```
BBBBBBBBBBBB        AAAAAAAAA        SSSSSSSSSSS  RRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
BBBBBBBBBBBB        AAAAAAAAA        SSSSSSSSSSS  RRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
BBBBBBBBBBBB        AAAAAAAAA        SSSSSSSSSSS  RRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
BBB         BBB   AAA         AAA   SSS          RRR         RRR         TTT        LLL
BBB         BBB   AAA         AAA   SSS          RRR         RRR         TTT        LLL
BBB         BBB   AAA         AAA   SSS          RRR         RRR         TTT        LLL
BBB         BBB   AAA         AAA   SSS          RRR         RRR         TTT        LLL
BBB         BBB   AAA         AAA   SSS          RRR         RRR         TTT        LLL
BBB         BBB   AAA         AAA   SSS          RRR         RRR         TTT        LLL
BBBBBBBBBBBB      AAA         AAA     SSSSSSSSS  RRRRRRRRRRR             TTT        LLL
BBBBBBBBBBBB      AAA         AAA     SSSSSSSSS  RRRRRRRRRRR             TTT        LLL
BBBBBBBBBBBB      AAA         AAA     SSSSSSSSS  RRRRRRRRRRR             TTT        LLL
BBB         BBB   AAAAAAAAAAAAAAA           SSS  RRR   RRR               TTT        LLL
BBB         BBB   AAAAAAAAAAAAAAA           SSS  RRR   RRR               TTT        LLL
BBB         BBB   AAAAAAAAAAAAAAA           SSS  RRR   RRR               TTT        LLL
BBB         BBB   AAA         AAA           SSS  RRR      RRR            TTT        LLL
BBB         BBB   AAA         AAA           SSS  RRR   RRR               TTT        LLL
BBB         BBB   AAA         AAA           SSS  RRR      RRR            TTT        LLL
BBBBBBBBBBBB      AAA         AAA   SSSSSSSSSSS     RRR         RRR      TTT        LLLLLLLLLLLLLLL
BBBBBBBBBBBB      AAA         AAA   SSSSSSSSSSS     RRR         RRR      TTT        LLLLLLLLLLLLLLL
BBBBBBBBBBBB      AAA         AAA   SSSSSSSSSSS     RRR         RRR      TTT        LLLLLLLLLLLLLLL
```

```
BBBBBBBB      AAAAAA      SSSSSSSS  VV      VV    IIIIII    RRRRRRRR    TTTTTTTTTT    IIIIII      000000
BBBBBBBB      AAAAAA      SSSSSSSS  VV      VV    IIIIII    RRRRRRRR    TTTTTTTTTT    IIIIII      000000
BB      BB  AA      AA  SS        VV      VV      II      RR      RR      TT        II      00      00
BB      BB  AA      AA  SS        VV      VV      II      RR      RR      TT        II      00      00
BB      BB  AA      AA  SS        VV      VV      II      RR      RR      TT        II      00      00
BB      BB  AA      AA  SS        VV      VV      II      RR      RR      TT        II      00      00
BBBBBBBB    AA      AA    SSSSSS    VV      VV      II      RRRRRRRR        TT        II      00      00
BBBBBBBB    AA      AA    SSSSSS    VV      VV      II      RRRRRRRR        TT        II      00      00
BB      BB  AAAAAAAAAA          SS  VV      VV      II      RR  RR          TT        II      00      00
BB      BB  AAAAAAAAAA          SS  VV      VV      II      RR  RR          TT        II      00      00
BB      BB  AA      AA          SS    VV  VV        II      RR    RR        TT        II      00      00      ....
BB      BB  AA      AA          SS    VV  VV        II      RR    RR        TT        II      00      00      ....
BBBBBBBB    AA      AA  SSSSSSSS        VV        IIIIII    RR      RR      TT      IIIIII      000000      ....
BBBBBBBB    AA      AA  SSSSSSSS        VV        IIIIII    RR      RR      TT      IIIIII      000000      ....

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 BAS$$VIRT_IO (                          ! Virtual array I/O
;     2          0002  0                 IDENT = '1-027'               ! File: BASVIRTIO.B32 Edit: MDL1027
;     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:  VAX-11 BASIC Virtual Array Support
;    32          0032  1 !
;    33          0033  1 ! ABSTRACT:
;    34          0034  1 !
;    35          0035  1 !       This module contains the I/O support for VAX-11 BASIC
;    36          0036  1 !       virtual arrays.  In the context of the RTL these are called
;    37          0037  1 !       "BASIC File Arrays", since they are not properly a part of
;    38          0038  1 !       the VAX architecture.  This module comprises the UDF and REC
;    39          0039  1 !       levels of I/O for this very simple I/O interface.
;    40          0040  1 !
;    41          0041  1 ! ENVIRONMENT:  VAX-11 User Mode
;    42          0042  1 !
;    43          0043  1 ! AUTHOR: John Sauter, CREATION DATE: 04-APR-1979
;    44          0044  1 !
;    45          0045  1 ! MODIFIED BY:
;    46          0046  1 !
;    47          0047  1 ! 1-001 - Original.  This version does no buffering.  It is just for
;    48          0048  1 !         checking out the indexing routines.  JBS 04-APR-1979
;    49          0049  1 ! 1-002 - Change BAS$$STOP to BAS$$STOP_IO wherever possible.
;    50          0050  1 !         JBS 09-APR-1979
;    51          0051  1 ! 1-003 - Improve comments based on DGP's review.  JBS 09-APR-1979
;    52          0052  1 ! 1-004 - Recover from Record Stream Active RMS error.  JBS 09-APR-1979
;    53          0053  1 !         JBS 09-APR-1979
;    54          0054  1 ! 1-005 - Today (actually late last night) the compiler began producing
;    55          0055  1 !         code for virtual arrays, so start debugging.  JBS 24-MAY-1979
;    56          0056  1 ! 1-006 - Take the ALQ parameter out of the $FAB_INIT, so that FAB$L_ALQ
;    57          0057  1 !         appears in the cross reference.  JBS 24-MAY-1979
```

```
;    58          0058  1 ! 1-007 - Don't allocate if the file is already allocated beyond the
;    59          0059  1 !           current record.  JBS 25-MAY-1979
;    60          0060  1 ! 1-008 - Worry about two descriptors pointing to the same file.
;    61          0061  1 !           JBS 11-JUN-1979
;    62          0062  1 ! 1-009 - Remove the redundent DSC$ defintiions.  JBS 19-JUN-1979
;    63          0063  1 ! 1-010 - POP I/O on error.  JBS 25-JUL-1979
;    64          0064  1 ! 1-011 - The buffer size for a virtual array file must be 512 bytes.
;    65          0065  1 !           JBS 20-AUG-1979
;    66          0066  1 ! 1-012 - Correct a typo in BAS$VA_PURGE.  JBS 30-AUG-1979
;    67          0067  1 ! 1-013 - Check for RMS$_RNF in parallel with RMS$_EOF.  JBS 17-SEP-1979
;    68          0068  1 ! 1-014 - Disable EXTEND until it is fixed.  JBS 17-SEP-1979
;    69          0069  1 ! 1-015 - When unwinding, mark that we have no buffer in memory, since
;    70          0070  1 !           we want to retry all I/O operations.  JBS 17-SEP-1979
;    71          0071  1 ! 1-016 - Signal errors if the RELEASE fails.  This will have to be
;    72          0072  1 !           disabled to run under release 1.  JBS 17-SEP-1979
;    73          0073  1 ! 1-017 - The VAH was bad design, because we cannot purge the virtual
;    74          0074  1 !           arrays whenever we lose control (consider a divide by zero
;    75          0075  1 !           under an ON ERROR GO BACK).  Therefore, remove VAH and do not
;    76          0076  1 !           use the HANDLE field.  Also, put the code back to using release
;    77          0077  1 !           1 RMS.  JBS 18-SEP-1979
;    78          0078  1 ! 1-018 - Don't allow stores into virtual arrays opened read only.
;    79          0079  1 !           JBS 07-NOV-1979
;    80          0080  1 ! 1-019 - Convert to automatic record locking and NXR processing.
;    81          0081  1 !           JBS 09-NOV-1979
;    82          0082  1 ! 1-020 - Remove BAS$VA_PURGE, which has been a no-op since September 18,
;    83          0083  1 !           since the compiler no longer refers to it.  JBS 26-NOV-1979
;    84          0084  1 ! 1-021 - Don't call BAS$$CB_POP if the I/O data base has already been
;    85          0085  1 !           cleaned up.  JBS 11-JUN-1980
;    86          0086  1 ! 1-022 - Add new entry points to fetch/store entire virtual arrays
;    87          0087  1 !           instead of individual array elements.  PLL 2-Apr-1982
;    88          0088  1 ! 1-023 - Add support for decimal.  PLL 12-Apr-1982
;    89          0089  1 ! 1-024 - Bug fix to entire array entry points - check to see if current
;    90          0090  1 !           buffer needs to be written out before getting/putting a new
;    91          0091  1 !           record.  PLL 3-May-1982
;    92          0092  1 ! 1-025 - Bug fix to entire array entry points again.  If the array size
;    93          0093  1 !           and the offset add up to less than 512, use the array size for
;    94          0094  1 !           the initial number of bytes to copy.  PLL 10-May-1982
;    95          0095  1 ! 1-026 - Make sure Record Access will always be by key in BAS$$VA_CLOSE.
;    96          0096  1 !           DG 27-Mar-1984
;    97          0097  1 ! 1-027 - check for RMS$_CONTROLC return status.  MDL 30-Mar-1984
;    98          0098  1 !--
;    99          0099  1
;   100          0100  1 !<BLF/PAGE>
```

```
;  102          010  1 !
;  103          010  1 ! SWITCHES:
;  104          010  1 !
;  105          010  1
;  106          010  1 SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
;  107          010  1
;  108          010  1 !
;  109          010  1 ! LINKAGES:
;  110          010  1 !
;  111          011  1
;  112          011  1 REQUIRE 'RTLIN:OTSLNK';                          ! Define linkages
;  113          054  1
;  114          054  1 !
;  115          054  1 ! TABLE OF CONTENTS:
;  116          054  1 !
;  117          054  1
;  118          054  1 FORWARD ROUTINE
;  119          054  1     BAS$$VA_FETCH : NOVALUE,                     ! Fetch routine
;  120          054  1     BAS$$VA_STORE : NOVALUE,                     ! Store routine
;  121          054  1     BAS$$VA_CLOSE : CALL_CCB NOVALUE,            ! CLOSE effector
;  122          054  1     BAS$$WHOLE_VA_FETCH : NOVALUE,               ! Fetch whole array
;  123          055  1     BAS$$WHOLE_VA_STORE : NOVALUE,               ! Store whole array
;  124          055  1     HANDLER;                                     ! POP CCB on UNWIND
;  125          055  1
;  126          055  1 !
;  127          055  1 ! INCLUDE FILES:
;  128          055  1 !
;  129          055  1
;  130          055  1 REQUIRE 'RTLML:OTSLUB';                         ! Get LUB definitions
;  131          069  1
;  132          069  1 REQUIRE 'RTLIN:BASIOERR';                       ! I/O error codes
;  133          075  1
;  134          075  1 REQUIRE 'RTLIN:RTLPSECT';                       ! Macros for defining psects
;  135          084  1
;  136          084  1 LIBRARY 'RTLSTARLE';                            ! System symbols
;  137          084  1
;  138          085  1 !
;  139          085  1 ! MACROS:
;  140          085  1 !
;  141          085  1 !       NONE
;  142          085  1 !
;  143          085  1 ! EQUATED SYMBOLS:
;  144          085  1 !
;  145          085  1 !+
;  146          085  1 ! The following literal determines the span of the interlock on shared
;  147          085  1 ! files.  That is, the number of bytes which are interlocked after a
;  148          086  1 ! reference to a location in a virtual array.  This is also the buffer
;  149          086  1 ! size required on the OPEN for the file.
;  150          086  1 !-
;  151          086  1
;  152          086  1 LITERAL
;  153          086  1     K_BLOCK_LENGTH = 512;                        ! Number of bytes in a virtual block
;  154          086  1
;  155          086  1 !
;  156          086  1 ! PSECTS:
;  157          086  1 !
;  158          087  1 DECLARE_PSECTS (BAS);                           ! Declare psects for BAS$ facility
```

```
;  159          0871  1 !
;  160          0872  1 ! OWN STORAGE:
;  161          0873  1 !
;  162          0874  1 !       NONE
;  163          0875  1 !
;  164          0876  1 ! EXTERNAL REFERENCES:
;  165          0877  1 !
;  166          0878  1
;  167          0879  1 EXTERNAL ROUTINE
;  168          0880  1     BAS$$STOP : NOVALUE,                    ! signals fatal error
;  169          0881  1     BAS$$CB_PUSH : JSB_CB_PUSH NOVALUE,     ! Load register CCB
;  170          0882  1     BAS$$CB_POP : JSB_CB_POP NOVALUE,       ! Done with register CCB
;  171          0883  1     BAS$$CB_GET : JSB_CB_GET NOVALUE,       ! Fetch current value of CCB
;  172          0884  1     BAS$$STOP_IO : NOVALUE,                 ! Signal fatal I/O error
;  173          0885  1     BAS$$SIGNAL_CTRLC : NOVALUE,            ! Signal CTRL/C
;  174          0886  1     LIB$STOP : NOVALUE,                     ! Signal fatal error
;  175          0887  1     LIB$MATCH_COND;                         ! Match condition values
;  176          0888  1
;  177          0889  1 !+
;  178          0890  1 ! The following are the error codes used in this module.
;  179          0891  1 !-
;  180          0892  1
;  181          0893  1 EXTERNAL LITERAL
;  182          0894  1     BAS$K_VIRARROPE : UNSIGNED (8),         ! Virtual array not opened
;  183          0895  1     BAS$K_VIRARRDIS : UNSIGNED (8),         ! Virtual array not on disk
;  184          0896  1     BAS$K_VIRBUFTOO : UNSIGNED (8),         ! Virtual buffer too large
;  185          0897  1     BAS$K_ILLOPE : UNSIGNED (8),            ! Illegal operation
;  186          0898  1     BAS$K_ILLILLACC : UNSIGNED (8),         ! Illegal or illogical access
;  187          0899  1     BAS$K_PROLOSSOR : UNSIGNED (8),         ! Program lost, sorry
;  188          0900  1     OTS$_FATINTERR;                         ! Fatal internal OTS error
;  189          0901  1
```

```
;  191          0902  1 GLOBAL ROUTINE BAS$$VA_FETCH (                        ! Fetch routine
;  192          0903  1         DESCRIP,                                      ! The descriptor for this virtual array
;  193          0904  1         INDEX,                                        ! Linearized index
;  194          0905  1         VALUE                                         ! Where to store array item
;  195          0906  1     ) : NOVALUE =
;  196          0907  1
;  197          0908  1 !++
;  198          0909  1 ! FUNCTIONAL DESCRIPTION:
;  199          0910  1 !
;  200          0911  1 !       Fetch a value from a virtual array.  Multiple bytes may be
;  201          0912  1 !       fetched with a single call.
;  202          0913  1 !
;  203          0914  1 ! FORMAL PARAMETERS:
;  204          0915  1 !
;  205          0916  1 !       DESCRIP.mz.r    The descriptor for the virtual array
;  206          0917  1 !       INDEX.rl.v      The byte offset into the array
;  207          0918  1 !       VALUE.wz.r      The place to store the value.  The number of
;  208          0919  1 !                       bytes to store is in the LENGTH field of
;  209          0920  1 !                       DESCRIP.
;  210          0921  1 !
;  211          0922  1 ! IMPLICIT INPUTS:
;  212          0923  1 !
;  213          0924  1 !       NONE
;  214          0925  1 !
;  215          0926  1 ! IMPLICIT OUTPUTS:
;  216          0927  1 !
;  217          0928  1 !       NONE
;  218          0929  1 !
;  219          0930  1 ! ROUTINE VALUE:
;  220          0931  1 ! COMPLETION CODES:
;  221          0932  1 !
;  222          0933  1 !       NONE
;  223          0934  1 !
;  224          0935  1 ! SIDE EFFECTS:
;  225          0936  1 !
;  226          0937  1 !       Signals if an error is encountered.
;  227          0938  1 !
;  228          0939  1 !--
;  229          0940  1
;  230          0941  2     BEGIN
;  231          0942  2
;  232          0943  2     MAP
;  233          0944  2         DESCRIP : REF BLOCK [8, BYTE];
;  234          0945  2
;  235          0946  2     GLOBAL REGISTER
;  236          0947  2         CCB = K_CCB_REG : REF BLOCK [, BYTE];
;  237          0948  2
;  238          0949  2     BUILTIN
;  239          0950  2         ASHP;
;  240          0951  2
;  241          0952  2     LOCAL
;  242          0953  2         CHAN,                                         ! The channel this array is defined on
;  243          0954  2         HANDLE,                                       ! Pointer to info for this array
;  244          0955  2         GET_STATUS,                                   ! Last RMS GET status
;  245          0956  2         PUT_STATUS,                                   ! Last RMS PUT status
;  246          0957  2         READ_RECORD,                                  ! 1 = we must read the record
;  247          0958  2         SAVE_CCB : VOLATILE;                          ! CCB to POP, or zero.
```

```
;  248          0959  2
;  249          0960  2 !+
;  250          0961  2 ! Establish a handler to pop the CCB when unwinding.
;  251          0962  2 !-
;  252          0963  2
;  253          0964  2     ENABLE
;  254          0965  2         HANDLER (SAVE_CCB);
;  255          0966  2
;  256          0967  2 !+
;  257          0968  2 ! Fetch the array's channel number from its descriptor
;  258          0969  2 !-
;  259          0970  2     CHAN = .DESCRIP [DSC$L_LOGUNIT];
;  260          0971  2 !+
;  261          0972  2 ! Get a pointer to the LUB/ISB/RAB for this channel.  If the channel has not
;  262          0973  2 ! been opened yet, this call will allocate the LUB/ISB/RAB, but we will reject
;  263          0974  2 ! it for lack of the LUB$V_OPENED bit.
;  264          0975  2 !-
;  265          0976  2     BAS$$CB_PUSH (.CHAN, LUB$K_LUN_MIN);
;  266          0977  2     SAVE_CCB = .CCB;
;  267          0978  2
;  268          0979  2     IF ( NOT .CCB [LUB$V_OPENED]) THEN BAS$$STOP (BAS$K_VIRARROPE);
;  269          0980  2
;  270          0981  2 !+
;  271          0982  2 ! If the channel was not opened with organization VIRTUAL, reject it.  This
;  272          0983  2 ! also catches channel 0, which is always open but never has VIRTUAL
;  273          0984  2 ! organization.
;  274          0985  2 !-
;  275          0986  2
;  276          0987  2     IF (.CCB [LUB$B_ORGAN] NEQ LUB$K_ORG_VIRTU) THEN BAS$$STOP_IO (BAS$K_VIRARRDIS);
;  277          0988  2
;  278          0989  2 !+
;  279          0990  2 ! If this channel has been used for block I/O, reject it.
;  280          0991  2 !-
;  281          0992  2
;  282          0993  2     IF (.CCB [LUB$V_BLK_USE]) THEN BAS$$STOP_IO (BAS$K_ILLOPE);
;  283          0994  2
;  284          0995  2 !+
;  285          0996  2 ! If the record size declared for the file is not 512 bytes, reject it.
;  286          0997  2 !-
;  287          0998  2
;  288          0999  2     IF (.CCB [RAB$W_USZ] NEQ K_BLOCK_LENGTH) THEN BAS$$STOP_IO (BAS$K_VIRBUFTOO);
;  289          1000  2
;  290          1001  2 !+
;  291          1002  2 ! Mark the LUB as being used for a virtual array.
;  292          1003  2 !-
;  293          1004  2     CCB [LUB$V_VA_USE] = 1;
;  294          1005  2 !+
;  295          1006  2 ! Record access will always be by key
;  296          1007  2 !-
;  297          1008  2     CCB [RAB$B_RAC] = RAB$C_KEY;
;  298          1009  2 !+
;  299          1010  2 ! Mark the RAB so that a $GET to a non-existent record will still lock it.
;  300          1011  2 !-
;  301          1012  2     CCB [RAB$V_NXR] = 1;
;  302          1013  2 !+
;  303          1014  2 ! Set the address of our CLOSE appendage in the LUB.  If somebody else's
;  304          1015  2 ! is already there, we have a serious problem.
```

```
;  305          1016  2 !-
;  306          1017  2
;  307          1018  2     IF (.CCB [LUB$A_CLOSE] EQLA 0) THEN CCB [LUB$A_CLOSE] = BAS$$VA_CLOSE;
;  308          1019  2
;  309          1020  2     IF (.CCB [LUB$A_CLOSE] NEQA BAS$$VA_CLOSE) THEN BAS$$STOP_IO (BAS$K_PROLOSSOR);
;  310          1021  2
;  311          1022  2 !+
;  312          1023  2 ! If this is not the first reference to this file, we may have to
;  313          1024  2 ! write out the current buffer.  We will write only if the current buffer
;  314          1025  2 ! is not the buffer we wish to access.  LUB$L_LOG_RECNO is initialized
;  315          1026  2 ! to zero for virtual files.
;  316          1027  2 !-
;  317          1028  2
;  318          1029  3     IF (.CCB [LUB$L_LOG_RECNO] EQL ((.INDEX + .DESCRIP [DSC$L_BYTEOFF])/K_BLOCK_LENGTH) + 1)
;  319          1030  2     THEN
;  320          1031  2         READ_RECORD = 0
;  321          1032  2     ELSE
;  322          1033  3         BEGIN
;  323          1034  3 !+
;  324          1035  3 ! We actually do the PUT only if the buffer has been changed since we last
;  325          1036  3 ! read it, as recorded by LUB$V_OUTBUF_DR.
;  326          1037  3 !-
;  327          1038  3
;  328          1039  4         IF (.CCB [LUB$V_OUTBUF_DR])
;  329          1040  3         THEN
;  330          1041  4             BEGIN
;  331          1042  4             PUT_STATUS = $PUT (RAB = .CCB);
;  332          1043  4
;  333          1044  4         IF .PUT_STATUS EQL RMS$_CONTROLC
;  334          1045  4         THEN
;  335          1046  4             BAS$$SIGNAL_CTRLC ();
;  336          1047  4
;  337          1048  4 !+
;  338          1049  4 ! If the PUT fails, we must worry about the RSA error, which can happen if
;  339          1050  4 ! we are running at AST level, and the AST interrupted some RMS I/O.  If
;  340          1051  4 ! we get this error, wait for it to go away.  Any other RMS error is fatal.
;  341          1052  4 !-
;  342          1053  4
;  343          1054  5             IF ( NOT .PUT_STATUS)
;  344          1055  4             THEN
;  345          1056  5                 BEGIN
;  346          1057  5
;  347          1058  5                 WHILE (.PUT_STATUS EQL RMS$_RSA) DO
;  348          1059  6                     BEGIN
;  349          1060  6                     $WAIT (RAB = .CCB);
;  350          1061  6                     PUT_STATUS = $PUT (RAB = .CCB);
;  351          1062  6
;  352          1063  6                     IF .PUT_STATUS EQL RMS$_CONTROLC
;  353          1064  6                     THEN
;  354          1065  6                         BAS$$SIGNAL_CTRLC ();
;  355          1066  6
;  356          1067  5                     END;
;  357          1068  5
;  358          1069  5                 IF ( NOT .PUT_STATUS) THEN BAS$$STOP_IO (BAS$K_IOERR_REC);
;  359          1070  5
;  360          1071  4                 END;
;  361          1072  4
```

```
;  362          1073  4 !+
;  363          1074  4 ! The buffer is no longer "dirty", mark it so.
;  364          1075  4 !-
;  365          1076  4             CCB [LUB$V_OUTBUF_DR] = 0;
;  366          1077  3             END;
;  367          1078  3
;  368          1079  3         READ_RECORD = 1;
;  369          1080  2         END;
;  370          1081  2
;  371          1082  2 !+
;  372          1083  2 ! If necessary, read in the record containing the element we want.
;  373          1084  2 !-
;  374          1085  2
;  375          1086  3     IF (.READ_RECORD)
;  376          1087  2     THEN
;  377          1088  3         BEGIN
;  378          1089  3         CCB [LUB$L_LOG_RECNO] = ((.INDEX + .DESCRIP [DSC$L_BYTEOFF])/K_BLOCK_LENGTH) + 1;
;  379          1090  3         GET_STATUS = $GET (RAB = .CCB);
;  380          1091  3
;  381          1092  3         IF .GET_STATUS EQL RMS$_CONTROLC
;  382          1093  3         THEN
;  383          1094  3             BAS$$SIGNAL_CTRLC ();
;  384          1095  3
;  385          1096  3 !+
;  386          1097  3 ! If we get EOF, just clear the buffer.  This is compatable with
;  387          1098  3 ! the PDP-11.
;  388          1099  3 !-
;  389          1100  3
;  390          1101  4         IF ((.GET_STATUS EQL RMS$_EOF) OR (.GET_STATUS EQL RMS$_OK_RNF))
;  391          1102  3         THEN
;  392          1103  4             BEGIN
;  393          1104  4             CH$FILL (0, .CCB [RAB$W_USZ], .CCB [RAB$L_UBF]);
;  394          1105  4             CCB [RAB$L_RBF] = .CCB [RAB$L_UBF];
;  395          1106  4             CCB [RAB$W_RSZ] = .CCB [RAB$W_USZ];
;  396          1107  4             END
;  397          1108  3         ELSE
;  398          1109  4             BEGIN
;  399          1110  4
;  400          1111  5             IF ( NOT .GET_STATUS)
;  401          1112  4             THEN
;  402          1113  5                 BEGIN
;  403          1114  5 !+
;  404          1115  5 ! Again, worry about the RSA RMS error.
;  405          1116  5 !-
;  406          1117  5
;  407          1118  5                 WHILE (.GET_STATUS EQL RMS$_RSA) DO
;  408          1119  6                     BEGIN
;  409          1120  6                     $WAIT (RAB = .CCB);
;  410          1121  6                     GET_STATUS = $GET (RAB = .CCB);
;  411          1122  6
;  412          1123  6                     IF .GET_STATUS EQL RMS$_CONTROLC
;  413          1124  6                     THEN
;  414          1125  6                         BAS$$SIGNAL_CTRLC ();
;  415          1126  6
;  416          1127  5                     END;
;  417          1128  5
;  418          1129  5                 IF ( NOT .GET_STATUS) THEN BAS$$STOP_IO (BAS$K_IOERR_REC);
```

```
;  419          1130  5
;  420          1131  4                    END;
;  421          1132  4
;  422          1133  3                END;
;  423          1134  3
;  424          1135  2            END;
;  425          1136  2
;  426          1137  2        !+
;  427          1138  2        ! At this point, the proper record is in the buffer, and we can copy
;  428          1139  2        ! data from it.
;  429          1140  2        !-
;  430          1141  2            IF .DESCRIP [DSC$B_DTYPE] NEQ DSC$K_DTYPE_P
;  431          1142  2            THEN
;  432          1143  2                CH$MOVE (.DESCRIP [DSC$W_LENGTH],
;  433          1144  2                    .CCB [RAB$L_RBF] + (.INDEX + .DESCRIP [DSC$L_BYTEOFF]) MOD K_BLOCK_LENGTH), .VALUE)
;  434          1145  2            ELSE
;  435          1146  3                BEGIN
;  436          1147  3                MAP
;  437          1148  3                    VALUE : REF BLOCK [12,BYTE],
;  438          1149  3                    DESCRIP : REF BLOCK [12,BYTE];
;  439          1150  3                LOCAL
;  440          1151  3                    COUNT;
;  441          1152  3                COUNT = .DESCRIP [DSC$B_SCALE] - .VALUE [DSC$B_SCALE];
;  442          1153  3                ASHP (COUNT, DESCRIP [DSC$W_LENGTH],
;  443          1154  3                    .CCB [RAB$L_RBF] + ((.INDEX + .DESCRIP [DSC$L_BYTEOFF]) MOD K_BLOCK_LENGTH),
;  444          1155  3                    %REF(0), VALUE [DSC$W_LENGTH], .VALUE [DSC$A_POINTER]);
;  445          1156  2                END;
;  446          1157  2
;  447          1158  2        !+
;  448          1159  2        ! Done with this I/O channel.
;  449          1160  2        !-
;  450          1161  2            BAS$$CB_POP ();
;  451          1162  1            END;                                    ! end of BAS$$VA_FETCH


                                    .TITLE  BAS$$VIRT_IO
                                    .IDENT  \1-027\

                                    .EXTRN  BAS$$STOP, BAS$$CB_PUSH
                                    .EXTRN  BAS$$CB_POP, BAS$$CB_GET
                                    .EXTRN  BAS$$STOP_IO, BAS$$SIGNAL_CTRLC
                                    .EXTRN  LIB$STOP, LIB$MATCH_COND
                                    .EXTRN  BAS$K_VIRARROPE
                                    .EXTRN  BAS$K_VIRARRDIS
                                    .EXTRN  BAS$K_VIRBUFTOO
                                    .EXTRN  BAS$K_ILLOPE, BAS$K_ILLILLACC
                                    .EXTRN  BAS$K_PROLOSSOR
                                    .EXTRN  OTS$_FATINTERR, SYS$PUT
                                    .EXTRN  SYS$WAIT, SYS$GET

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

                        0FFC 00000  .ENTRY  BAS$$VA_FETCH, Save R2,R3,R4,R5,R6,R7,R8,-  ; 0902
                                            R9,R10,R11
5A 00000000G   00   9E 00002        MOVAB   BAS$$SIGNAL_CTRLC, R10                      ;
59 00000000G   00   9E 00009        MOVAB   BAS$$STOP_IO, R9                            ;
                    7E  D4 00010    CLRL    SAVE_CCB                                    ; 0941
```

```
                        6D     01B4   CF   DE 00012          MOVAL   22$, (FP)                                       ;
                        57       04   AC   D0 00017          MOVL    DESCRIP, R7                                     ; 0970
                        52       FC   A7   D0 0001B          MOVL    -4(R7), CHAN                                    ;
                                      50   D4 0001F          CLRL    R0                                              ; 0976
                             00000000G 00   16 00021          JSB     BAS$$CB_PUSH                                    ;
                        6E             5B   D0 00027          MOVL    CCB, SAVE_CCB                                   ; 0977
                        0B       FC    AB   E8 0002A          BLBS    -4(CCB), T$                                     ; 0979
                        7E       00G   8F   9A 0002E          MOVZBL  #BAS$K_VIRARROPE, -(SP)                         ;
              00000000G 00             01   FB 00032          CALLS   #1, BAS$$STOP                                   ;
                        05       C4    AB   91 00039 1$:      CMPB    -60(CCB), #5                                    ; 0987
                                      07   13 0003D          BEQL    2$                                              ;
                        7E       00G   8F   9A 0003F          MOVZBL  #BAS$K_VIRARRDIS, -(SP)                         ;
                        69             01   FB 00043          CALLS   #1, BAS$$STOP_IO                                ;
                  07          FF   AB   01   E1 00046 2$:      BBC     #1, -1(CCB), 3$                                 ; 0993
                        7E       00G   8F   9A 0004B          MOVZBL  #BAS$K_ILLOPE, -(SP)                            ;
                        69             01   FB 0004F          CALLS   #1, BAS$$STOP_IO                                ;
                    0200    8F       20   AB   B1 00052 3$:      CMPW    32(CCB), #512                                   ; 0999
                                      07   13 00058          BEQL    4$                                              ;
                        7E       00G   8F   9A 0005A          MOVZBL  #BAS$K_VIRBUFTOO, -(SP)                         ;
                        69             01   FB 0005E          CALLS   #1, BAS$$STOP_IO                                ;
                        FF       AB         01   88 00061 4$:      BISB2   #1, -1(CCB)                                     ; 1004
                        1E       AB         01   90 00065          MOVB    #1, 30(CCB)                                     ; 1008
                        06       AB   80   8F   88 00069          BISB2   #128, 6(CCB)                                    ; 1012
                                 AB   A4   AB   D5 0006E          TSTL    -92(CCB)                                        ; 1018
                                      06   12 00071          BNEQ    5$                                              ;
                        A4       AB   0000V CF   9E 00073          MOVAB   BAS$$VA_CLOSE, -92(CCB)                         ;
                        50       0000V CF   9E 00079 5$:      MOVAB   BAS$$VA_CLOSE, R0                               ; 1020
                        50         A4   AB   D1 0007E          CMPL    -92(CCB), R0                                    ;
                                      07   13 00082          BEQL    6$                                              ;
                        7E       00G   8F   9A 00084          MOVZBL  #BAS$K_PROLOSSOR, -(SP)                         ;
                        69             01   FB 00088          CALLS   #1, BAS$$STOP_IO                                ;
                  58          08   AC   F8   A7   C1 0008B 6$:      ADDL3   -8(R7), INDEX, R8                               ; 1029
                  50             58 00000200 8F   C7 00091          DIVL3   #512, R8, R0                                    ;
                                 53       01   A0   9E 00099          MOVAB   1(R0), R3                                       ;
                                 53       E0   AB   D1 0009D          CMPL    -32(CCB), R3                                    ;
                                      04   12 000A1          BNEQ    7$                                              ;
                                      54   D4 000A3          CLRL    READ_RECORD                                     ; 1031
                                      5C   11 000A5          BRB     13$                                             ;
                  54          FE   AB         03   E1 000A7 7$:      BBC     #3, -2(CCB), 12$                                ; 1039
                                      5B   DD 000AC          PUSHL   CCB                                             ; 1042
                    00000000G 00             01   FB 000AE          CALLS   #1, SYS$PUT                                     ;
                                 52         50   D0 000B5          MOVL    R0, PUT_STATUS                                  ;
                    00010651    8F         52   D1 000B8          CMPL    PUT_STATUS, #67153                              ; 1044
                                      03   12 000BF          BNEQ    8$                                              ;
                        6A             00   FB 000C1          CALLS   #0, BAS$$SIGNAL_CTRLC                           ; 1046
                        35             52   E8 000C4 8$:      BLBS    PUT_STATUS, 11$                                 ; 1054
                    000182DA    8F         52   D1 000C7 9$:      CMPL    PUT_STATUS, #99034                              ; 1058
                                      23   12 000CE          BNEQ    10$                                             ;
                                      5B   DD 000D0          PUSHL   CCB                                             ; 1060
                    00000000G 00             01   FB 000D2          CALLS   #1, SYS$WAIT                                    ;
                                      5B   DD 000D9          PUSHL   CCB                                             ; 1061
                    00000000G 00             01   FB 000DB          CALLS   #1, SYS$PUT                                     ;
                                 52         50   D0 000E2          MOVL    R0, PUT_STATUS                                  ;
                    00010651    8F         52   D1 000E5          CMPL    PUT_STATUS, #67153                              ; 1063
                                      D9   12 000EC          BNEQ    9$                                              ;
                        6A             00   FB 000EE          CALLS   #0, BAS$$SIGNAL_CTRLC                           ; 1065
                                      D4   11 000F1          BRB     9$                                              ; 1058
```

```
                                 06              52  E8 000F3 10$:    BLBS    PUT_STATUS, 11$                                  ; 1069
                                 7E              01  CE 000F6        MNEGL   #1, -(SP)                                        ;
                                 69              01  FB 000F9        CALLS   #1, BAS$$STOP_IO                                 ;
                              FE AB              08  8A 000FC 11$:    BICB2   #8, -2(CCB)                                      ; 1076
                                 54              01  D0 00100 12$:    MOVL    #1, READ_RECORD                                  ; 1079
                                 7A              54  E9 00103 13$:    BLBC    READ_RECORD, 19$                                 ; 1086
                              E0 AB              53  D0 00106        MOVL    R3, -32(CCB)                                     ; 1089
                                                 5B  DD 0010A        PUSHL   CCB                                              ; 1090
                    00000000G 00                 01  FB 0010C        CALLS   #1, SYS$GET                                      ;
                                 56              50  D0 00113        MOVL    R0, GET_STATUS                                   ;
                    00010651  8F                 56  D1 00116        CMPL    GET_STATUS, #67153                               ; 1092
                                                 03  12 0011D        BNEQ    14$                                              ;
                                 6A              00  FB 0011F        CALLS   #0, BAS$$SIGNAL_CTRLC                            ; 1094
                    0001827A  8F                 56  D1 00122 14$:    CMPL    GET_STATUS, #98938                               ; 1101
                                                 09  13 00129        BEQL    15$                                              ;
                    00018049  8F                 56  D1 0012B        CMPL    GET_STATUS, #98377                               ;
                                                 14  12 00132        BNEQ    16$                                              ;
 20 AB          00              6E              00  2C 00134 15$:    MOVC5   #0, (SP), #0, 32(CCB), @36(CCB)                  ; 1104
                                                 24  BB    0013A                                                              ;
                       28 AB     24 AB           D0  0013C        MOVL    36(CCB), 40(CCB)                                 ; 1105
                       22 AB     20 AB           B0  00141        MOVW    32(CCB), 34(CCB)                                 ; 1106
                                                 38  11  00146        BRB     19$                                              ; 1101
                                 35              56  E8 00148 16$:    BLBS    GET_STATUS, 19$                                  ; 1111
                    000182DA  8F                 56  D1 0014B 17$:    CMPL    GET_STATUS, #99034                               ; 1118
                                                 23  12 00152        BNEQ    18$                                              ;
                                                 5B  DD 00154        PUSHL   CCB                                              ; 1120
                    00000000G 00                 01  FB 00156        CALLS   #1, SYS$WAIT                                     ;
                                                 5B  DD 0015D        PUSHL   CCB                                              ; 1121
                    00000000G 00                 01  FB 0015F        CALLS   #1, SYS$GET                                      ;
                                 56              50  D0 00166        MOVL    R0, GET_STATUS                                   ;
                    00010651  8F                 56  D1 00169        CMPL    GET_STATUS, #67153                               ; 1123
                                                 D9  12 00170        BNEQ    17$                                              ;
                                 6A              00  FB 00172        CALLS   #0, BAS$$SIGNAL_CTRLC                            ; 1125
                                                 D4  11 00175        BRB     17$                                              ; 1118
                                 06              56  E8 00177 18$:    BLBS    GET_STATUS, 19$                                  ; 1129
                                 7E              01  CE 0017A        MNEGL   #1, -(SP)                                        ;
                                 69              01  FB 0017D        CALLS   #1, BAS$$STOP_IO                                 ;
                                 56        0C    AC  D0 00180 19$:    MOVL    VALUE, R6                                        ; 1144
                                 15        02    A7  91 00184        CMPB    2(R7), #21                                       ; 1141
                                                 16  13 00188        BEQL    20$                                              ;
 7E             00               58              01  7A 0018A        EMUL    #1, R8, #0, -(SP)                                ; 1144
 50             50               8E 00000200     8F  7B 0018F        EDIV    #512, (SP)+, R0, R0                              ;
                66        28 BB40                 67  28 00198        MOVC3   (R7), @40(CCB)[R0], (R6)                         ;
                                                 23  11  0019E        BRB     21$                                              ; 1143
                                 51        08    A7  98 001A0 20$:    CVTBL   8(R7), COUNT                                     ; 1152
                                 50        08    A6  98 001A4        CVTBL   8(R6), R0                                        ;
                                 51              50  C2 001A8        SUBL2   R0, COUNT                                        ;
 7E             00               58              01  7A 001AB        EMUL    #1, R8, #0, -(SP)                                ; 1154
 50             50               8E 00000200     8F  7B 001B0        EDIV    #512, (SP)+, R0, R0                              ;
 00        28 BB40               67              51  F8 001B9        ASHP    COUNT, (R7), @40(CCB)[R0], #0, (R6), @4(R6)      ; 1155
                       04  B6                     66      001C0                                                              ;
                    00000000G 00                 16  001C3 21$:    JSB     BAS$$CB_POP                                      ; 1161
                                                 04  001C9        RET                                                      ; 1162
                                                0000 001CA 22$:    .WORD   Save nothing                                     ; 0941
                                 50        08    AC  D0 001CC        MOVL    8(AP), R0                                        ;
                                 50        04    A0  D0 001D0        MOVL    4(R0), R0                                        ;
                                           FC    A0  9F 001D4        PUSHAB  SAVE_CCB                                         ;
```

```
                                        01  DD 001D7          PUSHL   #1                                      ;
                                        5E  DD 001D9          PUSHL   SP                                      ;
                              7E    04  AC  7D 001DB          MOVQ    4(AP), -(SP)                            ;
                        0000V CF          03  FB 001DF          CALLS   #3, HANDLER                             ;
                                              04 001E4          RET                                           ;

; Routine Size:  485 bytes,    Routine Base:  _BAS$CODE + 0000

;  452          1163  1
```

```
;  454          1164  1 GLOBAL ROUTINE BAS$$VA_STORE (                        ! Store routine
;  455          1165  1         DESCRIP,                                      ! The descriptor for this virtual array
;  456          1166  1         INDEX,                                        ! Linearized index
;  457          1167  1         VALUE                                         ! Where to find item for array
;  458          1168  1     ) : NOVALUE =
;  459          1169  1
;  460          1170  1 !++
;  461          1171  1 ! FUNCTIONAL DESCRIPTION:
;  462          1172  1 !
;  463          1173  1 !       Store a value in a virtual array.  Multiple bytes may be stored
;  464          1174  1 !       with a single call.
;  465          1175  1 !
;  466          1176  1 ! FORMAL PARAMETERS:
;  467          1177  1 !
;  468          1178  1 !       DESCRIP.mz.r    The descriptor for the virtual array
;  469          1179  1 !       INDEX.rl.v      The byte offset into the array
;  470          1180  1 !       VALUE.rz.r      The place from which to fetch the value.  The
;  471          1181  1 !                       number of bytes to store is in the LENGTH field
;  472          1182  1 !                       of DESCRIP.
;  473          1183  1 !
;  474          1184  1 ! IMPLICIT INPUTS:
;  475          1185  1 !
;  476          1186  1 !       NONE
;  477          1187  1 !
;  478          1188  1 ! IMPLICIT OUTPUTS:
;  479          1189  1 !
;  480          1190  1 !       NONE
;  481          1191  1 !
;  482          1192  1 ! ROUTINE VALUE:
;  483          1193  1 ! COMPLETION CODES:
;  484          1194  1 !
;  485          1195  1 !       NONE
;  486          1196  1 !
;  487          1197  1 ! SIDE EFFECTS:
;  488          1198  1 !
;  489          1199  1 !       Signals if an error is encountered.
;  490          1200  1 !
;  491          1201  1 !--
;  492          1202  1
;  493          1203  2     BEGIN
;  494          1204  2
;  495          1205  2     MAP
;  496          1206  2         DESCRIP : REF BLOCK [8, BYTE];
;  497          1207  2
;  498          1208  2     GLOBAL REGISTER
;  499          1209  2         CCB = K_CCB_REG : REF BLOCK [, BYTE];
;  500          1210  2
;  501          1211  2     BUILTIN
;  502          1212  2         ASHP;
;  503          1213  2
;  504          1214  2     LOCAL
;  505          1215  2         CHAN,                                         ! The channel this array is defined on
;  506          1216  2         HANDLE,                                       ! Pointer to info for this array
;  507          1217  2         GET_STATUS,                                   ! Last RMS GET status
;  508          1218  2         PUT_STATUS,                                   ! Last RMS PUT status
;  509          1219  2         READ_RECORD,                                  ! 1 = we must read the record
;  510          1220  2         SAVE_CCB : VOLATILE;                          ! CCB to POP, or 0
```

```
;  511          1221  2
;  512          1222  2 !+
;  513          1223  2 ! Establish a handler to pop the CCB on unwind.
;  514          1224  2 !-
;  515          1225  2
;  516          1226  2     ENABLE
;  517          1227  2         HANDLER (SAVE_CCB);
;  518          1228  2
;  519          1229  2 !+
;  520          1230  2 ! Fetch the array's channel number from its descriptor
;  521          1231  2 !-
;  522          1232  2     CHAN = .DESCRIP [DSC$L_LOGUNIT];
;  523          1233  2 !+
;  524          1234  2 ! Get a pointer to the LUB/ISB/RAB for this channel.  If the channel has not
;  525          1235  2 ! been opened yet, this call will allocate the LUB/ISB/RAB, but we will reject
;  526          1236  2 ! it for lack of the LUB$V_OPENED bit.
;  527          1237  2 !-
;  528          1238  2     BAS$$CB_PUSH (.CHAN, LUB$K_LUN_MIN);
;  529          1239  2     SAVE_CCB = .CCB;
;  530          1240  2
;  531          1241  2     IF ( NOT .CCB [LUB$V_OPENED]) THEN BAS$$STOP (BAS$K_VIRARROPE);
;  532          1242  2
;  533          1243  2 !+
;  534          1244  2 ! If the channel was not opened with organization VIRTUAL, reject it.  This
;  535          1245  2 ! also catches channel 0, which is always open but never has VIRTUAL
;  536          1246  2 ! organization.
;  537          1247  2 !-
;  538          1248  2
;  539          1249  2     IF (.CCB [LUB$B_ORGAN] NEQ LUB$K_ORG_VIRTU) THEN BAS$$STOP_IO (BAS$K_VIRARRDIS);
;  540          1250  2
;  541          1251  2 !+
;  542          1252  2 ! If this channel has been used for block I/O, reject it.
;  543          1253  2 !-
;  544          1254  2
;  545          1255  2     IF (.CCB [LUB$V_BLK_USE]) THEN BAS$$STOP_IO (BAS$K_ILLOPE);
;  546          1256  2
;  547          1257  2 !+
;  548          1258  2 ! If the recordsize of the file is not 512 bytes, reject it.
;  549          1259  2 !-
;  550          1260  2
;  551          1261  2     IF (.CCB [RAB$W_USZ] NEQ K_BLOCK_LENGTH) THEN BAS$$STOP_IO (BAS$K_VIRBUFTOO);
;  552          1262  2
;  553          1263  2 !+
;  554          1264  2 ! If the file is marked read only, reject it.
;  555          1265  2 !-
;  556          1266  2
;  557          1267  2     IF (.CCB [LUB$V_READ_ONLY]) THEN BAS$$STOP_IO (BAS$K_ILLILLACC);
;  558          1268  2
;  559          1269  2 !+
;  560          1270  2 ! Mark the LUB as being used for a virtual array.
;  561          1271  2 !-
;  562          1272  2     CCB [LUB$V_VA_USE] = 1;
;  563          1273  2 !+
;  564          1274  2 ! Record access will always be by key
;  565          1275  2 !-
;  566          1276  2     CCB [RAB$B_RAC] = RAB$C_KEY;
;  567          1277  2 !+
```

```
;  568          1278  2 ! Set the RAB so that a $GET to a non-existent record will still lock
;  569          1279  2 ! that record.
;  570          1280  2 !-
;  571          1281  2     CCB [RAB$V_NXR] = 1;
;  572          1282  2 !+
;  573          1283  2 ! Set the address of our CLOSE appendage in the LUB.  If somebody else's
;  574          1284  2 ! is already there, we have a serious problem.
;  575          1285  2 !-
;  576          1286  2
;  577          1287  2     IF (.CCB [LUB$A_CLOSE] EQLA 0) THEN CCB [LUB$A_CLOSE] = BAS$$VA_CLOSE;
;  578          1288  2
;  579          1289  2     IF (.CCB [LUB$A_CLOSE] NEQA BAS$$VA_CLOSE) THEN BAS$$STOP_IO (BAS$K_PROLOSSOR);
;  580          1290  2
;  581          1291  2 !+
;  582          1292  2 ! If this is not the first reference to this file, we may have to
;  583          1293  2 ! write out the current buffer.  We will write only if the current buffer
;  584          1294  2 ! is not the buffer we wish to access.  LUB$L_LOG_RECNO is initialized
;  585          1295  2 ! to zero for virtual files.
;  586          1296  2 !-
;  587          1297  2
;  588          1298  3     IF (.CCB [LUB$L_LOG_RECNO] EQL ((.INDEX + .DESCRIP [DSC$L_BYTEOFF])/K_BLOCK_LENGTH) + 1)
;  589          1299  2     THEN
;  590          1300  2         READ_RECORD = 0
;  591          1301  2     ELSE
;  592          1302  3         BEGIN
;  593          1303  3 !+
;  594          1304  3 ! We actually do the PUT only if the buffer has been changed since we last
;  595          1305  3 ! read it, as recorded by LUB$V_OUTBUF_DR.
;  596          1306  3 !-
;  597          1307  3
;  598          1308  4         IF (.CCB [LUB$V_OUTBUF_DR])
;  599          1309  3         THEN
;  600          1310  4             BEGIN
;  601          1311  4             PUT_STATUS = $PUT (RAB = .CCB);
;  602          1312  4
;  603          1313  4         IF .PUT_STATUS EQL RMS$_CONTROLC                                    ;
;  604          1314  4         THEN
;  605          1315  4             BAS$$SIGNAL_CTRLC ();
;  606          1316  4                                                                             ;
;  607          1317  4 !+
;  608          1318  4 ! If the PUT fails, we must worry about the RSA error, which can happen if
;  609          1319  4 ! we are running at AST level, and the AST interrupted some RMS I/O.  If
;  610          1320  4 ! we get this error, wait for it to go away.  Any other RMS error is fatal.
;  611          1321  4 !-
;  612          1322  4
;  613          1323  5             IF ( NOT .PUT_STATUS)
;  614          1324  4             THEN
;  615          1325  5                 BEGIN
;  616          1326  5
;  617          1327  5                 WHILE (.PUT_STATUS EQL RMS$_RSA) DO
;  618          1328  6                     BEGIN
;  619          1329  6                     $WAIT (RAB = .CCB);
;  620          1330  6                     PUT_STATUS = $PUT (RAB = .CCB);
;  621          1331  6
;  622          1332  6                     IF .PUT_STATUS EQL RMS$_CONTROLC
;  623          1333  6                     THEN
;  624          1334  6                         BAS$$SIGNAL_CTRLC ();
```

```
;  625          1335  6                        END;
;  626          1336  5
;  627          1337  5
;  628          1338  5                    IF ( NOT .PUT_STATUS) THEN BAS$$STOP_IO (BAS$K_IOERR_REC);
;  629          1339  5
;  630          1340  4                    END;
;  631          1341  4
;  632          1342  4 !+
;  633          1343  4 ! The buffer is no longer "dirty", mark it so.
;  634          1344  4 !-
;  635          1345  4                CCB [LUB$V_OUTBUF_DR] = 0;
;  636          1346  4                END;
;  637          1347  3
;  638          1348  3            READ_RECORD = 1;
;  639          1349  2            END;
;  640          1350  2
;  641          1351  2 !+
;  642          1352  2 ! If necessary, read in the record containing the element we want.
;  643          1353  2 !-
;  644          1354  2
;  645          1355  3        IF (.READ_RECORD)
;  646          1356  2        THEN
;  647          1357  3            BEGIN
;  648          1358  3            CCB [LUB$L_LOG_RECNO] = ((.INDEX + .DESCRIP [DSC$L_BYTEOFF])/K_BLOCK_LENGTH) + 1;
;  649          1359  3            GET_STATUS = $GET (RAB = .CCB);
;  650          1360  3
;  651          1361  3            IF .GET_STATUS EQL RMS$_CONTROLC
;  652          1362  3            THEN
;  653          1363  3                BAS$$SIGNAL_CTRLC ();
;  654          1364  3
;  655          1365  3 !+
;  656          1366  3 ! If we are at EOF, extend the file.  This is compatable with the PDP-11.
;  657          1367  3 !-
;  658          1368  3
;  659          1369  4            IF ((.GET_STATUS EQL RMS$_EOF) OR (.GET_STATUS EQL RMS$_OK_RNF))
;  660          1370  3            THEN
;  661          1371  4                BEGIN
;  662          1372  4
;  663          1373  4                LOCAL
;  664          1374  4                    FAB_BLOCK : $FAB_DECL,
;  665          1375  4                    EXTEND_STATUS;
;  666          1376  4
;  667          1377  4 !+
;  668          1378  4 ! If the file is already allocated beyond the current end-of-file point
;  669          1379  4 ! (which can happen on disk if the cluster size is greater than 1) then
;  670          1380  4 ! do not do any allocation.
;  671          1381  4 !-
;  672          1382  4
;  673          1383  5                IF (.CCB [LUB$L_LOG_RECNO] GTR .CCB [LUB$L_REC_MAX])
;  674          1384  4                THEN
;  675          1385  5                    BEGIN
;  676          1386  5                    $FAB_INIT (FAB = FAB_BLOCK);
;  677          1387  5                    FAB_BLOCK [FAB$L_ALQ] = .CCB [LUB$L_LOG_RECNO] - .CCB [LUB$L_REC_MAX];
;  678          1388  5                    FAB_BLOCK [FAB$W_IFI] = .CCB [LUB$W_IFI];
;  679          1389  5                    CCB [LUB$A_FAB] = FAB_BLOCK;
;  680          1390  5                    CCB [RAB$L_STS] = SS$_NORMAL;
;  681          1391  5                    EXTEND_STATUS = $EXTEND (FAB = FAB_BLOCK);
```

```
;  682          1392  5
;  683          1393  5                IF ( NOT .EXTEND_STATUS) THEN BAS$$STOP_IO (BAS$K_IOERR_REC);
;  684          1394  5
;  685          1395  5                CCB [LUB$L_REC_MAX] = .CCB [LUB$L_REC_MAX] + .FAB_BLOCK [FAB$L_ALQ];
;  686          1396  5                CCB [LUB$A_FAB] = 0;
;  687          1397  4                END;
;  688          1398  4
;  689          1399  4 !+
;  690          1400  4 ! Since we did not really read a record, make sure the buffer contains
;  691          1401  4 ! all zeros.
;  692          1402  4 !-
;  693          1403  4                CH$FILL (0, .CCB [RAB$W_USZ], .CCB [RAB$L_UBF]);
;  694          1404  4                CCB [RAB$W_RSZ] = .CCB [RAB$W_USZ];
;  695          1405  4                CCB [RAB$L_RBF] = .CCB [RAB$L_UBF];
;  696          1406  4                END
;  697          1407  3            ELSE
;  698          1408  4                BEGIN
;  699          1409  4
;  700          1410  5                IF ( NOT .GET_STATUS)
;  701          1411  4                THEN
;  702          1412  5                    BEGIN
;  703          1413  5 !+
;  704          1414  5 ! Again, worry about the RSA RMS error.
;  705          1415  5 !-
;  706          1416  5
;  707          1417  5                    WHILE (.GET_STATUS EQL RMS$_RSA) DO
;  708          1418  6                        BEGIN
;  709          1419  6                        $WAIT (RAB = .CCB);
;  710          1420  6                        GET_STATUS = $GET (RAB = .CCB);
;  711          1421  6
;  712          1422  6                        IF .GET_STATUS EQL RMS$_CONTROLC
;  713          1423  6                        THEN
;  714          1424  6                            BAS$$SIGNAL_CTRLC ();
;  715          1425  6
;  716          1426  5                        END;
;  717          1427  5
;  718          1428  5                    IF ( NOT .GET_STATUS) THEN BAS$$STOP_IO (BAS$K_IOERR_REC);
;  719          1429  5
;  720          1430  4                    END;
;  721          1431  4
;  722          1432  3                END;
;  723          1433  3
;  724          1434  2            END;
;  725          1435  2
;  726          1436  2 !+
;  727          1437  2 ! At this point, the proper record is in the buffer, and we can copy
;  728          1438  2 ! data to it.
;  729          1439  2 !-
;  730          1440  2        IF .DESCRIP [DSC$B_DTYPE] NEQ DSC$K_DTYPE_P
;  731          1441  2        THEN
;  732          1442  2        CH$MOVE (.DESCRIP [DSC$W_LENGTH], .VALUE,
;  733          1443  2            .CCB [RAB$L_RBF] + (T.INDEX + .DESCRIP [DSC$L_BYTEOFF]) MOD K_BLOCK_LENGTH))
;  734          1444  2        ELSE
;  735          1445  3            BEGIN
;  736          1446  3            MAP
;  737          1447  3                DESCRIP : REF BLOCK [12,BYTE],
;  738          1448  3                VALUE : REF BLOCK [12,BYTE];
```

```
;  739          1449  3          LOCAL
;  740          1450  3              COUNT;
;  741          1451  3          COUNT = .VALUE [DSC$B_SCALE] - .DESCRIP [DSC$B_SCALE];
;  742          1452  3          ASHP (COUNT, VALUE [DSC$W_LENGTH],
;  743          1453  3              .VALUE [DSC$A_POINTER], %REF(0), DESCRIP [DSC$W_LENGTH],
;  744          1454  3              .CCB [RAB$L_RBF] + ((.INDEX + .DESCRIP [DSC$L_BYTEOFF]) MOD K_BLOCK_LENGTH));
;  745          1455  2          END;
;  746          1456  2      !+
;  747          1457  2      ! Since the buffer differs from the disk, mark it "dirty" so it will be
;  748          1458  2      ! written out.
;  749          1459  2      !-
;  750          1460  2          CCB [LUB$V_OUTBUF_DR] = 1;
;  751          1461  2      !+
;  752          1462  2      ! Done with this I/O channel.
;  753          1463  2      !-
;  754          1464  2          BAS$$CB_POP ();
;  755          1465  1          END;                                    ! end of BAS$$VA_STORE


                                                .EXTRN  SYS$EXTEND

                                    0FFC 00000  .ENTRY  BAS$$VA_STORE, Save R2,R3,R4,R5,R6,R7,R8,-   ; 1164
                                                        R9,R10,R11
                5A 00000000G    00  9E 00002    MOVAB   BAS$$STOP_IO, R10
                5E          AC  AE  9E 00009    MOVAB   -84(SP), SP
                            50  AE  D4 0000D    CLRL    SAVE_CCB                                    ; 1203
                6D        0222  CF  DE 00010    MOVAL   26$, (FP)
                57          04  AC  D0 00015    MOVL    DESCRIP, R7                                 ; 1232
                52          FC  A7  D0 00019    MOVL    -4(R7), CHAN
                            50  D4  0001D        CLRL    R0                                          ; 1238
                    00000000G   00  16 0001F    JSB     BAS$$CB_PUSH
                        50  AE      5B  D0 00025    MOVL    CCB, SAVE_CCB                           ; 1239
                        0B      FC  AB  E8 00029    BLBS    -4(CCB), T$                             ; 1241
                        7E      00G 8F  9A 0002D    MOVZBL  #BAS$K_VIRARROPE, -(SP)
                00000000G       00      01  FB 00031    CALLS   #1, BAS$$STOP
                        05      C4  AB  91 00038 1$:    CMPB    -60(CCB), #5                        ; 1249
                                    07  13 0003C        BEQL    2$
                        7E      00G 8F  9A 0003E        MOVZBL  #BAS$K_VIRARRDIS, -(SP)
                        6A          01  FB 00042        CALLS   #1, BAS$$STOP_IO
                        58      FE  AB  9E 00045 2$:    MOVAB   -2(CCB), R8                         ; 1255
                07      68          09  E1 00049        BBC     #9, (R8), 3$
                        7E      00G 8F  9A 0004D        MOVZBL  #BAS$K_ILLOPE, -(SP)
                        6A          01  FB 00051        CALLS   #1, BAS$$STOP_IO
                    0200    8F      20  AB  B1 00054 3$:    CMPW    32(CCB), #512                   ; 1261
                                    07  13 0005A        BEQL    4$
                        7E      00G 8F  9A 0005C        MOVZBL  #BAS$K_VIRBUFTOO, -(SP)
                        6A          01  FB 00060        CALLS   #1, BAS$$STOP_IO
                07      FC      AB      02  E1 00063 4$:    BBC     #2, -4(CCB), 5$                 ; 1267
                        7E      00G 8F  9A 00068        MOVZBL  #BAS$K_ILLILLACC, -(SP)
                        6A          01  FB 0006C        CALLS   #1, BAS$$STOP_IO
                        01      A8      01  88 0006F 5$:    BISB2   #1, 1(R8)                       ; 1272
                        1E      AB      01  90 00073        MOVB    #1, 30(CCB)                     ; 1276
                        06      AB  80  8F  88 00077        BISB2   #128, 6(CCB)                    ; 1281
                                    A4  AB  D5 0007C        TSTL    -92(CCB)                        ; 1287
                                        06  12 0007F        BNEQ    6$
                        A4      AB  0000V   CF  9E 00081    MOVAB   BAS$$VA_CLOSE, -92(CCB)
```

```
                        50      0000V   CF   9E 00087 6$:     MOVAB   BAS$$VA_CLOSE, R0                       ; 1289
                        50        A4    AB   D1 0008C         CMPL    -92(CCB), R0
                                        07   13 00090         BEQL    7$
                        7E        00G   8F   9A 00092         MOVZBL  #BAS$K_PROLOSSOR, -(SP)
                        6A              01   FB 00096         CALLS   #1, BAS$$STOP_IO
                59      08        AC      F8 A7   C1 00099 7$:     ADDL3   -8(R7), INDEX, R9                   ; 1298
                50              59 00000200   8F   C7 0009F         DIVL3   #512, R9, R0
                                53        01   A0   9E 000A7         MOVAB   1(R0), R3
                                53        E0   AB   D1 000AB         CMPL    -32(CCB), R3
                                               04   12 000AF         BNEQ    8$
                                               54   D4 000B1         CLRL    READ_RECORD                    ; 1300
                                               62   11 000B3         BRB     14$
                5B                      68     03   E1 000B5 8$:     BBC     #3, (R8), 13$                  ; 1308
                                               5B   DD 000B9         PUSHL   CCB                            ; 1311
                    00000000G           00     01   FB 000BB         CALLS   #1, SYS$PUT
                                            52   50   D0 000C2         MOVL    R0, PUT_STATUS
                    00010651            8F     52   D1 000C5         CMPL    PUT_STATUS, #67153             ; 1313
                                               07   12 000CC         BNEQ    9$
                    00000000G           00     00   FB 000CE         CALLS   #0, BAS$$SIGNAL_CTRLC          ; 1315
                                            39   52   E8 000D5 9$:     BLBS    PUT_STATUS, 12$                ; 1323
                    000182CA            8F     52   D1 000D8 10$:    CMPL    PUT_STATUS, #99034             ; 1327
                                               27   12 000DF         BNEQ    11$
                                               5B   DD 000E1         PUSHL   CCB                            ; 1329
                    00000000G           00     01   FB 000E3         CALLS   #1, SYS$WAIT
                                               5B   DD 000EA         PUSHL   CCB                            ; 1330
                    00000000G           00     01   FB 000EC         CALLS   #1, SYS$PUT
                                            52   50   D0 000F3         MOVL    R0, PUT_STATUS
                    00010651            8F     52   D1 000F6         CMPL    PUT_STATUS, #67153             ; 1332
                                               D9   12 000FD         BNEQ    10$
                    00000000G           00     00   FB 000FF         CALLS   #0, BAS$$SIGNAL_CTRLC          ; 1334
                                               D0   11 00106         BRB     10$                            ; 1327
                                            06   52   E8 00108 11$:    BLBS    PUT_STATUS, 12$                ; 1338
                                            7E   01   CE 0010B         MNEGL   #1, -(SP)
                                            6A   01   FB 0010E         CALLS   #1, BAS$$STOP_IO
                                            68   08   8A 00111 12$:    BICB2   #8, (R8)                       ; 1345
                                            54   01   D0 00114 13$:    MOVL    #1, READ_RECORD                ; 1348
                                            03   54   E8 00117 14$:    BLBS    READ_RECORD, 15$               ; 1355
                                                 00CC 31   0011A         BRW     23$
                                      E0    AB     53   D0 0011D 15$:    MOVL    R3, -32(CCB)                   ; 1358
                                               5B   DD 00121         PUSHL   CCB                            ; 1359
                        00000000G           00     01   FB 00123         CALLS   #1, SYS$GET
                                            56   50   D0 0012A         MOVL    R0, GET_STATUS
                        00010651            8F     56   D1 0012D         CMPL    GET_STATUS, #67153             ; 1361
                                               07   12 00134         BNEQ    16$
                        00000000G           00     00   FB 00136         CALLS   #0, BAS$$SIGNAL_CTRLC          ; 1363
                        0001827A            8F     56   D1 0013D 16$:    CMPL    GET_STATUS, #98938             ; 1369
                                               09   13 00144         BEQL    17$
                        00018049            8F     56   D1 00146         CMPL    GET_STATUS, #98377
                                               5E   12 0014D         BNEQ    20$
                                      E4    AB     E0   AB   D1 0014F 17$:    CMPL    -32(CCB), -28(CCB)             ; 1383
                                               43   15 00154         BLEQ    19$
        0050    8F              00             6E   00   2C 00156         MOVC5   #0, (SP), #0, #80, $RMS_PTR    ; 1386
                                               6E       0015D
                                               6E  5003   8F   B0 0015E         MOVW    #20483, $RMS_PTR
                                            16    AE   02   90 00163         MOVB    #2, $RMS_PTR+22
                                            1F    AE   02   90 00167         MOVB    #2, $RMS_PTR+31
                                10  AE      E0    AB     E4   AB   C3 0016B         SUBL3   -28(CCB), -32(CCB), FAB_BLOCK+16       ; 1387
```

```
                                02  AE      D0  AB  B0 00172          MOVW    -48(CCB), FAB_BLOCK+2                   ; 1388
                                E8  AB          6E  9E 00177          MOVAB   FAB_BLOCK, -24(CCB)                     ; 1389
                                08  AB          01  D0 0017B          MOVL    #1, 8(CCB)                              ; 1390
                                                5E  DD 0017F          PUSHL   SP                                      ; 1391
                         00000000G  00          01  FB 00181          CALLS   #1, SYS$EXTEND
                                06              50  E8 00188          BLBS    EXTEND_STATUS, 18$                      ; 1393
                                7E              01  CE 0018B          MNEGL   #1, -(SP)
                                6A              01  FB 0018E          CALLS   #1, BAS$$STOP_IO
                                E4  AB      10  AE  C0 00191  18$:     ADDL2   FAB_BLOCK+16, -28(CCB)                  ; 1395
                                            E8  AB  D4 00196          CLRL    -24(CCB)                                ; 1396
    20  AB              00              6E          2C 00199  19$:     MOVC5   #0, (SP), #0, 32(CCB), @36(CCB)         ; 1403
                                                24  BB    0019F
                                22  AB      20  AB  B0 001A1          MOVW    32(CCB), 34(CCB)                        ; 1404
                                28  AB      24  AB  D0 001A6          MOVL    36(CCB), 40(CCB)                        ; 1405
                                            3C  11 001AB              BRB     23$                                     ; 1369
                                39              56  E8 001AD  20$:     BLBS    GET_STATUS, 23$                         ; 1410
                         000182DA   8F          56  D1 001B0  21$:     CMPL    GET_STATUS, #99034                      ; 1417
                                                27  12 001B7          BNEQ    22$
                                                5B  DD 001B9          PUSHL   CCB                                     ; 1419
                         00000000G  00          01  FB 001BB          CALLS   #1, SYS$WAIT
                                                5B  DD 001C2          PUSHL   CCB                                     ; 1420
                         00000000G  00          01  FB 001C4          CALLS   #1, SYS$GET
                                56              50  D0 001CB          MOVL    R0, GET_STATUS
                         00010651   8F          56  D1 001CE          CMPL    GET_STATUS, #67153                      ; 1422
                                                D9  12 001D5          BNEQ    21$
                         00000000G  00          00  FB 001D7          CALLS   #0, BAS$$SIGNAL_CTRLC                   ; 1424
                                                D0  11 001DE          BRB     21$                                     ; 1417
                                06              56  E8 001E0  22$:     BLBS    GET_STATUS, 23$                         ; 1428
                                7E              01  CE 001E3          MNEGL   #1, -(SP)
                                6A              01  FB 001E6          CALLS   #1, BAS$$STOP_IO
                                56          0C  AC  D0 001E9  23$:     MOVL    VALUE, R6                               ; 1442
                                15          02  A7  91 001ED          CMPB    2(R7), #21                              ; 1440
                                                16  13 001F1          BEQL    24$
    7E              00              59          01  7A 001F3          EMUL    #1, R9, #0, -(SP)                       ; 1443
    50              50              8E  00000200    8F  7B 001F8          EDIV    #512, (SP)+, R0, R0
            28  BB40                  66          67  28 00201          MOVC3   (R7), (R6), @40(CCB)[R0]
                                                23  11 00207          BRB     25$                                     ; 1442
                                51          08  A6  98 00209  24$:     CVTBL   8(R6), COUNT                            ; 1451
                                50          08  A7  98 0020D          CVTBL   8(R7), R0
                                51              50  C2 00211          SUBL2   R0, COUNT
    7E              00              59          01  7A 00214          EMUL    #1, R9, #0, -(SP)                       ; 1454
    50              50              8E  00000200    8F  7B 00219          EDIV    #512, (SP)+, R0, R0
    00      04      B6              66          51  F8 00222          ASHP    COUNT, (R6), @4(R6), #0, (R7), @40(CCB)[R0]
                        28  BB40                67      00228
                                68              08  88 0022C  25$:     BISB2   #8, (R8)                                ; 1460
                                    00000000G   00  16 0022F          JSB     BAS$$CB_POP                             ; 1464
                                                04 00235              RET                                             ; 1465
                                            0000 00236  26$:     .WORD   Save nothing                            ; 1203
                                50          08  AC  D0 00238          MOVL    8(AP), R0
                                50          04  A0  D0 0023C          MOVL    4(R0), R0
                                            FC  A0  9F 00240          PUSHAB  SAVE_CCB
                                                01  DD 00243          PUSHL   #1
                                                5E  DD 00245          PUSHL   SP
                                7E          04  AC  7D 00247          MOVQ    4(AP), -(SP)
                        0000V   CF              03  FB 0024B          CALLS   #3, HANDLER
                                                04 00250              RET
```

; Routine Size: 593 bytes,   Routine Base: _BAS$CODE + 01E5

```
;  757          1466  1 GLOBAL ROUTINE BAS$$WHOLE_VA_FETCH (                  ! Fetch routine
;  758          1467  1         DESCRIP,                                      ! The descriptor for this virtual array
;  759          1468  1         VALUE                                         ! Where to store array item
;  760          1469  1     ) : NOVALUE =
;  761          1470  1
;  762          1471  1 !++
;  763          1472  1 ! FUNCTIONAL DESCRIPTION:
;  764          1473  1 !
;  765          1474  1 !       Fetch all values from a virtual array.
;  766          1475  1 !
;  767          1476  1 ! FORMAL PARAMETERS:
;  768          1477  1 !
;  769          1478  1 !       DESCRIP.mz.r    The descriptor for the virtual array
;  770          1479  1 !       VALUE.wz.r      Address of the 1st location to store
;  771          1480  1 !                       the values.  The number of bytes
;  772          1481  1 !                       to store is in the LENGTH field of
;  773          1482  1 !                       DESCRIP.
;  774          1483  1 !
;  775          1484  1 ! IMPLICIT INPUTS:
;  776          1485  1 !
;  777          1486  1 !       NONE
;  778          1487  1 !
;  779          1488  1 ! IMPLICIT OUTPUTS:
;  780          1489  1 !
;  781          1490  1 !       NONE
;  782          1491  1 !
;  783          1492  1 ! ROUTINE VALUE:
;  784          1493  1 ! COMPLETION CODES:
;  785          1494  1 !
;  786          1495  1 !       NONE
;  787          1496  1 !
;  788          1497  1 ! SIDE EFFECTS:
;  789          1498  1 !
;  790          1499  1 !       Signals if an error is encountered.
;  791          1500  1 !
;  792          1501  1 !--
;  793          1502  1
;  794          1503  2     BEGIN
;  795          1504  2
;  796          1505  2     MAP
;  797          1506  2         DESCRIP : REF BLOCK [8, BYTE];
;  798          1507  2
;  799          1508  2     GLOBAL REGISTER
;  800          1509  2         CCB = K_CCB_REG : REF BLOCK [, BYTE];
;  801          1510  2
;  802          1511  2     LOCAL
;  803          1512  2         CHAN,                                         ! The channel this array is defined on
;  804          1513  2         HANDLE,                                       ! Pointer to info for this array
;  805          1514  2         GET_STATUS,                                   ! Last RMS GET status
;  806          1515  2         DEST,                                         ! Updated VALUE
;  807          1516  2         LEN,                                          ! Number of bytes to move
;  808          1517  2         REMAINING_BYTES,                              ! Number of bytes left to move
;  809          1518  2         SAVE_CCB : VOLATILE,                          ! CCB to POP, or zero.
;  810          1519  2         PUT_STATUS;                                   ! status retd by RMS $PUT
;  811          1520  2
;  812          1521  2 !+
;  813          1522  2 ! Establish a handler to pop the CCB when unwinding.
```

```
;  814          1523  2 !-
;  815          1524  2
;  816          1525  2     ENABLE
;  817          1526  2         HANDLER (SAVE_CCB);
;  818          1527  2
;  819          1528  2 !+
;  820          1529  2 ! DEST is initially the same as VALUE, but will be updated as blocks
;  821          1530  2 ! of 512 bytes are moved.
;  822          1531  2 !-
;  823          1532  2
;  824          1533  2     DEST = .VALUE;
;  825          1534  2
;  826          1535  2 !+
;  827          1536  2 ! Block #1 may not be entirely this array.  Initialize REMAINING_BYTES
;  828          1537  2 ! to 512 minus whatever offset there is.
;  829          1538  2 !-
;  830          1539  2
;  831          1540  2     IF (.DESCRIP [DSC$L_ARSIZE] + .DESCRIP [DSC$L_BYTEOFF]) LSS K_BLOCK_LENGTH
;  832          1541  2     THEN
;  833          1542  2         REMAINING_BYTES = .DESCRIP [DSC$L_ARSIZE]
;  834          1543  2     ELSE
;  835          1544  2         REMAINING_BYTES = K_BLOCK_LENGTH - .DESCRIP [DSC$L_BYTEOFF];
;  836          1545  2
;  837          1546  2 !+
;  838          1547  2 ! Fetch the array's channel number from its descriptor
;  839          1548  2 !-
;  840          1549  2     CHAN = .DESCRIP [DSC$L_LOGUNIT];
;  841          1550  2 !+
;  842          1551  2 ! Get a pointer to the LUB/ISB/RAB for this channel.  If the channel has not
;  843          1552  2 ! been opened yet, this call will allocate the LUB/ISB/RAB, but we will reject
;  844          1553  2 ! it for lack of the LUB$V_OPENED bit.
;  845          1554  2 !-
;  846          1555  2     BAS$$CB_PUSH (.CHAN, LUB$K_LUN_MIN);
;  847          1556  2     SAVE_CCB = .CCB;
;  848          1557  2
;  849          1558  2     IF ( NOT .CCB [LUB$V_OPENED]) THEN BAS$$STOP (BAS$K_VIRARROPE);
;  850          1559  2
;  851          1560  2 !+
;  852          1561  2 ! If the channel was not opened with organization VIRTUAL, reject it.  This
;  853          1562  2 ! also catches channel 0, which is always open but never has VIRTUAL
;  854          1563  2 ! organization.
;  855          1564  2 !-
;  856          1565  2
;  857          1566  2     IF (.CCB [LUB$B_ORGAN] NEQ LUB$K_ORG_VIRTU) THEN BAS$$STOP_IO (BAS$K_VIRARRDIS);
;  858          1567  2
;  859          1568  2 !+
;  860          1569  2 ! If this channel has been used for block I/O, reject it.
;  861          1570  2 !-
;  862          1571  2
;  863          1572  2     IF (.CCB [LUB$V_BLK_USE]) THEN BAS$$STOP_IO (BAS$K_ILLOPE);
;  864          1573  2
;  865          1574  2 !+
;  866          1575  2 ! If the record size declared for the file is not 512 bytes, reject it.
;  867          1576  2 !-
;  868          1577  2
;  869          1578  2     IF (.CCB [RAB$W_USZ] NEQ K_BLOCK_LENGTH) THEN BAS$$STOP_IO (BAS$K_VIRBUFTOO);
;  870          1579  2
```

```
;  871          1580  2 !+
;  872          1581  2 ! Mark the LUB as being used for a virtual array.
;  873          1582  2 !-
;  874          1583  2     CCB [LUB$V_VA_USE] = 1;
;  875          1584  2 !+
;  876          1585  2 ! Record access will always be by key
;  877          1586  2 !-
;  878          1587  2     CCB [RAB$B_RAC] = RAB$C_KEY;
;  879          1588  2 !+
;  880          1589  2 ! Mark the RAB so that a $GET to a non-existent record will still lock it.
;  881          1590  2 !-
;  882          1591  2     CCB [RAB$V_NXR] = 1;
;  883          1592  2 !+
;  884          1593  2 ! Set the address of our CLOSE appendage in the LUB.  If somebody else's
;  885          1594  2 ! is already there, we have a serious problem.
;  886          1595  2 !-
;  887          1596  2
;  888          1597  2     IF (.CCB [LUB$A_CLOSE] EQLA 0) THEN CCB [LUB$A_CLOSE] = BAS$$VA_CLOSE;
;  889          1598  2
;  890          1599  2     IF (.CCB [LUB$A_CLOSE] NEQA BAS$$VA_CLOSE) THEN BAS$$STOP_IO (BAS$K_PROLOSSOR);
;  891          1600  2
;  892          1601  2 !+
;  893          1602  2 ! If this is not the first reference to the file, we may have to write out
;  894          1603  2 ! the current buffer.
;  895          1604  2 !-
;  896          1605  2
;  897          1606  2     IF .CCB [LUB$L_LOG_RECNO] NEQ 0
;  898          1607  2     THEN
;  899          1608  3         BEGIN
;  900          1609  4         IF (.CCB [LUB$V_OUTBUF_DR])     ! buffer has been changed
;  901          1610  3         THEN
;  902          1611  4             BEGIN
;  903          1612  4             PUT_STATUS = $PUT (RAB = .CCB);
;  904          1613  4
;  905          1614  4             IF .PUT_STATUS EQL RMS$_CONTROLC
;  906          1615  4             THEN
;  907          1616  4                 BAS$$SIGNAL_CTRLC ();
;  908          1617  4
;  909          1618  5             IF (NOT .PUT_STATUS)
;  910          1619  4             THEN
;  911          1620  5                 BEGIN
;  912          1621  5                 WHILE (.PUT_STATUS EQL RMS$_RSA) DO
;  913          1622  6                     BEGIN
;  914          1623  6                     $WAIT (RAB = .CCB);
;  915          1624  6                     PUT_STATUS = $PUT (RAB = .CCB);
;  916          1625  6
;  917          1626  6                     IF .PUT_STATUS EQL RMS$_CONTROLC
;  918          1627  6                     THEN
;  919          1628  6                         BAS$$SIGNAL_CTRLC ();
;  920          1629  6
;  921          1630  5                     END;
;  922          1631  5                 IF (NOT .PUT_STATUS) THEN BAS$$STOP_IO (BAS$K_IOERR_REC);
;  923          1632  4                 END;
;  924          1633  4             CCB [LUB$V_OUTBUF_DR] = 0;
;  925          1634  3             END;
;  926          1635  2         END;
;  927          1636  2
```

```
;  928          1637  2 !+
;  929          1638  2 ! Calculate the number of blocks in the virtual array and loop through
;  930          1639  2 ! all of them.
;  931          1640  2 !-
;  932          1641  2
;  933          1642  2     INCR BLKCNT FROM 1 TO ((.DESCRIP [DSC$L_BYTEOFF] + .DESCRIP [DSC$L_ARSIZE])/512 + 1) DO
;  934          1643  3         BEGIN
;  935          1644  3         CCB [LUB$L_LOG_RECNO] = .BLKCNT;
;  936          1645  3         GET_STATUS = $GET (RAB = .CCB);
;  937          1646  3
;  938          1647  3         IF .GET_STATUS EQL RMS$_CONTROLC
;  939          1648  3         THEN
;  940          1649  3             BAS$$SIGNAL_CTRLC ();
;  941          1650  3
;  942          1651  3 !+
;  943          1652  3 ! If we get EOF, just clear the buffer.  This is compatable with
;  944          1653  3 ! the PDP-11.
;  945          1654  3 !-
;  946          1655  3
;  947          1656  4         IF ((.GET_STATUS EQL RMS$_EOF) OR (.GET_STATUS EQL RMS$_OK_RNF))
;  948          1657  3         THEN
;  949          1658  4             BEGIN
;  950          1659  4             CH$FILL (0, .CCB [RAB$W_USZ], .CCB [RAB$L_UBF]);
;  951          1660  4             CCB [RAB$L_RBF] = .CCB [RAB$L_UBF];
;  952          1661  4             CCB [RAB$W_RSZ] = .CCB [RAB$W_USZ];
;  953          1662  4             END
;  954          1663  3         ELSE
;  955          1664  4             BEGIN
;  956          1665  4
;  957          1666  5             IF ( NOT .GET_STATUS)
;  958          1667  4             THEN
;  959          1668  5                 BEGIN
;  960          1669  5 !+
;  961          1670  5 ! Again, worry about the RSA RMS error.
;  962          1671  5 !-
;  963          1672  5
;  964          1673  5                 WHILE (.GET_STATUS EQL RMS$_RSA) DO
;  965          1674  6                     BEGIN
;  966          1675  6                     $WAIT (RAB = .CCB);
;  967          1676  6                     GET_STATUS = $GET (RAB = .CCB);
;  968          1677  6
;  969          1678  6                     IF .GET_STATUS EQL RMS$_CONTROLC
;  970          1679  6                     THEN
;  971          1680  6                         BAS$$SIGNAL_CTRLC ();
;  972          1681  6
;  973          1682  5                     END;
;  974          1683  5
;  975          1684  5                 IF ( NOT .GET_STATUS) THEN BAS$$STOP_IO (BAS$K_IOERR_REC);
;  976          1685  5
;  977          1686  4                 END;
;  978          1687  4
;  979          1688  3             END;
;  980          1689  3
;  981          1690  3 !+
;  982          1691  3 ! Copy the 512 byte buffer to the desired location.  If this is the last
;  983          1692  3 ! buffer (or the first), there may not be 512 bytes so check for this.
;  984          1693  3 !-
```

```
;  985          1694  3
;  986          1695  3          LEN = MIN (K_BLOCK_LENGTH, .REMAINING_BYTES);
;  987          1696  5          CH$MOVE (.LEN, (.CCB [RAB$L_RBF] + (IF .BLKCNT EQL 1
;  988          1697  5                                              THEN .DESCRIP [DSC$L_BYTEOFF]
;  989          1698  3                                              ELSE 0)), .DEST);
;  990          1699  3          DEST = .DEST + .LEN;
;  991          1700  3          REMAINING_BYTES = .REMAINING_BYTES - .LEN;
;  992          1701  3
;  993          1702  2          END;                                    ! loop through all blocks in v.a.
;  994          1703  2
;  995          1704  2 !+
;  996          1705  2 ! Done with this I/O channel.
;  997          1706  2 !-
;  998          1707  2      BAS$$CB_POP ();
;  999          1708  1      END;                                    ! end of BAS$$WHOLE_VA_FETCH


                                    0FFC 00000          .ENTRY  BAS$$WHOLE_VA_FETCH, Save R2,R3,R4.R5,R6,-      ; 1466
                                                                R7,R8,R9,R10,R11
                              5E            08  C2 00002     SUBL2   #8, SP
                                        04  AE  D4 00005     CLRL    SAVE_CCB                                    ; 1503
                              6D      01EA  CF  DE 00008     MOVAL   25$, (FP)
                                        08  AC  DD 0000D     PUSHL   VALUE                                       ; 1533
                              57        04  AC  D0 00010     MOVL    DESCRIP, R7                                 ; 1540
                        50          0C  A7    F8  A7  C1 00014  ADDL3   -8(R7), 12(R7), R0
                            00000200  8F        50  D1 0001A  CMPL    R0, #512
                                            06  18 00021     BGEQ    1$
                              56        0C  A7  D0 00023     MOVL    12(R7), REMAINING_BYTES                     ; 1542
                                            09  11 00027     BRB     2$
                        56  00000200  8F        F8  A7  C3 00029  1$:  SUBL3   -8(R7), #512, REMAINING_BYTES      ; 1544
                              52        FC  A7  D0 00032  2$:  MOVL    -4(R7), CHAN                                ; 1549
                                            50  D4 00036     CLRL    R0                                          ; 1555
                                  00000000G 00  16 00038     JSB     BAS$$CB_PUSH
                                    08  AE          5B  D0 0003E     MOVL    CCB, SAVE_CCB                       ; 1556
                                        0B  FC  AB  E8 00042     BLBS    -4(CCB), 3$                             ; 1558
                                    7E      00G 8F  9A 00046     MOVZBL  #BAS$K_VIRARPOPE, -(SP)
                                00000000G   00      01  FB 0004A     CALLS   #1, BAS$$STOP
                                        05  C4  AB  91 00051  3$:  CMPB    -60(CCB), #5                            ; 1566
                                            0B  13 00055     BEQL    4$
                                    7E      00G 8F  9A 00057     MOVZBL  #BAS$K_VIRARRDIS, -(SP)
                                00000000G   00      01  FB 0005B     CALLS   #1, BAS$$STOP_IO
                          0B        FF  AB          01  E1 00062  4$:  BBC     #1, -1(CCB), 5$                     ; 1572
                                    7E      00G 8F  9A 00067     MOVZBL  #BAS$K_ILLOPE, -(SP)
                                00000000G   00      01  FB 0006B     CALLS   #1, BAS$$STOP_IO
                                    0200    8F      20  AB  B1 00072  5$:  CMPW    32(CCB), #512               ; 1578
                                            0B  13 00078     BEQL    6$
                                    7E      00G 8F  9A 0007A     MOVZBL  #BAS$K_VIRBUFTOO, -(SP)
                                00000000G   00      01  FB 0007E     CALLS   #1, BAS$$STOP_IO
                                        FF  AB          01  88 00085  6$:  BISB2   #1, -1(CCB)                   ; 1583
                                        1E  AB          01  90 00089     MOVB    #1, 30(CCB)                     ; 1587
                                        06  AB      80  8F  88 0008D     BISB2   #128, 6(CCB)                    ; 1591
                                                A4  AB  D5 00092     TSTL    -92(CCB)                            ; 1597
                                                    06  12 00095     BNEQ    7$
                                        A4  AB  0000V   CF  9E 00097     MOVAB   BAS$$VA_CLOSE, -92(CCB)
```

```
                        50      0000V   CF   9E 0009D 7$:     MOVAB   BAS$$VA_CLOSE, R0                       ; 1599
                        50        A4    AB   D1 000A2         CMPL    -92(CCB), R0                            ;
                                        0B   13 000A6         BEQL    8$                                      ;
                        7E        00G   8F   9A 000A8         MOVZBL  #BAS$K_PROLOSSOR, -(SP)                 ;
    00000000G           00              01   FB 000AC         CALLS   #1, BAS$$STOP_IO                        ;
                              E0        AB   D5 000B3 8$:     TSTL    -32(CCB)                                ; 1606
                                        65   13 000B6         BEQL    13$                                     ;
60          FE          AB              03   E1 000B8         BBC     #3, -2(CCB), 13$                        ; 1609
                                        5B   DD 000BD         PUSHL   CCB                                     ; 1612
    00000000G           00              01   FB 000BF         CALLS   #1, SYS$PUT                             ;
                        52              50   D0 000C6         MOVL    R0, PUT_STATUS                          ;
    00010651            8F              52   D1 000C9         CMPL    PUT_STATUS, #67153                      ; 1614
                                        07   12 000D0         BNEQ    9$                                      ;
    00000000G           00              00   FB 000D2         CALLS   #0, BAS$$SIGNAL_CTRLC                   ; 1616
                        3D              52   E8 000D9 9$:     BLBS    PUT_STATUS, 12$                         ; 1618
    000182DA            8F              52   D1 000DC 10$:    CMPL    PUT_STATUS, #99034                      ; 1621
                                        27   12 000E3         BNEQ    11$                                     ;
                                        5B   DD 000E5         PUSHL   CCB                                     ; 1623
    00000000G           00              01   FB 000E7         CALLS   #1, SYS$WAIT                            ;
                                        5B   DD 000EE         PUSHL   CCB                                     ; 1624
    00000000G           00              01   FB 000F0         CALLS   #1, SYS$PUT                             ;
                        52              50   D0 000F7         MOVL    R0, PUT_STATUS                          ;
    00010651            8F              52   D1 000FA         CMPL    PUT_STATUS, #67153                      ; 1626
                                        D9   12 00101         BNEQ    10$                                     ;
    00000000G           00              00   FB 00103         CALLS   #0, BAS$$SIGNAL_CTRLC                   ; 1628
                                        D0   11 0010A         BRB     10$                                     ; 1621
                        0A              52   E8 0010C 11$:    BLBS    PUT_STATUS, 12$                         ; 1631
                        7E              01   CE 0010F         MNEGL   #1, -(SP)                               ;
    00000000G           00              01   FB 00112         CALLS   #1, BAS$$STOP_IO                        ;
                  FE    AB              08   8A 00119 12$:    BICB2   #8, -2(CCB)                             ; 1633
50          F8    A7          0C    A7   C1 0011D 13$:    ADDL3   12(R7), -8(R7), R0                      ; 1642
                        50 00000200   8F   C6 00123         DIVL2   #512, R0                                ;
                  04    AE          01    A0 9E 0012A         MOVAB   1(R0), 4(SP)                            ;
                                        59   D4 0012F         CLRL    BLKCNT                                  ; 1696
                                  00B4  31   00131         BRW     24$                                     ;
                  E0    AB              59   D0 00134 14$:    MOVL    BLKCNT, -32(CCB)                        ; 1644
                                        5B   DD 00138         PUSHL   CCB                                     ; 1645
    00000000G           00              01   FB 0013A         CALLS   #1, SYS$GET                             ;
                        58              50   D0 00141         MOVL    R0, GET_STATUS                          ;
    00010651            8F              58   D1 00144         CMPL    GET_STATUS, #67153                      ; 1647
                                        07   12 0014B         BNEQ    15$                                     ;
    00000000G           00              00   FB 0014D         CALLS   #0, BAS$$SIGNAL_CTRLC                   ; 1649
    0001827A            8F              58   D1 00154 15$:    CMPL    GET_STATUS, #98938                      ; 1656
                                        09   13 0015B         BEQL    16$                                     ;
    00018049            8F              58   D1 0015D         CMPL    GET_STATUS, #98377                      ;
                                        14   12 00164         BNEQ    17$                                     ;
20  AB          00              6E          00   2C 00166 16$:    MOVC5   #0, (SP), #0, 32(CCB), @36(CCB)         ; 1659
                                  24    BB        0016C         PUSHR   #^M<R2,R3,R4,R5>                        ;
                  28    AB        24    AB   D0 0016E         MOVL    36(CCB), 40(CCB)                        ; 1660
                  22    AB        20    AB   B0 00173         MOVW    32(CCB), 34(CCB)                        ; 1661
                                        40   11 00178         BRB     20$                                     ; 1656
                        3D              58   E8 0017A 17$:    BLBS    GET_STATUS, 20$                         ; 1666
    000182DA            8F              58   D1 0017D 18$:    CMPL    GET_STATUS, #99034                      ; 1673
                                        27   12 00184         BNEQ    19$                                     ;
                                        5B   DD 00186         PUSHL   CCB                                     ; 1675
    00000000G           00              01   FB 00188         CALLS   #1, SYS$WAIT                            ;
                                        5B   DD 0018F         PUSHL   CCB                                     ; 1676
```

```
                00000000G  00              01  FB 00191          CALLS   #1, SYS$GET                                    ;
                           58              50  D0 00198          MOVL    R0, GET_STATUS                                 ;
                00010651   8F              58  D1 0019B          CMPL    GET_STATUS, #67153                             ; 1678
                                           D9  12 001A2          BNEQ    18$                                            ;
                00000000G  00              00  FB 001A4          CALLS   #0, BAS$$SIGNAL_CTRLC                          ; 1680
                                           D0  11 001AB          BRB     18$                                            ; 1673
                           0A              58  E8 001AD 19$:     BLBS    GET_STATUS, 20$                                ; 1684
                           7E              01  CE 001B0          MNEGL   #1, -(SP)                                      ;
                00000000G  00              01  FB 001B3          CALLS   #1, BAS$$STOP_IO                               ;
                           50              56  D0 001BA 20$:     MOVL    REMAINING_BYTES, R0                            ; 1695
                00000200   8F              50  D1 001BD          CMPL    R0, #512                                       ;
                                           05  15 001C4          BLEQ    21$                                            ;
                           50      0200    8F  3C 001C6          MOVZWL  #512, R0                                       ;
                           5A              50  D0 001CB 21$:     MOVL    R0, LEN                                        ;
                           01              59  D1 001CE          CMPL    BLKCNT, #1                                     ; 1696
                                           06  12 001D1          BNEQ    22$                                            ;
                           50       F8     A7  D0 001D3          MOVL    -8(R7), R0                                     ; 1697
                                           02  11 001D7          BRB     23$                                            ;
                                           50  D4 001D9 22$:     CLRL    R0                                             ; 1696
          00    BE        28 BB40           5A  28 001DB 23$:     MOVC3   LEN, @40(CCB)[R0], @DEST                       ; 1698
                           6E              5A  C0 001E2          ADDL2   LEN, DEST                                      ; 1699
                           56              5A  C2 001E5          SUBL2   LEN, REMAINING_BYTES                           ; 1700
      FF45          59     01        04    AF  F1 001E8 24$:     ACBL    4(SP), #1, BLKCNT, 14$                         ; 1642
                                00000000G  00  16 001EF          JSB     BAS$$CB_POP                                    ; 1707
                                               04 001F5          RET                                                    ; 1708
                                         0000  001F6 25$:     .WORD   Save nothing                                   ; 1503
                           50        08    AC  D0 001F8          MOVL    8(AP), R0                                      ;
                           50        04    A0  D0 001FC          MOVL    4(R0), R0                                      ;
                                     FC    A0  9F 00200          PUSHAB  SAVE_CCB                                       ;
                                           01  DD 00203          PUSHL   #1                                             ;
                                           5E  DD 00205          PUSHL   SP                                             ;
                           7E        04    AC  7D 00207          MOVQ    4(AP), -(SP)                                   ;
                     0000V CF              03  FB 0020B          CALLS   #3, HANDLER                                    ;
                                               04 00210          RET                                                    ;

; Routine Size:  529 bytes,    Routine Base:  _BAS$CODE + 0436

; 1000          1709  1
```

```
;  1002          1710  1 GLOBAL ROUTINE BAS$$WHOLE_VA_STORE (                  ! Store routine
;  1003          1711  1         DESCRIP,                                      ! The descriptor for this virtual array
;  1004          1712  1         VALUE                                         ! Where to find items for array
;  1005          1713  1     ) : NOVALUE =
;  1006          1714  1
;  1007          1715  1 !++
;  1008          1716  1 ! FUNCTIONAL DESCRIPTION:
;  1009          1717  1 !
;  1010          1718  1 !       Store a values in a virtual array.
;  1011          1719  1 !
;  1012          1720  1 ! FORMAL PARAMETERS:
;  1013          1721  1 !
;  1014          1722  1 !       DESCRIP.mz.r    The descriptor for the virtual array
;  1015          1723  1 !       VALUE.rz.r      The place from which to fetch the value.  The
;  1016          1724  1 !                       number of bytes to store is in the LENGTH field
;  1017          1725  1 !                       of DESCRIP.
;  1018          1726  1 !
;  1019          1727  1 ! IMPLICIT INPUTS:
;  1020          1728  1 !
;  1021          1729  1 !       NONE
;  1022          1730  1 !
;  1023          1731  1 ! IMPLICIT OUTPUTS:
;  1024          1732  1 !
;  1025          1733  1 !       NONE
;  1026          1734  1 !
;  1027          1735  1 ! ROUTINE VALUE:
;  1028          1736  1 ! COMPLETION CODES:
;  1029          1737  1 !
;  1030          1738  1 !       NONE
;  1031          1739  1 !
;  1032          1740  1 ! SIDE EFFECTS:
;  1033          1741  1 !
;  1034          1742  1 !       Signals if an error is encountered.
;  1035          1743  1 !
;  1036          1744  1 !--
;  1037          1745  1
;  1038          1746  2     BEGIN
;  1039          1747  2
;  1040          1748  2     MAP
;  1041          1749  2         DESCRIP : REF BLOCK [8, BYTE];
;  1042          1750  2
;  1043          1751  2     GLOBAL REGISTER
;  1044          1752  2         CCB = K_CCB_REG : REF BLOCK [, BYTE];
;  1045          1753  2
;  1046          1754  2     LOCAL
;  1047          1755  2         CHAN,                                         ! The channel this array is defined on
;  1048          1756  2         HANDLE,                                       ! Pointer to info for this array
;  1049          1757  2         GET_STATUS,                                   ! Last RMS GET status
;  1050          1758  2         PUT_STATUS,                                   ! Last RMS PUT status
;  1051          1759  2         SOURCE,                                       ! Address of value
;  1052          1760  2         LEN,                                          ! Length of values to move
;  1053          1761  2         REMAINING_BYTES,                              ! Number of bytes left to move
;  1054          1762  2         SAVE_CCB : VOLATILE;                          ! CCB to POP, or 0
;  1055          1763  2
;  1056          1764  2 !+
;  1057          1765  2 ! Establish a handler to pop the CCB on unwind.
;  1058          1766  2 !-
```

```
;  1059          1767  2
;  1060          1768  2      ENABLE
;  1061          1769  2          HANDLER (SAVE_CCB);
;  1062          1770  2
;  1063          1771  2  !+
;  1064          1772  2  ! SOURCE is initially VALUE until some values have been copied to the
;  1065          1773  2  ! virtual array.
;  1066          1774  2  !-
;  1067          1775  2
;  1068          1776  2      SOURCE = .VALUE;
;  1069          1777  2
;  1070          1778  2  !+
;  1071          1779  2  ! The first buffer may contain data from a previous array.  Subtract the
;  1072          1780  2  ! offset from the normal 512 bytes.
;  1073          1781  2  !-
;  1074          1782  2
;  1075          1783  2      IF (.DESCRIP [DSC$L_ARSIZE] + .DESCRIP [DSC$L_BYTEOFF]) LSS K_BLOCK_LENGTH
;  1076          1784  2      THEN
;  1077          1785  2          REMAINING_BYTES = .DESCRIP [DSC$L_ARSIZE]
;  1078          1786  2      ELSE
;  1079          1787  2          REMAINING_BYTES = K_BLOCK_LENGTH - .DESCRIP [DSC$L_BYTEOFF];
;  1080          1788  2
;  1081          1789  2  !+
;  1082          1790  2  ! Fetch the array's channel number from its descriptor
;  1083          1791  2  !-
;  1084          1792  2      CHAN = .DESCRIP [DSC$L_LOGUNIT];
;  1085          1793  2  !+
;  1086          1794  2  ! Get a pointer to the LUB/ISB/RAB for this channel.  If the channe' nas not
;  1087          1795  2  ! been opened yet, this call will allocate the LUB/ISB/RAB, but we will reject
;  1088          1796  2  ! it for lack of the LUB$V_OPENED bit.
;  1089          1797  2  !-
;  1090          1798  2      BAS$$CB_PUSH (.CHAN, LUB$K_LUN_MIN);
;  1091          1799  2      SAVE_CCB = .CCB;
;  1092          1800  2
;  1093          1801  2      IF ( NOT .CCB [LUB$V_OPENED]) THEN BAS$$STOP (BAS$K_VIRARROPE);
;  1094          1802  2
;  1095          1803  2  !+
;  1096          1804  2  ! If the channel was not opened with organization VIRTUAL, reject it.  This
;  1097          1805  2  ! also catches channel 0, which is always open but never has VIRTUAL
;  1098          1806  2  ! organization.
;  1099          1807  2  !-
;  1100          1808  2
;  1101          1809  2      IF (.CCB [LUB$B_ORGAN] NEQ LUB$K_ORG_VIRTU) THEN BAS$$STOP_IO (BAS$K_VIRARRDIS);
;  1102          1810  2
;  1103          1811  2  !+
;  1104          1812  2  ! If this channel has been used for block I/O, reject it.
;  1105          1813  2  !-
;  1106          1814  2
;  1107          1815  2      IF (.CCB [LUB$V_BLK_USE]) THEN BAS$$STOP_IO (BAS$K_ILLOPE);
;  1108          1816  2
;  1109          1817  2  !+
;  1110          1818  2  ! If the recordsize of the file is not 512 bytes, reject it.
;  1111          1819  2  !-
;  1112          1820  2
;  1113          1821  2      IF (.CCB [RAB$W_USZ] NEQ K_BLOCK_LENGTH) THEN BAS$$STOP_IO (BAS$K_VIRBUFTOO);
;  1114          1822  2
;  1115          1823  2  !+
```

```
;  1116          1824  2 ! If the file is marked read only, reject it.
;  1117          1825  2 !-
;  1118          1826  2
;  1119          1827  2     IF (.CCB [LUB$V_READ_ONLY]) THEN BAS$$STOP_IO (BAS$K_ILLILLACC);
;  1120          1828  2
;  1121          1829  2 !+
;  1122          1830  2 ! Mark the LUB as being used for a virtual array.
;  1123          1831  2 !-
;  1124          1832  2     CCB [LUB$V_VA_USE] = 1;
;  1125          1833  2 !+
;  1126          1834  2 ! Record access will always be by key
;  1127          1835  2 !-
;  1128          1836  2     CCB [RAB$B_RAC] = RAB$C_KEY;
;  1129          1837  2 !+
;  1130          1838  2 ! Set the RAB so that a $GET to a non-existent record will still lock
;  1131          1839  2 ! that record.
;  1132          1840  2 !-
;  1133          1841  2     CCB [RAB$V_NXR] = 1;
;  1134          1842  2 !+
;  1135          1843  2 ! Set the address of our CLOSE appendage in the LUB.  If somebody else's
;  1136          1844  2 ! is already there, we have a serious problem.
;  1137          1845  2 !-
;  1138          1846  2
;  1139          1847  2     IF (.CCB [LUB$A_CLOSE] EQLA 0) THEN CCB [LUB$A_CLOSE] = BAS$$VA_CLOSE;
;  1140          1848  2
;  1141          1849  2     IF (.CCB [LUB$A_CLOSE] NEQA BAS$$VA_CLOSE) THEN BAS$$STOP_IO (BAS$K_PROLOSSOR);
;  1142          1850  2
;  1143          1851  2 !+
;  1144          1852  2 ! If this is not the first reference to the file, we may have to write out
;  1145          1853  2 ! the current buffer.
;  1146          1854  2 !-
;  1147          1855  2
;  1148          1856  3     IF (.CCB [LUB$L_LOG_RECNO] NEQ 0)
;  1149          1857  2     THEN
;  1150          1858  3         BEGIN
;  1151          1859  4         IF (.CCB [LUB$V_OUTBUF_DR])     ! only write if buffer changed
;  1152          1860  3         THEN
;  1153          1861  4             BEGIN
;  1154          1862  4             PUT_STATUS = $PUT (RAB = .CCB);
;  1155          1863  4
;  1156          1864  4             IF .PUT_STATUS EQL RMS$_CONTROLC
;  1157          1865  4             THEN
;  1158          1866  4                 BAS$$SIGNAL_CTRLC ();
;  1159          1867  4
;  1160          1868  5             IF (NOT .PUT_STATUS)
;  1161          1869  4             THEN
;  1162          1870  5                 BEGIN
;  1163          1871  5                 WHILE (.PUT_STATUS EQL RMS$_RSA) DO
;  1164          1872  6                     BEGIN
;  1165          1873  6                     $WAIT (RAB = .CCB);
;  1166          1874  6                     PUT_STATUS = $PUT (RAB = .CCB);
;  1167          1875  6
;  1168          1876  6                     IF .PUT_STATUS EQL RMS$_CONTROLC
;  1169          1877  6                     THEN
;  1170          1878  6                         BAS$$SIGNAL_CTRLC ();
;  1171          1879  6
;  1172          1880  5                     END;
```

```
;  1173          1881  5                 IF (NOT .PUT_STATUS) THEN BAS$$STOP_IO (BAS$K_IOERR_REC);
;  1174          1882  4                 END;
;  1175          1883  4             CCB [LUB$V_OUTBUF_DR] = 0;
;  1176          1884  3             END;
;  1177          1885  2         END;
;  1178          1886  2
;  1179          1887  2 !+
;  1180          1888  2 ! Loop through all the values to be stored.  Moves are done in blocks of 512
;  1181          1889  2 ! bytes, except for possibly the last buffer which may not be full.
;  1182          1890  2 !-
;  1183          1891  2
;  1184          1892  2     INCR BLKCNT FROM 1 TO ((.DESCRIP [DSC$L_BYTEOFF] + .DESCRIP [DSC$L_ARSIZE])/512 + 1) DO
;  1185          1893  3         BEGIN
;  1186          1894  3         CCB [LUB$L_LOG_RECNO] = .BLKCNT;
;  1187          1895  3         GET_STATUS = $GET (RAB = .CCB);
;  1188          1896  3
;  1189          1897  3         IF .GET_STATUS EQL RMS$_CONTROLC
;  1190          1898  3         THEN
;  1191          1899  3             BAS$$SIGNAL_CTRLC ();
;  1192          1900  3
;  1193          1901  3 !+
;  1194          1902  3 ! If we are at EOF, extend the file.  This is compatable with the PDP-11.
;  1195          1903  3 !-
;  1196          1904  3
;  1197          1905  4         IF ((.GET_STATUS EQL RMS$_EOF) OR (.GET_STATUS EQL RMS$_OK_RNF))
;  1198          1906  3         THEN
;  1199          1907  4             BEGIN
;  1200          1908  4
;  1201          1909  4             LOCAL
;  1202          1910  4                 FAB_BLOCK : $FAB_DECL,
;  1203          1911  4                 EXTEND_STATUS;
;  1204          1912  4
;  1205          1913  4 !+
;  1206          1914  4 ! If the file is already allocated beyond the current end-of-file point
;  1207          1915  4 ! (which can happen on disk if the cluster size is greater than 1) then
;  1208          1916  4 ! do not do any allocation.
;  1209          1917  4 !-
;  1210          1918  4
;  1211          1919  5             IF (.CCB [LUB$L_LOG_RECNO] GTR .CCB [LUB$L_REC_MAX])
;  1212          1920  4             THEN
;  1213          1921  5                 BEGIN
;  1214          1922  5                 $FAB_INIT (FAB = FAB_BLOCK);
;  1215          1923  5                 FAB_BLOCK [FAB$L_ALQ] = .CCB [LUB$L_LOG_RECNO] - .CCB [LUB$L_REC_MAX];
;  1216          1924  5                 FAB_BLOCK [FAB$W_IFI] = .CCB [LUB$W_IFI];
;  1217          1925  5                 CCB [LUB$A_FAB] = FAB_BLOCK;
;  1218          1926  5                 CCB [RAB$L_STS] = SS$_NORMAL;
;  1219          1927  5                 EXTEND_STATUS = $EXTEND (FAB = FAB_BLOCK);
;  1220          1928  5
;  1221          1929  5                 IF ( NOT .EXTEND_STATUS) THEN BAS$$STOP_IO (BAS$K_IOERR_REC);
;  1222          1930  5
;  1223          1931  5                 CCB [LUB$L_REC_MAX] = .CCB [LUB$L_REC_MAX] + .FAB_BLOCK [FAB$L_ALQ];
;  1224          1932  5                 CCB [LUB$A_FAB] = 0;
;  1225          1933  4                 END;
;  1226          1934  4
;  1227          1935  4 !+
;  1228          1936  4 ! Since we did not really read a record, make sure the buffer contains
;  1229          1937  4 ! all zeros.
```

```
;  1230     1938  4 !-
;  1231     1939  4             CH$FILL (0, .CCB [RAB$W_USZ], .CCB [RAB$L_UBF]);
;  1232     1940  4             CCB [RAB$W_RSZ] = .CCB [RAB$W_USZ];
;  1233     1941  4             CCB [RAB$L_RBF] = .CCB [RAB$L_UBF];
;  1234     1942  4             END
;  1235     1943  3         ELSE
;  1236     1944  4             BEGIN
;  1237     1945  4
;  1238     1946  5             IF ( NOT .GET_STATUS)
;  1239     1947  4             THEN
;  1240     1948  5                 BEGIN
;  1241     1949  5 !+
;  1242     1950  5 ! Again, worry about the RSA RMS error.
;  1243     1951  5 !-
;  1244     1952  5
;  1245     1953  5                 WHILE (.GET_STATUS EQL RMS$_RSA) DO
;  1246     1954  6                     BEGIN
;  1247     1955  6                     $WAIT (RAB = .CCB);
;  1248     1956  6                     GET_STATUS = $GET (RAB = .CCB);
;  1249     1957  6
;  1250     1958  6                     IF .GET_STATUS EQL RMS$_CONTROLC
;  1251     1959  6                     THEN
;  1252     1960  6                         BAS$$SIGNAL_CTRLC ();
;  1253     1961  6
;  1254     1962  5                     END;
;  1255     1963  5
;  1256     1964  5                 IF ( NOT .GET_STATUS) THEN BAS$$STOP_IO (BAS$K_IOERR_REC);
;  1257     1965  5
;  1258     1966  4                 END;
;  1259     1967  4
;  1260     1968  3             END;
;  1261     1969  3
;  1262     1970  3 !+
;  1263     1971  3 ! Move data from the value address to the virtual array.
;  1264     1972  3 !-
;  1265     1973  3
;  1266     1974  3     LEN = MIN (K_BLOCK_LENGTH, .REMAINING_BYTES);
;  1267     1975  4     CH$MOVE (.LEN, .SOURCE, (.CCB [RAB$L_RBF] +
;  1268     1976  3         (IF .BLKCNT EQL 1 THEN .DESCRIP [DSC$L_BYTEOFF] ELSE 0)));
;  1269     1977  3
;  1270     1978  3 !+
;  1271     1979  3 ! Write out the buffer.
;  1272     1980  3 !-
;  1273     1981  3
;  1274     1982  3     PUT_STATUS = $PUT (RAB = .CCB);
;  1275     1983  3
;  1276     1984  3     IF .PUT_STATUS EQL RMS$_CONTROLC
;  1277     1985  3     THEN
;  1278     1986  3         BAS$$SIGNAL_CTRLC ();
;  1279     1987  3
;  1280     1988  3 !+
;  1281     1989  3 ! If the PUT fails, we must worry about the RSA error, which can happen if
;  1282     1990  3 ! we are running at AST level, and the AST interrupted some RMS I/O.  If
;  1283     1991  3 ! we get this error, wait for it to go away.  Any other RMS error is fatal.
;  1284     1992  3 !-
;  1285     1993  3
;  1286     1994  4     IF (NOT .PUT_STATUS)
```

```
: 1287          1995  3     THEN
: 1288          1996  4         BEGIN
: 1289          1997  4         WHILE (.PUT_STATUS EQL RMS$_RSA) DO
: 1290          1998  5             BEGIN
: 1291          1999  5             $WAIT (RAB = .CCB);
: 1292          2000  5             PUT_STATUS = $PUT (RAB = .CCB);
: 1293          2001  5
: 1294          2002  5             IF .PUT_STATUS EQL RMS$_CONTROLC
: 1295          2003  5             THEN
: 1296          2004  5                 BAS$$SIGNAL_CTRLC ();
: 1297          2005  5
: 1298          2006  4             END;
: 1299          2007  4
: 1300          2008  4         IF (NOT .PUT_STATUS) THEN BAS$$STOP_IO (BAS$K_IOERR_REC);
: 1301          2009  3         END;
: 1302          2010  3
: 1303          2011  3 !+
: 1304          2012  3 ! Update pointer and the number of bytes left to move to the array.
: 1305          2013  3 !-
: 1306          2014  3
: 1307          2015  3     SOURCE = .SOURCE + .LEN;
: 1308          2016  3     REMAINING_BYTES = .REMAINING_BYTES - .LEN;
: 1309          2017  3
: 1310          2018  2     END;                                    ! end of loop through values
: 1311          2019  2
: 1312          2020  2 !+
: 1313          2021  2 ! Done with this I/O channel.
: 1314          2022  2 !-
: 1315          2023  2     BAS$$CB_POP ();
: 1316          2024  1     END;                                    ! end of BAS$$WHOLE_VA_STORE


                                    0FFC 00000          .ENTRY  BAS$$WHOLE_VA_STORE, Save R2,R3,R4,R5,R6,-      : 1710
                                                                R7,R8,R9,R10,R11
                          5E      A4   AE  9E 00002     MOVAB   -92(SP), SP                                     :
                                  58   AE  D4 00006     CLRL    SAVE_CCB                                        : 1746
                          6D  02AC    CF  DE 00009     MOVAL   32$, (FP)                                       :
                                  08   AC  DD 0000E     PUSHL   VALUE                                           : 1776
                          57      04   AC  D0 00011     MOVL    DESCRIP, R7                                     : 1783
                  50      0C      A7   F8  A7 C1 00015 ADDL3   -8(R7), 12(R7), R0                              :
              00000200    8F           50  D1 0001B     CMPL    R0, #512                                        :
                                       06  18 00022     BGEQ    1$                                              :
                          56      0C   A7  D0 00024     MOVL    12(R7), REMAINING_BYTES                         : 1785
                                       09  11 00028     BRB     2$                                              :
              56 00000200    8F   F8  A7  C3 0002A 1$:  SUBL3   -8(R7), #512, REMAINING_BYTES                   : 1787
                          52      FC   A7  D0 00033 2$:  MOVL    -4(R7), CHAN                                    : 1792
                                       50  D4 00037     CLRL    R0                                              : 1798
                              00000000G    00  16 00039     JSB     BAS$$CB_PUSH                                    :
                      5C  AE               5B  D0 0003F     MOVL    CCB, SAVE_CCB                                   : 1799
                                  0B   FC  AB  E8 00043     BLBS    -4(CCB), 3$                                     : 1801
                          7E      00G  8F  9A 00047     MOVZBL  #BAS$K_VIRARROPE, -(SP)                         :
                  00000000G       00       01  FB 0004B     CALLS   #1, BAS$$STOP                                   :
                                  05   C4  AB  91 00052 3$:  CMPB    -60(CCB), #5                                    : 1809
                                           0B  13 00056     BEQL    4$                                              :
```

```
                7E          00G  8F  9A 00058          MOVZBL  #BAS$K_VIRARRDIS, -(SP)
    00000000G   00               01  FB 0005C          CALLS   #1, BAS$$STOP_IO
0B      FF      AB               01  E1 00063 4$:      BBC     #1, -1(CCB), 5$          ;  1815
                7E          00G  8F  9A 00068          MOVZBL  #BAS$K_ILLOPE, -(SP)
    00000000G   00               01  FB 0006C          CALLS   #1, BAS$$STOP_IO
      0200      8F          20   AB  B1 00073 5$:      CMPW    32(CCB), #512            ;  1821
                                 0B  13 00079          BEQL    6$
                7E          00G  8F  9A 0007B          MOVZBL  #BAS$K_VIRBUFTOO, -(SP)
    00000000G   00               01  FB 0007F          CALLS   #1, BAS$$STOP_IO
0B      FC      AB               02  E1 00086 6$:      BBC     #2, -4(CCB), 7$          ;  1827
                7E          00G  8F  9A 0008B          MOVZBL  #BAS$K_ILLILLACC, -(SP)
    00000000G   00               01  FB 0008F          CALLS   #1, BAS$$STOP_IO
        FF      AB               01  88 00096 7$:      BISB2   #1, -1(CCB)              ;  1832
        1E      AB               01  90 0009A          MOVB    #1, 30(CCB)              ;  1836
        06      AB          80   8F  88 0009E          BISB2   #128, 6(CCB)             ;  1841
                            A4   AB  D5 000A3          TSTL    -92(CCB)                 ;  1847
                                 06  12 000A6          BNEQ    8$
        A4      AB     0000V CF  9E 000A8          MOVAB   BAS$$VA_CLOSE, -92(CCB)
                50     0000V CF  9E 000AE 8$:      MOVAB   BAS$$VA_CLOSE, R0        ;  1849
                50          A4   AB  D1 000B3          CMPL    -92(CCB), R0
                                 0B  13 000B7          BEQL    9$
                7E          00G  8F  9A 000B9          MOVZBL  #BAS$K_PROLOSSOR, -(SP)
    00000000G   00               01  FB 000BD          CALLS   #1, BAS$$STOP_IO
                            E0   AB  D5 000C4 9$:      TSTL    -32(CCB)                 ;  1856
                                 65  13 000C7          BEQL    14$
60      FE      AB               03  E1 000C9          BBC     #3, -2(CCB), 14$         ;  1859
                                 5B  DD 000CE          PUSHL   CCB                      ;  1862
    00000000G   00               01  FB 000D0          CALLS   #1, SYS$PUT
                58               50  D0 000D7          MOVL    R0, PUT_STATUS
    00010651    8F               58  D1 000DA          CMPL    PUT_STATUS, #67153       ;  1864
                                 07  12 000E1          BNEQ    10$
    00000000G   00               00  FB 000E3          CALLS   #0, BAS$$SIGNAL_CTRLC    ;  1866
                3D               58  E8 000EA 10$:     BLBS    PUT_STATUS, 13$          ;  1868
    000182DA    8F               58  D1 000ED 11$:     CMPL    PUT_STATUS, #99034       ;  1871
                                 27  12 000F4          BNEQ    12$
                                 5B  DD 000F6          PUSHL   CCB                      ;  1873
    00000000G   00               01  FB 000F8          CALLS   #1, SYS$WAIT
                                 5B  DD 000FF          PUSHL   CCB                      ;  1874
    00000000G   00               01  FB 00101          CALLS   #1, SYS$PUT
                58               50  D0 00108          MOVL    R0, PUT_STATUS
    00010651    8F               58  D1 0010B          CMPL    PUT_STATUS, #67153       ;  1876
                                 D9  12 00112          BNEQ    11$
    00000000G   00               00  FB 00114          CALLS   #0, BAS$$SIGNAL_CTRLC    ;  1878
                                 D0  11 0011B          BRB     11$                      ;  1871
                0A               58  E8 0011D 12$:     BLBS    PUT_STATUS, 13$          ;  1881
                7E               01  CE 00120          MNEGL   #1, -(SP)
    00000000G   00               01  FB 00123          CALLS   #1, BAS$$STOP_IO
        FE      AB               08  8A 0012A 13$:     BICB2   #8, -2(CCB)              ;  1883
50      F8      A7          0C   A7  C1 0012E 14$:     ADDL3   12(R7), -8(R7), R0       ;  1892
                50 00000200 8F   C6  00134             DIVL2   #512, R0
        08      AE          01   A0  9E 00138          MOVAB   1(R0), 8(SP)
                                 5A  D4 00140          CLRL    BLKCNT                   ;  1975
                                0166 31 00142          BRW     31$
        E0      AB               5A  D0 00145 15$:     MOVL    BLKCNT, -32(CCB)         ;  1894
                                 5B  DD 00149          PUSHL   CCB                      ;  1895
    00000000G   00               01  FB 0014B          CALLS   #1, SYS$GET
                59               50  D0 00152          MOVL    R0, GET_STATUS
```

```
                    00010651   8F                 59  D1 00155          CMPL    GET_STATUS, #67153                          ; 1897
                                                  07  12 0015C          BNEQ    16$
                    00000000G  00                 00  FB 0015E          CALLS   #0, BAS$$SIGNAL_CTRLC                       ; 1899
                    0001827A   8F                 59  D1 00165 16$:     CMPL    GET_STATUS, #98938                          ; 1905
                                                  09  13 0016C          BEQL    17$
                    00018049   8F                 59  D1 0016E          CMPL    GET_STATUS, #98377
                                                  66  12 00175          BNEQ    20$
                         E4    AB          E0     AB  D1 00177 17$:     CMPL    -32(CCB), -28(CCB)                          ; 1919
                                                  4B  15 0017C          BLEQ    19$
    0050   8F                 00                 00  2C 0017E          MOVC5   #0, (SP), #0, #80, $RMS_PTR                 ; 1922
                                              0C  AE      00185
                         0C    AE        5003     8F  B0 00187          MOVW    #20483, $RMS_PTR
                         22    AE                 02  90 0018D          MOVB    #2, $RMS_PTR+22
                         2B    AE                 02  90 00191          MOVB    #2, $RMS_PTR+31
                  1C     AE    E0    AB     E4    AB  C3 00195          SUBL3   -28(CCB), -32(CCB), FAB_BLOCK+16            ; 1923
                         0E    AE    D0    AB     B0 0019C          MOVW    -48(CCB), FAB_BLOCK+2                       ; 1924
                         E8    AB    0C    AE     9E 001A1          MOVAB   FAB_BLOCK, -24(CCB)                         ; 1925
                         08    AB                 01  D0 001A6          MOVL    #1, 8(CCB)                                  ; 1926
                                              0C  AE  9F 001AA          PUSHAB  FAB_BLOCK                                   ; 1927
                    00000000G  00                 01  FB 001AD          CALLS   #1, SYS$EXTEND
                                                  50  E8 001B4          BLBS    EXTEND_STATUS, 18$                          ; 1929
                                                  01  CE 001B7          MNEGL   #1, -(SP)
                    00000000G  00                 01  FB 001BA          CALLS   #1, BAS$$STOP_IO
                         E4    AB          1C     AE  C0 001C1 18$:     ADDL2   FAB_BLOCK+16, -28(CCB)                      ; 1931
                                              E8  AB  D4 001C6          CLRL    -24(CCB)                                    ; 1932
      20   AB                 00                 00  2C 001C9 19$:     MOVC5   #0, (SP), #0, 32(CCB), @36(CCB)             ; 1939
                                              24  BB      001CF
                         22    AB          20     AB  B0 001D1          MOVW    32(CCB), 34(CCB)                            ; 1940
                         28    AB          24     AB  D0 001D6          MOVL    36(CCB), 40(CCB)                            ; 1941
                                                  40  11 001DB          BRB     23$                                         ; 1905
                                                  3D  59  E8 001DD 20$:     BLBS    GET_STATUS, 23$                         ; 1946
                    000182DA   8F                 59  D1 001E0 21$:     CMPL    GET_STATUS, #99034                          ; 1953
                                                  27  12 001E7          BNEQ    22$
                                                  5B  DD 001E9          PUSHL   CCB                                         ; 1955
                    00000000G  00                 01  FB 001EB          CALLS   #1, SYS$WAIT
                                                  5B  DD 001F2          PUSHL   CCB                                         ; 1956
                    00000000G  00                 01  FB 001F4          CALLS   #1, SYS$GET
                                                  59  50  D0 001FB          MOVL    R0, GET_STATUS
                    00010651   8F                 59  D1 001FE          CMPL    GET_STATUS, #67153                          ; 1958
                                                  D9  12 00205          BNEQ    21$
                    00000000G  00                 00  FB 00207          CALLS   #0, BAS$$SIGNAL_CTRLC                       ; 1960
                                                  D0  11 0020E          BRB     21$                                         ; 1953
                                                  0A  59  E8 00210 22$:     BLBS    GET_STATUS, 23$                         ; 1964
                                                  7E  01  CE 00213          MNEGL   #1, -(SP)
                    00000000G  00                 01  FB 00216          CALLS   #1, BAS$$STOP_IO
                                                  50  56  D0 0021D 23$:     MOVL    REMAINING_BYTES, R0                     ; 1974
                    00000200   8F                 50  D1 00220          CMPL    R0, #512
                                                  05  15 00227          BLEQ    24$
                                                  50      0200  8F  3C 00229          MOVZWL  #512, R0
                         04    AE                 50  D0 0022E 24$:     MOVL    R0, LEN
                                                  01  5A  D1 00232          CMPL    BLKCNT, #1                              ; 1976
                                                  06  12 00235          BNEQ    25$
                                                  50        F8  A7  D0 00237          MOVL    -8(R7), R0
                                                  02  11 0023B          BRB     26$
                                                  50  D4 0023D 25$:     CLRL    R0
28 BB40            00   BE          04     AE  28 0023F 26$:     MOVC3   LEN, @SOURCE, @40(CCB)[R0]                  ; 1975
                                                  5B  DD 00247          PUSHL   CCB                                         ; 1982
```

```
                00000000G  00              01  FB 00249          CALLS   #1, SYS$PUT                                    ;
                           58              50  D0 00250          MOVL    R0, PUT_STATUS                                 ;
                00010651   8F              58  D1 00253          CMPL    PUT_STATUS, #67153                             ; 1984
                                           07  12 0025A          BNEQ    27$                                            ;
                00000000G  00              00  FB 0025C          CALLS   #0, BAS$$SIGNAL_CTRLC                          ; 1986
                           3D              58  E8 00263 27$:     BLBS    PUT_STATUS, 30$                                ; 1994
                000182DA   8F              58  D1 00266 28$:     CMPL    PUT_STATUS, #99034                             ; 1997
                                           27  12 0026D          BNEQ    29$                                            ;
                                           5B  DD 0026F          PUSHL   CCB                                            ; 1999
                00000000G  00              01  FB 00271          CALLS   #1, SYS$WAIT                                   ;
                                           5B  DD 00278          PUSHL   CCB                                            ; 2000
                00000000G  00              01  FB 0027A          CALLS   #1, SYS$PUT                                    ;
                           58              50  D0 00281          MOVL    R0, PUT_STATUS                                 ;
                00010651   8F              58  D1 00284          CMPL    PUT_STATUS, #67153                             ; 2002
                                           D9  12 0028B          BNEQ    28$                                            ;
                00000000G  00              00  FB 0028D          CALLS   #0, BAS$$SIGNAL_CTRLC                          ; 2004
                                           D0  11 00294          BRB     28$                                            ; 1997
                           0A              58  E8 00296 29$:     BLBS    PUT_STATUS, 30$                                ; 2008
                           7E              01  CE 00299          MNEGL   #1, -(SP)                                      ;
                00000000G  00              01  FB 0029C          CALLS   #1, BAS$$STOP_IO                               ;
                           6E        04    AE  C0 002A3 30$:     ADDL2   LEN, SOURCE                                    ; 2015
                           56        04    AE  C2 002A7          SUBL2   LEN, REMAINING_BYTES                           ; 2016
        FE93    5A         01        08    AE  F1 002AB 31$:     ACBL    8(SP), #1, BLKCNT, 15$                         ; 1892
                      00000000G      00    16  002B2             JSB     BAS$$CB_POP                                    ; 2023
                                           04  002B8             RET                                                    ; 2024
                                     0000  002B9 32$:     .WORD   Save nothing                                           ; 1746
                           50        08    AC  D0 002BB          MOVL    8(AP), R0                                      ;
                           50        04    A0  D0 002BF          MOVL    4(R0), R0                                      ;
                                     FC    A0  9F 002C3          PUSHAB  SAVE_CCB                                       ;
                                           01  DD 002C6          PUSHL   #1                                             ;
                                           5E  DD 002C8          PUSHL   SP                                             ;
                                 7E        04  AC  7D 002CA      MOVQ    4(AP), -(SP)                                   ;
                           0000V CF              03  FB 002CE    CALLS   #3, HANDLER                                    ;
                                                 04  002D3      RET                                                    ;

; Routine Size:  724 bytes,    Routine Base:  _BAS$CODE + 0647

; 1317          2025  1
```

```
;  1319          2026  1 ROUTINE BAS$$VA_CLOSE                          ! Close a virtual array
;  1320          2027  1     : CALL_CCB NOVALUE =
;  1321          2028  1
;  1322          2029  1 !++
;  1323          2030  1 ! FUNCTIONAL DESCRIPTION:
;  1324          2031  1 !
;  1325          2032  1 !       Handle the closing of a virtual array.
;  1326          2033  1 !
;  1327          2034  1 ! FORMAL PARAMETERS:
;  1328          2035  1 !
;  1329          2036  1 !       NONE
;  1330          2037  1 !
;  1331          2038  1 ! IMPLICIT INPUTS:
;  1332          2039  1 !
;  1333          2040  1 !       NONE
;  1334          2041  1 !
;  1335          2042  1 ! IMPLICIT OUTPUTS:
;  1336          2043  1 !
;  1337          2044  1 !       NONE
;  1338          2045  1 !
;  1339          2046  1 ! ROUTINE VALUE:
;  1340          2047  1 ! COMPLETION CODES:
;  1341          2048  1 !
;  1342          2049  1 !       NONE
;  1343          2050  1 !
;  1344          2051  1 ! SIDE EFFECTS:
;  1345          2052  1 !
;  1346          2053  1 !       Writes out the last I/O buffer (if it has been modified).
;  1347          2054  1 !
;  1348          2055  1 !--
;  1349          2056  1
;  1350          2057  2     BEGIN
;  1351          2058  2
;  1352          2059  2     EXTERNAL REGISTER
;  1353          2060  2         CCB : REF BLOCK [, BYTE];
;  1354          2061  2
;  1355          2062  2     LOCAL
;  1356          2063  2         PUT_STATUS;                              ! Status of last RMS PUT
;  1357          2064  2
;  1358          2065  2 !+
;  1359          2066  2 ! Record access will always be by key.
;  1360          2067  2 !-
;  1361          2068  2     CCB [RAB$B_RAC] = RAB$C_KEY;
;  1362          2069  2
;  1363          2070  2 !+
;  1364          2071  2 ! If the buffer is dirty, write it out.
;  1365          2072  2 !-
;  1366          2073  2
;  1367          2074  3     IF (.CCB [LUB$V_OUTBUF_DR])
;  1368          2075  2     THEN
;  1369          2076  3         BEGIN
;  1370          2077  3         PUT_STATUS = $PUT (RAB = .CCB);
;  1371          2078  3
;  1372          2079  3         IF .PUT_STATUS EQL RMS$_CONTROLC
;  1373          2080  3         THEN
;  1374          2081  3             BAS$$SIGNAL_CTRLC ();
;  1375          2082  3
```

```
;  1376          2083  4          IF ( NOT .PUT_STATUS)
;  1377          2084  3          THEN
;  1378          2085  4              BEGIN
;  1379          2086  4      !+
;  1380          2087  4      ! Worry about RMS RSA error.
;  1381          2088  4      !-
;  1382          2089  4
;  1383          2090  4              WHILE (.PUT_STATUS EQL RMS$_RSA) DO
;  1384          2091  5                  BEGIN
;  1385          2092  5                  $WAIT (RAB = .CCB);
;  1386          2093  5                  PUT_STATUS = $PUT (RAB = .CCB);
;  1387          2094  5
;  1388          2095  5                  IF .PUT_STATUS EQL RMS$_CONTROLC
;  1389          2096  5                  THEN
;  1390          2097  5                      BAS$$SIGNAL_CTRLC ();
;  1391          2098  5
;  1392          2099  4                  END;
;  1393          2100  4
;  1394          2101  4              IF ( NOT .PUT_STATUS) THEN BAS$$STOP_IO (BAS$K_IOERR_REC);
;  1395          2102  4
;  1396          2103  3              END;
;  1397          2104  3
;  1398          2105  3          CCB [LUB$V_OUTBUF_DR] = 0;
;  1399          2106  2          END.
;  1400          2107  2
;  1401          2108  1      END;                                    ! end of BAS$$VA_CLOSE


                                001C 00000 BAS$$VA_CLOSE:
                                           .WORD   Save R2,R3,R4                    ; 2026
                     54 00000000G   00  9E 00002           MOVAB   SYS$PUT, R4
                     53 00000000G   00  9E 00009           MOVAB   BAS$$SIGNAL_CTRLC, R3
                   1E AB            01  90 00010           MOVB    #1, 30(CCB)      ; 2068
                50 FE AB            03  E1 00014           BBC     #3, -2(CCB), 5$  ; 2074
                                    5B  DD 00019           PUSHL   CCB              ; 2077
                     64             01  FB 0001B           CALLS   #1, SYS$PUT
                     52             50  D0 0001E           MOVL    R0, PUT_STATUS
                00010651    8F      52  D1 00021           CMPL    PUT_STATUS, #67153 ; 2079
                                    03  12 00028           BNEQ    1$
                     63             00  FB 0002A           CALLS   #0, BAS$$SIGNAL_CTRLC ; 2081
                     35             52  E8 0002D 1$:       BLBS    PUT_STATUS, 4$   ; 2083
                000182DA    8F      52  D1 00030 2$:       CMPL    PUT_STATUS, #99034 ; 2090
                                    1F  12 00037           BNEQ    3$
                                    5B  DD 00039           PUSHL   CCB              ; 2092
                00000000G   00      01  FB 0003B           CALLS   #1, SYS$WAIT
                                    5B  DD 00042           PUSHL   CCB              ; 2093
                     64             01  FB 00044           CALLS   #1, SYS$PUT
                     52             50  D0 00047           MOVL    R0, PUT_STATUS
                00010651    8F      52  D1 0004A           CMPL    PUT_STATUS, #67153 ; 2095
                                    DD  12 00051           BNEQ    2$
                     63             00  FB 00053           CALLS   #0, BAS$$SIGNAL_CTRLC ; 2097
                                    D8  11 00056           BRB     2$               ; 2090
                     0A             52  E8 00058 3$:       BLBS    PUT_STATUS, 4$   ; 2101
                     7E             01  CE 0005B           MNEGL   #1, -(SP)
```

```
                00000000G  00          01  FB 0005E          CALLS   #1, BAS$$STOP_IO                    ;
                      FE   AB          08  8A 00065 4$:      BICB2   #8, -2(CCB)                        ; 2105
                                           04 00069 5$:      RET                                        ; 2108

; Routine Size:  106 bytes,    Routine Base:  _BAS$CODE + 091B
```

```
;  1403          2109  1 ROUTINE HANDLER (                             ! POP CCB on UNWIND
;  1404          2110  1         SIG,                                  ! signal args
;  1405          2111  1         MECH,                                 ! mechanism args
;  1406          2112  1         ENBL                                  ! variables
;  1407          2113  1     ) =
;  1408          2114  1
;  1409          2115  1 !++
;  1410          2116  1 ! FUNCTIONAL DESCRIPTION:
;  1411          2117  1 !
;  1412          2118  1 !       POP the CCB if one of the routines which does a PUSH gets an
;  1413          2119  1 !       error.  This is handled here rather than in the BASIC handler
;  1414          2120  1 !       so that the address of the frame of the caller does not have
;  1415          2121  1 !       to be passed all the way down to this module.  Also, we wish
;  1416          2122  1 !       to mark that there is no buffer in memory.
;  1417          2123  1 !
;  1418          2124  1 ! FORMAL PARAMETERS:
;  1419          2125  1 !
;  1420          2126  1 !       SIG.rl.a        Address of the signal vector.  This contains
;  1421          2127  1 !                       the condition.
;  1422          2128  1 !       MECH.rl.a       Address of the mechanism vector.  This contains
;  1423          2129  1 !                       the status of the frame that signalled.
;  1424          2130  1 !       ENBL.rl.a       Address of the enable vector.  This contains
;  1425          2131  1 !                       a pointer to the CCB, or 0.
;  1426          2132  1 !
;  1427          2133  1 ! IMPLICIT INPUTS:
;  1428          2134  1 !
;  1429          2135  1 !       NONE
;  1430          2136  1 !
;  1431          2137  1 ! IMPLICIT OUTPUTS:
;  1432          2138  1 !
;  1433          2139  1 !       NONE
;  1434          2140  1 !
;  1435          2141  1 ! ROUTINE VALUE:
;  1436          2142  1 !
;  1437          2143  1 !       NONE
;  1438          2144  1 !
;  1439          2145  1 ! COMPLETION CODES:
;  1440          2146  1 !
;  1441          2147  1 !       Always SS$_RESIGNAL, but this is ingored when we are
;  1442          2148  1 !       unwinding.
;  1443          2149  1 !
;  1444          2150  1 ! SIDE EFFECTS:
;  1445          2151  1 !
;  1446          2152  1 !       Usually calls BAS$$CB_POP to complete the I/O.
;  1447          2153  1 !       Also marks the buffer empty.
;  1448          2154  1 !
;  1449          2155  1 !--
;  1450          2156  1
;  1451          2157  2     BEGIN
;  1452          2158  2
;  1453          2159  2     MAP
;  1454          2160  2         SIG : REF VECTOR,                       ! signal vector
;  1455          2161  2         MECH : REF VECTOR,                      ! mechanism vector
;  1456          2162  2         ENBL : REF VECTOR;                      ! enable vector
;  1457          2163  2
;  1458          2164  2     GLOBAL REGISTER
;  1459          2165  2         CCB = K_CCB_REG : REF BLOCK [, BYTE];
```

```
;  1460          2166  2
;  1461          2167  2      BIND
;  1462          2168  2          SAVE_CCB = .ENBL [1];
;  1463          2169  2
;  1464          2170  2  !+
;  1465          2171  2  ! If this is the unwind condition, restore the CCB.
;  1466          2172  2  ! Mark that there is no buffer in memory.
;  1467          2173  2  !-
;  1468          2174  2
;  1469          2175  3      IF ((LIB$MATCH_COND (SIG [1], %REF (SS$_UNWIND))) AND (.SAVE_CCB NEQA 0))
;  1470          2176  2      THEN
;  1471          2177  3          BEGIN
;  1472          2178  3          BAS$$CB_GET ();
;  1473          2179  3
;  1474          2180  4          IF (.CCB EQLA .SAVE_CCB)
;  1475          2181  3          THEN
;  1476          2182  4              BEGIN
;  1477          2183  4              CCB [LUB$L_LOG_RECNO] = 0;
;  1478          2184  4              CCB [LUB$V_OUTBUF_DR] = 0;
;  1479          2185  4              BAS$$CB_POP ();
;  1480          2186  4              RETURN (SS$_RESIGNAL);
;  1481          2187  4              END
;  1482          2188  3          ELSE
;  1483          2189  3
;  1484          2190  3              IF (.CCB NEQA 0) THEN LIB$STOP (OTS$_FATINTERR);
;  1485          2191  3
;  1486          2192  2          END;
;  1487          2193  2
;  1488          2194  2  !+
;  1489          2195  2  ! We do not recognize the signal, pass it without comment.
;  1490          2196  2  !-
;  1491          2197  2      RETURN (SS$_RESIGNAL);
;  1492          2198  1      END;                                    ! of HANDLER


                                        0804 00000 HANDLER:.WORD   Save R2,R11                    ;  2109
                          52      0C    AC   D0 00002          MOVL    ENBL, R2                   ;  2168
                          7E    0920    8F   3C 00006          MOVZWL  #2336, -(SP)               ;  2175
                                        5E   DD 0000B          PUSHL   SP                         ;
                    7E      04    AC              04   C1 0000D          ADDL3   #4, SIG, -(SP)     ;
                00000000G   00              02   FB 00012          CALLS   #2, LIB$MATCH_COND     ;
                            31              50   E9 00019          BLBC    R0, 2$                 ;
                                      04    B2   D5 0001C          TSTL    @4(R2)                 ;
                                            2C   13 0001F          BEQL    2$                     ;
                                00000000G   00   16 00021          JSB     BAS$$CB_GET            ;  2178
                          04    B2                  5B   D1 00027          CMPL    CCB, @4(R2)    ;  2180
                                                    0F   12 0002B          BNEQ    1$             ;
                                      E0    AB   D4 0002D          CLRL    -32(CCB)               ;  2183
                          FE    AB                  08   8A 00030          BICB2   #8, -2(CCB)    ;  2184
                                00000000G           00   16 00034          JSB     BAS$$CB_POP    ;  2185
                                                    11   11 0003A          BRB     2$             ;  2186
                                                    5B   D5 0003C 1$:      TSTL    CCB            ;  2190
                                                    0D   13 0003E          BEQL    2$             ;
                                00000000G           8F   DD 00040          PUSHL   #OTS$_FATINTERR ;
```

```
                    00000000G  00              01  FB 00046          CALLS   #1, LIB$STOP                                    ;
                               50      0918     8F  3C 0004D 2$:      MOVZWL  #2328, R0                                       ; 2197
                                                    04 00052          RET                                                     ; 2198

; Routine Size:  83 bytes,    Routine Base:  _BAS$CODE + 0985

; 1493          2199  1 END                                                 ! end of module BAS$VIRT_IO
; 1494          2200  1
; 1495          2201  0 ELUDOM
```

```
;                                  PSECT SUMMARY
;
;      Name                      Bytes                          Attributes
;
;  _BAS$CODE                       2520  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         70         0        581         00:01.1




;                                       COMMAND QUALIFIERS

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

; Size:          2520 code + 0 data bytes
; Run Time:         00:44.1
; Elapsed Time:     01:31.2
; Lines/CPU Min:     2991
; Lexemes/CPU-Min:  26594
; Memory Used:  251 pages
; Compilation Complete
```

BASUDFWL
LIS

BASUNLOCK
LIS

BASVECTOR
LIS

BASVIRTUA
LIS

BASVAL
LIS

BASVIRTIO
LIS

BASUNWIND
LIS

BASUPDATE
LIS

BASVECTR2
LIS