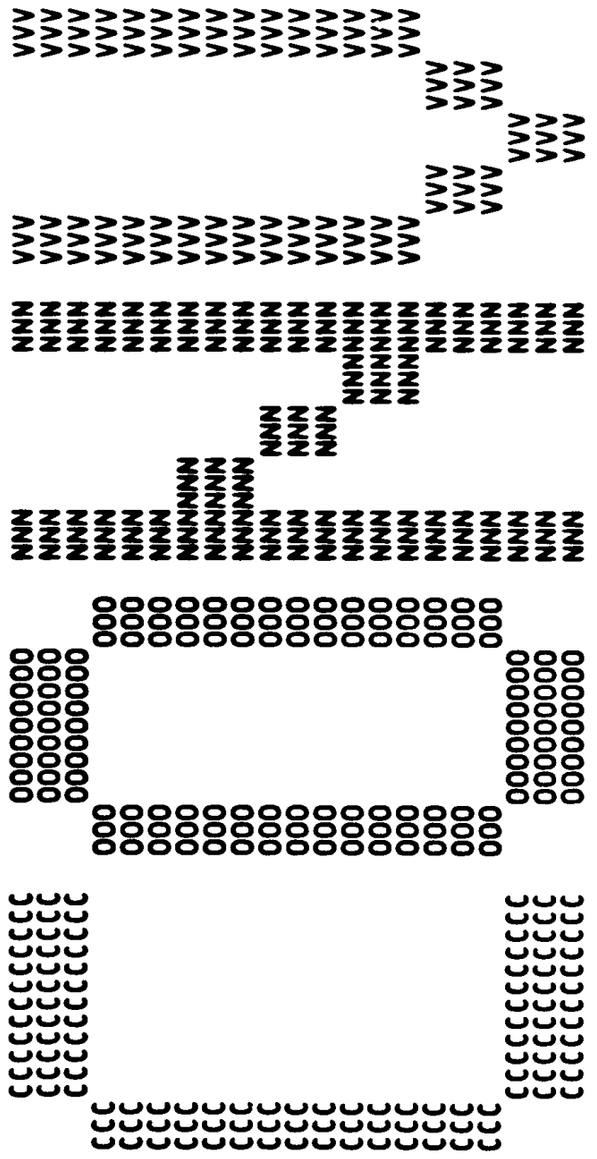

```
  CCCCCCCC    000000    NN      NN  VV      VV  EEEEEEEEEE  RRRRRRRR    RRRRRRRR      000000    RRRRRRRR
  CCCCCCCC    000000    NN      NN  VV      VV  EEEEEEEEEE  RRRRRRRR    RRRRRRRR      000000    RRRRRRRR
CC          00      00  NN      NN  VV      VV  EE          RR      RR  RR      RR  00      00  RR      RR
CC          00      00  NN      NN  VV      VV  EE          RR      RR  RR      RR  00      00  RR      RR
CC          00      00  NNNN    NN  VV      VV  EE          RR      RR  RR      RR  00      00  RR      RR
CC          00      00  NNNN    NN  VV      VV  EE          RR      RR  RR      RR  00      00  RR      RR
CC          00      00  NN  NN  NN  VV      VV  EEEEEEEE    RRRRRRRR    RRRRRRRR    00      00  RRRRRRRR
CC          00      00  NN  NN  NN  VV      VV  EEEEEEEE    RRRRRRRR    RRRRRRRR    00      00  RRRRRRRR
CC          00      00  NN    NNNN  VV      VV  EE          RR  RR      RR  RR      00      00  RR  RR
CC          00      00  NN    NNNN  VV      VV  EE          RR  RR      RR  RR      00      00  RR  RR
CC          00      00  NN      NN    VV  VV    EE          RR    RR    RR    RR    00      00  RR    RR      ....
CC          00      00  NN      NN    VV  VV    EE          RR    RR    RR    RR    00      00  RR    RR      ....
  CCCCCCCC    000000    NN      NN      VV      EEEEEEEEEE  RR      RR  RR      RR    000000    RR      RR    ....
  CCCCCCCC    000000    NN      NN      VV      EEEEEEEEEE  RR      RR  RR      RR    000000    RR      RR    ....

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 %TITLE  'VAX-11 CONVERT'
;     2        0002  0 MODULE  CONVSERROR      ( IDENT='V04-000'
;     3        0003  0                         ) =
;     4        0004  0
;     5        0005  1 BEGIN
;     6        0006  1
;     7        0007  1 !**************************************************************************
;     8        0008  1 !*                                                                        *
;     9        0009  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                               *
;    10        0010  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                 *
;    11        0011  1 !*  ALL RIGHTS RESERVED.                                                   *
;    12        0012  1 !*                                                                        *
;    13        0013  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
;    14        0014  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    15        0015  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;    16        0016  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;    17        0017  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;    18        0018  1 !*  TRANSFERRED.                                                           *
;    19        0019  1 !*                                                                        *
;    20        0020  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;    21        0021  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;    22        0022  1 !*  CORPORATION.                                                           *
;    23        0023  1 !*                                                                        *
;    24        0024  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
;    25        0025  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
;    26        0026  1 !*                                                                        *
;    27        0027  1 !*                                                                        *
;    28        0028  1 !**************************************************************************
```

```
;    30          0029  1 !++
;    31          0030  1 !
;    32          0031  1 ! Facility:     VAX-11 CONVERT
;    33          0032  1 !
;    34          0033  1 ! Environment:
;    35          0034  1 !
;    36          0035  1 !                VAX/VMS Operating System
;    37          0036  1 !
;    38          0037  1 ! Abstract:
;    39          0038  1 !
;    40          0039  1 !                CONVERT error handeling routines
;    41          0040  1 !
;    42          0041  1 ! Contents:
;    43          0042  1 !                RMS_ERROR
;    44          0043  1 !                RMS_OPEN_ERROR
;    45          0044  1 !                RMS_READ_ERROR
;    46          0045  1 !
;    47          0046  1 !--
;    48          0047  1
;    49          0048  1 !
;    50          0049  1 ! Author:       Keith B Thompson        Creation Date:  September-1981
;    51          0050  1 !
;    52          0051  1 !
;    53          0052  1 ! Modified by:
;    54          0053  1 !
;    55          0054  1 !       V03-001 RAS0273         Ron Schaefer            19-Mar-1984
;    56          0055  1 !               Fix CONV$$RMS_ERROR to have the correct signal arguments
;    57          0056  1 !               in order to return decent error messages.
;    58          0057  1 !
;    59          0058  1 !****
```

```
;    61          0059  1
;    62          0060  1 LIBRARY 'SYS$LIBRARY:LIB';
;    63          0061  1 REQUIRE 'SRC$:CONVERT';
;    64          0296  1
;    65          0297  1 OWN
;    66          0298  1         STRING_DESC     : DESC_BLK;
;    67          0299  1
```

```
;    69          0300  1 %SBTTL  'CONV$$RMS_ERROR'
;    70          0301  1 GLOBAL ROUTINE  CONV$$RMS_ERROR : NOVALUE =
;    71          0302  1 !++
;    72          0303  1 !
;    73          0304  1 ! Functional Description:
;    74          0305  1 !
;    75          0306  1 !       This routine will signal and rms error and stop execution.  It is
;    76          0307  1 !       to be primarly used for detecting errors during asynchronous operations
;    77          0308  1 !
;    78          0309  1 ! Calling Sequence:
;    79          0310  1 !
;    80          0311  1 !       This routine is call as an AST by RMS
;    81          0312  1 !
;    82          0313  1 ! Input Parameters:
;    83          0314  1 !
;    84          0315  1 !       AST argument block which has a pointer to a rms block
;    85          0316  1 !
;    86          0317  1 ! Implicit Inputs:
;    87          0318  1 !       none
;    88          0319  1 !
;    89          0320  1 ! Output Parameters:
;    90          0321  1 !       none
;    91          0322  1 !
;    92          0323  1 ! Implicit Outputs:
;    93          0324  1 !       none
;    94          0325  1 !
;    95          0326  1 ! Routine Value:
;    96          0327  1 !       none
;    97          0328  1 !
;    98          0329  1 ! Routines Called:
;    99          0330  1 !
;   100          0331  1 !       SIGNAL
;   101          0332  1 !       SIGNAL_STOP
;   102          0333  1 !
;   103          0334  1 ! Side Effects:
;   104          0335  1 !       none
;   105          0336  1 !
;   106          0337  1 !--
;   107          0338  1
;   108          0339  2     BEGIN
;   109          0340  2
;   110          0341  2     BUILTIN AP;
;   111          0342  2
;   112          0343  2     BIND
;   113          0344  2         AST_BLOCK = AP : REF VECTOR [ ,LONG ];
;   114          0345  2
;   115          0346  2     LOCAL
;   116          0347  2         RMS_BLOCK : REF BLOCK [ ,BYTE ],
;   117          0348  2         FAB       : REF BLOCK [ ,BYTE ],
;   118          0349  2         NAM       : REF BLOCK [ ,BYTE ];
;   119          0350  2
;   120          0351  2     ! Get the rms block (Pointed to by the second ast parameter)
;   121          0352  2     !
;   122          0353  2     RMS_BLOCK = .AST_BLOCK [ 1 ];
;   123          0354  2
;   124          0355  2     ! If this is really a RAB (from a connect) then get the fab it points to
;   125          0356  2     !
```

```
;  126          0357  2     IF .RMS_BLOCK [ RAB$B_BID ] EQLU RAB$C_BID
;  127          0358  2     THEN
;  128          0359  2         FAB = .RMS_BLOCK [ RAB$L_FAB ]
;  129          0360  2     ELSE
;  130          0361  2         FAB = .RMS_BLOCK;
;  131          0362  2
;  132          0363  2     ! Get the name block
;  133          0364  2     !
;  134          0365  2     NAM = .FAB [ FAB$L_NAM ];
;  135          0366  2
;  136          0367  2     ! Signal the CONVert error with the best file name string
;  137          0368  2     !
;  138          0369  2     ! First try the resultant string
;  139          0370  2     !
;  140          0371  2     IF .NAM [ NAM$B_RSL ] NEQU 0
;  141          0372  2     THEN
;  142          0373  3         BEGIN
;  143          0374  3         STRING_DESC [ DSC$W_LENGTH ] = .NAM [ NAM$B_RSL ];
;  144          0375  3         STRING_DESC [ DSC$A_POINTER ] = .NAM [ NAM$[_RSA ]
;  145          0376  3         END
;  146          0377  3
;  147          0378  3     ! Next try the expanded string
;  148          0379  3     !
;  149          0380  2     ELSE IF .NAM [ NAM$B_ESL ] NEQU 0
;  150          0381  2     THEN
;  151          0382  3         BEGIN
;  152          0383  3         STRING_DESC [ DSC$W_LENGTH ] = .NAM [ NAM$B_ESL ];
;  153          0384  3         STRING_DESC [ DSC$A_POINTER ] = .NAM [ NAM$[_ESA ]
;  154          0385  3         END
;  155          0386  3
;  156          0387  3     ! If all else fails use the name string
;  157          0388  3     !
;  158          0389  2     ELSE
;  159          0390  3         BEGIN
;  160          0391  3         STRING_DESC [ DSC$W_LENGTH ] = .FAB [ FAB$B_FNS ];
;  161          0392  3         STRING_DESC [ DSC$A_POINTER ] = .FAB [ FAB$[_FNA ]
;  162          0393  2         END;
;  163          0394  2
;  164          0395  2     ! NOTE: We use the RAB$x_zzz codes but they are valid for the FAB as well
;  165          0396  2     !
;  166          0397  2     ! Signal the CONVert error, RMS error and stop
;  167          0398  2     !
;  168          0399  2     SIGNAL_STOP( .RMS_BLOCK [ RAB$L_CTX ],1,STRING_DESC,
;  169          0400  2                  .RMS_BLOCK [ RAB$L_STS ],
;  170          0401  2                  .RMS_BLOCK [ RAB$L_STV ] )
;  171          0402  2
;  172          0403  1     END;

                                                        .TITLE  CONV$ERROR VAX-11 CONVERT
                                                        .IDENT  \V04-000\

                                                        .PSECT  $OWN$,NOEXE,2

                                                00000 STRING_DESC:
                                                        .BLKB   8
```

```
                                                        .PSECT  $CODE$,NOWRT,2

                                0000C 00000             .ENTRY  CONV$$RMS_ERROR, Save R2,R3                     ; 0301
                     53 0000'   CF   9E 00002           MOVAB   STRING_DESC, R3
                     52 04      AC   D0 00007           MOVL    4(AST_BLOCK), RMS_BLOCK                         ; 0353
                     01         62   91 0000B           CMPB    (RMS_BLOCK), #1                                 ; 0357
                                06   12 0000E           BNEQ    1$
                     51 3C      A2   D0 00010           MOVL    60(RMS_BLOCK), FAB                              ; 0359
                                03   11 00014           BRB     2$
                     51         52   D0 00016 1$:       MOVL    RMS_BLOCK, FAB                                  ; 0361
                     50 28      A1   D0 00019 2$:       MOVL    40(FAB), NAM                                    ; 0365
                        03      A0   95 0001D           TSTB    3(NAM)                                          ; 0371
                                0B   13 00020           BEQL    3$
                     63 03      A0   9B 00022           MOVZBW  3(NAM), STRING_DESC                             ; 0374
                  04 A3 04      A0   D0 00026           MOVL    4(NAM), STRING_DESC+4                           ; 0375
                                19   11 0002B           BRB     5$
                        08      A0   95 0002D 3$:       TSTB    11(NAM)                                         ; 0380
                                0B   13 00030           BEQL    4$
                     63 0B      A0   9B 00032           MOVZBW  11(NAM), STRING_DESC                            ; 0383
                  04 A3 0C      A0   D0 00036           MOVL    12(NAM), STRING_DESC+4                          ; 0384
                                09   11 0003B           BRB     5$
                     63 34      A1   9B 0003D 4$:       MOVZBW  52(FAB), STRING_DESC                            ; 0391
                  04 A3 2C      A1   D0 00041           MOVL    44(FAB), STRING_DESC+4                          ; 0392
                     7E 08      A2   7D 00046 5$:       MOVQ    8(RMS_BLOCK), -(SP)                             ; 0400
                                53   DD 0004A           PUSHL   R3                                              ; 0399
                                01   DD 0004C           PUSHL   #1
                                18   A2 DD 0004E        PUSHL   24(RMS_BLOCK)
                00000000G       00   05 FB 00051        CALLS   #5, LIB$STOP
                                     04 00058           RET                                                     ; 0403

; Routine Size:  89 bytes,    Routine Base:  $CODE$ + 0000

;  173          0404  1
```

```
;  175          0405  1 %SBTTL  'CONV$$RMS_OPEN_ERROR'
;  176          0406  1 GLOBAL ROUTINE  CONV$$RMS_OPEN_ERROR : NOVALUE =
;  177          0407  1 !++
;  178          0408  1 !
;  179          0409  1 ! Functional Description:
;  180          0410  1 !
;  181          0411  1 !       This routine will signal an rms error and stop execution.  It is
;  182          0412  1 !       to be primarly used for detecting errors during file opens.
;  183          0413  1 !
;  184          0414  1 ! Calling Sequence:
;  185          0415  1 !
;  186          0416  1 !       This routine is call as an AST by RMS
;  187          0417  1 !
;  188          0418  1 ! Input Parameters:
;  189          0419  1 !
;  190          0420  1 !       AST argument block which has a pointer to a FAB
;  191          0421  1 !
;  192          0422  1 ! Implicit Inputs:
;  193          0423  1 !       none
;  194          0424  1 !
;  195          0425  1 ! Output Parameters:
;  196          0426  1 !       none
;  197          0427  1 !
;  198          0428  1 ! Implicit Outputs:
;  199          0429  1 !       none
;  200          0430  1 !
;  201          0431  1 ! Routine Value:
;  202          0432  1 !       none
;  203          0433  1 !
;  204          0434  1 ! Routines Called:
;  205          0435  1 !
;  206          0436  1 !       SIGNAL
;  207          0437  1 !       SIGNAL_STOP
;  208          0438  1 !
;  209          0439  1 ! Side Effects:
;  210          0440  1 !       none
;  211          0441  1 !
;  212          0442  1 !--
;  213          0443  1
;  214          0444  2     BEGIN
;  215          0445  2
;  216          0446  2     BUILTIN
;  217          0447  2         AP;
;  218          0448  2
;  219          0449  2     BIND
;  220          0450  2         AST_BLOCK = AP : REF VECTOR [ ,LONG ];
;  221          0451  2
;  222          0452  2     LOCAL
;  223          0453  2         RMS_BLOCK       : REF BLOCK [ ,BYTE ],
;  224          0454  2         FAB             : REF BLOCK [ ,BYTE ],
;  225          0455  2         NAM             : REF BLOCK [ ,BYTE ];
;  226          0456  2
;  227          0457  2     ! Get the rms block (Pointed to by the second ast parameter)
;  228          0458  2     !
;  229          0459  2     RMS_BLOCK = .AST_BLOCK [ 1 ];
;  230          0460  2
;  231          0461  2     ! If this is really a RAB (from a connect) then get the fab it points to
```

```
;  232          0462  2     !
;  233          0463  2     IF .RMS_BLOCK [ RAB$B_BID ] EQLU RAB$C_BID
;  234          0464  2     THEN
;  235          0465  2         FAB = .RMS_BLOCK [ RAB$L_FAB ]
;  236          0466  2     ELSE
;  237          0467  2         FAB = .RMS_BLOCK;
;  238          0468  2
;  239          0469  2     ! Get the name block
;  240          0470  2     !
;  241          0471  2     NAM = .FAB [ FAB$L_NAM ];
;  242          0472  2
;  243          0473  2     ! Signal the CONVert error with the best file name string
;  244          0474  2     !
;  245          0475  2     ! First try the resultant string
;  246          0476  2     !
;  247          0477  2     IF .NAM [ NAM$B_RSL ] NEQU 0
;  248          0478  2     THEN
;  249          0479  3         BEGIN
;  250          0480  3         STRING_DESC [ DSC$W_LENGTH ] = .NAM [ NAM$B_RSL ];
;  251          0481  3         STRING_DESC [ DSC$A_POINTER ] = .NAM [ NAM$[_RSA ]
;  252          0482  3         END
;  253          0483  3
;  254          0484  3     ! Next try the expanded string
;  255          0485  3     !
;  256          0486  2     ELSE IF .NAM [ NAM$B_ESL ] NEQU 0
;  257          0487  2     THEN
;  258          0488  3         BEGIN
;  259          0489  3         STRING_DESC [ DSC$W_LENGTH ] = .NAM [ NAM$B_ESL ];
;  260          0490  3         STRING_DESC [ DSC$A_POINTER ] = .NAM [ NAM$[_ESA ]
;  261          0491  3         END
;  262          0492  3
;  263          0493  3     ! If all else fails use the name string
;  264          0494  3     !
;  265          0495  2     ELSE
;  266          0496  3         BEGIN
;  267          0497  3         STRING_DESC [ DSC$W_LENGTH ] = .FAB [ FAB$B_FNS ];
;  268          0498  3         STRING_DESC [ DSC$A_POINTER ] = .FAB [ FAB$[_FNA ]
;  269          0499  2         END;
;  270          0500  2
;  271          0501  2     ! Signal the RMS error and stop. NOTE: During the open the fab will have
;  272          0502  2     ! correct convert error code.  The rms_block will have the rms error code
;  273          0503  2     !
;  274          0504  2     SIGNAL_STOP( .FAB [ FAB$L_CTX ],1,STRING_DESC,
;  275          0505  2                  .RMS_BLOCK [ FAB$L_STS ],
;  276          0506  2                  .RMS_BLOCK [ FAB$L_STV ] )
;  277          0507  2
;  278          0508  1     END;


                                 000C 00000          .ENTRY  CONV$$RMS_OPEN_ERROR, Save R2,R3                ; 0406
                    53     0000'  CF   9E 00002      MOVAB   STRING_DESC, R3                                 ;
                    52       04   AC   D0 00007      MOVL    4(AST_BLOCK), RMS_BLOCK                         ; 0459
                    01            62   91 0000B      CMPB    (RMS_BLOCK), #1                                 ; 0463
                                  06   12 0000E      BNEQ    1$                                              ;
```

```
                          51      3C    A2  D0 00010          MOVL    60(RMS_BLOCK), FAB                      ;  465
                                        03  11 00014          BRB     2$                                      ;
                          51            52  D0 00016 1$:     MOVL    RMS_BLOCK, FAB                          ; 0467
                          50      28    A1  D0 00019 2$:     MOVL    40(FAB), NAM                            ; 0471
                                  03    A0  95 0001D         TSTB    3(NAM)                                  ; 0477
                                        0B  13 00020         BEQL    3$                                      ;
                          63      03    A0  9B 00022         MOVZBW  3(NAM), STRING_DESC                     ; 0480
                    04    A3      04    A0  D0 00026         MOVL    4(NAM), STRING_DESC+4                   ; 0481
                                        19  11 0002B         BRB     5$                                      ;
                                  0B    A0  95 0002D 3$:     TSTB    11(NAM)                                 ; 0486
                                        0B  13 00030         BEQL    4$                                      ;
                          63      0B    A0  9B 00032         MOVZBW  11(NAM), STRING_DESC                    ; 0489
                    04    A3      0C    A0  D0 00036         MOVL    12(NAM), STRING_DESC+4                  ; 0490
                                        09  11 0003B         BRB     5$                                      ;
                          63      34    A1  9B 0003D 4$:     MOVZBW  52(FAB), STRING_DESC                    ; 0497
                    04    A3      2C    A1  D0 00041         MOVL    44(FAB), STRING_DESC+4                  ; 0498
                          7E      08    A2  7D 00046 5$:     MOVQ    8(RMS_BLOCK), -(SP)                     ; 0505
                                        53  DD 0004A         PUSHL   R3                                      ; 0504
                                        01  DD 0004C         PUSHL   #1                                      ;
                                  18    A1  DD 0004E         PUSHL   24(FAB)                                 ;
                00000000G   00          05  FB 00051         CALLS   #5, LIB$STOP                            ;
                                        04  00058         RET                                             ; 0508

; Routine Size:  89 bytes,    Routine Base:  $CODE$ + 0059

;  279          0509  1
```

```
;  281          0510  1 %SBTTL  'CONV$$RMS_READ_ERROR'
;  282          0511  1 GLOBAL ROUTINE  CONV$$RMS_READ_ERROR : NOVALUE =
;  283          0512  1 !++
;  284          0513  1 !
;  285          0514  1 ! Functional Description:
;  286          0515  1 !
;  287          0516  1 !       This routine will signal an rms error and stop execution if the RMS
;  288          0517  1 !       error is NOT end of file.  It is to be used for detecting errors
;  289          0518  1 !       during rms $GETs or $READs.
;  290          0519  1 !
;  291          0520  1 ! Calling Sequence:
;  292          0521  1 !
;  293          0522  1 !       This routine is call as an AST by RMS
;  294          0523  1 !
;  295          0524  1 ! Input Parameters:
;  296          0525  1 !
;  297          0526  1 !       AST argument block which has a pointer to a RAB
;  298          0527  1 !
;  299          0528  1 ! Implicit Inputs:
;  300          0529  1 !       none
;  301          0530  1 !
;  302          0531  1 ! Output Parameters:
;  303          0532  1 !       none
;  304          0533  1 !
;  305          0534  1 ! Implicit Outputs:
;  306          0535  1 !       none
;  307          0536  1 !
;  308          0537  1 ! Routine Value:
;  309          0538  1 !       none
;  310          0539  1 !
;  311          0540  1 ! Routines Called:
;  312          0541  1 !
;  313          0542  1 !       SIGNAL
;  314          0543  1 !       SIGNAL_STOP
;  315          0544  1 !
;  316          0545  1 ! Side Effects:
;  317          0546  1 !       none
;  318          0547  1 !
;  319          0548  1 !--
;  320          0549  1
;  321          0550  2     BEGIN
;  322          0551  2
;  323          0552  2     BUILTIN
;  324          0553  2         AP;
;  325          0554  2
;  326          0555  2     BIND
;  327          0556  2         AST_BLOCK = AP : REF VECTOR [ ,LONG ];
;  328          0557  2
;  329          0558  2     LOCAL
;  330          0559  2         RAB : REF BLOCK [ ,BYTE ],
;  331          0560  2         FAB : REF BLOCK [ ,BYTE ],
;  332          0561  2         NAM : REF BLOCK [ ,BYTE ];
;  333          0562  2
;  334          0563  2     ! Get the rab (Pointer to by the second ast parameter)
;  335          0564  2     !
;  336          0565  2     RAB = .AST_BLOCK [ 1 ];
;  337          0566  2
```

```
;   338          0567  2      ! If this is only an end of file then return
;   339          0568  2      !
;   340          0569  2      IF .RAB [ RAB$L_STS ] EQLU RMS$_EOF
;   341          0570  2      THEN
;   342          0571  2          RETURN;
;   343          0572  2
;   344          0573  2      ! Now get the fab it points to
;   345          0574  2      !
;   346          0575  2      FAB = .RAB [ RAB$L_FAB ];
;   347          0576  2
;   348          0577  2      ! Get the name block
;   349          0578  2      !
;   350          0579  2      NAM = .FAB [ FAB$L_NAM ];
;   351          0580  2
;   352          0581  2      ! Signal the CONVert error with the best file name string
;   353          0582  2      !
;   354          0583  2      ! First try the resultant string
;   355          0584  2      !
;   356          0585  2      IF .NAM [ NAM$B_RSL ] NEQU 0
;   357          0586  2      THEN
;   358          0587  3          BEGIN
;   359          0588  3          STRING_DESC [ DSC$W_LENGTH ] = .NAM [ NAM$B_RSL ];
;   360          0589  3          STRING_DESC [ DSC$A_POINTER ] = .NAM [ NAM$[_RSA ]
;   361          0590  3          END
;   362          0591  3
;   363          0592  3      ! Next try the expanded string
;   364          0593  3      !
;   365          0594  2      ELSE IF .NAM [ NAM$B_ESL ] NEQU 0
;   366          0595  2      THEN
;   367          0596  3          BEGIN
;   368          0597  3          STRING_DESC [ DSC$W_LENGTH ] = .NAM [ NAM$B_ESL ];
;   369          0598  3          STRING_DESC [ DSC$A_POINTER ] = .NAM [ NAM$[_ESA ]
;   370          0599  3          END
;   371          0600  3
;   372          0601  3      ! If all else fails use the name string
;   373          0602  3      !
;   374          0603  2      ELSE
;   375          0604  3          BEGIN
;   376          0605  3          STRING_DESC [ DSC$W_LENGTH ] = .FAB [ FAB$B_FNS ];
;   377          0606  3          STRING_DESC [ DSC$A_POINTER ] = .FAB [ FAB$[_FNA ]
;   378          0607  2          END;
;   379          0608  2
;   380          0609  2      ! Signal the RMS error and stop
;   381          0610  2      !
;   382          0611  2      SIGNAL_STOP( .RAB [ RAB$L_CTX ],1,STRING_DESC,
;   383          0612  2                   .RAB [ FAB$L_STS ],
;   384          0613  2                   .RAB [ FAB$L_STV ] )
;   385          0614  2
;   386          0615  1      END;


                                 000C 00000          .ENTRY  CONV$$RMS_READ_ERROR, Save R2,R3          ; 0511
                     53      0000'  CF  9E 00002      MOVAB   STRING_DESC, R3                           ;
                     52        04   AC  D0 00007      MOVL    4(AST_BLOCK), RAB                         ; 0565
```

```
                 0001827A  8F      08  A2  D1 0000B          CMPL    8(RAB), #98938                  ; 0569
                                       43  13 00013          BEQL    4$                              ;
                           51      3C  A2  D0 00015          MOVL    60(RAB), FAB                    ; 0575
                           50      28  A1  D0 00019          MOVL    40(FAB), NAM                    ; 0579
                                   03  A0  95 0001D          TSTB    3(NAM)                          ; 0585
                                       0B  13 00020          BEQL    1$                              ;
                           63      03  A0  9B 00022          MOVZBW  3(NAM), STRING_DESC             ; 0588
                        04 A3      04  A0  D0 00026          MOVL    4(NAM), STRING_DESC+4           ; 0589
                                       19  11 0002B          BRB     3$                              ;
                                   0B  A0  95 0002D 1$:      TSTB    11(NAM)                         ; 0594
                                       0B  13 00030          BEQL    2$                              ;
                           63      0B  A0  9B 00032          MOVZBW  11(NAM), STRING_DESC            ; 0597
                        04 A3      0C  A0  D0 00036          MOVL    12(NAM), STRING_DESC+4          ; 0598
                                       09  11 00038          BRB     3$                              ;
                           63      34  A1  9B 0003D 2$:      MOVZBW  52(FAB), STRING_DESC            ; 0605
                        04 A3      2C  A1  D0 00041          MOVL    44(FAB), STRING_DESC+4          ; 0606
                           7E      08  A2  7D 00046 3$:      MOVQ    8(RAB), -(SP)                   ; 0612
                                       53  DD 0004A          PUSHL   R3                              ; 0611
                                       01  DD 0004C          PUSHL   #1                              ;
                                   18  A2  DD 0004E          PUSHL   24(RAB)                         ;
                 00000000G  00             05  FB 00051      CALLS   #5, LIB$STOP                    ;
                                       04  00058 4$:         RET                                     ; 0615

; Routine Size:  89 bytes,    Routine Base:  $CODE$ + 00B2

;  387          0616  1
;  388          0617  0 END     ELUDOM

                                                            .EXTRN  LIB$STOP

;                                PSECT SUMMARY
;
;      Name                      Bytes                           Attributes
;
;  $OWN$                            8  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;  $CODE$                         267  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)


;                              Library Statistics
;
;                                      -------- Symbols --------      Pages       Processing
;      File                            Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1        18619        19         0        1000        00:01.8
```

```
;                                       COMMAND QUALIFIERS

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

; Size:         267 code + 8 data bytes
; Run Time:       00:09.8
; Elapsed Time:   00:44.8
; Lines/CPU Min:   3777
; Lexemes/CPU-Min: 20877
; Memory Used:  79 pages
; Compilation Complete
```

CONVDATA
LIS

CONVFSTLD
LIS

CONVFSTIO
LIS

CONVFSTRC
LIS

CONVFILES
LIS

CONVERROR
LIS

CONVFASTM
LIS

CONVDCL
LIS