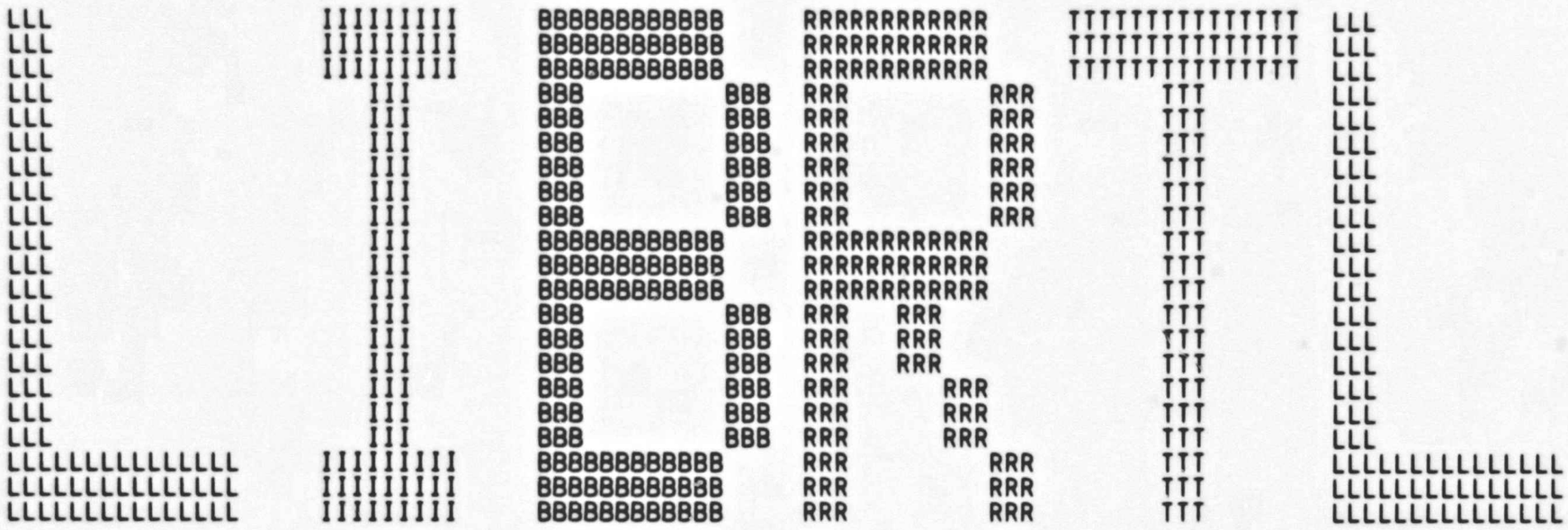
LLL IIIIIIIII BBBBBBBBBBB RRRRRRRRRRR TTTTTTTTTTTTTTT LLL
LLL IIIIIIIII BBBBBBBBBBBB RRRRRRRRRRRR TTTTTTTTTTTTTTT LLL
LLL IIIIIIIII BBBBBBBBBBBB RRRRRRRRRRR TTTTTTTTTTTTTTT LLL
LLL III BBB BBB RRR RRR TTT LLL
LLL III BBB BBB RRR RRR TTT LLL
LLL III BBB BBB RRR RRR TTT LLL
LLL III BBB BBB RRR RRR TTT LLL
LLL III BBB BBB RRR RRR TTT LLL
LLL III BBB BBB RRR RRR TTT LLL
LLL III BBBBBBBBBBB RRRRRRRRRRR 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
LLLLLLLLLLLLLLL IIIIIIIII BBBBBBBBBBBB RRR RRR TTT LLLLLLLLLLLLLLL
LLLLLLLLLLLLLLL IIIIIIIII BBBBBBBBBBBB RRR RRR TTT LLLLLLLLLLLLLLL
LLLLLLLLLLLLLLL IIIIIIIII BBBBBBBBBBBB RRR RRR TTT LLLLLLLLLLLLLLL

FILEID**STRALLOC

```
  SSSSSSSS  TTTTTTTTTT  RRRRRRRR      AAAAAA    LL          LL          OOOOOO      CCCCCCCC
  SSSSSSSS  TTTTTTTTTT  RRRRRRRR      AAAAAA    LL          LL          OOOOOO      CCCCCCCC
SS              TT      RR      RR   AA    AA   LL          LL        OO    OO    CC
SS              TT      RR      RR   AA    AA   LL          LL        OO    OO    CC
SS              TT      RR      RR   AA    AA   LL          LL        OO    OO    CC
SS              TT      RR      RR   AA    AA   LL          LL        OO    OO    CC
  SSSSSS        TT      RRRRRRRR     AA    AA   LL          LL        OO    OO    CC
  SSSSSS        TT      RRRRRRRR     AA    AA   LL          LL        OO    OO    CC
        SS      TT      RR  RR       AAAAAAAAAA LL          LL        OO    OO    CC
        SS      TT      RR  RR       AAAAAAAAAA LL          LL        OO    OO    CC
        SS      TT      RR    RR     AA    AA   LL          LL        OO    OO    CC                ....
        SS      TT      RR    RR     AA    AA   LL          LL        OO    OO    CC                ....
SSSSSSSS        TT      RR      RR   AA    AA   LLLLLLLLLL  LLLLLLLLLL  OOOOOO      CCCCCCCC      ....
SSSSSSSS        TT      RR      RR   AA    AA   LLLLLLLLLL  LLLLLLLLLL  OOOOOO      CCCCCCCC      ....

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
```

```
;     1          0001  0 MODULE STR$$ALLOC (                    !
;     2          0002  0                 IDENT = '1-012'        ! File: STRALLOC.B32    Edit: RKR1012
;     3          0003  0                 ) =
;     4          0004  1 BEGIN
;     5          0005  1 !
;     6          0006  1 !*****************************************************************************
;     7          0007  1 !*                                                                           *
;     8          0008  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
;     9          0009  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
;    10          0010  1 !*  ALL RIGHTS RESERVED.                                                     *
;    11          0011  1 !*                                                                           *
;    12          0012  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
;    13          0013  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    14          0014  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;    15          0015  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;    16          0016  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;    17          0017  1 !*  TRANSFERRED.                                                             *
;    18          0018  1 !*                                                                           *
;    19          0019  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;    20          0020  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;    21          0021  1 !*  CORPORATION.                                                             *
;    22          0022  1 !*                                                                           *
;    23          0023  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
;    24          0024  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
;    25          0025  1 !*                                                                           *
;    26          0026  1 !*                                                                           *
;    27          0027  1 !*****************************************************************************
;    28          0028  1 !
;    29          0029  1
;    30          0030  1 !++
;    31          0031  1 ! FACILITY:  String handling : allocation.
;    32          0032  1 !
;    33          0033  1 ! ABSTRACT:
;    34          0034  1 !
;    35          0035  1 !       String allocation is mostly done in-line in the STR$ routines
;    36          0036  1 !       by macros.  This module contains the data structure for short
;    37          0037  1 !       strings, the initialization routine for that data structure,
;    38          0038  1 !       and an allocation routine which is called by the macros when
;    39          0039  1 !       one of the short string queues runs dry.
;    40          0040  1 !
;    41          0041  1 ! ENVIRONMENT:  VAX-11 User Mode
;    42          0042  1 !
;    43          0043  1 ! AUTHOR: John Sauter, CREATION DATE: 12-MAR-1979
;    44          0044  1 !
;    45          0045  1 ! MODIFIED BY:
;    46          0046  1 !
;    47          0047  1 ! 1-001 - Original.  JBS 13-MAR-1979
;    48          0048  1 ! 1-002 - Store in STR$W_ALLOC_LEN the maximum number of data bytes
;    49          0049  1 !         which the user can place in this string.  This will always be
;    50          0050  1 !         a multiple of 8.  JBS 14-MAR-1979
;    51          0051  1 ! 1-003 - Compensate for the new structure of STR$$Q_SHORT_Q.  JBS 16-MAR-1979
;    52          0052  1 ! 1-004 - Rearrange how storage is gotten for an empty short queue to try
;    53          0053  1 !         to improve performance.  JBS 27-MAR-1979
;    54          0054  1 ! 1-005 - Get 128 pages at a time and hand them out to strings as needed.
;    55          0055  1 !         This is another attempt to improve performance.  JBS 27-MAR-1979
;    56          0056  1 ! 1-006 - Rearrange how the 128-page chunk is handed out so that we
;    57          0057  1 !         we need not give a whole page to each string size.  JBS 03-APR-1979
```

```
;    58          0058  1 ! 1-007 - Use the new STR messages for signaling.  JBS 16-MAY-1979
;    59          0059  1 ! 1-008 - Make some minor edits based on the code review.  JBS 12-JUN-1979
;    60          0060  1 ! 1-009 - Use the new symbol names in STRMACROS.REQ.  JBS 21-JUN-1979
;    61          0061  1 ! 1-010 - Change the calling sequence to STR$$ALLOC_SHORT so that the caller
;    62          0062  1 !           loops doing REMQUEs.  This shortens the code in the calling
;    63          0063  1 !           sequence.  JBS 21-JUN-1979
;    64          0064  1 ! 1-011 - String speedup, no signalling from this module.  RW  9-Jan-1980
;    65          0065  1 ! 1-012 - In STR$$ALLOC_SHORT, if we get error return from LIB$GET_VM
;    66          0066  1 !           return immmediately with status of STR$_INSVIRMEM.
;    67          0067  1 !           RKR 2-MAR-1982. (FT1/FT2 QAR #156).
;    68          0068  1 !--
;    69          0069  1
;    70          0070  1 !<BLF/PAGE>
```

```
;    72          0071  1 !
;    73          0072  1 ! SWITCHES:
;    74          0073  1 !
;    75          0074  1
;    76          0075  1 SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
;    77          0076  1
;    78          0077  1 !
;    79          0078  1 ! LINKAGES:
;    80          0079  1 !
;    81          0080  1 !       NONE
;    82          0081  1 !
;    83          0082  1 ! TABLE OF CONTENTS:
;    84          0083  1 !
;    85          0084  1
;    86          0085  1 FORWARD ROUTINE
;    87          0086  1     STR$$INIT : NOVALUE,                    ! Initialize the short queues
;    88          0087  1     STR$$ALOC_SHORT;                        ! Add space to a short queue
;    89          0088  1
;    90          0089  1 !
;    91          0090  1 ! INCLUDE FILES:
;    92          0091  1 !
;    93          0092  1
;    94          0093  1 REQUIRE 'RTLIN:RTLPSECT';                   ! Macros for defining psects
;    95          0188  1
;    96          0189  1 REQUIRE 'RTLIN:STRMACROS';                  ! String macros
;    97          1105  1
;    98          1106  1 LIBRARY 'RTLSTARLE';                        ! System symbols
;    99          1107  1
;   100          1108  1 !
;   101          1109  1 ! MACROS:
;   102          1110  1 !
;   103          1111  1 !       NONE
;   104          1112  1 !
;   105          1113  1 ! EQUATED SYMBOLS:
;   106          1114  1 !
;   107          1115  1
;   108          1116  1 LITERAL
;   109          1117  1 !+
;   110          1118  1 ! Define the number of bytes to get when a short string runs out.
;   111          1119  1 ! This space is divided up into short strings.
;   112          1120  1 !-
;   113          1121  1     STR$K_ALLOC_LRG = 512*128,
;   114          1122  1 !+
;   115          1123  1 ! Define the number of bytes to take out of that space for each length
;   116          1124  1 ! of string.  This must be larger than the largest possible short
;   117          1125  1 ! string.
;   118          1126  1 !-
;   119          1127  1     STR$K_ALLOC_SML = 512;
;   120          1128  1
;   121          1129  1 !
;   122          1130  1 ! PSECTS:
;   123          1131  1 !
;   124          1132  1
;   125          1133  1 DECLARE_PSECTS (STR);                       ! Declare psects for STR$ facility
;   126          1134  1
;   127          1135  1 !
;   128          1136  1 ! GLOBAL STORAGE:
```

```
;  129          1137  1 !
;  130          1138  1
;  131          1139  1 GLOBAL
;  132          1140  1 !+
;  133          1141  1 ! The following cell is zero until the other data has been initialized.
;  134          1142  1 !-
;  135          1143  1     STR$$V_INIT : INITIAL (0),
;  136          1144  1 !+
;  137          1145  1 ! The following are the queues.  They get initialized to empty before
;  138          1146  1 ! being used for the first time.
;  139          1147  1 !-
;  140          1148  1     STR$$Q_SHORT_Q : STR$$SHORT_STR [STR$K_NUM_SH_QS];
;  141          1149  1
;  142          1150  1 !
;  143          1151  1 ! OWN STORAGE
;  144          1152  1 !
;  145          1153  1
;  146          1154  1 OWN
;  147          1155  1 !+
;  148          1156  1 ! The following queue controls the passing out of the 128 pages gotten
;  149          1157  1 ! when we need more storage.  We get space 128 pages at a time to avoid
;  150          1158  1 ! fragmentation with LIB$GET_VM's storage.
;  151          1159  1 ! This queue usually holds 0 or 1 blocks.  The block is up to 128 pages
;  152          1160  1 ! in length.  If an AST goes off while we are allocating, a second large
;  153          1161  1 ! block will be allocated.  When a block is exhausted the queue is examined
;  154          1162  1 ! for another before calling LIB$GET_VM.
;  155          1163  1 !-
;  156          1164  1     PAGE_QUEUE : VECTOR [2];
;  157          1165  1
;  158          1166  1 !
;  159          1167  1 ! EXTERNAL REFERENCES:
;  160          1168  1 !
;  161          1169  1
;  162          1170  1 EXTERNAL ROUTINE
;  163          1171  1     LIB$GET_VM;                                    ! Get virtual storage
;  164          1172  1
;  165          1173  1 !+
;  166          1174  1 ! The following error codes are used by this module:
;  167          1175  1 !-
;  168          1176  1
;  169          1177  1 EXTERNAL LITERAL
;  170          1178  1     STR$_NORMAL : UNSIGNED (6),                        ! Successful completion
;  171          1179  1     STR$_INSVIRMEM;                                ! Insufficient virtual memory
;  172          1180  1
```

```
;  174          1181  1 GLOBAL ROUTINE STR$$INIT                      ! Initialize the queues
;  175          1182  1     : NOVALUE =
;  176          1183  1
;  177          1184  1 !++
;  178          1185  1 ! FUNCTIONAL DESCRIPTION:
;  179          1186  1 !
;  180          1187  1 !       Initialize the short string queues.  This routine is called
;  181          1188  1 !       by the allocation macros before touching them, provided that
;  182          1189  1 !       STR$$V_INIT is clear.  To be AST re-entrant, this routine
;  183          1190  1 !       does the initialization, with ASTs off, only if STR$$V_INIT
;  184          1191  1 !       is still clear.
;  185          1192  1 !
;  186          1193  1 ! FORMAL PARAMETERS:
;  187          1194  1 !
;  188          1195  1 !       NONE
;  189          1196  1 !
;  190          1197  1 ! IMPLICIT INPUTS:
;  191          1198  1 !
;  192          1199  1 !       STR$$V_INIT.mv  If 1, do nothing.  An AST has initialized
;  193          1200  1 !                       the queues.
;  194          1201  1 !
;  195          1202  1 ! IMPLICIT OUTPUTS:
;  196          1203  1 !
;  197          1204  1 !       STR$$V_INIT.mv          Set to 1.
;  198          1205  1 !       STR$$Q_SHORT_Q.wq       Set to "empty" for REMQUE/INSQUE
;  199          1206  1 !
;  200          1207  1 ! COMPLETION CODES:
;  201          1208  1 !
;  202          1209  1 !       NONE
;  203          1210  1 !
;  204          1211  1 ! SIDE EFFECTS:
;  205          1212  1 !
;  206          1213  1 !       Disables and re-enables ASTs if they are enabled at entry.
;  207          1214  1 !
;  208          1215  1 !--
;  209          1216  1
;  210          1217  2     BEGIN
;  211          1218  2
;  212          1219  2     LOCAL
;  213          1220  2         AST_STATUS;
;  214          1221  2
;  215          1222  2 !+
;  216          1223  2 ! Disable ASTs so we can test STR$$V_INIT.  We must not permit an AST
;  217          1224  2 ! after we start to initialize the queues, since such an AST could
;  218          1225  2 ! try to allocate a string.
;  219          1226  2 !-
;  220          1227  2     AST_STATUS = $SETAST (ENBFLG = 0);
;  221          1228  2
;  222          1229  3     IF ( NOT .STR$$V_INIT)
;  223          1230  2     THEN
;  224          1231  3         BEGIN
;  225          1232  3 !+
;  226          1233  3 ! We must do the initialization.  Mark all short string queues as empty.
;  227          1234  3 !-
;  228          1235  3
;  229          1236  3         INCR INDEX FROM 1 TO STR$K_NUM_SH_QS DO
;  230          1237  4             BEGIN
```

```
;  231          1238  4
;  232          1239  4              LOCAL
;  233          1240  4                  Q_ADDR;
;  234          1241  4
;  235          1242  4              Q_ADDR = STR$$Q_SHORT_Q [.INDEX*STR$K_ALL_QUA, 0];
;  236          1243  4              .Q_ADDR = .Q_ADDR;
;  237          1244  4              .Q_ADDR + %UPVAL = .Q_ADDR;
;  238          1245  3              END;
;  239          1246  3
;  240          1247  3      !+
;  241          1248  3      ! Mark that we have obtained no space from LIB$GET_VM.
;  242          1249  3      !-
;  243          1250  3              PAGE_QUEUE [0] = PAGE_QUEUE [1] = PAGE_QUEUE [0];
;  244          1251  3      !+
;  245          1252  3      ! Mark the initialization as complete.
;  246          1253  3      !-
;  247          1254  3              BLOCK [STR$$V_INIT, 0, 0, 1, 0] = 1;
;  248          1255  2              END;
;  249          1256  2
;  250          1257  2      !+
;  251          1258  2      ! If ASTs were enabled when we entered, re-enable them.
;  252          1259  2      !-
;  253          1260  2
;  254          1261  2          IF (.AST_STATUS EQL SS$_WASSET) THEN $SETAST (ENBFLG = 1);
;  255          1262  2
;  256          1263  2          RETURN;
;  257          1264  1          END;                                        ! end of STR$$INIT

                                                                .TITLE  STR$$ALLOC
                                                                .IDENT  \1-012\

                                                                .PSECT  _STR$DATA,NOEXE,  PIC,2

                                00000000  00000 STR$$V_INIT::
                                                .LONG   0                                       ;
                                          00004 STR$$Q_SHORT_Q::
                                                .BLKB   240
                                          000F4 PAGE_QUEUE:
                                                .BLKB   8

                                                .EXTRN  LIB$GET_VM, STR$_NORMAL
                                                .EXTRN  STR$_INSVIRMEM, SYS$SETAST

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

                                     001C 00000         .ENTRY  STR$$INIT, Save R2,R3,R4        ; 1181
                  54 00000000G   00   9E 00002         MOVAB   SYS$SETAST, R4                  ;
                  53 00000000'   EF   9E 00009         MOVAB   STR$$V_INIT, R3                 ;
                                 7E   D4 00010         CLRL    -(SP)                           ; 1227
                  64             01   FB 00012         CALLS   #1, SYS$SETAST                  ;
                  2E             63   E8 00015         BLBS    STR$$V_INIT, 2$                 ; 1229
                  51             01   D0 00018         MOVL    #1, INDEX                       ; 1236
     52           51             03   78 0001B 1$:     ASHL    #3, INDEX, R2                   ; 1242
                                 52   D7 0001F         DECL    R2                              ;
                  52             07   8A 00021         BICB2   #7, R2                          ;
                  52       04 A342   9E 00024         MOVAB   STR$$Q_SHORT_Q[R2], Q_ADDR      ;
```

```
                        62              52  D0 00029          MOVL    Q_ADDR, (Q_ADDR)                        ; 1243
                    04  A2              52  D0 0002C          MOVL    Q_ADDR, 4(Q_ADDR)                       ; 1244
                E7      51              1E  F3 00030          AOBLEQ  #30, INDEX, 1$                          ; 1236
                        51      00F4    C3  9E 00034          MOVAB   PAGE_QUEUE, R1                          ; 1250
                00F8    C3              51  D0 00039          MOVL    R1, PAGE_QUEUE+4                        ;
                00F4    C3              51  D0 0003E          MOVL    R1, PAGE_QUEUE                          ;
                        63              01  88 00043          BISB2   #1, STR$$V_INIT                         ; 1254
                        09              50  D1 00046 2$:      CMPL    AST_STATUS, #9                          ; 1261
                                        05  12 00049          BNEQ    3$                                      ;
                                        01  DD 0004B          PUSHL   #1                                      ;
                        64              01  FB 0004D          CALLS   #1, SYS$SETAST                          ;
                                            04 00050 3$:      RET                                             ; 1264

; Routine Size:  81 bytes,    Routine Base:  _STR$CODE + 0000

;  258          1265  1
```



```
ADI
CLA
CLA
CLA
CLA
CLA
CLA
CLA
CLA
CLA
CLA
CLA
CLA
CLA
DES
DSC
DSC
DSC
DSC
DSC
DSC
ERI
LEI
LIB
STI
STI
STI

PSE
---

$AB
 S
-S
-

Pha
---
In
Co
Pa
Sy
Pa
Sy
Ps
Cr
As
```

```
;  260     1266  1 GLOBAL ROUTINE STR$$ALOC_SHORT (                  ! Allocate a short string
;  261     1267  1         LENGTH                                    ! Length of the string to be allocated
;  262     1268  1     ) =
;  263     1269  1
;  264     1270  1 !++
;  265     1271  1 ! FUNCTIONAL DESCRIPTION:
;  266     1272  1 !
;  267     1273  1 !       Allocate a short string.  This is called only if the short
;  268     1274  1 !       queue for the string is empty.  Space is obtained in large
;  269     1275  1 !       chunks for the allocation pool, if necessary, and one string
;  270     1276  1 !       from this pool is used to satisfy the request.  Unlike other
;  271     1277  1 !       versions of the string package, no unallocated strings of this
;  272     1278  1 !       same length are created.
;  273     1279  1 !       The calling sequence invloves doing a REMQUE,
;  274     1280  1 !       observing that it fails, and then calling this routine.
;  275     1281  1 !       After this routine returns, loop back to the REMQUE, waiting
;  276     1282  1 !       for it to succeed.
;  277     1283  1 !
;  278     1284  1 ! FORMAL PARAMETERS:
;  279     1285  1 !
;  280     1286  1 !       LENGTH.rl.v     The number of bytes wanted in the string.
;  281     1287  1 !                       The attempt to allocate a string with this
;  282     1288  1 !                       many data bytes from the short queues failed.
;  283     1289  1 !
;  284     1290  1 ! IMPLICIT INPUTS:
;  285     1291  1 !
;  286     1292  1 !       STR$$Q_SHORT_Q.mq       One of these queues is empty or we
;  287     1293  1 !                               wouldn't have been called.
;  288     1294  1 !       PAGE_QUEUE.mq           Get a page from here if necessary.
;  289     1295  1 !
;  290     1296  1 ! IMPLICIT OUTPUTS:
;  291     1297  1 !
;  292     1298  1 !       STR$$Q_SHORT_Q.mq       The residue of the old big chunk may be
;  293     1299  1 !                               put on its queue for later use.
;  294     1300  1 !       PAGE_QUEUE.mq           Put 128 pages on here if it is empty.
;  295     1301  1 !
;  296     1302  1 ! COMPLETION CODES:
;  297     1303  1 !
;  298     1304  1 !       STR$_NORMAL
;  299     1305  1 !       STR$_INSVIRMEM          If not enough memory to allocate to queues
;  300     1306  1 !
;  301     1307  1 ! SIDE EFFECTS:
;  302     1308  1 !
;  303     1309  1 !       May allocate virtual storage.
;  304     1310  1 !       If it must but cannot, return STR$_INSVIRMEM
;  305     1311  1 !
;  306     1312  1 !--
;  307     1313  1
;  308     1314  2     BEGIN
;  309     1315  2
;  310     1316  2     BUILTIN
;  311     1317  2         INSQUE,
;  312     1318  2         REMQUE;
;  313     1319  2
;  314     1320  2     LOCAL
;  315     1321  2         RETURN_STATUS,
;  316     1322  2         ALLOC_LENGTH,
```

```
;  317          1323  2          ALLOC_OFFSET,
;  318          1324  2          ALLOC_DONE,
;  319          1325  2          ALLOC_BLOCK : REF VECTOR,
;  320          1326  2          LARGE_LEFT,
;  321          1327  2          INSQUE_ADDR,
;  322          1328  2          STRING_BLOCK : REF BLOCK [, BYTE] FIELD (STR$SHORT_FIELD);
;  323          1329  2
;  324          1330  2      !+
;  325          1331  2      ! Define the offsets into ALLOC_BLOCK.  This vector remembers how much of the
;  326          1332  2      ! current large block can still be used for allocation.
;  327          1333  2      !-
;  328          1334  2
;  329          1335  2          LITERAL
;  330          1336  2              A_ALLOC_FWD = 0,                        ! Forward pointer
;  331          1337  2              A_ALLOC_BAK = 1,                        ! Backward pointer
;  332          1338  2              L_ALLOC_AMT = 2,                        ! Number of free bytes
;  333          1339  2              K_ALLOC_LEN = 3;                        ! Length of this header
;  334          1340  2
;  335          1341  2      !+
;  336          1342  2      ! Compute the number of bytes we must allocate in this string.  The
;  337          1343  2      ! number of bytes wanted is rounded up to the next eight, and the
;  338          1344  2      ! header length is added.
;  339          1345  2      !-
;  340          1346  3          ALLOC_LENGTH = ((.LENGTH + (STR$K_ALL_QUA - 1)) AND ( NOT (STR$K_ALL_QUA - 1)))     !
;  341          1347  2          + STR$K_HED_LEN + STR$K_RESIDUE;
;  342          1348  2      !+
;  343          1349  2      ! If there is already a page on the page queue, use it to satisfy this request.
;  344          1350  2      !-
;  345          1351  2
;  346          1352  3          IF (REMQUE (.PAGE_QUEUE [0], ALLOC_BLOCK))
;  347          1353  2          THEN
;  348          1354  3              BEGIN
;  349          1355  3      !+
;  350          1356  3      ! We need more space on the page queue.  Get a lot of pages to avoid
;  351          1357  3      ! checkerboarding LIB$GET_VM's space, since we will never return our pages.
;  352          1358  3      !-
;  353          1359  3
;  354          1360  3              LOCAL
;  355          1361  3                  CHUNK_BASE;
;  356          1362  3
;  357          1363  3              RETURN_STATUS = LIB$GET_VM (%REF (STR$K_ALLOC_LRG), CHUNK_BASE);
;  358          1364  3              IF ( NOT .RETURN_STATUS) THEN RETURN STR$_INSVIRMEM;
;  359          1365  3
;  360          1366  3              ALLOC_BLOCK = .CHUNK_BASE;
;  361          1367  3              LARGE_LEFT = STR$K_ALLOC_LRG;
;  362          1368  3              END
;  363          1369  2          ELSE
;  364          1370  3              BEGIN
;  365          1371  3              LARGE_LEFT = .ALLOC_BLOCK [L_ALLOC_AMT];
;  366          1372  3              RETURN_STATUS = STR$_NORMAL;
;  367          1373  2              END;
;  368          1374  2
;  369          1375  2      !+
;  370          1376  2      ! Point to the short queue that these strings are to go on.
;  371          1377  2      !-
;  372          1378  2          INSQUE_ADDR = STR$$Q_SHORT_Q [.LENGTH, 0];
;  373          1379  2      !+
```

```
;  374          1380  2 ! Divide part of this space into strings of the requested length and
;  375          1381  2 ! put them in the appropriate short string queue.
;  376          1382  2 !-
;  377          1383  2     ALLOC_OFFSET = 0;
;  378          1384  2     ALLOC_DONE = 0;
;  379          1385  2
;  380          1386  4     WHILE (((.ALLOC_OFFSET + .ALLOC_LENGTH) LSS STR$K_ALLOC_SML)
;  381          1387  2         AND (.LARGE_LEFT GEQ .ALLOC_LENGTH)) DO
;  382          1388  3         BEGIN
;  383          1389  3         STRING_BLOCK =
;  384          1390  3             .ALLOC_BLOCK + .ALLOC_OFFSET + STR$K_HED_LEN + STR$K_RESIDUE;
;  385          1391  3         STRING_BLOCK [STR$W_ALLOC_LEN] =
;  386          1392  3             .ALLOC_LENGTH - STR$K_HED_LEN - STR$K_RESIDUE;
;  387          1393  3         INSQUE (.STRING_BLOCK, ..INSQUE_ADDR);
;  388          1394  3         ALLOC_DONE = 1;
;  389          1395  3         ALLOC_OFFSET = .ALLOC_OFFSET + .ALLOC_LENGTH;
;  390          1396  3         LARGE_LEFT = .LARGE_LEFT - .ALLOC_LENGTH;
;  391          1397  2         END;
;  392          1398  2
;  393          1399  2 !+
;  394          1400  2 ! The space remaining, if any, is returned to the page queue for the next
;  395          1401  2 ! string bucket that comes up empty.  Note that, if there was so little space
;  396          1402  2 ! that we were unable to allocate even a single string, the space is discarded.
;  397          1403  2 ! This can happen only at the end of the "large" blocks.  In this case we
;  398          1404  2 ! will be called again, since the REMQUE will fail again, and we will
;  399          1405  2 ! allocate more space on the page queue.
;  400          1406  2 !-
;  401          1407  2
;  402          1408  3     IF ((.LARGE_LEFT GEQ (K_ALLOC_LEN*%UPVAL)) AND .ALLOC_DONE)
;  403          1409  2     THEN
;  404          1410  3         BEGIN
;  405          1411  3 !+
;  406          1412  3 ! There is enough space left in the large block to put it on the queue.
;  407          1413  3 !-
;  408          1414  3
;  409          1415  3         LOCAL
;  410          1416  3             INSQUE_ADDR;
;  411          1417  3
;  412          1418  3         ALLOC_BLOCK = .ALLOC_BLOCK + .ALLOC_OFFSET;
;  413          1419  3         ALLOC_BLOCK [L_ALLOC_AMT] = .LARGE_LEFT;
;  414          1420  3         INSQUE_ADDR = PAGE_QUEUE [0];
;  415          1421  3         INSQUE (ALLOC_BLOCK [A_ALLOC_FWD], ..INSQUE_ADDR);
;  416          1422  2         END;
;  417          1423  2
;  418          1424  2     RETURN .RETURN_STATUS;
;  419          1425  1     END;                                    ! of routine STR$ALOC_SHORT


                                        03FC 00000          .ENTRY  STR$$ALOC_SHORT, Save R2,R3,R4,R5,R6,R7,R8,-; 1266
                                                                    R9
                    59 00000000'    EF  9E 00002            MOVAB   PAGE_QUEUE, R9                                ;
                    5E              08  C2 00009            SUBL2   #8, SP                                        ;
              52    04              AC  07  C1 0000C        ADDL3   #7, LENGTH, R2                                ; 1346
                    52              07  8A 00011            BICB2   #7, R2                                        ;
```

```
                                52          04  C0 00014          ADDL2   #4, ALLOC_LENGTH                                    ; 1347
                                54       00 B9  0F 00017          REMQUE  @PAGE_QUEUE, ALLOC_BLOCK                           ; 1352
                                           2D  1C 0001B          BVC     2$
                                        04 AE  9F 0001D          PUSHAB  CHUNK_BASE                                         ; 1363
                             04 AE 00010000 8F  D0 00020          MOVL    #65536, 4(SP)
                                        04 AE  9F 00028          PUSHAB  4(SP)
                   00000000G 00             02  FB 0002B          CALLS   #2, LIB$GET_VM
                             08             50  E8 00032          BLBS    RETURN_STATUS, 1$                                  ; 1364
                             50 00000000G   8F  D0 00035          MOVL    #STR$_INSVIRMEM, R0
                                            04 0003C          RET
                                54       04 AE  D0 0003D 1$:      MOVL    CHUNK_BASE, ALLOC_BLOCK                            ; 1366
                                56 00010000 8F  D0 00041          MOVL    #65536, LARGE_LEFT                                 ; 1367
                                           07  11 00048          BRB     3$                                                 ; 1352
                                56       08 A4  D0 0004A 2$:      MOVL    8(ALLOC_BLOCK), LARGE_LEFT                         ; 1371
                                50          00G 9A 0004E          MOVZBL  S^STR$_NORMAL, RETURN_STATUS                       ; 1372
                          51       04 AC          01  C3 00051 3$:      SUBL3   #1, LENGTH, R1                             ; 1378
                                51          07  8A 00056          BICB2   #7, R1
                                58    FF10 C941 9E 00059          MOVAB   STR$$Q_SHORT_Q[R1], INSQUE_ADDR
                                              51  D4 0005F          CLRL    ALLOC_OFFSET                                       ; 1383
                                              57  D4 00061          CLRL    ALLOC_DONE                                         ; 1384
                          55                  51      52  C1 00063 4$:      ADDL3   ALLOC_LENGTH, ALLOC_OFFSET, R5             ; 1386
                             00000200 8F              55  D1 00067          CMPL    R5, #512
                                              1E  18 0006E          BGEQ    5$
                                52                    56  D1 00070          CMPL    LARGE_LEFT, ALLOC_LENGTH                   ; 1387
                                              19  19 00073          BLSS    5$
                                53       04 A144 9E 00075          MOVAB   4(ALLOC_OFFSET)[ALLOC_BLOCK], STRING_BLOCK         ; 1390
                          FE    A3              52      04  A3 0007A          SUBW3   #4, ALLOC_LENGTH, -2(STRING_BLOCK)         ; 1392
                                00          B8      63  0E 0007F          INSQUE  (STRING_BLOCK), @0(INSQUE_ADDR)            ; 1393
                                57                    01  D0 00083          MOVL    #1, ALLOC_DONE                             ; 1394
                                51                    52  C0 00086          ADDL2   ALLOC_LENGTH, ALLOC_OFFSET                 ; 1395
                                56                    52  C2 00089          SUBL2   ALLOC_LENGTH, LARGE_LEFT                   ; 1396
                                              D5  11 0008C          BRB     4$                                                 ; 1386
                                              0C  56  D1 0008E 5$:      CMPL    LARGE_LEFT, #12                            ; 1408
                                              11  19 00091          BLSS    6$
                                              0E  57  E9 00093          BLBC    ALLOC_DONE, 6$
                                              54  51  C0 00096          ADDL2   ALLOC_OFFSET, ALLOC_BLOCK                  ; 1418
                                08    A4              56  D0 00099          MOVL    LARGE_LEFT, 8(ALLOC_BLOCK)                 ; 1419
                                              51  69  9E 0009D          MOVAB   PAGE_QUEUE, INSQUE_ADDR                    ; 1420
                                00    B1              64  0E 000A0          INSQUE  (ALLOC_BLOCK), @0(INSQUE_ADDR)             ; 1421
                                                  04 000A4 6$:      RET                                                        ; 1425

; Routine Size:  165 bytes,    Routine Base:  _STR$CODE + 0051

;  420          1426  1
;  421          1427  1 END                                                 ! end of module STR$$ALLOC
;  422          1428  1
;  423          1429  0 ELUDOM



;                                PSECT SUMMARY
;
;       Name                    Bytes                       Attributes
```

```
;
;  _STR$DATA                    252  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;  _STR$CODE                    246  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         00:00.8




;                                  COMMAND QUALIFIERS

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

; Size:         246 code + 252 data bytes
; Run Time:         00:06.7
; Elapsed Time:     00:22.0
; Lines/CPU Min:    12835
; Lexemes/CPU-Min: 27449
; Memory Used:  89 pages
; Compilation Complete
```

OTSPKDIVL
LIS

OTSPKDIVS
LIS

STRABMUL
LIS

STRCHESTA
LIS

RTLLIB
LIS

STRAPPEND
LIS

STRARITH
LIS

STRALLOC
LIS

STRANASTR
LIS

STRCOMCAS
LIS

OTSSCOPY
LIS

OTSTERMIO
LIS