```
TTTTTTTTTTTTTTTT  RRRRRRRRRRRR        AAAAAAAAA        CCCCCCCCCCCC  EEEEEEEEEEEEEEEE
TTTTTTTTTTTTTTTT  RRRRRRRRRRRRR       AAAAAAAAA        CCCCCCCCCCCCC  EEEEEEEEEEEEEEEE
TTTTTTTTTTTTTTTT  RRRRRRRRRRRRR       AAAAAAAAA        CCCCCCCCCCCCC  EEEEEEEEEEEEEEEE
      TTT         RRR          RRR   AAA          AAA   CCC            EEE
      TTT         RRR          RRR   AAA          AAA   CCC            EEE
      TTT         RRR          RRR   AAA          AAA   CCC            EEE
      TTT         RRR          RRR   AAA          AAA   CCC            EEE
      TTT         RRR          RRR   AAA          AAA   CCC            EEE
      TTT         RRR          RRR   AAA          AAA   CCC            EEE
      TTT         RRRRRRRRRRRRR      AAA          AAA   CCC            EEEEEEEEEEEE
      TTT         RRRRRRRRRRRRR      AAA          AAA   CCC            EEEEEEEEEEEE
      TTT         RRRRRRRRRRRRR      AAA          AAA   CCC            EEEEEEEEEEEE
      TTT         RRR    RRR         AAAAAAAAAAAAAAAA   CCC            EEE
      TTT         RRR    RRR         AAAAAAAAAAAAAAAA   CCC            EEE
      TTT         RRR    RRR         AAAAAAAAAAAAAAAA   CCC            EEE
      TTT         RRR       RRR      AAA          AAA   CCC            EEE
      TTT         RRR       RRR      AAA          AAA   CCC            EEE
      TTT         RRR       RRR      AAA          AAA   CCC            EEE
      TTT         RRR          RRR   AAA          AAA      CCCCCCCCCCCC  EEEEEEEEEEEEEEEE
      TTT         RRR          RRR   AAA          AAA      CCCCCCCCCCCC  EEEEEEEEEEEEEEEE
      TTT         RRR          RRR   AAA          AAA      CCCCCCCCCCCC  EEEEEEEEEEEEEEEE
```

```
TTTTTTTTTT  BBBBBBBB    KK      KK    SSSSSSSS    SSSSSSSS  VV      VV
TTTTTTTTTT  BBBBBBBB    KK      KK    SSSSSSSS    SSSSSSSS  VV      VV
    TT      BB      BB  KK      KK  SS          SS          VV      VV
    TT      BB      BB  KK      KK  SS          SS          VV      VV
    TT      BB      BB  KK    KK    SS          SS          VV      VV
    TT      BB      BB  KK    KK    SS          SS          VV      VV
    TT      BBBBBBBB    KKKKKK        SSSSSS      SSSSSS    VV      VV
    TT      BBBBBBBB    KKKKKK        SSSSSS      SSSSSS    VV      VV
    TT      BB      BB  KK    KK            SS          SS  VV      VV
    TT      BB      BB  KK    KK            SS          SS  VV      VV
    TT      BB      BB  KK      KK          SS          SS    VV  VV        ....
    TT      BB      BB  KK      KK          SS          SS    VV  VV        ....
    TT      BBBBBBBB    KK      KK  SSSSSSSS    SSSSSSSS        VV          ....
    TT      BBBBBBBB    KK      KK  SSSSSSSS    SSSSSSSS        VV          ....


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 TBKSSV ( IDENT = 'V04-000' ) =
;     2          0002  1 BEGIN
;     3          0003  1
;     4          0004  1 !**************************************************************************
;     5          0005  1 !*                                                                        *
;     6          0006  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                             *
;     7          0007  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.               *
;     8          0008  1 !*  ALL RIGHTS RESERVED.                                                 *
;     9          0009  1 !*                                                                        *
;    10          0010  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
;    11          0011  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE *
;    12          0012  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER *
;    13          0013  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
;    14          0014  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY *
;    15          0015  1 !*  TRANSFERRED.                                                           *
;    16          0016  1 !*                                                                        *
;    17          0017  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE *
;    18          0018  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT *
;    19          0019  1 !*  CORPORATION.                                                           *
;    20          0020  1 !*                                                                        *
;    21          0021  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS *
;    22          0022  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                *
;    23          0023  1 !*                                                                        *
;    24          0024  1 !*                                                                        *
;    25          0025  1 !**************************************************************************
;    26          0026  1 !
;    27          0027  1 ! FACILITY:
;    28          0028  1 !       TRACEBACK
;    29          0029  1 !
;    30          0030  1 !++
;    31          0031  1 ! Abstract:
;    32          0032  1 !       FAO formatting routine and output
;    33          0033  1 !
;    34          0034  1 ! Version:      11
;    35          0035  1 !
;    36          0036  1 ! History:
;    37          0037  1 !       Author:
;    38          0038  1 !               Carol Peters, 11 January 1978 : Version 01
;    39          0039  1 !
;    40          0040  1 !       Modified by:
;    41          0041  1 !               Dale Roedger, 14 August 1978: Version 11
;    42          0042  1 !               Victoria Holt, 21 December 1982
;    43          0043  1 !
;    44          0044  1 ! Revision history:
;    45          0045  1 !
;    46          0046  1 !       02      18-jan-78       KGP     -Added a modified version of
;    47          0047  1 !                                        DBGEXC's dbg$report_msg
;    48          0048  1 !       03      24-feb-78       KGP     -Threw away REPORT_MSG as we now
;    49          0049  1 !                                        use a system call for this.
;    50          0050  1 !       04      28-fev-78       KGP     -New routine TBK$IO_SETUP to allow us
;    51          0051  1 !                                        to do I/O as SYS$PUTMSG does it.
;    52          0052  1 !       05      01-mar-78       KGP     -New I/O scheme more like DEBUG's.  i.e.
;    53          0053  1 !                                        we always 'encode' into an output buffer
;    54          0054  1 !                                        and then only do 1 $PUT
;    55          0055  1 !                                       -IO_SETUP now works from LOCALs to do
;    56          0056  1 !                                        the PUTMSG logical name translation.
;    57          0057  1 !                                       -TBK$OUT_PUT now $PUTs to both
```

```
;    58          0058  1 !                                    SYS$OUTPUT and SYS$ERROR if the two
;    59          0059  1 !                                    are not the same.
;    60          0060  1 !          06      7-mar-78    KGP   -New routine  TBK$PUTMSG which writes
;    61          0061  1 !                                    out messages (via SYS$PUTMSG)
;    62          0062  1 !          07      8-mar-78    KGP   -Changed all REQUIRE file names so
;    63          0063  1 !                                    that TRACE is now separate from DEBUG.
;    64          0064  1 !          08      27-mar-78   KGP   -Renamed TBK$PUTMSG to TBK$FAKE_MSG
;    65          0065  1 !                                    -Added TBK$PUT_MSG which is almost
;    66          0066  1 !                                    excatly like DEBUG's new (5X33) DBG$PU'MSG.
;    67          0067  1 !                                    This routine sorts out the problem of
;    68          0068  1 !                                    software- vs hardware-generated SIGNALs.
;    69          0069  1 !          09      26-APR-78   DAR   Modified require and library directives
;    70          0070  1 !                                    for native build.
;    71          0071  1 !          10      15-JUN-78   DAR   Changed all DBG$ symbols to TBK$.
;    72          0072  1 !                                    TBK$FAO_OUT checks for stream active and
;    73          0073  1 !                                    retries $PUT.
;    74          0074  1 !          11      14-AUG-78   DAR   In TBK$PUTMSG we should subtract 2 from
;    75          0075  1 !                                    SIG_ARG_LIST count if the exception code
;    76          0076  1 !                                    is not found in the table of hardware
;    77          0077  1 !                                    exceptions.
;    78          0078  1 !          12      21-Dec-82   VJH   Made corrections so that the original
;    79          0079  1 !                                    status is returned when the user turns
;    80          0080  1 !                                    off all the default message flags
;    81          0081  1 !                                    (SYS$PUTMSG does not define SYS$OUTPUT
;    82          0082  1 !                                    and SYS$ERROR in that case).
;    83          0083  1 !--
```

```
;    85          0084  1 ! Table of contents:
;    86          0085  1 !
;    87          0086  1 FORWARD ROUTINE
;    88          0087  1         TBK$FAKE_MSG : NOVALUE,         ! Write out fake traceback messages.
;    89          0088  1         tbk$put_msg,                    ! Write out system-generated SIGNAL messages.
;    90          0089  1         tbk$fao_put : novalue,          ! Format into an output buffer.
;    91          0090  1         TBK$IO_SETUP,                   ! Init I/O to SYS$OUTPUT and SYS$ERROR.
;    92          0091  1         tbk$out_put : novalue,
;    93          0092  1         TBK$FAO_OUT : NOVALUE;          ! Formatted output routine.
;    94          0093  1
;    95          0094  1 !
;    96          0095  1 ! Include files:
;    97          0096  1 !
;    98          0097  1 REQUIRE 'SRC$:TBKPROLOG.REQ';
;    99          0369  1
;   100          0370  1 !
;   101          0371  1 ! External symbols:
;   102          0372  1 !
;   103          0373  1 EXTERNAL ROUTINE
;   104          0374  1         sys$putmsg: ADDRESSING_MODE (GENERAL),  ! write out messages.
;   105          0375  1         sys$trnlog: ADDRESSING_MODE (GENERAL),  ! translate logical names
;   106          0376  1         sys$faol: ADDRESSING_MODE (GENERAL);    ! Format an ASCII string
;   107          0377  1
;   108          0378  1 EXTERNAL
;   109          0379  1         tbk$cp_out_str : REF VECTOR[, BYTE],            ! POINTS INTO CURRENT OUTPUT BUFFER.
;   110          0380  1         tbk$gl_buf_siz,                 ! holds current character count in output buffer
;   111          0381  1         tbk$output_buf : VECTOR[,byte],
;   112          0382  1         tbk$gl_outprab: BLOCK [, BYTE],         ! RAB for 'SYS$OUTPUT'
;   113          0383  1         tbk$gl_errrab: BLOCK [, BYTE];          ! RAB for 'SYS$ERROR'
```

```
;  115     0384  1 GLOBAL ROUTINE tbk$fao_out (string, arguments) : NOVALUE =
;  116     0385  1 !++
;  117     0386  1 ! Functional description:
;  118     0387  1 !       Sets up input and output string descriptors. Then calls
;  119     0388  1 !       FAO to format the string. Then sends the formatted string to
;  120     0389  1 !       SYS$OUTPUT, and then to SYS$ERROR as the two are not
;  121     0390  1 !       the same.
;  122     0391  1 !
;  123     0392  1 ! Formal parameters:
;  124     0393  1 !       string          - the address of a counted control string to FAO
;  125     0394  1 !       arguments       - may be absent. arguments to be applied to the
;  126     0395  1 !                         FAO control string.
;  127     0396  1 !
;  128     0397  1 ! Implicit inputs:
;  129     0398  1 !       The RABs have been set up in TBK$IO_SETUP.
;  130     0399  1 !
;  131     0400  1 ! Output parameters:
;  132     0401  1 !       none
;  133     0402  1 !
;  134     0403  1 ! Implicit outputs:
;  135     0404  1 !       none.
;  136     0405  1 !
;  137     0406  1 ! Routine value:
;  138     0407  1 !       novalue
;  139     0408  1 !
;  140     0409  1 ! Side effects:
;  141     0410  1 !       If either $PUT fails we cause an $EXIT
;  142     0411  1 !       with the associated status code.
;  143     0412  1 !--
;  144     0413  1
;  145     0414  2 BEGIN
;  146     0415  2
;  147     0416  2         MAP
;  148     0417  2                 string          : REF VECTOR [, BYTE];
;  149     0418  2
;  150     0419  2         LOCAL
;  151     0420  2                 out_buf         : VECTOR [tty_out_width, BYTE],
;  152     0421  2                 inp_desc        : VECTOR [2],
;  153     0422  2                 out_desc        : VECTOR [2],
;  154     0423  2                 status;
;  155     0424  2
;  156     0425  2         ! Set up the needed string descriptors
;  157     0426  2         ! and do the formatting.
;  158     0427  2
;  159     0428  2         inp_desc [0] = .string [0];
;  160     0429  2         inp_desc [1] = string [1];
;  161     0430  2         out_desc [0] = tty_out_width;
;  162     0431  2         out_desc [1] = out_buf;
;  163     0432  2         sys$faol (inp_desc, tbk$gl_outprab [rab$w_rsz], out_desc, arguments);
;  164     0433  2
;  165     0434  2         ! Output always goes to SYS$OUTPUT.
;  166     0435  2
;  167     0436  2         tbk$gl_outprab [rab$l_rbf] = out_buf;
;  168     0437  2         status = $PUT (RAB = tbk$gl_outprab);
;  169     0438  2
;  170     0439  2         IF .status EQL RMS$_RSA                 ! Record stream active error
;  171     0440  2         THEN
```

```
;  172          0441  3            BEGIN
;  173          0442  3            $WAIT (RAB = tbk$gl_outprab);
;  174          0443  3            status = $PUT (RAB = tbk$gl_outprab);
;  175          0444  2            END;
;  176          0445  2
;  177          0446  2
;  178          0447  2        !+
;  179          0448  2        ! Output only goes to SYS$ERROR if its ISI
;  180          0449  2        ! number is different from SYS$OUTPUT's.
;  181          0450  2        !-
;  182          0451  3        IF( .TBK$GL_OUTPRAB[ RAB$W_ISI ] NEQ .TBK$GL_ERRRAB[ RAB$W_ISI ] )
;  183          0452  2        THEN
;  184          0453  3                BEGIN
;  185          0454  3                tbk$gl_errrab [ rab$w_rsz ] = .tbk$gl_outprab[ rab$w_rsz ];
;  186          0455  3                tbk$gl_errrab [rab$l_rbf] = out_buf;
;  187          0456  3                status = $PUT (RAB = tbk$gl_errrab);
;  188          0457  3                IF .status EQL RMS$_RSA         ! Record stream active error
;  189          0458  3                THEN
;  190          0459  4                    BEGIN
;  191          0460  4                    $WAIT (RAB = tbk$gl_errrab);
;  192          0461  4                    status = $PUT (RAB = tbk$gl_errrab);
;  193          0462  3                    END;
;  194          0463  2                END;
;  195          0464  2
;  196          0465  1 END;


                                                .TITLE  TBKSSV
                                                .IDENT  \V04-000\

                                                .EXTRN  SYS$PUTMSG, SYS$TRNLOG
                                                .EXTRN  SYS$FAOL, TBK$CP_OUT_STR
                                                .EXTRN  TBK$GL_BUF_SIZ, TBK$OUTPUT_BUF
                                                .EXTRN  TBK$GL_OUTPRAB, TBK$GL_ERRRAB
                                                .EXTRN  SYS$PUT, SYS$WAIT

                                                .PSECT  TBK$CODE,NOWRT,  SHR,  PIC,0

                                    007C 00000  .ENTRY  TBK$FAO_OUT, Save R2,R3,R4,R5,R6            ; 0384
                          56 00000000G  00  9E 00002  MOVAB   SYS$WAIT, R6                          ;
                          55 00000000G  00  9E 00009  MOVAB   SYS$PUT, R5                           ;
                          54 00000000G  00  9E 00010  MOVAB   TBK$GL_ERRRAB, R4                     ;
                          53 00000000G  00  9E 00017  MOVAB   TBK$GL_OUTPRAB, R3                    ;
                          5E      FF70  CE  9E 0001E  MOVAB   -144(SP), SP                          ;
                    04    AE        04  BC  9A 00023  MOVZBL  @STRING, INP_DESC                     ; 0428
08  AE                04  AC          01  C1 00028  ADDL3   #1, STRING, INP_DESC+4                ; 0429
                      7E        84  8F  9A 0002E  MOVZBL  #132, OUT_DESC                          ; 0430
                    04  AE        10  AE  9E 00032  MOVAB   OUT_BUF, OUT_DESC+4                   ; 0431
                                08  AC  9F 00037  PUSHAB  ARGUMENTS                               ; 0432
                                04  AE  9F 0003A  PUSHAB  OUT_DESC                               ;
                                22  A3  9F 0003D  PUSHAB  TBK$GL_OUTPRAB+34                      ;
                                14  AE  9F 00040  PUSHAB  INP_DESC                               ;
                000C0000C   00          04  FB 00043  CALLS   #4, SYS$FAOL                       ;
                    28  A3        10  AE  9E 0004A  MOVAB   OUT_BUF, TBK$GL_OUTPRAB+40           ; 0436
                                    53  DD 0004F  PUSHL   R3                                    ; 0437
                          65            01  FB 00051  CALLS   #1, SYS$PUT                        ;
                          52            50  D0 00054  MOVL    R0, STATUS                         ;
```

```
                000182DA   8F            52  D1 00057          CMPL    STATUS, #99034                          ; 0439
                                         0D  12 0005E          BNEQ    1$                                      ;
                                         53  DD 00060          PUSHL   R3                                      ; 0442
                           66            01  FB 00062          CALLS   #1, SYS$WAIT                            ;
                                         53  DD 00065          PUSHL   R3                                      ; 0443
                           65            01  FB 00067          CALLS   #1, SYS$PUT                             ;
                           52            50  D0 0006A          MOVL    R0, STATUS                              ;
                     02    A4      02    A3  B1 0006D 1$:      CMPW    TBK$GL_OUTPRAB+2, TBK$GL_ERRRAB+2       ; 0451
                                         28  13 00072          BEQL    2$                                      ;
                     22    A4      22    A3  B0 00074          MOVW    TBK$GL_OUTPRAB+34, TBK$GL_ERRRAB+34     ; 0454
                     28    A4      10    AE  9E 00079          MOVAB   OUT_BUF, TBK$GL_ERRRAB+40               ; 0455
                                         54  DD 0007E          PUSHL   R4                                      ; 0456
                           65            01  FB 00080          CALLS   #1, SYS$PUT                             ;
                           52            50  D0 00083          MOVL    R0, STATUS                              ;
                000182DA   8F            52  D1 00086          CMPL    STATUS, #99034                          ; 0457
                                         0D  12 0008D          BNEQ    2$                                      ;
                                         54  DD 0008F          PUSHL   R4                                      ; 0460
                           66            01  FB 00091          CALLS   #1, SYS$WAIT                            ;
                                         54  DD 00094          PUSHL   R4                                      ; 0461
                           65            01  FB 00096          CALLS   #1, SYS$PUT                             ;
                           52            50  D0 00099          MOVL    R0, STATUS                              ;
                                             04 0009C 2$:      RET                                             ; 0465

; Routine Size:  157 bytes,    Routine Base:  TBK$CODE + 0000
```

```
;  198          0466  1 GLOBAL ROUTINE TBK$OUT_PUT  : NOVALUE =
;  199          0467  1 !++
;  200          0468  1 ! Functional Description:
;  201          0469  1 !       Cause the current output buffer to be output
;  202          0470  1 !       to SYS$OUTPUT [and to SYS$ERROR if the latter is
;  203          0471  1 !       different from the former].
;  204          0472  1 !
;  205          0473  1 !       Also do the needed initialization so that a subsequent
;  206          0474  1 !       call to TBK$FAO_PUT will go as expected.
;  207          0475  1 !
;  208          0476  1 ! Calling Sequence:
;  209          0477  1 !       TBK$OUT_PUT ()
;  210          0478  1 !
;  211          0479  1 ! Inputs:
;  212          0480  1 !       none.
;  213          0481  1 !
;  214          0482  1 ! Implicit Inputs:
;  215          0483  1 !       TBK$OUTPUT_BUF  - is a pointer to the beginning of the current
;  216          0484  1 !                         output buffer.  This is supposed to be a counted
;  217          0485  1 !                         string except that no one has supplied the count yet.
;  218          0486  1 !                         i.e. we expect that the actual string to be printed
;  219          0487  1 !                         starts in byte TBK$OUTPUT_BUF+1.
;  220          0488  1 !       tbk$cp_out_str - points into this output buffer at the first
;  221          0489  1 !                        place in the buffer which is NOT to be printed.
;  222          0490  1 !       tbk$gl_buf_siz - holds count of characters in buffer
;  223          0491  1 !
;  224          0492  1 ! Outputs:
;  225          0493  1 !       Except for the leading <cr><lf> (by definition of SYS$OUTPUT),
;  226          0494  1 !       the string is printed, exactly as it is in the
;  227          0495  1 !       buffer - ie, it should contain whatever carriage control
;  228          0496  1 !       you want.
;  229          0497  1 !
;  230          0498  1 ! Implicit Outputs:
;  231          0499  1 !       A setup is done for subsequent I/O.
;  232          0500  1 !
;  233          0501  1 ! Routine Value:
;  234          0502  1 !       NOVALUE.
;  235          0503  1 !
;  236          0504  1 ! Side Effects.
;  237          0505  1 !       1 or 2 lines of output are $PUT.
;  238          0506  1 !--
;  239          0507  1
;  240          0508  2         BEGIN
;  241          0509  2
;  242          0510  2         ! Fill in the count, and pass it to QIO.
;  243          0511  2
;  244          0512  2         tbk$output_buf [0] = .tbk$gl_buf_siz;
;  245          0513  2         tbk$fao_out( uplit( %ascic '!AC'), tbk$output_buf );
;  246          0514  2
;  247          0515  2         tbk$cp_out_str = tbk$output_buf +1;
;  248          0516  2         tbk$gl_buf_siz = 0;
;  249          0517  1         END;


                                                        .PSECT  TBK$PLIT,NOWRT,  SHR,  PIC,0
```

```
                                43  41  21  03  00000 P.AAA:  .ASCII  <3>\!AC\                                    ;


                                                            .PSECT  TBK$CODE,NOWRT,  SHR,  PIC,0

                                                 000C 00000 .ENTRY  TBK$OUT_PUT, Save R2,R3                      ; 0466
                            53 00000000G   00    9E 00002 MOVAB   TBK$GL_BUF_SIZ, R3                         ;
                            52 00000000G   00    9E 00009 MOVAB   TBK$OUTPUT_BUF, R2                         ;
                            62                   63 90 00010 MOVB    TBK$GL_BUF_SIZ, TBK$OUTPUT_BUF         ; 0512
                                                 52 DD 00013 PUSHL   R2                                      ; 0513
                                      0000'      CF 9F 00015 PUSHAB  P.AAA                                   ;
                                FF45  CF         02 FB 00019 CALLS   #2, TBK$FAO_OUT                         ;
                         00000000G    00    01    A2 9E 0001E MOVAB   TBK$OUTPUT_BUF+1, TBK$CP_OUT_STR       ; 0515
                                                 63 D4 00026 CLRL    TBK$GL_BUF_SIZ                          ; 0516
                                                    04 00028 RET                                             ; 0517

; Routine Size:  41 bytes,    Routine Base:  TBK$CODE + 009D
```

```
;  251          0518  1 GLOBAL ROUTINE tbk$fake_msg( exception_name, parameter ) : novalue =
;  252          0519  1 !++
;  253          0520  1 ! Functional Description:
;  254          0521  1 !
;  255          0522  1 !       Put out TRACE 'error' messages via SYS$PUTMSG.
;  256          0523  1 !
;  257          0524  1 !       The reason we specify that this is via SYS$PUTMSG is that
;  258          0525  1 !       we want IT to decide where the output should go to.
;  259          0526  1 !       The reason we can't use PUTMSG directly is that it
;  260          0527  1 !       wants the argument of a signal array as its parameter
;  261          0528  1 !       when we simply have a message number and must therefore
;  262          0529  1 !       fake the signal.
;  263          0530  1 !       The reason we can't use LIB$SIGNAL to put out the
;  264          0531  1 !       message is 1) we don't want a signal to be generated
;  265          0532  1 !                     on top of the signal we are already handling,
;  266          0533  1 !       and
;  267          0534  1 !                  2) LIB$SIGNAL's signal leaves it up to someone else
;  268          0535  1 !                     (namely CLI's default frame or last chance handler)
;  269          0536  1 !                     to decide where the output goes to, and we
;  270          0537  1 !                     don't want to loose control of this, and
;  271          0538  1 !                     more importantly, this someone else takes it
;  272          0539  1 !                     upon itself to decide whether to give us
;  273          0540  1 !                     back control again - we ALWAYS want control back
;  274          0541  1 !                     when we do TBK$PUTMSG.
;  275          0542  1 !
;  276          0543  1 ! Formal Parameters:
;  277          0544  1 !
;  278          0545  1 !       exception_name  -The error message longword
;  279          0546  1 !       parameter       -If non-zero, this parameter is passed on
;  280          0547  1 !                        to PUTMSG in the signal arg list.
;  281          0548  1 !
;  282          0549  1 ! Implicit Inputs:
;  283          0550  1 !
;  284          0551  1 !       At most 1 parameter will be passed along with the
;  285          0552  1 !       signal array list.
;  286          0553  1 !
;  287          0554  1 ! Side Effects:
;  288          0555  1 !
;  289          0556  1 !       The message gets put out to the same place that all TRACE's
;  290          0557  1 !       other stuff goes to.
;  291          0558  1 !
;  292          0559  1 !       The usual action taken on messages, based on their severity
;  293          0560  1 !       level, is overridden here - we always get control back
;  294          0561  1 !       and continue TRACEback.
;  295          0562  1 !
;  296          0563  1 !--
;  297          0564  1
;  298          0565  2 BEGIN
;  299          0566  2         LOCAL
;  300          0567  2                                 ! 'FAKE' signal array to build
;  301          0568  2                                 ! args to SYS$PUTMSG in.
;  302          0569  2                 SIGNAL_ARRAY : VECTOR[6,LONG];
;  303          0570  2
;  304          0571  2         ! There must be atleast 2 entries in the
;  305          0572  2         ! signal array.  The 0th element is the
;  306          0573  2         ! count of the rest, the next is always the
;  307          0574  2         ! exception name. Subsequent ones are
```

```
;   308          0575  2          ! supposed parameters to the error message
;   309          0576  2          ! along with their FAO arg count.
;   310          0577  2          ! We only support having 0 or 1 associated
;   311          0578  2          ! parameters becasue that's all we need.
;   312          0579  2
;   313          0580  2          signal_array[0] = 1;
;   314          0581  2          signal_array[1] = .exception_name;
;   315          0582  2
;   316          0583  2          ! If we got an arg, we pass it on.
;   317          0584  2
;   318          0585  3          IF( .PARAMETER NEQ 0 )
;   319          0586  2          THEN
;   320          0587  3                  BEGIN
;   321          0588  3
;   322          0589  3                  ! Adding 1 arg means upping the total arg count
;   323          0590  3                  ! by two because we stuff in the new arg, and
;   324          0591  3                  ! its count (1), too.
;   325          0592  3
;   326          0593  3                  SIGNAL_ARRAY[0] = .SIGNAL_ARRAY[0] + 2;
;   327          0594  3                  signal_array[2] = 1;
;   328          0595  3                  SIGNAL_ARRAY[3] = .PARAMETER;
;   329          0596  2                  END;
;   330          0597  2
;   331          0598  2          ! Now we can just let the system routine
;   332          0599  2          ! take care of formatting and writing out
;   333          0600  2          ! the message.
;   334          0601  2
;   335          0602  2          SYS$PUTMSG( signal_array, 0, 0, 0 );
;   336          0603  1 END;


                                                0000 00000          .ENTRY  TBK$FAKE_MSG, Save nothing                      ; 0518
                                          5E    14   C2 00002          SUBL2   #20, SP                                         ;
                                                01   DD 00005          PUSHL   #1                                              ; 0580
                                    04    AE    04   AC   D0 00007    MOVL    EXCEPTION_NAME, SIGNAL_ARRAY+4                  ; 0581
                                                08   AC   D5 0000C    TSTL    PARAMETER                                       ; 0585
                                                0C   13 0000F          BEQL    1$                                              ;
                                          6E    02   C0 00011          ADDL2   #2, SIGNAL_ARRAY                                ; 0593
                                    08    AE    01   D0 00014          MOVL    #1, SIGNAL_ARRAY+8                              ; 0594
                                    0C    AE    08   AC   D0 00018    MOVL    PARAMETER, SIGNAL_ARRAY+12                      ; 0595
                                                7E   7C 0001D 1$:      CLRQ    -(SP)                                           ; 0602
                                                7E   D4 0001F          CLRL    -(SP)                                           ;
                                                0C   AE   9F 00021     PUSHAB  SIGNAL_ARRAY                                    ;
                                  00000000G     00        04   FB 00024     CALLS   #4, SYS$PUTMSG                          ;
                                                            04 0002B     RET                                                     ; 0603

; Routine Size:  44 bytes,    Routine Base:  TBK$CODE + 00C6
```

```
;  338     0604  1 GLOBAL ROUTINE TBK$PUT_MSG (sig_arg_list) =
;  339     0605  1 !++
;  340     0606  1 ! Functional description:
;  341     0607  1 !       Reports a system-generated message by calling SYS$PUTMSG.
;  342     0608  1 !       This routine checks the exception name to see if the exception is not
;  343     0609  1 !       a hardware exception.  If it is not a hardware exception 2 is subtracted
;  344     0610  1 !       from the signal argument list count before calling SYS$PUTMSG.
;  345     0611  1 !       After SYS$PUTMSG returns the original count is restored.
;  346     0612  1 !
;  347     0613  1 ! Inputs:
;  348     0614  1 !       sig_arg_list    - the address of the signal argument list.
;  349     0615  1 !
;  350     0616  1 ! Implicit inputs:
;  351     0617  1 !       None
;  352     0618  1 !
;  353     0619  1 ! Outputs:
;  354     0620  1 !       none
;  355     0621  1 !
;  356     0622  1 ! Implicit outputs:
;  357     0623  1 !       none
;  358     0624  1 !
;  359     0625  1 ! Routine value:
;  360     0626  1 !       The status value returned by SYS$PUTMSG.
;  361     0627  1 !
;  362     0628  1 ! Side effects:
;  363     0629  1 !       SYS$PUTMSG is called - this defines the process
;  364     0630  1 !       logical name upon which TBK$IO_SETUP depends.
;  365     0631  1 !
;  366     0632  1 !       The message is output.
;  367     0633  1 !--
;  368     0634  1
;  369     0635  2 BEGIN
;  370     0636  2
;  371     0637  2         LOCAL
;  372     0638  2                 status,
;  373     0639  2                 orig_arg_count,
;  374     0640  2                 index,
;  375     0641  2                 excep_name : BLOCK [%UPVAL, BYTE],
;  376     0642  2                 table_value : BLOCK [%UPVAL, BYTE];
;  377     0643  2
;  378     0644  2         MAP
;  379     0645  2                 sig_arg_list : REF VECTOR;
;  380     0646  2
;  381     0647  2         BIND
;  382     0648  2                 hardware_excep = UPLIT WORD (ss$_accvio, ss$_artres, ss$_intovf,
;  383     0649  2                                              ss$_intdiv, ss$_fltovf, ss$_fltdiv, ss$_fltund,
;  384     0650  2                                              ss$_decovf, ss$_subrng, ss$_astflt, ss$_break,
;  385     0651  2                                              ss$_cmodsupr, ss$_cmoduser, ss$_compat,
;  386     0652  2                                              ss$_debug, ss$_opccus, ss$_opcdec, ss$_pagrderr,
;  387     0653  2                                              ss$_radrmod, ss$_roprand, ss$_ssfail, ss$_tbit,
;  388     0654  2                                              0) : VECTOR [, WORD];
;  389     0655  2
;  390     0656  2         orig_arg_count = .sig_arg_list[0];              ! Get orignial arg. count
;  391     0657  2         excep_name = .sig_arg_list[1];                  ! Get exception name
;  392     0658  3         IF (.excep_name[STS$V_FAC_NO] NEQ 0)            ! Not SYSTEM facility
;  393     0659  2         THEN
;  394     0660  3                 BEGIN
```

```
;  395     0661  3                 sig_arg_list[0] = .sig_arg_list[0] - 2; ! update argument count
;  396     0662  3                 END
;  397     0663  2         ELSE
;  398     0664  3                 BEGIN
;  399     0665  3                 index = 0;
;  400     0666  3                 !+
;  401     0667  3                 ! This loop will exit with -1 if the exception name is not found.
;  402     0668  3                 ! In that case we must subtract 2 from the signal argument list
;  403     0669  3                 ! argument count before calling SYS$PUTMSG.
;  404     0670  3                 !-
;  405     0671  5                 IF (WHILE (.hardware_excep[.index] NEQ 0)
;  406     0672  4                     DO
;  407     0673  5                     BEGIN
;  408     0674  5                     table_value = .hardware_excep[.index]; ! pick up next value
;  409     0675  6                     IF (.excep_name[STS$V_CODE] EQL .table_value[STS$V_CODE])
;  410     0676  5                     THEN
;  411     0677  5                         EXITLOOP 0;
;  412     0678  5
;  413     0679  5                     index = .index + 1;
;  414     0680  4                     END)
;  415     0681  3                 THEN
;  416     0682  3                     sig_arg_list[0] = .sig_arg_list[0] - 2;
;  417     0683  2                 END;
;  418     0684  2
;  419     0685  2         ! Now, finally, output the message.
;  420     0686  2
;  421     0687  2         status = SYS$PUTMSG (.sig_arg_list, 0, 0);
;  422     0688  2         sig_arg_list[0] = .orig_arg_count;
;  423     0689  2
;  424     0690  2         ! Return the status which PUTMSG returned.
;  425     0691  2
;  426     0692  2         RETURN(.STATUS);
;  427     0693  2
;  428     0694  1 END;


                                                        .PSECT  TBK$PLIT,NOWRT,  SHR,  PIC,0

040C  04AC  04A4  049C  0494  048C  0484  047C  0474  000C  00004 P.AAB:  .WORD   12, 1140, 1148, 1156, 1164, 1172, 1180, -      ;
0454  044C  0444  043C  0434  046C  042C  0424  041C  0414  00018                 1188, 1196, 1036, 1044, 1052, 1060, 1068, - ;
                                      0000  0464  045C  0002C                   1132, 1076, 1084, 1092, 1100, 1108, 1116, - ;
                                                                                1124, 0                                     ;

                                                                HARDWARE_EXCEP=     P.AAB

                                                        .PSECT  TBK$CODE,NOWRT,  SHR,  PIC,0

                                                  003C 00000            .ENTRY  TBK$PUT_MSG, Save R2,R3,R4,R5                ; 0604
                                        52    04  AC  D0 00002            MOVL    SIG_ARG_LIST, R2                     ; 0656
                                        55        62  D0 00006            MOVL    (R2), ORIG_ARG_COUNT                 ;
                                        53    04  A2  D0 00009            MOVL    4(R2), EXCEP_NAME                    ; 0657
                00              53        0C        10  ED 0C00D          CMPZV   #16, #12, EXCEP_NAME, #0             ; 0658
                                                    1C  12 00012          BNEQ    2$                                   ;
                                                    50  D4 00014          CLRL    INDEX                                ; 0665
                                        51  0000'CF40  3C 00016 1$:       MOVZWL  HARDWARE_EXCEP[INDEX], R1            ; 0671
```

```
                                        12  13 0001C          BEQL    2$                                                      ;
                                54      51  D0 0001E          MOVL    R1, TABLE_VALUE                                         ; 0674
                          54    54      53  AD 00021          XORW3   EXCEP_NAME, TABLE_VALUE, R1                             ; 0675
                    7FF8  8F            51  B3 00025          BITW    R1, #32760                                              ;
                                        07  13 0002A          BEQL    3$                                                      ;
                                        50  D6 0002C          INCL    INDEX                                                   ; 0679
                                        E6  11 0002E          BRB     1$                                                      ; 0671
                                62      02  C2 00030 2$:      SUBL2   #2, (R2)                                                ; 0682
                                        7E  7C 00033 3$:      CLRQ    -(SP)                                                   ; 0687
                                        52  DD 00035          PUSHL   R2                                                      ;
                00000000G       00      03  FB 00037          CALLS   #3, SYS$PUTMSG                                          ;
                                62      55  D0 0003E          MOVL    ORIG_ARG_COUNT, (R2)                                    ; 0688
                                            04 00041          RET                                                             ; 0694

; Routine Size:  66 bytes,    Routine Base:  TBK$CODE + 00F2
```

```
;  430     0695  1 GLOBAL ROUTINE tbk$fao_put( STRING, ARGUMENTS ) : NOVALUE =
;  431     0696  1 !++
;  432     0697  1 ! FUNCTIONAL DESCRIPTION
;  433     0698  1 !       DO JUST WHAT $FAO DOES, ONLY HERE WE WORK IN
;  434     0699  1 !       CO-OPERATION WITH A GLOBAL CHARACTER BUFFER INTO
;  435     0700  1 !       WHICH WE ARE ENCODING ARBITRARY LINES OF OUTPUT.
;  436     0701  1 !
;  437     0702  1 !       All output done within TRACE should use this
;  438     0703  1 !       routine to build output lines.  Then, once the buffer
;  439     0704  1 !       has been filled, a call to TBK$OUT_PUT should be
;  440     0705  1 !       made to cause the $PUT.
;  441     0706  1 !
;  442     0707  1 ! FORMAL PARAMETERS:
;  443     0708  1 !       STRING    - A COUNTED STRING WHICH CONTAINS THE DIRECTIVES FOR $FAO.
;  444     0709  1 !       ARGUMENTS - THE ARGS FOR $FAO.
;  445     0710  1 !
;  446     0711  1 ! IMPLICIT INPUTS:
;  447     0712  1 !       tbk$cp_out_str  - POINTER TO WHERE WE ARE IN THE
;  448     0713  1 !                         CURRENT OUTPUT BUFFER.
;  449     0714  1 !       tbk$gl_buf_siz  - count of characters in output buffer.
;  450     0715  1 !
;  451     0716  1 ! OUTPUTS:
;  452     0717  1 !       THE $FAO OUTPUT IS PUT INTO THE OUTPUT BUFFER.
;  453     0718  1 !
;  454     0719  1 ! IMPLICIT OUTPUTS:
;  455     0720  1 !       THE GLOBAL CHARACTER POINTER IS INCREMENTED SO THAT IT
;  456     0721  1 !       POINTS (AS ALWAYS) TO THE NEXT AVAILABLE PLACE IN THE
;  457     0722  1 !       OUTPUT BUFFER. The buffer count variable is incremented
;  458     0723  1 !       by the size of this string.
;  459     0724  1 !
;  460     0725  1 ! ROUTINE VALUE:
;  461     0726  1 !       NONE.
;  462     0727  1 !
;  463     0728  1 ! SIDE EFFECTS:
;  464     0729  1 !       NONE.
;  465     0730  1 !--
;  466     0731  1
;  467     0732  2         BEGIN
;  468     0733  2         MAP
;  469     0734  2                 STRING : REF VECTOR[,BYTE];
;  470     0735  2         LOCAL
;  471     0736  2                 INP_DESC : VECTOR[2],           ! INPUT DESC FOR $FAO.
;  472     0737  2                 OUT_DESC : VECTOR[2],           ! OUTPUT DESC FOR $FAO.
;  473     0738  2                 STR_SIZE : WORD;                ! $FAO RETURNS OUTPUT SIZE HERE.
;  474     0739  2         !+
;  475     0740  2         ! BUILD THE DESCRIPTORS THAT $FAO WANTS, ASK IT TO DO
;  476     0741  2         ! THE ENCODING, COPYING THE OUTPUT INTO OUR GLOBAL
;  477     0742  2         ! OUTPUT BUFFER, AND FINALLY UPDATE THE GLOBAL POINTER
;  478     0743  2         ! TO THE NEXT FREE CHARACTER POSITION IN THE BUFFER.
;  479     0744  2         !-
;  480     0745  2
;  481     0746  2         INP_DESC[0] = .STRING[0];
;  482     0747  2         INP_DESC[1] = STRING[1];
;  483     0748  2         OUT_DESC [0] = TTY_OUT_WIDTH - 1 - .tbk$gl_buf_siz;
;  484     0749  2         OUT_DESC[1] = .tbk$cp_out_str;
;  485     0750  2         SYS$FAOL( INP_DESC, STR_SIZE, OUT_DESC, ARGUMENTS );
;  486     0751  2         tbk$cp_out_str = .tbk$cp_out_str + .STR_SIZE;
```

```
;  487          0752  2          tbk$gl_buf_siz = .tbk$gl_buf_siz + .STR_SIZE;
;  488          0753  1          END;


                                                000C 00000          .ENTRY  TBK$FAO_PUT, Save R2,R3                 ; 0695
                                    53 00000000G  00  9E 00002      MOVAB   TBK$GL_BUF_SIZ, R3                      ;
                                    52 00000000G  00  9E 00009      MOVAB   TBK$CP_OUT_STR, R2                      ;
                                    5E            14  C2 00010      SUBL2   #20, SP                                 ;
                                    0C    AE      04  BC 9A 00013   MOVZBL  @STRING, INP_DESC                       ; 0746
                              10    AE    04      AC  01  C1 00018  ADDL3   #1, STRING, INP_DESC+4                  ; 0747
                              04    AE 00000083   8F  63  C3 0001E  SUBL3   TBK$GL_BUF_SIZ, #131, OUT_DESC         ; 0748
                                    08    AE          62  D0 00027  MOVL    TBK$CP_OUT_STR, OUT_DESC+4              ; 0749
                                                08  AC  9F 0002B   PUSHAB  ARGUMENTS                               ; 0750
                                                08  AE  9F 0002E   PUSHAB  OUT_DESC                                ;
                                                08  AE  9F 00031   PUSHAB  STR_SIZE                                ;
                                                18  AE  9F 00034   PUSHAB  INP_DESC                                ;
                                 00000000G  00          04  FB 00037  CALLS   #4, SYS$FAOL                        ;
                                    50          6E  3C 0003E      MOVZWL  STR_SIZE, R0                            ; 0751
                                    62          50  C0 00041      ADDL2   R0, TBK$CP_OUT_STR                      ;
                                    50          6E  3C 00044      MOVZWL  STR_SIZE, R0                            ; 0752
                                    63          50  C0 00047      ADDL2   R0, TBK$GL_BUF_SIZ                      ;
                                                    04 0004A      RET                                             ; 0753

; Routine Size:  75 bytes,    Routine Base:  TBK$CODE + 0134
```

```
;  490     0754  1 GLOBAL ROUTINE TBK$IO_SETUP =
;  491     0755  1 !++
;  492     0756  1 ! Functional Description:
;  493     0757  1 !
;  494     0758  1 !       Set up the output FABs so that TRACE can
;  495     0759  1 !       output without opening or connecting to SYS$OUTPUT
;  496     0760  1 !       and/or SYS$ERROR.
;  497     0761  1 !
;  498     0762  1 !       This is done by assuning that SYS$PUTMSG has already
;  499     0763  1 !       done this AND that it has created a process logical
;  500     0764  1 !       name (SYS$PUTMSG) to allow us to 'steal' the ISI
;  501     0765  1 !       numbers we need to properly initialize the RABs.
;  502     0766  1 !
;  503     0767  1 ! Implicit Inputs:
;  504     0768  1 !
;  505     0769  1 !       The equivalence name for SYS$PUTMSG has the following
;  506     0770  1 !       format:
;  507     0771  1 !
;  508     0772  1 !               |  2 bytes    |  2 bytes    |  2 bytes    |
;  509     0773  1 !               |-------------|-------------|-------------|
;  510     0774  1 !               |  ERROR_ISI  |  OUTPUT_ISI |   ^x11B     |
;  511     0775  1 !               |-------------|-------------|-------------|
;  512     0776  1 !
;  513     0777  1 !       The ^X11B is simply a prefix made up in the SYS$PUTMSG
;  514     0778  1 !       creation-of-logical-name code to ensure that if
;  515     0779  1 !       we get a TRNLOG here we get what we expect.  If this
;  516     0780  1 !       prefix is NOT there we punt.
;  517     0781  1 !
;  518     0782  1 ! Side Effects:
;  519     0783  1 !
;  520     0784  1 !       The equivalence string for SYS$PUTMSG is requested
;  521     0785  1 !       from the 'process' logical name table.  From this 'fake'
;  522     0786  1 !       string is extracted the ISI numbers for SYS$ERROR and
;  523     0787  1 !       SYS$OUTPUT.  These ISIs are stuffed into the
;  524     0788  1 !       2 RABs we use for $PUTting.
;  525     0789  1 !--
;  526     0790  1
;  527     0791  2 BEGIN
;  528     0792  2         LOCAL
;  529     0793  2                                 ! Character string descriptor for
;  530     0794  2                                 ! SYS$PUTMSG logical name
;  531     0795  2                 putmsg_desc : vector[2,long],
;  532     0796  2
;  533     0797  2                                 ! Character string descriptor for resultant
;  534     0798  2                                 ! equivalent string (from TRNLOG)
;  535     0799  2                 equiv_desc : vector[2,long],
;  536     0800  2
;  537     0801  2                                 ! Equivalence name has a supposed
;  538     0802  2                                 ! fixed format.  (See above).
;  539     0803  2                 equiv_string : vector[8,word],
;  540     0804  2
;  541     0805  2                 status;
;  542     0806  2
;  543     0807  2
;  544     0808  2
;  545     0809  2         ! Set up a descriptor for the logical name
;  546     0810  2         ! SYS$PUTMSG and for the equivalence name
```

```
;   547          0811  2          ! we assume SYS$PUTMSG has created, and then
;   548          0812  2          ! translate this name to its 'equivalence' string.
;   549          0813  2
;   550          0814  2          PUTMSG_DESC[0] = %CHARCOUNT (%ASCII 'SYS$PUTMSG' );
;   551          0815  2          PUTMSG_DESC[1] = UPLIT BYTE (%ASCII 'SYS$PUTMSG' );
;   552          0816  2          EQUIV_DESC[0] = 16;              ! Special name fits into 8 words.
;   553          0817  2          EQUIV_DESC[1] = EQUIV_STRING;
;   554          0818  2
;   555          0819  2          STATUS = SYS$TRNLOG(    PUTMSG_DESC,
;   556          0820  2                                  0,              ! Ignore resultant length
;   557          0821  2                                  EQUIV_DESC,     ! Resultant string descriptor.
;   558          0822  2                                  0,              ! Ignore "found" table.
;   559          0823  2                                  0,              ! Ignore access mode
;   560          0824  2                                  LOG$C_PROCESS   ! Look in "process" table only
;   561          0825  2                              );
;   562          0826  2          IF .STATUS EQL SS$_NOTRAN
;   563          0827  2          THEN
;   564          0828  2              RETURN (.STATUS);           ! No translation.
;   565          0829  3          IF(NOT .STATUS EQL SS$_NORMAL)
;   566          0830  2          THEN
;   567          0831  2                  $EXIT( code = .STATUS );
;   568          0832  2                  ! no return
;   569          0833  2
;   570          0834  2          ! Check for the special identifying word
;   571          0835  2          ! in the equivalence string.
;   572          0836  2
;   573          0837  3          IF( .EQUIV_STRING[0] NEQ %X'11B' )
;   574          0838  2          THEN
;   575          0839  2                  $EXIT( code = TBK$_NOIOCHAN );
;   576          0840  2
;   577          0841  2          ! Initialize what is needed by the rest of
;   578          0842  2          ! TRACE's output routines. i.e. The output buffer
;   579          0843  2          ! pointers, and the ISI numbers we stuff
;   580          0844  2          ! into the RABs we use for SYS$ERROR and SYS$OUTPUT.
;   581          0845  2
;   582          0846  2          tbk$gl_outprab[ RAB$W_ISI ] = .EQUIV_STRING[1];
;   583          0847  2          tbk$gl_errrab[ RAB$W_ISI ] = .EQUIV_STRING[2];
;   584          0848  2
;   585          0849  2          tbk$cp_out_str = tbk$output_buf +1;
;   586          0850  2          tbk$gl_buf_siz = 0;
;   587          0851  2
;   588          0852  2          RETURN (.STATUS);
;   589          0853  2
;   590          0854  1 END;


                                                        .PSECT  TBK$PLIT,NOWRT,  SHR,  PIC,0

                47  53  4D  54  55  50  24  53  59  53  00032 P.AAC:  .ASCII  \SYS$PUTMSG\                          ;

                                                        .EXTRN  SYS$EXIT

                                                        .PSECT  TBK$CODE,NOWRT,  SHR,  PIC,0

                                                000C 00000          .ENTRY  TBK$IO_SETUP, Save R2,R3                ; 0754
                                    53 00000000G  00   9E 00002     MOVAB   SYS$EXIT, R3                           ;
```

```
              5E                20  C2 00009         SUBL2   #32, SP                                        ;
        18    AE                0A  D0 0000C         MOVL    #10, PUTMSG_DESC                               ; 0814
        1C    AE        0000'   CF  9E 00010         MOVAB   P.AAC, PUTMSG_DESC+4                           ; 0815
        10    AE                10  D0 00016         MOVL    #16, EQUIV_DESC                                ; 0816
        14    AE                6E  9E 0001A         MOVAB   EQUIV_STRING, EQUIV_DESC+4                     ; 0817
                                02  DD 0001E         PUSHL   #2                                             ; 0819
                                7E  7C 00020         CLRQ    -(SP)                                          ;
                          1C    AE  9F 00022         PUSHAB  EQUIV_DESC                                     ;
                                7E  D4 00025         CLRL    -(SP)                                          ;
                          2C    AE  9F 00027         PUSHAB  PUTMSG_DESC                                    ;
  00000000G   00                06  FB 0002A         CALLS   #6, SYS$TRNLOG                                 ;
              52                50  D0 00031         MOVL    R0, STATUS                                     ;
  00000629    8F                52  D1 00034         CMPL    STATUS, #1577                                  ; 0826
                                3B  13 0003B         BEQL    3$                                             ;
              01                52  D1 0003D         CMPL    STATUS, #1                                     ; 0829
                                05  13 00040         BEQL    1$                                             ;
                                52  DD 00042         PUSHL   STATUS                                         ; 0831
              63                01  FB 00044         CALLS   #1, SYS$EXIT                                   ;
      011B    8F                6E  B1 00047 1$:     CMPW    EQUIV_STRING, #283                             ; 0837
                                09  13 0004C         BEQL    2$                                             ;
                    000984B4    8F  DD 0004E         PUSHL   #623796                                        ; 0839
              63                01  FB 00054         CALLS   #1, SYS$EXIT                                   ;
  00000000G   00          02    AE  B0 00057 2$:     MOVW    EQUIV_STRING+2, TBK$GL_OUTPRAB+2               ; 0846
  00000000G   00          04    AE  B0 0005F         MOVW    EQUIV_STRING+4, TBK$GL_ERRRAB+2                ; 0847
  00000000G   00  00000000G     00  9E 00067         MOVAB   TBK$OUTPUT_BUF+1, TBK$CP_OUT_STR               ; 0849
                  00000000G     00  D4 00072         CLRL    TBK$GL_BUF_SIZ                                 ; 0850
              50                52  D0 00078 3$:     MOVL    STATUS, R0                                     ; 0852
                                    04 0007B         RET                                                    ; 0854

; Routine Size:  124 bytes,    Routine Base:  TBK$CODE + 017F
```

```
;  592          0855  1 END
;  593          0856  0 ELUDOM
```

```
;                                   PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  TBK$CODE                        507  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)
;  TBK$PLIT                         60  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(0)
```

```
;                                 Library Statistics
;
;                                        -------- Symbols --------      Pages       Processing
;       File                             Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1          18619        40         0        1000        00:01.9
;  _$255$DUA28:[TRACE.OBJ]TBKLIB.L32;1      157         1         0          14        00:00.3
;  _$255$DUA28:[TRACE.OBJ]STRUCDEF.L32;1     32         0         0           7        00:00.1
;  _$255$DUA28:[TRACE.OBJ]TBKDST.L32;1      414         0         0          30        00:00.3
```

```
;                                    COMMAND QUALIFIERS

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

; Size:          507 code + 60 data bytes
; Run Time:          00:16.5
; Elapsed Time:      00:52.7
; Lines/CPU Min:      3122
; Lexemes/CPU-Min:  25334
; Memory Used:  103 pages
; Compilation Complete
```

0402 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

DZVDRIVER
MAP

DZDRIVER
LIS

TBKSVM
LIS

YFDRIVER
MAP

DZVDRIVER
LIS

TBKSSV
LIS

TTDRVR

DZDRIVER
MAP

TTDRIVER
MAP

YCDRIVER
MAP

TBKSTO
LIS

TTYDEF
SDL

TTYMACS
MAR

TBKSTART
LIS