOC
LIS

STRANASTR
LIS

STRCOMCAS
LIS

OTSSCOPY
LIS

OTSTERMIO
LIS