
SSSSSSSSSSS DDDDDDDDDDD AAAAAAAAA
SSSSSSSSSSS DDDDDDDDDDD AAAAAAAAA
SSSSSSSSSSS DDDDDDDDDDD AAAAAAAAA
SSS DDD DDD AAA AAA
SSS DDD DDD AAA AAA
SSS DDD DDD AAA AAA
SSS DDD DDD AAA AAA
SSS DDD DDD AAA AAA
SSS DDD DDD AAA AAA
SSSSSSSSS DDD DDD AAA AAA
SSSSSSSSS DDD DDD AAA AAA
SSSSSSSSS DDD DDD AAA AAA
SSS DDD DDD AAAAAAAAAAAAAAA
SSS DDD DDD AAAAAAAAAAAAAAA
SSS DDD DDD AAAAAAAAAAAAAAA
SSS DDD DDD AAA AAA
SSS DDD DDD AAA AAA
SSS DDD DDD AAA AAA
SSSSSSSSSSS DDDDDDDDDDD AAA AAA
SSSSSSSSSSS DDDDDDDDDDD AAA AAA
SSSSSSSSSSS DDDDDDDDDDD AAA AAA

```
RRRRRRRR    MM       MM    SSSSSSSS
RRRRRRRR    MM       MM    SSSSSSSS
RR      RR  MMMM   MMMM  SS
RR      RR  MMMM   MMMM  SS
RR      RR  MM  MM  MM  SS
RR      RR  MM  MM  MM  SS
RRRRRRRR    MM       MM    SSSSSS
RRRRRRRR    MM       MM    SSSSSS
RR  RR      MM       MM          SS
RR  RR      MM       MM          SS
RR    RR    MM       MM          SS      ....
RR    RR    MM       MM          SS      ....
RR      RR  MM       MM  SSSSSSSS        ....
RR      RR  MM       MM  SSSSSSSS        ....


LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

Table of contents

```
0000     1          .TITLE  RMS     Display RMS Data Structures
0000     2          .SBTTL  Copyright Notice
0000     3          .IDENT  'V04-000'
0000     4 ;
0000     5 ;*****************************************************************************
0000     6 ;*                                                                           *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9 ;*  ALL RIGHTS RESERVED.                                                     *
0000    10 ;*                                                                           *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    16 ;*  TRANSFERRED.                                                             *
0000    17 ;*                                                                           *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    19 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    20 ;*  CORPORATION.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24 ;*                                                                           *
0000    25 ;*                                                                           *
0000    26 ;*****************************************************************************
0000    27 ;
```

```
0000     29 ;++
0000     30 ;   Facility
0000     31 ;
0000     32 ;       System Dump Analyzer
0000     33 ;
0000     34 ;   Abstract
0000     35 ;
0000     36 ;       This module contains code to dump most internal (and some user)
0000     37 ;       RMS data structures.
0000     38 ;
0000     39 ;   Environment
0000     40 ;
0000     41 ;       Native mode, User mode
0000     42 ;
0000     43 ;   Author
0000     44 ;
0000     45 ;       K. E. Kinnear   September 1980 - July 1981
0000     46 ;
0000     47 ;   Modified by
0000     48 ;
0000     49 ;       V03-030 RAS0271         Ron Schaefer            14-Mar-1984
0000     50 ;               Yet more FWA changes.
0000     51 ;
0000     52 ;       V03-029 RASC234         Ron Schaefer            31-Jan-1984
0000     53 ;               More FWA changes for input sticky searchlists.
0000     54 ;
0000     55 ;       V03-028 RAS0230         Ron Schaefer             5-Jan-1984
0000     56 ;               Reflect significant FWA structure changes.
0000     57 ;
0000     58 ;       V03-027 SHZ0002         Stephen Zalewski        06-Dec-1983
0000     59 ;               Add flag definition to BLB flags.
0000     60 ;
0000     61 ;       V03-026 JWT0141         Jim Teague              14-Nov-1983
0000     62 ;               Change IFB$V_RUM to IFB$V_ONLY_RU
0000     63 ;
0000     64 ;       V03-025 KBT0571         Keith B. Thompson       24-Jul-1983
0000     65 ;               Track RMS changes
0000     66 ;
0000     67 ;       V03-024 SHZ0001         Stephen H. Zalewski     26-Jun-1983
0000     68 ;               Remove undefined symbol.
0000     69 ;
0000     70 ;       V03-023 KBT0550         Keith B. Thompson       24-Jun-1983
0000     71 ;               Fix KBT0548
0000     72 ;
0000     73 ;       V03-022 KBT0548         Keith B. Thompson       23-Jun-1983
0000     74 ;               More FWA changes
0000     75 ;
0000     76 ;       V03-021 KPL0002         Peter Lieberwirth        1-Jun-1983
0000     77 ;               Support fields added to IFAB and IRAB for journaling,
0000     78 ;               including the MJB (Miscellaneous Journaling Buffer).
0000     79 ;               Add some new GBD and GBH fields and one new IFAB field.
0000     80 ;
0000     81 ;       V03-020 KBT0534         Keith B. Thompson       1-Jun-1983
0000     82 ;               Fix some FWA flags
0000     83 ;
0000     84 ;       V03-019 KPL0001         Peter Lieberwirth       30-May-1983
0000     85 ;               Eliminate references due to obsolete RJB fields.
```

```
0000     86 ;
0000     87 ;       V03-018 KEK0030         K. E. Kinnear           15-Apr-1983
0000     88 ;               Fix LBN display in WCB to deal with relative
0000     89 ;               volume numbers in high part of LBN.
0000     90 ;
0000     91 ;       V03-017 DAS0003         David Solomon           11-Mar-1983
0000     92 ;               Display RLB$W_FLAGS2, RLB$B_TMO (new with timeout on
0000     93 ;               record lock). Formatting changes (add form feeds).
0000     94 ;
0000     95 ;       V03-016 RAS0123         Ron Schaefer             8-Feb-1983
0000     96 ;               Fix several display bugs and update to reflect MANY,
0000     97 ;               MANY RMS structure changes.
0000     98 ;
0000     99 ;       V03-015 TMK0004         Todd M. Katz            22-Jan-1983
0000    100 ;               Make a change to TMK0003.
0000    101 ;
0000    102 ;       V03-014 TMK0003         Todd M. Katz            21-Jan-1983
0000    103 ;               Add the definition of the bits IRB$V_RU_UPDATE and
0000    104 ;               IRB$V_NO_Q_WAIT to the IRAB bookkeeping bit table.
0000    105 ;
0000    106 ;       V03-013 TMK0002         Todd M. Katz            07-Jan-1983
0000    107 ;               Display IRB$L_OLDBUF and IRB$L_RECBUF. Add the displaying
0000    108 ;               of the bit field IFB$B_RECVRFLGS. Add the definition of the
0000    109 ;               bits IRB$V_RU_DELETE and IRB$V_RU_UNDEL to the IRAB bookkeeping
0000    110 ;               bit table.
0000    111 ;
0000    112 ;       V03-012 RAS0112         Ron Schaefer            22-Dec-1982
0000    113 ;               Change symbols:
0000    114 ;                       IFB$W_DEVBUFSIZ  ->  IFB$L_DEVBUFSIZ
0000    115 ;                       IFB$W_ASDEVBSIZ  ->  IFB$L_ASDEVBSIZ
0000    116 ;
0000    117 ;       V03-011 MCN0001         Maria del C. Nasr       29-Oct-1982
0000    118 ;               Eliminate COUNT_DUP, NORFA, and PRG_D_RFA from IDX
0000    119 ;               table to reflect changes made to RMSINTSTR.MDL.  Add
0000    120 ;               KEY_COMPR to IDX_CHR1.
0000    121 ;
0000    122 ;       V03-010 KBT0380         Keith B. Thompson       21-Oct-1982
0000    123 ;               Display new fields in asb and display all BIDs and BLNs
0000    124 ;               in decimal
0000    125 ;
0000    126 ;       V03-009 KBT0362         Keith B. Thompson       8-Oct-1982
0000    127 ;               Change asb$b_stksiz to a word
0000    128 ;
0000    129 ;       V03-008 KBT0320         Keith B. Thompson       8-Sep-1982
0000    130 ;               Make changes for new file sharing structures, and
0000    131 ;               change irb$b_srchflags to word and rlb$w_owner to long and
0000    132 ;               add ifb$v_stall_lock, irb$v_dup_key and irb$v_last_gt
0000    133 ;
0000    134 ;       V03-007 RAS0094         Ron Schaefer            27-Aug-1982
0000    135 ;               Delete symbol IRB$B_DIFF_CHAR.
0000    136 ;
0000    137 ;       V03-006 CWH0001         CW Hobbs                20-Aug-1982
0000    138 ;               Change symbol FWA$B_UNDERLINE to FWA$B_UNDER_DEV
0000    139 ;
0000    140 ;       V03-005 TMK0001         Todd M. Katz            18-Jul-1982
0000    141 ;               Add to the table SCH_CHR, a reference to the bit IRB$V_DEL_SEEN
0000    142 ;               and to the table IRBBKP_CHR a reference to the bit IRB$V_CON_EOF
```

```
0000   143 ;               and the bit IRB$V_UPDATE_IF.
0000   144 ;
0000   145 ;               Do not print the field IRB$L_NRP if the file is an indexed
0000   146 ;               file. The field IRB$B_NRP_KREF no longer exists so don't bother
0000   147 ;               printing it either.
0000   148 ;
0000   149 ;               Print out the fields IRB$L_NEXT_VBN, IRB$W_NEXT_ID,
0000   150 ;               IRB$L_PUTUP_VBN, IRB$W_PUTUP_ID, IRB$L_FIRST_VBN, and
0000   151 ;               IRB$W_FIRST_ID.
0000   152 ;
0000   153 ;               Print out IRB$L_CUR_VBN, IRB$W_CUR_ID, IRB$L_POS_VBN,
0000   154 ;               IRB$W_POS_ID, IRB$L_UDR_VBN, IRB$W_UDR_ID, IRB$L_SIDR_VBN,
0000   155 ;               IRB$W_SIDR_ID, IRB$W_CUR_COUNT and IRN$B_CUR_KREF.
0000   156 ;
0000   157 ;               Also do some juggling in the order in which things are printed
0000   158 ;               in both the IRAB and the IFAB.
0000   159 ;
0000   160 ;       V03-004 DWT0057         David Thiel             18-Jul-1982
0000   161 ;               Fix references to IRB$V_UPD_NRP, IFB$L_NRP_LIST,
0000   162 ;               IRB$B_NRP_KREF, IRB$L_NRP_PTR which were deleted
0000   163 ;               in RMSINTSTR V03-011.
0000   164 ;
0000   165 ;       V03-003 MLJ0084         Martin L. Jack          30-Mar-1982
0000   166 ;               Fix reference to IFB$B_KBUFSZ.
0000   167 ;
0000   168 ;       V03-002 KEK0027         K. E. Kinnear           1-Mar-1982
0000   169 ;               Add error messages for no RMS structures and premature
0000   170 ;               termination of the structure dump.  Also update IRAB,
0000   171 ;               GBD, and GBPB to RMSINTSTR V02-085.
0000   172 ;
0000   173 ;       V03-001 KEK0022         K. E. Kinnear           3-Feb-1982
0000   174 ;               Correct buffer reporting in GBD summary.
0000   175 ;
0000   176 ;       V3A Source Merge
0000   177 ;
0000   178 ;       V03-010 KEK0017         K. E. Kinnear           6-Jan-1982
0000   179 ;               Add FWA, GBH, and GBD summary dump routines.
0000   180 ;
0000   181 ;       V03-009 KEK0016         K. E. Kinnear           17-Dec-1981
0000   182 ;               Improve tracing code for global buffer debug.
0000   183 ;
0000   184 ;       V03-008 KEK0015         K. E. Kinnear           11-Dec-1981
0000   185 ;               Remove READAHEAD flags from cache and BLB flags.
0000   186 ;
0000   187 ;       V03-007 KEK0013         K. E. Kinnear           23-Oct-1981
0000   188 ;               New BLB flags, make AVLCL show for all orgs, and FRBPTR
0000   189 ;               for rel and indexed orgs.
0000   190 ;
0000   191 ;       V03-006 KEK0012         K. E. Kinnear           9-Sep-1981
0000   192 ;               Add trace support and initialization so that it
0000   193 ;               fails quietly on no-P1 space processes.
0000   194 ;
0000   195 ;       V03-005 KEK0011         K. E. Kinnear           2-Sep-1981
0000   196 ;               Fix references to $BCBDEF which has gone away.
0000   197 ;               Remove references to RLS$V_KEEP_ACC which has also
0000   198 ;               gone away.
0000   199 ;               Fix sequential problem in SHOW_BLB_SUM.
```

```
                                I 10

0000    200 ;
0000    201 ;       V03-004 KEK0010         K. E. Kinnear           26-Aug-1981
0000    202 ;               React to changes to internal RMS structures for
0000    203 ;               global buffering.  Support for BLB, and changes to
0000    204 ;               IFAB and IRAB.  Preliminary support for GBD, GBH, and
0000    205 ;               TRC structures.
0000    206 ;
0000    207 ;       V03-003 KEK0009         K. E. Kinnear           15-Aug-1981
0000    208 ;               Update for changes in IFB, IRB, IDX for prolog 3
0000    209 ;               files.
0000    210 ;
0000    211 ;       V03-002 KEK0005         K. E. Kinnear           1-Aug-1981
0000    212 ;               React to changing RMS structures.  RFA_ID in
0000    213 ;               IRAB went from a word to a byte.
0000    214 ;--
```

```
0000    216          .SBTTL  Declarations
0000    217 ;
0000    218 ;        Symbol Defintions
0000    219 ;
0000    220
0000    221          $CHFDEF                    ; Condition Handling Definitions
0000    222          $OPTDEF                    ; Parsing options
0000    223          $TPADEF                    ; TPARSE definitions
0000    224
0000    225 ;
0000    226 ;        RMS and SYSTEM definitions
0000    227 ;
0000    228
0000    229          $ASBDEF                                ; Asynchronous context block
0000    230          $BDBDEF                                ; Buffer descriptor block
0000    231          $BLBDEF                                ; Bucket lock block
0000    232          $CCBDEF                                ; Channel control block
0000    233          $CSHDEF                                ; Cache control bits
0000    234          $DCDEF                                 ; Device class and type defns
0000    235          $DDBDEF                                ; Device data block def
0000    236          $DEVDEF                                ; Device characteristic bits
0000    237          $DRCDEF                                ; Directory cache definitions
0000    238          $DYNDEF                                ; System control block idents
0000    239          $FABDEF                                ; File access block
0000    240          $FCBDEF                                ; File control block
0000    241          $FWADEF                                ; File string work area
0000    242          $GBDDEF                                ; Global buffer descriptor
0000    243          $GBHDEF                                ; Global buffer header
0000    244          $GBPBDEF                               ; Global buffer param block
0000    245          $IDXDEF                                ; Index Descriptor definitions
0000    246          $IFBDEF                                ; Internal FAB definitions
0000    247          $IMPDEF                                ; Process I/O control area
0000    248          $IRBDEF                                ; Internal RAB definitions
0000    249          $NAMDEF                                ; Name access block
0000    250          $PLGDEF                                ; Prologue (VBN 1) of indexed file
0000    251          $RABDEF                                ; Record access block
0000    252          $RJBDEF                                ; Record journaling block
0000    253          $RLBDEF                                ; Record lock block
0000    254          $RLSDEF                                ; Release bit definitions
0000    255          $RMSDEF                                ; RMS message definitions
0000    256          $SFSBDEF                               ; Shared file sync. block
0000    257          $TRCDEF                                ; Trace block definition
0000    258          $UCBDEF                                ; Unit control block defns
0000    259          $WCBDEF                                ; Window control block
0000    260          $XABDEF                                ; Extended access blocks
0000    261          $XABFHCDEF                             ; XAB - file header
0000    262          $XABALLDEF                             ; XAB - allocation
0000    263          $XABDATDEF                             ; XAB - date/time
0000    264          $XABRDTDEF                             ; XAB - revision date/time
0000    265          $XABPRODEF                             ; XAB - protection
0000    266          $XABTRMDEF                             ; XAB - terminal
0000    267          $XABSUMDEF                             ; XAB - indexed summary
0000    268          $XABKEYDEF                             ; XAB - indexed keys
0000    269          $MJBDEF                                ; Miscellaneous Journaling Block
0000    270
```

```
0000    272          .SBTTL  Macro Definitions
0000    273
0000    274 ;
0000    275 ;       Macro Definitions
0000    276 ;
0000    277
0000    278 ;
0000    279 ; GET -- Get a particular control block in memory.
0000    280 ;
0000    281
0000    282          .MACRO  GET     BLOCK,ADDR,ERRADDR,BUFFER,?LAB,?LAB1
0000    283
0000    284          .IF NB  ADDR
0000    285          MOVAL   ADDR,BLOCK-4
0000    286          BNEQ    LAB1
0000    287          BRW     ERRADDR
0000    288 LAB1:
0000    289          .ENDC
0000    290
0000    291          .IF NB  BUFFER
0000    292          PUSHAL  BUFFER
0000    293          .IFF
0000    294          PUSHAL  BLOCK
0000    295          .ENDC
0000    296
0000    297          CALLS   #1,GET_'BLOCK
0000    298          BLBS    R0,LAB
0000    299          CLRL    BLOCK-4
0000    300          BRW     ERRADDR
0000    301 LAB:
0000    302
0000    303          .IF NB  BUFFER
0000    304          MOVAB   BUFFER,R2
0000    305          .IFF
0000    306          MOVAB   BLOCK,R2
0000    307          .ENDC
0000    308
0000    309          .ENDM
0000    310
0000    311 ;
0000    312 ; SHOW -- Format a control block on the output file.
0000    313 ;
0000    314
0000    315          .MACRO  SHOW    BLOCK,BUFFER
0000    316
0000    317          .IF NB  BUFFER
0000    318          PUSHAL  BUFFER
0000    319          .IFF
0000    320          PUSHAL  BLOCK
0000    321          .ENDC
0000    322
0000    323          CALLS   #1,SHOW_'BLOCK
0000    324
0000    325          .ENDM
0000    326
0000    327 ;
0000    328 ; TABLE Macro which subtracts a number from symbol values for
```

```
0000    329 ; bits whose values are all over 32.
0000    330 ;
0000    331
0000    332         .MACRO  TABLEM  N,PREFIX,VALUES
0000    333         .IRP    VALUE,VALUES
0000    334         .SAVE
0000    335         .PSECT  __RMSLIT,EXE,NOWRT
0000    336 $$$     =       .
0000    337         .ASCIC  \VALUE\
0000    338         .RESTORE
0000    339         .LONG   'PREFIX''VALUE'-<N>
0000    340         .LONG   $$$
0000    341         .ENDR
0000    342         .LONG   -1,-1
0000    343         .ENDM   TABLEM
0000    344
0000    345 ;
0000    346 ; CASETBL macro, builds lists of information which
0000    347 ; is indexed by specific values.
0000    348 ;
0000    349
0000    350         .MACRO  CASETBL VALUES
0000    351         .IRP    VALUE,VALUES
0000    352         .SAVE
0000    353         .PSECT  __RMSLIT,EXE,NOWRT
0000    354 $$$     =       .
0000    355         .ASCIC  \VALUE\
0000    356         .RESTORE
0000    357         .LONG   $$$
0000    358         .ENDR
0000    359         .ENDM   CASETBL
0000    360
0000    361 ;
0000    362 ; Macro to push and adjust a descriptor for an !AF FAO parameter.
0000    363 ;
0000    364
0000    365         .MACRO  PUSHD2  VALUE,ADDRESS
0000    366         PUSHL   ADDRESS+4(R3)
0000    367         ADDL2   VALUE,(SP)
0000    368         PUSHL   ADDRESS(R3)
0000    369         .ENDM
0000    370
0000    371 ;
0000    372 ; Macro to print the two longwords of a descriptor.
0000    373 ;
0000    374
0000    375         .MACRO  PRINTD  BLOCK,NAME
0000    376         PUSHL   BLOCK''NAME'+4(R3)
0000    377         PUSHL   BLOCK''NAME'(R3)
0000    378         .IF LESS_THAN 6-%LENGTH(NAME)
0000    379         PRINT   2,<-
0000    380 NAME':  !XL              -
0000    381                 !XL>
0000    382         .IFF
0000    383         PRINT   2,<-
0000    384 NAME':          !XL             -
0000    385                 !XL>
```

```
0000    386          .ENDC
0000    387          .ENDM
0000    388
0000    389 ;
0000    390 ; Macro to print the two longwords of a descriptor and print out
0000    391 ; the string pointed to by the descriptor.
0000    392 ;
0000    393 ; MUST BE USED WITHIN LOCAL SYMBOL BLOCK!
0000    394 ;
0000    395 ; ONLY FOR USE IN SHOW_FWA!
0000    396 ;
0000    397 ; REQUIRES R5 TO BE OFFSET FOR FWA DESCRIPTORS!
0000    398 ;
0000    399
0000    400          .MACRO  DUMPDF  NAME,?X
0000    401          PRINTD  FWA$Q_NAME
0000    402          TSTL    FWA$Q_'NAME'(R3)
0000    403          BEQL    X
0000    404          MOVAL   FWA$Q_'NAME'(R3),R2
0000    405          CALLS   #0,FWADD
0000    406 X:
0000    407          .ENDM
0000    408
0000    409
0000    410 ;
0000    411 ; Macro to translate bits for fields at the beginning of the FWA
0000    412 ;
0000    413
0000    414          .MACRO  FWABITS TAG
0000    415          MOVL    #ALLOCSZ,(SP)
0000    416          MOVZBL  FWA$B_'TAG'FLGS(R3),-(SP)
0000    417          PUSHAB  TAG'_CHR
0000    418          CALLS   #2,TRANSLATE_BITS
0000    419          PUSHL   R4
0000    420          MOVZBL  FWA$B_'TAG'FLGS(R3),-(SP)
0000    421          PRINT   2,<-
0000    422 TAG'FLGS:         !XB     !AS>
0000    423          .ENDM
```

```
                     0000    425          .SBTTL  Storage Definitions
                     0000    426 ;
                     0000    427 ; Storage Definitions
                     0000    428 ;
                     0000    429
                 00000000    430          .PSECT  SDADATA,NOEXE,WRT
                     0000    431
                     0000    432 ;
                     0000    433 ; /DISPLAY Options
                     0000    434 ;
                     0000    435
                 FFFFFFFE    436 RMS_DIS_OPT::.LONG OPT$M_RMSALL          ; master options word
                     0000    437 RMS_IFI::.WORD  0                        ; contains IFI value (0=all)
                     0006    438
                     0006    439 ;
                     0006    440 ; "real" options for main routine: SHOW_RMS1
                     0006    441 ;
                     0006    442
                 00000000    443 RMS_DIS_OPT1:.LONG 0                     ; dynamic options word
                     0000    444 RMS_IFIT:.WORD  0                        ; dynamic ifi word
                     000C    445
                     000C    446 ;
                     000C    447 ; Temporary options work area
                     000C    448 ;
                     000C    449
                 00000000    450 RMS_DIS_TMP.:.LONG 0                     ; holds options from TPARSE !DISPLAY
                     0000    451 RMS_IFI_TMP::.WORD 0                     ; holds IFI value from TPARSE !DISPLAY
                 00000000    452 RMS_DIS_TMP1::.LONG 0                    ; work area for !DISPLAY
                     0016    453
                     0016    454 ;
                     0016    455 ; General Temporary Data Storage
                     0016    456 ;
                     0016    457
                     0000    458 IFI:    .WORD   0                        ; Current IFI on which we are working
                     0000    459 IFBS:   .WORD   0                        ; number of ifabs seen this run
                     001A    460
                     001A    461 ;
                     001A    462 ; Buffer areas into which are read the various control blocks from the dump.
                     001A    463 ;
                     001A    464 ;       All are in the format:
                     001A    465 ;
                     001A    466 ;               .LONG   0
                     001A    467 ;       LABEL:  .BLKB   length of structure
                     001A    468 ;
                     001A    469 ;       where the longword prior to the label contains the virtual address
                     001A    470 ;       of this control block in the actual dump, or 0 if block doesn't
                     001A    471 ;       exist or couldn't be read.
                     001A    472 ;
                     001A    473
                     001A    474          ASSUME IMP$C_NPIOFILES GE IMP$C_ENTPERSEG
                     001A    475
                 0000011A    476 IFABTBL:.BLKL   IMP$C_NPIOFILES+1
                 0000011E    477 IFBTBLSIZ:.BLKL 1
                 00000000    478          .LONG   0
                 000001DA    479 IFAB:   .BLKB   IFB$C_BLN
                 00000000    480          .LONG   0
                 000002A2    481 IRAB:   .BLKB   IRB$C_BLN_IDX
```

```
00000000  02A2   482          .LONG   0
000002F6  02A6   483 BDB:     .BLKB   BDB$C_BLN
00000000  02F6   484          .LONG   0
00000C36  02FA   485 FWA:     .BLKB   FWA$C_BLN
00000000  0C36   486          .LONG   0
00000C8A  0C3A   487 FAB:     .BLKB   FAB$C_BLN
00000000  0C8A   488          .LONG   0
00000CD2  0C8E   489 RAB:     .BLKB   RAB$C_BLN
00000000  0CD2   490          .LONG   0
00000D36  0CD6   491 NAM:     .BLKB   NAM$C_BLN
00000000  0D36   492          .LONG   0
00000D4A  0D3A   493 CCB:     .BLKB   CCB$C_LENGTH
00000000  0D4A   494          .LONG   0
00000F7E  0D4E   495 WCB:     .BLKB   WCB$C_LENGTH+512
          0F7E   496 WCBEND:
00000000  0F7E   497          .LONG   0
00001036  0F82   498 FCB:     .BLKB   FCB$C_LENGTH
00000000  1036   499          .LONG   0
00001086  103A   500 IDX:     .BLKB   IDX$C_FIXED_BLN+32
00000000  1086   501          .LONG   0
0000128A  108A   502 ASB:     .BLKB   ASB$C_BLN_IDX
00000000  128A   503          .LONG   0
0000131E  128E   504 UCB:     .BLKB   UCB$C_LENGTH
00000000  131E   505          .LONG   0
00001366  1322   506 DDB:     .BLKB   DDB$C_LENGTH
00000000  1366   507          .LONG   0
00001386  136A   508 RLB:     .BLKB   RLB$C_BLN
00000000  1386   509          .LONG   0
000013C2  138A   510 BLB:     .BLKB   BLB$C_BLN
00000000  13C2   511          .LONG   0
000013EE  13C6   512 GBD:     .BLKB   GBD$C_BLN
00000000  13EE   513          .LONG   0
0000144A  13F2   514 GBH:     .BLKB   GBH$C_BLN
00000000  144A   515          .LONG   0
0000145A  144E   516 RJB:     .BLKB   RJB$C_BLN
00000000  145A   517          .LONG   0
0000149E  145E   518 TRC:     .BLKB   TRC$C_BLN
00000000  149E   519          .LONG   0
000014C2  14A2   520 MJB:     .BLKB   MJB$C_BLN
          14C2   521
```

```
                        .SBTTL  Read-only Data Definitions
                ;
                ;       Read-only Data Definitions
                ;
00000000                .PSECT  RMS,EXE,NOWRT

DEV_CHR:TABLE   DEV$V_,<ALL,AVL,CCL,DIR,DMT,ELG,FOD,FOR,-
                        GEN,IDV,MBX,NET,ODV,RCK,REC,RND,RTM,SDI,-
                        SHR,SPL,SQD,SWL,TRM,WCK>

NAM_CHR:TABLE   NAM$V_,<EXP_DEV,EXP_DIR,EXP_NAME,EXP_TYPE,-
                        EXP_VER,GRP_MBR,HIGHVER,LOWVER,NODE,PPF,-
                        QUOTED,WILDCARD,WILD_DIR,WILD_GRP,WILD_MBR,-
                        WILD_NAME,WILD_SFD1,WILD_SFD2,WILD_SFD3,-
                        WILD_SFD4,WILD_SFD5,WILD_SFD6,WILD_SFD7,-
                        WILD_TYPE,WILD_UFD,WILD_VER>

FAC_CHR:TABLE   FAB$V_,<BIO,BRO,DEL,GET,PUT,TRN,UPD,EXE>

RAT_CHR:TABLE   FAB$V_,<BLK,CR,FTN,PRN>

BKP_CHR:TABLEM  32,IFB$V_,<BUSY,EOF,PPF_IMAGE,ASYNC,ASYNCWAIT,ACCESSED,-
                        ANSI_D,RWC,DMO,SPL,SCF,DLT,DFW,SQO,PPF_INPUT,NFS,-
                        WRTACC,MSE,CREATE,NORECLK,RW_ATTR,TMP,TEF,STALL_LOCK,-
                        SEQFIL,SEARCH,RMS_STALL,RESTART,FILEFOUND,DAP_OPEN,DAP,NSP>

RECFLG_CHR:TABLE IFB$V_,<RU_RECVR,AI_RECVR,BI_RECVR>

JNLFLG_CHR:TABLE IFB$V_,<ONLY_RU,RU,AI,AT,BI,NEVER_RU>

IRBBKP_CHR:TABLEM 32,IRB$V_,<BUSY,EOF,PPF_IMAGE,ASYNC,ASYNCWAIT,FIND_LAST,-
                        PUTS_LAST,BIO_LAST,BRO_SW,FIND,WRITE,RAHWBH,SKIP_NEXT,-
                        DUP,UNLOCK_RP,PPF_EOF,PPF_SKIP,PPF_FNDSV,IDX_ERR,RRV_ERR,-
                        UPDATE,UPDATE_IF,ABOVELCKD,GBLBUFF,CON_EOF,RMS_STALL,DAP_CONN,-
                        RU_DELETE,RU_UNDEL,RU_UPDATE,NO_Q_WAIT,PPF_ECHO,RESTART>

BDB_CHR:TABLE   BDB$V_,<AST_DCL,DRT,IOP,NOLOCATE,PRM,VAL,WFO>

ORG_TBL:CASETBL <Sequential,Relative,Indexed>

RFM_TBL:CASETBL <UDF,FIX,VAR,VFC,STM,STMLF,STMCR>

RFMORG_TBL:CASETBL <SEQ,REL,IDX>

SPL_CHR:TABLE   IRB$V_,<BKT_NO_LO,REC_W_LO,CONT_BKT,CONT_R,EMPTY_BKT,-
                        DUPS_SEEN,BRT_NO,BIG_SPLIT,SPL_IDX,EMPT_SEEN>

SCH_CHR:TABLE   IRB$V_,<POSINSERT,SRCHGT,POSDELETE,NEW_IDX,SRCHGE,-
                        NORLS_RNF,FIRST_TIM,PRM,DEL_SEEN,DUP_KEY,LAST_GT>

IDX_CHR1:TABLE  IDX$V_,<DUPKEYS,CHGKEYS,NULKEYS,INITIDX,IDX_COMPR,KEY_COMPR>

IDX_CHR0:TABLE  IDX$V_,<DUPKEYS,IDX_COMPR,INITIDX,KEY_COMPR,REC_COMPR>

IDX_TBL:CASETBL <STRING,SGNWORD,UNSGNWORD,SGNLONG,UNSGNLONG,PACKED,-
```

```
0614   580                  SGNQUAD,UNSGNQUAD>
0634   581
0634   582 IDX_TBL1:CASETBL <V2_BKT,CMPIDX,NCMPIDX,CMPCMP,CMPNCMP,NCMPCMP,NCMPNCMP>
0650   583
0650   584 CCB_CHR:TABLE    CCB$V_,<AMB>
0660   585
0660   586 AMODE_TBL:CASETBL <,Kernel,Executive,Supervisor,User,Invalid>
0678   587
0678   588 WCBSHR_CHR:TABLE WCB$V_,<READ,WRITE,NOTFCP,SHRWCB,OVERDRAWN>
06A8   589
06A8   590 WCBACC_CHR:TABLE WCB$V_,<NOWRITE,DLOCK,SPOOL,WRITECK,SEQONLY,WRITEAC,-
06A8   591                  READCK,NOREAD,NOTRUNC>
06F8   592
06F8   593 PRO_TBL:CASETBL <,R,W,RW,E,RE,WE,RWE,D,RD,WD,RWD,ED,RED,WED,RWED>
0738   594
0738   595 FCB_CHR:TABLE    FCB$V_,<DIR,MARKDEL,BADBLK,EXCL,SPOOL,RMSLOCK>
0770   596
0770   597 CSH_CHR:TABLE    CSH$V_,<LOCK,NOWAIT,NOREAD,NOBUFFER>
0798   598
0798   599 RLS_CHR:TABLE    RLS$V_,<RETURN,WRT_THRU,KEEP_LOCK,DEQ>
07C0   600
07C0   601 RLB_CHR:TABLE    RLB$V_,<WAIT,CR,PW,PR,CONV,LV2,FAKE,TMO>
0808   602
0808   603 RLB2_CHR:TABLE   RLB$V_,<TIMER_INPROG>    ; RLB$W_FLAGS2
0818   604
0818   605 OPT_CHR:TABLE    OPT$V_,<IFB,IRB,IDX,BDB,BDBSUM,ASB,CCB,WCB,FCB,FAB,-
0818   606                  RAB,RLB,NAM,XAB,BLB,GBD,GBH,TRC,BLBSUM,FWA,GBDSUM,RJB>
08D0   607
08D0   608 BLB_CHR:TABLE    BLB$V_,<LOCK,NOWAIT,NOREAD,NOBUFFER,IOLOCK,DFW,WRITEBACK>
0910   609
0910   610 BLB_TBL:CASETBL <NL,CR,CW,PR,PW,EX,??>
092C   611
092C   612 GBH_CHR:TABLE    GBH$V_,<CACHE_IN,CACHE_OUT,RLS_IN,RLS_OUT,QIO_START,-
092C   613                  QIO_DONE,STALL,THREADGO,BLB_ENQ,BLB_GRANT,BLB_DEQ,-
092C   614                  BLB_BLOCK,F1,F2,F3,F4>
09B4   615
09B4   616 GBD_CHR:TABLE    GBD$V_,<VALID>
09C4   617
09C4   618 PASS_CHR:TABLE   FWA$V_,<DUPOK,NOCOPY,SL_PASS,RLF_PASS,FNA_PASS,NAM_DVI,-
09C4   619                  EXP_NODE>
0A04   620
0A04   621 FLD_CHR:TABLEM   FWA$B_FLDFLGS*8,FWA$V_,<VERSION,TYPE,NAME,DIR,DEVICE>
0A34   622
0A34   623 WILD_CHR:TABLEM FWA$B_WILDFLGS*8,FWA$V_,<EXP_VER,EXP_TYPE,EXP_NAME,WC_VER,-
0A34   624                  WC_TYPE,WC_NAME,EXP_DIR,EXP_DEV>
0A7C   625
0A7C   626 PARSE_CHR:TABLEM FWA$B_PARSEFLGS*8,FWA$V_,<WILDCARD,NODE,QUOTED,GRPMBR,-
0A7C   627                  WILD_DIR>
0AAC   628
0AAC   629 DIR_CHR:TABLEM   FWA$B_DIRFLGS*8,FWA$V_,<DIR1,DIR2>
0AC4   630
0AC4   631 DIRWC_CHR:TABLEM FWA$B_DIRWCFLGS*8,FWA$V_,<WILD_UFD,WILD_SFD1>
0ADC   632
0ADC   633 LN_CHR:TABLEM    FWA$B_LNFLGS*8,FWA$V_,<LOGNAME,OBJTYPE,NETSTR,DEV_UNDER,-
0ADC   634                  REMRESULT,FILEFOUND,SYNTAX_CHK>
0B1C   635
0B1C   636 SL_CHR:TABLEM    FWA$B_SLFLGS*8,FWA$V_,<SLPRESENT,CONCEAL_DEV,ROOT_DIR,-
```

```
                0B1C   637                     DFLT_MFD,EXP_ROOT>
                0B4C   638
                0B4C   639 RJB_CHR:TABLE    RJB$V_,<RU,BI,AT,AI,OPEN>
                0B7C   640
                0B7C   641 MJB_CHR:TABLE    MJB$V_,<INIT,FORCE,FILE,SYNCH_SHARE>
                0BA4   642
                0BA4   643 JNLFLG2_CHR:     TABLE   IFB$V_,<VALID_AT,JNL,RUP,RU_RLK,DONE_ASS_JNL>
                0BD4   644
                0BD4   645 JNLFLG3_CHR:     TABLE   IRB$V_,<VALID_AT>
                0BE4   646
                0BE4   647 ;
                0BE4   648 ; Local Strings
                0BE4   649 ;
                0BE4   650
                0BE4   651 PIOSTRING:       STRING  <PIO$GW_IIOIMPA>
                0BFA   652 TRACE_DEPTH:     STRING  <RMS$TRACE_DEPTH>
                0C11   653 TRACE_FLAGS:     STRING  <RMS$TRACE_FLAGS>
                0C28   654 GSTRING:         STRING  <G>             ; GBPB signal for BDBSUM
                0C31   655 SPSTRING:        STRING  < >             ; opposite of above
                0C3A   656
                0C3A   657 ;
                0C3A   658 ; Constant Data Declarations
                0C3A   659 ;
                0C3A   660
00000087        0C3A   661 ALLOCSZ =        135             ; size of data buf alloc'ed on stack
```

```
                        0C3A   663          .SBTTL  SHOW_RMS1 -- Display RMS Data Structures
                        0C3A   664
                        0C3A   665 ;+++
                        0C3A   666 ; SHOW_RMS1 -- Display RMS Data Stuctures
                        0C3A   667 ;
                        0C3A   668 ; Inputs:
                        0C3A   669 ;
                        0C3A   670 ;       None.
                        0C3A   671 ;
                        0C3A   672 ; Outputs:
                        0C3A   673 ;
                        0C3A   674 ;       RMS Structures displayed.
                        0C3A   675 ;---
                        0C3A   676
                        0C3A   677          .ENABL  LSB
                        0C3A   678 SHOW_RMS1::
                  03FC  0C3A   679          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9>
6D  0000139F'EF   DE    0C3C   680          MOVAL   RMS_HANDLER,(FP)        ; Establish handler for SHOW RMS
56  0000001A'EF   9E    0C43   681          MOVAB   IFABTBL,R6              ; setup address for IFAB table
                        0C4A   682
                        0C4A   683 ;
                        0C4A   684 ; Look up PIO$GW_IIOIMPA is SDA symbol table, so it may be redefined
                        0C4A   685 ; for ITESTD.
                        0C4A   686 ;
                        0C4A   687
    00000018'EF   B4    0C4A   688          CLRW    IFBS                    ; no IFAB's found yet
          91 AF   DF    0C50   689          PUSHAL  PIOSTRING               ; argument for symbol_value
00000000'EF   01   FB   0C53   690          CALLS   #1,SYMBOL_VALUE
        59   51   D0    0C5A   691          MOVL    R1,R9                   ; put value (if any) in right register
           07 50   E8    0C5D   692          BLBS    R0,5$                   ; branch if ok
59  00000000'EF   D0    0C60   693          MOVL    PIO$GW_IIOIMPA,R9       ; get address of image i/o impure area
     57   20 A9   DE    0C67   694 5$:       MOVAL   IMP$W_ENTPERSEG(R9),R7  ; find table size
                        0C6B   695          GETMEM  (R7),IFBTBLSIZ          ; get value
           24 50   E9    0C7B   696          BLBC    R0,15$                  ; quit if none
        59   18   C0    0C7E   697          ADDL    #IMP$L_IFABTBL,R9       ; form pointer to table pointer
                        0C81   698          GETMEM  (R9),(R6)               ; get P1 space address
           12 50   E9    0C8D   699          BLBC    R0,15$                  ; if none, don't complain
                        0C90   700          SKIP    PAGE
    00000016'EF   B4    0C97   701          CLRW    IFI                     ; initialize IFI
                        0C9D   702
                        0C9D   703 ;
                        0C9D   704 ; Got a new table
                        0C9D   705 ;
                        0C9D   706
        59   66   D0    0C9D   707 10$:      MOVL    (R6),R9                 ; get next link to IFAB table
             1D   12    0CA0   708          BNEQ    18$
                        0CA2   709
    00000018'EF   B5    0CA2   710 15$:      TSTW    IFBS                    ; found any IFAB's?
             12   12    0CA8   711          BNEQ    17$                     ; yes, don't message
                        0CAA   712
                        0CAA   713 ;
                        0CAA   714 ; SIGNAL the fact that there are no RMS structures
                        0CAA   715 ;
                        0CAA   716
                        0CAA   717          SIGNAL  0,NOIMGRMS              ; no image RMS structures
                        0CBC   718
           0629   31    0CBC   719 17$:      BRW     900$
```

```
                57   0000011A'EF   3C   0CBF    720 18$:    MOVZWL  IFBTBLSIZ,R7            ; compute tablesize
                               57   D6   0CC6    721         INCL    R7                      ; + 1 for link
                           57   04   C4   0CC8    722         MULL2   #4,R7
                                     0CCB    723         GETMEM  (R9),(R6),R7            ; read in image IFAB table
                             C7 50   E9   0CD8    724         BLBC    R0,15$                  ; can't read in table
                57   0000011A'EF   3C   0CDB    725         MOVZWL  IFBTBLSIZ,R7            ; number of entries in table
                           53   56   D0   0CE2    726         MOVL    R6,R3                   ; set up address of first IFAB slot
                                83   D5   0CE5    727         TSTL    (R3)+                   ; space over link to next IFAB table
                                     0CE7    728 ;
                                     0CE7    729 ; Scan thru IFAB table -- 0 means no entry here
                                     0CE7    730 ;
                                     0CE7    731 ;       R3 = pointer into IFABTBL buffer at next IFAB address
                                     0CE7    732 ;       R6 = address of IFABTBL buffer in local memory
                                     0CE7    733 ;       R7 = number of entries in IFAB table
                                     0CE7    734 ;       R9 = virtual address of IFAB table
                                     0CE7    735 ;
                                     0CE7    736
                00000016'EF    B6   0CE7    737 20$:    INCW    IFI                     ; bump IFI value
                                     0CED    738         GET     IFAB,@(R3)+,800$        ; find IFAB if any, 080$ on error
                00000018'EF    B6   0D19    739         INCW    IFBS                    ; we found an IFAB, so count it
                                     0D1F    740 ;
                                     0D1F    741 ; If this is display for specific IFI, then check here, and go around
                                     0D1F    742 ; everything else!
                                     0D1F    743 ;
                                     0D1F    744
                0000000A'EF    B5   0D1F    745         TSTW    RMS_IFI1                        ; specific IFI?
                          10   13   0D25    746         BEQL    25$                     ; if eql no
00000016'EF     0000000A'EF    B1   0D27    747         CMPW    RMS_IFI1,IFI            ; this the one?
                         03   13   0D32    748         BEQL    25$                     ; if eql yes
                       05A7   31   0D34    749         BRW     800$                    ; just like its not there
                                     0D37    750
                                     0D37    751 ;
                                     0D37    752 ; SHOW IFAB
                                     0D37    753 ;
                                     0D37    754
                                     0D37    755 25$:    SHOW    IFAB
                                     0D44    756
                                     0D44    757 ;
                                     0D44    758 ; Get and Show ASB if any for this IFAB and if enabled.
                                     0D44    759 ;
                                     0D44    760
00000006'EF     00000040 8F    D3   0D44    761         BITL    #OPT$M_ASB,RMS_DIS_OPT1 ; ASB display enabled?
                            3D   13   0D4F    762         BEQL    30$                     ; if eql no
                                     0D51    763         GET     ASB,@IFB$L_ASBADDR+IFAB,30$
                                     0D81    764         SHOW    ASB
                                     0D8E    765
                                     0D8E    766 ;
                                     0D8E    767 ; Display index descriptors (IDX$) if indexed file.
                                     0D8E    768 ;
                                     0D8E    769
                02   00000145'EF    91   0D8E    770 30$:    CMPB    IFB$B_ORGCASE+IFAB,#IFB$C_IDX ;is file indexed?
                               75   12   0D95    771         BNEQ    40$                     ; if neq no -- skip index descriptors
                                     0D97    772
                                     0D97    773 ;
                                     0D97    774 ; Do we need the IDX descriptors at all?
                                     0D97    775 ;
                                     0D97    776
```

```
                00000006'EF    08   D3  0D97   777          BITL    #OPT$M_IDX,RMS_DIS_OPT1 ; display the IDX descriptors?
                               6C   13  0D9E   778          BEQL    40$                     ; if eql no
                                          0DA0   779
                                          0DA0   780          GET     IDX,@IFB$L_IDX_PTR(R2),40$
                               2D   11  0DCE   781          BRB     37$                     ; enter display loop in progress
                                          0DD0   782
                                          0DD0   783 35$:     GET     IDX,@IDX$L_IDXFL(R2),40$
                                          0DFD   784 37$:     SHOW    IDX
                               C4   11  0E0A   785          BRB     35$
                                          0E0C   786
                                          0E0C   787 ;
                                          0E0C   788 ; If it looks like the first BDB contains a FWA, see if we can dump it.
                                          0E0C   789 ;
                                          0E0C   790
          03 00000006'EF        14   E0  0E0C   791 40$:     BBS     #OPT$V_FWA,RMS_DIS_OPT1,41$ ; if we dont want FWA, skip it
                              007A   31  0E14   792          BRW     43$
                                          0E17   793 41$:     GET     FWA,@IFB$L_FWA_PTR+IFAB,43$     ; get FWA
                                          0E47   794          SHOW    FWA
                                          0E54   795          GET     FWA,@FWA$L_FWA_PTR+FWA,43$      ; get second FWA for $RENAME
                                          0E84   796          SHOW    FWA
                                          0E91   797
                                          0E91   798 ;
                                          0E91   799 ; Start on path to file id.  First find and dump CCB, then WCB, then FCB.
                                          0E91   800 ;
                                          0E91   801
    00000006'EF    00000380 8F   D3  0E91   802 43$:     BITL    #<OPT$M_CCB!OPT$M_WCB!OPT$M_FCB>,RMS_DIS_OPT1 ; any of these?
                                03   12  0E9C   803          BNEQ    44$
                              00A2   31  0E9E   804          BRW     46$
    00000D36'EF    00000142'EF    3C  0EA1   805 44$:     MOVZWL  IFB$W_CHNL+IFAB,CCB-4
                   00000D3A'EF    DF  0EAC   806          PUSHAL  CCB
            00003C2C'EF    01   FB  0EB2   807          CALLS   #1,GET_CCB
                          03 50   E8  0EB9   808          BLBS    R0,45$
                            0084   31  0EBC   809          BRW     46$
                                          0EBF   810
                                          0EBF   811 45$:     SHOW    CCB
                                          0ECC   812          GET     WCB,@CCB$L_WIND+CCB,46$
                                          0EFC   813          SHOW    WCB
                                          0F09   814          GET     FCB,@WCB$L_FCB(R2),46$
                                          0F36   815          SHOW    FCB
                                          0F43   816
                                          0F43   817 ;
                                          0F43   818 ; RJB - RMS Journaling Block
                                          0F43   819 ; IFB MJB (AI extend Journal buffer)
                                          0F43   820 ;
                                          0F43   821
                                          0F43   822 46$:
                                          0F43   823 ;
                                          0F43   824 ; If RJB display enabled, Get and Show the RJB (if any).
                                          0F43   825 ;
                                          0F43   826
    00000006'EF    00400000 8F   D3  0F43   827          BITL    #OPT$M_RJB,RMS_DIS_OPT1
                                3D   13  0F4E   828          BEQL    47$
                                          0F50   829          GET     RJB,@IFB$L_RJB+IFAB,47$
                                          0F80   830          SHOW    RJB
                                          0F8D   831
                                          0F8D   832 ;
                                          0F8D   833 ; If MJB display enabled, Get and Show the MJB (if any).
```

```
                                 0F8D   834 ;
                                 0F8D   835
                                 0F8D   836 47$:
                                 0F8D   837          GET     MJB,@IFB$L_EXTJNLBUF+IFAB,48$
                                 0FBD   838          SHOW    MJB
                                 0FCA   839
                                 0FCA   840
                                 0FCA   841 ;
                                 0FCA   842 ; Scan thru IRAB chain and show individual IRAB's
                                 0FCA   843 ;
                                 0FCA   844
00000006'EF   00004044 8F   D3   0FCA   845 48$:     BITL    #<OPT$M_IRB!OPT$M_ASB!OPT$M_RLB>,RMS_DIS_OPT1
                       03   12   0FD5   846          BNEQ    49$
                     0136   31   0FD7   847          BRW     60$
          52   000001DE'EF   DE   0FDA   848 49$:     MOVAL   IRAB,R2
        1C A2  0000013E'EF   D0   0FE1   849          MOVL    IFB$L_IRAB_LNK+IFAB,IRB$L_IRAB_LNK(R2) ; set initial addr IRAB
                             0FE9   850 50$:     GET     IRAB,@IRB$L_IRAB_LNK+IRAB,60$
                             1019   851          SHOW    IRAB
                             1026   852
                             1026   853 ;
                             1026   854 ; Get and Show the IRB MJB (if any).
                             1026   855 ;
                             1026   856
                             1026   857          GET     MJB,@IRB$L_ATJNLBUF+IRAB,51$
                             1056   858          SHOW    MJB
                             1063   859
                             1063   860 ;
                             1063   861 ; If ASB display enabled, Get and Show the ASB (if any).
                             1063   862 ;
                             1063   863
00000006'EF   00000040 8F   D3   1063   864 51$:     BITL    #OPT$M_ASB,RMS_DIS_OPT1
                       3D   13   106E   865          BEQL    52$
                             1070   866          GET     ASB,@IRB$L_ASBADDR+IRAB,50$
                             10A0   867          SHOW    ASB
                             10AD   868
                             10AD   869 ;
                             10AD   870 ; Display all RLB's attached to this IRAB
                             10AD   871 ;
                             10AD   872
00000006'EF   00004000 8F   D3   10AD   873 52$:     BITL    #OPT$M_RLB,RMS_DIS_OPT1 ; is it selected?
                       03   12   10B8   874          BNEQ    55$                     ; if neq yes
                     FF2C   31   10BA   875 54$:     BRW     50$
0000136A'EF   00000216'EF   D0   10BD   876 55$:     MOVL    IRB$L_RLB_LNK+IRAB,RLB$L_LNK+RLB ; set up  pointer to 1st rlb
                       F0   13   10C8   877          BEQL    54$
                             10CA   878          SKIP    PAGE
                             10D1   879 57$:     GET     RLB,@RLB$L_LNK+RLB,50$
                             1101   880          SHOW    RLB
                       C1   11   110E   881          BRB     57$
                             1110   882
                             1110   883 ;
                             1110   884 ; Scan thru BDB list and show information from BDB's
                             1110   885 ;
                             1110   886
      00002CC4'EF      00   FB   1110   887 60$:     CALLS   #0,SHOW_BDB_SUM         ; show summary of BDB's if selected
      00002DF4'EF      00   FB   1117   888          CALLS   #0,SHOW_BLB_SUM         ; show summary of BLB's if selected
      00000006'EF      10   D3   111E   889          BITL    #OPT$M_BDB,RMS_DIS_OPT1
                       73   13   1125   890          BEQL    70$
```

```
                52   0000011E'EF    D0  1127   891          MOVL    IFAB-4,R2                       ; get virt addr of IFAB
                52   00000040 8F    C0  112E   892          ADDL2   #IFB$L_BDB_FLNK,R2              ; form virt addr of queue header
          00000162'EF      52        D1  1135   893          CMPL    R2,IFB$L_BDB_FLNK+IFAB          ; does queue header point to itself?
                             5C      13  113C   894          BEQL    70$                             ; if eql yes -- queue empty
    000002A6'EF    00000162'EF        D0  113E   895          MOVL    IFB$L_BDB_FLNK+IFAB,BDB         ; initialize first BDB
                                           1149   896          SKIP    PAGE
                                           1150   897 65$:     GET     BDB,@BDB$L_FLINK+BDB,70$
                                           1180   898          SHOW    BDB
    000002A2'EF    00000166'EF        D1  118D   899          CMPL    IFB$L_BDB_BLNK+IFAB,BDB-4       ; is this the last BDB?
                             B6      12  1198   900          BNEQ    65$                             ; if neq no
                                           119A   901
                                           119A   902 ;
                                           119A   903 ; Scan through the BLB list and print out contents
                                           119A   904 ;
                                           119A   905
          09 00000122'EF     33      E0  119A   906 70$:     BBS     #IFB$V_NORECLK,IFAB,71$         ; if this doesn't have BLB's
             00   00000145'EF        91  11A2   907          CMPB    IFB$B_ORGCASE+IFAB,#IFB$C_SEQ   ; is this seq?
                             03      12  11A9   908          BNEQ    72$
                           0080      31  11AB   909 71$:     BRW     80$
    00000006'EF    00008000 8F        D3  11AE   910 72$:     BITL    #OPT$M_BLB,RMS_DIS_OPT1         ; should we do BLB's at all?
                             73      13  11B9   911          BEQL    80$                             ; if eql no, exit
                52   0000011E'EF    D0  11BB   912          MOVL    IFAB-4,R2                       ; address of ifab
                52   00000098 8F    C0  11C2   913          ADDL2   #IFB$L_BLBFLNK,R2               ; "real" address of blbflnk
          000001BA'EF      52        D1  11C9   914          CMPL    R2,IFB$L_BLBFLNK+IFAB           ; does queue point to itself
                             5C      13  11D0   915          BEQL    80$                             ; if eql yes -- no blb's
    0000138A'EF    000001BA'EF        D0  11D2   916          MOVL    IFB$L_BLBFLNK+IFAB,BLB          ; put blb address in BLB
                                           11DD   917          SKIP    PAGE
                                           11E4   918 73$:     GET     BLB,@BLB$L_FLNK+BLB,80$
                                           1214   919          SHOW    BLB
    00001386'EF    000001BE'EF        D1  1221   920          CMPL    IFB$L_BLBBLNK+IFAB,BLB-4        ; is this last blb?
                             B6      12  122C   921          BNEQ    73$                             ; if neq no, go for more
                                           122E   922
                                           122E   923 ;
                                           122E   924 ; Try to dump GBH if any.  Get the GBH in SDA memory if anyone down
                                           122E   925 ; stream needs it, even if we don't want to dump it explicitly.
                                           122E   926 ;
                                           122E   927
                002E0000 8F        D3  122E   928 80$:     BITL    #<OPT$M_GBH!OPT$M_GBD!OPT$M_GBDSUM!OPT$M_TRC>,-
                00000006'EF                1234   929                  RMS_DIS_OPT1
                         03        12  1239   930          BNEQ    82$                             ; need GBH
                       00A0        31  123B   931 81$:     BRW     800$                            ; don't nee  GBH
                000001AA'EF        D5  123E   932 82$:     TSTL    IFB$L_GBH_PTR+IFAB              ; do we ha   one?
                         F5        13  1244   933          BEQL    81$                             ; if eql no, skip all who need it
                                           1246   934          GET     GBH,@IFB$L_GBH_PTR+IFAB,800$
          0D 00000006'EF     12      E1  1276   935          BBC     #OPT$V_GBH,RMS_DIS_OPT1,90$     ; we don't dump the GBH
                                           127E   936          SHOW    GBH
                                           128B   937
             00003744'EF     00      FB  128B   938 90$:     CALLS   #0,SHOW_GBD_SUM
                                           1292   939
                                           1292   940 ;
                                           1292   941 ; If tracing enabled, then get values of rms$trace_depth and rms$trace_flags
                                           1292   942 ; and pass them on to SHOW_TRACE.
                                           1292   943 ;
                                           1292   944
          44 00000006'EF     13      E1  1292   945 100$:    BBC     #OPT$V_TRC,RMS_DIS_OPT1,800$    ; if not enabled
                             52      D4  129A   946          CLRL    R2                              ; initialize trace depth
                      F95A   CF      DF  129C   947          PUSHAL  TRACE_DEPTH                     ; get set to look up symbol
```

```
                00000000'EF   01    FB   12A0    948          CALLS   #1,SYMBOL_VALUE                  ; look up value of RMS$TRACE_DEPTH
                        03 50 E9    12A7    949          BLBC    R0,102$                         ; if lbc then no symbol, use default
                     52   51    D0   12AA    950          MOVL    R1,R2                           ; use value of RMS$TRACE_DEPTH
                          52    DD   12AD    951 102$:    PUSHL   R2                              ; put on stack for call to SHOW_TRACE
                52   FFFFFFFF 8F   D0   12AF    952          MOVL    #-1,R2                          ; default for trace flags
                        F957 CF   DF   12B6    953          PUSHAL  TRACE_FLAGS                     ; set up for look up
                00000000'EF   01    FB   12BA    954          CALLS   #1,SYMBOL_VALUE                 ; look up value of RMS$TRACE_FLAGS
                        03 50 E9    12C1    955          BLBC    R0,104$                         ; if lbc then no symbol, use default
                     52   51    CA   12C4    956          BICL    R1,R2                           ; use value of RMS$TRACE_FLAGS as list
                                        12C7    957                                              ;  of things NOT to trace.
                          52    DD   12C7    958 104$:    PUSHL   R2                              ; push on stack for  call to SHOW_TRACE
             7E 000013EE'EF   20    C1   12C9    959          ADDL3   #GBH$L_TRC_FLNK,GBH-4,-(SP) ; get address of trace queue
                   00001412'EF   DF   12D1    960          PUSHAL  GBH$L_TRC_FLNK+GBH              ; add local (SDA) address of queue
                00003885'EF   04    FB   12D7    961          CALLS   #4,SHOW_TRACE
                                        12DE    962 ;
                                        12DE    963 ; End of processing for specific IFAB and its subsidiary control blocks.
                                        12DE    964 ;
                                        12DE    965
                          57    D7   12DE    966 800$:    DECL    R7                              ; one less in this table
                          03    13   12E0    967          BEQL    810$                            ; if eql table empty now
                        FA02    31   12E2    968          BRW     20$                             ; go back and get next entry from table
                                        12E5    969
                                        12E5    970 ;
                                        12E5    971 ; Current table exhausted -- try to get new table segment
                                        12E5    972 ;
                                        12E5    973
                        F9B5    31   12E5    974 810$:    BRW     10$                             ; go back for more table
                                        12E8    975
                                        12E8    976 900$:    STATUS  SUCCESS                         ; end of IFAB tables
                          04    12EF    977          RET
                                        12F0    978          .DSABL  LSB
```

```
                                12F0   980          .SBTTL  SHOW_RMS -- Display current default RMS Data Struct.
                                12F0   981
                                12F0   982 ;+++
                                12F0   983 ; SHOW_RMS -- Print out default RMS structures
                                12F0   984 ;
                                12F0   985 ; Inputs:
                                12F0   986 ;
                                12F0   987 ;       RMS_DIS_OPT = Display options for this command.
                                12F0   988 ;       RMS_IFI     = ''
                                12F0   989 ;
                                12F0   990 ; Outputs:
                                12F0   991 ;
                                12F0   992 ;       None.
                                12F0   993 ;---
                                12F0   994
                         03FC   12F0   995 SHOW_RMS::.WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9>
00000006'EF    00000000'EF   D0  12F2   996          MOVL    RMS_DIS_OPT,RMS_DIS_OPT1
0000000A'EF    00000004'EF   B0  12FD   997          MOVW    RMS_IFI,RMS_IFI1
             F92D CF    00   FB  1308   998          CALLS   #0,SHOW_RMS1
                             04  130D   999          RET
                                130E  1000
                                130E  1001 ;+++
                                130E  1002 ; SHOW_RMS_DIS -- Print out RMS structures specified by /DISPLAY qualifer.
                                130E  1003 ;
                                130E  1004 ; Inputs:
                                130E  1005 ;
                                130E  1006 ;       RMS_DIS_TMP = Display options for this command.
                                130E  1007 ;       RMS_IFI_TMP = ''
                                130E  1008 ;
                                130E  1009 ; Outputs:
                                130E  1010 ;
                                130E  1011 ;       None.
                                130E  1012 ;---
                                130E  1013
                         03FC   130E  1014 SHOW_RMS_DIS::.WORD     ^M<R2,R3,R4,R5,R6,R7,R8,R9>
00000006'EF    0000000C'EF   D0  1310  1015          MOVL    RMS_DIS_TMP,RMS_DIS_OPT1
0000000A'EF    00000010'EF   B0  131B  1016          MOVW    RMS_IFI_TMP,RMS_IFI1
             F90F CF    00   FB  1326  1017          CALLS   #0,SHOW_RMS1
                             04  132B  1018          RET
```

```
                     132C  1020          .SBTTL  SHOW_RMS_OPT
                     132C  1021
                     132C  1022 ;+++
                     132C  1023 ; SHOW_RMS_OPT -- Print out current SET RMS, SHOW RMS options.
                     132C  1024 ;
                     132C  1025 ; Inputs:
                     132C  1026 ;
                     132C  1027 ;       RMS_DIS_OPT = Current RMS display options.
                     132C  1028 ;       RMS_IFI     = IFI for which to display structures.
                     132C  1029 ;
                     132C  1030 ; Outputs:
                     132C  1031 ;
                     132C  1032 ;       None.
                     132C  1033 ;---
                     132C  1034
                     132C  1035          .ENABL  LSB
                     132C  1036 SHOW_RMS_OPT::
                03FC 132C  1037          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9>
                     132E  1038          SKIP    1
                     1337  1039          ALLOC   ALLOCSZ,R4              ; Create some stack space for translate
    00000000'EF   DD 1349  1040          PUSHL   RMS_DIS_OPT             ; put longword bit argument on stack
      F4C5 CF   9F 134F  1041          PUSHAB  OPT_CHR                 ; translation table for options bits
00000000'EF   02 FB 1353  1042          CALLS   #2,TRANSLATE_BITS       ; perform translation
              54 DD 135A  1043          PUSHL   R4                      ; put result back on stack
                     135C  1044          PRINT   1,<-
                     135C  1045 RMS Display Options: !AS>
7E  00000004'EF   3C 1369  1046          MOVZWL  RMS_IFI,-(SP)           ; put IFI value on stack
              0F 13 1370  1047          BEQL    10$                     ; if all IFI
                     1372  1048          PRINT   1,<-
                     1372  1049 Display RMS structures only for IFI=!UW.>
              0D 11 137F  1050          BRB     20$
                     1381  1051 10$:    PRINT   0,<-
                     1381  1052 Display RMS structures for all IFI values.>
                     138E  1053 20$:    SKIP    1
                     1397  1054          STATUS  SUCCESS
                   04 139E  1055          RET
                     139F  1056          .DSABL  LSB
                     139F  1057
                     139F  1058          .SBTTL  RMS_HANDLER -- SHOW RMS exception handler
                     139F  1059 ;+++
                     139F  1060 ; RMS_HANDLER -- Turn ACCVIO's into INVBLKTYP from SHOW RMS
                     139F  1061 ;
                     139F  1062 ; Inputs:
                     139F  1063 ;
                     139F  1064 ;       4(AP) = Pointer to SIGNAL arguments.
                     139F  1065 ;       8(AP) = Pointer to MECHANISM arguments.
                     139F  1066 ;
                     139F  1067 ; Outputs:
                     139F  1068 ;
                     139F  1069 ;       Resignal if not ACCVIO.
                     139F  1070 ;
                     139F  1071 ;       Turn ACCVIO into INVBLKYP and unwind to caller of SHOW_RMS.
                     139F  1072 ;---
                     139F  1073
                     139F  1074 RMS_HANDLER:
                000C 139F  1075          .WORD   ^M<R2,R3>
52  04 AC             7D 13A1  1076          MOVQ    4(AP),R2                ; Get addresses of arrays
```

```
00000000'8F    04 A2    D1  13A5  1077          CMPL    CHF$L_SIG_NAME(R2),#SS$_ACCVIO ; is this access violation?
                   08    12  13AD  1078          BNEQ    10$                            ; if neq no
04 A2    00000000'8F    D0  13AF  1079          MOVL    #MSG$_RMSTERM,CHF$L_SIG_NAME(R2) ; reset the error
       50    0000'8F    3C  13B7  1080 10$:     MOVZWL  #SS$_RESIGNAL,R0               ; continue ...
                        04  13BC  1081          RET
```

```
                                13BD  1083          .SBTTL  SHOW_IFAB -- Print out useful contents of IFAB
                                13BD  1084
                                13BD  1085 ;+++
                                13BD  1086 ; SHOW_IFAB -- Print out useful contents of IFAB
                                13BD  1087 ;
                                13BD  1088 ; Inputs:
                                13BD  1089 ;
                                13BD  1090 ;       4(AP) = Address of local buffer containing IFAB
                                13BD  1091 ;
                                13BD  1092 ; Outputs:
                                13BD  1093 ;
                                13BD  1094 ;       IFAB information printed.
                                13BD  1095 ;---
                                13BD  1096
                                13BD  1097          .ENABL  LSB
                                13BD  1098 SHOW_IFAB::
                         001C   13BD  1099          .WORD   ^M<R2,R3,R4>
                                13BF  1100
                                13BF  1101 ;
                                13BF  1102 ; Do we need to show this structure?
                                13BF  1103 ;
                                13BF  1104
00000006'EF    02    D3         13BF  1105          BITL    #OPT$M_IFB,RMS_DIS_OPT1  ; Show IFAB?
               03    12         13C6  1106          BNEQ    2$                       ; if neq yes
             0450    31         13C8  1107          BRW     100$                     ; if eql no
       53    04 AC   D0         13CB  1108 2$:      MOVL    4(AP),R3                 ; R3 = address
                                13CF  1109          SKIP    PAGE
       52    23 A3   9A         13D6  1110          MOVZBL  IFB$B_ORGCASE(R3),R2     ; get the organization byte
   7E   F0E1 CF42   D0         13DA  1111          MOVL    ORG_TBL[R2],-(SP)        ; put the descriptor on stack
7E   00000016'EF    3C         13E0  1112          MOVZWL  IFI,-(SP)                ; put current IFI on stack
            FC A3   DD         13E7  1113          PUSHL   -4(R3)                   ; get set up to print out address
                                13EA  1114          PRINT   0,<-
                                13EA  1115 =============>
                                13F7  1116          PRINT   3,<-
                                13F7  1117 IFAB Address: !XL                IFI: !XW        Organization: !AC>
                                1404  1118          PRINT   0,<-
                                1404  1119 =============                    ----            ------------->
                                1411  1120          SKIP    1
                                141A  1121          ALLOC   ALLOCSZ,R4               ; allocate data area for text str
          63    DD              142C  1122          PUSHL   (R3)                     ; set up argument -- dev chr bits
        EBCE    CF    9F        142E  1123          PUSHAB  DEV_CHR                  ; address of table describing bits
00000000'EF     02    FB        1432  1124          CALLS   #2,TRANSLATE_BITS        ; only two args to keep descrip around
          54    DD              1439  1125          PUSHL   R4                       ; address of descriptor
          63    DD              143B  1126          PUSHL   (R3)                     ; device characteristics bits
                                143D  1127          PRINT   2,<-
                                143D  1128 PRIMDEV:         !XL    !AS>
6E  00000087    8F    D0        144A  1129          MOVL    #ALLOCSZ,(SP)            ; reset size of allocated buffer
          04    A3    DD        1451  1130          PUSHL   IFB$L_BKPBITS(R3)        ; get bookkeeping bits
        EDB8    CF    9F        145[illegible]  1131          PUSHAB  BKP_CHR
00000000'EF     02    FB        1458  1132          CALLS   #2,TRANSLATE_BITS
          54    DD              145F  1133          PUSHL   R4                       ; buffer which contains string
          04    A3    DD        1461  1134          PUSHL   IFB$L_BKPBITS(R3)        ; get the bits again
                                1464  1135          PRINT   2,<-
                                1464  1136 BKPBITS:       !XL    !AS>
       7E   08 A3   9A         1471  1137          MOVZBL  IFB$B_BID(R3),-(SP)      ; block ID
       7E   09 A3   9A         1475  1138          MOVZBL  IFB$B_BLN(R3),-(SP)      ; block length
                                1479  1139          PRINT   2,<-
```

```
                        1479  1140 BLN:            !XB     !-!8UB. -
                        1479  1141 BID:            !XB     !-!8UB.>
          7E  0A A3  9A 1486  1142          MOVZBL  IFB$B_MODE(R3),-(SP)    ; mode
          7E  0B A3  9A 148A  1143          MOVZBL  IFB$B_EFN(R3),-(SP)     ; event flag number
                        148E  1144          PRINT   2,<-
                        148E  1145 EFN:              !XB                     -
                        148E  1146 MODE:             !XB>
              14 A3  DD 149B  1147          PUSHL   IFB$L_ASBADDR(R3)       ; ASB address
              0C A3  DD 149E  1148          PUSHL   IFB$L_IOS(R3)           ; I/O status Block
                        14A1  1149          PRINT   2,<-
                        14A1  1150 IOS:              !XL                 -
                        14A1  1151 ASBADDR:          !XL>
              18 A3  DD 14AE  1152          PUSHL   IFB$L_ARGLST(R3)        ; User argument list
              10 A3  DD 14B1  1153          PUSHL   IFB$L_IOS4(R3)          ; end of I/O status block
                        14B4  1154          PRINT   2,<-
                        14B4  1155 IOS4:             !XL                 -
                        14B4  1156 ARGLST:           !XL>
          7E  20 A3  3C 14C1  1157          MOVZWL  IFB$W_CHNL(R3),-(SP)    ; I/O channel number
              1C A3  DD 14C5  1158          PUSHL   IFB$L_IRAB_LNK(R3)      ; link to IRAB list
                        14C8  1159          PRINT   2,<-
                        14C8  1160 IRAB_LNK:         !XL                 -
                        14C8  1161 CHNL:             !XW>
      6E  00000087 8F  D0 14D5  1162          MOVL    #ALLOCSZ,(SP)           ; reset size of allocated buffer
            7E  22 A3  9A 14DC  1163          MOVZBL  IFB$B_FAC(R3),-(SP)     ; FAC bits for translate
              ECBC CF  9F 14E0  1164          PUSHAB  FAC_CHR
00000000'EF    02  FB 14E4  1165          CALLS   #2,TRANSLATE_BITS
                54  DD 14EB  1166          PUSHL   R4
            7E  22 A3  9A 14ED  1167          MOVZBL  IFB$B_FAC(R3),-(SP)     ; FAC bits
                        14F1  1168          PRINT   2,<-
                        14F1  1169 FAC:              !XB         !AS>
            52  23 A3  9A 14FE  1170          MOVZBL  IFB$B_ORGCASE(R3),R2
      7E   EFB9 CF42  D0 1502  1171          MOVL    ORG_TBL[R2],-(SP)
            7E  23 A3  9A 1508  1172          MOVZBL  IFB$B_ORGCASE(R3),-(SP) ; organization
                        150C  1173          PRINT   2,<-
                        150C  1174 ORGCASE:          !XB         !AC>
              3C A3  DD 1519  1175          PUSHL   IFB$L_NWA_PTR(R3)       ; Network work area
              24 A3  DD 151C  1176          PUSHL   IFB$L_LAST_FAB(R3)      ; last user FAB address
                        151F  1177          PRINT   2,<-
                        151F  1178 LAST_FAB:         !XL                 -
                        151F  1179 NWA_PTR:          !XL>
      7E  2A A3  3C 152C  1180          MOVZWL  IFB$W_ECHO_ISI(R3),-(SP); ISI for echoing SYS$INPUT
      7E  28 A3  3C 1530  1181          MOVZWL  IFB$W_IFI(R3),-(SP)     ; saved IFI
                        1534  1182          PRINT   2,<-
                        1534  1183 IFI:              !XW                     -
                        1534  1184 ECHO_ISI:         !XW>
              38 A3  DD 1541  1185          PUSHL   IFB$L_FWA_PTR(R3)       ; pointer to FWA
              30 A3  DD 1544  1186          PUSHL   IFB$L_JNLBDB(R3)        ; journaling BDB
                        1547  1187          PRINT   2,<-
                        1547  1188 JNLBDB:           !XL                 -
                        1547  1189 FWA_PTR:          !XL>
      7E  48 A3  3C 1554  1190          MOVZWL  IFB$L_DEVBUFSIZ(R3),-(SP) ; default dev buffer size
              40 A3  DD 1558  1191          PUSHL   IFB$L_BDB_FLNK(R3)      ; BDB forward link
                        155B  1192          PRINT   2,<-
                        155B  1193 BDB_FLNK:         !XL                 -
                        155B  1194 DEVBUFSIZ:        !XW     !-!8UW.>
      7E  4C A3  3C 1568  1195          MOVZWL  IFB$W_RTDEQ(R3),-(SP)   ; default extend qty
              44 A3  DD 156C  1196          PUSHL   IFB$L_BDB_BLNK(R3)      ; BDB backward link
```

```
                                156F  1197          PRINT   2,<-
                                156F  1198 BDB_BLNK:        !XL             -
                                156F  1199 RTDEQ:           !XW     !-!8UW.>
52   50 A3    04    00    EF   157C  1200          EXTZV   #IFB$V_RFM,#4,IFB$B_RFMORG(R3),R2 ; get record format into R2
  7E   EF45 CF42    D0    1582  1201          MOVL    RFM_TB[R2],-(SP)        ; put string on stack
       7E    50 A3    9A   1588  1202          MOVZBL  IFB$B_RFMORG(R3),-(SP)  ; record format
                                158C  1203          PRINT   2,<-
                                158C  1204 RFMORG:          !XB             !AC>
     6E   00000087 8F    D0   1599  1205          MOVL    #ALLOCSZ,(SP)           ; reset size of allocated buffer
       7E    51 A3    9A   15A0  1206          MOVZBL  IFB$B_RAT(R3),-(SP)     ; record attributes for trans
          EC40 CF    9F   15A4  1207          PUSHAB  RAT_CRR
     00000000'EF    02    FB   15A8  1208          CALLS   #2,TRANSLATE_BITS
                 54    DD   15AF  1209          PUSHL   R4
       7E    51 A3    9A   15B1  1210          MOVZBL  IFB$B_RAT(R3),-(SP)     ; record attributes
                                15B5  1211          PRINT   2,<-
                                15B5  1212 RAT:             !XB             !AS>
          54 A3    DD   15C2  1213          PUSHL   IFB$L_HBK_DISK(R3)      ; highest block alloc (SWAPPED)
       7E    52 A3    3C   15C5  1214          MOVZWL  IFB$W_LRL(R3),-(SP)     ; longest record length
                                15C9  1215          PRINT   2,<-
                                15C9  1216 LRL:             !XW     !-!8UW. -
                                15C9  1217 HBK_DISK:        !XL>
          58 A3    DD   15D6  1218          PUSHL   IFB$L_EBK_DISK(R3)      ; end of file block (SWAPPED)
       7E    5C A3    3C   15D9  1219          MOVZWL  IFB$W_FFB(R3),-(SP)     ; first free byte
                                15DD  1220          PRINT   2,<-
                                15DD  1221 FFB:             .XW     !-!8UW. -
                                15DD  1222 EBK_DISK:        !XL>
       7E    5E A3    9A   15EA  1223          MOVZBL  IFB$B_BKS(R3),-(SP)     ; bucket size
       7E    5F A3    9A   15EE  1224          MOVZBL  IFB$B_FSZ(R3),-(SP)     ; header size
                                15F2  1225          PRINT   2,<-
                                15F2  1226 FSZ:             !XB     !-!8UB. -
                                15F2  1227 BKS:             !XB     !-!8UB.>
    7E    60 A3    3C   15FF  1228          MOVZWL  IFB$W_MRS(R3),-(SP)     ; maximum record size
    7E    62 A3    3C   1603  1229          MOVZWL  IFB$W_DEQ(R3),-(SP)     ; default extend size
                                1607  1230          PRINT   2,<-
                                1607  1231 DEQ:             !XW     !-!8UW. -
                                1607  1232 MRS:             !XW     !-!8UW.>
    7E    64 A3    3C   1614  1233          MOVZWL  IFB$W_GBC(R3),-(SP)     ; global buff count
                                1618  1234          PRINT   1,<-
                                1618  1235 GBC:             !XW     !-!8UW.>
          70 A3    DD   1625  1236          PUSHL   IFB$L_HBK(R3)           ; high block alloc
                                1628  1237          PRINT   1,<-
                                1628  1238 HBK:             !XL     !-!8UL.>
          74 A3    DD   1635  1239          PUSHL   IFB$L_EBK(R3)           ; end-of-file
                                1638  1240          PRINT   1,<-
                                1638  1241 EBK:             !XL     !-!8UL.>
          6C A3    DD   1645  1242          PUSHL   IFB$L_LOCK_BDB(R3)      ; lockbdb address
          6C A3    DD   1648  1243          PUSHL   IFB$L_RNS_LEN(R3)       ; multi-use field
                                164B  1244          PRINT   2,<-
                                164B  1245 RNS_LEN:         !XL             -
                                164B  1246 LOCK_BDB:        !XL>
 7E   0084 C3    3C   1658  1247          MOVZWL  IFB$W_AVLCL(R3),-(SP)   ; local buffers available
       78 A3    DD   165D  1248          PUSHL   IFB$L_SFSB_PTR(R3)      ; pointer to shared file synch block
                                1660  1249          PRINT   2,<-
                                1660  1250 SFSB_PTR:         !XL             -
                                1660  1251 AVLCL:            !XW     !-!8UW.>
                                166D  1252
 7E   0086 C3    3C   166D  1253          MOVZWL  IFB$W_AVGBPB(R3),-(SP)  ; global pointer blocks available
```

```
                7C A3    DD  1672  1254          PUSHL   IFB$L_GBSB_PTR(R3)       ; pointer to global buffer sync block
                         DD  1675  1255          PRINT   2,<-
                         DD  1675  1256 GBSB_PTR:        !XL                -
                         DD  1675  1257 AVGBPB:          !XW     !-!8UW.>
                         DD  1682  1258
             00A4 C3     DD  1682  1259          PUSHL   IFB$L_RJB(R3)            ; pointer to journaling block
             0088 C3     DD  1686  1260          PUSHL   IFB$L_GBH_PTR(R3)        ; pointer to global header
                         DD  168A  1261          PRINT   2,<-
                         DD  168A  1262 GBH_PTR:         !XL                -
                         DD  168A  1263 RJB_PTR:         !XL>
      6E  00000087 8F    D0  1697  1264          MOVL    #ALLOCSZ,(SP)            ; reset size of allocated buffer
          7E   00A0 C3     9A  169E  1265          MOVZBL  IFB$B_JNLFLG(R3),-(SP)   ; journal flag bits for translate
               EC91 CF     9F  16A3  1266          PUSHAB  JNLFLG_CHR
00000000'EF      02     FB  16A7  1267          CALLS   #2,TRANSLATE_BITS
                 54     DD  16AE  1268          PUSHL   R4
          7E   00A0 C3     9A  16B0  1269          MOVZBL  IFB$B_JNLFLG(R3),-(SP)   ; journal flag bits
                         DD  16B5  1270          PRINT   2,<-
                         DD  16B5  1271 JNLFLGS:         !XB        !AS>
      6E  00000087 8F    D0  16C2  1272          MOVL    #ALLOCSZ,(SP)            ; reset size of allocated buffer
          7E   00A1 C3     9A  16C9  1273          MOVZBL  IFB$B_RECVRFLGS(R3),-(SP); recover flag bits for translate
               EC46 CF     9F  16CE  1274          PUSHAB  RECFLG_CHR
00000000'EF      02     FB  16D2  1275          CALLS   #2,TRANSLATE_BITS
                 54     DD  16D9  1276          PUSHL   R4
          7E   00A1 C3     9A  16DB  1277          MOVZBL  IFB$B_RECVRFLGS(R3),-(SP); recover flag bits
                         DD  16E0  1278          PRINT   2,<-
                         DD  16E0  1279 RECVRFLGS:       !XB        !AS>
      6E  00000087 8F    D0  16ED  1280          MOVL    #ALLOCSZ,(SP)            ; reset size of allocated buffer
          7E   00A2 C3     9A  16F4  1281          MOVZBL  IFB$B_JNLFLG2(R3),-(SP)  ; journal flag2 bits for translate
               F4A7 CF     9F  16F9  1282          PUSHAB  JNLFLG2_CHR
00000000'EF      02     FB  16FD  1283          CALLS   #2,TRANSLATE_BITS
                 54     DD  1704  1284          PUSHL   R4
          7E   00A2 C3     9A  1706  1285          MOVZBL  IFB$B_JNLFLG2(R3),-(SP)  ; journal flag bits
                         DD  170B  1286          PRINT   2,<-
                         DD  170B  1287 JNLFLGS2:        !XB        !AS>
             0080 C3     DD  1718  1288          PUSHL   IFB$L_PAR_LOCK_ID(R3)    ; parent lock id
               34 A3     DD  171C  1289          PUSHL   IFB$L_EXTJNLBUF(R3)      ; pointer to extend MJB
                         DD  171F  1290          PRINT   2,<-
                         DD  171F  1291 EXTJNL_PTR:      !XL                -
                         DD  171F  1292 PAR_LOCK_ID:     !XL>
                         DD  172C  1293
                         DD  172C  1294 ;
                         DD  172C  1295 ; Case to organization dependent code
                         DD  172C  1296 ;
                         DD  172C  1297
   02    00    23 A3     8F  172C  1298          CASEB   IFB$B_ORGCASE(R3),#0,#2  ; branch to org dependant code
                    000E' 1731  1299 5$:      .WORD   10$-5$                   ; sequential
                    004C' 1733  1300          .WORD   15$-5$                   ; relative
                    004C' 1735  1301          .WORD   15$-5$                   ; indexed
                          1737  1302          STATUS  NOTVALID
                      04  173E  1303          RET
                         173F  1304
                         173F  1305 ;
                         173F  1306 ; Sequential organization dependant code
                         173F  1307 ;
                         173F  1308
      6E  00000087 8F    D0  173F  1309 10$:     MOVL    #ALLOCSZ,(SP)            ; reset size of allocated buffer
             008C C3     DD  1746  1310          PUSHL   IFB$L_AS_DEV(R3)         ; get set up to trans char bits
```

```
                E8B2 CF    9F   174A  1311          PUSHAB  DEV_CHR
      00000000'EF    02    FB   174E  1312          CALLS   #2,TRANSLATE_BITS
                  54       DD   1755  1313          PUSHL   R4
                008C C3    DD   1757  1314          PUSHL   IFB$L_AS_DEV(R3)        ; assigned dev characteristic bits
                           PRINT   175B  1315          PRINT   2,<-
                           175B  1316 AS_DEV:             !XL   !AS>
          7E    0094 C3    3C   1768  1317          MOVZWL  IFB$L_ASDEVBSIZ(R3),-(SP) ; assigned dev buffer size
                           PRINT   176D  1318          PRINT   1,<-
                           176D  1319 ASDEVBSIZ:          !XW     !-!8UW.>
                   009E    31   177A  1320          BRW     100$
                           177D  1321
                           177D  1322 ;
                           177D  1323 ; Code common to relative and indexed, but not sequential
                           177D  1324 ;
                           177D  1325
                0098 C3    DD   177D  1326 15$:      PUSHL   IFB$L_BLBFLNK(R3)
                           1781  1327          PRINT   1,<-
                           1781  1328 BLBFLNK:          !XL>
                009C C3    DD   178E  1329          PUSHL   IFB$L_BLBBLNK(R3)
                           1792  1330          PRINT   1,<-
                           1792  1331 BLBBLNK:          !XL>
                           179F  1332
      02    01    23 A3    8F   179F  1333          CASEB   IFB$B_ORGCASE(R3),#1,#2
                     0004' 17A4  1334 16$:      .WORD   20$-16$                 ; relative
                     001C' 17A6  1335          .WORD   30$-16$                 ; indexed
                           17A8  1336
                           17A8  1337 ;
                           17A8  1338 ; Relative organization dependant code
                           17A8  1339 ;
                           17A8  1340
                00B0 C3    DD   17A8  1341 20$:      PUSHL   IFB$L_DVBN(R3)          ; first data bucket VBN
                00AC C3    DD   17AC  1342          PUSHL   IFB$L_MRN(R3)           ; maximum record number
                           17B0  1343          PRINT   2,<-
                           17B0  1344 MRN:                !XL                -
                           17B0  1345 DVBN:               !XL!-!8UL.>
                   005B    31   17BD  1346          BRW     100$
                           17C0  1347
                           17C0  1348 ;
                           17C0  1349 ; Indexed organization dependant code
                           17C0  1350 ;
                           17C0  1351
          7E    00B0 C3    9A   17C0  1352 30$:      MOVZBL  IFB$B_AVBN(R3),-(SP)    ; VBN  of 1st area descriptor
                00AC C3    DD   17C5  1353          PUSHL   IFB$L_IDX_PTR(R3)       ; pointer to primary key idx desc
                           17C9  1354          PRINT   2,<-
                           17C9  1355 IDX_PTR:            !XL                -
                           17C9  1356 AVBN:               !XB     !-!8UB.>
          7E    00B2 C3    9A   17D6  1357          MOVZBL  IFB$B_NUM_KEYS(R3),-(SP) ; number of keys in file
          7E    00B1 C3    9A   17DB  1358          MOVZBL  IFB$B_AMAX(R3),-(SP)    ; total number of area descriptors
                           17E0  1359          PRINT   2,<-
                           17E0  1360 AMAX:               !XB     !-!8UB. -
                           17E0  1361 NUM_KEYS:           !XB     !-!8UB.>
          7E    00B4 C3    3C   17ED  1362          MOVZWL  IFB$W_KBUFSZ(R3),-(SP) ; key buffer size
          7E    00B3 C3    9A   17F2  1363          MOVZBL  IFB$B_UBUFSZ(R3),-(SP) ; update buffer size
                           17F7  1364          PRINT   2,<-
                           17F7  1365 UBUFSZ:             !XB     !-!8UB. -
                           17F7  1366 KBUFSZ:             !XB     !-!8UB.>
          7E    00B6 C3    9A   1804  1367          MOVZBL  IFB$B_EXTRABUF(R3),-(SP) ; number of buffers for cacheing
```

```
                7E    00B7 C3    9A  1809  1368          MOVZBL  IFB$B_PLG_VER(R3),-(SP) ; prolog version
                                     180E  1369          PRINT   2,<-
                                     180E  1370 PLG_VER:         !XB                    -
                                     180E  1371 EXTRABUF:        !XB      !-!8UB.>
                                     181B  1372
                                     181B  1373 100$:    STATUS  SUCCESS
                                     1822  1374          RET
                                     1823  1375          .DSABL  LSB
```

```
                     1823  1377          .SBTTL  SHOW_IDX
                     1823  1378
                     1823  1379 ;+++
                     1823  1380 ; SHOW_IDX -- Display useful contents of index descriptor.
                     1823  1381 ;
                     1823  1382 ; Inputs:
                     1823  1383 ;
                     1823  1384 ;       4(AP) = Address of buffer containing index descriptor.
                     1823  1385 ;
                     1823  1386 ; Outputs:
                     1823  1387 ;
                     1823  1388 ;       None.
                     1823  1389 ;---
                     1823  1390
                     1823  1391          .ENABL  LSB
                     1823  1392 SHOW_IDX:
                001C 1823  1393          .WORD   ^M<R2,R3,R4>
53   04 AC    D0     1825  1394          MOVL    4(AP),R3                ; get address of buffer into R3
                     1829  1395          ENSURE  23
                     1841  1396          SKIP    2
     FC A3    DD     184A  1397          PUSHL   -4(R3)                  ; virtual address of IDX
                     184D  1398          PRINT   1,<-
                     184D  1399 IDX Address: !XL>
                     185A  1400          PRINT   0,<-
                     185A  1401 ------------>
                     1867  1402          SKIP    1
7E   08 A3    9A     1870  1403          MOVZBL  IDX$B_BID(R3),-(SP)
        63    DD     1874  1404          PUSHL   IDX$L_IDXFL(R3)
                     1876  1405          PRINT   2,<-
                     1876  1406 IDXFL:           !XL             -
                     1876  1407 BID:             !XB     !-!8UB.>
7E   09 A3    9A     1883  1408          MOVZBL  IDX$B_BLN(R3),-(SP)
7E   12 A3    9A     1887  1409          MOVZBL  IDX$B_IANUM(R3),-(SP)
                     188E  1410          PRINT   2,<-
                     188B  1411 IANUM:           !XB     !-!8UB. -
                     188B  1412 BLN:             !XB     !-!8UB.>
7E   13 A3    9A     1898  1413          MOVZBL  IDX$B_LANUM(R3),-(SP)
7E   14 A3    9A     189C  1414          MOVZBL  IDX$B_DANUM(R3),-(SP)
                     18A0  1415          PRINT   2,<-
                     18A0  1416 DANUM:           !XB     !-!8UB. -
                     18A0  1417 LANUM:           !XB     !-!8UB.>
7E   15 A3    9A     18AD  1418          MOVZBL  IDX$B_ROOTLEV(R3),-(SP)
7E   16 A3    9A     18B1  1419          MOVZBL  IDX$B_IDXBKTSZ(R3),-(SP)
                     18B5  1420          PRINT   2,<-
                     18B5  1421 IDXBKTSZ:        !XB     !-!8UB. -
                     18B5  1422 ROOTLEV:         !XB     !-!8UB.>
     18 A3    DD     18C2  1423          PUSHL   IDX$L_ROOTVBN(R3)
7E   17 A3    9A     18C5  1424          MOVZBL  IDX$B_DATBKTSZ(R3),-(SP)
                     18C9  1425          PRINT   2,<-
                     18C9  1426 DATBKTSZ:        !XB     !-!8UB. -
                     18C9  1427 ROOTVBN:         !XL!-!8UL.>
                     18D6  1428
                     18D6  1429 ;
                     18D6  1430 ; Translate and dump the flags
                     18D6  1431 ;
                     18D6  1432
                     18D6  1433
```

```
                     18D6  1434          ALLOC   ALLOCSZ,R4
          7E  1C A3  9A    18E8  1435          MOVZBL  IDX$B_FLAGS(R3),-(SP)
              21 A3  95    18EC  1436          TSTB    IDX$B_KEYREF(R3)          ; is it primary key?
                 06  12    18EF  1437          BNEQ    5$                        ; if neq no
             ECEF CF  9F    18F1  1438          PUSHAB  IDX_CHR0
                 04  11    18F5  1439          BRB     7$
             ECB1 CF  9F    18F7  1440 5$:       PUSHAB  IDX_CHR1                  ; seconday key, use other table
00000000'EF      02  FB    18FB  1441 7$:       CALLS   #2,TRANSLATE_BITS
                 54  DD    1902  1442          PUSHL   R4
          7E  1C A3  9A    1904  1443          MOVZBL  IDX$B_FLAGS(R3),-(SP)
                     1908  1444          PRINT   2,<-
                     1908  1445 FLAGS:         !XB         !AS>
          52  1D A3  9A    1915  1446          MOVZBL  IDX$B_DATATYPE(R3),R2
    7E  ECF6 CF42    D0    1919  1447          MOVL    IDX_TBL[R2],-(SP)
          7E  1D A3  9A    191F  1448          MOVZBL  IDX$B_DATATYPE(R3),-(SP)
                     1923  1449          PRINT   2,<-
                     1923  1450 DATATYPE:      !XB         !AC>
          7E  1F A3  9A    1930  1451          MOVZBL  IDX$B_NULLCHAR(R3),-(SP)
                 01  DD    1934  1452          PUSHL   #1
          7E  1E A3  9A    1936  1453          MOVZBL  IDX$B_SEGMENTS(R3),-(SP)
                     193A  1454          PRINT   3,<-
                     193A  1455 SEGMENTS:      !XB      !-!8UB. -
                     193A  1456 NULLCHR:       !XB      "!AF">
          7E  20 A3  9A    1947  1457          MOVZBL  IDX$B_KEYSZ(R3),-(SP)
          7E  21 A3  9A    194B  1458          MOVZBL  IDX$B_KEYREF(R3),-(SP)
                     194F  1459          PRINT   2,<-
                     194F  1460 KEYREF:        !XB      !-!8UB. -
                     194F  1461 KEYSZ:         !XB      !-!8UB.>
          7E  22 A3  3C    195C  1462          MOVZWL  IDX$W_MINRECSZ(R3),-(SP)
          7E  24 A3  3C    1960  1463          MOVZWL  IDX$W_IDXFILL(R3),-(SP)
                     1964  1464          PRINT   2,<-
                     1964  1465 IDXFILL:       !XW      !-!8UW. -
                     1964  1466 MINRECSZ:      !XW      !-!8UW.>
          7E  26 A3  3C    1971  1467          MOVZWL  IDX$W_DATFILL(R3),-(SP)
                     1975  1468          PRINT   1,<-
                     1975  1469 DATFILL:       !XW      !-!8UW.>
          52  28 A3  9A    1982  1470          MOVZBL  IDX$B_IDXBKTYP(R3),R2
    7E  ECA9 CF42    D0    1986  1471          MOVL    IDX_TBL1[R2],-(SP)
          7E  28 A3  9A    198C  1472          MOVZBL  IDX$B_IDXBKTYP(R3),-(SP)
                     1990  1473          PRINT   2,<-
                     1990  1474 IDXBKTYP:      !XB         !AC>
          52  29 A3  9A    199D  1475          MOVZBL  IDX$B_DATBKTYP(R3),R2
    7E  EC8E CF42    D0    19A1  1476          MOVL    IDX_TBL1[R2],-(SP)
          7E  29 A3  9A    19A7  1477          MOVZBL  IDX$B_DATBKTYP(R3),-(SP)
                     19AB  1478          PRINT   2,<-
                     19AB  1479 DATBKTYP:      !XB         !AC>
                     19B8  1480
                     19B8  1481 ;
                     19B8  1482 ; Display the specific segments
                     19B8  1483 ;
                     19B8  1484
          52  2C A3  3E    19B8  1485          MOVAW   IDX$W_POSITION(R3),R2     ; get starting address of segments
          54  1E A3  9A    19BC  1486          MOVZBL  IDX$B_SEGMENTS(R3),R4     ; get number of segments
    54    54    02    78  19C0  1487          ASHL    #2,R4,R4                  ; multiply by 4 (1 longword / segment)
              54  52  C0    19C4  1488          ADDL2   R2,R4                     ; R4 = address beyond last segment
                     19C7  1489          SKIP    1
                     19D0  1490          PRINT   0,<-
```

```
                        19D0  1491          POSITION                SIZE            TYPE>
                        19DD  1492          PRINT   0,<-
                        19DD  1493          --------                ----            ---->
                        19EA  1494          SKIP    1
                        19F3  1495
      54   52   D1   19F3  1496 10$:       CMPL    R2,R4                   ; beyond last segment?
           28   13   19F6  1497            BEQL    90$                     ; if eql yes -- exit
   50   03 A2   9A   19F8  1498            MOVZBL  3(R2),R0                ; get datatype
7E  EC13 CF40   D0   19FC  1499            MOVL    IDX_TBL[R0],-(SP)       ; get string on stack
   7E   03 A2   9A   1A02  1500            MOVZBL  3(R2),-(SP)             ; datatype
           7E   D4   1A06  1501            CLRL    -(SP)                   ; make some room on stack
      7E   82   3C   1A08  1502            MOVZWL  (R2)+,-(SP)             ; put size on stack first
   04 AE   82   9A   1A0B  1503            MOVZBL  (R2)+,4(SP)             ; put position into previous hole
           82   95   1A0F  1504            TSTB    (R2)+                   ; step over datatype
                        1A11  1505          PRINT   4,<-
                        1A11  1506         !XW!-!8UW.          !XB!-!8UB.      !XB   !AC>
           D3   11   1A1E  1507            BRB     10$
                        1A20  1508
                        1A20  1509 90$:     STATUS  SUCCESS
                  04    1A27  1510          RET
                        1A28  1511          .DSABL  LSB
```

```
                        1A28  1513          .SBTTL  SHOW_FWA
                        1A28  1514
                        1A28  1515 ;+++
                        1A28  1516 ; SHOW_FWA -- Show information in FWA.
                        1A28  1517 ;
                        1A28  1518 ;  Inputs:
                        1A28  1519 ;
                        1A28  1520 ;       4(AP) = Address of buffer containing FWA
                        1A28  1521 ;
                        1A28  1522 ;  Outputs:
                        1A28  1523 ;
                        1A28  1524 ;       Information printed out for FWA.
                        1A28  1525 ;--
                        1A28  1526
                        1A28  1527          .ENABL  LSB
                        1A28  1528 SHOW_FWA:
                003C    1A28  1529          .WORD   ^M<R2,R3,R4,R5>
    53    04 AC   D0    1A2A  1530          MOVL    4(AP),R3
                        1A2E  1531          SKIP    PAGE
          FC A3   DD    1A35  1532          PUSHL   -4(R3)
                        1A38  1533          PRINT   1,<-
                        1A38  1534 FWA Address: !XL>
                        1A45  1535          PRINT   0,<-
                        1A45  1536 ------------>
                        1A52  1537 ;
                        1A52  1538 ; Form value which will be used to adjust descriptors in the
                        1A52  1539 ; FWA.  If  x=address in FWA, y=address of FWA in users image, z=address of
                        1A52  1540 ; FWA in SDA, then address of descriptor in SDA = x-y+z.  This is also
                        1A52  1541 ; x+(-y+z) = x+(z-y).  We now form (z-y) in R5.
                        1A52  1542 ;
                        1A52  1543
       55    53   D0    1A52  1544          MOVL    R3,R5                   ; copy z (address of FWA in SDA)
    55    FC A3   C2    1A55  1545          SUBL2   -4(R3),R5               ; form z-y
                        1A59  1546          ALLOC   ALLOCSZ,R4
                        1A6B  1547 ;
                        1A6B  1548 ; Now dump the bit fields
                        1A6B  1549 ;
                        1A6B  1550
          63    DD      1A6B  1551          PUSHL   FWA$Q_FLAGS(R3)
                        1A6D  1552          PRINT   1,<-
                        1A6D  1553 FLAGS:           !XL>
          04 A3   DD    1A7A  1554          PUSHL   FWA$Q_FLAGS+4(R3)
                        1A7D  1555          PRINT   1,<-
                        1A7D  1556                  !XL>
                        1A8A  1557          SKIP    1
                        1A93  1558          FWABITS PASS
                        1ABA  1559          FWABITS FLD
                        1AE3  1560          FWABITS WILD
                        1B0C  1561          FWABITS PARSE
                        1B35  1562          FWABITS DIR
                        1B5E  1563          FWABITS DIRWC
                        1B87  1564          FWABITS LN
                        1BB0  1565          FWABITS SL
                        1BD9  1566          SKIP    1
                        1BE2  1567
                        1BE2  1568 ;
                        1BE2  1569 ; More normal dumping of intermediate FWA cells
```

```
                        1BE2  1570 ;
                        1BE2  1571
        0B A3    9F     1BE2  1572          PUSHAB  FWA$B_ROOTERM(R3)        ; address of string
           01    DD     1BE5  1573          PUSHL   #1                       ; length of string
  7E    0B A3    9A     1BE7  1574          MOVZBL  FWA$B_ROOTERM(R3),-(SP) ; value of string
        0A A3    9F     1BEB  1575          PUSHAB  FWA$B_DIRTERM(R3)        ; address of string
           01    DD     1BEE  1576          PUSHL   #1                       ; length of string
  7E    0A A3    9A     1BF0  1577          MOVZBL  FWA$B_DIRTERM(R3),-(SP) ; value of string
                        1BF4  1578          PRINT   6,<-
                        1BF4  1579 DIRTERM:         !XB     "!AF"           -
                        1BF4  1580 ROOTERM:         !XB     "!AF">
                        1C01  1581
  7E    0E A3    9A     1C01  1582          MOVZBL  FWA$W_ESCIFI(R3),-(SP)
        0C A3    DD     1C05  1583          PUSHL   FWA$L_ESCSTRING(R3)
                        1C08  1584          PRINT   2,<-
                        1C08  1585 ESCSTRING:       !XL             -
                        1C08  1586 ESCIFI:          !XW>
7E  009E C3    3C       1C15  1587          MOVZWL  FWA$W_XLTSIZ(R3),-(SP)
7E  009D C3    9A       1C1A  1588          MOVZBL  FWA$B_XLTMODE(R3),-(SP)
                        1C1F  1589          PRINT   2,<-
                        1C1F  1590 XLTMODE:         !XB                     -
                        1C1F  1591 XLTSIZ:          !XW!-!8UW.>
                        1C2C  1592          PRINTD  FWA$Q_,FIB
        30 A3    DD     1C3F  1593          PUSHL   FWA$L_DIRBDB(R3)
7E  009C C3    9A       1C42  1594          MOVZBL  FWA$B_BUFFLG(R3),-(SP)
                        1C47  1595          PRINT   2,<-
                        1C47  1596 BUFFLG:          !XB                     -
                        1C47  1597 DIRBDB:          !XL>
                        1C54  1598          PRINTD  FWA$Q_,LOGNAM
        34 A3    DD     1C69  1599          PUSHL   FWA$L_LOOKUP(R3)
        28 A3    DD     1C6C  1600          PUSHL   FWA$L_UIC(R3)
                        1C6F  1601          PRINT   2,<-
                        1C6F  1602 UIC:             !XL             -
                        1C6F  1603 LOOKUP:          !XL>
  7E    2C A3    3C     1C7C  1604          MOVZWL  FWA$W_PRO(R3),-(SP)
        38 A3    DD     1C80  1605          PUSHL   FWA$L_DEVNODADR(R3)
                        1C83  1606          PRINT   2,<-
                        1C83  1607 DEVNODADR:       !XL             -
                        1C83  1608 PRO:             !XW>
                        1C90  1609          PRINTD  FWA$Q_,DIR
  7E    44 A3    3C     1CA3  1610          MOVZWL  FWA$W_UCHAR(R3),-(SP)
7E  00C3 C3    9A       1CA7  1611          MOVZBL  FWA$B_LEVEL(R3),-(SP)
                        1CAC  1612          PRINT   2,<-
                        1CAC  1613 LEVEL:           !XB!-!8UB.              -
                        1CAC  1614 UCHAR:           !XW>
  7E    2F A3    9A     1CB9  1615          MOVZBL  FWA$B_SUBNODCNT(R3),-(SP)
  7E    2E A3    9A     1CBD  1616          MOVZBL  FWA$B_DIRLEN(R3),-(SP)
                        1CC1  1617          PRINT   2,<-
                        1CC1  1618 DIRLEN:          !XB!-!8UB.              -
                        1CC1  1619 SUBNODCNT:       !XB!-!8UB.>
        4C A3    DD     1CCE  1620          PUSHL   FWA$L_SWB_PTR(R3)
                        1CD1  1621          PRINT   1,<-
                        1CD1  1622 SWB_PTR:         !XL>
     00AC C3    DD      1CDE  1623          PUSHL   FWA$L_SLB_PTR(R3)
     00A8 C3    DD      1CE2  1624          PUSHL   FWA$L_SLBH_PTR(R3)
                        1CE6  1625          PRINT   2,<-
                        1CE6  1626 SLBH_PTR:        !XL             -
```

```
                 1CE6  1627 SLB_PTR:          .XL>
00B4 C3    DD    1CF3  1628          PUSHL    FWA$L_SLBH_BLINK(R3)
00B0 C3    DD    1CF7  1629          PUSHL    FWA$L_SLBH_FLINK(R3)
                 1CFB  1630          PRINT    2,<-
                 1CFB  1631 SLBH_FLINK:       !XL              -
                 1CFB  1632 SLBH_BLINK:       !XL>
                 1D08  1633
                 1D08  1634 ;
                 1D08  1635 ; Dump Descriptors
                 1D08  1636 ;
                 1D08  1637
                 1D08  1638          SKIP     1
                 1D11  1639          DUMPDF   NODE
                 1D38  1640          DUMPDF   DEVICE
                 1D5F  1641          DUMPDF   CONCEAL_DEV
                 1D86  1642          DUMPDF   CDIR1
                 1DAD  1643          DUMPDF   CDIR2
                 1DD4  1644          DUMPDF   CDIR3
                 1DFB  1645          DUMPDF   CDIR4
                 1E22  1646          DUMPDF   CDIR5
                 1E49  1647          DUMPDF   CDIR6
                 1E70  1648          DUMPDF   CDIR7
                 1E97  1649          DUMPDF   CDIR8
                 1EBE  1650          DUMPDF   DIR1
                 1EE5  1651          DUMPDF   DIR2
                 1F0C  1652          DUMPDF   DIR3
                 1F33  1653          DUMPDF   DIR4
                 1F5A  1654          DUMPDF   DIR5
                 1F81  1655          DUMPDF   DIR6
                 1FA8  1656          DUMPDF   DIR7
                 1FCF  1657          DUMPDF   DIR8
                 1FF6  1658          DUMPDF   NAME
                 201D  1659          DUMPDF   TYPE
                 2044  1660          DUMPDF   RNS
                 206B  1661          DUMPDF   VERSION
                 2092  1662          DUMPDF   SHRFIL
                 20B9  1663
                 20B9  1664          SKIP     1
                 20C2  1665          PRINTD   FWA$Q_,NODE1
                 20D7  1666          PRINTD   FWA$Q_,NODE2
                 20EC  1667          PRINTD   FWA$Q_,NODE3
                 2101  1668          PRINTD   FWA$Q_,NODE4
                 2116  1669          PRINTD   FWA$Q_,NODE5
                 212B  1670          PRINTD   FWA$Q_,NODE6
                 2140  1671          PRINTD   FWA$Q_,NODE7
                 2155  1672          PRINTD   FWA$Q_,NODE8
                 216A  1673
                 216A  1674          SKIP     1
                 2173  1675          DUMPDF   AIJNL
                 219A  1676          DUMPDF   ATJNL
                 21C1  1677          DUMPDF   BIJNL
                 21E8  1678
                 21E8  1679 ;
                 21E8  1680 ; Additional non-descriptor cells
                 21E8  1681 ;
                 21E8  1682
                 21E8  1683 100$:    STATUS   SUCCESS
```

```
                    04  21EF  1684          RET
                        21F0  1685          .DSABL  LSB
                        21F0  1686
                        21F0  1687 ;+++
                        21F0  1688 ; FWADD -- Dump descriptors in FWA
                        21F0  1689 ;
                        21F0  1690 ; Inputs:
                        21F0  1691 ;
                        21F0  1692 ;       R2=Address of descriptor.
                        21F0  1693 ;       R5=Offset to add to address for local descriptor pointer.
                        21F0  1694 ;
                        21F0  1695 ; Outputs:
                        21F0  1696 ;
                        21F0  1697 ;       String printed if possible.
                        21F0  1698 ;---
                        21F0  1699
                        21F0  1700          .ENABL  LSB
                003C    21F0  1701 FWADD:   .WORD   ^M<R2,R3,R4,R5>
              04 A2     DD    21F2  1702          PUSHL   4(R2)                   ; see if descriptor points into FWA
    6E  000002F6'EF     C2    21F5  1703          SUBL2   FWA-4,(SP)              ; offset from FWA
    8E  0000093C 8F     D1    21FC  1704          CMPL    #FWA$C_BLN,(SP)+        ; within FWA?
                08      1F    2203  1705          BLSSU   10$                     ; if blssu no
                        2205  1706
                        2205  1707 ;
                        2205  1708 ; Within FWA -- dump more or less normally
                        2205  1709 ;
                        2205  1710
              04 A2     DD    2205  1711          PUSHL   4(R2)                   ; address in process
            6E  55      C0    2208  1712          ADDL2   R5,(SP)                 ; correction previously calc for FWA
                62      DD    220B  1713          PUSHL   (R2)                    ; length
              002E      31    220D  1714          BRW     20$                     ; print statement
                        2210  1715
                        2210  1716 ;
                        2210  1717 ; See if we can find it...
                        2210  1718 ;
                        2210  1719
                        2210  1720 10$:     ALLOC   ALLOCSZ,R4              ; make some space for it
64      62      D1      2222  1721          CMPL    (R2),(R4)               ; is it enough space?
          24    1A      2225  1722          BGTRU   30$                     ; if gtr no
                        2227  1723          GETMEM  @4(R2),@4(R4),(R2)      ; get memory if possible
  12 50         E9      2236  1724          BLBC    R0,30$                  ; if lbc then couldn't get it
  04 A4         DD      2239  1725          PUSHL   4(R4)                   ; push address of string
      62        DD      223C  1726          PUSHL   (R2)                    ; length too
                        223E  1727
                        223E  1728 20$:     PRINT   2,<-
                        223E  1729                          !AF>
                        224B  1730
                    04  224B  1731 30$:     RET
                        224C  1732          .DSABL  LSB
```

```
                                224C  1734          .SBTTL  SHOW_CCB
                                224C  1735
                                224C  1736 ;+++
                                224C  1737 ; SHOW_CCB -- Show information from CCB.
                                224C  1738 ;
                                224C  1739 ; Inputs:
                                224C  1740 ;
                                224C  1741 ;       4(AP) = Address of buffer containing CCB
                                224C  1742 ;
                                224C  1743 ; Outputs:
                                224C  1744 ;
                                224C  1745 ;       None.
                                224C  1746 ;---
                                224C  1747
                                224C  1748          .ENABL  LSB
                                224C  1749 SHOW_CCB:
                         001C   224C  1750          .WORD   ^M<R2,R3,R4>
00000006'EF   00000080 8F  D3   224E  1751          BITL    #OPT$M_CCB,RMS_DIS_OPT1 ; show ccb's?
                       03  12   2259  1752          BNEQ    1$                      ; if neq yes
                     00AE  31   225B  1753          BRW     90$                     ; if eql no
                  53   04 AC  D0   225E  1754 1$:     MOVL    4(AP),R3                ; get ccb buffer address
                                2262  1755          SKIP    PAGE
                       FC A3  DD   2269  1756          PUSHL   -4(R3)                  ; address in dump of CCB
                                226C  1757          PRINT   1,<-
                                226C  1758 CCB Address: !XL>
                                2279  1759          PRINT   0,<-
                                2279  1760 ------------>
                                2286  1761          SKIP    1
                       04 A3  DD   228F  1762          PUSHL   CCB$L_WIND(R3)          ; address of window control block
                          63  DD   2292  1763          PUSHL   CCB$L_UCB(R3)           ; UCB address of the device
                                2294  1764          PRINT   2,<-
                                2294  1765 UCB:               !XL             -
                                2294  1766 WIND:              !XL>
                                22A1  1767          ALLOC   ALLOCSZ,R4
                    7E    08 A3  9A   22B3  1768          MOVZBL  CCB$B_STS(R3),-(SP)
                       E395 CF  9F   22B7  1769          PUSHAB  CCB_CHR
00000000'EF            02  FB   22BB  1770          CALLS   #2,TRANSLATE_BITS
                          54  DD   22C2  1771          PUSHL   R4
                    7E    08 A3  9A   22C4  1772          MOVZBL  CCB$B_STS(R3),-(SP)
                                22C8  1773          PRINT   2,<-
                                22C8  1774 STS:             !XB         !AS>
                    52    09 A3  9A   22D5  1775          MOVZBL  CCB$B_AMOD(R3),R2
                       52   05  D1   22D9  1776          CMPL    #5,R2
                            03  14   22DC  1777          BGTR    30$
                       52   05  D0   22DE  1778          MOVL    #5,R2
                7E   E37A CF42  D0   22E1  1779 30$:     MOVL    AMODE_TBL[R2],-(SP)
                    7E    09 A3  9A   22E7  1780          MOVZBL  CCB$B_AMOD(R3),-(SP)
                                22EB  1781          PRINT   2,<-
                                22EB  1782 AMOD:            !XB         !AC>
                          0C A3  DD   22F8  1783          PUSHL   CCB$L_DIRP(R3)
                    7E    0A A3  3C   22FB  1784          MOVZWL  CCB$W_IOC(R3),-(SP)
                                22FF  1785          PRINT   2,<-
                                22FF  1786 IOC:             !XW     !-!8UW. -
                                22FF  1787 DIRP:            !XL>
                                230C  1788 90$:     STATUS  SUCCESS
                             04 2313  1789          RET
                                2314  1790          .DSABL  LSB
```

```
                                2314  1792          .SBTTL  SHOW_WCB
                                2314  1793
                                2314  1794 ;+++
                                2314  1795 ; SHOW_WCB -- Show information from Window Control Block.
                                2314  1796 ;
                                2314  1797 ; Inputs:
                                2314  1798 ;
                                2314  1799 ;       4(AP) = Address of buffer containing WCB.
                                2314  1800 ;
                                2314  1801 ; Outputs:
                                2314  1802 ;
                                2314  1803 ;       None.
                                2314  1804 ;---
                                2314  1805          .ENABL  LSB
                                2314  1806 SHOW_WCB:
                           003C 2314  1807          .WORD   ^M<R2,R3,R4,R5>
00000006'EF    00000100 8F   D3 2316  1808          BITL    #OPT$M_WCB,RMS_DIS_OPT1  ; display wcb's?
                        03   12 2321  1809          BNEQ    1$                       ; if neq yes
                      01BA   31 2323  1810          BRW     90$                      ; if eql no
                   53 04 AC   D0 2326  1811 1$:     MOVL    4(AP),R3                 ; address of buffer with WCB
                             232A  1812          ENSURE  19
                             2342  1813          SKIP    2
                      FC A3   DD 234B  1814          PUSHL   -4(R3)                   ; address in dump of WCB
                             234E  1815          PRINT   1,<-
                             234E  1816 WCB Address: !XL>
                             235B  1817          PRINT   0,<-
                             235B  1818 ------------>
                             2368  1819          SKIP    1
                   7E 08 A3   3C 2371  1820          MOVZWL  WCB$W_SIZE(R3),-(SP)
                      63   DD 2375  1821          PUSHL   WCB$L_WLFL(R3)
                             2377  1822          PRINT   2,<-
                             2377  1823 WLFL:             !XL                -
                             2377  1824 SIZE:             !XW     !-!8UW.>
                7E 0A A3   9A 2384  1825          MOVZBL  WCB$B_TYPE(R3),-(SP)
                   04 A3   DD 2388  1826          PUSHL   WCB$L_WLBL(R3)
                             238B  1827          PRINT   2,<-
                             238B  1828 WLBL:             !XL                -
                             238B  1829 TYPE:             !XB>
                             2398  1830          ALLOC   ALLOCSZ,R4
                7E 0B A3   9A 23AA  1831          MOVZBL  WCB$B_ACCESS(R3),-(SP)
                   E2C6 CF   9F 23AE  1832          PUSHAB  WCBSHR_CHR
00000000'EF        02   FB 23B2  1833          CALLS   #2,TRANSLATE_BITS
                      54   DD 23B9  1834          PUSHL   R4
                7E 0B A3   9A 23BB  1835          MOVZBL  WCB$B_ACCESS(R3),-(SP)
                             23BF  1836          PRINT   2,<-
                             23BF  1837 ACCESS:           !XB        !AS>
                   10 A3   DD 23CC  1838          PUSHL   WCB$L_ORGUCB(R3)
                0B A3   08   93 23CF  1839          BITB    #WCB$M_SHRWCB,WCB$B_ACCESS(R3) ; is this shared WCB?
                        12   12 23D3  1840          BNEQ    10$                      ; if neq yes
                             23D5  1841
                             23D5  1842 ;
                             23D5  1843 ; Unshared WCB -- print PID of owner
                             23D5  1844 ;
                             23D5  1845
                   0C A3   DD 23D5  1846          PUSHL   WCB$L_PID(R3)
                             23D8  1847          PRINT   2,<-
                             23D8  1848 PID:              !XL                -
```

```
                                23D8  1849 ORGUCB:          !XL>
                          11    23E5  1850          BRB     20$
                                23E7  1851
                                23E7  1852 ;
                                23E7  1853 ; Shared WCB -- prirt reference count of WCB
                                23E7  1854 ;
                                23E7  1855
                    7E  0E A3   23E7  1856 10$:     MOVZWL  WCB$W_REFCNT(R3),-(SP)
                                23EB  1857          PRINT   2,<-
                                23EB  1858 REFCNT:          !XW     !-!8UW. -
                                23EB  1859 ORGUCB:          !XL>
                                23F8  1860
                6E  00000087 8F 23F8  1861 20$:     MOVL    #ALLOCSZ,(SP)           ; reset buffer on stack
                    7E  14 A3   23FF  1862          MOVZWL  WCB$W_ACON(R3),-(SP)
                        E2A1 CF 2403  1863          PUSHAB  WCBACC_CHR
        00000000'EF     02 FB 2407  1864          CALLS   #2,TRANSLATE_BITS
                        54 DD 240E  1865          PUSHL   R4
                    7E  14 A3   2410  1866          MOVZWL  WCB$W_ACON(R3),-(SP)
                                2414  1867          PRINT   2,<-
                                2414  1868 ACON:          !XW         !AS>
                        18 A3   2421  1869          PUSHL   WCB$L_FCB(R3)           ; FCB address
                    7E  16 A3   2424  1870          MOVZWL  WCB$W_NMAP(R3),-(SP)    ; number of map pointers
                                2428  1871          PRINT   2,<-
                                2428  1872 NMAP:          !XW     !-!8UW. -
                                2428  1873 FCB:           !XL>
                        1C A3   2435  1874          PUSHL   WCB$L_RVT(R3)
                        2C A3   2438  1875          PUSHL   WCB$L_STVBN(R3)
                                243B  1876          PRINT   2,<-
                                243B  1877 STVBN:          !XL!-!8UL.      -
                                243B  1878 RVT:            !XL>
                                2448  1879
                                2448  1880 ;
                                2448  1881 ; Print out all mapping pointers that fit into our buffer.
                                2448  1882 ;
                                2448  1883
                                2448  1884          SKIP    1
                                2451  1885          PRINT   0,<-
                                2451  1886           VBN     RVN       Starting LBN                Count>
                                245E  1887          PRINT   0,<-
                                245E  1888           ---     ---       ------------                ----->
                                246B  1889          SKIP    1
                    55  2C A3   DO   2474  1890          MOVL    WCB$L_STVBN(R3),R5      ; initial VBN for this window
                    52  30 A3   9E   2478  1891          MOVAB   WCB$W_P1_COUNT(R3),R2   ; address of start of mapping pointers
                    54  16 A3   3C   247C  1892          MOVZWL  WCB$W_NMAP(R3),R4       ; number of mapping pointer
                        54  06   C4   2480  1893          MULL2   #6,R4                   ; number of bytes of mapping pointers
                        54  52   C0   2483  1894          ADDL2   R2,R4                   ; address beyond end of mapping ptrs
        00000F7E'8F     54  D1   2486  1895          CMPL    R4,#WCBEND              ; is it beyond the buffer?
                        26  15   248D  1896          BLEQ    30$                     ; if leq no -- its ok
    54  00000F7E'EF     DE   248F  1897          MOVAL   WCBEND,R4               ; set new end of buffer
                                2496  1898          SKIP    1
                                249F  1899          PRINT   0,<-
                                249F  1900          Pointer table doesn't fit SDA memory buffer>
                                24AC  1901          SKIP    1
                                24B5  1902
                    54  52   D1   24B5  1903 30$:     CMPL    R2,R4                   ; end of pointers yet?
                        26  18   24B8  1904          BGEQ    90$                     ; if geq yes -- exit
                    7E  82   3C   24BA  1905          MOVZWL  (R2)+,-(SP)             ; get size
```

```
                        82   DD  24BD  1906          PUSHL   (R2)+                   ; get LBN
          50   6E   08   18   EF  24BF  1907          EXTZV   #24,#8,(SP),R0          ; get Relative volume number
          6E   6E   18   00   EF  24C4  1908          EXTZV   #0,#24,(SP),(SP)       ; replace LBN with LBN
                        50   DD  24C9  1909          PUSHL   R0                      ; push RVN
                        55   DD  24CB  1910          PUSHL   R5                      ; push VBN
                55   0C AE   C0  24CD  1911          ADDL2   12(SP),R5               ; adjust VBN for next time
                                24D1  1912          PRINT   4,<-
                                24D1  1913 !12UL.     !3UL  !XL !-!12UL.      !XW !-!8UW.>
                        D5   11 24DE  1914          BRB     30$
                                24E0  1915
                                24E0  1916 90$:     STATUS  SUCCESS
                             04 24E7  1917          RET
                                24E8  1918          .DSABL  LSB
```

```
                                24E8  1920          .SBTTL  SHOW_FCB
                                24E8  1921
                                24E8  1922 ;+++
                                24E8  1923 ; SHOW_FCB -- Show information from File Control Block.
                                24E8  1924 ;
                                24E8  1925 ; Inputs:
                                24E8  1926 ;
                                24E8  1927 ;       4(AP) = Address of buffer containing FCB.
                                24E8  1928 ;
                                24E8  1929 ; Outputs:
                                24E8  1930 ;
                                24E8  1931 ;       None.
                                24E8  1932 ;---
                                24E8  1933
                                24E8  1934          .ENABL  LSB
                                24E8  1935 SHOW_FCB:
                         001C   24E8  1936          .WORD   ^M<R2,R3,R4>
00000006'EF    00000200 8F       D3  24EA  1937          BITL    #OPT$M_FCB,RMS_DIS_OPT1 ; display FCB's?
                       03       12  24F5  1938          BNEQ    1$                      ; if neq yes
                     026C       31  24F7  1939          BRW     90$                     ; if eql no
                  53   04 AC    D0  24FA  1940 1$:       MOVL    4(AP),R3                ; address of FCB buffer
                                24FE  1941          ENSURE  19
                                2516  1942          SKIP    2
                       FC A3    DD  251F  1943          PUSHL   -4(R3)                  ; address of FCB in dump
                                2522  1944          PRINT   1,<-
                                2522  1945 FCB Address: !XL>
                                252F  1946          PRINT   0,<-
                                252F  1947 ------------>
                                253C  1948          SKIP    1
7E    08 A3    3C               2545  1949          MOVZWL  FCB$W_SIZE(R3),-(SP)
         63    DD               2549  1950          PUSHL   FCB$L_FCBFL(R3)
                                254B  1951          PRINT   2,<-
                                254B  1952 FCBFL:           !XL             -
                                254B  1953 SIZE:            !XW     !-!8UW.>
7E    0A A3    9A               2558  1954          MOVZBL  FCB$B_TYPE(R3),-(SP)
      04 A3    DD               255C  1955          PUSHL   FCB$L_FCBBL(R3)
                                255F  1956          PRINT   2,<-
                                255F  1957 FCBBL:           !XL             -
                                255F  1958 TYPE:            !XB>
      0C A3    DD               256C  1959          PUSHL   FCB$L_EXFCB(R3)
7E    1A A3    3C               256F  1960          MOVZWL  FCB$W_ACNT(R3),-(SP)
                                2573  1961          PRINT   2,<-
                                2573  1962 ACNT:            !XW     !-!8UW. -
                                2573  1963 EXFCB:           !XL>
      10 A3    DD               2580  1964          PUSHL   FCB$L_WLFL(R3)
7E    1E A3    3C               2583  1965          MOVZWL  FCB$W_LCNT(R3),-(SP)
                                2587  1966          PRINT   2,<-
                                2587  1967 LCNT:            !XW     !-!8UW. -
                                2587  1968 WLFL:            !XL>
      14 A3    DD               2594  1969          PUSHL   FCB$L_WLBL(R3)
7E    1C A3    3C               2597  1970          MOVZWL  FCB$W_WCNT(R3),-(SP)
                                259B  1971          PRINT   2,<-
                                259B  1972 WCNT:            !XW     !-!8UW. -
                                259B  1973 WLBL:            !XL>
                                25A8  1974
                                25A8  1975 ;
                                25A8  1976 ; display the status bits in the FCB
```

```
                              25A8   ;
                              25A8
                              25A8          ALLOC   ALLOCSZ,R4
             7E   22 A3   3C  25BA          MOVZWL  FCB$W_STATUS(R3),-(SP)
               E176 CF    9F  25BE          PUSHAB  FCB_CHR
        00000000'EF  02   FB  25C2          CALLS   #2,TRANSLATE_BITS
                     54   DD  25C9          PUSHL   R4
             7E   22 A3   3C  25CB          MOVZWL  FCB$W_STATUS(R3),-(SP)
                              25CF          PRINT   2,<-
                              25CF   STATUS:        !XW          !AS>
             7E   28 A3   3C  25DC          MOVZWL  FCB$W_FID_RVN(R3),-(SP)
             7E   24 A3   3C  25E0          MOVZWL  FCB$W_FID_NUM(R3),-(SP)
                              25E4          PRINT   2,<-
                              25E4   FID_NUM:       !XW     !-!8UW. -
                              25E4   FID_RVN:       !XW     !-!8UW.>
             7E   2A A3   3C  25F1          MOVZWL  FCB$W_SEGN(R3),-(SP)
             7E   26 A3   3C  25F5          MOVZWL  FCB$W_FID_SEQ(R3),-(SP)
                              25F9          PRINT   2,<-
                              25F9   FID_SEQ:       !XW     !-!8UW. -
                              25F9   SEGN:          !XW     !-!8UW.>
                  30 A3   DD  2606          PUSHL   FCB$L_STLBN(R3)
                  2C A3   DD  2609          PUSHL   FCB$L_STVBN(R3)
                              260C          PRINT   2,<-
                              260C   STVBN:         !XL!-!8UL.      -
                              260C   STLBN:         !XL!-!8UL.>
                  34 A3   DD  2619          PUSHL   FCB$L_HDLBN(R3)
                  38 A3   DD  261C          PUSHL   FCB$L_FILESIZE(R3)
                              261F          PRINT   2,<-
                              261F   FILESIZE:      !XL!-!8UL.      -
                              261F   HDLBN:         !XL!-!8UL.>
             7E   58 A3   3C  262C          MOVZWL  FCB$W_UICMEMBER(R3),-(SP)
             7E   5A A3   3C  2630          MOVZWL  FCB$W_UICGROUP(R3),-(SP)
                              2634          PRINT   2,<-
                              2634   UICGROUP:      !OW                   -
                              2634   UICMEMBER:     !OW>
                              2641
                              2641   ;
                              2641   ; Display the file protection !
                              2641   ;
                              2641
             54       0C  D0  2641          MOVL    #12,R4
                              2644
  52   70 A3   04     54  EF  2644   10$:   EXTZV   R4,#4,FCB$W_FILEPROT(R3),R2 ; get the next prot nibble
               52     0F  CC  264A          XORL2   #^XF,R2                     ; complement lower four bits
        7E   E0A6 CF42    D0  264D          MOVL    PRO_TBL[R2],-(SP)           ; push descriptor for this nibble
               54     04  C2  2653          SUBL2   #4,R4                       ; move to next section of word
                      EC  18  2656          BGEQ    10$                         ; if geq then more to go
                              2658
             7E   70 A3   3C  2658   20$:   MOVZWL  FCB$W_FILEPROT(R3),-(SP)
                              265C          PRINT   5,<-
                              265C   FILEPROT:      !XW      System:!AC, Owner:!AC, Group:!AC, World:!AC>
             7E   40 A3   3C  2669          MOVZWL  FCB$W_VERSIONS(R3),-(SP)
                  3C A3   DD  266D          PUSHL   FCB$L_EFBLK(R3)
                              2670          PRINT   2,<-
                              2670   EFBLK:         !XL!-!8UL.      -
                              2670   VERSIONS:      !XW     !-!8UW.>
             7E   42 A3   3C  267D          MOVZWL  FCB$W_DIRSEQ(R3),-(SP)
```

```
                7E    20 A3    3C  2681  2034          MOVZWL  FCB$W_TCNT(R3),-(SP)
                                   2685  2035          PRINT   2,<-
                                   2685  2036 TCNT:             !XW     !-!8UW. -
                                   2685  2037 DIRSEQ:           !XW>
                1A A3    01    B1  2692  2038          CMPW    #1,FCB$W_ACNT(R3)       ; is this file open by more than one?
                         03    19  2696  2039          BLSS    30$                     ; if lss yes
                       00CB    31  2698  2040          BRW     90$                     ; if geq no -- exit
                                   269B  2041
                                   269B  2042 ;
                                   269B  2043 ; Here we will attempt to print out the WCB addresses and associated
                                   2699  2044 ; PID's for all the accessors of this file.
                                   269B  2045 ;
                                   269B  2046
        54    00000F7E'EF    D0  269B  2047 30$:    MOVL    FCB-4,R4                ; get virtual address of FCB
                 54    10    C0  26A2  2048          ADDL2   #FCB$L_WLFL,R4          ; create virt addr of queue head
        52    00000D4E'EF    9E  26A5  2049          MOVAB   WCB,R2
                 62    10 A3    D0  26AC  2050          MOVL    FCB$L_WLFL(R3),WCB$L_WLFL(R2) ; set up starting address 1st WCB
                                   26B0  2051          SKIP    1
                                   26B9  2052          PRINT   0,<-
                                   26B9  2053                  Current Accessors>
                                   26C6  2054          PRINT   0,<-
                                   26C6  2055                  ----------------->
                                   26D3  2056          SKIP    1
                                   26DC  2057          PRINT   0 <-
                                   26DC  2058                  WCB Address             PID>
                                   26E9  2059          PRINT   0,<-
                                   26E9  2060                  -----------             --->
                                   26F6  2061          SKIP    1
                                   26FF  2062
                                   26FF  2063 40$:    GET     WCB,@WCB$L_WLFL(R2),90$
             0B A2    08    93  272C  2064          BITB    #WCB$M_SHRWCB,WCB$B_ACCESS(R2) ; is this shared WCB?
                      19    13  2730  2065          BEQL    50$                     ; if eql no
                                   2732  2066
                                   2732  2067 ;
                                   2732  2068 ; Shared WCB -- print out access count instead of PID
                                   2732  2069 ;
                                   2732  2070
             7E    0E A2    3C  2732  2071          MOVZWL  WCB$W_REFCNT(R2),-(SP)
        00000D4A'EF    DD  2736  2072          PUSHL   WCB-4
                                   273C  2073          PRINT   2,<-
                                   273C  2074          !XL     REFCNT: !XW !-!8UW.>
                      16    11  2749  2075          BRB     60$
                                   274B  2076
                0C A2    DD  274B  2077 50$:    PUSHL   WCB$L_PID(R2)
        00000D4A'EF    DD  274E  2078          PUSHL   WCB-4
                                   2754  2079          PRINT   2,<-
                                   2754  2080          !XL             !XL>
             62    54    D1  2761  2081 60$:    CMPL    R4,WCB$L_WLFL(R2)
                      99    12  2764  2082          BNEQ    40$
                                   2766  2083
                                   2766  2084 90$:    STATUS  SUCCESS
                            04  276D  2085          RET
                                   276E  2086          .DSABL  LSB
```

```
                     276E  2088          .SBTTL  SHOW_IRAB
                     276E  2089
                     276E  2090 ;+++
                     276E  2091 ; SHOW_IRAB -- Show information from IRAB.
                     276E  2092 ;
                     276E  2093 ; Inputs:
                     276E  2094 ;
                     276E  2095 ;       4(AP) = Address of IRAB in buffer in memory.
                     276E  2096 ;
                     276E  2097 ; Outputs:
                     276E  2098 ;
                     276E  2099 ;       None.
                     276E  2100 ;---
                     276E  2101
                     276E  2102          .ENABL  LSB
                     276E  2103 SHOW_IRAB:
                001C 276E  2104          .WORD   ^M<R2,R3,R4>
00000006'EF    04    D3   2770  2105          BITL    #OPT$M_IRB,RMS_DIS_OPT1 ; display IRAB's?
               03    12   2777  2106          BNEQ    1$                      ; if neq yes
             0538    31   2779  2107          BRW     90$                     ; if no -- exit
          53 04 AC   D0   277C  2108 1$:       MOVL    4(AP),R3                ; get address of buffer into R3
                     2780  2109          SKIP    PAGE
          7E 52 A3   3C   2787  2110          MOVZWL  IRB$W_OWN_ISI(R3),-(SP) ; push ISI value for this IRAB
             FC A3   DD   278B  2111          PUSHL   -4(R3)                  ; real address of this IRAB
                     278E  2112          PRINT   2,<-
                     278E  2113 IRAB Address: !XL                ISI: !XW>
                     279B  2114          PRINT   0,<-
                     279B  2115 ----------------                 ---->
                     27A8  2116          SKIP    1
               63    DD   27B1  2117          PUSHL   IRB$L_IFAB_LNK(R3)      ; address of FAB to which this lnkd
                     27B3  2118          PRINT   1,<-
                     27B3  2119 IFAB_LNK:        !XL>
                     27C0  2120          ALLOC   ALLOCSZ,R4                  ; make a buffer to trans BKP bits
             04 A3   DD   27D2  2121          PUSHL   IRB$L_BKPBITS(R3)       ; set of the bits for a call
          DB97 CF    9F   27D5  2122          PUSHAB  IRBBKP_CHR              ; address of translation table
00000000'EF    02    FB   27D9  2123          CALLS   #2,TRANSLATE_BITS
               54    DD   27E0  2124          PUSHL   R4
             04 A3   DD   27E2  2125          PUSHL   IRB$L_BKPBITS(R3)       ; the bit again, this time for display
                     27E5  2126          PRINT   2,<-
                     27E5  2127 BKPBITS:         !XL   !AS>
          7E 08 A3   9A   27F2  2128          MOVZBL  IRB$B_BID(R3),-(SP)     ; block ID
          7E 09 A3   9A   27F6  2129          MOVZBL  IRB$B_BLN(R3),-(SP)     ; block length
                     27FA  2130          PRINT   2,<-
                     27FA  2131 BLN:             !XB     !-!8UB. -
                     27FA  2132 BID:             !XB     !-!8UB.>
          7E 0A A3   9A   2807  2133          MOVZBL  IRB$B_MODE(R3),-(SP)    ; mode
          7E 0B A3   9A   280B  2134          MOVZBL  IRB$B_EFN(R3),-(SP)     ; event flag number
                     280F  2135          PRINT   2,<-
                     280F  2136 EFN:             !XB                         -
                     280F  2137 MODE:            !XB>
             14 A3   DD   281C  2138          PUSHL   IRB$L_ASBADDR(R3)       ; ASB address
             0C A3   DD   281F  2139          PUSHL   IRB$L_IOS(R3)           ; I/O status Block
                     2822  2140          PRINT   2,<-
                     2822  2141 IOS:             !XL             -
                     2822  2142 ASBADDR:        !XL>
             18 A3   DD   282F  2143          PUSHL   IRB$L_ARGLST(R3)        ; User argument list
             10 A3   DD   2832  2144          PUSHL   IRB$L_IOS4(R3)          ; end of I/O status block
```

```
                                                J 13
                        2835  2145            PRINT   2,<-
                        2835  2146 IOS4:               !XL                -
                        2835  2147 ARGLST:              !XL>
                20 A3   DD   2842  2148            PUSHL   IRB$L_CURBDB(R3)        ; current BDB pointer
                1C A3   DD   2845  2149            PUSHL   IRB$L_IRAB_LNK(R3)      ; link to IRAB list
                        2848  2150            PRINT   2,<-
                        2848  2151 IRAB_LNK:           !XL                -
                        2848  2152 CURBDB:             !XL>
                3C A3   DD   2855  2153            PUSHL   IRB$L_NXTBDB(R3)        ; next BDB
                24 A3   DD   2858  2154            PUSHL   IRB$L_LAST_RAB(R3)      ; most recently used RAB in users space
                        285B  2155            PRINT   2,<-
                        285B  2156 LAST_RAB:           !XL                -
                        285B  2157 NXTBDB:             !XL>
                30 A3   DD   2868  2158            PUSHL   IRB$L_JNLBDB(R3)
                38 A3   DD   286B  2159            PUSHL   IRB$L_RLB_LNK(R3)
                        286E  2160            PRINT   2,<-
                        286E  2161 RLB_LNK:            !XL                -
                        286E  2162 JNLBDB:             !XL>
                34 A3   DD   287B  2163            PUSHL   IRB$L_IDENT(R3)
             7E 28 A3   3C   287E  2164            MOVZWL  IRB$W_ISI(R3),-(SP)
                        2882  2165            PRINT   2,<-
                        2882  2166 ISI:               !XW                -
                        2882  2167 IDENT:             !XL>
          6E 00000087   8F   DO   288F  2168            MOVL    #ALLOCSZ,(SP)           ; reset size of allocated buffer
             7E 5D A3   9A   2896  2169            MOVZBL  IRB$B_JNLFLG3(R3),-(SP) ; journal flag3 bits for translate
                E336    CF   289A  2170            PUSHAB  JNLFLG3_CHR
00000000'EF     02     FB   289E  2171            CALLS   #2,TRANSLATE_BITS
                54     DD   28A5  2172            PUSHL   R4
             7E 5D A3   9A   28A7  2173            MOVZBL  IRB$B_JNLFLG3(R3),-(SP) ; journal flag bits
                        28AB  2174            PRINT   2,<-
                        28AB  2175 JNLFLGS3:           !XB        !AS>
                2C A3   DD   28B8  2176            PUSHL   IRB$L_ATJNLBUF(R3)      ; pointer to AT MJB
                        28BB  2177            PRINT   1,<-
                        28BB  2178 ATJNL_PTR:          !XL>
                        28C8  2179
                        28C8  2180 ;
                        28C8  2181 ; Case now for some organization dependant fields
                        28C8  2182 ;
                        28C8  2183
02   00   00000145'EF   8F   28C8  2184            CASEB   IFB$B_ORGCASE+IFAB,#0,#2
                0009'   28D0  2185 2$:      .WORD   10$-2$                  ; Sequential
                0032'   28D2  2186         .WORD   20$-2$                  ; Relative
                0093'   28D4  2187         .WORD   30$-2$                  ; Indexed
                 03E3   31   28D6  2188            BRW     99$                     ; organization error
                        28D9  2189
                        28D9  2190 ;
                        28D9  2191 ; Sequential
                        28D9  2192 ;
                        28D9  2193
                48 A3   DD   28D9  2194 10$:     PUSHL   IRB$L_RP_VBN(R3)        ; VBN from current record pointer
                40 A3   DD   28DC  2195            PUSHL   IRB$L_NRP_VBN(R3)       ; VBN from next record pointer
                        28DF  2196            PRINT   2,<-
                        28DF  2197 NRP_VBN:            !XL!-!8UL.      -
                        28DF  2198 RP_VBN:             !XL!-!8UL.>
                4C A3   DD   28EC  2199            PUSHL   IRB$L_RP_OFF(R3)        ; current record pointer offset
                44 A3   DD   28EF  2200            PUSHL   IRB$L_NRP_OFF(R3)       ; next record pointer offset
                        28F2  2201            PRINT   2,<-
```

```
                        28F2  2202 NRP_OFF:          !XL.-!8UL.      -
                        28F2  2203 RP_OFF:           !XL!-!8UL.>
             0130   31  28FF  2204          BRW     40$
                        2902  2205
                        2902  2206 ;
                        2902  2207 ; Relative
                        2902  2208 ;
                        2902  2209
          48 A3    DD   2902  2210 20$:     PUSHL   IRB$L_RP_VBN(R3)        ; record pointer VBN
          40 A3    DD   2905  2211          PUSHL   IRB$L_NRP_VBN(R3)       ; next record pointer VBN
                        2908  2212          PRINT   2,<-
                        2908  2213 NRP_VBN:          !XL!-!8UL.      -
                        2908  2214 RP_VBN:           !XL!-!8UL.>
          48 A3    DD   2915  2215          PUSHL   IRB$L_RP(R3)
          40 A3    DD   2918  2216          PUSHL   IRB$L_NRP(R3)
                        291B  2217          PRINT   2,<-
                        291B  2218 RP:               !XL!-!8UL.      -
                        291B  2219 NRP:              !XL!-!8UL.>
          4C A3    DD   2928  2220          PUSHL   IRB$L_RP_OFF(R3)
          44 A3    DD   292B  2221          PUSHL   IRB$L_NRP_OFF(R3)
                        292E  2222          PRINT   2,<-
                        292E  2223 NRP_OFF:          !XL!-!8UL.      -
                        292E  2224 RP_OFF:           !XL!-!8UL.>
       7E 4C A3    3C   293B  2225          MOVZWL  IRB$W_RP_OFF(R3),-(SP)
       7E 44 A3    3C   293F  2226          MOVZWL  IRB$W_NRP_OFF(R3),-(SP)
                        2943  2227          PRINT   2,<-
                        2943  2228 NRP_OFF:          !XW     !-!8UW. -
                        2943  2229 RP_OFF:           !XW     !-!8UW.>
          44 A3    DD   2950  2230          PUSHL   IRB$L_CURVBN(R3)
                        2953  2231          PRINT   1,<-
                        2953  2232 CURVBN:           !XL!-!8UL.>
             00CF   31  2960  2233          BRW     40$
                        2963  2234
                        2963  2235 ;
                        2963  2236 ; Indexed
                        2963  2237 ;
                        2963  2238
    6E 00000087 8F    D0   2963  2239 30$:     MOVL    #ALLOCSZ,(SP)
       7E    40 A3    9A   296A  2240          MOVZBL  IRB$B_CACHEFLGS(R3),-(SP)
          DDFE CF    9F   296E  2241          PUSHAB  CSH_CAR
00000000'EF    02    FB   2972  2242          CALLS   #2,TRANSLATE_BITS
                54    DD   2979  2243          PUSHL   R4
       7E    40 A3    9A   297B  2244          MOVZBL  IRB$B_CACHEFLGS(R3),-(SP)
                        297F  2245          PRINT   2,<-
                        297F  2246 CACHEFLGS:        !XB     !AS>
    6E 00000087 8F    D0   298C  2247          MOVL    #ALLOCSZ,(SP)           ; reset buffer size on stack
       7E    42 A3    3C   2993  2248          MOVZWL  IRB$W_SRCHFLAGS(R3),-(SP) ; push bits to translate
          DBB1 CF    9F   2997  2249          PUSHAB  SCH_CAR                 ;  address of translate table
00000000'EF    02    FB   299B  2250          CALLS   #2,TRANSLATE_BITS
                54    DD   29A2  2251          PUSHL   R4                      ; address of text string
       7E    42 A3    3C   29A4  2252          MOVZWL  IRB$W_SRCHFLAGS(R3),-(SP) ; bits again
                        29A8  2253          PRINT   2,<-
                        29A8  2254 SRCHFLAGS:        !XW     !AS>
    6E 00000087 8F    D0   29B5  2255          MOVL    #ALLOCSZ,(SP)           ; reset buffer size on stack
       7E    44 A3    9A   29BC  2256          MOVZBL  IRB$B_SPL_BITS(R3),-(SP) ; bits for translation
          DB30 CF    9F   29C0  2257          PUSHAB  SPL_CAR                 ; address of translation table
00000000'EF    02    FB   29C4  2258          CALLS   #2,TRANSLATE_BITS
```

```
                    54   DD  29CB  2259          PUSHL   R4                              ; buffer full of bit names
                7E  44 A3   9A  29CD  2260          MOVZBL  IRB$B_SPL_BITS(R3),-(SP) ; bit values again
                            29D1  2261          PRINT   2,<-
                            29D1  2262 SPL_BITS:        !XB         !AS>
                7E  41 A3   9A  29DE  2263          MOVZBL  IRB$B_STOPLEVEL(R3),-(SP)
          7E  44 A3 02  01  EF  29E2  2264          EXTZV   #IRB$V_NEW_BKTS,#2,IRB$B_SPL_BITS(R3),-(SP)
                            29E8  2265          PRINT   2,<-
                            29E8  2266 NEWBKTS:         .XB       !-!8UB. -
                            29E8  2267 STOPLEVEL:       .XB       !-!8UB.>
                7E  4A A3   3C  29F5  2268          MOVZWL  IRB$W_SPLIT(R3),-(SP)
                7E  48 A3   3C  29F9  2269          MOVZWL  IRB$W_POS_INS(R3),-(SP)
                            29FD  2270          PRINT   2,<-
                            29FD  2271 POS_INS:         !XW       !-!8UW. -
                            29FD  2272 SPLIT:           !XW       !-!8UW.>
                    4C A3   DD  2A0A  2273          PUSHL   IRB$L_PTR_VBN(R3)
                    4C A3   DD  2A0D  2274          PUSHL   IRB$L_LST_REC(R3)
                            2A10  2275          PRINT   2,<-
                            2A10  2276 LST_REC:         !XL                 -
                            2A10  2277 PRT_VBN:         !XL!-!8UL.>
                7E  4E A3   3C  2A1D  2278          MOVZWL  IRB$W_SPLIT_2(R3),-(SP)
                7E  4C A3   3C  2A21  2279          MOVZWL  IRB$W_SPLIT_1(R3),-(SP)
                            2A25  2280          PRINT   2,<-
                            2A25  2281 SPLIT_1:         !XL!-!8UL.       -
                            2A25  2282 SPLIT_2:         !XL!-!8UL.>
                            2A32  2283
                            2A32  2284 ;
                            2A32  2285 ; Common path again
                            2A32  2286 ;
                            2A32  2287
                7E  54 A3   9A  2A32  2288 40$:     MOVZBL  IRB$B_BCNT(R3),-(SP)     ; buffer count
                    50 A3   DD  2A36  2289          PUSHL   IRB$L_OWNER_ID(R3)       ; full owner ID longword
                            2A39  2290          PRINT   2,<-
                            2A39  2291 OWNER_ID:        !XL                 -
                            2A39  2292 BCNT:            !XB       !-!8UB.>
                7E  55 A3   9A  2A46  2293          MOVZBL  IRB$B_MBC(R3),-(SP)      ; multibuffer count
                7E  52 A3   3C  2A4A  2294          MOVZWL  IRB$W_OWN_ISI(R3),-(SP) ; ISI of this IRAB
                            2A4E  2295          PRINT   2,<-
                            2A4E  2296 OWN_ISI:         !XW       !-!8UW. -
                            2A4E  2297 MBC:             !XB       !-!8UB.>
                7E  50 A3   3C  2A5B  2298          MOVZWL  IRB$W_OWN_ID(R3),-(SP)   ; process ID (low word)
                7E  52 A3   9A  2A5F  2299          MOVZBL  IRB$B_PPF_ISI(R3),-(SP)
                            2A63  2300          PRINT   2,<-
                            2A63  2301 PPF_ISI:         !XB       !-!8UB. -
                            2A63  2302 OWN_ID:          !XW>
                            2A70  2303
                            2A70  2304 ;
                            2A70  2305 ; Case for organization dependant fields
                            2A70  2306 ;
                            2A70  2307
02  00  00000145'EF   8F    2A70  2308          CASEB   IFB$B_ORGCASE+IFAB,#0,#2
                0009' 2A78  2309 41$:     .WORD   50$-41$                          ; sequential
                005E' 2A7A  2310          .WORD   60$-41$                          ; relative
                0072' 2A7C  2311          .WORD   70$-41$                          ; indexed
            023B    31  2A7E  2312          BRW     99$                              ; organization error
                            2A81  2313
                            2A81  2314 ;
                            2A81  2315 ; Sequential
```

```
                        2A81  2316 ;
                        2A81  2317
7E   62 A3    3C        2A81  2318 50$:     MOVZWL  IRB$W_CSIZ(R3),-(SP)
     64 A3    DD        2A85  2319           PUSHL   IRB$L_TEMP0(R3)
                        2A88  2320           PRINT   2,<-
                        2A88  2321 TEMP0:            !XL             -
                        2A88  2322 CSIZ:             !XW     !-!8UW.>
7E   66 A3    3C        2A95  2323           MOVZWL  IRB$W_RTOTLSZ(R3),-(SP)
7E   64 A3    3C        2A99  2324           MOVZWL  IRB$W_ROVHDSZ(R3),-(SP)
                        2A9D  2325           PRINT   2,<-
                        2A9D  2326 ROVHDSZ:          !XW     !-!8UW. -
                        2A9D  2327 RTOTLSZ:          !XW     !-!8UW.>
7E   65 A3    9A        2AAA  2328           MOVZBL  IRB$B_POST_CCTL(R3),-(SP)
7E   64 A3    9A        2AAE  2329           MOVZBL  IRB$B_PRE_CCTL(R3),-(SP)
                        2AB2  2330           PRINT   2,<-
                        2AB2  2331 PRE_CCTL:         !XB     !-!8UB. -
                        2AB2  2332 POST_CCTL:        !XB     !-!8UB.>
7E   68 A3    9A        2ABF  2333           MOVZBL  IRB$B_NVBNS(R3),-(SP)
     68 A3    DD        2AC3  2334           PUSHL   IRB$L_TEMP1(R3)
                        2AC6  2335           PRINT   2,<-
                        2AC6  2336 TEMP1:            !XL             -
                        2AC6  2337 NVBNS:            !XB     !-!8UB.>
      01DE   31         2AD3  2338           BRW     90$
                        2AD6  2339
                        2AD6  2340 ;
                        2AD6  2341 ; Relative
                        2AD6  2342 ;
                        2AD6  2343
7E   62 A3    9A        2AD6  2344 60$:     MOVZBL  IRB$W_CSIZ(R3),-(SP)
                        2ADA  2345           PRINT   1,<-
                        2ADA  2346 CSIZ:             !XB     !-!8UB.>
      01CA   31         2AE7  2347           BRW     90$
                        2AEA  2348
                        2AEA  2349 ;
                        2AEA  2350 ; Indexed
                        2AEA  2351 ;
                        2AEA  2352
     64 A3    DD        2AEA  2353 70$:     PUSHL   IRB$L_UPDBUF(R3)
     60 A3    DD        2AED  2354           PUSHL   IRB$L_KEYBUF(R3)
                        2AF0  2355           PRINT   2,<-
                        2AF0  2356 KEYBUF:           !XL             -
                        2AF0  2357 UPDBUF:           !XL>
     68 A3    DD        2AFD  2358           PUSHL   IRB$L_RECBUF(R3)
     6C A3    DD        2B00  2359           PUSHL   IRB$L_OLDBUF(R3)
                        2B03  2360           PRINT   2,<-
                        2B03  2361 RECBUF:           !XL             -
                        2B03  2362 OLDBUF:           !XL>
     70 A3    DD        2B10  2363           PUSHL   IRB$L_UPD_BDB(R3)
     70 A3    DD        2B13  2364           PUSHL   IRB$L_RFA_VBN(R3)
                        2B16  2365           PRINT   2,<-
                        2B16  2366 RFA_VBN:          !XL!-!8UL.      -
                        2B16  2367 UPD_BDB:          !XL>
     70 A3    DD        2B23  2368           PUSHL   IRB$L_LAST_VBN(R3)
7E   74 A3    3C        2B26  2369           MOVZWL  IRB$W_RFA_ID(R3),-(SP)
                        2B2A  2370           PRINT   2,<-
                        2B2A  2371 RFA_ID:           !XW     !-!8UW. -
                        2B2A  2372 LAST_VBN:         !XL!-!8UL.>
```

```
                7E   74 A3    3C   2B37  2373          MOVZWL  IRB$W_LAST_ID(R3),-(SP)
                7E   76 A3    3C   2B3B  2374          MOVZWL  IRB$W_SAVE_POS(R3),-(SP)
                                   2B3F  2375          PRINT   2,<-
                                   2B3F  2376 SAVE_POS:       !XW     !-!8UW. -
                                   2B3F  2377 LAST_ID:        !XW     !-!8UW.>
                     78 A3    DD   2B4C  2378          PUSHL   IRB$L_PUTUP_VBN(R3)
                     78 A3    DD   2B4F  2379          PUSHL   IRB$L_NEXT_VBN(R3)
                                   2B52  2380          PRINT   2,<-
                                   2B52  2381 NEXT_VBN:       !XL!-!8UL.      -
                                   2B52  2382 PUTUP_VBN:      !XL!-!8UL.>
                7E   0080 C3    3C   2B5F  2383          MOVZWL  IRB$W_PUTUP_ID(R3),-(SP)
                7E   0080 C3    3C   2B64  2384          MOVZWL  IRB$W_NEXT_ID(R3),-(SP)
                                   2B69  2385          PRINT   2,<-
                                   2B69  2386 NEXT_ID:        !XW     !-!8UW. -
                                   2B69  2387 PUTUP_ID:       !XW     !-!8UW.>
                7E   0082 C3    3C   2B76  2388          MOVZWL  IRB$W_FIRST_ID(R3),-(SP)
                     7C A3      DD   2B7B  2389          PUSHL   IRB$L_FIRST_VBN(R3)
                                   2B7E  2390          PRINT   2,<-
                                   2B7E  2391 FIRST_VBN:      !XL!-!8UL.      -
                                   2B7E  2392 FIRST_ID:       !XW     !-!8UW.>
                     0090 C3    DD   2B8B  2393          PUSHL   IRB$L_NEXT_DOWN(R3)
                     0084 C3    DD   2B8F  2394          PUSHL   IRB$L_LOCK_BDB(R3)
                                   2B93  2395          PRINT   2,<-
                                   2B93  2396 LOCK_BDB:       !XL             -
                                   2B93  2397 NEXT_DOWN:      !XL>
                     0088 C3    DD   2BA0  2398          PUSHL   IRB$L_MIDX_TMP1(R3)
                     0088 C3    DD   2BA4  2399          PUSHL   IRB$L_VBN_LEFT(R3)
                                   2BA8  2400          PRINT   2,<-
                                   2BA8  2401 VBN_LEFT:       !XL!-!8UL.      -
                                   2BA8  2402 MIDX_TMP1:      !XL>
                     008C C3    DD   2BB5  2403          PUSHL   IRB$L_MIDX_TMP2(R3)
                     008C C3    DD   2BB9  2404          PUSHL   IRB$L_VBN_RIGHT(R3)
                                   2BBD  2405          PRINT   2,<-
                                   2BBD  2406 VBN_RIGHT:      !XL!-!8UL.      -
                                   2BBD  2407 MIDX_TMP2:      !XL>
                     0090 C3    DD   2BCA  2408          PUSHL   IRB$L_MIDX_TMP3(R3)
                     0090 C3    DD   2BCE  2409          PUSHL   IRB$L_VBN_MID(R3)
                                   2BD2  2410          PRINT   2,<-
                                   2BD2  2411 VBN_MID:        !XL!-!8UL.      -
                                   2BD2  2412 MIDX_TMP3:      !XL>
                     0098 C3    DD   2BDF  2413          PUSHL   IRB$L_LST_NCMP(R3)
                     0094 C3    DD   2BE3  2414          PUSHL   IRB$L_REC_COUNT(R3)
                                   2BE7  2415          PRINT   2,<-
                                   2BE7  2416 REC_COUNT:      !XL!-!8UL.      -
                                   2BE7  2417 LST_NCMP:       !XL>
             7E      00A0 C3    3C   2BF4  2418          MOVZWL  IRB$W_NID_RIGHT(R3),-(SP)
                     009C C3    DD   2BF9  2419          PUSHL   IRB$L_SPL_COUNT(R3)
                                   2BFD  2420          PRINT   2,<-
                                   2BFD  2421 SPL_COUNT:      !XL!-!8UL.      -
                                   2BFD  2422 NID_RIGHT:      !XW     !-!8UW.>
                7E   00A2 C3    3C   2C0A  2423          MOVZWL  IRB$W_NID_MID(R3),-(SP)
                7E   00A4 C3    3C   2C0F  2424          MOVZWL  IRB$W_RFA_NID(R3),-(SP)
                                   2C14  2425          PRINT   2,<-
                                   2C14  2426 RFA_NID:        !XW     !-!8UW. -
                                   2C14  2427 NID_MID:        !XW     !-!8UW.>
                7E   00A6 C3    9A   2C21  2428          MOVZBL  IRB$B_KEYSZ(R3),-(SP)
                                   2C26  2429          PRINT   1,<-
```

```
                        2C26  2430 KEYSZ:          .XB     !-.8UB.>
        00AC C3   DD    2C33  2431          PUSHL   IRB$L_POS_VBN(R3)
        00A8 C3   DD    2C37  2432          PUSHL   IRB$L_CUR_VBN(R3)
                        2C3B  2433          PRINT   2,<-
                        2C3B  2434 CUR_VBN:         !XL!-!8UL.      -
                        2C3B  2435 POS_VBN:         !XL!-!8UL.>
7E      00BA C3   3C    2C48  2436          MOVZWL  IRB$W_POS_ID(R3),-(SP)
7E      00B8 C3   3C    2C4D  2437          MOVZWL  IRB$W_CUR_ID(R3),-(SP)
                        2C52  2438          PRINT   2,<-
                        2C52  2439 CUR_ID:          !XW     !-!8UW. -
                        2C52  2440 POS_ID:          !XW     !-!8UW.>
7E      00C0 C3   3C    2C5F  2441          MOVZWL  IRB$W_CUR_COUNT(R3),-(SP)
                        2C64  2442          PRINT   1,<-
                        2C64  2443 CUR_COUNT:       !XW     !-!8UW.>
        00B4 C3   DD    2C71  2444          PUSHL   IRB$L_SIDR_VBN(R3)
        00B0 C3   DD    2C75  2445          PUSHL   IRB$L_UDR_VBN(R3)
                        2C79  2446          PRINT   2,<-
                        2C79  2447 UDR_VBN:         !XL!-!8UL.      -
                        2C79  2448 SIDR_VBN:        !XL!-!8UL.>
7E      00BE C3   3C    2C86  2449          MOVZWL  IRB$W_SIDR_ID(R3),-(SP)
7E      00BC C3   3C    2C8B  2450          MOVZWL  IRB$W_UDR_ID(R3),-(SP)
                        2C90  2451          PRINT   2,<-
                        2C90  2452 UDR_ID:          !XW     !-!8UW. -
                        2C90  2453 SIDR_ID:         !XW     !-!8UW.>
7E      00C2 C3   9A    2C9D  2454          MOVZBL  IRB$B_RP_KREF(R3),-(SP)
7E      00C3 C3   9A    2CA2  2455          MOVZBL  IRB$B_CUR_KREF(R3),-(SP)
                        2CA7  2456          PRINT   2,<-
                        2CA7  2457 CUR_KREF:        !XB     !-!8UB. -
                        2CA7  2458 RP_KREF:         !XB     !-!8UB.>
                        2CB4  2459 90$:     STATUS  SUCCESS
                  04    2CBB  2460          RET
                        2CBC  2461 99$:     STATUS  NOTVALID
                  04    2CC3  2462          RET
                        2CC4  2463          .DSABL  LSB
```

```
                                2CC4  2465          .SBTTL  SHOW_BDB_SUM
                                2CC4  2466
                                2CC4  2467 ;+++
                                2CC4  2468 ; SHOW_BDB_SUM -- Show one line summaries of important BDB information.
                                2CC4  2469 ;
                                2CC4  2470 ; Inputs:
                                2CC4  2471 ;
                                2CC4  2472 ;       None.
                                2CC4  2473 ;
                                2CC4  2474 ; Implicit Inputs:
                                2CC4  2475 ;
                                2CC4  2476 ;       RMS_DIS_OPT1 = Display options.
                                2CC4  2477 ;       IFAB         = Block buffer with valid (current) IFAB.
                                2CC4  2478 ;
                                2CC4  2479 ; Outputs:
                                2CC4  2480 ;
                                2CC4  2481 ;       None.
                                2CC4  2482 ;
                                2CC4  2483 ; Implicit Outputs:
                                2CC4  2484 ;
                                2CC4  2485 ;       Current BDB buffer destroyed.
                                2CC4  2486 ;---
                                2CC4  2487
                                2CC4  2488          .ENABLE LSB
                                2CC4  2489 SHOW_BDB_SUM:
                         001C   2CC4  2490          .WORD   ^M<R2,R3,R4>
        03 00000006'EF    05    E0  2CC6  2491          BBS     #OPT$V_BDBSUM,RMS_DIS_OPT1,10$ ; branch if summary enabled
                          011B    31  2CCE  2492 5$:     BRW     90$                     ; optimize branches
           52    0000011E'EF    D0  2CD1  2493 10$:     MOVL    IFAB-4,R2               ; virtual address of current IFAB
           52    00000040 8F    C0  2CD8  2494          ADDL2   #IFB$L_BDB_FLNK,R2      ; virtual address of BDB queue header
           00000162'EF    52    D1  2CDF  2495          CMPL    R2,IFB$L_BDB_FLNK+IFAB  ; does queue header point to itself?
                           E6    13  2CE6  2496          BEQL    5$                      ; if eql yes -- empty queue, exit
    000002A6'EF    00000162'EF    D0  2CE8  2497          MOVL    IFB$L_BDB_FLNK+IFAB,BDB ; initialize BDB buffer to point to 1st
                                2CF3  2498          SKIP    PAGE                    ; we're really going to print something
                                2CFA  2499          PRINT   0,<-
                                2CFA  2500                                          BDB Summary>
                                2D07  2501          PRINT   0,<-
                                2D07  2502                                          ---------->
                                2D14  2503          SKIP    1
                                2D1D  2504          PRINT   0,<-
                                2D1D  2505   BDB                                                    CACHE_>
                                2D2A  2506          PRINT   0,<-
                                2D2A  2507 Address USERS SIZE  NUMB    VBN    BLB_PTR    ADDR    VAL    FLGS>
                                2D37  2508          PRINT   0,<-
                                2D37  2509 ------- ----- ----  ----    ---    -------    ----    ---    ---->
                                2D44  2510          SKIP    1
                                2D4D  2511
                                2D4D  2512 ;
                                2D4D  2513 ; Allocate memory for translate bits output.
                                2D4D  2514 ;
                                2D4D  2515
                                2D4D  2516          ALLOC   ALLOCSZ,R4
                                2D5F  2517
                                2D5F  2518 ;
                                2D5F  2519 ; Main loop -- get BDB's from list and display important info, one line
                                2D5F  2520 ; per BDB.
                                2D5F  2521 ;
```

```
                                2D5F  2522
                                2D5F  2523 20$:     GET     BDB,@BDB$L_FLINK+BDB,90$
          6E    00000087 8F     D0  2D8F  2524         MOVL    #ALLOCSZ,(SP)           ; reset size of allocated buffer
                 7E    0A A2    9A  2D96  2525         MOVZBL  BDB$B_FLGS(R2),-(SP)    ; get flags byte for translate
                       D6E2 CF  9F  2D9A  2526         PUSHAB  BDB_CHR                 ; address of bit table for translate
          00000000'EF    02     FB  2D9E  2527         CALLS   #2,TRANSLATE_BITS
                         54     DD  2DA5  2528         PUSHL   R4                      ; push result string for PRINT
                 7E    0B A2    9A  2DA7  2529         MOVZBL  BDB$B_CACHE_VAL(R2),-(SP)
                       18 A2    DD  2DAB  2530         PUSHL   BDB$L_ADDR(R2)
                       10 A2    DD  2DAE  2531         PUSHL   BDB$L_BLB_PTR(R2)
                       1C A2    DD  2DB1  2532         PUSHL   BDB$L_VBN(R2)
                 7E    14 A2    3C  2DB4  2533         MOVZWL  BDB$W_NUMB(R2),-(SP)
                 7E    16 A2    3C  2DB8  2534         MOVZWL  BDB$W_SIZE(R2),-(SP)
                 7E    0C A2    3C  2DBC  2535         MOVZWL  BDB$W_USERS(R2),-(SP)
                                2DC0  2536
                                2DC0  2537 ;
                                2DC0  2538 ; If we have a GBPB instead of a BDB then flag it.
                                2DC0  2539 ;
                                2DC0  2540
                       DE6D CF  DF  2DC0  2541         PUSHAL  SPSTRING                ; assume BDB
                    08 A2    15  91  2DC4  2542         CMPB    #GBPB$C_BID,BDB$B_BID(R2) ; is it GBPB?
                             05  12  2DC8  2543         BNEQ    30$                     ; if neq no
                 6E    DE5A CF  DE  2DCA  2544         MOVAL   GSTRING,(SP)            ; flag as GBPB
                       FC A2    DD  2DCF  2545 30$:     PUSHL   -4(R2)
                                2DD2  2546         PRINT   10,<-
                                2DD2  2547 !XL!AS!3UW !5UW !5UW !7SL !XL !XL !3UB !AS>
          FC A2    00000166'EF  D1  2DDF  2548         CMPL    IFB$L_BDB_BLNK+IFAB,-4(R2) ; done?
                             03  13  2DE7  2549         BEQL    90$                     ; if eql yes
                           FF73  31  2DE9  2550         BRW     20$                     ; if neq then go for more
                                2DEC  2551
                                2DEC  2552 90$:     STATUS  SUCCESS
                             04  2DF3  2553         RET
                                2DF4  2554         .DSABL  LSB
```

```
                                2DF4  2556          .SBTTL  SHOW_BLB_SUM
                                2DF4  2557
                                2DF4  2558 ;+++
                                2DF4  2559 ; SHOW_BLB_SUM -- Show one line summaries of important BLB information.
                                2DF4  2560 ;
                                2DF4  2561 ; Inputs:
                                2DF4  2562 ;
                                2DF4  2563 ;       None.
                                2DF4  2564 ;
                                2DF4  2565 ; Implicit Inputs:
                                2DF4  2566 ;
                                2DF4  2567 ;       RMS_DIS_OPT1 = Display options.
                                2DF4  2568 ;       IFAB         = Block buffer with valid (current) IFAB.
                                2DF4  2569 ;
                                2DF4  2570 ; Outputs:
                                2DF4  2571 ;
                                2DF4  2572 ;       None.
                                2DF4  2573 ;
                                2DF4  2574 ; Implicit Outputs:
                                2DF4  2575 ;
                                2DF4  2576 ;       Current BLB buffer destroyed.
                                2DF4  2577 ;---
                                2DF4  2578
                                2DF4  2579          .ENABLE LSB
                                2DF4  2580 SHOW_BLB_SUM:
                         001C   2DF4  2581          .WORD   ^M<R2,R3,R4>
11 00000122'EF    33     E0   2DF6  2582          BBS     #IFB$V_NORECLK,IFAB,5$  ; if no blb's on this IFAB
   00   00000145'EF    91   2DFE  2583          CMPB    IFB$B_ORGCASE+IFAB,#IFB$C_SEQ ; is file sequential?
                  08    13   2E05  2584          BEQL    5$                      ; if eql yes, exit
03 00000006'EF    10     E0   2E07  2585          BBS     #OPT$V_BLBSUM,RMS_DIS_OPT1,10$ ; branch if summary enabled
                0113    31   2E0F  2586 5$:       BRW     90$                     ; optimize branches
   52   0000011E'EF    D0   2E12  2587 10$:      MOVL    IFAB-4,R2               ; virtual address of current IFAB
   52   00000098 8F    C0   2E19  2588          ADDL2   #IFB$L_BLBFLNK,R2       ; virtual address of BLB queue header
   000001BA'EF    52     D1   2E20  2589          CMPL    R2,IFB$L_BLBFLNK+IFAB   ; does queue header point to itself?
                  E6    13   2E27  2590          BEQL    5$                      ; if eql yes -- empty queue, exit
0000138A'EF    000001BA'EF    D0   2E29  2591          MOVL    IFB$L_BLBFLNK+IFAB,BLB ; initialize BLB buffer to point to 1st
                              2E34  2592          SKIP    PAGE                    ; we're really going to print something
                              2E3B  2593          PRINT   0,<-
                              2E3B  2594                                      BLB Summary>
                              2E48  2595          PRINT   0,<-
                              2E48  2596                                      ---------->
                              2E55  2597          SKIP    1
                              2E5E  2598          PRINT   0,<-
                              2E5E  2599    BLB       BDB             MODE>
                              2E6B  2600          PRINT   0,<-
                              2E6B  2601 Address  Address   OWNER  HELD   VBN   LOCK_ID  VALSEQNO    BLBFLGS>
                              2E78  2602          PRINT   0,<-
                              2E78  2603 -------  -------   -----  ----   ---   -------  --------    ------->
                              2E85  2604          SKIP    1
                              2E8E  2605
                              2E8E  2606 ;
                              2E8E  2607 ; Allocate memory for translate bits output.
                              2E8E  2608 ;
                              2E8E  2609
                              2E8E  2610          ALLOC   ALLOCSZ,R4
                              2EA0  2611
                              2EA0  2612 ;
```

```
                                2EA0  2613 ; Main loop -- get BLB's from list and display important info, one line
                                2EA0  2614 ; per BLB.
                                2EA0  2615 ;
                                2EA0  2616
                                2EA0  2617 20$:     GET     BLB,@BLB$L_FLNK+BLB,90$
      6E   00000087 8F    D0    2ED0  2618         MOVL    #ALLOCSZ,(SP)             ; reset size of allocated buffer
            7E    0A A2   9A    2ED7  2619         MOVZBL  BLB$B_BLBFLGS(R2),-(SP)   ; get flags byte for translate
             D9F1 CF    9F    2EDB  2620         PUSHAB  BLB_CHR                   ; address of bit table for translate
00000000'EF     02    FB    2EDF  2621         CALLS   #2,TRANSLATE_BITS
                54    DD    2EE6  2622         PUSHL   R4                        ; push result string for PRINT
              28 A2   DD    2EE8  2623         PUSHL   BLB$L_VALSEQNO(R2)
              24 A2   DD    2EEB  2624         PUSHL   BLB$L_LOCK_ID(R2)
              14 A2   DD    2EEE  2625         PUSHL   BLB$L_VBN(R2)
           53 0B A2   9A    2EF1  2626         MOVZBL  BLB$B_MODEHELD(R2),R3
              53 06   91    2EF5  2627         CMPB    #6,R3
                 03   1A    2EF8  2628         BGTRU   22$
              53 06   9A    2EFA  2629         MOVZBL  #6,R3
            DA0E CF43 DD    2EFD  2630 22$:     PUSHL   BLB_TBL[R3]
              10 A2   DD    2F02  2631         PUSHL   BLB$L_OWNER(R2)
              0C A2   DD    2F05  2632         PUSHL   BLB$L_BDB_ADDR(R2)
              FC A2   DD    2F08  2633         PUSHL   -4(R2)
                      2F0B  2634         PRINT   8,<-
                      2F0B  2635 !XL !XL !XL !AC !7UL  !XL !XL !AS>
FC A2   000001BE'EF   D1    2F18  2636         CMPL    IFB$L_BLBBLNK+IFAB,-4(R2) ; done?
                 03   13    2F20  2637         BEQL    90$                       ; if eql yes
               FF7B   31    2F22  2638         BRW     20$                       ; if neq then go for more
                      2F25  2639
                      2F25  2640 90$:     STATUS  SUCCESS
                   04 2F2C  2641         RET
                      2F2D  2642         .DSABL  LSB
```

```
                     2F2D  2644          .SBTTL  SHOW_BDB
                     2F2D  2645
                     2F2D  2646 ;+++
                     2F2D  2647 ; SHOW_BDB -- Show information from BDB.
                     2F2D  2648 ;
                     2F2D  2649 ; Inputs:
                     2F2D  2650 ;
                     2F2D  2651 ;       4(AP) = Address of BDB in local memory.
                     2F2D  2652 ;
                     2F2D  2653 ; Outputs:
                     2F2D  2654 ;
                     2F2D  2655 ;       None.
                     2F2D  2656 ;---
                     2F2D  2657
                     2F2D  2658          .ENABL  LSB
                     2F2D  2659 SHOW_BDB:
                001C 2F2D  2660          .WORD   ^M<R2,R3,R4>
     53   04 AC    D0  2F2F  2661          MOVL    4(AP),R3                ; get address of BDB into R3
                     2F33  2662          ENSURE  17
                     2F4B  2663          SKIP    1
          FC A3     DD  2F54  2664          PUSHL   -4(R3)                  ; real address of BDB
  08 A3    15     91  2F57  2665          CMPB    #GBPB$C_BID,BDB$B_BID(R3) ; is it GBPB
           1C     12  2F5B  2666          BNEQ    10$                     ;if neq no
                     2F5D  2667          PRINT   1,<-
                     2F5D  2668 GBPB Address: !XL>
                     2F6A  2669          PRINT   0,<-
                     2F6A  2670 ------------->
           1A     11  2F77  2671          BRB     20$
                     2F79  2672 10$:     PRINT   1,<-
                     2F79  2673 BDB Address: !XL>
                     2F86  2674          PRINT   0,<-
                     2F86  2675 ------------>
                     2F93  2676 20$:     SKIP    1
        7E   08 A3    9A  2F9C  2677          MOVZBL  BDB$B_BID(R3),-(SP)     ; block ID
             63    DD  2FA0  2678          PUSHL   BDB$L_FLINK(R3)         ; forward BDB link
                     2FA2  2679          PRINT   2,<-
                     2FA2  2680 FLINK:           !XL                 -
                     2FA2  2681 BID:             !XB     !-!8UB.>
        7E   09 A3    9A  2FAF  2682          MOVZBL  BDB$B_BLN(R3),-(SP)     ; block length
             04 A3    DD  2FB3  2683          PUSHL   BDB$L_BLINK(R3)         ; back link for BDB chain
                     2FB6  2684          PRINT   2,<-
                     2FB6  2685 BLINK:           !XL                 -
                     2FB6  2686 BLN:             !XB     !-!8UB.>
                     2FC3  2687          ALLOC   ALLOCSZ,R4              ; make some space for translate bits
        7E   0A A3    9A  2FD5  2688          MOVZBL  BDB$B_FLGS(R3),-(SP)    ; push flags byte for translate
           D4A3 CF    9F  2FD9  2689          PUSHAB  BDB_CHR                 ; address of translate table
00000000'EF    02    FB  2FDD  2690          CALLS   #2,TRANSLATE_BITS
             54    DD  2FE4  2691          PUSHL   R4
        7E   0A A3    9A  2FE6  2692          MOVZBL  BDB$B_FLGS(R3),-(SP)    ; flags again for print this time
                     2FEA  2693          PRINT   2,<-
                     2FEA  2694 FLGS:            !XB         !AS>
             10 A3    DD  2FF7  2695          PUSHL   BDB$L_BLB_PTR(R3)
        7E   0C A3    3C  2FFA  2696          MOVZWL  BDB$W_USERS(R3),-(SP)   ; use count
                     2FFE  2697          PRINT   2,<-
                     2FFE  2698 USERS:           !XW     !-!8UW. -
                     2FFE  2699 BLB_PTR:         !XL>
        7E   0E A3    3C  300B  2700          MOVZWL  BDB$W_BUFF_ID(R3),-(SP)
```

```
                7E    0B A3    9A  300F   2701            MOVZBL  BDB$B_CACHE_VAL(R3),-(SP)
                                   3013   2702            PRINT   2,<-
                                   3013   2703 CACHE_VAL:        !XB     !-!8UB. -
                                   3013   2704 BUFF_ID:          !XW     !-!8UW.>
                7E    14 A3    3C  3020   2705            MOVZWL  BDB$W_NUMB(R3),-(SP)    ; number of bytes in use
                7E    16 A3    3C  3024   2706            MOVZWL  BDB$W_SIZE(R3),-(SP)    ; buffer size
                                   3028   2707            PRINT   2,<-
                                   3028   2708 SIZE:            !XW     !-!8UW. -
                                   3028   2709 NUMB:            !XW     !-!8UW.>
                     1C A3    DD  3035   2710            PUSHL   BDB$L_VBN(R3)           ; VBN for this buffer
                     18 A3    DD  3038   2711            PUSHL   BDB$L_ADDR(R3)          ; address of buffer
                                   303B   2712            PRINT   2,<-
                                   303B   2713 ADDR:              !XL             -
                                   303B   2714 VBN:               !XL!-!8UL.>
                08 A3    15  91  3048   2715            CMPB    #GBPB$C_BID,BDB$B_BID(R3) ; is it GBPB
                         16  12  304C   2716            BNEQ    30$                     ; if neq no
                     24 A3    DD  304E   2717            PUSHL   GBPB$L_GBD_PTR(R3)
                     20 A3    DD  3051   2718            PUSHL   GBPB$L_VBNSEQNO(R3)
                                   3054   2719            PRINT   2,<-
                                   3054   2720 VBNSEQNO:         !XL             -
                                   3054   2721 GBD_PTR:          !XL>
                      0089    31  3061   2722            BRW     90$
                                   3064   2723
                     24 A3    DD  3064   2724 30$:       PUSHL   BDB$L_WAIT(R3)
                     20 A3    DD  3067   2725            PUSHL   BDB$L_VBNSEQNO(R3)
                                   306A   2726            PRINT   2,<-
                                   306A   2727 VBNSEQNO:         !XL             -
                                   306A   2728 WAIT:              !XL>
                     4C A3    DD  3077   2729            PUSHL   BDB$L_CURBUFADR(R3)
                     48 A3    DD  307A   2730            PUSHL   BDB$L_WK1(R3)
                                   307D   2731            PRINT   2,<-
                                   307D   2732 WK1:               !XL             -
                                   307D   2733 CURBUFADR:         !XL>
                7E   4A A3    9A  308A   2734            MOVZBL  BDB$B_PRE_CCTL(R3),-(SP)
                7E   48 A3    9A  308E   2735            MOVZBL  BDB$B_REL_VBN(R3),-(SP)
                                   3092   2736            PRINT   2,<-
                                   3092   2737 REL_VBN:          !XB     !-!8UB. -
                                   3092   2738 PRE_CCTL:         !XB>
                7E   4B A3    9A  309F   2739            MOVZBL  BDB$B_POST_CCTL(R3),-(SP)
                7E   49 A3    9A  30A3   2740            MOVZBL  BDB$B_VAL_VBNS(R3),-(SP)
                                   30A7   2741            PRINT   2,<-
                                   30A7   2742 VAL_VBNS:          !XB     !-!8UB. -
                                   30A7   2743 POST_CCTL:         !XB>
                     44 A3    DD  30B4   2744            PUSHL   BDB$T_JNLSEQ+12(R3)     ; journal sequence info
                     40 A3    DD  30B7   2745            PUSHL   BDB$T_JNLSEQ+8(R3)      ; journal sequence info
                     3C A3    DD  30BA   2746            PUSHL   BDB$T_JNLSEQ+4(R3)      ; journal sequence info
                     38 A3    DD  30BD   2747            PUSHL   BDB$T_JNLSEQ+0(R3)      ; journal sequence info
                                   30C0   2748            PRINT   4,<-
                                   30C0   2749 JNLSEQ:            !XL  !XL  !XL  !XL>
                     48 A3    DD  30CD   2750            PUSHL   BDB$L_IOSB(R3)
                                   30D0   2751            PRINT   1,<-
                                   30D0   2752 IOSB:              !XL>
                     4C A3    DD  30DD   2753            PUSHL   BDB$L_IOSB+4(R3)
                                   30E0   2754            PRINT   1,<-
                                   30E0   2755                    !XL>
                                   30ED   2756 90$:       STATUS  SUCCESS
                         04        30F4   2757            RET
```

```
                30F5  2758          .DSABL  LSB
```

```
                        30F5  2760          .SBTTL  SHOW_BLB
                        30F5  2761
                        30F5  2762 ;+++
                        30F5  2763 ; SHOW_BLB -- Show information from BLB.
                        30F5  2764 ;
                        30F5  2765 ; Inputs:
                        30F5  2766 ;
                        30F5  2767 ;       4(AP) = Address of BLB in local memory.
                        30F5  2768 ;
                        30F5  2769 ; Outputs:
                        30F5  2770 ;
                        30F5  2771 ;       None.
                        30F5  2772 ;---
                        30F5  2773
                        30F5  2774          .ENABL  LSB
                        30F5  2775 SHOW_BLB:
                001C    30F5  2776          .WORD   ^M<R2,R3,R4>
53      04 AC     D0    30F7  2777          MOVL    4(AP),R3                ; get address of BLB into R3
                        30FB  2778          ENSURE  16
                        3113  2779          SKIP    1
        FC A3     DD    311C  2780          PUSHL   -4(R3)                  ; real address of BLB
                        311F  2781          PRINT   1,<-
                        311F  2782 BLB Address: !XL>
                        312C  2783          PRINT   0,<-
                        312C  2784 ------------>
                        3139  2785          SKIP    1
7E      08 A3     9A    3142  2786          MOVZBL  BLB$B_BID(R3),-(SP)     ; block ID
          63      DD    3146  2787          PUSHL   BLB$L_FLNK(R3)          ; forward BLB link
                        3148  2788          PRINT   2,<-
                        3148  2789 FLNK:              !XL                 -
                        3148  2790 BID:               !XB     !-!8UB.>
        7E    09 A3     9A    3155  2791          MOVZBL  BLB$B_BLN(R3),-(SP)     ; block length
              04 A3     DD    3159  2792          PUSHL   BLB$L_BLNK(R3)          ; back link for BLB chain
                        315C  2793          PRINT   2,<-
                        315C  2794 BLNK:              !XL                 -
                        315C  2795 BLN:               !XB     !-!8UB.>
                        3169  2796          ALLOC   ALLOCSZ,R4              ; make some space for translate bits
        7E    0A A3     9A    317B  2797          MOVZBL  BLB$B_BLBFLGS(R3), (SP) ; push flags byte for translate
            D74D CF     9F    317F  2798          PUSHAB  BLB_CHR                 ; address of translate table
00000000'EF       02    FB    3183  2799          CALLS   #2,TRANSLATE_BITS
                  54    DD    318A  2800          PUSHL   R4
        7E    0A A3     9A    318C  2801          MOVZBL  BLB$B_BLBFLGS(R3),-(SP) ; flags again for print this time
                        3190  2802          PRINT   2,<-
                        3190  2803 BLBFLGS:          !XB         !AS>
        52    0B A3     9A    319D  2804          MOVZBL  BLB$B_MODEHELD(R3),R2
              52    06  91    31A1  2805          CMPB    #6,R2
                    03  1A    31A4  2806          BGTRU   10$
              52    06  9A    31A6  2807          MOVZBL  #6,R2
            D762  CF42  DD    31A9  2808 10$:     PUSHL   BLB_TBL[R2]
        7E    0B A3     9A    31AE  2809          MOVZBL  BLB$B_MODEHELD(R3),-(SP)
              0C A3     DD    31B2  2810          PUSHL   BLB$L_BDB_ADDR(R3)
                        31B5  2811          PRINT   3,<-
                        31B5  2812 BDB_ADDR:          !XL                 -
                        31B5  2813 MODEHELD:          !XB         !AC>
              10 A3     DD    31C2  2814          PUSHL   BLB$L_OWNER(R3)
              14 A3     DD    31C5  2815          PUSHL   BLB$L_VBN(R3)
                        31C8  2816          PRINT   2,<-
```

```
                        31C8  2817 VBN:            .XL!-.8UL.      -
                        31C8  2818 OWNER:          !XL>
          24 A3   DD    31D5  2819          PUSHL  BLB$L_LOCK_ID(R3)
       7E 20 A3   3C    31D8  2820          MOVZWL BLB$W_LKSTS(R3),-(SP)
                        31DC  2821          PRINT  2,<-
                        31DC  2822 LKSTS:          !XW                     -
                        31DC  2823 LOCK_ID:        !XL>
          18 A3   DD    31E9  2824          PUSHL  BLB$L_RESDSC(R3)
                        31EC  2825          PRINT  1,<-
                        31EC  2826 RESDSC:         !XL>
          1C A3   DD    31F9  2827          PUSHL  BLB$L_RESDSC+4(R3)
                        31FC  2828          PRINT  1,<-
                        31FC  2829 RESDSC+4:       !XL>
          28 A3   DD    3209  2830          PUSHL  BLB$L_VALSEQNO(R3)
          28 A3   DD    320C  2831          PUSHL  BLB$L_VALBLK(R3)
                        320F  2832          PRINT  2,<-
                        320F  2833 VALBLK:         !XL             -
                        320F  2834 VALSEQNO:       !XL!-!8UL.>
          2C A3   DD    321C  2835          PUSHL  BLB$L_VALBLK+4(R3)
                        321F  2836          PRINT  1,<-
                        321F  2837 VALBLK+4:       !XL>
          30 A3   DD    322C  2838          PUSHL  BLB$L_VALBLK+8(R3)
                        322F  2839          PRINT  1,<-
                        322F  2840 VALBLK+8:       !XL>
          34 A3   DD    323C  2841          PUSHL  BLB$L_VALBLK+12(R3)
                        323F  2842          PRINT  1,<-
                        323F  2843 VALBLK+12:      !XL>
                        324C  2844          STATUS SUCCESS
                  04    3253  2845          RET
                        3254  2846          .DSABL LSB
```

```
                     3254  2848          .SBTTL  SHOW_RLB
                     3254  2849
                     3254  2850 ;+++
                     3254  2851 ; SHOW_RLB -- Show information from the RLB.
                     3254  2852 ;
                     3254  2853 ; Inputs:
                     3254  2854 ;
                     3254  2855 ;       4(AP) = Address of buffer containing RLB.
                     3254  2856 ;
                     3254  2857 ; Outputs:
                     3254  2858 ;
                     3254  2859 ;       RLB displayed.
                     3254  2860 ;---
                     3254  2861
                     3254  2862 SHOW_RLB:
               001C  3254  2863          .WORD   ^M<R2,R3,R4>
53   04 AC     D0  3256  2864          MOVL    4(AP),R3                ; get address of block buffer
                     325A  2865          ENSURE  10
                     3272  2866          SKIP    1
     FC A3     DD  327B  2867          PUSHL   -4(R3)                  ; push virtual addresss of RLB
                     327E  2868          PRINT   1,<-
                     327E  2869 RLB Address: !XL>
                     328B  2870          PRINT   0,<-
                     328B  2871 ------------>
                     3298  2872          SKIP    1
7E   08 A3     9A  32A1  2873          MOVZBL  RLB$B_BID(R3),-(SP)
        63     DD  32A5  2874          PUSHL   RLB$L_LNK(R3)
                     32A7  2875          PRINT   2,<-
                     32A7  2876 LNK:             !XL                 -
                     32A7  2877 BID:             !XB     !-!8UB.>
     7E   09 A3     9A  32B4  2878          MOVZBL  RLB$B_BLN(R3),-(SP)
          10 A3     DD  32B8  2879          PUSHL   RLB$L_OWNER(R3)
                     32BB  2880          PRINT   2,<-
                     32BB  2881 OWNER:           !XL     !-!8UL.-
                     32BB  2882 BLN:             !XB     !-!8UB.>
     7E   06 A3     3C  32C8  2883          MOVZWL  RLB$W_RFA4(R3),-(SP)
          0C A3     DD  32CC  2884          PUSHL   RLB$L_RFA0(R3)
                     32CF  2885          PRINT   2,<-
                     32CF  2886 RFA0:            !XL!-!8UL.      -
                     32CF  2887 RFA4:            !XW     !-!8UW.>
     7E   0A A3     9A  32DC  2888          MOVZBL  RLB$B_TMO(R3),-(SP)
                     32E0  2889          PRINT   1,<-
                     32E0  2890 TMO:             !XB     !-!8UB.>
                     32ED  2891          ALLOC   ALLOCSZ,R4
     7E   0B A3     9A  32FF  2892          MOVZBL  RLB$B_FLAGS(R3),-(SP)
        D4B9 CF     9F  3303  2893          PUSHAB  RLB_CHR
00000000'EF     02  FB  3307  2894          CALLS   #2,TRANSLATE_BITS
          54     DD  330E  2895          PUSHL   R4
     7E   0B A3     9A  3310  2896          MOVZBL  RLB$B_FLAGS(R3),-(SP)
                     3314  2897          PRINT   2,<-
                     3314  2898 FLAGS:           !XB     !AS>
                     3321  2899          ALLOC   ALLOCSZ,R4
     7E   04 A3     3C  3333  2900          MOVZWL  RLB$W_FLAGS2(R3),-(SP)
        D4CD CF     9F  3337  2901          PUSHAB  RLB2_CHR
00000000'EF     02  FB  333B  2902          CALLS   #2,TRANSLATE_BITS
          54     DD  3342  2903          PUSHL   R4
     7E   04 A3     3C  3344  2904          MOVZWL  RLB$W_FLAGS2(R3),-(SP)
```

```
                     3348  2905            PRINT   2,<-
                     3348  2906 FLAGS2:             !XW     !AS>
          18 A3   DD 3355  2907            PUSHL   RLB$L_LOCK_ID(R3)
       7E 14 A3   3C 3358  2908            MOVZWL  RLB$W_STATUS(R3),-(SP)
                     335C  2909            PRINT   2,<-
                     335C  2910 STATUS:              !XW                     -
                     335C  2911 LOCK_ID:             !XL>
                     3369  2912            STATUS  SUCCESS
                  04 3370  2913            RET
```

```
                        3371  2915          .SBTTL  SHOW_ASB
                        3371  2916
                        3371  2917 ;+++
                        3371  2918 ; SHOW_ASB -- Show useful information from ASB.
                        3371  2919 ;
                        3371  2920 ; Inputs:
                        3371  2921 ;
                        3371  2922 ;       4(AP) = Address of block buffer.
                        3371  2923 ;
                        3371  2924 ; Outputs:
                        3371  2925 ;
                        3371  2926 ;       ASB Displayed.
                        3371  2927 ;---
                        3371  2928
                        3371  2929          .ENABLE LSB
                   001C 3371  2930 SHOW_ASB::.WORD  ^M<R2,R3,R4>
53   04 AC    D0    3373  2931          MOVL    4(AP),R3                ; get address of ASB
                        3377  2932          SKIP    PAGE
     FC A3    DD    337E  2933          PUSHL   -4(R3)
                        3381  2934          PRINT   1,<-
                        3381  2935 ASB Address: !XL>
                        338E  2936          PRINT   0,<-
                        338E  2937 ------------>
                        339B  2938          SKIP    1
7E   08 A3    9A    33A4  2939          MOVZBL  ASB$B_BID(R3),-(SP)
7E   0C A3    9A    33A8  2940          MOVZBL  ASB$B_ARGCNT(R3),-(SP)
                        33AC  2941          PRINT   2,<-
                        33AC  2942 ARGCNT:          !XB     !-!8UB. -
                        33AC  2943 BID:             !XB     !-!8UB.>
7E   09 A3    9A    33B9  2944          MOVZBL  ASB$B_BLN(R3),-(SP)
     10 A3    DD    33BD  2945          PUSHL   ASB$L_FABRAB(R3)
                        33C0  2946          PRINT   2,<-
                        33C0  2947 FABRAB:          !XL             -
                        33C0  2948 BLN:             !XB     !-!8UB.>
   7E   63    3C    33CD  2949          MOVZWL  ASB$W_STKLEN(R3),-(SP)
     14 A3    DD    33D0  2950          PUSHL   ASB$L_ERR(R3)
                        33D3  2951          PRINT   1,<-
                        33D3  2952 ERR:             !XL             -
                        33D3  2953 STKLEN:          !XW     !-!8UW.>
7E   02 A3    3C    33E0  2954          MOVZWL  ASB$W_STKSIZ(R3),-(SP)
     18 A3    DD    33E4  2955          PUSHL   ASB$L_SUC(R3)
                        33E7  2956          PRINT   1,<-
                        33E7  2957 SUC:             !XL             -
                        33E7  2958 STKSIZ:          !XW     !-!8UW.>
                        33F4  2959          SKIP    1
52   1C A3    DE    33FD  2960          MOVAL   ASB$L_REGS(R3),R2
   7E   82    D0    3401  2961          MOVL    (R2)+,-(SP)
                        3404  2962          PRINT   1,<-
                        3404  2963 R6:            !XL>
   7E   82    D0    3411  2964          MOVL    (R2)+,-(SP)
                        3414  2965          PRINT   1,<-
                        3414  2966 R7:            !XL>
   7E   82    D0    3421  2967          MOVL    (R2)+,-(SP)
                        3424  2968          PRINT   1,<-
                        3424  2969 R8:            !XL>
   7E   82    D0    3431  2970          MOVL    (R2)+,-(SP)
                        3434  2971          PRINT   1,<-
```

```
                                3434  2972 R10:          !XL>
                7E    82   D0   3441  2973            MOVL    (R2)+,-(SP)
                                3444  2974            PRINT   1,<-
                                3444  2975 R11:          !XL>
                  02  A3   B5   3451  2976            TSTW    ASB$W_STKSIZ(R3)        ; is there any stack?
                      46   13   3454  2977            BEQL    90$                     ; if eql no
                                3456  2978            SKIP    1
                                345F  2979            PRINT   0,<-
                                345F  2980 Saved Stack:>
                                346C  2981            PRINT   0,<-
                                346C  2982 ------------>
                                3479  2983            SKIP    1
            52    FC  A3   D0   3482  2984            MOVL    -4(R3),R2
                  30  A2   DF   3486  2985            PUSHAL  ASB$L_STK(R2)           ; this is "stack pointer"
                7E    02  A3   3C   3489  2986            MOVZWL  ASB$W_STKSIZ(R3),-(SP)  ; this is "length of stack space"
          7E    04  AE   6E   C1   348D  2987            ADDL3   (SP),4(SP),-(SP)        ; this is "initial addr of stack"
00000000'EF       03   FB   3492  2988            CALLS   #3,DUMP_STACK           ; dump formatted stack
                      0007 31   3499  2989            BRW     100$
                                349C  2990 90$:        STATUS  SUCCESS
                                34A3  2991 100$:       RET
                                34A4  2992            .DSABL  LSB
```

```
                        34A4  2994          .SBTTL  SHOW_RJB
                        34A4  2995
                        34A4  2996 ;+++
                        34A4  2997 ; SHOW_RJB -- Show information from the RJB.
                        34A4  2998 ;
                        34A4  2999 ; Inputs:
                        34A4  3000 ;
                        34A4  3001 ;       4(AP) = Address of buffer containing RJB.
                        34A4  3002 ;
                        34A4  3003 ; Outputs:
                        34A4  3004 ;
                        34A4  3005 ;       RJB displayed.
                        34A4  3006 ;---
                        34A4  3007
                        34A4  3008 SHOW_RJB:
                  001C  34A4  3009          .WORD   ^M<R2,R3,R4>
53    04 AC      D0   34A6  3010          MOVL    4(AP),R3                ; get address of block buffer
                        34AA  3011          ENSURE  10
                        34C2  3012          SKIP    1
      FC A3        DD   34CB  3013          PUSHL   -4(R3)                  ; push virtual addresss of RJB
                        34CE  3014          PRINT   1,<-
                        34CE  3015 RJB Address: !XL>
                        34DB  3016          PRINT   0,<-
                        34DB  3017 ------------>
                        34E8  3018          SKIP    1
7E    08 A3        9A   34F1  3019          MOVZBL  RJB$B_BID(R3),-(SP)
         63        DD   34F5  3020          PUSHL   RJB$Q_CHAN(R3)
                        34F7  3021          PRINT   2,<-
                        34F7  3022 CHAN:            !XL                 -
                        34F7  3023 BID:             !XB     !-!8UB.>
      7E    09 A3        9A   3504  3024          MOVZBL  RJB$B_BLN(R3),-(SP)
            04 A3        DD   3508  3025          PUSHL   RJB$Q_CHAN+4(R3)
                        350B  3026          PRINT   2,<-
                        350B  3027 CHAN+4:          !XL                 -
                        350B  3028 BLN:             !XB     !-!8UB.>
                        3518  3029          ALLOC   ALLOCSZ,R4
      7E    0A A3        3C   352A  3030          MOVZWL  RJB$W_FLAGS(R3),-(SP)
         D61A CF        9F   352E  3031          PUSHAB  RJB_CHR
00000000'EF          02   FB   3532  3032          CALLS   #2,TRANSLATE_BITS
                  54   DD   3539  3033          PUSHL   R4
      7E    0A A3        3C   353B  3034          MOVZWL  RJB$W_FLAGS(R3),-(SP)
                        353F  3035          PRINT   2,<-
                        353F  3036 FLAGS:           !XW     !AS>
                        354C  3037          STATUS  SUCCESS
                  04    3553  3038          RET
```

```
                        3554  3040          .SBTTL  SHOW_MJB
                        3554  3041 ;+++
                        3554  3042 ; SHOW_MJB -- Show information from the MJB.
                        3554  3043 ;
                        3554  3044 ; Inputs:
                        3554  3045 ;
                        3554  3046 ;       4(AP) = Address of buffer containing MJB.
                        3554  3047 ;
                        3554  3048 ; Outputs:
                        3554  3049 ;
                        3554  3050 ;       MJB displayed.
                        3554  3051 ;---
                        3554  3052
                        3554  3053 SHOW_MJB:
                001C    3554  3054          .WORD   ^M<R2,R3,R4>
53   04 AC    D0        3556  3055          MOVL    4(AP),R3                ; get address of block buffer
                        355A  3056          ENSURE  10
                        3572  3057          SKIP    1
     FC A3    DD        357B  3058          PUSHL   -4(R3)                  ; push virtual addresses of MJB
                        357E  3059          PRINT   1,<-
                        357E  3060 MJB Address:     !XL>
                        358F  3061          PRINT   0,<-
                        358B  3062 ------------>
                        3598  3063          SKIP    1
7E   08 A3    9A        35A1  3064          MOVZBL  MJB$B_BID(R3),-(SP)
7E   0C A3    9A        35A5  3065          MOVZBL  MJB$B_JNL(R3),-(SP)
                        35A9  3066          PRINT   2,<-
                        35A9  3067 JNL:             !XL             -
                        35A9  3068 BID:             !XB     !-!8UB.>
7E   09 A3    9A        35B6  3069          MOVZBL  MJB$B_BLN(R3),-(SP)
     10 A3    DD        35BA  3070          PUSHL   MJB$Q_DESC(R3)
                        35BD  3071          PRINT   2,<-
                        35BD  3072 DESC:            !XL             -
                        35BD  3073 BLN:             !XB     !-!8UB.>
          14 A3    DD   35CA  3074          PUSHL   MJB$Q_DESC+4(R3)
                        35CD  3075          PRINT   1,<-
                        35CD  3076 DESC+4:          !XL             >
          1C A3    DD   35DA  3077          PUSHL   MJB$Q_IOSB+4(R3)
          18 A3    DD   35DD  3078          PUSHL   MJB$Q_IOSB(R3)
                        35E0  3079          PRINT   2,<-
                        35E0  3080 IOSB:            !XL             -
                        35E0  3081 IOSB+4:          !XL             >
                        35ED  3082          ALLOC   ALLOCSZ,R4
        7E   0A A3    3C    35FF  3083          MOVZWL  MJB$W_FLAGS(R3),-(SP)
           D575 CF    9F    3603  3084          PUSHAB  MJB_CHR
00000000'EF   02    FB    3607  3085          CALLS   #2,TRANSLATE_BITS
              54    DD    360E  3086          PUSHL   R4
        7E   0A A3    3C    3610  3087          MOVZWL  MJB$W_FLAGS(R3),-(SP)
                        3614  3088          PRINT   2,<-
                        3614  3089 FLAGS:           !XW     !AS>
                        3621  3090          STATUS  SUCCESS
                  04    3628  3091          RET
```

```
                        3629  3093          .SBTTL  SHOW_GBH
                        3629  3094
                        3629  3095 ;+++
                        3629  3096 ; SHOW_GBH -- Show useful information from Global Buffer Header.
                        3629  3097 ;
                        3629  3098 ;  Inputs:
                        3629  3099 ;
                        3629  3100 ;        4(AP) = Address of block containing GBH.
                        3629  3101 ;
                        3629  3102 ;  Outputs:
                        3629  3103 ;
                        3629  3104 ;       GBH displayed.
                        3629  3105 ;---
                        3629  3106
                   001C 3629  3107 SHOW_GBH:.WORD  ^M<R2,R3,R4>
          53   04 AC    D0 362B  3108          MOVL    4(AP),R3                ; get address of block buffer
                        362F  3109          SKIP    PAGE                    ; format display
               FC A3    DD 3636  3110          PUSHL   -4(R3)                  ; address of GBH
                        3639  3111          PRINT   1,<-
                        3639  3112 GBH Address: !XL>
                        3646  3113          PRINT   0,<-
                        3646  3114 ------------>
                        3653  3115          SKIP    1                       ; one line down
          7E   08 A3    9A 365C  3116          MOVZBL  GBH$B_BID(R3),-(SP)
               63       DD 3660  3117          PUSHL   GBH$L_GBD_FLNK(R3)
                        3662  3118          PRINT   2,<-
                        3662  3119 GBD_FLNK:         !XL                     -
                        3662  3120 BID:              !XB     !-!8UB.>
          7E   09 A3    9A 366F  3121          MOVZBL  GBH$B_BLN(R3),-(SP)
               04 A3    DD 3673  3122          PUSHL   GBH$L_GBD_BLNK(R3)
                        3676  3123          PRINT   2,<-
                        3676  3124 GBD_BLNK:         !XL                     -
                        3676  3125 BLN:              !XB     !-!8UB.>
                        3683  3126          ALLOC   ALLOCSZ,R4
          7E    0A A3    3C 3695  3127          MOVZWL  GBH$W_TRC_FLGS(R3),-(SP)
             D28F CF    9F 3699  3128          PUSHAB  GBH_CAR
00000000'EF    02    FB 369D  3129          CALLS   #2,TRANSLATE_BITS
               54    DD 36A4  3130          PUSHL   R4
          7E    0A A3    3C 36A6  3131          MOVZWL  GBH$W_TRC_FLGS(R3),-(SP)
                        36AA  3132          PRINT   2,<-
                        36AA  3133 FLGS:               !XW         !AS>
               20 A3    DD 36B7  3134          PUSHL   GBH$L_TRC_FLNK(R3)
               28 A3    DD 36BA  3135          PUSHL   GBH$L_GBD_START(R3)
                        36BD  3136          PRINT   2,<-
                        36BD  3137 GBD_START:          !XL                     -
                        36BD  3138 TRC_FLNK:           !XL>
               24 A3    DD 36CA  3139          PUSHL   GBH$L_TRC_BLNK(R3)
               2C A3    DD 36CD  3140          PUSHL   GBH$L_GBD_END(R3)
                        36D0  3141          PRINT   2,<-
                        36D0  3142 GBD_END:            !XL                     -
                        36D0  3143 TRC_BLNK:           !XL>
               34 A3    DD 36DD  3144          PUSHL   GBH$L_SCAN_NUM(R3)
               30 A3    DD 36E0  3145          PUSHL   GBH$L_GBD_NEXT(R3)
                        36E3  3146          PRINT   2,<-
                        36E3  3147 GBD_NEXT:           !XL                     -
                        36E3  3148 SCAN_NUM:           !XL!-!8UL.>
               38 A3    DD 36F0  3149          PUSHL   GBH$L_HIT(R3)
```

```
                40 A3    DD   36F3  3150            PUSHL   GBH$L_READ(R3)
                         36F6  3151                PRINT   2,<-
                         36F6  3152 READ:                  !XL!-!8UL.      -
                         36F6  3153 HIT:                   !XL!-!8UL.>
                3C A3    DD   3703  3154            PUSHL   GBH$L_MISS(R3)
                44 A3    DD   3706  3155            PUSHL   GBH$L_WRITE(R3)
                         3709  3156                PRINT   2,<-
                         3709  3157 WRITE:                 !XL!-!8UL.      -
                         3709  3158 MISS:                  !XL!-!8UL.>
                10 A3    DD   3716  3159            PUSHL   GBH$L_GS_SIZE(R3)
                48 A3    DD   3719  3160            PUSHL   GBH$L_DFW_WRITE(R3)
                         371C  3161                PRINT   1,<-
                         371C  3162 DFW_WRITE:             !XL!-!8UL.      -
                         371C  3163 GS_SIZE:               !XL!-!8UL.>
                14 A3    DD   3729  3164            PUSHL   GBH$L_LOCK_ID(R3)
                1C A3    DD   372C  3165            PUSHL   GBH$L_USECNT(R3)
                         372F  3166                PRINT   2,<-
                         372F  3167 USECNT:                !XL!-!8UL.      -
                         372F  3168 LOCK_ID:               !XL!-!8UL.>
                         373C  3169                STATUS  SUCCESS
                         04    3743  3170            RET
```

```
                                3744  3172          .SBTTL  SHOW_GBD_SUM
                                3744  3173
                                3744  3174 ;+++
                                3744  3175 ; SHOW_GBD_SUM -- Show summary of GBD information.  In this case that's
                                3744  3176 ; all there is in a GBD.
                                3744  3177 ;
                                3744  3178 ;  Inputs:
                                3744  3179 ;
                                3744  3180 ;         None.
                                3744  3181 ;
                                3744  3182 ;  Outputs:
                                3744  3183 ;
                                3744  3184 ;         None.
                                3744  3185 ;--
                                3744  3186
                                3744  3187          .ENABL  LSB
                                3744  3188 SHOW_GBD_SUM:
                         003C   3744  3189          .WORD   ^M<R2,R3,R4,R5>
     03 00000006'EF    15   E0  3746  3190          BBS     #OPT$V_GBDSUM,RMS_DIS_OPT1,10$
                       012C   31  374E  3191 5$:      BRW     90$
        52    000013C6'EF   9E  3751  3192 10$:     MOVAB   GBD,R2                  ; set up R2 with block buff address
     FC A2    000013EE'EF   D0  3758  3193          MOVL    GBH-4,-4(R2)            ; begin to set up for "1st" GBD
                                3760  3194
                                3760  3195          ASSUME  GBH$L_GBD_FLNK EQ 0
                                3760  3196
        62    000013F2'EF   D0  3760  3197          MOVL    GBH,GBD$L_FLINK(R2)     ; set up forward link
                         E5   13  3767  3198          BEQL    5$
                                3769  3199          SKIP    PAGE
                                3770  3200          PRINT   0,<-
                                3770  3201                                  GBD Summary>
                                377D  3202          PRINT   0,<-
                                377D  3203                                  ---------->
                                378A  3204          SKIP    1
                                3793  3205          PRINT   0,<-
                                3793  3206    GBD     USE                           REL_  CACHE_>
                                37A0  3207          PRINT   0,<-
                                37A0  3208 Address  CNT  SIZE  NUMB  VBN  VBNSEQNUM  ADDR   VAL   LOCK_ID FLAGS>
                                37AD  3209          PRINT   0,<-
                                37AD  3210 -------  ---  ----  ----  ----  ---------  ----   ---   ------- ----->
                                37BA  3211          SKIP    1
                                37C3  3212          ALLOC   ALLOCSZ,R4
                           53   D4  37D5  3213          CLRL    R3                      ; R3 = count of GBD's
     55    000013F6'EF         D0  37D7  3214          MOVL    GBH$L_GBD_BLNK+GBH,R5   ; R5 = address of last GBD
     55    000013EE'EF         C0  37DE  3215          ADDL2   GBH-4,R5                ; make absolute address
                                37E5  3216
        62    FC A2         C0  37E5  3217 20$:     ADDL2   -4(R2),GBD$L_FLINK(R2)  ; make rel addr abs addr
                                37E9  3218          GET     GBD,@GBD$L_FLINK(R2),90$
                           53   D6  3816  3219          INCL    R3                      ; count the GBD's found
                                3818  3220
     6E    00000087 8F         D0  3818  3221          MOVL    #ALLOCSZ,(SP)
           7E    0A A2         9A  381F  3222          MOVZBL  GBD$B_FLAGS(R2),-(SP)
              D18D CF         9F  3823  3223          PUSHAB  GBD_CAR
     00000000'EF               FB  3827  3224          CALLS   #2,TRANSLATE_BITS
                           54   DD  382E  3225          PUSHL   R4
                    14 A2       DD  3830  3226          PUSHL   GBD$L_LOCK_ID(R2)
           7E    0B A2         9A  3833  3227          MOVZBL  GBD$B_CACHE_VAL(R2),-(SP)
                 1C A2         DD  3837  3228          PUSHL   GBD$L_REL_ADDR(R2)
```

```
                10 A2    DD   383A  3229          PUSHL   GBD$L_VBNSEQNUM(R2)
                0C A2    DD   383D  3230          PUSHL   GBD$L_VBN(R2)
          7E    18 A2    3C   3840  3231          MOVZWL  GBD$W_NUMB(R2),-(SP)
          7E    1A A2    3C   3844  3232          MOVZWL  GBD$W_SIZE(R2),-(SP)
          7E    20 A2    3C   3848  3233          MOVZWL  GBD$W_USECNT(R2),-(SP)
                FC A2    DD   384C  3234          PUSHL   -4(R2)
                         384F  3235          PRINT   10,<-
                         384F  3236  !XL !3UW !5UW !5UW !7SL !XL !XL !3UB !XL !AS>
          55    FC A2    D1   385C  3237          CMPL    -4(R2),R5               ; is it last GBD?
                   03    13   3860  3238          BEQL    80$                     ; if eql yes
                 FF80    31   3862  3239          BRW     20$                     ; go back for more
                         3865  3240
                   53    DD   3865  3241  80$:    PUSHL   R3                      ; print count of GBD's
                         3867  3242          SKIP    1
                         3870  3243          PRINT   1,<-
                         3870  3244          !8UL. GBD's>
                         387D  3245
                         387D  3246  90$:    STATUS  SUCCESS
                   04    3884  3247          RET
                         3885  3248          .DSABL  LSB
```

```
                        3885  3250          .SBTTL  SHOW_TRACE
                        3885  3251
                        3885  3252 ;+++
                        3885  3253 ; SHOW_TRACE -- Show trace information from trace block pool.
                        3885  3254 ;
                        3885  3255 ; Inputs:
                        3885  3256 ;
                        3885  3257 ;        4(AP) = Address of trace queue header in SDA process
                        3885  3258 ;        8(AP) = Address of trace queue header in dumped process
                        3885  3259 ;       12(AP) = Trace flags (1=yes)
                        3885  3260 ;       16(AP) = Depth to trace (0=all)
                        3885  3261 ;
                        3885  3262 ; Outputs:
                        3885  3263 ;
                        3885  3264 ;       Trace contents dumped to current output file/device.
                        3885  3265 ;---
                        3885  3266
                        3885  3267          .ENABL  LSB
                        3885  3268 SHOW_TRACE:
                  03FC  3885  3269          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9>
          54    04 AC   7D    3887  3270          MOVQ    4(AP),R4                ; get arguments
          56    0C AC   7D    388B  3271          MOVQ    12(AP),R6               ; more arguments
     52   0000145E'EF   DE    388F  3272          MOVAL   TRC,R2                  ; address of TRC block buffer
                        3896  3273
                        3896  3274 ;
                        3896  3275 ; Note well that trace queue is a self relative queue!
                        3896  3276 ;
                        3896  3277 ; R2 = Address of trace block buffer (in SDA)
                        3896  3278 ; R4 = Address of trace queue header (in SDA)
                        3896  3279 ; R5 = Address of trace queue header in process being dumped
                        3896  3280 ; R6 = Trace Flags
                        3896  3281 ; R7 = Depth (0=all)
                        3896  3282 ;
                        3896  3283
04 A2    04 A4    D0    3896  3284 3$:       MOVL    4(R4),TRC$L_BLNK(R2)    ; move q header back link
            03    12    389B  3285          BNEQ    5$                      ; if neq queue is not empty
         027B     31    389D  3286          BRW     90$                     ; if eql empty queue
   FC A2    55    D0    38A0  3287 5$:       MOVL    R5,-4(R2)               ; fake up address of trc block
                        38A4  3288                                          ;  from address of queue header
      62    64    D0    38A4  3289          MOVL    (R4),TRC$L_FLNK(R2)     ; set up to scan queue from front
            57    D5    38A7  3290          TSTL    R7                      ; check trace depth
            47    13    38A9  3291          BEQL    15$                     ; do entire q -- we're already set
   59    04 A4    D0    38AB  3292          MOVL    4(R4),R9                ; begin to form real address of end
      59    55    C0    38AF  3293          ADDL    R5,R9                   ; form real address from self rel one
            57    D6    38B2  3294          INCL    R7                      ; bump depth to account for algorithm
                        38B4  3295
                        38B4  3296 ;
                        38B4  3297 ; At this point, we have a non-zero depth.  We will scan the trace queue
                        38B4  3298 ; from the front until we have passed over the necessary number of queue
                        38B4  3299 ; entries.  No dumping is done at this point, as we are going in the wrong
                        38B4  3300 ; order.  After we have passed over the proper number of queue entries, the
                        38B4  3301 ; we will enter the loop that scans them in the other direction, dumping
                        38B4  3302 ; as it goes.
                        38B4  3303 ;
                        38B4  3304 ; R2 = address of trace queue block buffer ( TRC )
                        38B4  3305 ; R7 = Depth (initially +1)
                        38B4  3306 ; R9 = Address of last trace queue entry (when we get there we're done)
```

```
                        38B4  3307 ;
                        38B4  3308
62   FC A2   C0         38B4  3309 10$:     ADDL    -4(R2),TRC$L_FLNK(R2)    ; make self-rel q absolute address
                        38B8  3310          GET     TRC,@TRC$L_FLNK(R2),90$  ; get next trace queue element
     57   D7            38E5  3311          DECL    R7                       ; count this one in the down count
     09   13            38E7  3312          BEQL    15$                      ; we've done enough
59   FC A2   D1         38E9  3313 12$:     CMPL    -4(R2),R9                ; are we done with all elements?
     C5   12            38ED  3314          BNEQ    10$                      ; if neq no, do some more
    FFA4   31            38EF  3315          BRW     3$                       ; yes, too bad.  Go dump them all.
                        38F2  3316
                        38F2  3317 ;
                        38F2  3318 ; Now we are about to go into the loop which scans from the trace
                        38F2  3319 ; element currently in the TRC block buffer, forward until the
                        38F2  3320 ; beginning of the queue.  The TRC block buffer either has the depth+1'th
                        38F2  3321 ; element in it, or it has enough of the queue header such that
                        38F2  3322 ; scanning forward from it will work.
                        38F2  3323 ;
                        38F2  3324
59   62   D0            38F2  3325 15$:     MOVL    TRC$L_FLNK(R2),R9        ; begin to form address of newest entry
59   FC A2   C0         38F5  3326          ADDL    -4(R2),R9                ; turn it into absolute address
                        38F9  3327
                        38F9  3328 ;
                        38F9  3329 ; Set up constant part of header
                        38F9  3330 ;
                        38F9  3331
                        38F9  3332          SKIP    PAGE
                        3900  3333          PRINT   0,<-
                        3900  3334                                   RMS Trace>
                        390D  3335          PRINT   0,<-
                        390D  3336                                   --------->
                        391A  3337          SKIP    3
                        3923  3338          ALLOC   ALLOCSZ,R4               ; set up translate bits buffer
                        3935  3339          PRINT   0,<-
                        3935  3340            PROC                                                        TRC>
                        3942  3341          PRINT   0,<-
                        3942  3342 OPERATION  IN  STRUCTURE    VBN                          SEQUENCE      ADDRESS>
                        394F  3343          PRINT   0,<-
                        394F  3344 --------- --  ---------    ---                          --------      ------->
                        395C  3345          SKIP 1
                        3965  3346
                        3965  3347 ;
                        3965  3348 ; R2 = pointer to TRC block buffer
                        3965  3349 ; R6 = trace flags
                        3965  3350 ; R9 = pointer to latest entry in the trace queue, after which we should stop
                        3965  3351 ;
                        3965  3352
04 A2   FC A2   C0      3965  3353 20$:     ADDL    -4(R2),TRC$L_BLNK(R2)    ; turn rel into absolute address
                        396A  3354          GET     TRC,@TRC$L_BLNK(R2),90$  ; get next trace block
                        3997  3355
                        3997  3356 ;
                        3997  3357 ; Process this trace block for display information
                        3997  3358 ;
                        3997  3359
56   0A A2   D3         3997  3360          BITL    TRC$W_FUNCTION(R2),R6    ; is this trace block wanted?
          03   12       399B  3361          BNEQ    21$
        0172   31       399D  3362          BRW     50$
6E  00000087 8F   D0    39A0  3363 21$:     MOVL    #ALLOCSZ,(SP)            ; renew translate_bits buffer
```

```
                7E    0A A2    3C   39A7  3364          MOVZWL  TRC$W_FUNCTION(R2),-(SP) ; put bits to translate
                   CF7D CF    9F   39AB  3365          PUSHAB  GBH_CHR                  ; translation table
00000000'EF           02    FB   39AF  3366          CALLS   #2,TRANSLATE_BITS        ; do the translation
                                   39B6  3367
                      FC A2    DD   39B6  3368          PUSHL   -4(R2)                   ; virtual address of trace block
                7E    12 A2    3C   39B9  3369          MOVZWL  TRC$W_SEQNUM(R2),-(SP)   ; sequence number of this trace block
                      14 A2    DD   39BD  3370          PUSHL   TRC$L_VBN(R2)
                      0C A2    DD   39C0  3371          PUSHL   TRC$L_STRUCTURE(R2)
                7E    10 A2    3C   39C3  3372          MOVZWL  TRC$W_PID(R2),-(SP)      ; just process index portion of pid
                         54    DD   39C7  3373          PUSHL   R4                       ; the function type
                                   39C9  3374          PRINT   6,<-
                                   39C9  3375 !9AS !XW !XL !7UL                         !5UW          !XL>
                                   39D6  3376
                                   39D6  3377
                0A A2       05   D3   39D6  3378          BITL    #<GBH$M_CACHE_IN!GBH$M_RLS_IN>,TRC$W_FUNCTION(R2) ; interesting
                         33    13   39DA  3379          BEQL    25$                      ; if eql no, just do return2
                                   39DC  3380
                                   39DC  3381 ;
                                   39DC  3382 ; Dump the argument flags for cache and release
                                   39DC  3383 ;
                                   39DC  3384 ; If not cache or release, just dump return status (ARG_FLG for now)
                                   39DC  3385 ;
                                   39DC  3386
6E         00000087 8F       DO   39DC  3387          MOVL    #ALLOCSZ,(SP)
                      20 A2    DD   39E3  3388          PUSHL   TRC$L_ARG_FLG(R2)
                   CDAE CF    9F   39E6  3389          PUSHAB  RLS_CHR
                05 0A A2       02   E0   39EA  3390          BBS     #GBH$V_RLS_IN,TRC$W_FUNCTION(R2),22$
                   6E   CD7D CF    9E   39EF  3391          MOVAB   CSH_CHR,(SP)
00000000'EF           02    FB   39F4  3392 22$:     CALLS   #2,TRANSLATE_BITS
                                   39FB  3393
                         54    DD   39FB  3394          PUSHL   R4
                      20 A2    DD   39FD  3395          PUSHL   TRC$L_ARG_FLG(R2)
                                   3A00  3396          PRINT   2,<-
                                   3A00  3397                                  ARGFLG=!XL !AS>
                         10    11   3A0D  3398          BRB     30$
                      20 A2    DD   3A0F  3399 25$:     PUSHL   TRC$L_ARG_FLG(R2)
                                   3A12  3400          PRINT   1,<-
                                   3A12  3401                                  R0=!XL>
                                   3A1F  3402
                                   3A1F  3403 ;
                                   3A1F  3404 ; Symbolize the return addresses
                                   3A1F  3405 ;
                                   3A1F  3406
                      18 A2    D5   3A1F  3407 30$:     TSTL    TRC$L_RETURN1(R2)        ; should we dump returns?
                         4A    13   3A22  3408          BEQL    35$                      ; if return1=0 then no.
6E         00000087 8F       DO   3A24  3409          MOVL    #ALLOCSZ,(SP)            ; reset string buffer
                         54    DD   3A2B  3410          PUSHL   R4                       ; address of buffer descriptor
                      18 A2    DD   3A2D  3411          PUSHL   TRC$L_RETURN1(R2)        ; address of return
00000000'EF           02    FB   3A30  3412          CALLS   #2,SYMBOLIZE             ; get string for this symbol
                         51    DD   3A37  3413          PUSHL   R1                       ; address of descriptor
                      18 A2    DD   3A39  3414          PUSHL   TRC$L_RETURN1(R2)        ; address itself
                                   3A3C  3415          PRINT   2,<-
                                   3A3C  3416                                  RETURN=!XL !AS>
6E         00000087 8F       DO   3A49  3417          MOVL    #ALLOCSZ,(SP)
                         54    DD   3A50  3418          PUSHL   R4
                      1C A2    DD   3A52  3419          PUSHL   TRC$L_RETURN2(R2)
00000000'EF           02    FB   3A55  3420          CALLS   #2,SYMBOLIZE
```

```
                51   DD   3A5C  3421          PUSHL   R1
              1C A2  DD   3A5E  3422          PUSHL   TRC$L_RETURN2(R2)
                     3A61  3423          PRINT   2,<-
                     3A61  3424                                  !XL !AS>
                     3A6E  3425
                     3A6E  3426 ;
                     3A6E  3427 ; If this element is proper, then dump the BDB information held within
                     3A6E  3428 ; the element.  Allow TRC$L_BDB_ADDR = 0 to signal no BDB information.
                     3A6E  3429 ;
                     3A6E  3430
              24 A2  D5   3A6E  3431 35$:    TSTL    TRC$L_BDB_ADDR(R2)      ; if the address = 0
                 49  13   3A71  3432          BEQL    40$                     ; if eql yes, skip BDB
              2E A2  DD   3A73  3433          PUSHL   TRC$L_BDB_SEQ(R2)       ; buffer sequence number
           7E 2A A2  3C   3A76  3434          MOVZWL  TRC$W_BDB_BUFF(R2),-(SP)
           7E 2C A2  9A   3A7A  3435          MOVZBL  TRC$B_BDB_CACHE(R2),-(SP)
           7E 28 A2  3C   3A7E  3436          MOVZWL  TRC$W_BDB_USERS(R2),-(SP)
              24 A2  DD   3A82  3437          PUSHL   TRC$L_BDB_ADDR(R2)
                     3A85  3438          PRINT   5,<-
                     3A85  3439                                  BDB=!XL U=!4UW C=!XB B=!XW S=!XL>
                     3A92  3440
                     3A92  3441 ;
                     3A92  3442 ; Interpret and dump flags, if any.
                     3A92  3443 ;
                     3A92  3444
              2D A2  95   3A92  3445          TSTB    TRC$B_BDB_FLAGS(R2)
                 25  13   3A95  3446          BEQL    40$
        6E 00000087 8F   DO   3A97  3447          MOVL    #ALLOCSZ,(SP)
           7E    2D A2  9A   3A9E  3448          MOVZBL  TRC$B_BDB_FLAGS(R2),-(SP)
                 C9DA CF   9F   3AA2  3449          PUSHAB  BDB_CHR
        00000000'EF    02   FB   3AA6  3450          CALLS   #2,TRANSLATE_BITS
                 54   DD   3AAD  3451          PUSHL   R4
                     3AAF  3452          PRINT   1,<-
                     3AAF  3453                                      F=!AS>
                     3ABC  3454
                     3ABC  3455 ;
                     3ABC  3456 ; Print the BLB trace information if the TRC$L_BLB_ADDR doesn't equal zero.
                     3ABC  3457 ;
                     3ABC  3458
              34 A2  D5   3ABC  3459 40$:    TSTL    TRC$L_BLB_ADDR(R2)
                 51  13   3ABF  3460          BEQL    50$
              3C A2  DD   3AC1  3461          PUSHL   TRC$L_BLB_SEQ(R2)       ; blb sequence number
              38 A2  DD   3AC4  3462          PUSHL   TRC$L_BLB_LOCK(R2)
        53    32 A2  9A   3AC7  3463          MOVZBL  TRC$B_BLB_MODE(R2),R3   ; get the mode held byte
           53    06  91   3ACB  3464          CMPB    #6,R3                   ; is it less then 6
                 03  1A   3ACE  3465          BGTRU   42$                     ; if gtru then yes
           53    06  9A   3AD0  3466          MOVZBL  #6,R3                   ; set to 6 (i.e. ??)
         CE38 CF43  DD   3AD3  3467 42$:    PUSHL   BLB_TBL[R3]             ; set translation into args
              34 A2  DD   3AD8  3468          PUSHL   TRC$L_BLB_ADDR(R2)
                     3ADB  3469          PRINT   4,<-
                     3ADB  3470                                      BLB=!XL M=!AC L=!XL    S=!XL>
                     3AE8  3471
                     3AE8  3472 ;
                     3AE8  3473 ; Interpret flags (if any).
                     3AE8  3474 ;
                     3AE8  3475
              33 A2  95   3AE8  3476          TSTB    TRC$B_BLB_FLAGS(R2)
                 25  13   3AEB  3477          BEQL    50$
```

```
6E   00000087 8F   D0   3AED  3478          MOVL    #ALLOCSZ,(SP)
      7E    33 A2   9A   3AF4  3479          MOVZBL  TRC$B_BLB_FLAGS(R2),-(SP)
         CDD4 CF   9F   3AF8  3480          PUSHAB  BLB_CRR
00000000'EF    02   FB   3AFC  3481          CALLS   #2,TRANSLATE_BITS
               54   DD   3B03  3482          PUSHL   R4
                         3B05  3483          PRINT   1,<-
                         3B05  3484                                  F=.AS>
                         3B12  3485 ;
                         3B12  3486 ; End of display, loop for next trace element
                         3B12  3487 ;
                         3B12  3488
     59   FC A2   D1   3B12  3489 50$:      CMPL    -4(R2),R9               ; is this the last element?
             03   13   3B16  3490          BEQL    90$
           FE4A   31   3B18  3491          BRW     20$
                         3B1B  3492
                         3B1B  3493 90$:     STATUS  SUCCESS
                   04   3B22  3494          RET
                         3B23  3495          .DSABL  LSB
```

```
                        3B23  3497          .SBTTL  SHOW_NAM
                        3B23  3498
                        3B23  3499 ;+++
                        3B23  3500 ; SHOW_NAM -- Show valuable information in NAM block.
                        3B23  3501 ;
                        3B23  3502 ; Inputs:
                        3B23  3503 ;
                        3B23  3504 ;       4(AP) = Address of NAM block in 'Block Buffer'.
                        3B23  3505 ;
                        3B23  3506 ; Outputs.
                        3B23  3507 ;
                        3B23  3508 ;       None.
                        3B23  3509 ;---
                        3B23  3510
                        3B23  3511 SHOW_NAM:
                001C    3B23  3512          .WORD   ^M<R2,R3,R4>
    52  04 AC    D0    3B25  3513          MOVL    4(AP),R2                ; Get address of buffer in R2
       14 A2     DF    3B29  3514          PUSHAL  NAM$T_DVI(R2)           ; Push address of Device ID Field
       0C A2     DD    3B2C  3515          PUSHL   NAM$L_ESA(R2)           ; Expanded string address
    7E 0B A2     9A    3B2F  3516          MOVZBL  NAM$B_ESL(R2),-(SP)     ; Expanded string length
       04 A2     DD    3B33  3517          PUSHL   NAM$L_RSA(R2)           ; Resultant string address
    7E 03 A2     9A    3B36  3518          MOVZBL  NAM$B_RSL(R2),-(SP)     ; Resultant string length
                        3B3A  3519          PRINT   5,<-
                        3B3A  3520 Resultant String Name:  !AF!/-
                        3B3A  3521 Expanded String Name:   !AF!/-
                        3B3A  3522 Device Identification:  !AC>
                        3B47  3523          PRINT   0,<-
                        3B47  3524                                          RVN     File #    File Seq. #!/-
                        3B47  3525                                          ---     ------    ----------->
                        3B54  3526          ALLOC   80,R4                   ; Allocate buffer for status bits
          34 A2     DD    3B66  3527          PUSHL   NAM$L_FNB(R2)           ; Set up args to TRANSLATE_BITS
        C55B CF     9F    3B69  3528          PUSHAB  NAM_CHR                 ; Character translation table
00000000'EF     02  FB    3B6D  3529          CALLS   #2,TRANSLATE_BITS
          54    DD    3B74  3530          PUSHL   R4                      ; Send buffer to print routine
          34 A2     DD    3B76  3531          PUSHL   NAM$L_FNB(R2)           ; Send bare bits too
                        3B79  3532          PRINT   2,<-
                        3B79  3533 Status Bits (FNB):      !XL     !AS>
          30 A2     DD    3B86  3534          PUSHL   NAM$L_WCC(R2)           ; Set up wildcard context bits
                        3B89  3535          PRINT   1,<-
                        3B89  3536 Wildcard Context:       !XL>
                  04    3B96  3537          RET
```

```
                     3B97  3539          .SBTTL  GET_IFAB
                     3B97  3540
                     3B97  3541 ;+++
                     3B97  3542 ; GET_IFAB -- Get and Verify IFAB from Dump.
                     3B97  3543 ;
                     3B97  3544 ; Inputs:
                     3B97  3545 ;
                     3B97  3546 ;       4(AP) = Block Buffer.
                     3B97  3547 ;
                     3B97  3548 ; Outputs:
                     3B97  3549 ;
                     3B97  3550 ;       Buffer contains IFAB if success.
                     3B97  3551 ;       If failure, BUFFER-4 = 0.
                     3B97  3552 ;---
                     3B97  3553
                     3B97  3554 GET_IFAB:
              001C   3B97  3555          .WORD   ^M<R2,R3,R4>
53    04 AC    D0   3B99  3556          MOVL    4(AP),R3                ; Get address of buffer
                     3B9D  3557          GETMEM  @-4(R3),(R3),#IFB$C_BLN
     15 50     E9   3BAF  3558          BLBC    R0,100$
08 A3    0B    91   3BB2  3559          CMPB    #IFB$C_BID,IFB$B_BID(R3)
         08    12   3BB6  3560          BNEQ    99$
                     3BB8  3561 90$:     STATUS  SUCCESS
               04    3BBF  3562          RET
                     3BC0  3563 99$:     STATUS  INVBLKTYP
               04    3BC7  3564 100$:    RET
```

```
                     3BC8  3566          .SBTTL  GET_IDX
                     3BC8  3567 ;+++
                     3BC8  3568 ; GET_IDX -- Get and verify Index descriptor.
                     3BC8  3569 ;
                     3BC8  3570 ; Inputs:
                     3BC8  3571 ;
                     3BC8  3572 ;       4(AP) = Address of block buffer.
                     3BC8  3573 ;
                     3BC8  3574 ; Outputs:
                     3BC8  3575 ;
                     3BC8  3576 ;       Buffer contains index descriptor.
                     3BC8  3577 ;---
                     3BC8  3578
             001C    3BC8  3579 GET_IDX:.WORD   ^M<R2,R3,R4>
53    04 AC    D0    3BCA  3580          MOVL    4(AP),R3                ; address of buffer
                     3BCE  3581          GETMEM  @-4(R3),(R3),#IDX$C_FIXED_BLN+32
     15 50     E9    3BE0  3582          BLBC    R0,100$
08 A3  0F     91    3BE3  3583          CMPB    #IDX$C_BID,IDX$B_BID(R3)
       08     12    3BE7  3584          BNEQ    99$
                     3BE9  3585 90$:     STATUS  SUCCESS
            04       3BF0  3586          RET
                     3BF1  3587 99$:     STATUS  INVBLKTYP
            04       3BF8  3588 100$:    RET
```

```
                        3BF9  3590          .SBTTL  GET_FWA
                        3BF9  3591 ;+++
                        3BF9  3592 ; GET_FWA -- Get and verify FWA if any.
                        3BF9  3593 ;
                        3BF9  3594 ;  Inputs:
                        3BF9  3595 ;
                        3BF9  3596 ;       4(AP) = Address of block buffer for FWA.
                        3BF9  3597 ;
                        3BF9  3598 ;  Outputs:
                        3BF9  3599 ;
                        3BF9  3600 ;       Buffer contains FWA.
                        3BF9  3601 ;---
                        3BF9  3602
                001C    3BF9  3603 GET_FWA:.WORD   ^M<R2,R3,R4>
      53    04 AC   D0  3BFB  3604          MOVL    4(AP),R3                ; address of buffer
                        3BFF  3605          GETMEM  @-4(R3),(R3),#FWA$C_BLN
            17 50   E9  3C11  3606          BLBC    R0,100$
5F 8F   05E8 C3     91  3C14  3607          CMPB    FWA$B_UNDER_DEV(R3),#^A'_' ; is there an underline in the
                        3C1A  3608                                          ; right place?
               08   12  3C1A  3609          BNEQ    99$
                        3C1C  3610 90$:     STATUS  SUCCESS
                    04  3C23  3611          RET
                        3C24  3612 99$:     STATUS  INVBLKTYP
                    04  3C2B  3613 100$:    RET
```

```
                                3C2C  3615          .SBTTL  GET_CCB
                                3C2C  3616 ;+++
                                3C2C  3617 ; GET_CCB -- Get and verify Channel Control Block from dump.
                                3C2C  3618 ;
                                3C2C  3619 ; Inputs:
                                3C2C  3620 ;
                                3C2C  3621 ;       4(AP) = Address of block buffer.
                                3C2C  3622 ;
                                3C2C  3623 ; Outputs:
                                3C2C  3624 ;
                                3C2C  3625 ;       Block buffer contains CCB.
                                3C2C  3626 ;---
                                3C2C  3627
                                3C2C  3628 GET_CCB:
                         001C   3C2C  3629          .WORD   ^M<R2,R3,R4>
                   53    04 AC   D0   3C2E  3630          MOVL    4(AP),R3
                                3C32  3631          GETMEM  @CTL$GL_CCBBASE,(R3),#4
             FC A3 63    FC A3   C3   3C42  3632          SUBL3   -4(R3),(R3),-4(R3)
                                3C48  3633          GETMEM  @-4(R3),(R3),#CCB$C_LENGTH
                      13 50     E9   3C56  3634          BLBC    R0,100$
                                3C59  3635          STATUS  INVBLKTYP
                      09 A3     95   3C60  3636          TSTB    CCB$B_AMOD(R3)
                         07     13   3C63  3637          BEQL    100$
                                3C65  3638          STATUS  SUCCESS
                                04   3C6C  3639 100$:    RET
```

```
                     3C6D  3641          .SBTTL  GET_WCB
                     3C6D  3642 ;+++
                     3C6D  3643 ; GET_WCB -- Get and verify Window Contol Block from dump.
                     3C6D  3644 ;
                     3C6D  3645 ; Inputs:
                     3C6D  3646 ;
                     3C6D  3647 ;       4(AP) = Address of block buffer.
                     3C6D  3648 ;
                     3C6D  3649 ; Outputs:
                     3C6D  3650 ;
                     3C6D  3651 ;       Block buffer contains WCB.
                     3C6D  3652 ;---
                     3C6D  3653
                     3C6D  3654 GET_WCB:
                001C 3C6D  3655          .WORD   ^M<R2,R3,R4>
53   04 AC       D0  3C6F  3656          MOVL    4(AP),R3
                     3C73  3657          GETMEM  @-4(R3),(R3),#WCB$C_LENGTH+512
     15 50       E9  3C85  3658          BLBC    R0,100$
0A A3    12      91  3C88  3659          CMPB    #DYN$C_WCB,WCB$B_TYPE(R3)
         08      12  3C8C  3660          BNEQ    99$
                     3C8E  3661 90$:     STATUS  SUCCESS
                 04  3C95  3662          RET
                     3C96  3663 99$:     STATUS  INVBLKTYP
                 04  3C9D  3664 100$:    RET
```

```
                     3C9E  3666          .SBTTL  GET_FCB
                     3C9E  3667 ;+++
                     3C9E  3668 ; GET_FCB -- Get and verify File Control Block from dump.
                     3C9E  3669 ;
                     3C9E  3670 ; Inputs:
                     3C9E  3671 ;
                     3C9E  3672 ;       4(AP) = Address of block buffer.
                     3C9E  3673 ;
                     3C9E  3674 ; Outputs:
                     3C9E  3675 ;
                     3C9E  3676 ;       Block buffer contains FCB.
                     3C9E  3677 ;---
                     3C9E  3678
                     3C9E  3679 GET_FCB:
                001C 3C9E  3680          .WORD   ^M<R2,R3,R4>
53   04 AC    D0     3CA0  3681          MOVL    4(AP),R3
                     3CA4  3682          GETMEM  @-4(R3),(R3),#FCB$C_LENGTH
     15 50    E9     3CB6  3683          BLBC    R0,100$
0A A3    07    91     3CB9  3684          CMPB    #DYN$C_FCB,FCB$B_TYPE(R3)
         08    12     3CBD  3685          BNEQ    99$
                     3CBF  3686 90$:      STATUS  SUCCESS
              04     3CC6  3687          RET
                     3CC7  3688 99$:      STATUS  INVBLKTYP
              04     3CCE  3689 100$:     RET
```

```
                     3CCF  3691          .SBTTL  GET_IRAB
                     3CCF  3692 ;+++
                     3CCF  3693 ; GET_IRAB -- Get and Verify IRAB from Dump.
                     3CCF  3694 ;
                     3CCF  3695 ; Inputs:
                     3CCF  3696 ;
                     3CCF  3697 ;       4(AP) = Address of block buffer for IRAB.
                     3CCF  3698 ;
                     3CCF  3699 ; Outputs:
                     3CCF  3700 ;
                     3CCF  3701 ;       IRAB in block buffer if success, R) contains
                     3CCF  3702 ;       code if failure.
                     3CCF  3703 ;---
                     3CCF  3704
                     3CCF  3705 GET_IRAB:
                001C 3CCF  3706          .WORD   ^M<R2,R3,R4>
          53  04 AC  D0   3CD1  3707          MOVL    4(AP),R3                ; get address of IRAB buffer
                     3CD5  3708          GETMEM  @-4(R3),(R3),#IRB$C_BLN_IDX ; get what is supposed to be IRAB
              1E 50  E9   3CE7  3709          BLBC    R0,100$                 ; if error
          08 A3  0A  91   3CEA  3710          CMPB    #IRB$C_BID,IRB$B_BID(R3) ; is this really IRAB?
                 11  12   3CEE  3711          BNEQ    99$                     ; if ne no, must be imposter
  0000011E'EF    63  D1   3CF0  3712          CMPL    IRB$L_IFAB_LNK(R3),IFAB-4 ; is this correct IRAB for IFAB?
                 08  12   3CF7  3713          BNEQ    99$                     ; if neq, then no.  Error
                     3CF9  3714 90$:     STATUS  SUCCESS
                 04  3D00  3715          RET
                     3D01  3716 99$:     STATUS  INVBLKTYP
                 04  3D08  3717 100$:    RET
                     3D09  3718
```

```
                     3D09  3720          .SBTTL  GET_BDB
                     3D09  3721 ;+++
                     3D09  3722 ; GET_BDB -- Get and verify BDB.
                     3D09  3723 ;
                     3D09  3724 ; Inputs:
                     3D09  3725 ;
                     3D09  3726 ;       4(AP) = Address of buffer to fill with BDB.
                     3D09  3727 ;
                     3D09  3728 ; Ouputs:
                     3D09  3729 ;
                     3D09  3730 ;       None.
                     3D09  3731 ;---
                     3D09  3732
                001C 3D09  3733 GET_BDB:.WORD   ^M<R2,R3,R4>
53   04 AC     D0   3D0B  3734         MOVL    4(AP),R3                ; address of buffer for BDB
                     3D0F  3735         GETMEM  @-4(R3),(R3),#BDB$C_BLN
     1B 50     E9   3D21  3736         BLBC    R0,100$                 ; if error in memory access
08 A3  0C     91   3D24  3737         CMPB    #BDB$C_BID,BDB$B_BID(R3) ; is this valid BDB?
       08     12   3D28  3738         BNEQ    95$
                     3D2A  3739 90$:    STATUS  SUCCESS
            04     3D31  3740         RET
08 A3  15     91   3D32  3741 95$:    CMPB    #GBPB$C_BID,BDB$B_BID(R3) ; is it perchance GBPB?
       F2     13   3D36  3742         BEQL    90$                     ; if eql yes, allow it.
                     3D38  3743 99$:    STATUS  INVBLKTYP
            04     3D3F  3744 100$:   RET
```

```
                        3D40  3746          .SBTTL  GET_BLB
                        3D40  3747 ;+++
                        3D40  3748 ; GET_BLB -- Get and verify BLB.
                        3D40  3749 ;
                        3D40  3750 ; Inputs:
                        3D40  3751 ;
                        3D40  3752 ;       4(AP) = Address of buffer to fill with BLB.
                        3D40  3753 ;
                        3D40  3754 ; Ouputs:
                        3D40  3755 ;
                        3D40  3756 ;       None.
                        3D40  3757 ;---
                        3D40  3758
                001C    3D40  3759 GET_BLB:.WORD   ^M<R2,R3,R4>
53   04 AC      D0      3D42  3760         MOVL    4(AP),R3                ; address of buffer for BLB
                        3D46  3761         GETMEM  @-4(R3),(R3),#BLB$C_BLN
     15 50      E9      3D54  3762         BLBC    R0,100$                 ; if error in memory access
08 A3   10      91      3D57  3763         CMPB    #BLB$C_BID,BLB$B_BID(R3) ; is this valid BLB?
           08   12      3D5B  3764         BNEQ    99$
                        3D5D  3765 90$:    STATUS  SUCCESS
                   04   3D64  3766         RET
                        3D65  3767 99$:    STATUS  INVBLKTYP
                   04   3D6C  3768 100$:   RET
```

```
                     3D6D  3770          .SBTTL  GET_GBD
                     3D6D  3771 ;+++
                     3D6D  3772 ; GET_GBD -- Get and verify GBD.
                     3D6D  3773 ;
                     3D6D  3774 ;  Inputs:
                     3D6D  3775 ;
                     3D6D  3776 ;       4(AP) = Address of buffer to fill with GBD.
                     3D6D  3777 ;
                     3D6D  3778 ;  Ouputs:
                     3D6D  3779 ;
                     3D6D  3780 ;       Valid GBD.
                     3D6D  3781 ;---
                     3D6D  3782
               001C  3D6D  3783 GET_GBD:.WORD   ^M<R2,R3,R4>
53   04 AC     D0   3D6F  3784          MOVL    4(AP),R3                ; address of buffer for GBD
                     3D73  3785          GETMEM  @-4(R3),(R3),#GBD$C_BLN
     15 50     E9   3D81  3786          BLBC    R0,100$                 ; if error in memory access
08 A3    13    91   3D84  3787          CMPB    #GBD$C_BID,GBD$B_BID(R3) ; is this valid GBD?
         08    12   3D88  3788          BNEQ    99$
                     3D8A  3789 90$:     STATUS  SUCCESS
               04    3D91  3790          RET
                     3D92  3791 99$:     STATUS  INVBLKTYP
               04    3D99  3792 100$:    RET
```

```
                        3D9A  3794          .SBTTL  GET_TRC
                        3D9A  3795 ;+++
                        3D9A  3796 ; GET_TRC -- Get and verify TRC.
                        3D9A  3797 ;
                        3D9A  3798 ; Inputs:
                        3D9A  3799 ;
                        3D9A  3800 ;       4(AP) = Address of buffer to fill with TRC.
                        3D9A  3801 ;
                        3D9A  3802 ; Ouputs:
                        3D9A  3803 ;
                        3D9A  3804 ;       None.
                        3D9A  3805 ;---
                        3D9A  3806
                001C    3D9A  3807 GET_TRC:.WORD   ^M<R2,R3,R4>
53   04 AC    D0        3D9C  3808          MOVL    4(AP),R3                    ; address of buffer for TRC
                        3DA0  3809          GETMEM  @-4(R3),(R3),#TRC$C_BLN
     15 50    E9        3DB2  3810          BLBC    R0,100$                     ; if error in memory access
08 A3   12    91        3DB5  3811          CMPB    #TRC$C_BID,TRC$B_BID(R3)    ; is this valid TRC?
           08 12         3DB9  3812          BNEQ    99$
                        3DBB  3813 90$:     STATUS  SUCCESS
                  04    3DC2  3814          RET
                        3DC3  3815 99$:     STATUS  INVBLKTYP
                  04    3DCA  3816 100$:    RET
```

```
                     3DCB  3818          .SBTTL  GET_GBH
                     3DCB  3819 ;+++
                     3DCB  3820 ; GET_GBH -- Get and verify GBH.
                     3DCB  3821 ;
                     3DCB  3822 ; Inputs:
                     3DCB  3823 ;
                     3DCB  3824 ;       4(AP) = Address of buffer to fill with GBH.
                     3DCB  3825 ;
                     3DCB  3826 ; Ouputs:
                     3DCB  3827 ;
                     3DCB  3828 ;       None.
                     3DCB  3829 ;---
                     3DCB  3830
                001C 3DCB  3831 GET_GBH:.WORD   ^M<R2,R3,R4>
53   04 AC     D0   3DCD  3832          MOVL    4(AP),R3                    ; address of buffer for GBH
                     3DD1  3833          GETMEM  @-4(R3),(R3),#GBH$C_BLN
     15 50     E9   3DE3  3834          BLBC    R0,100$                     ; if error in memory access
08 A3  11     91   3DE6  3835          CMPB    #GBH$C_BID,GBH$B_BID(R3) ; is this valid GBH?
        08     12   3DEA  3836          BNEQ    99$
                     3DEC  3837 90$:     STATUS  SUCCESS
               04    3DF3  3838          RET
                     3DF4  3839 99$:     STATUS  INVBLKTYP
               04    3DFB  3840 100$:    RET
```

```
                                                B 1
                     3DFC  3842          .SBTTL  GET_RLB
                     3DFC  3843 ;+++
                     3DFC  3844 ; GET_RLB -- Get an RLB from the dump file.
                     3DFC  3845 ;
                     3DFC  3846 ; Inputs:
                     3DFC  3847 ;
                     3DFC  3848 ;       4(AP) = Address of block buffer for RLB.
                     3DFC  3849 ;
                     3DFC  3850 ; Outputs:
                     3DFC  3851 ;
                     3DFC  3852 ;       RLB in block buffer.
                     3DFC  3853 ;---
                     3DFC  3854
                001C 3DFC  3855 GET_RLB:.WORD   ^M<R2,R3,R4>
53   04 AC    D0     3DFE  3856          MOVL    4(AP),R3                        ; address of block buffer
                     3E02  3857          GETMEM  @-4(R3),(R3),#RLB$C_BLN ; get memory from dump for this rlb
     15 50    E9     3E10  3858          BLBC    R0,100$                         ; if error pass it back up
08 A3    0E   91     3E13  3859          CMPB    #RLB$C_BID,RLB$B_BID(R3) ; is this rlb?
         08   12     3E17  3860          BNEQ    99$                             ; if neq no -- error
                     3E19  3861          STATUS  SUCCESS
              04     3E20  3862          RET
                     3E21  3863 99$:     STATUS  INVBLKTYP
              04     3E28  3864 100$:    RET
                     3E29  3865
```

```
                     3E29  3867          .SBTTL  GET_ASB
                     3E29  3868 ;+++
                     3E29  3869 ; GET_ASB -- Get an ASB from Dump.
                     3E29  3870 ;
                     3E29  3871 ; Inputs:
                     3E29  3872 ;
                     3E29  3873 ;       4(AP) = Address of block buffer for ASB.
                     3E29  3874 ;
                     3E29  3875 ; Outputs:
                     3E29  3876 ;
                     3E29  3877 ;       ASB in block buffer.
                     3E29  3878 ;---
                     3E29  3879
              001C   3E29  3880 GET_ASB:.WORD   ^M<R2,R3,R4>
    53  04 AC     D0  3E2B  3881          MOVL    4(AP),R3
                     3E2F  3882          GETMEM  @-4(R3),(R3),#ASB$C_BLN_IDX
        07 50     E9  3E41  3883          BLBC    R0,100$
                     3E44  3884          STATUS  SUCCESS
                  04 3E4B  3885 100$:    RET
```

```
                     3E4C  3887          .SBTTL  GET_RJB
                     3E4C  3888 ;+++
                     3E4C  3889 ; GET_RJB -- Get an RJB from the dump file.
                     3E4C  3890 ;
                     3E4C  3891 ; Inputs:
                     3E4C  3892 ;
                     3E4C  3893 ;       4(AP) = Address of block buffer for RJB.
                     3E4C  3894 ;
                     3E4C  3895 ; Outputs:
                     3E4C  3896 ;
                     3E4C  3897 ;       RJB in block buffer.
                     3E4C  3898 ;---
                     3E4C  3899
               001C  3E4C  3900 GET_RJB:.WORD   ^M<R2,R3,R4>
    53   04 AC    D0  3E4E  3901          MOVL    4(AP),R3                ; address of block buffer
                     3E52  3902          GETMEM  @-4(R3),(R3),#RJB$C_BLN ; get memory from dump for this RJB
         15 50    E9  3E60  3903          BLBC    R0,100$                 ; if error pass it back up
08 A3      16    91  3E63  3904          CMPB    #RJB$C_BID,RJB$B_BID(R3) ; is this RJB?
           08    12  3E67  3905          BNEQ    99$                     ; if neq no -- error
                     3E69  3906          STATUS  SUCCESS
                 04  3E70  3907          RET
                     3E71  3906 99$:     STATUS  INVBLKTYP
                 04  3E78  3909 100$:    RET
```

```
                     3E79  3911          .SBTTL  GET_MJB
                     3E79  3912
                     3E79  3913 ;+++
                     3E79  3914 ; GET_MJB -- Get an MJB from the dump file.
                     3E79  3915 ;
                     3E79  3916 ; Inputs:
                     3E79  3917 ;
                     3E79  3918 ;       4(AP) = Address of block buffer for MJB.
                     3E79  3919 ;
                     3E79  3920 ; Outputs:
                     3E79  3921 ;
                     3E79  3922 ;       MJB in block buffer.
                     3E79  3923 ;---
                     3E79  3924
                001C 3E79  3925 GET_MJB:.WORD   ^M<R2,R3,R4>
53   04 AC     D0   3E7B  3926         MOVL    4(AP),R3                        ; address of block buffer
                     3E7F  3927         GETMEM  @-4(R3),(R3),#MJB$C_BLN ; get memory from dump for this MJB
     15 50     E9   3E8D  3928         BLBC    R0,100$                         ; if error pass it back up
08 A3    18    91   3E90  3929         CMPB    #MJB$C_BID,MJB$B_BID(R3) ; is this an MJB?
         08    12   3E94  3930         BNEQ    99$                             ; if neq no -- error
                     3E96  3931         STATUS  SUCCESS
               04    3E9D  3932         RET
                     3E9E  3933 99$:     STATUS  INVBLKTYP
               04    3EA5  3934 100$:    RET
                     3EA6  3935
```

```
                                3EA6  3937          .SBTTL  GET_FAB
                                3EA6  3938 ;+++
                                3EA6  3939 ; GET_FAB -- Get and Verify FAB from Dump.
                                3EA6  3940 ;
                                3EA6  3941 ; Inputs:
                                3EA6  3942 ;
                                3EA6  3943 ;       4(AP) = Address of block buffer for FAB.
                                3EA6  3944 ;       IFI   = Contains the IFI of the current IFAB.
                                3EA6  3945 ;
                                3EA6  3946 ; Outputs:
                                3EA6  3947 ;
                                3EA6  3948 ;       FAB in block buffer if success, R0 contains
                                3EA6  3949 ;       code if failure.  BUFFER-4 = 0 if failure.
                                3EA6  3950 ;---
                                3EA6  3951
                                3EA6  3952 GET_FAB:
                         001C   3EA6  3953          .WORD   ^M<R2,R3,R4>
                53   04 AC     D0   3EA8  3954          MOVL    4(AP),R3
                                3EAC  3955          GETMEM  @-4(R3),(R3),#FAB$C_BLN
                     1E 50     E9   3EBE  3956          BLBC    R0,100$
                   63   03     91   3EC1  3957          CMPB    #FAB$C_BID,FAB$B_BID(R3)
                        12     12   3EC4  3958          BNEQ    99$
        02 A3   00000016'EF    B1   3EC6  3959          CMPW    IFI,FAB$W_IFI(R3)       ; Is this the FAB for this IFAB
                        08     12   3ECE  3960          BNEQ    99$
                               3ED0  3961 90$:     STATUS  SUCCESS
                        04     3ED7  3962          RET
                               3ED8  3963 99$:     STATUS  INVBLKTYP
                        04     3EDF  3964 100$:    RET
```

```
                     3EE0  3966          .SBTTL  GET_NAM
                     3EE0  3967 ;+++
                     3EE0  3968 ; GET_NAM -- Get and Verify NAM block from Dump.
                     3EE0  3969 ;
                     3EE0  3970 ; Inputs:
                     3EE0  3971 ;
                     3EE0  3972 ;       4(AP) = Address of block buffer for NAM block.
                     3EE0  3973 ;
                     3EE0  3974 ; Outputs:
                     3EE0  3975 ;
                     3EE0  3976 ;       Block buffer contains NAM block if success,
                     3EE0  3977 ;       BUFFER-4 = 0 if failure, R0 = status.
                     3EE0  3978 ;---
                     3EE0  3979
             001C    3EE0  3980 GET_NAM:.WORD   ^M<R2,R3,R4>
 53   04 AC    D0    3EE2  3981         MOVL    4(AP),R3
                     3EE6  3982         GETMEM  @-4(R3),(R3),#NAM$C_BLN
     14 50    E9    3EF8  3983         BLBC    R0,100$
  63    02    91    3EFB  3984         CMPB    #NAM$C_BID,NAM$B_BID(R3)
        08    12    3EFE  3985         BNEQ    99$
                     3F00  3986 90$:    STATUS  SUCCESS
              04     3F07  3987         RET
                     3F08  3988 99$:    STATUS  INVBLKTYP
              04     3F0F  3989 100$:   RET
                     3F10  3990
                     3F10  3991         .END
```

$$$	=	00000570	R	04	BLB$C_BLN	=	00000038				
ALLOCSZ	=	00000087			BLB$L_BLB_ADDR	=	0000000C				
AMODE_TBL		00000660	R	03	BLB$L_BLNK	=	00000004				
ARGS	=	00000003			BLB$L_FLNK	=	00000000				
ASB		0000108A	R	02	BLB$L_LOCK_ID	=	00000024				
ASB$B_ARGCNT	=	0000000C			BLB$L_OWNER	=	00000010				
ASB$B_BID	=	00000008			BLB$L_RESDSC	=	00000018				
ASB$B_BLN	=	00000009			BLB$L_VALBLK	=	00000028				
ASB$C_BLN_IDX	=	00000200			BLB$L_VALSEQNO	=	00000028				
ASB$L_ERR	=	00000014			BLB$L_VBN	=	00000014				
ASB$L_FABRAB	=	00000010			BLB$V_DFW	=	00000005				
ASB$L_REGS	=	0000001C			BLB$V_IOLOCK	=	00000004				
ASB$L_STK	=	00000030			BLB$V_LOCK	=	00000000				
ASB$L_SUC	=	00000018			BLB$V_NOBUFFER	=	00000003				
ASB$W_STKLEN	=	00000000			BLB$V_NOREAD	=	00000002				
ASB$W_STKSIZ	=	00000002			BLB$V_NOWAIT	=	00000001				
BDB		000002A6	R	02	BLB$V_WRITEBACK	=	00000006				
BDB$B_BID	=	00000008			BLB$W_LKSTS	=	00000020				
BDB$B_BLN	=	00000009			BLB_CHR		000008D0	R	03		
BDB$B_CACHE_VAL	=	0000000B			BLB_TBL		00000910	R	03		
BDB$B_FLGS	=	0000000A			CCB		00000D3A	R	02		
BDB$B_POST_CCTL	=	0000004B			CCB$B_AMOD	=	00000009				
BDB$B_PRE_CCTL	=	0000004A			CCB$B_STS	=	00000008				
BDB$B_REL_VBN	=	00000048			CCB$C_LENGTH	=	00000010				
BDB$B_VAL_VBNS	=	00000049			CCB$L_DIRP	=	0000000C				
BDB$C_BID	=	0000000C			CCB$L_UCB	=	00000000				
BDB$C_BLN	=	00000050			CCB$L_WIND	=	00000004				
BDB$L_ADDR	=	00000018			CCB$V_AMB	=	00000000				
BDB$L_BLB_PTR	=	00000010			CCB$W_IOC	=	0000000A				
BDB$L_BLINK	=	00000004			CCB_CHR		00000650	R	03		
BDB$L_CURBUFADR	=	0000004C			CHF$L_SIG_NAME	=	00000004				
BDB$L_FLINK	=	00000000			CSH$V_LOCK	=	00000000				
BDB$L_IOSB	=	00000048			CSH$V_NOBUFFER	=	00000003				
BDB$L_VBN	=	0000001C			CSH$V_NOREAD	=	00000002				
BDB$L_VBNSEQNO	=	00000020			CSH$V_NOWAIT	=	00000001				
BDB$L_WAIT	=	00000024			CSH_CHR		00000770	R	03		
BDB$L_WK1	=	00000048			CTL$GL_CCBBASE		********	X	03		
BDB$T_JNLSEQ	=	00000038			DDB		00001322	R	02		
BDB$V_AST_DCL	=	00000006			DDB$C_LENGTH	=	00000044				
BDB$V_DRT	=	00000001			DEV$V_ALL	=	00000017				
BDB$V_IOP	=	00000002			DEV$V_AVL	=	00000012				
BDB$V_NOLOCATE	=	00000004			DEV$V_CCL	=	00000001				
BDB$V_PRM	=	00000003			DEV$V_DIR	=	00000003				
BDB$V_VAL	=	00000000			DEV$V_DMT	=	00000015				
BDB$V_WFO	=	00000005			DEV$V_ELG	=	00000016				
BDB$W_BUFF_ID	=	0000000E			DEV$V_FOD	=	0000000E				
BDB$W_NUMB	=	00000014			DEV$V_FOR	=	00000018				
BDB$W_SIZE	=	00000016			DEV$V_GEN	=	00000011				
BDB$W_USERS	=	0000000C			DEV$V_IDV	=	0000001A				
BDB_CHR		00000480	R	03	DEV$V_MBX	=	00000014				
BKP_CHR		00000210	R	03	DEV$V_NET	=	00000005				
BLB		0000138A	R	02	DEV$V_ODV	=	0000001B				
BLB$B_BID	=	00000008			DEV$V_RCK	=	0000001E				
BLB$B_BLBFLGS	=	0000000A			DEV$V_REC	=	00000000				
BLB$B_BLN	=	00000009			DEV$V_RND	=	0000001C				
BLB$B_MODEHELD	=	0000000B			DEV$V_RTM	=	0000001D				
BLB$C_BID	=	00000010			DEV$V_SDI	=	00000004				

```
DEV$V_SHR                = 00000010
DEV$V_SPL                = 00000006
DEV$V_SQD                = 00000005
DEV$V_SWL                = 00000019
DEV$V_TRM                = 00000002
DEV$V_WCK                = 0000001F
DEV_CHR                    00000000 R      03
DIRWC_CHR                  00000AC4 R      03
DIR_CHR                    00000AAC R      03
DUMP_STACK                 ******** X    03
DYN$C_FCB                = 00000007
DYN$C_WCB                = 00000012
FAB                        00000C3A R      02
FAB$B_BID                = 00000000
FAB$C_BID                = 00000003
FAB$C_BLN                = 00000050
FAB$V_BIO                = 00000005
FAB$V_BLK                = 00000003
FAB$V_BRO                = 00000006
FAB$V_CR                 = 00000001
FAB$V_DEL                = 00000002
FAB$V_EXE                = 00000007
FAB$V_FTN                = 00000000
FAB$V_GET                = 00000001
FAB$V_PRN                = 00000002
FAB$V_PUT                = 00000000
FAB$V_TRN                = 00000004
FAB$V_UPD                = 00000003
FAB$W_IFI                = 00000002
FAC_CHR                    000001A0 R      03
FCB                        00000F82 R      02
FCB$B_TYPE               = 0000000A
FCB$C_LENGTH             = 000000B4
FCB$L_EFBLK              = 0000003C
FCB$L_EXFCB              = 0000000C
FCB$L_FCBBL              = 00000004
FCB$L_FCBFL              = 00000000
FCB$L_FILESIZE           = 00000038
FCB$L_HDLBN              = 00000034
FCB$L_STLBN              = 00000030
FCB$L_STVBN              = 0000002C
FCB$L_WLBL               = 00000014
FCB$L_WLFL               = 00000010
FCB$V_BADBLK             = 00000002
FCB$V_DIR                = 00000000
FCB$V_EXCL               = 00000003
FCB$V_MARKDEL            = 00000001
FCB$V_RMSLOCK            = 00000005
FCB$V_SPOOL              = 00000004
FCB$W_ACNT               = 0000001A
FCB$W_DIRSEQ             = 00000042
FCB$W_FID_NUM            = 00000024
FCB$W_FID_RVN            = 00000028
FCB$W_FID_SEQ            = 00000026
FCB$W_FILEPROT           = 00000070
FCB$W_LCNT               = 0000001E
FCB$W_SEGN               = 0000002A
FCB$W_SIZE               = 00000008
FCB$W_STATUS             = 00000022
FCB$W_TCNT               = 00000020
FCB$W_UICGROUP           = 0000005A
FCB$W_UICMEMBER          = 00000058
FCB$W_VERSIONS           = 00000040
FCB$W_WCNT               = 0000001C
FCB_CHR                    00000738 R      03
FLD_CHR                    00000A04 R      03
FWA                        000002FA R      02
FWA$B_BUFFLG             = 0000009C
FWA$B_DIRFLGS            = 00000004
FWA$B_DIRLEN             = 0000002E
FWA$B_DIRTERM            = 0000000A
FWA$B_DIRWCFLGS          = 00000005
FWA$B_FLDFLGS            = 00000001
FWA$B_LEVEL              = 000000C3
FWA$B_LNFLGS             = 00000006
FWA$B_PARSEFLGS          = 00000003
FWA$B_PASSFLGS           = 00000000
FWA$B_ROOTERM            = 0000000B
FWA$B_SLFLGS             = 00000007
FWA$B_SUBNODCNT          = 0000002F
FWA$B_UNDER_DEV          = 000005E8
FWA$B_WILDFLGS           = 00000002
FWA$B_XLTMODE            = 0000009D
FWA$C_BLN                = 0000093C
FWA$L_DEVNODADR          = 00000038
FWA$L_DIRBDB             = 00000030
FWA$L_ESCSTRING          = 0000000C
FWA$L_FWA_PTR            = 00000048
FWA$L_LOOKUP             = 00000034
FWA$L_SLBH_BLINK         = 000000B4
FWA$L_SLBH_FLINK         = 000000B0
FWA$L_SLBH_PTR           = 000000A8
FWA$L_SLB_PTR            = 000000AC
FWA$L_SWB_PTR            = 0000004C
FWA$L_UIC                = 00000028
FWA$Q_AIJNL              = 000008D0
FWA$Q_ATJNL              = 000008D8
FWA$Q_BIJNL              = 000008C8
FWA$Q_CDIR1              = 000000F0
FWA$Q_CDIR2              = 000000F8
FWA$Q_CDIR3              = 00000100
FWA$Q_CDIR4              = 00000108
FWA$Q_CDIR5              = 00000110
FWA$Q_CDIR6              = 00000118
FWA$Q_CDIR7              = 00000120
FWA$Q_CDIR8              = 00000128
FWA$Q_CONCEAL_DEV        = 000000E8
FWA$Q_DEVICE             = 000000E0
FWA$Q_DIR                = 0000003C
FWA$Q_DIR1               = 00000130
FWA$Q_DIR2               = 00000138
FWA$Q_DIR3               = 00000140
FWA$Q_DIR4               = 00000148
FWA$Q_DIR5               = 00000150
```

```
FWA$Q_DIR6              = 00000158
FWA$Q_DIR7              = 00000160
FWA$Q_DIR8              = 00000168
FWA$Q_FIB               = 00000010
FWA$Q_FLAGS             = 00000000
FWA$Q_LOGNAM            = 000000D0
FWA$Q_NAME              = 00000170
FWA$Q_NODE              = 000000D8
FWA$Q_NODE1             = 000001B4
FWA$Q_NODE2             = 000001BC
FWA$Q_NODE3             = 000001C4
FWA$Q_NODE4             = 000001CC
FWA$Q_NODE5             = 000001D4
FWA$Q_NODE6             = 000001DC
FWA$Q_NODE7             = 000001E4
FWA$Q_NODE8             = 000001EC
FWA$Q_RNS               = 00000188
FWA$Q_SHRFIL            = 00000190
FWA$Q_TYPE              = 00000178
FWA$Q_VERSION           = 00000180
FWA$V_CONCEAL_DEV       = 00000039
FWA$V_DEVICE            = 0000000F
FWA$V_DEV_UNDER         = 00000033
FWA$V_DFLT_MFD          = 0000003B
FWA$V_DIR               = 0000000E
FWA$V_DIR1              = 00000020
FWA$V_DIR2              = 00000021
FWA$V_DUPOK             = 00000000
FWA$V_EXP_DEV           = 00000017
FWA$V_EXP_DIR           = 00000016
FWA$V_EXP_NAME          = 00000012
FWA$V_EXP_NODE          = 00000006
FWA$V_EXP_ROOT          = 0000003C
FWA$V_EXP_TYPE          = 00000011
FWA$V_EXP_VER           = 00000010
FWA$V_FILEFOUND         = 00000034
FWA$V_FNA_PASS          = 00000004
FWA$V_GRPMBR            = 0000001B
FWA$V_LOGNAME           = 00000030
FWA$V_NAME              = 0000000D
FWA$V_NAM_DVI           = 00000005
FWA$V_NETSTR            = 00000032
FWA$V_NOCOPY            = 00000001
FWA$V_NODE              = 00000019
FWA$V_OBJTYPE           = 00000031
FWA$V_QUOTED            = 0000001A
FWA$V_REMRESULT         = 00000035
FWA$V_RLF_PASS          = 00000003
FWA$V_ROOT_DIR          = 0000003A
FWA$V_SLPRESENT         = 00000038
FWA$V_SL_PASS           = 00000002
FWA$V_SYNTAX_CHK        = 00000036
FWA$V_TYPE              = 0000000C
FWA$V_VERSION           = 0000000B
FWA$V_WC_NAME           = 00000015
FWA$V_WC_TYPE           = 00000014
FWA$V_WC_VER            = 00000013
FWA$V_WILDCARD          = 00000018
FWA$V_WILD_DIR          = 0000001C
FWA$V_WILD_SFD1         = 00000029
FWA$V_WILD_UFD          = 00000028
FWA$W_ESCIFI            = 0000000E
FWA$W_PRO               = 0000002C
FWA$W_UCHAR             = 00000044
FWA$W_XLTSIZ            = 0000009E
FWADD                     000021F0 R      03
GBD                       00003C6 R       02
GBD$B_BID               = 00000008
GBD$B_CACHE_VAL         = 0000000B
GBD$B_FLAGS             = 0000000A
GBD$C_BID               = 00000013
GBD$C_BLN               = 00000028
GBD$L_FLINK             = 00000000
GBD$L_LOCK_ID           = 00000014
GBD$L_REL_ADDR          = 0000001C
GBD$L_VBN               = 0000000C
GBD$L_VBNSEQNUM         = 00000010
GBD$V_VALID             = 00000000
GBD$W_NUMB              = 00000018
GBD$W_SIZE              = 0000001A
GBD$W_USECNT            = 00000020
GBD_CHR                   000009B4 R      03
GBH                       000013F2 R      02
GBH$B_BID               = 00000008
GBH$B_BLN               = 00000009
GBH$C_BID               = 00000011
GBH$C_BLN               = 00000058
GBH$L_DFW_WRITE         = 00000048
GBH$L_GBD_BLNK          = 00000004
GBH$L_GBD_END           = 0000002C
GBH$L_GBD_FLNK          = 00000000
GBH$L_GBD_NEXT          = 00000030
GBH$L_GBD_START         = 00000028
GBH$L_GS_SIZE           = 00000010
GBH$L_HIT               = 00000038
GBH$L_LOCK_ID           = 00000014
GBH$L_MISS              = 0000003C
GBH$L_READ              = 00000040
GBH$L_SCAN_NUM          = 00000034
GBH$L_TRC_BLNK          = 00000024
GBH$L_TRC_FLNK          = 00000020
GBH$L_USECNT            = 0000001C
GBH$L_WRITE             = 00000044
GBH$M_CACHE_IN          = 00000001
GBH$M_RLS_IN            = 00000004
GBH$V_BLB_BLOCK         = 0000000B
GBH$V_BLB_DEQ           = 0000000A
GBH$V_BLB_ENQ           = 00000008
GBH$V_BLB_GRANT         = 00000009
GBH$V_CACHE_IN          = 00000000
GBH$V_CACHE_OUT         = 00000001
GBH$V_F1                = 0000000C
GBH$V_F2                = 0000000D
GBH$V_F3                = 0000000E
```

```
GBH$V_F4                =  0000000F
GBH$V_QIO_DONE          =  00000005
GBH$V_QIO_START         =  00000004
GBH$V_RLS_IN            =  00000002
GBH$V_RLS_OUT           =  00000003
GBH$V_STALL             =  00000006
GBH$V_THREADGO          =  00000007
GBH$W_TRC_FLGS          =  0000000A
GBH_CHR                    0000092C R      03
GBPB$C_BID              =  00000015
GBPB$L_GBD_PTR          =  00000024
GBPB$L_VBNSEQNO         =  00000020
GETMEM                     ********   X    03
GET_ASB                    00003E29 R      03
GET_BDB                    00003D09 R      03
GET_BLB                    00003D40 R      03
GET_CCB                    00003C2C R      03
GET_FAB                    00003EA6 R      03
GET_FCB                    00003C9E R      03
GET_FWA                    00003BF9 R      03
GET_GBD                    00003D6D R      03
GET_GBH                    00003DCB R      03
GET_IDX                    00003BC8 R      03
GET_IFAB                   00003B97 R      03
GET_IRAB                   00003CCF R      03
GET_MJB                    00003E79 R      03
GET_NAM                    00003EE0 R      03
GET_RJB                    00003E4C R      03
GET_RLB                    00003DFC R      03
GET_TRC                    00003D9A R      03
GET_WCB                    00003C6D R      03
GSTRING                    00000C28 R      03
IDX                        0000103A R      02
IDX$B_BID               =  00000008
IDX$B_BLN               =  00000009
IDX$B_DANUM             =  00000014
IDX$B_DATATYPE          =  0000001D
IDX$B_DATBKTSZ          =  00000017
IDX$B_DATBKTYP          =  00000029
IDX$B_FLAGS             =  0000001C
IDX$B_IANUM             =  00000012
IDX$B_IDXBKTSZ          =  00000016
IDX$B_IDXBKTYP          =  00000028
IDX$B_KEYREF            =  00000021
IDX$B_KEYSZ             =  00000020
IDX$B_LANUM             =  00000013
IDX$B_NULLCHAR          =  0000001F
IDX$B_ROOTLEV           =  00000015
IDX$B_SEGMENTS          =  0000001E
IDX$C_BID               =  0000000F
IDX$C_FIXED_BLN         =  0000002C
IDX$L_IDXFL             =  00000000
IDX$L_ROOTVBN           =  00000018
IDX$V_CHGKEYS           =  00000001
IDX$V_DUPKEYS           =  00000000
IDX$V_IDX_COMPR         =  00000003
IDX$V_INITIDX           =  00000004
IDX$V_KEY_COMPR         =  00000006
IDX$V_NULKEYS           =  00000002
IDX$V_REC_COMPR         =  00000007
IDX$W_DATFILL           =  00000026
IDX$W_IDXFILL           =  00000024
IDX$W_MINRECSZ          =  00000022
IDX$W_POSITION          =  0000002C
IDX_CHR0                   000005E4 R      03
IDX_CHR1                   000005AC R      03
IDX_TBL                    00000614 R      03
IDX_TBL1                   00000634 R      03
IFAB                       00000122 R      02
IFABTBL                    0000001A R      02
IFB$B_AMAX              =  000000B1
IFB$B_AVBN              =  000000B0
IFB$B_BID               =  00000008
IFB$B_BKS               =  0000005E
IFB$B_BLN               =  00000009
IFB$B_EFN               =  0000000B
IFB$B_EXTRABUF          =  000000B6
IFB$B_FAC               =  00000022
IFB$B_FSZ               =  0000005F
IFB$B_JNLFLG            =  000000A0
IFB$B_JNLFLG2           =  000000A2
IFB$B_MODE              =  0000000A
IFB$B_NUM_KEYS          =  000000B2
IFB$B_ORGCASE           =  00000023
IFB$B_PLG_VER           =  000000B7
IFB$B_RAT               =  00000051
IFB$B_RECVRFLGS         =  000000A1
IFB$B_RFMORG            =  00000050
IFB$B_UBUFSZ            =  000000B3
IFB$C_BID               =  0000000B
IFB$C_BLN               =  000000B8
IFB$C_IDX               =  00000002
IFB$C_SEQ               =  00000000
IFB$L_ARGLST            =  00000018
IFB$L_ASBADDR           =  00000014
IFB$L_ASDEVBSIZ         =  00000094
IFB$L_AS_DEV            =  0000008C
IFB$L_BDB_BLNK          =  00000044
IFB$L_BDB_FLNK          =  00000040
IFB$L_BKPBITS           =  00000004
IFB$L_BLBBLNK           =  0000009C
IFB$L_BLBFLNK           =  00000098
IFB$L_DEVBUFSIZ         =  00000048
IFB$L_DVBN              =  000000B0
IFB$L_EBK               =  00000074
IFB$L_EBK_DISK          =  00000058
IFB$L_EXTJNLBUF         =  00000034
IFB$L_FWA_PTR           =  00000038
IFB$L_GBH_PTR           =  00000088
IFB$L_GBSB_PTR          =  0000007C
IFB$L_HBK               =  00000070
IFB$L_HBK_DISK          =  00000054
IFB$L_IDX_PTR           =  000000AC
IFB$L_IOS               =  0000000C
```

```
IFB$L_IOS4                      = 00000010
IFB$L_IRAB_LNK                  = 0000001C
IFB$L_JNLBDB                    = 00000030
IFB$L_LAST_FAB                  = 00000024
IFB$L_LOCK_BDB                  = 0000006C
IFB$L_MRN                       = 000000AC
IFB$L_NWA_PTR                   = 0000003C
IFB$L_PAR_LOCK_ID               = 00000080
IFB$L_RJB                       = 000000A4
IFB$L_RNS_LEN                   = 0C00006C
IFB$L_SFSB_PTR                  = 00000078
IFB$V_ACCESSED                  = 00000025
IFB$V_AI                        = 00000003
IFB$V_AI_RECVR                  = 00000001
IFB$V_ANSI_D                    = 00000026
IFB$V_ASYNC                     = 00000023
IFB$V_ASYNCWAIT                 = 00000024
IFB$V_AT                        = 00000004
IFB$V_BI                        = 00000002
IFB$V_BI_RECVR                  = 00000002
IFB$V_BUSY                      = 00000020
IFB$V_CREATE                    = 00000032
IFB$V_DAP                       = 0000003E
IFB$V_DAP_OPEN                  = 0000003D
IFB$V_DFW                       = 0000002C
IFB$V_DLT                       = 0000002B
IFB$V_DMO                       = 00000028
IFB$V_DONE_ASS_JNL              = 00000004
IFB$V_EOF                       = 00000021
IFB$V_FILEFOUND                 = 0000003C
IFB$V_JNL                       = 00000001
IFB$V_MSE                       = 00000031
IFB$V_NEVER_RU                  = 00000005
IFB$V_NFS                       = 0000002F
IFB$V_NORECLK                   = 00000033
IFB$V_NSP                       = 0000003F
IFB$V_ONLY_RU                   = 00000000
IFB$V_PPF_IMAGE                 = 00000022
IFB$V_PPF_INPUT                 = 0000002E
IFB$V_RESTART                   = 0000003B
IFB$V_RFM                       = 00000000
IFB$V_RMS_STALL                 = 0000003A
IFB$V_RU                        = 00C00001
IFB$V_RUP                       = 00000002
IFB$V_RU_RECVR                  = 00000000
IFB$V_RU_RLK                    = 00000003
IFB$V_RWC                       = 00000027
IFB$V_RW_ATTR                   = 00000034
IFB$V_SCF                       = 0000002A
IFB$V_SEARCH                    = 00000039
IFB$V_SEQFIL                    = 00000038
IFB$V_SPL                       = 00000029
IFB$V_SQO                       = 0000002D
IFB$V_STALL_LOCK                = 00000037
IFB$V_TEF                       = 00000036
IFB$V_TMP                       = 00000035
IFB$V_VALID_AT                  = 00000000
IFB$V_WRTACC                    = 00000030
IFB$W_AVGBPB                    = 00000086
IFB$W_AVLCL                     = 00000084
IFB$W_CHNL                      = 00000020
IFB$W_DEQ                       = 00000062
IFB$W_ECHO_ISI                  = 0000002A
IFB$W_FFB                       = 0000005C
IFB$W_GBC                       = 00000064
IFB$W_IFI                       = 00000028
IFB$W_KBUFSZ                    = 000000B4
IFB$W_LRL                       = 00000052
IFB$W_MRS                       = 00000060
IFB$W_RTDEQ                     = 0000004C
IFBS                            = 00000018  R    02
IFBTBLSIZ                       = 0000011A  R    02
IFI                             = 00000016  R    02
IMP$C_ENTPERSEG                 = 0000000F
IMP$C_NPIOFILES                 = 0000003F
IMP$L_IFABTBL                   = 00000013
IMP$W_ENTPERSEG                 = 00000020
IRAB                            = 000001DE  R    02
IRB$B_BCNT                      = 00000054
IRB$B_BID                       = 00000008
IRB$B_BLN                       = 00000009
IRB$B_CACHEFLGS                 = 00000040
IRB$B_CUR_KREF                  = 000000C3
IRB$B_EFN                       = 0000000B
IRB$B_JNLFLG3                   = 0000005D
IRB$B_KEYSZ                     = 000000A6
IRB$B_MBC                       = 00000055
IRB$B_MODE                      = 0000000A
IRB$B_NVBNS                     = 00000068
IRB$B_POST_CCTL                 = 00000065
IRB$B_PPF_ISI                   = 00000052
IRB$B_PRE_CCTL                  = 00000064
IRB$B_RP_KREF                   = 000000C2
IRB$B_SPC_BITS                  = 00000044
IRB$B_STOPLEVEL                 = 00000041
IRB$C_BID                       = 0000000A
IRB$C_BLN_IDX                   = 000000C4
IRB$L_ARGLST                    = 00000018
IRB$L_ASBADDR                   = 00000014
IRB$L_ATJNLBUF                  = 0000002C
IRB$L_BKPBITS                   = 00000004
IRB$L_CURBDB                    = 00000020
IRB$L_CURVBN                    = 00000044
IRB$L_CUR_VBN                   = 000000A8
IRB$L_FIRST_VBN                 = 0000007C
IRB$L_IDENT                     = 00000034
IRB$L_IFAB_LNK                  = 00000000
IRB$L_IOS                       = 0000000C
IRB$L_IOS4                      = 00000010
IRB$L_IRAB_LNK                  = 0000001C
IRB$L_JNLBDB                    = 00000030
IRB$L_KEYBUF                    = 00000060
IRB$L_LAST_RAB                  = 00000024
IRB$L_LAST_VBN                  = 00000070
```

```
IRB$L_LOCK_BDB          = 00000084      IRB$V_GBLBUFF           = 00000036
IRB$L_LST_NCMP          = 00000098      IRB$V_IDX_ERR           = 00000031
IRB$L_LST_REC           = 0000004C      IRB$V_LAST_GT           = 0000000A
IRB$L_MIDX_TMP1         = 00000088      IRB$V_NEW_BKTS          = 00000001
IRB$L_MIDX_TMP2         = 0000008C      IRB$V_NEW_IDX           = 00000003
IRB$L_MIDX_TMP3         = 00000090      IRB$V_NORLS_RNF         = 00000005
IRB$L_NEXT_DOWN         = 00000090      IRB$V_NO_Q_WAIT         = 00000038
IRB$L_NEXT_VBN          = 00000078      IRB$V_POSDELETE         = 00000002
IRB$L_NRP               = 00000040      IRB$V_POSINSERT         = 00000000
IRB$L_NRP_OFF           = 00000044      IRB$V_PPF_ECHO          = 00000039
IRB$L_NRP_VBN           = 00000040      IRB$V_PPF_EOF           = 0000002E
IRB$L_NXTBDB            = 0000003C      IRB$V_PPF_FNDSV         = 00000030
IRB$L_OLDBUF            = 0000006C      IRB$V_PPF_IMAGE         = 00000022
IRB$L_OWNER_ID          = 00000050      IRB$V_PPF_SKIP          = 0000002F
IRB$L_POS_VBN           = 000000AC      IRB$V_PRM               = 00000007
IRB$L_PTR_VBN           = 0000004C      IRB$V_PUTS_LAST         = 00000026
IRB$L_PUTOP_VBN         = 00000078      IRB$V_RAHWBH            = 0000002A
IRB$L_RECBUF            = 00000068      IRB$V_REC_W_LO          = 00000003
IRB$L_REC_COUNT         = 00000094      IRB$V_RESTART           = 0000003B
IRB$L_RFA_VBN           = 00000070      IRB$V_RMS_STALL         = 0000003A
IRB$L_RLB_LNK           = 00000038      IRB$V_RRV_ERR           = 00000032
IRB$L_RP                = 00000048      IRB$V_RU_DELETE         = 0000003D
IRB$L_RP_OFF            = 0000004C      IRB$V_RU_UNDEL          = 0000003E
IRB$L_RP_VBN            = 00000048      IRB$V_RU_UPDATE         = 0000003F
IRB$L_SIDR_VBN          = 000000B4      IRB$V_SKIP_NEXT         = 0000002B
IRB$L_SPL_COUNT         = 0000009C      IRB$V_SPL_IDX           = 00000000
IRB$L_TEMP0             = 00000064      IRB$V_SRCAGE            = 00000004
IRB$L_TEMP1             = 00000068      IRB$V_SRCHGT            = 00000001
IRB$L_UDR_VBN           = 000000B0      IRB$V_UNLOCK_RP         = 0000002D
IRB$L_UPDBUF            = 00000064      IRB$V_UPDATE            = 00000033
IRB$L_UPD_BDB           = 00000070      IRB$V_UPDATE_IF         = 00000034
IRB$L_VBN_LEFT          = 00000088      IRB$V_VALID_AT          = 00000000
IRB$L_VBN_MID           = 00000090      IRB$V_WRITE             = 00000029
IRB$L_VBN_RIGHT         = 0000008C      IRB$W_CSIZ              = 00000062
IRB$V_ABOVELCKD         = 00000035      IRB$W_CUR_COUNT         = 000000C0
IRB$V_ASYNC             = 00000023      IRB$W_CUR_ID            = 000000B8
IRB$V_ASYNCWAIT         = 00000024      IRB$W_FIRST_ID          = 00000082
IRB$V_BIG_SPLIT         = 00000002      IRB$W_ISI               = 00000028
IRB$V_BIO_LAST          = 00000027      IRB$W_LAST_ID           = 00000074
IRB$V_BKT_NO            = 00000000      IRB$W_NEXT_ID           = 00000080
IRB$V_BKT_NO_LO         = 00000000      IRB$W_NID_MID           = 000000A2
IRB$V_BRO_SW            = 00000028      IRB$W_NID_RIGHT         = 000000A0
IRB$V_BUSY              = 00000020      IRB$W_NRP_OFF           = 00000044
IRB$V_CONT_BKT          = 00000004      IRB$W_OWN_ID            = 00000050
IRB$V_CONT_R            = 00000005      IRB$W_OWN_ISI           = 00000052
IRB$V_CON_EOF           = 00000037      IRB$W_POS_ID            = 000000BA
IRB$V_DAP_CONN          = 0000003C      IRB$W_POS_INS           = 00000048
IRB$V_DEL_SEEN          = 00000009      IRB$W_PUTOP_ID          = 00000080
IRB$V_DUP               = 0000002C      IRB$W_RFA_ID            = 00000074
IRB$V_DUPS_SEEN         = 00000007      IRB$W_RFA_NID           = 000000A4
IRB$V_DUP_KEY           = 00000008      IRB$W_ROVADSZ           = 00000064
IRB$V_EMPTY_BKT         = 00000006      IRB$W_RP_OFF            = 0000004C
IRB$V_EMPT_SEEN         = 00000001      IRB$W_RTOTLSZ           = 00000066
IRB$V_EOF               = 00000021      IRB$W_SAVE_POS          = 00000076
IRB$V_FIND              = 00000029      IRB$W_SIDR_ID           = 000000BE
IRB$V_FIND_LAST         = 00000025      IRB$W_SPLIT             = 0000004A
IRB$V_FIRST_TIM         = 00000006      IRB$W_SPLIT_1           = 0000004C
```

```
IRB$W_SPLIT_2                   = 0000004E
IRB$W_SRCHFLAGS                 = 00000042
IRB$W_UDR_ID                    = 000000BC
IRBBKP_CHR                      00000370 R      03
JNLFLG2_CHR                     00000BA4 R      03
JNLFLG3_CHR                     00000BD4 R      03
JNLFLG_CHR                      00000338 R      03
LIB$SIGNAL                      ********    X   03
LINE_COUNT                      ********    X   03
LN_CHR                          00000ADC R      03
MJB                             000014A2 R      02
MJB$B_BID                       = 00000008
MJB$B_BLN                       = 00000009
MJB$B_JNL                       = 0000000C
MJB$C_BID                       = 00000018
MJB$C_BLN                       = 00000020
MJB$Q_DESC                      = 00000010
MJB$Q_IOSB                      = 00000018
MJB$V_FILE                      = 00000002
MJB$V_FORCE                     = 00000001
MJB$V_INIT                      = 00000000
MJB$V_SYNCH_SHARE               = 00000003
MJB$W_FLAGS                     = 0000000A
MJB_CHR                         00000B7C R      03
MSG$_INVBLKTYP                  ********    X   03
MSG$_NOIMGRMS                   ********    X   03
MSG$_NOTVALID                   ********    X   03
MSG$_RMSTERM                    ********    X   03
MSG$_SUCCESS                    ********    X   03
NAM                             00000CD6 R      02
NAM$B_BID                       = 00000000
NAM$B_ESL                       = 0000000B
NAM$B_RSL                       = 00000003
NAM$C_BID                       = 00000002
NAM$C_BLN                       = 00000060
NAM$L_ESA                       = 0000000C
NAM$L_FNB                       = 00000034
NAM$L_RSA                       = 00000004
NAM$L_WCC                       = 00000030
NAM$T_DVI                       = 00000014
NAM$V_EXP_DEV                   = 00000007
NAM$V_EXP_DIR                   = 00000006
NAM$V_EXP_NAME                  = 00000002
NAM$V_EXP_TYPE                  = 00000001
NAM$V_EXP_VER                   = 00000000
NAM$V_GRP_MBR                   = 00000013
NAM$V_HIGHVER                   = 0000000F
NAM$V_LOWVER                    = 0000000E
NAM$V_NODE                      = 00000011
NAM$V_PPF                       = 00000010
NAM$V_QUOTED                    = 00000012
NAM$V_WILDCARD                  = 00000008
NAM$V_WILD_DIR                  = 00000014
NAM$V_WILD_GRP                  = 00000018
NAM$V_WILD_MBR                  = 00000019
NAM$V_WILD_NAME                 = 00000005
NAM$V_WILD_SFD1                 = 00000019
NAM$V_WILD_SFD2                 = 0000001A
NAM$V_WILD_SFD3                 = 0000001B
NAM$V_WILD_SFD4                 = 0000001C
NAM$V_WILD_SFD5                 = 0000001D
NAM$V_WILD_SFD6                 = 0000001E
NAM$V_WILD_SFD7                 = 0000001F
NAM$V_WILD_TYPE                 = 00000004
NAM$V_WILD_UFD                  = 00000018
NAM$V_WILD_VER                  = 00000003
NAM_CHR                         000000C8 R      03
NEW_PAGE                        ********    X   03
OPT$M_ASB                       = 00000040
OPT$M_BDB                       = 00000010
OPT$M_BLB                       = 00008000
OPT$M_CCB                       = 00000080
OPT$M_FCB                       = 00000200
OPT$M_GBD                       = 00020000
OPT$M_GBDSUM                    = 00200000
OPT$M_GBH                       = 00040000
OPT$M_IDX                       = 00000008
OPT$M_IFB                       = 00000002
OPT$M_IRB                       = 00000004
OPT$M_RJB                       = 00400000
OPT$M_RLB                       = 00004000
OPT$M_RMSALL                    = FFFFFFFE
OPT$M_TRC                       = 00080000
OPT$M_WCB                       = 00001000
OPT$V_ASB                       = 00000006
OPT$V_BDB                       = 00000004
OPT$V_BDBSUM                    = 00000005
OPT$V_BLB                       = 0000000F
OPT$V_BLBSUM                    = 00000010
OPT$V_CCB                       = 00000007
OPT$V_FAB                       = 0000000A
OPT$V_FCB                       = 00000009
OPT$V_FWA                       = 00000014
OPT$V_GBD                       = 00000011
OPT$V_GBDSUM                    = 00000015
OPT$V_GBH                       = 00000012
OPT$V_IDX                       = 00000003
OPT$V_IFB                       = 00000001
OPT$V_IRB                       = 00000002
OPT$V_NAM                       = 0000000C
OPT$V_RAB                       = 0000000B
OPT$V_RJB                       = 00000016
OPT$V_RLB                       = 0000000E
OPT$V_TRC                       = 00000013
OPT$V_WCB                       = 00000008
OPT$V_XAB                       = 0000000D
OPT_CHR                         00000818 R      03
ORG_TBL                         000004C0 R      03
PAGE_SIZE                       ********    X   03
PARSE_CHR                       00000A7C R      03
PASS_CHR                        000009C4 R      03
PIO$GW_IIOIMPA                  ********    X   03
PIOSTRING                       00000BE4 R      03
PRINT                           ********    X   03
```

```
PRU_TBL                         000006F8 R      03
RAB                             00000C8E R      02
RAB$C_BLN                   =   00000044
RAT_CHR                         000001E8 R      03
RECFLG_CHR                      00000318 R      03
RFMORG_TBL                      000004E8 R      03
RFM_TBL                         000004CC R      03
RJB                             0000144E R      02
RJB$B_BID                   =   00000008
RJB$B_BLN                   =   00000009
RJB$C_BID                   =   00000016
RJB$C_BLN                   =   0000000C
RJB$Q_CHAN                  =   00000000
RJB$V_AI                    =   00000002
RJB$V_AT                    =   00000003
RJB$V_BI                    =   00000001
RJB$V_OPEN                  =   00000004
RJB$V_RU                    =   00000000
RJB$W_FLAGS                 =   0000000A
RJB_CHR                         00000B4C R      03
RLB                             0000136A R      02
RLB$B_BID                   =   00000008
RLB$B_BLN                   =   00000009
RLB$B_FLAGS                 =   0000000B
RLB$B_TMO                   =   0000000A
RLB$C_BID                   =   0000000E
RLB$C_BLN                   =   0000001C
RLB$L_LNK                   =   00000000
RLB$L_LOCK_ID               =   00000018
RLB$L_OWNER                 =   00000010
RLB$L_RFAO                  =   0000000C
RLB$V_CONV                  =   00000004
RLB$V_CR                    =   00000001
RLB$V_FAKE                  =   00000006
RLB$V_LV2                   =   00000005
RLB$V_PR                    =   00000003
RLB$V_PW                    =   00000002
RLB$V_TIMER_INPROG          =   00000000
RLB$V_TMO                   =   00000007
RLB$V_WAIT                  =   00000000
RLB$W_FLAGS2                =   00000004
RLB$W_RFA4                  =   00000006
RLB$W_STATUS                =   00000014
RLB2_CHR                        00000808 R      03
RLB_CHR                         000007C0 R      03
RLS$V_DEQ                   =   00000003
RLS$V_KEEP_LOCK             =   00000002
RLS$V_RETURN                =   00000000
RLS$V_WRT_THRU              =   00000001
RLS_CHR                         00000798 R      03
RMS_DIS_OPT                     00000000 RG     02
RMS_DIS_OPT1                    00000006 R      02
RMS_DIS_TMP                     0000000C RG     02
RMS_DIS_TMP1                    00000012 RG     02
RMS_HANDLER                     0000139F R      03
RMS_IFI                         00000004 RG     02
RMS_IFI1                        0000000A R      02
RMS_IFI_TMP                     00000010 RG     02
SCH_CHR                         0000054C R      03
SHOW_ASB                        00003371 R      03
SHOW_BDB                        00002F2D R      03
SHOW_BDB_SUM                    00002CC4 R      03
SHOW_BLB                        000030F5 R      03
SHOW_BLB_SUM                    00002DF4 R      03
SHOW_CCB                        0000224C R      03
SHOW_FCB                        000024E8 R      03
SHOW_FWA                        00001A28 R      03
SHOW_GBD_SUM                    00003744 R      03
SHOW_GBH                        00003629 R      03
SHOW_IDX                        00001823 R      03
SHOW_IFAB                       000013BD RG     03
SHOW_IRAB                       0000276E R      03
SHOW_MJB                        00003554 R      03
SHOW_NAM                        00003B23 R      03
SHOW_RJB                        000034A4 R      03
SHOW_RLB                        00003254 R      03
SHOW_RMS                        000012F0 RG     03
SHOW_RMS1                       00000C3A RG     03
SHOW_RMS_DIS                    0000130E RG     03
SHOW_RMS_OPT                    0000132C RG     03
SHOW_TRACE                      00003885 R      03
SHOW_WCB                        00002314 R      03
SKIP_LINES                      ********    X   03
SL_CHR                          00000B1C R      03
SPC_CHR                         000004F4 R      03
SPSTRING                        00000C31 R      03
SS$_ACCVIO                      ********    X   03
SS$_RESIGNAL                    ********    X   03
SYMBOLIZE                       ********    X   03
SYMBOL_VALUE                    ********    X   03
TRACE_DEPTH                     00000BFA R      03
TRACE_FLAGS                     00000C11 R      03
TRANSLATE_BITS                  ********    X   03
TRC                             0000145E R      02
TRC$B_BDB_CACHE             =   0000002C
TRC$B_BDB_FLAGS             =   0000002D
TRC$B_BID                   =   00000008
TRC$B_BLB_FLAGS             =   00000033
TRC$B_BLB_MODE              =   00000032
TRC$C_BID                   =   00000012
TRC$C_BLN                   =   00000040
TRC$L_ARG_FLG               =   00000020
TRC$L_BDB_ADDR              =   00000024
TRC$L_BDB_SEQ               =   0000002E
TRC$L_BLB_ADDR              =   00000034
TRC$L_BLB_LOCK              =   00000038
TRC$L_BLB_SEQ               =   0000003C
TRC$L_BLNK                  =   00000004
TRC$L_FLNK                  =   00000000
TRC$L_RETURN1               =   00000018
TRC$L_RETURN2               =   0000001C
TRC$L_STRUCTURE             =   0000000C
TRC$L_VBN                   =   00000014
TRC$W_BDB_BUFF              =   0000002A
```

52

```
TRC$W_BDB_USERS                 = 00000028
TRC$W_FUNCTION                  = 0000000A
TRC$W_PID                       = 00000010
TRC$W_SEQNUM                    = 00000012
UCB                               0000128E R      02
UCB$C_LENGTH                    = 00000090
WCB                               00000D4E R      02
WCB$B_ACCESS                    = 0000000B
WCB$B_TYPE                      = 0000000A
WCB$C_LENGTH                    = 00000030
WCB$L_FCB                       = 00000018
WCB$L_ORGUCB                    = 00000010
WCB$L_PID                       = 0000000C
WCB$L_RVT                       = 0000001C
WCB$L_STVBN                     = 0000002C
WCB$L_WLBL                      = 00000004
WCB$L_WLFL                      = 00000000
WCB$M_SHRWCB                    = 00000008
WCB$V_DLOCK                     = 00000001
WCB$V_NOREAD                    = 0000000A
WCB$V_NOTFCP                    = 00000002
WCB$V_NOTRUNC                   = 0000000B
WCB$V_NOWRITE                   = 00000000
WCB$V_OVERDRAWN                 = 00000004
WCB$V_READ                      = 00000000
WCB$V_READCK                    = 00000009
WCB$V_SEQONLY                   = 00000006
WCB$V_SHRWCB                    = 00000003
WCB$V_SPOOL                     = 00000004
WCB$V_WRITE                     = 00000001
WCB$V_WRITEAC                   = 00000008
WCB$V_WRITECK                   = 00000005
WCB$W_ACON                      = 00000014
WCB$W_NMAP                      = 00000016
WCB$W_P1_COUNT                  = 00000030
WCB$W_REFCNT                    = 0000000E
WCB$W_SIZE                      = 00000008
WCBACC_CHR                        000006A8 R      03
WCBEND                            00000F7E R      02
WCBSHR_CHR                        00000678 R      03
WILD_CHR                          00000A34 R      03
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
SDADATA	000014C2 (5314.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC BYTE
RMS	00003F10 (16144.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE
LITERALS	00003343 (13123.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE
_RMSLIT	00000421 (1057.)	05 (5.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	35	00:00:00.04	00:00:00.58
Command processing	142	00:00:00.57	00:00:04.62
Pass 1	1197	00:00:39.04	00:02:15.74
Symbol table sort	0	00:00:04.10	00:00:16.76
Pass 2	624	00:00:11.63	00:00:50.98
Symbol table output	1	00:00:00.54	00:00:01.89
Psect synopsis output	0	00:00:00.02	00:00:00.30
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	2031	00:00:55.94	00:03:30.87

The working set limit was 3000 pages.
399927 bytes (782 pages) of virtual memory were used to buffer the intermediate code.
There were 190 pages of symbol table space allocated to hold 3358 non-local and 597 local symbols.
3991 source lines were read in Pass 1, producing 146 object records in Pass 2.
69 pages of virtual memory were used to define 68 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SHRLIB]RMS.MLB;1	32
_255DUA28:[SDA.OBJ]SDALIB.MLB;1	10
_255DUA28:[SYS.OBJ]LIB.MLB;1	7
_255DUA28:[SYSLIB]STARLET.MLB;2	8
TOTALS (all libraries)	57

3540 GETS were required to define 57 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:RMS/OBJ=OBJ$:RMS MSRC$:RMS/UPDATE=(ENH$:RMS)+EXECML$/LIB+LIB$:SDALIB/LIB+SHRLIB$:RMS/LIB
```

PROCESS
LIS

RMS
LIS

QAST
LIS

RELEASE
LIS

POOL
LIS

SYMBOLS
LIS

SMGRTL

SMGBLOTRM
MAP

SDAMSG
LIS

VAXINST
LIS

SMGMAPTRM
MAP

SMGKCB
SDL

VALIDATE
LIS

STACKS
LIS

SMGDEF
SDL

SMGKDE
SDL

SMGSHR
MAP