F 5

```
   SSSSSSSSSSSSS DDDDDDDDDDDD        AAAAAAAAA
   SSSSSSSSSSSSS DDDDDDDDDDDD        AAAAAAAAA
   SSSSSSSSSSSSS DDDDDDDDDDDD        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
SSSSSSSSSSSS     DDDDDDDDDDDD     AAA         AAA
SSSSSSSSSSSS     DDDDDDDDDDDD     AAA         AAA
SSSSSSSSSSSS     DDDDDDDDDDDD     AAA         AAA
```

```
_S

Sy
--
CT
CT
CT
CT
CU
CU
CU
CU
DC
DC
DC
DC
DC
DC
DE
DE
DE
DE
DE
DE
DE
DE
DI
DI
DI
DI
DI
DI
DI
DI
DI
DI
DI
DI
DU
DU
DU
DU
DU
DU
DU
DU
DV
DV
DV
DY
DY
DY
DY
DY
ER
ES
```

FILEID**RMS

```
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         .TITLE  RMS     Display RMS Data Structures
0000         .SBTTL  Copyright Notice
0000         .IDENT  'V04-000'
0000   ;
0000   ;****************************************************************************
0000   ;*                                                                          *
0000   ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000   ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000   ;*  ALL RIGHTS RESERVED.                                                    *
0000   ;*                                                                          *
0000   ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000   ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000   ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000   ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000   ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000   ;*  TRANSFERRED.                                                            *
0000   ;*                                                                          *
0000   ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000   ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000   ;*  CORPORATION.                                                            *
0000   ;*                                                                          *
0000   ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000   ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000   ;*                                                                          *
0000   ;*                                                                          *
0000   ;****************************************************************************
0000   ;
```

```
0000    ;++
0000    ;       Facility
0000    ;
0000    ;           System Dump Analyzer
0000    ;
0000    ;       Abstract
0000    ;
0000    ;           This module contains code to dump most internal (and some user)
0000    ;           RMS data structures.
0000    ;
0000    ;       Environment
0000    ;
0000    ;           Native mode, User mode
0000    ;
0000    ;       Author
0000    ;
0000    ;           K. E. Kinnear   September 1980 - July 1981
0000    ;
0000    ;       Modified by
0000    ;
0000    ;           V03-030 RAS0271         Ron Schaefer            14-Mar-1984
0000    ;                   Yet more FWA changes.
0000    ;
0000    ;           V03-029 RAS0234         Ron Schaefer            31-Jan-1984
0000    ;                   More FWA changes for input sticky searchlists.
0000    ;
0000    ;           V03-028 RAS0230         Ron Schaefer            5-Jan-1984
0000    ;                   Reflect significant FWA structure changes.
0000    ;
0000    ;           V03-027 SHZ0002         Stephen Zalewski        06-Dec-1983
0000    ;                   Add flag definition to BLB flags.
0000    ;
0000    ;           V03-026 JWT0141         Jim Teague              14-Nov-1983
0000    ;                   Change IFB$V_RUM to IFB$V_ONLY_RU
0000    ;
0000    ;           V03-025 KBT0571         Keith B. Thompson       24-Jul-1983
0000    ;                   Track RMS changes
0000    ;
0000    ;           V03-024 SHZ0001         Stephen H. Zalewski     26-Jun-1983
0000    ;                   Remove undefined symbol.
0000    ;
0000    ;           V03-023 KBT0550         Keith B. Thompson       24-Jun-1983
0000    ;                   Fix KBT0548
0000    ;
0000    ;           V03-022 KBT0548         Keith B. Thompson       23-Jun-1983
0000    ;                   More FWA changes
0000    ;
0000    ;           V03-021 KPL0002         Peter Lieberwirth       1-Jun-1983
0000    ;                   Support fields added to IFAB and IRAB for journaling,
0000    ;                   including the MJB (Miscellaneous Journaling Buffer).
0000    ;                   Add some new GBD and GBH fields and one new IFAB field.
0000    ;
0000    ;           V03-020 KBT0534         Keith B. Thompson       1-Jun-1983
0000    ;                   Fix some FWA flags
0000    ;
0000    ;           V03-019 KPL0001         Peter Lieberwirth       30-May-1983
0000    ;                   Eliminate references due to obsolete RJB fields.
```

```
0000  ;
0000  ;       V03-018 KEK0030         K. E. Kinnear           15-Apr-1983
0000  ;               Fix LBN display in WCB to deal with relative
0000  ;               volume numbers in high part of LBN.
0000  ;
0000  ;       V03-017 DAS0003         David Solomon           11-Mar-1983
0000  ;               Display RLB$W_FLAGS2, RLB$B_TMO (new with timeout on
0000  ;               record lock). Formatting changes (add form feeds).
0000  ;
0000  ;       V03-016 RAS0123         Ron Schaefer             8-Feb-1983
0000  ;               Fix several display bugs and update to reflect MANY,
0000  ;               MANY RMS structure changes.
0000  ;
0000  ;       V03-015 TMK0004         Todd M. Katz            22-Jan-1983
0000  ;               Make a change to TMK0003.
0000  ;
0000  ;       V03-014 TMK0003         Todd M. Katz            21-Jan-1983
0000  ;               Add the definition of the bits IRB$V_RU_UPDATE and
0000  ;               IRB$V_NO_Q_WAIT to the IRAB bookkeeping bit table.
0000  ;
0000  ;       V03-013 TMK0002         Todd M. Katz            07-Jan-1983
0000  ;               Display IRB$L_OLDBUF and IRB$L_RECBUF. Add the displaying
0000  ;               of the bit field IFB$B_RECVRFLGS. Add the definition of the
0000  ;               bits IRB$V_RU_DELETE and IRB$V_RU_UNDEL to the IRAB bookkeeping
0000  ;               bit table.
0000  ;
0000  ;       V03-012 RAS0112         Ron Schaefer            22-Dec-1982
0000  ;               Change symbols:
0000  ;                       IFB$W_DEVBUFSIZ  ->  IFB$L_DEVBUFSIZ
0000  ;                       IFB$W_ASDEVBSIZ  ->  IFB$L_ASDEVBSIZ
0000  ;
0000  ;       V03-011 MCN0001         Maria del C. Nasr       29-Oct-1982
0000  ;               Eliminate COUNT_DUP, NORFA, and PRG_D_RFA from IDX
0000  ;               table to reflect changes made to RMSINTSTR.MDL.  Add
0000  ;               KEY_COMPR to IDX_CHR1.
0000  ;
0000  ;       V03-010 KBT0380         Keith B. Thompson       21-Oct-1982
0000  ;               Display new fields in asb and display all BIDs and BLNs
0000  ;               in decimal
0000  ;
0000  ;       V03-009 KBT0362         Keith B. Thompson       8-Oct-1982
0000  ;               Change asb$b_stksiz to a word
0000  ;
0000  ;       V03-008 KBT0320         Keith B. Thompson       8-Sep-1982
0000  ;               Make changes for new file sharing structures, and
0000  ;               change irb$b_srchflags to word and rlb$w_owner to long and
0000  ;               add ifb$v_stall_lock, irb$v_dup_key and irb$v_last_gt
0000  ;
0000  ;       V03-007 RAS0094         Ron Schaefer            27-Aug-1982
0000  ;               Delete symbol IRB$B_DIFF_CHAR.
0000  ;
0000  ;       V03-006 CWH0001         CW Hobbs                20-Aug-1982
0000  ;               Change symbol FWA$B_UNDERLINE to FWA$B_UNDER_DEV
0000  ;
0000  ;       V03-005 TMK0001         Todd M. Katz            18-Jul-1982
0000  ;               Add to the table SCH_CHR, a reference to the bit IRB$V_DEL_SEEN
0000  ;               and to the table IRBBKP_CHR a reference to the bit IRB$V_CON_EOF
```

```
0000  ;                        and the bit IRB$V_UPDATE_IF.
0000  ;
0000  ;                        Do not print the field IRB$L_NRP if the file is an indexed
0000  ;                        file. The field IRB$B_NRP_KREF no longer exists so don't bother
0000  ;                        printing it either.
0000  ;
0000  ;                        Print out the fields IRB$L_NEXT_VBN, IRB$W_NEXT_ID,
0000  ;                        IRB$L_PUTUP_VBN, IRB$W_PUTUP_ID, IRB$L_FIRST_VBN, and
0000  ;                        IRB$W_FIRST_ID.
0000  ;
0000  ;                        Print out IRB$L_CUR_VBN, IRB$W_CUR_ID, IRB$L_POS_VBN,
0000  ;                        IRB$W_POS_ID, IRB$L_UDR_VBN, IRB$W_UDR_ID, IRB$L_SIDR_VBN,
0000  ;                        IRB$W_SIDR_ID, IRB$W_CUR_COUNT and IRN$B_CUR_KREF.
0000  ;
0000  ;                        Also do some juggling in the order in which things are printed
0000  ;                        in both the IRAB and the IFAB.
0000  ;
0000  ;               V03-004 DWT0057         David Thiel             18-Jul-1982
0000  ;                        Fix references to IRB$V_UPD_NRP, IFB$L_NRP_LIST,
0000  ;                        IRB$B_NRP_KREF, IRB$L_NRP_PTR which were deleted
0000  ;                        in RMSINTSTR V03-011.
0000  ;
0000  ;               V03-003 MLJ0084         Martin L. Jack          30-Mar-1982
0000  ;                        Fix reference to IFB$B_KBUFSZ.
0000  ;
0000  ;               V03-002 KEK0027         K. E. Kinnear           1-Mar-1982
0000  ;                        Add error messages for no RMS structures and premature
0000  ;                        termination of the structure dump.  Also update IRAB,
0000  ;                        GBD, and GBPB to RMSINTSTR V02-085.
0000  ;
0000  ;               V03-001 KEK0022         K. E. Kinnear           3-Feb-1982
0000  ;                        Correct buffer reporting in GBD summary.
0000  ;
0000  ;               V3A Source Merge
0000  ;
0000  ;               V03-010 KEK0017         K. E. Kinnear           6-Jan-1982
0000  ;                        Add FWA, GBH, and GBD summary dump routines.
0000  ;
0000  ;               V03-009 KEK0016         K. E. Kinnear           17-Dec-1981
0000  ;                        Improve tracing code for global buffer debug.
0000  ;
0000  ;               V03-008 KEK0015         K. E. Kinnear           11-Dec-1981
0000  ;                        Remove READAHEAD flags from cache and BLB flags.
0000  ;
0000  ;               V03-007 KEK0013         K. E. Kinnear           23-Oct-1981
0000  ;                        New BLB flags, make AVLCL show for all orgs, and FRBPTR
0000  ;                        for rel and indexed orgs.
0000  ;
0000  ;               V03-006 KEK0012         K. E. Kinnear           9-Sep-1981
0000  ;                        Add trace support and initialization so that it
0000  ;                        fails quietly on no-P1 space processes.
0000  ;
0000  ;               V03-005 KEK0011         K. E. Kinnear           2-Sep-1981
0000  ;                        Fix references to $BCBDEF which has gone away.
0000  ;                        Remove references to RLS$V_KEEP_ACC which has also
0000  ;                        gone away.
0000  ;                        Fix sequential problem in SHOW_BLB_SUM.
```

```
0000     ;
0000     ;       V03-004 KEK0010         K. E. Kinnear           26-Aug-1981
0000     ;               React to changes to internal RMS structures for
0000     ;               global buffering.  Support for BLB, and changes to
0000     ;               IFAB and IRAB.  Preliminary support for GBD, GBH, and
0000     ;               TRC structures.
0000     ;
0000     ;       V03-003 KEK0009         K. E. Kinnear           15-Aug-1981
0000     ;               Update for changes in IFB, IRB, IDX for prolog 3
0000     ;               files.
0000     ;
0000     ;       V03-002 KEK0005         K. E. Kinnear           1-Aug-1981
0000     ;               React to changing RMS structures.  RFA_ID in
0000     ;               IRAB went from a word to a byte.
0000     ;--
```

```
0000            .SBTTL  Declarations
0000    ;
0000    ;       Symbol Defintions
0000    ;
0000
0000            $CHFDEF                 ; Condition Handling Definitions
0000            $OPTDEF                 ; Parsing options
0000            $TPADEF                 ; TPARSE definitions
0000
0000    ;
0000    ;       RMS and SYSTEM definitions
0000    ;
0000
0000            $ASBDEF                         ; Asynchronous context block
0000            $BDBDEF                         ; Buffer descriptor block
0000            $BLBDEF                         ; Bucket lock block
0000            $CCBDEF                         ; Channel control block
0000            $CSHDEF                         ; Cache control bits
0000            $DCDEF                          ; Device class and type defns
0000            $DDBDEF                         ; Device data block def
0000            $DEVDEF                         ; Device characteristic bits
0000            $DRCDEF                         ; Directory cache definitions
0000            $DYNDEF                         ; System control block idents
0000            $FABDEF                         ; File access block
0000            $FCBDEF                         ; File control block
0000            $FWADEF                         ; File string work area
0000            $GBDDEF                         ; Global buffer descriptor
0000            $GBHDEF                         ; Global buffer header
0000            $GBPBDEF                        ; Global buffer param block
0000            $IDXDEF                         ; Index Descriptor definitions
0000            $IFBDEF                         ; Internal FAB definitions
0000            $IMPDEF                         ; Process I/O control area
0000            $IRBDEF                         ; Internal RAB definitions
0000            $NAMDEF                         ; Name access block
0000            $PLGDEF                         ; Prologue (VBN 1) of indexed file
0000            $RABDEF                         ; Record access block
0000            $RJBDEF                         ; Record journaling block
0000            $RLBDEF                         ; Record lock block
0000            $RLSDEF                         ; Release bit definitions
0000            $RMSDEF                         ; RMS message definitions
0000            $SFSBDEF                        ; Shared file sync. block
0000            $TRCDEF                         ; Trace block definition
0000            $UCBDEF                         ; Unit control block defns
0000            $WCBDEF                         ; Window control block
0000            $XABDEF                         ; Extended access blocks
0000            $XABFHCDEF                      ; XAB - file header
0000            $XABALLDEF                      ; XAB - allocation
0000            $XABDATDEF                      ; XAB - date/time
0000            $XABRDTDEF                      ; XAB - revision date/time
0000            $XABPRODEF                      ; XAB - protection
0000            $XABTRMDEF                      ; XAB - terminal
0000            $XABSUMDEF                      ; XAB - indexed summary
0000            $XABKEYDEF                      ; XAB - indexed keys
0000            $MJBDEF                         ; Miscellaneous Journaling Block
0000
```

```
0000            .SBTTL  Macro Definitions
0000
0000    ;
0000    ;       Macro Definitions
0000    ;
0000
0000    ;
0000    ; GET -- Get a particular control block in memory.
0000    ;
0000
0000            .MACRO  GET     BLOCK,ADDR,ERRADDR,BUFFER,?LAB,?LAB1
0000
0000            .IF NB  ADDR
0000            MOVAL   ADDR,BLOCK-4
0000            BNEQ    LAB1
0000            BRW     ERRADDR
0000    LAB1:
0000            .ENDC
0000
0000            .IF NB  BUFFER
0000            PUSHAL  BUFFER
0000            .IFF
0000            PUSHAL  BLOCK
0000            .ENDC
0000
0000            CALLS   #1,GET_'BLOCK
0000            BLBS    R0,LAB
0000            CLRL    BLOCK-4
0000            BRW     ERRADDR
0000    LAB:
0000
0000            .IF NB  BUFFER
0000            MOVAB   BUFFER,R2
0000            .IFF
0000            MOVAB   BLOCK,R2
0000            .ENDC
0000
0000            .ENDM
0000
0000    ;
0000    ; SHOW -- Format a control block on the output file.
0000    ;
0000
0000            .MACRO  SHOW    BLOCK,BUFFER
0000
0000            .IF NB  BUFFER
0000            PUSHAL  BUFFER
0000            .IFF
0000            PUSHAL  BLOCK
0000            .ENDC
0000
0000            CALLS   #1,SHOW_'BLOCK
0000
0000            .ENDM
0000
0000    ;
0000    ; TABLE Macro which subtracts a number from symbol values for
```

```
0000        ; bits whose values are all over 32.
0000        ;
0000
0000                .MACRO  TABLEM  N,PREFIX,VALUES
0000                .IRP    VALUE,VALUES
0000                .SAVE
0000                .PSECT  __RMSLIT,EXE,NOWRT
0000        $$$     =       .
0000                .ASCIC  \VALUE\
0000                .RESTORE
0000                .LONG   'PREFIX''VALUE'-<N>
0000                .LONG   $$$
0000                .ENDR
0000                .LONG   -1,-1
0000                .ENDM   TABLEM
0000
0000        ;
0000        ; CASETBL macro, builds lists of information which
0000        ; is indexed by specific values.
0000        ;
0000
0000                .MACRO  CASETBL VALUES
0000                .IRP    VALUE,VALUES
0000                .SAVE
0000                .PSECT  __RMSLIT,EXE,NOWRT
0000        $$$     =       .
0000                .ASCIC  \VALUE\
0000                .RESTORE
0000                .LONG   $$$
0000                .ENDR
0000                .ENDM   CASETBL
0000
0000        ;
0000        ; Macro to push and adjust a descriptor for an !AF FAO parameter.
0000        ;
0000
0000                .MACRO  PUSHD2  VALUE,ADDRESS
0000                PUSHL   ADDRESS+4(R3)
0000                ADDL2   VALUE,(SP)
0000                PUSHL   ADDRESS(R3)
0000                .ENDM
0000
0000        ;
0000        ; Macro to print the two longwords of a descriptor.
0000        ;
0000
0000                .MACRO  PRINTD  BLOCK,NAME
0000                PUSHL   BLOCK''NAME'+4(R3)
0000                PUSHL   BLOCK''NAME'(R3)
0000                .IF LESS_THAN 6-%LENGTH(NAME)
0000                PRINT   2,<-
0000        NAME':  !XL             -
0000                        !XL>
0000                .IFF
0000                PRINT   2,<-
0000        NAME':          !XL             -
0000                        !XL>
```

```
0000            .ENDC
0000            .ENDM
0000
0000    ;
0000    ; Macro to print the two longwords of a descriptor and print out
0000    ; the string pointed to by the descriptor.
0000    ;
0000    ; MUST BE USED WITHIN LOCAL SYMBOL BLOCK!
0000    ;
0000    ; ONLY FOR USE IN SHOW_FWA!
0000    ;
0000    ; REQUIRES R5 TO BE OFFSET FOR FWA DESCRIPTORS!
0000    ;
0000
0000            .MACRO  DUMPDF  NAME,?X
0000            PRINTD  FWA$Q_,NAME
0000            TSTL    FWA$Q_'NAME'(R3)
0000            BEQL    X
0000            MOVAL   FWA$Q_'NAME'(R3),R2
0000            CALLS   #0,FWADD
0000    X:
0000            .ENDM
0000
0000
0000    ;
0000    ; Macro to translate bits for fields at the beginning of the FWA
0000    ;
0000
0000            .MACRO  FWABITS TAG
0000            MOVL    #ALLOCSZ,(SP)
0000            MOVZBL  FWA$B_'TAG'FLGS(R3),-(SP)
0000            PUSHAB  TAG'_CHR
0000            CALLS   #2,TRANSLATE_BITS
0000            PUSHL   R4
0000            MOVZBL  FWA$B_'TAG'FLGS(R3),-(SP)
0000            PRINT   2,<-
0000    TAG'FLGS:       !XB     !AS>
0000            .ENDM
```

```
              0000          .SBTTL  Storage Definitions
              0000    ;
              0000    ; Storage Definitions
              0000    ;
              0000
          00000000          .PSECT  SDADATA,NOEXE,WRT
              0000
              0000    ;
              0000    ; /DISPLAY Options
              0000    ;
              0000
FFFFFFFE      0000    RMS_DIS_OPT::.LONG OPT$M_RMSALL          ; master options word
    0000      0004    RMS_IFI::.WORD  0                        ; contains IFI value (0=all)
              0006
              0006    ;
              0006    ; "real" options for main routine: SHOW_RMS1
              0006    ;
              0006
00000000      0006    RMS_DIS_OPT1:.LONG 0                     ; dynamic options word
    0000      000A    RMS_IFIT:.WORD  0                        ; dynamic ifi word
              000C
              000C    ;
              000C    ; Temporary options work area
              000C    ;
              000C
00000000      000C    RMS_DIS_TMP::.LONG 0                     ; holds options from TPARSE !DISPLAY
    0000      0010    RMS_IFI_TMP::.WORD 0                     ; holds IFI value from TPARSE !DISPLAY
00000000      0012    RMS_DIS_TMP1::.LONG 0                    ; work area for !DISPLAY
              0016
              0016    ;
              0016    ; General Temporary Data Storage
              0016    ;
              0016
    0000      0016    IFI:    .WORD   0                        ; Current IFI on which we are working
    0000      0018    IFBS:   .WORD   0                        ; number of ifabs seen this run
              001A
              001A    ;
              001A    ; Buffer areas into which are read the various control blocks from the dump.
              001A    ;
              001A    ;       All are in the format:
              001A    ;
              001A    ;               .LONG   0
              001A    ;       LABEL:  .BLKB   length of structure
              001A    ;
              001A    ;       where the longword prior to the label contains the virtual address
              001A    ;       of this control block in the actual dump, or 0 if block doesn't
              001A    ;       exist or couldn't be read.
              001A    ;
              001A
              001A          ASSUME IMP$C_NPIOFILES GE IMP$C_ENTPERSEG
              001A
0000011A      001A    IFABTBL:.BLKL   IMP$C_NPIOFILES+1
0000011E      011A    IFBTBLSIZ:.BLKL 1
00000000      011E            .LONG   0
000001DA      0122    IFAB:   .BLKB   IFB$C_BLN
00000000      01DA            .LONG   0
000002A2      01DE    IRAB:   .BLKB   IRB$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
```

```
    14C2          .SBTTL  Read-only Data Definitions
    14C2
    14C2  ;
    14C2  ;       Read-only Data Definitions
    14C2  ;
    14C2
00000000          .PSECT  RMS,EXE,NOWRT
    0000
    0000  DEV_CHR:TABLE    DEV$V_,<ALL,AVL,CCL,DIR,DMT,ELG,FOD,FOR,-
    0000                   GEN,IDV,MBX,NET,ODV,RCK,REC,RND,RTM,SDI,-
    0000                   SHR,SPL,SQD,SWL,TRM,WCK>
    00C8
    00C8  NAM_CHR:TABLE    NAM$V_,<EXP_DEV,EXP_DIR,EXP_NAME,EXP_TYPE,-
    00C8                   EXP_VER,GRP_MBR,HIGHVER,LOWVER,NODE,PPF,-
    00C8                   QUOTED,WILDCARD,WILD_DIR,WILD_GRP,WILD_MBR,-
    00C8                   WILD_NAME,WILD_SFD1,WILD_SFD2,WILD_SFD3,-
    00C8                   WILD_SFD4,WILD_SFD5,WILD_SFD6,WILD_SFD7,-
    00C8                   WILD_TYPE,WILD_UFD,WILD_VER>
    01A0
    01A0  FAC_CHR:TABLE    FAB$V_,<BIO,BRO,DEL,GET,PUT,TRN,UPD,EXE>
    01E8
    01E8  RAT_CHR:TABLE    FAB$V_,<BLK,CR,FTN,PRN>
    0210
    0210  BKP_CHR:TABLEM   32,IFB$V_,<BUSY,EOF,PPF_IMAGE,ASYNC,ASYNCWAIT,ACCESSED,-
    0210                   ANSI_D,RWC,DMO,SPL,SCF,DLT,DFW,SQO,PPF_INPUT,NFS,-
    0210                   WRTACC,MSE,CREATE,NORECLK,RW_ATTR,TMP,TEF,STALL_LOCK,-
    0210                   SEQFIL,SEARCH,RMS_STALL,RESTART,FILEFOUND,DAP_OPEN,DAP,NSP>
    0318
    0318  RECFLG_CHR:TABLE IFB$V_,<RU_RECVR,AI_RECVR,BI_RECVR>
    0338
    0338  JNLFLG_CHR:TABLE IFB$V_,<ONLY_RU,RU,AI,AT,BI,NEVER_RU>
    0370
    0370  IRBBKP_CHR:TABLEM 32,IRB$V_,<BUSY,EOF,PPF_IMAGE,ASYNC,ASYNCWAIT,FIND_LAST,-
    0370                   PUTS_LAST,BIO_LAST,BRO_SW,FIND,WRITE,RAHWBH,SKIP_NEXT,-
    0370                   DUP,UNLOCK_RP,PPF_EOF,PPF_SKIP,PPF_FNDSV,IDX_ERR,RRV_ERR,-
    0370                   UPDATE,UPDATE_IF,ABOVELCKD,GBLBUFF,CON_EOF,RMS_STALL,DAP_CONN,-
    0370                   RU_DELETE,RU_UNDEL,RU_UPDATE,NO_Q_WAIT,PPF_ECHO,RESTART>
    0480
    0480  BDB_CHR:TABLE    BDB$V_,<AST_DCL,DRT,IOP,NOLOCATE,PRM,VAL,WFO>
    04C0
    04C0  ORG_TBL:CASETBL <Sequential,Relative,Indexed>
    04CC
    04CC  RFM_TBL:CASETBL <UDF,FIX,VAR,VFC,STM,STMLF,STMCR>
    04E8
    04E8  RFMORG_TBL:CASETBL <SEQ,REL,IDX>
    04F4
    04F4  SPL_CHR:TABLE    IRB$V_,<BKT_NO_LO,REC_W_LO,CONT_BKT,CONT_R,EMPTY_BKT,-
    04F4                   DUPS_SEEN,BKT_NO,BIG_SPLIT,SPL_IDX,EMPT_SEEN>
    054C
    054C  SCH_CHR:TABLE    IRB$V_,<POSINSERT,SRCHGT,POSDELETE,NEW_IDX,SRCHGE,-
    054C                   NORLS_RNF,FIRST_TIM,PRM,DEL_SEEN,DUP_KEY,LAST_GT>
    05AC
    05AC  IDX_CHR1:TABLE   IDX$V_,<DUPKEYS,CHGKEYS,NULKEYS,INITIDX,IDX_COMPR,KEY_COMPR>
    05E4
    05E4  IDX_CHR0:TABLE   IDX$V_,<DUPKEYS,IDX_COMPR,INITIDX,KEY_COMPR,REC_COMPR>
    0614
    0614  IDX_TBL:CASETBL <STRING,SGNWORD,UNSGNWORD,SGNLONG,UNSGNLONG,PACKED,-
```

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

```
          0B1C                          DFLT_MFD,EXP_ROOT>
          0B4C
          0B4C  RJB_CHR:TABLE       RJB$V_,<RU,BI,AT,AI,OPEN>
          0B7C
          0B7C  MJB_CHR:TABLE       MJB$V_,<INIT,FORCE,FILE,SYNCH_SHARE>
          0BA4
          0BA4  JNLFLG2_CHR:        TABLE    IFB$V_,<VALID_AT,JNL,RUP,RU_RLK,DONE_ASS_JNL>
          0BD4
          0BD4  JNLFLG3_CHR:        TABLE    IRB$V_,<VALID_AT>
          0BE4
          0BE4  ;
          0BE4  ; Local Strings
          0BE4  ;
          0BE4
          0BE4  PIOSTRING:          STRING   <PIO$GW_IIOIMPA>
          0BFA  TRACE_DEPTH:        STRING   <RMS$TRACE_DEPTH>
          0C11  TRACE_FLAGS:        STRING   <RMS$TRACE_FLAGS>
          0C28  GSTRING:            STRING   <G>               ; GBPB signal for BDBSUM
          0C31  SPSTRING:           STRING   < >               ; opposite of above
          0C3A
          0C3A  ;
          0C3A  ; Constant Data Declarations
          0C3A  ;
          0C3A
00000087  0C3A  ALLOCSZ =           135                ; size of data buf alloc'ed on stack
```

```
                              0C3A          .SBTTL  SHOW_RMS1 -- Display RMS Data Structures
                              0C3A
                              0C3A     ;+++
                              0C3A     ; SHOW_RMS1 -- Display RMS Data Stuctures
                              0C3A     ;
                              0C3A     ; Inputs:
                              0C3A     ;
                              0C3A     ;        None.
                              0C3A     ;
                              0C3A     ; Outputs:
                              0C3A     ;
                              0C3A     ;        RMS Structures displayed.
                              0C3A     ;---
                              0C3A
                              0C3A            .ENABL  LSB
                              0C3A     SHOW_RMS1::
                        03FC  0C3A            .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9>
  6D    0000139F'EF     DE    0C3C            MOVAL   RMS_HANDLER,(FP)         ; Establish handler for SHOW RMS
  56    0000001A'EF     9E    0C43            MOVAB   IFABTBL,R6               ; setup address for IFAB table
                              0C4A
                              0C4A     ;
                              0C4A     ; Look up PIO$GW_IIOIMPA is SDA symbol table, so it may be redefined
                              0C4A     ; for ITESTD.
                              0C4A     ;
                              0C4A
        00000018'EF     B4    0C4A            CLRW    IFBS                     ; no IFAB's found yet
              91 AF     DF    0C50            PUSHAL  PIOSTRING                ; argument for symbol_value
      00000000'EF 01    FB    0C53            CALLS   #1,SYMBOL_VALUE
              59 51     D0    0C5A            MOVL    R1,R9                    ; put value (if any) in right register
              07 50     E8    0C5D            BLBS    R0,5$                    ; branch if ok
  59    00000000'EF     D0    0C60            MOVL    PIO$GW_IIOIMPA,R9        ; get address of image i/o impure area
        57    20 A9     DE    0C67     5$:    MOVAL   IMP$W_ENTPERSEG(R9),R7   ; find table size
                              0C6B            GETMEM  (R7),IFBTBLSIZ           ; get value
              24 50     E9    0C7B            BLBC    R0,15$                   ; quit if none
              59 18     C0    0C7E            ADDL    #IMP$L_IFABTBL,R9        ; form pointer to table pointer
                              0C81            GETMEM  (R9),(R6)                ; get P1 space address
              12 50     E9    0C8D            BLBC    R0,15$                   ; if none, don't complain
                              0C90            SKIP    PAGE
        00000016'EF     B4    0C97            CLRW    IFI                      ; initialize IFI
                              0C9D
                              0C9D     ;
                              0C9D     ; Got a new table
                              0C9D     ;
                              0C9D
              59 66     D0    0C9D     10$:   MOVL    (R6),R9                  ; get next link to IFAB table
                 1D     12    0CA0            BNEQ    18$
                              0CA2
        00000018'EF     B5    0CA2     15$:   TSTW    IFBS                     ; found any IFAB's?
                 12     12    0CA8            BNEQ    17$                      ; yes, don't message
                              0CAA
                              0CAA     ;
                              0CAA     ; SIGNAL the fact that there are no RMS structures
                              0CAA     ;
                              0CAA
                              0CAA            SIGNAL  0,NOIMGRMS               ; no image RMS structures
                              0CBC
               0629     31    0CBC     17$:   BRW     900$
```

```
          57  0000011A'EF   3C  0CBF  18$:    MOVZWL  IFBTBLSIZ,R7              ; compute tablesize
                      57    D6  0CC6          INCL    R7                        ; + 1 for link
               57     04    C4  0CC8          MULL2   #4,R7
                                0CCB          GETMEM  (R9),(R6),R7              ; read in image IFAB table
               C7 50        E9  0CD8          BLBC    R0,15$                    ; can't read in table
          57  0000011A'EF   3C  0CDB          MOVZWL  IFBTBLSIZ,R7              ; number of entries in table
               53     56    D0  0CE2          MOVL    R6,R3                     ; set up address of first IFAB slot
                      83    D5  0CE5          TSTL    (R3)+                     ; space over link to next IFAB table
                                0CE7  ;
                                0CE7  ; Scan thru IFAB table -- 0 means no entry here
                                0CE7  ;
                                0CE7  ;       R3 = pointer into IFABTBL buffer at next IFAB address
                                0CE7  ;       R6 = address of IFABTBL buffer in local memory
                                0CE7  ;       R7 = number of entries in IFAB table
                                0CE7  ;       R9 = virtual address of IFAB table
                                0CE7  ;
                                0CE7
              00000016'EF   B6  0CE7  20$:    INCW    IFI                       ; bump IFI value
                                0CED          GET     IFAB,@(R3)+,800$          ; find IFAB if any, 080$ on error
              00000018'EF   B6  0D19          INCW    IFBS                      ; we found an IFAB, so count it
                                0D1F  ;
                                0D1F  ; If this is display for specific IFI, then check here, and go around
                                0D1F  ; everything else!
                                0D1F  ;
                                0D1F
              0000000A'EF   B5  0D1F          TSTW    RMS_IFI1                          ; specific IFI?
                      10    13  0D25          BEQL    25$                       ; if eql no
 00000016'EF  0000000A'EF   B1  0D27          CMPW    RMS_IFI1,IFI              ; this the one?
                      03    13  0D32          BEQL    25$                       ; if eql yes
                    05A7    31  0D34          BRW     800$                      ; just like its not there
                                0D37
                                0D37  ;
                                0D37  ; SHOW IFAB
                                0D37  ;
                                0D37
                                0D37  25$:    SHOW    IFAB
                                0D44
                                0D44  ;
                                0D44  ; Get and Show ASB if any for this IFAB and if enabled.
                                0D44  ;
                                0D44
 00000006'EF  00000040 8F   D3  0D44          BITL    #OPT$M_ASB,RMS_DIS_OPT1 ; ASB display enabled?
                      3D    13  0D4F          BEQL    30$                       ; if eql no
                                0D51          GET     ASB,@IFB$L_ASBADDR+IFAB,30$
                                0D81          SHOW    ASB
                                0D8E
                                0D8E  ;
                                0D8E  ; Display index descriptors (IDX$) if indexed file.
                                0D8E  ;
                                0D8E
          02  00000145'EF   91  0D8E  30$:    CMPB    IFB$B_ORGCASE+IFAB,#IFB$C_IDX ;is file indexed?
                      75    12  0D95          BNEQ    40$                       ; if neq no -- skip index descriptors
                                0D97
                                0D97  ;
                                0D97  ; Do we need the IDX descriptors at all?
                                0D97  ;
                                0D97
```

```
          00000006'EF    08  D3  0D97       BITL    #OPT$M_IDX,RMS_DIS_OPT1 ; display the IDX descriptors?
                         6C  13  0D9E       BEQL    40$                     ; if eql no
                                 0DA0
                                 0DA0       GET     IDX,@IFB$L_IDX_PTR(R2),40$
                         2D  11  0DCE       BRB     37$                     ; enter display loop in progress
                                 0DD0
                                 0DD0 35$:  GET     IDX,@IDX$L_IDXFL(R2),40$
                                 0DFD 37$:  SHOW    IDX
                         C4  11  0E0A       BRB     35$
                                 0E0C
                                 0E0C ;
                                 0E0C ; If it looks like the first BDB contains a FWA, see if we can dump it.
                                 0E0C ;
                                 0E0C
       03 00000006'EF    14  E0  0E0C 40$:  BBS     #OPT$V_FWA,RMS_DIS_OPT1,41$ ; if we dont want FWA, skip it
                       007A  31  0E14       BRW     43$
                                 0E17 41$:  GET     FWA,@IFB$L_FWA_PTR+IFAB,43$     ; get FWA
                                 0E47       SHOW    FWA
                                 0E54       GET     FWA,@FWA$L_FWA_PTR+FWA,43$      ; get second FWA for $RENAME
                                 0E84       SHOW    FWA
                                 0E91
                                 0E91 ;
                                 0E91 ; Start on path to file id.  First find and dump CCB, then WCB, then FCB.
                                 0E91 ;
                                 0E91
00000006'EF     00000380 8F  D3  0E91 43$:  BITL    #<OPT$M_CCB!OPT$M_WCB!OPT$M_FCB>,RMS_DIS_OPT1 ; any of these?
                         03  12  0E9C       BNEQ    44$
                       00A2  31  0E9E       BRW     46$
00000D36'EF     00000142'EF  3C  0EA1 44$:  MOVZWL  IFB$W_CHNL+IFAB,CCB-4
                00000D3A'EF  DF  0EAC       PUSHAL  CCB
         00003C2C'EF     01  FB  0EB2       CALLS   #1,GET_CCB
                      03 50  E8  0EB9       BLBS    R0,45$
                       0084  31  0EBC       BRW     46$
                                 0EBF
                                 0EBF 45$:  SHOW    CCB
                                 0ECC       GET     WCB,@CCB$L_WIND+CCB,46$
                                 0EFC       SHOW    WCB
                                 0F09       GET     FCB,@WCB$L_FCB(R2),46$
                                 0F36       SHOW    FCB
                                 0F43
                                 0F43 ;
                                 0F43 ; RJB - RMS Journaling Block
                                 0F43 ; IFB MJB (AI extend Journal buffer)
                                 0F43 ;
                                 0F43
                                 0F43 46$:
                                 0F43
                                 0F43 ; If RJB display enabled, Get and Show the RJB (if any).
                                 0F43 ;
                                 0F43
00000006'EF     00400000 8F  D3  0F43       BITL    #OPT$M_RJB,RMS_DIS_OPT1
                         3D  13  0F4E       BEQL    47$
                                 0F50       GET     RJB,@IFB$L_RJB+IFAB,47$
                                 0F80       SHOW    RJB
                                 0F8D
                                 0F8D ;
                                 0F8D ; If MJB display enabled, Get and Show the MJB (if any).
```

```
                                    0F8D     ;
                                    0F8D
                                    0F8D     47$:
                                    0F8D             GET     MJB,@IFB$L_EXTJNLBUF+IFAB,48$
                                    0FBD             SHOW    MJB
                                    0FCA
                                    0FCA
                                    0FCA     ;
                                    0FCA     ; Scan thru IRAB chain and show individual IRAB's
                                    0FCA     ;
                                    0FCA
00000006'EF     00004044 8F   D3    0FCA     48$:    BITL    #<OPT$M_IRB!OPT$M_ASB!OPT$M_RLB>,RMS_DIS_OPT1
                         03   12    0FD5             BNEQ    49$
                       0136   31    0FD7             BRW     60$
         52     000001DE'EF   DE    0FDA     49$:    MOVAL   IRAB,R2
      1C A2     0000013E'EF   D0    0FE1             MOVL    IFB$L_IRAB_LNK+IFAB,IRB$L_IRAB_LNK(R2) ; set initial addr IRAB
                                    0FE9     50$:    GET     IRAB,@IRB$L_IRAB_LNK+IRAB,60$
                                    1019             SHOW    IRAB
                                    1026
                                    1026     ;
                                    1026     ; Get and Show the IRB MJB (if any).
                                    1026     ;
                                    1026
                                    1026             GET     MJB,@IRB$L_ATJNLBUF+IRAB,51$
                                    1056             SHOW    MJB
                                    1063
                                    1063     ;
                                    1063     ; If ASB display enabled, Get and Show the ASB (if any).
                                    1063     ;
                                    1063
00000006'EF     00000040 8F   D3    1063     51$:    BITL    #OPT$M_ASB,RMS_DIS_OPT1
                         3D   13    106E             BEQL    52$
                                    1070             GET     ASB,@IRB$L_ASBADDR+IRAB,50$
                                    10A0             SHOW    ASB
                                    10AD
                                    10AD     ;
                                    10AD     ; Display all RLB's attached to this IRAB
                                    10AD     ;
                                    10AD
00000006'EF     00004000 8F   D3    10AD     52$:    BITL    #OPT$M_RLB,RMS_DIS_OPT1 ; is it selected?
                         03   12    10B8             BNEQ    55$                     ; if neq yes
                       FF2C   31    10BA     54$:    BRW     50$
0000136A'EF     00000216'EF   D0    10BD     55$:    MOVL    IRB$L_RLB_LNK+IRAB,RLB$L_LNK+RLB ; set up  pointer to 1st rlb
                         F0   13    10C8             BEQL    54$
                                    10CA             SKIP    PAGE
                                    10D1     57$:    GET     RLB,@RLB$L_LNK+RLB,50$
                                    1101             SHOW    RLB
                         C1   11    110E             BRB     57$
                                    1110
                                    1110     ;
                                    1110     ; Scan thru BDB list and show information from BDB's
                                    1110     ;
                                    1110
    00002CC4'EF          00   FB    1110     60$:    CALLS   #0,SHOW_BDB_SUM         ; show summary of BDB's if selected
    00002DF4'EF          00   FB    1117             CALLS   #0,SHOW_BLB_SUM         ; show summary of BLB's if selected
    00000006'EF          10   D3    111E             BITL    #OPT$M_BDB,RMS_DIS_OPT1
                         73   13    1125             BEQL    70$
```

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

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

```
                                  12F0          .SBTTL  SHOW_RMS -- Display current default RMS Data Struct.
                                  12F0
                                  12F0  ;+++
                                  12F0  ; SHOW_RMS -- Print out default RMS structures
                                  12F0  ;
                                  12F0  ; Inputs:
                                  12F0  ;
                                  12F0  ;       RMS_DIS_OPT = Display options for this command.
                                  12F0  ;       RMS_IFI     =       "
                                  12F0  ;
                                  12F0  ; Outputs:
                                  12F0  ;
                                  12F0  ;       None.
                                  12F0  ;---
                                  12F0
                            03FC  12F0  SHOW_RMS::.WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9>
00000006'EF   00000000'EF     D0  12F2          MOVL    RMS_DIS_OPT,RMS_DIS_OPT1
0000000A'EF   00000004'EF     B0  12FD          MOVW    RMS_IFI,RMS_IFIT
          F92D CF       00    FB  1308          CALLS   #0,SHOW_RMS1
                              04  130D          RET
                                  130E
                                  130E  ;+++
                                  130E  ; SHOW_RMS_DIS -- Print out RMS structures specified by /DISPLAY qualifer.
                                  130E  ;
                                  130E  ; Inputs:
                                  130E  ;
                                  130E  ;       RMS_DIS_TMP = Display options for this command.
                                  130E  ;       RMS_IFI_TMP =       "
                                  130E  ;
                                  130E  ; Outputs:
                                  130E  ;
                                  130E  ;       None.
                                  130E  ;---
                                  130E
                            03FC  130E  SHOW_RMS_DIS::.WORD     ^M<R2,R3,R4,R5,R6,R7,R8,R9>
00000006'EF   0000000C'EF     D0  1310          MOVL    RMS_DIS_TMP,RMS_DIS_OPT1
0000000A'EF   00000010'EF     B0  131B          MOVW    RMS_IFI_TMP,RMS_IFIT
          F90F CF       00    FB  1326          CALLS   #0,SHOW_RMS1
                              04  132B          RET
```

```
                             132C          .SBTTL  SHOW_RMS_OPT
                             132C
                             132C  ;+++
                             132C  ; SHOW_RMS_OPT -- Print out current SET RMS, SHOW RMS options.
                             132C  ;
                             132C  ; Inputs:
                             132C  ;
                             132C  ;       RMS_DIS_OPT = Current RMS display options.
                             132C  ;       RMS_IFI     = IFI for which to display structures.
                             132C  ;
                             132C  ; Outputs:
                             132C  ;
                             132C  ;       None.
                             132C  ;---
                             132C
                             132C          .ENABL  LSB
                             132C  SHOW_RMS_OPT::
                      03FC   132C          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9>
                             132E          SKIP    1
                             1337          ALLOC   ALLOCSZ,R4              ; Create some stack space for translate
         00000000'EF    DD   1349          PUSHL   RMS_DIS_OPT             ; put longword bit argument on stack
             F4C5 CF    9F   134F          PUSHAB  OPT_CHR                 ; translation table for options bits
  00000000'EF    02     FB   1353          CALLS   #2,TRANSLATE_BITS       ; perform translation
                  54    DD   135A          PUSHL   R4                      ; put result back on stack
                             135C          PRINT   1,<-
                             135C  RMS Display Options: !AS>
  7E     00000004'EF    3C   1369          MOVZWL  RMS_IFI,-(SP)           ; put IFI value on stack
                  0F    13   1370          BEQL    10$                     ; if all IFI
                             1372          PRINT   1,<-
                             1372  Display RMS structures only for IFI=!UW.>
                  0D    11   137F          BRB     20$
                             1381  10$:    PRINT   0,<-
                             1381  Display RMS structures for all IFI values.>
                             138E  20$:    SKIP    1
                             1397          STATUS  SUCCESS
                        04   139E          RET
                             139F          .DSABL  LSB
                             139F
                             139F          .SBTTL  RMS_HANDLER -- SHOW RMS exception handler
                             139F  ;+++
                             139F  ; RMS_HANDLER -- Turn ACCVIO's into INVBLKTYP from SHOW RMS
                             139F  ;
                             139F  ; Inputs:
                             139F  ;
                             139F  ;       4(AP) = Pointer to SIGNAL arguments.
                             139F  ;       8(AP) = Pointer to MECHANISM arguments.
                             139F  ;
                             139F  ; Outputs:
                             139F  ;
                             139F  ;       Resignal if not ACCVIO.
                             139F  ;
                             139F  ;       Turn ACCVIO into INVBLKYP and unwind to caller of SHOW_RMS.
                             139F  ;---
                             139F
                             139F  RMS_HANDLER:
                      000C   139F          .WORD   ^M<R2,R3>
          52    04 AC   7D   13A1          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          .SBTTL  SHOW_IFAB -- Print out useful contents of IFAB
                            13BD
                            13BD   ;+++
                            13BD   ; SHOW_IFAB -- Print out useful contents of IFAB
                            13BD   ;
                            13BD   ; Inputs:
                            13BD   ;
                            13BD   ;       4(AP) = Address of local buffer containing IFAB
                            13BD   ;
                            13BD   ; Outputs:
                            13BD   ;
                            13BD   ;       IFAB information printed.
                            13BD   ;---
                            13BD
                            13BD          .ENABL  LSB
                            13BD   SHOW_IFAB::
                     001C   13BD          .WORD   ^M<R2,R3,R4>
                            13BF
                            13BF   ;
                            13BF   ; Do we need to show this structure?
                            13BF   ;
                            13BF
00000006'EF    02    D3     13BF          BITL    #OPT$M_IFB,RMS_DIS_OPT1   ; Show IFAB?
               03    12     13C6          BNEQ    2$                        ; if neq yes
             0450    31     13C8          BRW     100$                      ; if eql no
     53     04 AC    D0     13CB   2$:    MOVL    4(AP),R3                  ; R3 = address
                            13CF          SKIP    PAGE
     52     23 A3    9A     13D6          MOVZBL  IFB$B_ORGCASE(R3),R2      ; get the organization byte
   7E    F0E1 CF42   D0     13DA          MOVL    ORG_TBL[R2],-(SP)         ; put the descriptor on stack
   7E   00000016'EF  3C     13E0          MOVZWL  IFI,-(SP)                 ; put current IFI on stack
            FC A3    DD     13E7          PUSHL   -4(R3)                    ; get set up to print out address
                            13EA          PRINT   0,<-
                            13EA   =============>
                            13F7          PRINT   3,<-
                            13F7   IFAB Address: !XL              IFI: !XW        Organization: !AC>
                            1404          PRINT   0,<-
                            1404   =============                  ----            ------------->
                            1411          SKIP    1
                            141A          ALLOC   ALLOCSZ,R4                ; allocate data area for text str
               63    DD     142C          PUSHL   (R3)                      ; set up argument -- dev chr bits
          EBCE CF    9F     142E          PUSHAB  DEV_CHR                   ; address of table describing bits
00000000'EF    02    FB     1432          CALLS   #2,TRANSLATE_BITS         ; only two args to keep descrip around
               54    DD     1439          PUSHL   R4                        ; address of descriptor
               63    DD     143B          PUSHL   (R3)                      ; device characteristics bits
                            143D          PRINT   2,<-
                            143D   PRIMDEV:       !XL   !AS>
  6E   00000087 8F   D0     144A          MOVL    #ALLOCSZ,(SP)             ; reset size of allocated buffer
            04 A3    DD     1451          PUSHL   IFB$L_BKPBITS(R3)         ; get bookkeeping bits
          EDB8 CF    9F     1454          PUSHAB  BKP_CHR
00000000'EF    02    FB     1458          CALLS   #2,TRANSLATE_BITS
               54    DD     145F          PUSHL   R4                        ; buffer which contains string
            04 A3    DD     1461          PUSHL   IFB$L_BKPBITS(R3)         ; get the bits again
                            1464          PRINT   2,<-
                            1464   BKPBITS:       !XL   !AS>
     7E     08 A3    9A     1471          MOVZBL  IFB$B_BID(R3),-(SP)       ; block ID
     7E     09 A3    9A     1475          MOVZBL  IFB$B_BLN(R3),-(SP)       ; block length
                            1479          PRINT   2,<-
```

```
                              1479  BLN:              !XB       !-!8UB. -
                              1479  BID:              !XB       !-!8UB.>
         7E    0A A3    9A    1486          MOVZBL    IFB$B_MODE(R3),-(SP)        ; mode
         7E    0B A3    9A    148A          MOVZBL    IFB$B_EFN(R3),-(SP)         ; event flag number
                              148E          PRINT     2,<-
                              148E  EFN:              !XB                 -
                              148E  MODE:             !XB>
               14 A3    DD    149B          PUSHL     IFB$L_ASBADDR(R3)           ; ASB address
               0C A3    DD    149E          PUSHL     IFB$L_IOS(R3)               ; I/O status Block
                              14A1          PRINT     2,<-
                              14A1  IOS:              !XL                 -
                              14A1  ASBADDR:          !XL>
               18 A3    DD    14AE          PUSHL     IFB$L_ARGLST(R3)            ; User argument list
               10 A3    DD    14B1          PUSHL     IFB$L_IOS4(R3)              ; end of I/O status block
                              14B4          PRINT     2,<-
                              14B4  IOS4:             !XL                 -
                              14B4  ARGLST:           !XL>
         7E    20 A3    3C    14C1          MOVZWL    IFB$W_CHNL(R3),-(SP)        ; I/O channel number
               1C A3    DD    14C5          PUSHL     IFB$L_IRAB_LNK(R3)          ; link to IRAB list
                              14C8          PRINT     2,<-
                              14C8  IRAB_LNK:         !XL                 -
                              14C8  CHNL:             !XW>
    6E   00000087 8F    D0    14D5          MOVL      #ALLOCSZ,(SP)               ; reset size of allocated buffer
         7E    22 A3    9A    14DC          MOVZBL    IFB$B_FAC(R3),-(SP)         ; FAC bits for translate
            ECBC CF     9F    14E0          PUSHAB    FAC_CHR
00000000'EF    02       FB    14E4          CALLS     #2,TRANSLATE_BITS
                  54    DD    14EB          PUSHL     R4
         7E    22 A3    9A    14ED          MOVZBL    IFB$B_FAC(R3),-(SP)         ; FAC bits
                              14F1          PRINT     2,<-
                              14F1  FAC:              !XB       !AS>
         52    23 A3    9A    14FE          MOVZBL    IFB$B_ORGCASE(R3),R2
     7E    EFB9 CF42    D0    1502          MOVL      ORG_TBL[R2],-(SP)
         7E    23 A3    9A    1508          MOVZBL    IFB$B_ORGCASE(R3),-(SP) ; organization
                              150C          PRINT     2,<-
                              150C  ORGCASE:          !XB       !AC>
               3C A3    DD    1519          PUSHL     IFB$L_NWA_PTR(R3)           ; Network work area
               24 A3    DD    151C          PUSHL     IFB$L_LAST_FAB(R3)          ; last user FAB address
                              151F          PRINT     2,<-
                              151F  LAST_FAB:         !XL                 -
                              151F  NWA_PTR:          !XL>
         7E    2A A3    3C    152C          MOVZWL    IFB$W_ECHO_ISI(R3),-(SP); ISI for echoing SYS$INPUT
         7E    28 A3    3C    1530          MOVZWL    IFB$W_IFI(R3),-(SP)         ; saved IFI
                              1534          PRINT     2,<-
                              1534  IFI:              !XW                 -
                              1534  ECHO_ISI:         !XW>
               38 A3    DD    1541          PUSHL     IFB$L_FWA_PTR(R3)           ; pointer to FWA
               30 A3    DD    1544          PUSHL     IFB$L_JNLBDB(R3)            ; journaling BDB
                              1547          PRINT     2,<-
                              1547  JNLBDB:           !XL                 -
                              1547  FWA_PTR:          !XL>
         7E    48 A3    3C    1554          MOVZWL    IFB$L_DEVBUFSIZ(R3),-(SP) ; default dev buffer size
               40 A3    DD    1558          PUSHL     IFB$L_BDB_FLNK(R3)          ; BDB forward link
                              155B          PRINT     2,<-
                              155B  BDB_FLNK:         !XL                 -
                              155B  DEVBUFSIZ:        !XW       !-!8UW.>
         7E    4C A3    3C    1568          MOVZWL    IFB$W_RTDEQ(R3),-(SP)       ; default extend qty
               44 A3    DD    156C          PUSHL     IFB$L_BDB_BLNK(R3)          ; BDB backward link
```

```
                              156F           PRINT   2,<-
                              156F      BDB_BLNK:    !XL             -
                              156F      RTDEQ:       !XW       !-!8UW.>
  52   50 A3    04    00  EF  157C           EXTZV   #IFB$V_RFM,#4,IFB$B_RFMORG(R3),R2 ; get record format into R2
         7E   EF45 CF42  D0  1582           MOVL    RFM_TB[R2],-(SP)      ; put string on stack
            7E    50 A3  9A  1588           MOVZBL  IFB$B_RFMORG(R3),-(SP)  ; record format
                              158C           PRINT   2,<-
                              158C      RFMORG:      !XB        !AC>
       6E   00000087 8F  D0  1599           MOVL    #ALLOCSZ,(SP)         ; reset size of allocated buffer
            7E    51 A3  9A  15A0           MOVZBL  IFB$B_RAT(R3),-(SP)   ; record attributes for trans
               EC40 CF   9F  15A4           PUSHAB  RAT_CHR
        00000000'EF   02  FB  15A8           CALLS   #2,TRANSLATE_BITS
                    54   DD  15AF           PUSHL   R4
            7E    51 A3  9A  15B1           MOVZBL  IFB$B_RAT(R3),-(SP)   ; record attributes
                              15B5           PRINT   2,<-
                              15B5      RAT:         !XB        !AS>
                 54 A3   DD  15C2           PUSHL   IFB$L_HBK_DISK(R3)    ; highest block alloc (SWAPPED)
            7E    52 A3  3C  15C5           MOVZWL  IFB$W_LRL(R3),-(SP)   ; longest record length
                              15C9           PRINT   2,<-
                              15C9      LRL:         !XW       !-!8UW. -
                              15C9      HBK_DISK:    !XL>
                 58 A3   DD  15D6           PUSHL   IFB$L_EBK_DISK(R3)    ; end of file block (SWAPPED)
            7E    5C A3  3C  15D9           MOVZWL  IFB$W_FFB(R3),-(SP)   ; first free byte
                              15DD           PRINT   2,<-
                              15DD      FFB:         !XW       !-!8UW. -
                              15DD      EBK_DISK:    !XL>
            7E    5E A3  9A  15EA           MOVZBL  IFB$B_BKS(R3),-(SP)   ; bucket size
            7E    5F A3  9A  15EE           MOVZBL  IFB$B_FSZ(R3),-(SP)   ; header size
                              15F2           PRINT   2,<-
                              15F2      FSZ:         !XB       !-!8UB. -
                              15F2      BKS:         !XB       !-!8UB.>
            7E    60 A3  3C  15FF           MOVZWL  IFB$W_MRS(R3),-(SP)   ; maximum record size
            7E    62 A3  3C  1603           MOVZWL  IFB$W_DEQ(R3),-(SP)   ; default extend size
                              1607           PRINT   2,<-
                              1607      DEQ:         !XW       !-!8UW. -
                              1607      MRS:         !XW       !-!8UW.>
            7E    64 A3  3C  1614           MOVZWL  IFB$W_GBC(R3),-(SP)   ; global buff count
                              1618           PRINT   1,<-
                              1618      GBC:         !XW       !-!8UW.>
                 70 A3   DD  1625           PUSHL   IFB$L_HBK(R3)         ; high block alloc
                              1628           PRINT   1,<-
                              1628      HBK:         !XL       !-!8UL.>
                 74 A3   DD  1635           PUSHL   IFB$L_EBK(R3)         ; end-of-file
                              1638           PRINT   1,<-
                              1638      EBK:         !XL       !-!8UL.>
                 6C A3   DD  1645           PUSHL   IFB$L_LOCK_BDB(R3)    ; lockbdb address
                 6C A3   DD  1648           PUSHL   IFB$L_RNS_LEN(R3)     ; multi-use field
                              164B           PRINT   2,<-
                              164B      RNS_LEN:     !XL             -
                              164B      LOCK_BDB:    !XL>
          7E    0084 C3  3C  1658           MOVZWL  IFB$W_AVLCL(R3),-(SP)  ; local buffers available
                 78 A3   DD  165D           PUSHL   IFB$L_SFSB_PTR(R3)    ; pointer to shared file synch block
                              1660           PRINT   2,<-
                              1660      SFSB_PTR:    !XL             -
                              1660      AVLCL:       !XW       !-!8UW.>
                              166D
          7E    0086 C3  3C  166D           MOVZWL  IFB$W_AVGBPB(R3),-(SP)  ; global pointer blocks available
```

```
            7C A3   DD 1672           PUSHL   IFB$L_GBSB_PTR(R3)         ; pointer to global buffer sync block
                       1675           PRINT   2,<-
                       1675 GBSB_PTR:         !XL                        -
                       1675 AVGBPB:           !XW       !-!8UW.>
                       1682
          00A4 C3   DD 1682           PUSHL   IFB$L_RJB(R3)              ; pointer to journaling block
          0088 C3   DD 1686           PUSHL   IFB$L_GBH_PTR(R3)          ; pointer to global header
                       168A           PRINT   2,<-
                       168A GBH_PTR:          !XL                        -
                       168A RJB_PTR:          !XL>
6E    00000087 8F   D0 1697           MOVL    #ALLOCSZ,(SP)              ; reset size of allocated buffer
    7E    00A0 C3   9A 169E           MOVZBL  IFB$B_JNLFLG(R3),-(SP)     ; journal flag bits for translate
          EC91 CF   9F 16A3           PUSHAB  JNLFLG_CHR
00000000'EF     02  FB 16A7           CALLS   #2,TRANSLATE_BITS
               54   DD 16AE           PUSHL   R4
    7E    00A0 C3   9A 16B0           MOVZBL  IFB$B_JNLFLG(R3),-(SP)     ; journal flag bits
                       16B5           PRINT   2,<-
                       16B5 JNLFLGS:          !XB           !AS>
6E    00000087 8F   D0 16C2           MOVL    #ALLOCSZ,(SP)              ; reset size of allocated buffer
    7E    00A1 C3   9A 16C9           MOVZBL  IFB$B_RECVRFLGS(R3),-(SP); recover flag bits for translate
          EC46 CF   9F 16CE           PUSHAB  RECFLG_CHR
00000000'EF     02  FB 16D2           CALLS   #2,TRANSLATE_BITS
               54   DD 16D9           PUSHL   R4
    7E    00A1 C3   9A 16DB           MOVZBL  IFB$B_RECVRFLGS(R3),-(SP); recover flag bits
                       16E0           PRINT   2,<-
                       16E0 RECVRFLGS:        !XB           !AS>
6E    00000087 8F   D0 16ED           MOVL    #ALLOCSZ,(SP)              ; reset size of allocated buffer
    7E    00A2 C3   9A 16F4           MOVZBL  IFB$B_JNLFLG2(R3),-(SP)    ; journal flag2 bits for translate
          F4A7 CF   9F 16F9           PUSHAB  JNLFLG2_CHR
00000000'EF     02  FB 16FD           CALLS   #2,TRANSLATE_BITS
               54   DD 1704           PUSHL   R4
    7E    00A2 C3   9A 1706           MOVZBL  IFB$B_JNLFLG2(R3),-(SP)    ; journal flag bits
                       170B           PRINT   2,<-
                       170B JNLFLGS2:         !XB           !AS>
          0080 C3   DD 1718           PUSHL   IFB$L_PAR_LOCK_ID(R3)      ; parent lock id
            34 A3   DD 171C           PUSHL   IFB$L_EXTJNLBUF(R3)        ; pointer to extend MJB
                       171F           PRINT   2,<-
                       171F EXTJNL_PTR:       !XL                        -
                       171F PAR_LOCK_ID:      !XL>
                       172C
                       172C ;
                       172C ; Case to organization dependent code
                       172C ;
                       172C
 02    00    23 A3   8F 172C           CASEB   IFB$B_ORGCASE(R3),#0,#2    ; branch to org dependant code
               000E' 1731 5$:         .WORD   10$-5$                     ; sequential
               004C' 1733             .WORD   15$-5$                     ; relative
               004C' 1735             .WORD   15$-5$                     ; indexed
                       1737           STATUS  NOTVALID
               04      173E           RET
                       173F
                       173F ;
                       173F ; Sequential organization dependant code
                       173F ;
                       173F
6E    00000087 8F   D0 173F 10$:      MOVL    #ALLOCSZ,(SP)              ; reset size of allocated buffer
          008C C3   DD 1746           PUSHL   IFB$L_AS_DEV(R3)           ; get set up to trans char bits
```

```
            E8B2 CF   9F  174A          PUSHAB  DEV_CHR
       00000000'EF 02 FB  174E          CALLS   #2,TRANSLATE_BITS
                 54   DD  1755          PUSHL   R4
            008C C3   DD  1757          PUSHL   IFB$L_AS_DEV(R3)          ; assigned dev characteristic bits
                          175B          PRINT   2,<-
                          175B AS_DEV:          !XL     !AS>
       7E   0094 C3   3C  1768          MOVZWL  IFB$L_ASDEVBSIZ(R3),-(SP) ; assigned dev buffer size
                          176D          PRINT   1,<-
                          176D ASDEVBSIZ:       !XW     !-!8UW.>
               009E   31  177A          BRW     100$
                          177D
                          177D ;
                          177D ; Code common to relative and indexed, but not sequential
                          177D ;
                          177D
            0098 C3   DD  177D 15$:     PUSHL   IFB$L_BLBFLNK(R3)
                          1781          PRINT   1,<-
                          1781 BLBFLNK:         !XL>
            009C C3   DD  178E          PUSHL   IFB$L_BLBBLNK(R3)
                          1792          PRINT   1,<-
                          1792 BLBBLNK:         !XL>
                          179F
   02    01    23 A3  8F  179F          CASEB   IFB$B_ORGCASE(R3),#1,#2
                 0004'    17A4 16$:     .WORD   20$-16$                   ; relative
                 001C'    17A6          .WORD   30$-16$                   ; indexed
                          17A8
                          17A8 ;
                          17A8 ; Relative organization dependant code
                          17A8 ;
                          17A8
            00B0 C3   DD  17A8 20$:     PUSHL   IFB$L_DVBN(R3)            ; first data bucket VBN
            00AC C3   DD  17AC          PUSHL   IFB$L_MRN(R3)             ; maximum record number
                          17B0          PRINT   2,<-
                          17B0 MRN:             !XL                     -
                          17B0 DVBN:            !XL!-!8UL.>
               005B   31  17BD          BRW     100$
                          17C0
                          17C0 ;
                          17C0 ; Indexed organization dependant code
                          17C0 ;
                          17C0
       7E   00B0 C3   9A  17C0 30$:     MOVZBL  IFB$B_AVBN(R3),-(SP)      ; VBN  of 1st area descriptor
            00AC C3   DD  17C5          PUSHL   IFB$L_IDX_PTR(R3)         ; pointer to primary key idx desc
                          17C9          PRINT   2,<-
                          17C9 IDX_PTR:         !XL                     -
                          17C9 AVBN:            !XB          !-!8UB.>
       7E   00B2 C3   9A  17D6          MOVZBL  IFB$B_NUM_KEYS(R3),-(SP) ; number of keys in file
       7E   00B1 C3   9A  17DB          MOVZBL  IFB$B_AMAX(R3),-(SP)     ; total number of area descriptors
                          17E0          PRINT   2,<-
                          17E0 AMAX:            !XB       !-!8UB. -
                          17E0 NUM_KEYS:        !XB       !-!8UB.>
       7E   00B4 C3   3C  17ED          MOVZWL  IFB$W_KBUFSZ(R3),-(SP) ; key buffer size
       7E   00B3 C3   9A  17F2          MOVZBL  IFB$B_UBUFSZ(R3),-(SP) ; update buffer size
                          17F7          PRINT   2,<-
                          17F7 UBUFSZ:          !XB       !-!8UB. -
                          17F7 KBUFSZ:          !XB       !-!8UB.>
       7E   00B6 C3   9A  1804          MOVZBL  IFB$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
               04  1822  1374            RET
                   1823  1375            .DSABL  LSB
```

```
                       1823          .SBTTL  SHOW_IDX
                       1823
                       1823   ;+++
                       1823   ; SHOW_IDX -- Display useful contents of index descriptor.
                       1823   ;
                       1823   ; Inputs:
                       1823   ;
                       1823   ;         4(AP) = Address of buffer containing index descriptor.
                       1823   ;
                       1823   ; Outputs:
                       1823   ;
                       1823   ;         None.
                       1823   ;---
                       1823
                       1823          .ENABL  LSB
                       1823   SHOW_IDX:
                 001C  1823          .WORD   ^M<R2,R3,R4>
 53     04 AC     D0   1825          MOVL    4(AP),R3                ; get address of buffer into R3
                       1829          ENSURE  23
                       1841          SKIP    2
        FC A3     DD   184A          PUSHL   -4(R3)                  ; virtual address of IDX
                       184D          PRINT   1,<-
                       184D   IDX Address: !XL>
                       185A          PRINT   0,<-
                       185A   ------------->
                       1867          SKIP    1
 7E     08 A3     9A   1870          MOVZBL  IDX$B_BID(R3),-(SP)
           63     DD   1874          PUSHL   IDX$L_IDXFL(R3)
                       1876          PRINT   2,<-
                       1876   IDXFL:         !XL               -
                       1876   BID:           !XB     !-!8UB.>
 7E     09 A3     9A   1883          MOVZBL  IDX$B_BLN(R3),-(SP)
 7E     12 A3     9A   1887          MOVZBL  IDX$B_IANUM(R3),-(SP)
                       188B          PRINT   2,<-
                       188B   IANUM:         !XB     !-!8UB. -
                       188B   BLN:           !XB     !-!8UB.>
 7E     13 A3     9A   1898          MOVZBL  IDX$B_LANUM(R3),-(SP)
 7E     14 A3     9A   189C          MOVZBL  IDX$B_DANUM(R3),-(SP)
                       18A0          PRINT   2,<-
                       18A0   DANUM:         !XB     !-!8UB. -
                       18A0   LANUM:         !XB     !-!8UB.>
 7E     15 A3     9A   18AD          MOVZBL  IDX$B_ROOTLEV(R3),-(SP)
 7E     16 A3     9A   18B1          MOVZBL  IDX$B_IDXBKTSZ(R3),-(SP)
                       18B5          PRINT   2,<-
                       18B5   IDXBKTSZ:      !XB     !-!8UB. -
                       18B5   ROOTLEV:       !XB     !-!8UB.>
        18 A3     DD   18C2          PUSHL   IDX$L_ROOTVBN(R3)
 7E     17 A3     9A   18C5          MOVZBL  IDX$B_DATBKTSZ(R3),-(SP)
                       18C9          PRINT   2,<-
                       18C9   DATBKTSZ:      !XB     !-!8UB. -
                       18C9   ROOTVBN:       !XL!-!8UL.>
                       18D6
                       18D6   ;
                       18D6   ; Translate and dump the flags
                       18D6   ;
                       18D6
                       18D6
```

```
                      18D6               ALLOC   ALLOCSZ,R4
        7E  1C A3  9A 18E8               MOVZBL  IDX$B_FLAGS(R3),-(SP)
            21 A3  95 18EC               TSTB    IDX$B_KEYREF(R3)           ; is it primary key?
               06  12 18EF               BNEQ    5$                         ; if neq no
          ECEF CF  9F 18F1               PUSHAB  IDX_CHR0
               04  11 18F5               BRB     7$
          ECB1 CF  9F 18F7      5$:      PUSHAB  IDX_CHR1                   ; seconday key, use other table
 00000000'EF   02  FB 18FB      7$:      CALLS   #2,TRANSLATE_BITS
               54  DD 1902               PUSHL   R4
        7E  1C A3  9A 1904               MOVZBL  IDX$B_FLAGS(R3),-(SP)
                      1908               PRINT   2,<-
                      1908      FLAGS:           !XB         !AS>
        52  1D A3  9A 1915               MOVZBL  IDX$B_DATATYPE(R3),R2
   7E  ECF6 CF42   D0 1919               MOVL    IDX_TBL[R2],-(SP)
        7E  1D A3  9A 191F               MOVZBL  IDX$B_DATATYPE(R3),-(SP)
                      1923               PRINT   2,<-
                      1923      DATATYPE:        !XB         !AC>
        7E  1F A3  9A 1930               MOVZBL  IDX$B_NULLCHAR(R3),-(SP)
               01  DD 1934               PUSHL   #1
        7E  1E A3  9A 1936               MOVZBL  IDX$B_SEGMENTS(R3),-(SP)
                      193A               PRINT   3,<-
                      193A      SEGMENTS:        !XB         !-!8UB. -
                      193A      NULLCHR:         !XB         "!AF">
        7E  20 A3  9A 1947               MOVZBL  IDX$B_KEYSZ(R3),-(SP)
        7E  21 A3  9A 194B               MOVZBL  IDX$B_KEYREF(R3),-(SP)
                      194F               PRINT   2,<-
                      194F      KEYREF:          !XB         !-!8UB. -
                      194F      KEYSZ:           !XB         !-!8UB.>
        7E  22 A3  3C 195C               MOVZWL  IDX$W_MINRECSZ(R3),-(SP)
        7E  24 A3  3C 1960               MOVZWL  IDX$W_IDXFILL(R3),-(SP)
                      1964               PRINT   2,<-
                      1964      IDXFILL:         !XW         !-!8UW. -
                      1964      MINRECSZ:        !XW         !-!8UW.>
        7E  26 A3  3C 1971               MOVZWL  IDX$W_DATFILL(R3),-(SP)
                      1975               PRINT   1,<-
                      1975      DATFILL:         !XW         !-!8UW.>
        52  28 A3  9A 1982               MOVZBL  IDX$B_IDXBKTYP(R3),R2
   7E  ECA9 CF42   D0 1986               MOVL    IDX_TBL1[R2],-(SP)
        7E  28 A3  9A 198C               MOVZBL  IDX$B_IDXBKTYP(R3),-(SP)
                      1990               PRINT   2,<-
                      1990      IDXBKTYP:        !XB         !AC>
        52  29 A3  9A 199D               MOVZBL  IDX$B_DATBKTYP(R3),R2
   7E  EC8E CF42   D0 19A1               MOVL    IDX_TBL1[R2],-(SP)
        7E  29 A3  9A 19A7               MOVZBL  IDX$B_DATBKTYP(R3),-(SP)
                      19AB               PRINT   2,<-
                      19AB      DATBKTYP:        !XB         !AC>
                      19B8
                      19B8      ;
                      19B8      ; Display the specific segments
                      19B8      ;
                      19B8
        52  2C A3  3E 19B8               MOVAW   IDX$W_POSITION(R3),R2      ; get starting address of segments
        54  1E A3  9A 19BC               MOVZBL  IDX$B_SEGMENTS(R3),R4      ; get number of segments
     54     54 02  78 19C0               ASHL    #2,R4,R4                   ; multiply by 4 (1 longword / segment)
            54 52  C0 19C4               ADDL2   R2,R4                      ; R4 = address beyond last segment
                      19C7               SKIP    1
                      19D0               PRINT   0,<-
```

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

```
                        1A28         .SBTTL  SHOW_FWA
                        1A28
                        1A28   ;+++
                        1A28   ; SHOW_FWA -- Show information in FWA.
                        1A28   ;
                        1A28   ;  Inputs:
                        1A28   ;
                        1A28   ;       4(AP) = Address of buffer containing FWA
                        1A28   ;
                        1A28   ;  Outputs:
                        1A28   ;
                        1A28   ;       Information printed out for FWA.
                        1A28   ;--
                        1A28
                        1A28         .ENABL  LSB
                        1A28   SHOW_FWA:
                  003C  1A28         .WORD   ^M<R2,R3,R4,R5>
       53   04 AC   D0  1A2A         MOVL    4(AP),R3
                        1A2E         SKIP    PAGE
            FC A3   DD  1A35         PUSHL   -4(R3)
                        1A38         PRINT   1,<-
                        1A38   FWA Address: !XL>
                        1A45         PRINT   0,<-
                        1A45   ------------>
                        1A52   ;
                        1A52   ; Form value which will be used to adjust descriptors in the
                        1A52   ; FWA.  If  x=address in FWA, y=address of FWA in users image, z=address of
                        1A52   ; FWA in SDA, then address of descriptor in SDA = x-y+z.  This is also
                        1A52   ; x+(-y+z) = x+(z-y).  We now form (z-y) in R5.
                        1A52   ;
                        1A52
       55      53   D0  1A52         MOVL    R3,R5                  ; copy z (address of FWA in SDA)
       55   FC A3   C2  1A55         SUBL2   -4(R3),R5              ; form z-y
                        1A59         ALLOC   ALLOCSZ,R4
                        1A6B   ;
                        1A6B   ; Now dump the bit fields
                        1A6B   ;
                        1A6B
               63   DD  1A6B         PUSHL   FWA$Q_FLAGS(R3)
                        1A6D         PRINT   1,<-
                        1A6D   FLAGS:        !XL>
            04 A3   DD  1A7A         PUSHL   FWA$Q_FLAGS+4(R3)
                        1A7D         PRINT   1,<-
                        1A7D                 !XL>
                        1A8A         SKIP    1
                        1A93         FWABITS PASS
                        1ABA         FWABITS FLD
                        1AE3         FWABITS WILD
                        1B0C         FWABITS PARSE
                        1B35         FWABITS DIR
                        1B5E         FWABITS DIRWC
                        1B87         FWABITS LN
                        1BB0         FWABITS SL
                        1BD9         SKIP    1
                        1BE2
                        1BE2   ;
                        1BE2   ; More normal dumping of intermediate FWA cells
```

```
                   1BE2  ;
                   1BE2
        0B A3  9F  1BE2            PUSHAB  FWA$B_ROOTERM(R3)           ; address of string
           01  DD  1BE5            PUSHL   #1                          ; length of string
   7E   0B A3  9A  1BE7            MOVZBL  FWA$B_ROOTERM(R3),-(SP)     ; value of string
        0A A3  9F  1BEB            PUSHAB  FWA$B_DIRTERM(R3)           ; address of string
           01  DD  1BEE            PUSHL   #1                          ; length of string
   7E   0A A3  9A  1BF0            MOVZBL  FWA$B_DIRTERM(R3),-(SP)     ; value of string
                   1BF4            PRINT   6,<-
                   1BF4  DIRTERM:          !XB      "!AF"              -
                   1BF4  ROOTERM:          !XB      "!AF">
                   1C01
   7E   0E A3  9A  1C01            MOVZBL  FWA$W_ESCIFI(R3),-(SP)
        0C A3  DD  1C05            PUSHL   FWA$L_ESCSTRING(R3)
                   1C08            PRINT   2,<-
                   1C08  ESCSTRING:        !XL                -
                   1C08  ESCIFI:           !XW>
   7E 009E C3  3C  1C15            MOVZWL  FWA$W_XLTSIZ(R3),-(SP)
   7E 009D C3  9A  1C1A            MOVZBL  FWA$B_XLTMODE(R3),-(SP)
                   1C1F            PRINT   2,<-
                   1C1F  XLTMODE:          !XB                         -
                   1C1F  XLTSIZ:           !XW!-!8UW.>
                   1C2C            PRINTD  FWA$Q_,FIB
        30 A3  DD  1C3F            PUSHL   FWA$L_DIRBDB(R3)
   7E 009C C3  9A  1C42            MOVZBL  FWA$B_BUFFLG(R3),-(SP)
                   1C47            PRINT   2,<-
                   1C47  BUFFLG:           !XB                         -
                   1C47  DIRBDB:           !XL>
                   1C54            PRINTD  FWA$Q_,LOGNAM
        34 A3  DD  1C69            PUSHL   FWA$L_LOOKUP(R3)
        28 A3  DD  1C6C            PUSHL   FWA$L_UIC(R3)
                   1C6F            PRINT   2,<-
                   1C6F  UIC:              !XL                -
                   1C6F  LOOKUP:           !XL>
   7E   2C A3  3C  1C7C            MOVZWL  FWA$W_PRO(R3),-(SP)
        38 A3  DD  1C80            PUSHL   FWA$L_DEVNODADR(R3)
                   1C83            PRINT   2,<-
                   1C83  DEVNODADR:        !XL                -
                   1C83  PRO:              !XW>
                   1C90            PRINTD  FWA$Q_,DIR
   7E   44 A3  3C  1CA3            MOVZWL  FWA$W_UCHAR(R3),-(SP)
   7E 00C3 C3  9A  1CA7            MOVZBL  FWA$B_LEVEL(R3),-(SP)
                   1CAC            PRINT   2,<-
                   1CAC  LEVEL:            !XB!-!8UB.                  -
                   1CAC  UCHAR:            !XW>
   7E   2F A3  9A  1CB9            MOVZBL  FWA$B_SUBNODCNT(R3),-(SP)
   7E   2E A3  9A  1CBD            MOVZBL  FWA$B_DIRLEN(R3),-(SP)
                   1CC1            PRINT   2,<-
                   1CC1  DIRLEN:           !XB!-!8UB.                  -
                   1CC1  SUBNODCNT:        !XB!-!8UB.>
        4C A3  DD  1CCE            PUSHL   FWA$L_SWB_PTR(R3)
                   1CD1            PRINT   1,<-
                   1CD1  SWB_PTR:          !XL>
      00AC C3  DD  1CDE            PUSHL   FWA$L_SLB_PTR(R3)
      00A8 C3  DD  1CE2            PUSHL   FWA$L_SLBH_PTR(R3)
                   1CE6            PRINT   2,<-
                   1CE6  SLBH_PTR:         !XL                -
```

```
                  1CE6  SLB_PTR:          !XL>
00B4 C3   DD      1CF3            PUSHL   FWA$L_SLBH_BLINK(R3)
00B0 C3   DD      1CF7            PUSHL   FWA$L_SLBH_FLINK(R3)
                  1CFB            PRINT   2,<-
                  1CFB  SLBH_FLINK:       !XL                    -
                  1CFB  SLBH_BLINK:       !XL>
                  1D08
                  1D08  ;
                  1D08  ; Dump Descriptors
                  1D08  ;
                  1D08
                  1D08            SKIP    1
                  1D11            DUMPDF  NODE
                  1D38            DUMPDF  DEVICE
                  1D5F            DUMPDF  CONCEAL_DEV
                  1D86            DUMPDF  CDIR1
                  1DAD            DUMPDF  CDIR2
                  1DD4            DUMPDF  CDIR3
                  1DFB            DUMPDF  CDIR4
                  1E22            DUMPDF  CDIR5
                  1E49            DUMPDF  CDIR6
                  1E70            DUMPDF  CDIR7
                  1E97            DUMPDF  CDIR8
                  1EBE            DUMPDF  DIR1
                  1EE5            DUMPDF  DIR2
                  1F0C            DUMPDF  DIR3
                  1F33            DUMPDF  DIR4
                  1F5A            DUMPDF  DIR5
                  1F81            DUMPDF  DIR6
                  1FA8            DUMPDF  DIR7
                  1FCF            DUMPDF  DIR8
                  1FF6            DUMPDF  NAME
                  201D            DUMPDF  TYPE
                  2044            DUMPDF  RNS
                  206B            DUMPDF  VERSION
                  2092            DUMPDF  SHRFIL
                  20B9
                  20B9            SKIP    1
                  20C2            PRINTD  FWA$Q_,NODE1
                  20D7            PRINTD  FWA$Q_,NODE2
                  20EC            PRINTD  FWA$Q_,NODE3
                  2101            PRINTD  FWA$Q_,NODE4
                  2116            PRINTD  FWA$Q_,NODE5
                  212B            PRINTD  FWA$Q_,NODE6
                  2140            PRINTD  FWA$Q_,NODE7
                  2155            PRINTD  FWA$Q_,NODE8
                  216A
                  216A            SKIP    1
                  2173            DUMPDF  AIJNL
                  219A            DUMPDF  ATJNL
                  21C1            DUMPDF  BIJNL
                  21E8
                  21E8  ;
                  21E8  ; Additional non-descriptor cells
                  21E8  ;
                  21E8
                  21E8  100$:     STATUS  SUCCESS
```

```
                          04  21EF          RET
                              21F0          .DSABL  LSB
                              21F0
                              21F0  ;+++
                              21F0  ; FWADD -- Dump descriptors in FWA
                              21F0  ;
                              21F0  ; Inputs:
                              21F0  ;
                              21F0  ;       R2=Address of descriptor.
                              21F0  ;       R5=Offset to add to address for local descriptor pointer.
                              21F0  ;
                              21F0  ; Outputs:
                              21F0  ;
                              21F0  ;       String printed if possible.
                              21F0  ;---
                              21F0
                              21F0          .ENABL  LSB
                        003C  21F0  FWADD:  .WORD   ^M<R2,R3,R4,R5>
                  04 A2   DD  21F2          PUSHL   4(R2)                  ; see if descriptor points into FWA
  6E      000002F6'EF     C2  21F5          SUBL2   FWA-4,(SP)             ; offset from FWA
  8E      0000093C 8F     D1  21FC          CMPL    #FWA$C_BLN,(SP)+       ; within FWA?
                     0B   1F  2203          BLSSU   10$                    ; if blssu no
                              2205
                              2205  ;
                              2205  ; Within FWA -- dump more or less normally
                              2205  ;
                              2205
                  04 A2   DD  2205          PUSHL   4(R2)                  ; address in process
              6E     55   C0  2208          ADDL2   R5,(SP)                ; correction previously calc for FWA
                     62   DD  220B          PUSHL   (R2)                   ; length
                   002E   31  220D          BRW     20$                    ; print statement
                              2210
                              2210  ;
                              2210  ; See if we can find it...
                              2210  ;
                              2210
                              2210  10$:    ALLOC   ALLOCSZ,R4             ; make some space for it
              64     62   D1  2222          CMPL    (R2),(R4)              ; is it enough space?
                     24   1A  2225          BGTRU   30$                    ; if gtr no
                              2227          GETMEM  @4(R2),@4(R4),(R2)     ; get memory if possible
                  12 50   E9  2236          BLBC    R0,30$                 ; if lbc then couldn't get it
                  04 A4   DD  2239          PUSHL   4(R4)                  ; push address of string
                     62   DD  223C          PUSHL   (R2)                   ; length too
                              223E
                              223E  20$:    PRINT   2,<-
                              223E                          !AF>
                              224B
                          04  224B  30$:    RET
                              224C          .DSABL  LSB
```

```
                                   224C          .SBTTL  SHOW_CCB
                                   224C
                                   224C  ;+++
                                   224C  ; SHOW_CCB -- Show information from CCB.
                                   224C  ;
                                   224C  ; Inputs:
                                   224C  ;
                                   224C  ;       4(AP) = Address of buffer containing CCB
                                   224C  ;
                                   224C  ; Outputs:
                                   224C  ;
                                   224C  ;       None.
                                   224C  ;---
                                   224C
                                   224C          .ENABL  LSB
                                   224C  SHOW_CCB:
                            001C   224C          .WORD   ^M<R2,R3,R4>
 00000006'EF    00000080 8F   D3   224E          BITL    #OPT$M_CCB,RMS_DIS_OPT1 ; show ccb's?
                          03  12   2259          BNEQ    1$                      ; if neq yes
                        00AE  31   225B          BRW     90$                     ; if eql no
            53       04 AC    D0   225E  1$:     MOVL    4(AP),R3                ; get ccb buffer address
                                   2262          SKIP    PAGE
                     FC A3    DD   2269          PUSHL   -4(R3)                  ; address in dump of CCB
                                   226C          PRINT   1,<-
                                   226C  CCB Address: !XL>
                                   2279          PRINT   0,<-
                                   2279  ------------->
                                   2286          SKIP    1
                     04 A3    DD   228F          PUSHL   CCB$L_WIND(R3)          ; address of window control block
                        63    DD   2292          PUSHL   CCB$L_UCB(R3)           ; UCB address of the device
                                   2294          PRINT   2,<-
                                   2294  UCB:            !XL             -
                                   2294  WIND:           !XL>
                                   22A1          ALLOC   ALLOCSZ,R4
            7E       08 A3    9A   22B3          MOVZBL  CCB$B_STS(R3),-(SP)
                   E395 CF    9F   22B7          PUSHAB  CCB_CHR
       00000000'EF      02    FB   22BB          CALLS   #2,TRANSLATE_BITS
                        54    DD   22C2          PUSHL   R4
            7E       08 A3    9A   22C4          MOVZBL  CCB$B_STS(R3),-(SP)
                                   22C8          PRINT   2,<-
                                   22C8  STS:            !XB             !AS>
            52       09 A3    9A   22D5          MOVZBL  CCB$B_AMOD(R3),R2
                  52    05    D1   22D9          CMPL    #5,R2
                        03    14   22DC          BGTR    30$
                  52    05    D0   22DE          MOVL    #5,R2
       7E      E37A CF42      D0   22E1  30$:    MOVL    AMODE_TBL[R2],-(SP)
            7E       09 A3    9A   22E7          MOVZBL  CCB$B_AMOD(R3),-(SP)
                                   22EB          PRINT   2,<-
                                   22EB  AMOD:           !XB             !AC>
                     0C A3    DD   22F8          PUSHL   CCB$L_DIRP(R3)
            7E       0A A3    3C   22FB          MOVZWL  CCB$W_IOC(R3),-(SP)
                                   22FF          PRINT   2,<-
                                   22FF  IOC:            !XW     !-!8UW. -
                                   22FF  DIRP:           !XL>
                                   230C  90$:    STATUS  SUCCESS
                        04         2313          RET
                                   2314          .DSABL  LSB
```

```
                                   2314        .SBTTL  SHOW_WCB
                                   2314
                                   2314  ;+++
                                   2314  ; SHOW_WCB -- Show information from Window Control Block.
                                   2314  ;
                                   2314  ; Inputs:
                                   2314  ;
                                   2314  ;       4(AP) = Address of buffer containing WCB.
                                   2314  ;
                                   2314  ; Outputs:
                                   2314  ;
                                   2314  ;       None.
                                   2314  ;---
                                   2314          .ENABL  LSB
                                   2314  SHOW_WCB:
                           003C    2314          .WORD   ^M<R2,R3,R4,R5>
 00000006'EF    00000100 8F   D3   2316          BITL    #OPT$M_WCB,RMS_DIS_OPT1  ; display wcb's?
                         03   12   2321          BNEQ    1$                       ; if neq yes
                       01BA   31   2323          BRW     90$                      ; if eql no
             53      04 AC    D0   2326  1$:     MOVL    4(AP),R3                 ; address of buffer with WCB
                                   232A          ENSURE  19
                                   2342          SKIP    2
                     FC A3    DD   234B          PUSHL   -4(R3)                   ; address in dump of WCB
                                   234E          PRINT   1,<-
                                   234E  WCB Address: !XL>
                                   235B          PRINT   0,<-
                                   235B  ------------>
                                   2368          SKIP    1
             7E      08 A3    3C   2371          MOVZWL  WCB$W_SIZE(R3),-(SP)
                         63   DD   2375          PUSHL   WCB$L_WLFL(R3)
                                   2377          PRINT   2,<-
                                   2377  WLFL:           !XL             -
                                   2377  SIZE:           !XW     !-!8UW.>
             7E      0A A3    9A   2384          MOVZBL  WCB$B_TYPE(R3),-(SP)
                     04 A3    DD   2388          PUSHL   WCB$L_WLBL(R3)
                                   238B          PRINT   2,<-
                                   238B  WLBL:           !XL             -
                                   238B  TYPE:           !XB>
                                   2398          ALLOC   ALLOCSZ,R4
             7E      0B A3    9A   23AA          MOVZBL  WCB$B_ACCESS(R3),-(SP)
                    E2C6 CF   9F   23AE          PUSHAB  WCBSHR_CHR
         00000000'EF     02   FB   23B2          CALLS   #2,TRANSLATE_BITS
                         54   DD   23B9          PUSHL   R4
             7E      0B A3    9A   23BB          MOVZBL  WCB$B_ACCESS(R3),-(SP)
                                   23BF          PRINT   2,<-
                                   23BF  ACCESS:         !XB             !AS>
                  10 A3       DD   23CC          PUSHL   WCB$L_ORGUCB(R3)
             0B A3   08       93   23CF          BITB    #WCB$M_SHRWCB,WCB$B_ACCESS(R3) ; is this shared WCB?
                     12       12   23D3          BNEQ    10$                      ; if neq yes
                                   23D5
                                   23D5  ;
                                   23D5  ; Unshared WCB -- print PID of owner
                                   23D5  ;
                                   23D5
                  0C A3       DD   23D5          PUSHL   WCB$L_PID(R3)
                                   23D8          PRINT   2,<-
                                   23D8  PID:            !XL             -
```

```
                                 23D8  ORGUCB:            !XL>
                   11      11    23E5            BRB      20$
                                 23E7
                                 23E7  ;
                                 23E7  ; Shared WCB -- print reference count of WCB
                                 23E7  ;
                                 23E7
          7E    0E A3      3C    23E7  10$:      MOVZWL   WCB$W_REFCNT(R3),-(SP)
                                 23EB            PRINT    2,<-
                                 23EB  REFCNT:            !XW      !-!8UW. -
                                 23EB  ORGUCB:            !XL>
                                 23F8
    6E    00000087 8F      D0    23F8  20$:      MOVL     #ALLOCSZ,(SP)              ; reset buffer on stack
          7E    14 A3      3C    23FF            MOVZWL   WCB$W_ACON(R3),-(SP)
              E2A1 CF      9F    2403            PUSHAB   WCBACC_CHR
   00000000'EF     02      FB    2407            CALLS    #2,TRANSLATE_BITS
                   54      DD    240E            PUSHL    R4
          7E    14 A3      3C    2410            MOVZWL   WCB$W_ACON(R3),-(SP)
                                 2414            PRINT    2,<-
                                 2414  ACON:              !XW          !AS>
                18 A3      DD    2421            PUSHL    WCB$L_FCB(R3)              ; FCB address
          7E    16 A3      3C    2424            MOVZWL   WCB$W_NMAP(R3),-(SP)       ; number of map pointers
                                 2428            PRINT    2,<-
                                 2428  NMAP:              !XW      !-!8UW. -
                                 2428  FCB:               !XL>
                1C A3      DD    2435            PUSHL    WCB$L_RVT(R3)
                2C A3      DD    2438            PUSHL    WCB$L_STVBN(R3)
                                 243B            PRINT    2,<-
                                 243B  STVBN:             !XL!-!8UL.      -
                                 243B  RVT:               !XL>
                                 2448
                                 2448  ;
                                 2448  ; Print out all mapping pointers that fit into our buffer.
                                 2448  ;
                                 2448
                                 2448            SKIP     1
                                 2451            PRINT    0,<-
                                 2451             VBN      RVN       Starting LBN                 Count>
                                 245E            PRINT    0,<-
                                 245E             ---      ---       ------------                 ----->
                                 246B            SKIP     1
          55    2C A3      D0    2474            MOVL     WCB$L_STVBN(R3),R5         ; initial VBN for this window
          52    30 A3      9E    2478            MOVAB    WCB$W_P1_COUNT(R3),R2      ; address of start of mapping pointers
          54    16 A3      3C    247C            MOVZWL   WCB$W_NMAP(R3),R4          ; number of mapping pointer
                54 06      C4    2480            MULL2    #6,R4                      ; number of bytes of mapping pointers
                54 52      C0    2483            ADDL2    R2,R4                      ; address beyond end of mapping ptrs
   00000F7E'8F     54      D1    2486            CMPL     R4,#WCBEND                 ; is it beyond the buffer?
                   26      15    248D            BLEQ     30$                        ; if leq no -- its ok
    54    00000F7E'EF      DE    248F            MOVAL    WCBEND,R4                  ; set new end of buffer
                                 2496            SKIP     1
                                 249F            PRINT    0,<-
                                 249F            Pointer table doesn't fit SDA memory buffer>
                                 24AC            SKIP     1
                                 24B5
             54    52      D1    24B5  30$:      CMPL     R2,R4                      ; end of pointers yet?
                   26      18    24B8            BGEQ     90$                        ; if geq yes -- exit
             7E    82      3C    24BA            MOVZWL   (R2)+,-(SP)                ; get size
```

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

```
                                 24E8         .SBTTL  SHOW_FCB
                                 24E8
                                 24E8  ;+++
                                 24E8  ; SHOW_FCB -- Show information from File Control Block.
                                 24E8  ;
                                 24E8  ; Inputs:
                                 24E8  ;
                                 24E8  ;       4(AP) = Address of buffer containing FCB.
                                 24E8  ;
                                 24E8  ; Outputs:
                                 24E8  ;
                                 24E8  ;       None.
                                 24E8  ;---
                                 24E8
                                 24E8         .ENABL  LSB
                                 24E8  SHOW_FCB:
                           001C  24E8         .WORD   ^M<R2,R3,R4>
00000006'EF  00000200 8F    D3   24EA         BITL    #OPT$M_FCB,RMS_DIS_OPT1  ; display FCB's?
                     03     12   24F5         BNEQ    1$                       ; if neq yes
                   026C     31   24F7         BRW     90$                      ; if eql no
         53       04 AC     D0   24FA  1$:    MOVL    4(AP),R3                 ; address of FCB buffer
                                 24FE         ENSURE  19
                                 2516         SKIP    2
                  FC A3     DD   251F         PUSHL   -4(R3)                   ; address of FCB in dump
                                 2522         PRINT   1,<-
                                 2522  FCB Address: !XL>
                                 252F         PRINT   0,<-
                                 252F  ------------>
                                 253C         SKIP    1
         7E       08 A3     3C   2545         MOVZWL  FCB$W_SIZE(R3),-(SP)
                     63     DD   2549         PUSHL   FCB$L_FCBFL(R3)
                                 254B         PRINT   2,<-
                                 254B  FCBFL:         !XL                  -
                                 254B  SIZE:          !XW      !-!8UW.>
         7E       0A A3     9A   2558         MOVZBL  FCB$B_TYPE(R3),-(SP)
                  04 A3     DD   255C         PUSHL   FCB$L_FCBBL(R3)
                                 255F         PRINT   2,<-
                                 255F  FCBBL:         !XL                  -
                                 255F  TYPE:          !XB>
                  0C A3     DD   256C         PUSHL   FCB$L_EXFCB(R3)
         7E       1A A3     3C   256F         MOVZWL  FCB$W_ACNT(R3),-(SP)
                                 2573         PRINT   2,<-
                                 2573  ACNT:          !XW      !-!8UW. -
                                 2573  EXFCB:         !XL>
                  10 A3     DD   2580         PUSHL   FCB$L_WLFL(R3)
         7E       1E A3     3C   2583         MOVZWL  FCB$W_LCNT(R3),-(SP)
                                 2587         PRINT   2,<-
                                 2587  LCNT:          !XW      !-!8UW. -
                                 2587  WLFL:          !XL>
                  14 A3     DD   2594         PUSHL   FCB$L_WLBL(R3)
         7E       1C A3     3C   2597         MOVZWL  FCB$W_WCNT(R3),-(SP)
                                 259B         PRINT   2,<-
                                 259B  WCNT:          !XW      !-!8UW. -
                                 259B  WLBL:          !XL>
                                 25A8
                                 25A8  ;
                                 25A8  ; 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        MOVZWL  FCB$W_TCNT(R3),-(SP)
                        2685        PRINT   2,<-
                        2685  TCNT:         !XW      !-!8UW. -
                        2685  DIRSEQ:       !XW>
       1A A3   01   B1  2692        CMPW    #1,FCB$W_ACNT(R3)          ; is this file open by more than one?
               03   19  2696        BLSS    30$                        ; if lss yes
             00CB   31  2698        BRW     90$                        ; if geq no -- exit
                        269B
                        269B  ;
                        269B  ; Here we will attempt to print out the WCB addresses and associated
                        269B  ; PID's for all the accessors of this file.
                        269B  ;
                        269B
 54  00000F7E'EF    D0  269B  30$:  MOVL    FCB-4,R4                   ; get virtual address of FCB
            54 10   C0  26A2        ADDL2   #FCB$L_WLFL,R4             ; create virt addr of queue head
 52  00000D4E'EF    9E  26A5        MOVAB   WCB,R2
       62   10 A3   D0  26AC        MOVL    FCB$L_WLFL(R3),WCB$L_WLFL(R2) ; set up starting address 1st WCB
                        26B0        SKIP    1
                        26B9        PRINT   0,<-
                        26B9                Current Accessors>
                        26C6        PRINT   0,<-
                        26C6                ----------------->
                        26D3        SKIP    1
                        26DC        PRINT   0,<-
                        26DC      WCB Address               PID>
                        26E9        PRINT   0,<-
                        26E9      -----------               --->
                        26F6        SKIP    1
                        26FF
                        26FF  40$:  GET     WCB,@WCB$L_WLFL(R2),90$
       0B A2   08   93  272C        BITB    #WCB$M_SHRWCB,WCB$B_ACCESS(R2) ; is this shared WCB?
               19   13  2730        BEQL    50$                        ; if eql no
                        2732
                        2732  ;
                        2732  ; Shared WCB -- print out access count instead of PID
                        2732  ;
                        2732
       7E   0E A2   3C  2732        MOVZWL  WCB$W_REFCNT(R2),-(SP)
     00000D4A'EF    DD  2736        PUSHL   WCB-4
                        273C        PRINT   2,<-
                        273C        !XL     REFCNT: !XW !-!8UW.>
               16   11  2749        BRB     60$
                        274B
            0C A2   DD  274B  50$:  PUSHL   WCB$L_PID(R2)
     00000D4A'EF    DD  274E        PUSHL   WCB-4
                        2754        PRINT   2,<-
                        2754        !XL             !XL>
          62   54   D1  2761  60$:  CMPL    R4,WCB$L_WLFL(R2)
               99   12  2764        BNEQ    40$
                        2766
                        2766  90$:  STATUS  SUCCESS
                    04  276D        RET
                        276E        .DSABL  LSB
```

```
                          276E          .SBTTL  SHOW_IRAB
                          276E
                          276E  ;+++
                          276E  ; SHOW_IRAB -- Show information from IRAB.
                          276E  ;
                          276E  ; Inputs:
                          276E  ;
                          276E  ;       4(AP) = Address of IRAB in buffer in memory.
                          276E  ;
                          276E  ; Outputs:
                          276E  ;
                          276E  ;       None.
                          276E  ;---
                          276E
                          276E          .ENABL  LSB
                          276E  SHOW_IRAB:
                    001C  276E          .WORD   ^M<R2,R3,R4>
00000006'EF     04    D3  2770          BITL    #OPT$M_IRB,RMS_DIS_OPT1 ; display IRAB's?
                03    12  2777          BNEQ    1$                      ; if neq yes
              0538    31  2779          BRW     90$                     ; if no -- exit
     53     04 AC     D0  277C  1$:     MOVL    4(AP),R3                ; get address of buffer into R3
                          2780          SKIP    PAGE
     7E     52 A3     3C  2787          MOVZWL  IRB$W_OWN_ISI(R3),-(SP) ; push ISI value for this IRAB
            FC A3     DD  278B          PUSHL   -4(R3)                  ; real address of this IRAB
                          278E          PRINT   2,<-
                          278E  IRAB Address: !XL               ISI: !XW>
                          279B          PRINT   0,<-
                          279B  -------------               ---->
                          27A8          SKIP    1
                63    DD  27B1          PUSHL   IRB$L_IFAB_LNK(R3)      ; address of FAB to which this lnkd
                          27B3          PRINT   1,<-
                          27B3  IFAB_LNK:       !XL>
                          27C0          ALLOC   ALLOCSZ,R4              ; make a buffer to trans BKP bits
            04 A3     DD  27D2          PUSHL   IRB$L_BKPBITS(R3)       ; set of the bits for a call
          DB97 CF     9F  27D5          PUSHAB  IRBBKP_CHR              ; address of translation table
00000000'EF     02    FB  27D9          CALLS   #2,TRANSLATE_BITS
                54    DD  27E0          PUSHL   R4
            04 A3     DD  27E2          PUSHL   IRB$L_BKPBITS(R3)       ; the bit again, this time for display
                          27E5          PRINT   2,<-
                          27E5  BKPBITS:        !XL     !AS>
     7E     08 A3     9A  27F2          MOVZBL  IRB$B_BID(R3),-(SP)     ; block ID
     7E     09 A3     9A  27F6          MOVZBL  IRB$B_BLN(R3),-(SP)     ; block length
                          27FA          PRINT   2,<-
                          27FA  BLN:            !XB     !-!8UB. -
                          27FA  BID:            !XB     !-!8UB.>
     7E     0A A3     9A  2807          MOVZBL  IRB$B_MODE(R3),-(SP)    ; mode
     7E     0B A3     9A  280B          MOVZBL  IRB$B_EFN(R3),-(SP)     ; event flag number
                          280F          PRINT   2,<-
                          280F  EFN:            !XB                     -
                          280F  MODE:           !XB>
            14 A3     DD  281C          PUSHL   IRB$L_ASBADDR(R3)       ; ASB address
            0C A3     DD  281F          PUSHL   IRB$L_IOS(R3)           ; I/O status Block
                          2822          PRINT   2,<-
                          2822  IOS:            !XL             -
                          2822  ASBADDR:        !XL>
            18 A3     DD  282F          PUSHL   IRB$L_ARGLST(R3)        ; User argument list
            10 A3     DD  2832          PUSHL   IRB$L_IOS4(R3)          ; end of I/O status block
```

```
                              2835               PRINT   2,<-
                              2835      IOS4:            !XL                  -
                              2835      ARGLST:          !XL>
                   20 A3  DD  2842               PUSHL   IRB$L_CURBDB(R3)          ; current BDB pointer
                   1C A3  DD  2845               PUSHL   IRB$L_IRAB_LNK(R3)        ; link to IRAB list
                              2848               PRINT   2,<-
                              2848      IRAB_LNK:        !XL                  -
                              2848      CURBDB:          !XL>
                   3C A3  DD  2855               PUSHL   IRB$L_NXTBDB(R3)          ; next BDB
                   24 A3  DD  2858               PUSHL   IRB$L_LAST_RAB(R3)        ; most recently used RAB in users space
                              285B               PRINT   2,<-
                              285B      LAST_RAB:        !XL                  -
                              285B      NXTBDB:          !XL>
                   30 A3  DD  2868               PUSHL   IRB$L_JNLBDB(R3)
                   38 A3  DD  286B               PUSHL   IRB$L_RLB_LNK(R3)
                              286E               PRINT   2,<-
                              286E      RLB_LNK:         !XL                  -
                              286E      JNLBDB:          !XL>
                   34 A3  DD  287B               PUSHL   IRB$L_IDENT(R3)
             7E    28 A3  3C  287E               MOVZWL  IRB$W_ISI(R3),-(SP)
                              2882               PRINT   2,<-
                              2882      ISI:             !XW                            -
                              2882      IDENT:           !XL>
       6E   00000087 8F   D0  288F               MOVL    #ALLOCSZ,(SP)             ; reset size of allocated buffer
             7E    5D A3  9A  2896               MOVZBL  IRB$B_JNLFLG3(R3),-(SP) ; journal flag3 bits for translate
                 E336 CF  9F  289A               PUSHAB  JNLFLG3_CHR
       00000000'EF    02  FB  289E               CALLS   #2,TRANSLATE_BITS
                      54  DD  28A5               PUSHL   R4
             7E    5D A3  9A  28A7               MOVZBL  IRB$B_JNLFLG3(R3),-(SP) ; journal flag bits
                              28AB               PRINT   2,<-
                              28AB      JNLFLGS3:        !XB          !AS>
                   2C A3  DD  28B8               PUSHL   IRB$L_ATJNLBUF(R3)        ; pointer to AT MJB
                              28BB               PRINT   1,<-
                              28BB      ATJNL_PTR:       !XL>
                              28C8
                              28C8      ;
                              28C8      ; Case now for some organization dependant fields
                              28C8      ;
                              28C8
  02     00    00000145'EF  8F  28C8             CASEB   IFB$B_ORGCASE+IFAB,#0,#2
                         0009'  28D0    2$:      .WORD   10$-2$                    ; Sequential
                         0032'  28D2             .WORD   20$-2$                    ; Relative
                         0093'  28D4             .WORD   30$-2$                    ; Indexed
                     03E3  31   28D6             BRW     99$                       ; organization error
                                28D9
                                28D9    ;
                                28D9    ; Sequential
                                28D9    ;
                                28D9
                   48 A3  DD  28D9      10$:     PUSHL   IRB$L_RP_VBN(R3)          ; VBN from current record pointer
                   40 A3  DD  28DC               PUSHL   IRB$L_NRP_VBN(R3)         ; VBN from next record pointer
                              28DF               PRINT   2,<-
                              28DF      NRP_VBN:         !XL!-!8UL.           -
                              28DF      RP_VBN:          !XL!-!8UL.>
                   4C A3  DD  28EC               PUSHL   IRB$L_RP_OFF(R3)          ; current record pointer offset
                   44 A3  DD  28EF               PUSHL   IRB$L_NRP_OFF(R3)         ; next record pointer offset
                              28F2               PRINT   2,<-
```

```
                          28F2  NRP_OFF:         !XL!-!8UL.       -
                          28F2  RP_OFF:          !XL!-!8UL.>
        0130    31        28FF          BRW      40$
                          2902
                          2902  ;
                          2902  ; Relative
                          2902  ;
                          2902
       48 A3    DD        2902  20$:    PUSHL    IRB$L_RP_VBN(R3)            ; record pointer VBN
       40 A3    DD        2905          PUSHL    IRB$L_NRP_VBN(R3)           ; next record pointer VBN
                          2908          PRINT    2,<-
                          2908  NRP_VBN:         !XL!-!8UL.       -
                          2908  RP_VBN:          !XL!-!8UL.>
       48 A3    DD        2915          PUSHL    IRB$L_RP(R3)
       40 A3    DD        2918          PUSHL    IRB$L_NRP(R3)
                          291B          PRINT    2,<-
                          291B  RP:              !XL!-!8UL.       -
                          291B  NRP:             !XL!-!8UL.>
       4C A3    DD        2928          PUSHL    IRB$L_RP_OFF(R3)
       44 A3    DD        292B          PUSHL    IRB$L_NRP_OFF(R3)
                          292E          PRINT    2,<-
                          292E  NRP_OFF:         !XL!-!8UL.       -
                          292E  RP_OFF:          !XL!-!8UL.>
   7E  4C A3    3C        293B          MOVZWL   IRB$W_RP_OFF(R3),-(SP)
   7E  44 A3    3C        293F          MOVZWL   IRB$W_NRP_OFF(R3),-(SP)
                          2943          PRINT    2,<-
                          2943  NRP_OFF:         !XW      !-!8UW. -
                          2943  RP_OFF:          !XW      !-!8UW.>
       44 A3    DD        2950          PUSHL    IRB$L_CURVBN(R3)
                          2953          PRINT    1,<-
                          2953  CURVBN:          !XL!-!8UL.>
        00CF    31        2960          BRW      40$
                          2963
                          2963  ;
                          2963  ; Indexed
                          2963  ;
                          2963
6E   00000087 8F    D0    2963  30$:    MOVL     #ALLOCSZ,(SP)
     7E    40 A3    9A    296A          MOVZBL   IRB$B_CACHEFLGS(R3),-(SP)
         DDFE CF    9F    296E          PUSHAB   CSH_CHR
00000000'EF   02    FB    2972          CALLS    #2,TRANSLATE_BITS
              54    DD    2979          PUSHL    R4
     7E    40 A3    9A    297B          MOVZBL   IRB$B_CACHEFLGS(R3),-(SP)
                          297F          PRINT    2,<-
                          297F  CACHEFLGS:       !XB         !AS>
6E   00000087 8F    D0    298C          MOVL     #ALLOCSZ,(SP)               ; reset buffer size on stack
     7E    42 A3    3C    2993          MOVZWL   IRB$W_SRCHFLAGS(R3),-(SP)   ; push bits to translate
         DBB1 CF    9F    2997          PUSHAB   SCH_CHR                     ;  address of translate table
00000000'EF   02    FB    299B          CALLS    #2,TRANSLATE_BITS
              54    DD    29A2          PUSHL    R4                          ; address of text string
     7E    42 A3    3C    29A4          MOVZWL   IRB$W_SRCHFLAGS(R3),-(SP)   ; bits again
                          29A8          PRINT    2,<-
                          29A8  SRCHFLAGS:       !XW         !AS>
6E   00000087 8F    D0    29B5          MOVL     #ALLOCSZ,(SP)               ; reset buffer size on stack
     7E    44 A3    9A    29BC          MOVZBL   IRB$B_SPL_BITS(R3),-(SP)    ; bits for translation
         DB30 CF    9F    29C0          PUSHAB   SPL_CHR                     ; address of translation table
00000000'EF   02    FB    29C4          CALLS    #2,TRANSLATE_BITS
```

```
                   54    DD  29CB                   PUSHL   R4                          ; buffer full of bit names
           7E   44 A3    9A  29CD                   MOVZBL  IRB$B_SPL_BITS(R3),-(SP)    ; bit values again
                             29D1                   PRINT   2,<-
                             29D1       SPL_BITS:           !XB         !AS>
           7E   41 A3    9A  29DE                   MOVZBL  IRB$B_STOPLEVEL(R3),-(SP)
  7E    44 A3   02  01   EF  29E2                   EXTZV   #IRB$V_NEW_BKTS,#2,IRB$B_SPL_BITS(R3),-(SP)
                             29E8                   PRINT   2,<-
                             29E8       NEWBKTS:            !XB         !-!8UB. -
                             29E8       STOPLEVEL:          !XB         !-!8UB.>
           7E   4A A3    3C  29F5                   MOVZWL  IRB$W_SPLIT(R3),-(SP)
           7E   48 A3    3C  29F9                   MOVZWL  IRB$W_POS_INS(R3),-(SP)
                             29FD                   PRINT   2,<-
                             29FD       POS_INS:            !XW         !-!8UW. -
                             29FD       SPLIT:              !XW         !-!8UW.>
                4C A3    DD  2A0A                   PUSHL   IRB$L_PTR_VBN(R3)
                4C A3    DD  2A0D                   PUSHL   IRB$L_LST_REC(R3)
                             2A10                   PRINT   2,<-
                             2A10       LST_REC:            !XL                 -
                             2A10       PRT_VBN:            !XL!-!8UL.>
           7E   4E A3    3C  2A1D                   MOVZWL  IRB$W_SPLIT_2(R3),-(SP)
           7E   4C A3    3C  2A21                   MOVZWL  IRB$W_SPLIT_1(R3),-(SP)
                             2A25                   PRINT   2,<-
                             2A25       SPLIT_1:            !XL!-!8UL.          -
                             2A25       SPLIT_2:            !XL!-!8UL.>
                             2A32
                             2A32       ;
                             2A32       ; Common path again
                             2A32       ;
                             2A32
           7E   54 A3    9A  2A32       40$:        MOVZBL  IRB$B_BCNT(R3),-(SP)        ; buffer count
                50 A3    DD  2A36                   PUSHL   IRB$L_OWNER_ID(R3)          ; full owner ID longword
                             2A39                   PRINT   2,<-
                             2A39       OWNER_ID:           !XL                 -
                             2A39       BCNT:               !XB         !-!8UB.>
           7E   55 A3    9A  2A46                   MOVZBL  IRB$B_MBC(R3),-(SP)         ; multibuffer count
           7E   52 A3    3C  2A4A                   MOVZWL  IRB$W_OWN_ISI(R3),-(SP)     ; ISI of this IRAB
                             2A4E                   PRINT   2,<-
                             2A4E       OWN_ISI:            !XW         !-!8UW. -
                             2A4E       MBC:                !XB         !-!8UB.>
           7E   50 A3    3C  2A5B                   MOVZWL  IRB$W_OWN_ID(R3),-(SP)      ; process ID (low word)
           7E   52 A3    9A  2A5F                   MOVZBL  IRB$B_PPF_ISI(R3),-(SP)
                             2A63                   PRINT   2,<-
                             2A63       PPF_ISI:            !XB         !-!8UB. -
                             2A63       OWN_ID:             !XW>
                             2A70
                             2A70       ;
                             2A70       ; Case for organization dependant fields
                             2A70       ;
                             2A70
  02    00   00000145'EF  8F 2A70                   CASEB   IFB$B_ORGCASE+IFAB,#0,#2
                     0009'   2A78       41$:        .WORD   50$-41$                     ; sequential
                     005E'   2A7A                   .WORD   60$-41$                     ; relative
                     0072'   2A7C                   .WORD   70$-41$                     ; indexed
                  023B   31  2A7E                   BRW     99$                         ; organization error
                             2A81
                             2A81       ;
                             2A81       ; Sequential
```

```
                         2A81  ;
                         2A81
        7E  62 A3   3C   2A81  50$:     MOVZWL  IRB$W_CSIZ(R3),-(SP)
            64 A3   DD   2A85           PUSHL   IRB$L_TEMPO(R3)
                         2A88           PRINT   2,<-
                         2A88  TEMPO:           !XL                 -
                         2A88  CSIZ:            !XW       !-!8UW.>
        7E  66 A3   3C   2A95           MOVZWL  IRB$W_RTOTLSZ(R3),-(SP)
        7E  64 A3   3C   2A99           MOVZWL  IRB$W_ROVHDSZ(R3),-(SP)
                         2A9D           PRINT   2,<-
                         2A9D  ROVHDSZ:         !XW       !-!8UW. -
                         2A9D  RTOTLSZ:         !XW       !-!8UW.>
        7E  65 A3   9A   2AAA           MOVZBL  IRB$B_POST_CCTL(R3),-(SP)
        7E  64 A3   9A   2AAE           MOVZBL  IRB$B_PRE_CCTL(R3),-(SP)
                         2AB2           PRINT   2,<-
                         2AB2  PRE_CCTL:        !XB       !-!8UB. -
                         2AB2  POST_CCTL:       !XB       !-!8UB.>
        7E  68 A3   9A   2ABF           MOVZBL  IRB$B_NVBNS(R3),-(SP)
            68 A3   DD   2AC3           PUSHL   IRB$L_TEMP1(R3)
                         2AC6           PRINT   2,<-
                         2AC6  TEMP1:           !XL                 -
                         2AC6  NVBNS:           !XB       !-!8UB.>
             01DE   31   2AD3           BRW     90$
                         2AD6
                         2AD6  ;
                         2AD6  ; Relative
                         2AD6  ;
                         2AD6
        7E  62 A3   9A   2AD6  60$:     MOVZBL  IRB$W_CSIZ(R3),-(SP)
                         2ADA           PRINT   1,<-
                         2ADA  CSIZ:            !XB       !-!8UB.>
             01CA   31   2AE7           BRW     90$
                         2AEA
                         2AEA  ;
                         2AEA  ; Indexed
                         2AEA  ;
                         2AEA
            64 A3   DD   2AEA  70$:     PUSHL   IRB$L_UPDBUF(R3)
            60 A3   DD   2AED           PUSHL   IRB$L_KEYBUF(R3)
                         2AF0           PRINT   2,<-
                         2AF0  KEYBUF:          !XL                 -
                         2AF0  UPDBUF:          !XL>
            68 A3   DD   2AFD           PUSHL   IRB$L_RECBUF(R3)
            6C A3   DD   2B00           PUSHL   IRB$L_OLDBUF(R3)
                         2B03           PRINT   2,<-
                         2B03  RECBUF:          !XL                 -
                         2B03  OLDBUF:          !XL>
            70 A3   DD   2B10           PUSHL   IRB$L_UPD_BDB(R3)
            70 A3   DD   2B13           PUSHL   IRB$L_RFA_VBN(R3)
                         2B16           PRINT   2,<-
                         2B16  RFA_VBN:         !XL!-!8UL.          -
                         2B16  UPD_BDB:         !XL>
            70 A3   DD   2B23           PUSHL   IRB$L_LAST_VBN(R3)
        7E  74 A3   3C   2B26           MOVZWL  IRB$W_RFA_ID(R3),-(SP)
                         2B2A           PRINT   2,<-
                         2B2A  RFA_ID:          !XW       !-!8UW. -
                         2B2A  LAST_VBN:        !XL!-!8UL.>
```

```
   7E    74 A3     3C  2B37              MOVZWL  IRB$W_LAST_ID(R3),-(SP)
   7E    76 A3     3C  2B3B              MOVZWL  IRB$W_SAVE_POS(R3),-(SP)
                       2B3F              PRINT   2,<-
                       2B3F  SAVE_POS:           !XW       !-!8UW. -
                       2B3F  LAST_ID:            !XW       !-!8UW.>
         78 A3     DD  2B4C              PUSHL   IRB$L_PUTUP_VBN(R3)
         78 A3     DD  2B4F              PUSHL   IRB$L_NEXT_VBN(R3)
                       2B52              PRINT   2,<-
                       2B52  NEXT_VBN:           !XL!-!8UL.      -
                       2B52  PUTUP_VBN:          !XL!-!8UL.>
   7E  0080 C3     3C  2B5F              MOVZWL  IRB$W_PUTUP_ID(R3),-(SP)
   7E  0080 C3     3C  2B64              MOVZWL  IRB$W_NEXT_ID(R3),-(SP)
                       2B69              PRINT   2,<-
                       2B69  NEXT_ID:            !XW       !-!8UW. -
                       2B69  PUTUP_ID:           !XW       !-!8UW.>
   7E  0082 C3     3C  2B76              MOVZWL  IRB$W_FIRST_ID(R3),-(SP)
         7C A3     DD  2B7B              PUSHL   IRB$L_FIRST_VBN(R3)
                       2B7E              PRINT   2,<-
                       2B7E  FIRST_VBN:          !XL!-!8UL.      -
                       2B7E  FIRST_ID:           !XW       !-!8UW.>
       0090 C3     DD  2B8B              PUSHL   IRB$L_NEXT_DOWN(R3)
       0084 C3     DD  2B8F              PUSHL   IRB$L_LOCK_BDB(R3)
                       2B93              PRINT   2,<-
                       2B93  LOCK_BDB:           !XL                     -
                       2B93  NEXT_DOWN:          !XL>
       0088 C3     DD  2BA0              PUSHL   IRB$L_MIDX_TMP1(R3)
       0088 C3     DD  2BA4              PUSHL   IRB$L_VBN_LEFT(R3)
                       2BA8              PRINT   2,<-
                       2BA8  VBN_LEFT:           !XL!-!8UL.      -
                       2BA8  MIDX_TMP1:          !XL>
       008C C3     DD  2BB5              PUSHL   IRB$L_MIDX_TMP2(R3)
       008C C3     DD  2BB9              PUSHL   IRB$L_VBN_RIGHT(R3)
                       2BBD              PRINT   2,<-
                       2BBD  VBN_RIGHT:          !XL!-!8UL.      -
                       2BBD  MIDX_TMP2:          !XL>
       0090 C3     DD  2BCA              PUSHL   IRB$L_MIDX_TMP3(R3)
       0090 C3     DD  2BCE              PUSHL   IRB$L_VBN_MID(R3)
                       2BD2              PRINT   2,<-
                       2BD2  VBN_MID:            !XL!-!8UL.      -
                       2BD2  MIDX_TMP3:          !XL>
       0098 C3     DD  2BDF              PUSHL   IRB$L_LST_NCMP(R3)
       0094 C3     DD  2BE3              PUSHL   IRB$L_REC_COUNT(R3)
                       2BE7              PRINT   2,<-
                       2BE7  REC_COUNT:          !XL!-!8UL.      -
                       2BE7  LST_NCMP:           !XL>
   7E  00A0 C3     3C  2BF4              MOVZWL  IRB$W_NID_RIGHT(R3),-(SP)
       009C C3     DD  2BF9              PUSHL   IRB$L_SPL_COUNT(R3)
                       2BFD              PRINT   2,<-
                       2BFD  SPL_COUNT:          !XL!-!8UL.      -
                       2BFD  NID_RIGHT:          !XW       !-!8UW.>
   7E  00A2 C3     3C  2C0A              MOVZWL  IRB$W_NID_MID(R3),-(SP)
   7E  00A4 C3     3C  2C0F              MOVZWL  IRB$W_RFA_NID(R3),-(SP)
                       2C14              PRINT   2,<-
                       2C14  RFA_NID:            !XW       !-!8UW. -
                       2C14  NID_MID:            !XW       !-!8UW.>
   7E  00A6 C3     9A  2C21              MOVZBL  IRB$B_KEYSZ(R3),-(SP)
                       2C26              PRINT   1,<-
```

```
                         2C26  KEYSZ:                !XB       !-!8UB.>
         00AC C3    DD   2C33               PUSHL    IRB$L_POS_VBN(R3)
         00A8 C3    DD   2C37               PUSHL    IRB$L_CUR_VBN(R3)
                         2C3B               PRINT    2,<-
                         2C3B  CUR_VBN:              !XL!-!8UL.           -
                         2C3B  POS_VBN:              !XL!-!8UL.>
    7E   00BA C3    3C   2C48               MOVZWL   IRB$W_POS_ID(R3),-(SP)
    7E   00B8 C3    3C   2C4D               MOVZWL   IRB$W_CUR_ID(R3),-(SP)
                         2C52               PRINT    2,<-
                         2C52  CUR_ID:               !XW       !-!8UW. -
                         2C52  POS_ID:               !XW       !-!8UW.>
    7E   00C0 C3    3C   2C5F               MOVZWL   IRB$W_CUR_COUNT(R3),-(SP)
                         2C64               PRINT    1,<-
                         2C64  CUR_COUNT:            !XW       !-!8UW.>
         00B4 C3    DD   2C71               PUSHL    IRB$L_SIDR_VBN(R3)
         00B0 C3    DD   2C75               PUSHL    IRB$L_UDR_VBN(R3)
                         2C79               PRINT    2,<-
                         2C79  UDR_VBN:              !XL!-!8UL.           -
                         2C79  SIDR_VBN:             !XL!-!8UL.>
    7E   00BE C3    3C   2C86               MOVZWL   IRB$W_SIDR_ID(R3),-(SP)
    7E   00BC C3    3C   2C8B               MOVZWL   IRB$W_UDR_ID(R3),-(SP)
                         2C90               PRINT    2,<-
                         2C90  UDR_ID:               !XW       !-!8UW. -
                         2C90  SIDR_ID:              !XW       !-!8UW.>
    7E   00C2 C3    9A   2C9D               MOVZBL   IRB$B_RP_KREF(R3),-(SP)
    7E   00C3 C3    9A   2CA2               MOVZBL   IRB$B_CUR_KREF(R3),-(SP)
                         2CA7               PRINT    2,<-
                         2CA7  CUR_KREF:             !XB       !-!8UB. -
                         2CA7  RP_KREF:              !XB       !-!8UB.>
                         2CB4  90$:         STATUS   SUCCESS
                    04   2CBB               RET
                         2CBC  99$:         STATUS   NOTVALID
                    04   2CC3               RET
                         2CC4               .DSABL   LSB
```

```
                                    2CC4          .SBTTL  SHOW_BDB_SUM
                                    2CC4
                                    2CC4  ;+++
                                    2CC4  ; SHOW_BDB_SUM -- Show one line summaries of important BDB information.
                                    2CC4  ;
                                    2CC4  ; Inputs:
                                    2CC4  ;
                                    2CC4  ;       None.
                                    2CC4  ;
                                    2CC4  ; Implicit Inputs:
                                    2CC4  ;
                                    2CC4  ;       RMS_DIS_OPT1 = Display options.
                                    2CC4  ;       IFAB         = Block buffer with valid (current) IFAB.
                                    2CC4  ;
                                    2CC4  ; Outputs:
                                    2CC4  ;
                                    2CC4  ;       None.
                                    2CC4  ;
                                    2CC4  ; Implicit Outputs:
                                    2CC4  ;
                                    2CC4  ;       Current BDB buffer destroyed.
                                    2CC4  ;---
                                    2CC4
                                    2CC4          .ENABLE LSB
                                    2CC4  SHOW_BDB_SUM:
                              001C  2CC4          .WORD   ^M<R2,R3,R4>
      03 00000006'EF    05      E0  2CC6          BBS     #OPT$V_BDBSUM,RMS_DIS_OPT1,10$ ; branch if summary enabled
                      011B      31  2CCE  5$:     BRW     90$                         ; optimize branches
      52   0000011E'EF          D0  2CD1  10$:    MOVL    IFAB-4,R2                   ; virtual address of current IFAB
      52   00000040 8F          C0  2CD8          ADDL2   #IFB$L_BDB_FLNK,R2          ; virtual address of BDB queue header
      00000162'EF       52      D1  2CDF          CMPL    R2,IFB$L_BDB_FLNK+IFAB      ; does queue header point to itself?
                        E6      13  2CE6          BEQL    5$                          ; if eql yes -- empty queue, exit
  000002A6'EF    00000162'EF    D0  2CE8          MOVL    IFB$L_BDB_FLNK+IFAB,BDB     ; initialize BDB buffer to point to 1st
                                    2CF3          SKIP    PAGE                        ; we're really going to print something
                                    2CFA          PRINT   0,<-
                                    2CFA                                BDB Summary>
                                    2D07          PRINT   0,<-
                                    2D07                                ----------->
                                    2D14          SKIP    1
                                    2D1D          PRINT   0,<-
                                    2D1D   BDB                                             CACHE_>
                                    2D2A          PRINT   0,<-
                                    2D2A  Address USERS SIZE  NUMB    VBN   BLB_PTR   ADDR   VAL    FLGS>
                                    2D37          PRINT   0,<-
                                    2D37  ------- ----- ----  ----    ---   -------   ----   ---    ---->
                                    2D44          SKIP    1
                                    2D4D
                                    2D4D  ;
                                    2D4D  ; Allocate memory for translate bits output.
                                    2D4D  ;
                                    2D4D
                                    2D4D          ALLOC   ALLOCSZ,R4
                                    2D5F
                                    2D5F  ;
                                    2D5F  ; Main loop -- get BDB's from list and display important info, one line
                                    2D5F  ; per BDB.
                                    2D5F  ;
```

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

```
                                      2DF4          .SBTTL  SHOW_BLB_SUM
                                      2DF4
                                      2DF4  ;+++
                                      2DF4  ; SHOW_BLB_SUM -- Show one line summaries of important BLB information.
                                      2DF4  ;
                                      2DF4  ; Inputs:
                                      2DF4  ;
                                      2DF4  ;       None.
                                      2DF4  ;
                                      2DF4  ; Implicit Inputs:
                                      2DF4  ;
                                      2DF4  ;       RMS_DIS_OPT1 = Display options.
                                      2DF4  ;       IFAB         = Block buffer with valid (current) IFAB.
                                      2DF4  ;
                                      2DF4  ; Outputs:
                                      2DF4  ;
                                      2DF4  ;       None.
                                      2DF4  ;
                                      2DF4  ; Implicit Outputs:
                                      2DF4  ;
                                      2DF4  ;       Current BLB buffer destroyed.
                                      2DF4  ;---
                                      2DF4
                                      2DF4          .ENABLE LSB
                                      2DF4  SHOW_BLB_SUM:
                                001C  2DF4          .WORD   ^M<R2,R3,R4>
        11 00000122'EF    33    E0    2DF6          BBS     #IFB$V_NORECLK,IFAB,5$  ; if no blb's on this IFAB
        00    00000145'EF    91       2DFE          CMPB    IFB$B_ORGCASE+IFAB,#IFB$C_SEQ ; is file sequential?
                          08    13    2E05          BEQL    5$                      ; if eql yes, exit
        03 00000006'EF    10    E0    2E07          BBS     #OPT$V_BLBSUM,RMS_DIS_OPT1,10$ ; branch if summary enabled
                        0113    31    2E0F  5$:     BRW     90$                     ; optimize branches
        52    0000011E'EF    D0       2E12  10$:    MOVL    IFAB-4,R2               ; virtual address of current IFAB
        52    00000098 8F    C0       2E19          ADDL2   #IFB$L_BLBFLNK,R2       ; virtual address of BLB queue header
           000001BA'EF    52    D1    2E20          CMPL    R2,IFB$L_BLBFLNK+IFAB   ; does queue header point to itself?
                          E6    13    2E27          BEQL    5$                      ; if eql yes -- empty queue, exit
  0000138A'EF    000001BA'EF    D0    2E29          MOVL    IFB$L_BLBFLNK+IFAB,BLB ; initialize BLB buffer to point to 1st
                                      2E34          SKIP    PAGE                    ; we're really going to print something
                                      2E3B          PRINT   0,<-
                                      2E3B                                  BLB Summary>
                                      2E48          PRINT   0,<-
                                      2E48                                  ----------->
                                      2E55          SKIP    1
                                      2E5E          PRINT   0,<-
                                      2E5E   BLB      BDB              MODE>
                                      2E6B          PRINT   0,<-
                                      2E6B  Address  Address  OWNER  HELD   VBN   LOCK_ID  VALSEQNO    BLBFLGS>
                                      2E78          PRINT   0,<-
                                      2E78  -------  -------  -----  ----   ---   -------  --------    ------->
                                      2E85          SKIP    1
                                      2E8E
                                      2E8E  ;
                                      2E8E  ; Allocate memory for translate bits output.
                                      2E8E  ;
                                      2E8E
                                      2E8E          ALLOC   ALLOCSZ,R4
                                      2EA0
                                      2EA0  ;
```

```
                            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          .SBTTL  SHOW_BDB
                          2F2D
                          2F2D  ;+++
                          2F2D  ; SHOW_BDB -- Show information from BDB.
                          2F2D  ;
                          2F2D  ; Inputs:
                          2F2D  ;
                          2F2D  ;       4(AP) = Address of BDB in local memory.
                          2F2D  ;
                          2F2D  ; Outputs:
                          2F2D  ;
                          2F2D  ;       None.
                          2F2D  ;---
                          2F2D
                          2F2D          .ENABL  LSB
                          2F2D  SHOW_BDB:
                    001C  2F2D          .WORD   ^M<R2,R3,R4>
        53    04 AC   D0  2F2F          MOVL    4(AP),R3                  ; get address of BDB into R3
                          2F33          ENSURE  17
                          2F4B          SKIP    1
              FC A3   DD  2F54          PUSHL   -4(R3)                    ; real address of BDB
        08 A3    15   91  2F57          CMPB    #GBPB$C_BID,BDB$B_BID(R3) ; is it GBPB
                 1C   12  2F5B          BNEQ    10$                       ;if neq no
                          2F5D          PRINT   1,<-
                          2F5D  GBPB Address: !XL>
                          2F6A          PRINT   0,<-
                          2F6A  ------------->
                 1A   11  2F77          BRB     20$
                          2F79  10$:    PRINT   1,<-
                          2F79  BDB Address: !XL>
                          2F86          PRINT   0,<-
                          2F86  ------------->
                          2F93  20$:    SKIP    1
        7E    08 A3   9A  2F9C          MOVZBL  BDB$B_BID(R3),-(SP)       ; block ID
                 63   DD  2FA0          PUSHL   BDB$L_FLINK(R3)           ; forward BDB link
                          2FA2          PRINT   2,<-
                          2FA2  FLINK:          !XL                 -
                          2FA2  BID:            !XB      !-!8UB.>
        7E    09 A3   9A  2FAF          MOVZBL  BDB$B_BLN(R3),-(SP)       ; block length
              04 A3   DD  2FB3          PUSHL   BDB$L_BLINK(R3)           ; back link for BDB chain
                          2FB6          PRINT   2,<-
                          2FB6  BLINK:          !XL                 -
                          2FB6  BLN:            !XB      !-!8UB.>
                          2FC3          ALLOC   ALLOCSZ,R4                ; make some space for translate bits
        7E    0A A3   9A  2FD5          MOVZBL  BDB$B_FLGS(R3),-(SP)      ; push flags byte for translate
            D4A3 CF   9F  2FD9          PUSHAB  BDB_CHR                   ; address of translate table
   00000000'EF   02   FB  2FDD          CALLS   #2,TRANSLATE_BITS
                 54   DD  2FE4          PUSHL   R4
        7E    0A A3   9A  2FE6          MOVZBL  BDB$B_FLGS(R3),-(SP)      ; flags again for print this time
                          2FEA          PRINT   2,<-
                          2FEA  FLGS:           !XB      !AS>
              10 A3   DD  2FF7          PUSHL   BDB$L_BLB_PTR(R3)
        7E    0C A3   3C  2FFA          MOVZWL  BDB$W_USERS(R3),-(SP)     ; use count
                          2FFE          PRINT   2,<-
                          2FFE  USERS:          !XW      !-!8UW. -
                          2FFE  BLB_PTR:        !XL>
        7E    0E A3   3C  300B          MOVZWL  BDB$W_BUFF_ID(R3),-(SP)
```

```
7E   0B A3   9A  300F              MOVZBL  BDB$B_CACHE_VAL(R3),-(SP)
                 3013              PRINT   2,<-
                 3013  CACHE_VAL:          !XB      !-!8UB. -
                 3013  BUFF_ID:            !XW      !-!8UW.>
7E   14 A3   3C  3020              MOVZWL  BDB$W_NUMB(R3),-(SP)    ; number of bytes in use
7E   16 A3   3C  3024              MOVZWL  BDB$W_SIZE(R3),-(SP)    ; buffer size
                 3028              PRINT   2,<-
                 3028  SIZE:               !XW      !-!8UW. -
                 3028  NUMB:               !XW      !-!8UW.>
     1C A3   DD  3035              PUSHL   BDB$L_VBN(R3)           ; VBN for this buffer
     18 A3   DD  3038              PUSHL   BDB$L_ADDR(R3)          ; address of buffer
                 303B              PRINT   2,<-
                 303B  ADDR:               !XL              -
                 303B  VBN:                !XL!-!8UL.>
08 A3    15  91  3048              CMPB    #GBPB$C_BID,BDB$B_BID(R3) ; is it GBPB
         16  12  304C              BNEQ    30$                     ; if neq no
     24 A3   DD  304E              PUSHL   GBPB$L_GBD_PTR(R3)
     20 A3   DD  3051              PUSHL   GBPB$L_VBNSEQNO(R3)
                 3054              PRINT   2,<-
                 3054  VBNSEQNO:           !XL              -
                 3054  GBD_PTR:            !XL>
       0089  31  3061              BRW     90$
                 3064
     24 A3   DD  3064  30$:        PUSHL   BDB$L_WAIT(R3)
     20 A3   DD  3067              PUSHL   BDB$L_VBNSEQNO(R3)
                 306A              PRINT   2,<-
                 306A  VBNSEQNO:           !XL              -
                 306A  WAIT:               !XL>
     4C A3   DD  3077              PUSHL   BDB$L_CURBUFADR(R3)
     48 A3   DD  307A              PUSHL   BDB$L_WK1(R3)
                 307D              PRINT   2,<-
                 307D  WK1:                !XL              -
                 307D  CURBUFADR:          !XL>
7E   4A A3   9A  308A              MOVZBL  BDB$B_PRE_CCTL(R3),-(SP)
7E   48 A3   9A  308E              MOVZBL  BDB$B_REL_VBN(R3),-(SP)
                 3092              PRINT   2,<-
                 3092  REL_VBN:            !XB      !-!8UB. -
                 3092  PRE_CCTL:           !XB>
7E   4B A3   9A  309F              MOVZBL  BDB$B_POST_CCTL(R3),-(SP)
7E   49 A3   9A  30A3              MOVZBL  BDB$B_VAL_VBNS(R3),-(SP)
                 30A7              PRINT   2,<-
                 30A7  VAL_VBNS:           !XB      !-!8UB. -
                 30A7  POST_CCTL:          !XB>
     44 A3   DD  30B4              PUSHL   BDB$T_JNLSEQ+12(R3)     ; journal sequence info
     40 A3   DD  30B7              PUSHL   BDB$T_JNLSEQ+8(R3)      ; journal sequence info
     3C A3   DD  30BA              PUSHL   BDB$T_JNLSEQ+4(R3)      ; journal sequence info
     38 A3   DD  30BD              PUSHL   BDB$T_JNLSEQ+0(R3)      ; journal sequence info
                 30C0              PRINT   4,<-
                 30C0  JNLSEQ:             !XL  !XL  !XL  !XL>
     48 A3   DD  30CD              PUSHL   BDB$L_IOSB(R3)
                 30D0              PRINT   1,<-
                 30D0  IOSB:               !XL>
     4C A3   DD  30DD              PUSHL   BDB$L_IOSB+4(R3)
                 30E0              PRINT   1,<-
                 30E0                      !XL>
                 30ED  90$:        STATUS  SUCCESS
             04  30F4              RET
```

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

```
                              30F5           .SBTTL  SHOW_BLB
                              30F5
                              30F5   ;+++
                              30F5   ; SHOW_BLB -- Show information from BLB.
                              30F5   ;
                              30F5   ; Inputs:
                              30F5   ;
                              30F5   ;       4(AP) = Address of BLB in local memory.
                              30F5   ;
                              30F5   ; Outputs:
                              30F5   ;
                              30F5   ;       None.
                              30F5   ;---
                              30F5
                              30F5           .ENABL  LSB
                              30F5   SHOW_BLB:
                      001C    30F5           .WORD   ^M<R2,R3,R4>
          53    04 AC   D0    30F7           MOVL    4(AP),R3                    ; get address of BLB into R3
                              30FB           ENSURE  16
                              3113           SKIP    1
                FC A3   DD    311C           PUSHL   -4(R3)                      ; real address of BLB
                              311F           PRINT   1,<-
                              311F   BLB Address: !XL>
                              312C           PRINT   0,<-
                              312C   ------------->
                              3139           SKIP    1
          7E    08 A3   9A    3142           MOVZBL  BLB$B_BID(R3),-(SP)        ; block ID
                   63   DD    3146           PUSHL   BLB$L_FLNK(R3)             ; forward BLB link
                              3148           PRINT   2,<-
                              3148   FLNK:           !XL                 -
                              3148   BID:            !XB       !-!8UB.>
          7E    09 A3   9A    3155           MOVZBL  BLB$B_BLN(R3),-(SP)        ; block length
                04 A3   DD    3159           PUSHL   BLB$L_BLNK(R3)             ; back link for BLB chain
                              315C           PRINT   2,<-
                              315C   BLNK:           !XL                 -
                              315C   BLN:            !XB       !-!8UB.>
                              3169           ALLOC   ALLOCSZ,R4                 ; make some space for translate bits
          7E    0A A3   9A    317B           MOVZBL  BLB$B_BLBFLGS(R3),-(SP)    ; push flags byte for translate
             D74D CF    9F    317F           PUSHAB  BLB_CHR                    ; address of translate table
     00000000'EF   02   FB    3183           CALLS   #2,TRANSLATE_BITS
                   54   DD    318A           PUSHL   R4
          7E    0A A3   9A    318C           MOVZBL  BLB$B_BLBFLGS(R3),-(SP)    ; flags again for print this time
                              3190           PRINT   2,<-
                              3190   BLBFLGS:        !XB            !AS>
          52    0B A3   9A    319D           MOVZBL  BLB$B_MODEHELD(R3),R2
             52    06   91    31A1           CMPB    #6,R2
                   03   1A    31A4           BGTRU   10$
             52    06   9A    31A6           MOVZBL  #6,R2
          D762 CF42     DD    31A9   10$:    PUSHL   BLB_TBL[R2]
          7E    0B A3   9A    31AE           MOVZBL  BLB$B_MODEHELD(R3),-(SP)
                0C A3   DD    31B2           PUSHL   BLB$L_BDB_ADDR(R3)
                              31B5           PRINT   3,<-
                              31B5   BDB_ADDR:       !XL                 -
                              31B5   MODEHELD:       !XB            !AC>
                10 A3   DD    31C2           PUSHL   BLB$L_OWNER(R3)
                14 A3   DD    31C5           PUSHL   BLB$L_VBN(R3)
                              31C8           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         .SBTTL  SHOW_RLB
                          3254
                          3254 ;+++
                          3254 ; SHOW_RLB -- Show information from the RLB.
                          3254 ;
                          3254 ; Inputs:
                          3254 ;
                          3254 ;         4(AP) = Address of buffer containing RLB.
                          3254 ;
                          3254 ; Outputs:
                          3254 ;
                          3254 ;         RLB displayed.
                          3254 ;---
                          3254
                          3254 SHOW_RLB:
                    001C  3254         .WORD   ^M<R2,R3,R4>
        53   04 AC    D0  3256         MOVL    4(AP),R3                ; get address of block buffer
                          325A         ENSURE  10
                          3272         SKIP    1
             FC A3    DD  327B         PUSHL   -4(R3)                  ; push virtual addresss of RLB
                          327E         PRINT   1,<-
                          327E RLB Address: !XL>
                          328B         PRINT   0,<-
                          328B ------------>
                          3298         SKIP    1
        7E   08 A3    9A  32A1         MOVZBL  RLB$B_BID(R3),-(SP)
                63    DD  32A5         PUSHL   RLB$L_LNK(R3)
                          32A7         PRINT   2,<-
                          32A7 LNK:            !XL             -
                          32A7 BID:            !XB     !-!8UB.>
        7E   09 A3    9A  32B4         MOVZBL  RLB$B_BLN(R3),-(SP)
             10 A3    DD  32B8         PUSHL   RLB$L_OWNER(R3)
                          32BB         PRINT   2,<-
                          32BB OWNER:          !XL     !-!8UL.-
                          32BB BLN:            !XB     !-!8UB.>
        7E   06 A3    3C  32C8         MOVZWL  RLB$W_RFA4(R3),-(SP)
             0C A3    DD  32CC         PUSHL   RLB$L_RFA0(R3)
                          32CF         PRINT   2,<-
                          32CF RFA0:           !XL!-!8UL.      -
                          32CF RFA4:           !XW     !-!8UW.>
        7E   0A A3    9A  32DC         MOVZBL  RLB$B_TMO(R3),-(SP)
                          32E0         PRINT   1,<-
                          32E0 TMO:            !XB     !-!8UB.>
                          32ED         ALLOC   ALLOCSZ,R4
        7E   0B A3    9A  32FF         MOVZBL  RLB$B_FLAGS(R3),-(SP)
           D4B9 CF    9F  3303         PUSHAB  RLB_CHR
 00000000'EF    02    FB  3307         CALLS   #2,TRANSLATE_BITS
                54    DD  330E         PUSHL   R4
        7E   0B A3    9A  3310         MOVZBL  RLB$B_FLAGS(R3),-(SP)
                          3314         PRINT   2,<-
                          3314 FLAGS:          !XB     !AS>
                          3321         ALLOC   ALLOCSZ,R4
        7E   04 A3    3C  3333         MOVZWL  RLB$W_FLAGS2(R3),-(SP)
           D4CD CF    9F  3337         PUSHAB  RLB2_CHR
 00000000'EF    02    FB  333B         CALLS   #2,TRANSLATE_BITS
                54    DD  3342         PUSHL   R4
        7E   04 A3    3C  3344         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          .SBTTL  SHOW_ASB
                             3371
                             3371  ;+++
                             3371  ; SHOW_ASB -- Show useful information from ASB.
                             3371  ;
                             3371  ; Inputs:
                             3371  ;
                             3371  ;       4(AP) = Address of block buffer.
                             3371  ;
                             3371  ; Outputs:
                             3371  ;
                             3371  ;       ASB Displayed.
                             3371  ;---
                             3371
                             3371          .ENABLE LSB
                       001C  3371  SHOW_ASB:.WORD  ^M<R2,R3,R4>
 53    04 AC     D0          3373          MOVL    4(AP),R3                        ; get address of ASB
                             3377          SKIP    PAGE
       FC A3     DD          337E          PUSHL   -4(R3)
                             3381          PRINT   1,<-
                             3381  ASB Address: !XL>
                             338E          PRINT   0,<-
                             338E  ------------>
                             339B          SKIP    1
 7E    08 A3     9A          33A4          MOVZBL  ASB$B_BID(R3),-(SP)
 7E    0C A3     9A          33A8          MOVZBL  ASB$B_ARGCNT(R3),-(SP)
                             33AC          PRINT   2,<-
                             33AC  ARGCNT:         !XB     !-!8UB. -
                             33AC  BID:            !XB     !-!8UB.>
 7E    09 A3     9A          33B9          MOVZBL  ASB$B_BLN(R3),-(SP)
       10 A3     DD          33BD          PUSHL   ASB$L_FABRAB(R3)
                             33C0          PRINT   2,<-
                             33C0  FABRAB:         !XL             -
                             33C0  BLN:            !XB     !-!8UB.>
    7E    63     3C          33CD          MOVZWL  ASB$W_STKLEN(R3),-(SP)
       14 A3     DD          33D0          PUSHL   ASB$L_ERR(R3)
                             33D3          PRINT   1,<-
                             33D3  ERR:            !XL             -
                             33D3  STKLEN:         !XW     !-!8UW.>
 7E    02 A3     3C          33E0          MOVZWL  ASB$W_STKSIZ(R3),-(SP)
       18 A3     DD          33E4          PUSHL   ASB$L_SUC(R3)
                             33E7          PRINT   1,<-
                             33E7  SUC:            !XL             -
                             33E7  STKSIZ:         !XW     !-!8UW.>
                             33F4          SKIP    1
 52    1C A3     DE          33FD          MOVAL   ASB$L_REGS(R3),R2
    7E    82     D0          3401          MOVL    (R2)+,-(SP)
                             3404          PRINT   1,<-
                             3404  R6:             !XL>
    7E    82     D0          3411          MOVL    (R2)+,-(SP)
                             3414          PRINT   1,<-
                             3414  R7:             !XL>
    7E    82     D0          3421          MOVL    (R2)+,-(SP)
                             3424          PRINT   1,<-
                             3424  R8:             !XL>
    7E    82     D0          3431          MOVL    (R2)+,-(SP)
                             3434          PRINT   1,<-
```

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

```
                          34A4          .SBTTL  SHOW_RJB
                          34A4
                          34A4  ;+++
                          34A4  ; SHOW_RJB -- Show information from the RJB.
                          34A4  ;
                          34A4  ; Inputs:
                          34A4  ;
                          34A4  ;       4(AP) = Address of buffer containing RJB.
                          34A4  ;
                          34A4  ; Outputs:
                          34A4  ;
                          34A4  ;       RJB displayed.
                          34A4  ;---
                          34A4
                          34A4  SHOW_RJB:
                    001C  34A4          .WORD   ^M<R2,R3,R4>
        53   04 AC    D0  34A6          MOVL    4(AP),R3                ; get address of block buffer
                          34AA          ENSURE  10
                          34C2          SKIP    1
             FC A3    DD  34CB          PUSHL   -4(R3)                  ; push virtual addresss of RJB
                          34CE          PRINT   1,<-
                          34CE  RJB Address:  !XL>
                          34DB          PRINT   0,<-
                          34DB  ------------->
                          34E8          SKIP    1
        7E   08 A3    9A  34F1          MOVZBL  RJB$B_BID(R3),-(SP)
                63    DD  34F5          PUSHL   RJB$Q_CHAN(R3)
                          34F7          PRINT   2,<-
                          34F7  CHAN:                   !XL             -
                          34F7  BID:                    !XB     !-!8UB.>
        7E   09 A3    9A  3504          MOVZBL  RJB$B_BLN(R3),-(SP)
             04 A3    DD  3508          PUSHL   RJB$Q_CHAN+4(R3)
                          350B          PRINT   2,<-
                          350B  CHAN+4:                 !XL             -
                          350B  BLN:                    !XB     !-!8UB.>
                          3518          ALLOC   ALLOCSZ,R4
        7E   0A A3    3C  352A          MOVZWL  RJB$W_FLAGS(R3),-(SP)
           D61A CF    9F  352E          PUSHAB  RJB_CHR
00000000'EF     02    FB  3532          CALLS   #2,TRANSLATE_BITS
                54    DD  3539          PUSHL   R4
        7E   0A A3    3C  353B          MOVZWL  RJB$W_FLAGS(R3),-(SP)
                          353F          PRINT   2,<-
                          353F  FLAGS:                  !XW     !AS>
                          354C          STATUS  SUCCESS
                      04  3553          RET
```

```
                            3554          .SBTTL  SHOW_MJB
                            3554  ;+++
                            3554  ; SHOW_MJB -- Show information from the MJB.
                            3554  ;
                            3554  ; Inputs:
                            3554  ;
                            3554  ;       4(AP) = Address of buffer containing MJB.
                            3554  ;
                            3554  ; Outputs:
                            3554  ;
                            3554  ;       MJB displayed.
                            3554  ;---
                            3554
                            3554  SHOW_MJB:
                      001C  3554          .WORD   ^M<R2,R3,R4>
        53    04 AC     D0  3556          MOVL    4(AP),R3                ; get address of block buffer
                            355A          ENSURE  10
                            3572          SKIP    1
              FC A3     DD  357B          PUSHL   -4(R3)                  ; push virtual addresses of MJB
                            357E          PRINT   1,<-
                            357E  MJB Address:    !XL>
                            358B          PRINT   0,<-
                            358B  ------------>
                            3598          SKIP    1
        7E    08 A3     9A  35A1          MOVZBL  MJB$B_BID(R3),-(SP)
        7E    0C A3     9A  35A5          MOVZBL  MJB$B_JNL(R3),-(SP)
                            35A9          PRINT   2,<-
                            35A9  JNL:            !XL                 -
                            35A9  BID:            !XB         !-!8UB.>
        7E    09 A3     9A  35B6          MOVZBL  MJB$B_BLN(R3),-(SP)
              10 A3     DD  35BA          PUSHL   MJB$Q_DESC(R3)
                            35BD          PRINT   2,<-
                            35BD  DESC:           !XL                 -
                            35BD  BLN:            !XB         !-!8UB.>
              14 A3     DD  35CA          PUSHL   MJB$Q_DESC+4(R3)
                            35CD          PRINT   1,<-
                            35CD  DESC+4:         !XL                 >
              1C A3     DD  35DA          PUSHL   MJB$Q_IOSB+4(R3)
              18 A3     DD  35DD          PUSHL   MJB$Q_IOSB(R3)
                            35E0          PRINT   2,<-
                            35E0  IOSB:           !XL                 -
                            35E0  IOSB+4:         !XL                 >
                            35ED          ALLOC   ALLOCSZ,R4
        7E    0A A3     3C  35FF          MOVZWL  MJB$W_FLAGS(R3),-(SP)
            D575 CF     9F  3603          PUSHAB  MJB_CHR
  00000000'EF     02    FB  3607          CALLS   #2,TRANSLATE_BITS
                  54    DD  360E          PUSHL   R4
        7E    0A A3     3C  3610          MOVZWL  MJB$W_FLAGS(R3),-(SP)
                            3614          PRINT   2,<-
                            3614  FLAGS:          !XW     !AS>
                            3621          STATUS  SUCCESS
                        04  3628          RET
```

```
                          3629          .SBTTL  SHOW_GBH
                          3629
                          3629  ;+++
                          3629  ; SHOW_GBH -- Show useful information from Global Buffer Header.
                          3629  ;
                          3629  ;  Inputs:
                          3629  ;
                          3629  ;       4(AP) = Address of block containing GBH.
                          3629  ;
                          3629  ;  Outputs:
                          3629  ;
                          3629  ;       GBH displayed.
                          3629  ;---
                          3629
                    001C  3629  SHOW_GBH:.WORD  ^M<R2,R3,R4>
        53   04 AC    D0  362B          MOVL    4(AP),R3                 ; get address of block buffer
                          362F          SKIP    PAGE                     ; format display
             FC A3    DD  3636          PUSHL   -4(R3)                   ; address of GBH
                          3639          PRINT   1,<-
                          3639  GBH Address: !XL>
                          3646          PRINT   0,<-
                          3646  ------------->
                          3653          SKIP    1                        ; one line down
        7E   08 A3    9A  365C          MOVZBL  GBH$B_BID(R3),-(SP)
                63    DD  3660          PUSHL   GBH$L_GBD_FLNK(R3)
                          3662          PRINT   2,<-
                          3662  GBD_FLNK:               !XL             -
                          3662  BID:                    !XB     !-!8UB.>
        7E   09 A3    9A  366F          MOVZBL  GBH$B_BLN(R3),-(SP)
             04 A3    DD  3673          PUSHL   GBH$L_GBD_BLNK(R3)
                          3676          PRINT   2,<-
                          3676  GBD_BLNK:               !XL             -
                          3676  BLN:                    !XB     !-!8UB.>
                          3683          ALLOC   ALLOCSZ,R4
        7E   0A A3    3C  3695          MOVZWL  GBH$W_TRC_FLGS(R3),-(SP)
           D28F CF    9F  3699          PUSHAB  GBH_CHR
 00000000'EF    02    FB  369D          CALLS   #2,TRANSLATE_BITS
                54    DD  36A4          PUSHL   R4
        7E   0A A3    3C  36A6          MOVZWL  GBH$W_TRC_FLGS(R3),-(SP)
                          36AA          PRINT   2,<-
                          36AA  FLGS:                   !XW     !AS>
             20 A3    DD  36B7          PUSHL   GBH$L_TRC_FLNK(R3)
             28 A3    DD  36BA          PUSHL   GBH$L_GBD_START(R3)
                          36BD          PRINT   2,<-
                          36BD  GBD_START:              !XL             -
                          36BD  TRC_FLNK:               !XL>
             24 A3    DD  36CA          PUSHL   GBH$L_TRC_BLNK(R3)
             2C A3    DD  36CD          PUSHL   GBH$L_GBD_END(R3)
                          36D0          PRINT   2,<-
                          36D0  GBD_END:                !XL             -
                          36D0  TRC_BLNK:               !XL>
             34 A3    DD  36DD          PUSHL   GBH$L_SCAN_NUM(R3)
             30 A3    DD  36E0          PUSHL   GBH$L_GBD_NEXT(R3)
                          36E3          PRINT   2,<-
                          36E3  GBD_NEXT:               !XL             -
                          36E3  SCAN_NUM:               !XL!-!8UL.>
             38 A3    DD  36F0          PUSHL   GBH$L_HIT(R3)
```

```
40 A3   DD  36F3              PUSHL   GBH$L_READ(R3)
            36F6              PRINT   2,<-
            36F6   READ:              !XL!-!8UL.          -
            36F6   HIT:               !XL!-!8UL.>
3C A3   DD  3703              PUSHL   GBH$L_MISS(R3)
44 A3   DD  3706              PUSHL   GBH$L_WRITE(R3)
            3709              PRINT   2,<-
            3709   WRITE:             !XL!-!8UL.          -
            3709   MISS:              !XL!-!8UL.>
10 A3   DD  3716              PUSHL   GBH$L_GS_SIZE(R3)
48 A3   DD  3719              PUSHL   GBH$L_DFW_WRITE(R3)
            371C              PRINT   1,<-
            371C   DFW_WRITE:         !XL!-!8UL.          -
            371C   GS_SIZE:           !XL!-!8UL.>
14 A3   DD  3729              PUSHL   GBH$L_LOCK_ID(R3)
1C A3   DD  372C              PUSHL   GBH$L_USECNT(R3)
            372F              PRINT   2,<-
            372F   USECNT:            !XL!-!8UL.          -
            372F   LOCK_ID:           !XL!-!8UL.>
            373C              STATUS  SUCCESS
        04  3743              RET
```

```
                                  3744            .SBTTL  SHOW_GBD_SUM
                                  3744
                                  3744    ;+++
                                  3744    ; SHOW_GBD_SUM -- Show summary of GBD information.  In this case that's
                                  3744    ; all there is in a GBD.
                                  3744    ;
                                  3744    ;  Inputs:
                                  3744    ;
                                  3744    ;       None.
                                  3744    ;
                                  3744    ;  Outputs:
                                  3744    ;
                                  3744    ;       None.
                                  3744    ;--
                                  3744
                                  3744            .ENABL  LSB
                                  3744    SHOW_GBD_SUM:
                            003C  3744            .WORD   ^M<R2,R3,R4,R5>
   03 00000006'EF    15      E0   3746            BBS     #OPT$V_GBDSUM,RMS_DIS_OPT1,10$
                   012C      31   374E    5$:     BRW     90$
     52    000013C6'EF       9E   3751    10$:    MOVAB   GBD,R2                         ; set up R2 with block buff address
  FC A2    000013EE'EF       D0   3758            MOVL    GBH-4,-4(R2)                   ; begin to set up for "1st" GBD
                                  3760
                                  3760            ASSUME  GBH$L_GBD_FLNK EQ 0
                                  3760
     62    000013F2'EF       D0   3760            MOVL    GBH,GBD$L_FLINK(R2)            ; set up forward link
                     E5      13   3767            BEQL    5$
                                  3769            SKIP    PAGE
                                  3770            PRINT   0,<-
                                  3770                                          GBD Summary>
                                  377D            PRINT   0,<-
                                  377D                                          ----------->
                                  378A            SKIP    1
                                  3793            PRINT   0,<-
                                  3793     GBD     USE                                          REL_  CACHE_>
                                  37A0            PRINT   0,<-
                                  37A0    Address  CNT  SIZE  NUMB  VBN  VBNSEQNUM  ADDR   VAL   LOCK_ID FLAGS>
                                  37AD            PRINT   0,<-
                                  37AD    -------  ---  ----  ---- ----  ---------  ----   ---   ------- ----->
                                  37BA            SKIP    1
                                  37C3            ALLOC   ALLOCSZ,R4
                     53      D4   37D5            CLRL    R3                             ; R3 = count of GBD's
     55    000013F6'EF       D0   37D7            MOVL    GBH$L_GBD_BLNK+GBH,R5          ; R5 = address of last GBD
     55    000013EE'EF       C0   37DE            ADDL2   GBH-4,R5                       ; make absolute address
                                  37E5
          62     FC A2       C0   37E5    20$:    ADDL2   -4(R2),GBD$L_FLINK(R2)         ; make rel addr abs addr
                                  37E9            GET     GBD,@GBD$L_FLINK(R2),90$
                     53      D6   3816            INCL    R3                             ; count the GBD's found
                                  3818
     6E    00000087  8F      D0   3818            MOVL    #ALLOCSZ,(SP)
          7E     0A A2       9A   381F            MOVZBL  GBD$B_FLAGS(R2),-(SP)
              D18D CF        9F   3823            PUSHAB  GBD_CHR
   00000000'EF       02      FB   3827            CALLS   #2,TRANSLATE_BITS
                     54      DD   382E            PUSHL   R4
                  14 A2      DD   3830            PUSHL   GBD$L_LOCK_ID(R2)
          7E      0B A2      9A   3833            MOVZBL  GBD$B_CACHE_VAL(R2),-(SP)
                  1C A2      DD   3837            PUSHL   GBD$L_REL_ADDR(R2)
```

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

```
                              3885          .SBTTL  SHOW_TRACE
                              3885
                              3885  ;+++
                              3885  ; SHOW_TRACE -- Show trace information from trace block pool.
                              3885  ;
                              3885  ; Inputs:
                              3885  ;
                              3885  ;          4(AP) = Address of trace queue header in SDA process
                              3885  ;          8(AP) = Address of trace queue header in dumped process
                              3885  ;         12(AP) = Trace flags (1=yes)
                              3885  ;         16(AP) = Depth to trace (0=all)
                              3885  ;
                              3885  ; Outputs:
                              3885  ;
                              3885  ;         Trace contents dumped to current output file/device.
                              3885  ;---
                              3885
                              3885          .ENABL  LSB
                              3885  SHOW_TRACE:
                      03FC    3885          .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9>
        54    04 AC     7D    3887          MOVQ    4(AP),R4                        ; get arguments
        56    0C AC     7D    388B          MOVQ    12(AP),R6                       ; more arguments
 52   0000145E'EF       DE    388F          MOVAL   TRC,R2                          ; address of TRC block buffer
                              3896
                              3896  ;
                              3896  ; Note well that trace queue is a self relative queue!
                              3896  ;
                              3896  ; R2 = Address of trace block buffer (in SDA)
                              3896  ; R4 = Address of trace queue header (in SDA)
                              3896  ; R5 = Address of trace queue header in process being dumped
                              3896  ; R6 = Trace Flags
                              3896  ; R7 = Depth (0=all)
                              3896  ;
                              3896
     04 A2    04 A4     D0    3896  3$:     MOVL    4(R4),TRC$L_BLNK(R2)            ; move q header back link
                03      12    389B          BNEQ    5$                              ; if neq queue is not empty
              027B      31    389D          BRW     90$                             ; if eql empty queue
     FC A2      55      D0    38A0  5$:     MOVL    R5,-4(R2)                       ; fake up address of trc block
                              38A4                                                  ;  from address of queue header
        62      64      D0    38A4          MOVL    (R4),TRC$L_FLNK(R2)             ; set up to scan queue from front
                57      D5    38A7          TSTL    R7                              ; check trace depth
                47      13    38A9          BEQL    15$                             ; do entire q -- we're already set
     59   04 A4         D0    38AB          MOVL    4(R4),R9                        ; begin to form real address of end
        59      55      C0    38AF          ADDL    R5,R9                           ; form real address from self rel one
                57      D6    38B2          INCL    R7                              ; bump depth to account for algorithm
                              38B4
                              38B4  ;
                              38B4  ; At this point, we have a non-zero depth.  We will scan the trace queue
                              38B4  ; from the front until we have passed over the necessary number of queue
                              38B4  ; entries.  No dumping is done at this point, as we are going in the wrong
                              38B4  ; order.  After we have passed over the proper number of queue entries, the
                              38B4  ; we will enter the loop that scans them in the other direction, dumping
                              38B4  ; as it goes.
                              38B4  ;
                              38B4  ; R2 = address of trace queue block buffer ( TRC )
                              38B4  ; R7 = Depth (initially +1)
                              38B4  ; R9 = Address of last trace queue entry (when we get there we're done)
```

```
                        38B4    ;
                        38B4
        62  FC A2  C0   38B4    10$:    ADDL    -4(R2),TRC$L_FLNK(R2)      ; make self-rel q absolute address
                        38B8            GET     TRC,@TRC$L_FLNK(R2),90$    ; get next trace queue element
               57  D7   38E5            DECL    R7                         ; count this one in the down count
               09  13   38E7            BEQL    15$                        ; we've done enough
        59  FC A2  D1   38E9    12$:    CMPL    -4(R2),R9                  ; are we done with all elements?
               C5  12   38ED            BNEQ    10$                        ; if neq no, do some more
             FFA4  31   38EF            BRW     3$                         ; yes, too bad.  Go dump them all.
                        38F2
                        38F2    ;
                        38F2    ; Now we are about to go into the loop which scans from the trace
                        38F2    ; element currently in the TRC block buffer, forward until the
                        38F2    ; beginning of the queue.  The TRC block buffer either has the depth+1'th
                        38F2    ; element in it, or it has enough of the queue header such that
                        38F2    ; scanning forward from it will work.
                        38F2    ;
                        38F2
           59   62  D0  38F2    15$:    MOVL    TRC$L_FLNK(R2),R9          ; begin to form address of newest entry
        59  FC A2  C0   38F5            ADDL    -4(R2),R9                          ; turn it into absolute address
                        38F9
                        38F9    ;
                        38F9    ; Set up constant part of header
                        38F9    ;
                        38F9
                        38F9            SKIP    PAGE
                        3900            PRINT   0,<-
                        3900                                    RMS Trace>
                        390D            PRINT   0,<-
                        390D                                    --------->
                        391A            SKIP    3
                        3923            ALLOC   ALLOCSZ,R4                 ; set up translate bits buffer
                        3935            PRINT   0,<-
                        3935              PROC                                                              TRC>
                        3942            PRINT   0,<-
                        3942    OPERATION  IN  STRUCTURE    VBN                          SEQUENCE         ADDRESS>
                        394F            PRINT   0,<-
                        394F    ---------  --  ---------    ---                          --------         ------->
                        395C            SKIP 1
                        3965
                        3965    ;
                        3965    ; R2 = pointer to TRC block buffer
                        3965    ; R6 = trace flags
                        3965    ; R9 = pointer to latest entry in the trace queue, after which we should stop
                        3965    ;
                        3965
  04 A2     FC A2  C0   3965    20$:    ADDL    -4(R2),TRC$L_BLNK(R2)      ; turn rel into absolute address
                        396A            GET     TRC,@TRC$L_BLNK(R2),90$    ; get next trace block
                        3997
                        3997    ;
                        3997    ; Process this trace block for display information
                        3997    ;
                        3997
        56  0A A2  D3   3997            BITL    TRC$W_FUNCTION(R2),R6      ; is this trace block wanted?
               03  12   399B            BNEQ    21$
             0172  31   399D            BRW     50$
  6E   00000087 8F  D0  39A0    21$:    MOVL    #ALLOCSZ,(SP)              ; renew translate_bits buffer
```

```
        7E   0A A2   3C  39A7          MOVZWL  TRC$W_FUNCTION(R2),-(SP) ; put bits to translate
           CF7D CF   9F  39AB          PUSHAB  GBH_CHR                  ; translation table
  00000000'EF   02   FB  39AF          CALLS   #2,TRANSLATE_BITS        ; do the translation
                         39B6
             FC A2   DD  39B6          PUSHL   -4(R2)                   ; virtual address of trace block
        7E   12 A2   3C  39B9          MOVZWL  TRC$W_SEQNUM(R2),-(SP)   ; sequence number of this trace block
             14 A2   DD  39BD          PUSHL   TRC$L_VBN(R2)
             0C A2   DD  39C0          PUSHL   TRC$L_STRUCTURE(R2)
        7E   10 A2   3C  39C3          MOVZWL  TRC$W_PID(R2),-(SP)      ; just process index portion of pid
                54   DD  39C7          PUSHL   R4                       ; the function type
                         39C9          PRINT   6,<-
                         39C9  !9AS !XW !XL !7UL                        !5UW          !XL>
                         39D6
                         39D6
     0A A2      05   D3  39D6          BITL    #<GBH$M_CACHE_IN!GBH$M_RLS_IN>,TRC$W_FUNCTION(R2) ; interesting
                33   13  39DA          BEQL    25$                      ; if eql no, just do return2
                         39DC
                         39DC  ;
                         39DC  ; Dump the argument flags for cache and release
                         39DC  ;
                         39DC  ; If not cache or release, just dump return status (ARG_FLG for now)
                         39DC  ;
                         39DC
  6E     00000087 8F   D0  39DC          MOVL    #ALLOCSZ,(SP)
             20 A2   DD  39E3          PUSHL   TRC$L_ARG_FLG(R2)
           CDAE CF   9F  39E6          PUSHAB  RLS_CHR
     05 0A A2   02   E0  39EA          BBS     #GBH$V_RLS_IN,TRC$W_FUNCTION(R2),22$
      6E CD7D CF   9E  39EF          MOVAB   CSH_CHR,(SP)
  00000000'EF   02   FB  39F4  22$:    CALLS   #2,TRANSLATE_BITS
                         39FB
                54   DD  39FB          PUSHL   R4
             20 A2   DD  39FD          PUSHL   TRC$L_ARG_FLG(R2)
                         3A00          PRINT   2,<-
                         3A00                                    ARGFLG=!XL !AS>
                10   11  3A0D          BRB     30$
             20 A2   DD  3A0F  25$:    PUSHL   TRC$L_ARG_FLG(R2)
                         3A12          PRINT   1,<-
                         3A12                                    R0=!XL>
                         3A1F
                         3A1F  ;
                         3A1F  ; Symbolize the return addresses
                         3A1F  ;
                         3A1F
             18 A2   D5  3A1F  30$:    TSTL    TRC$L_RETURN1(R2)        ; should we dump returns?
                4A   13  3A22          BEQL    35$                      ; if return1=0 then no.
  6E     00000087 8F   D0  3A24          MOVL    #ALLOCSZ,(SP)            ; reset string buffer
                54   DD  3A2B          PUSHL   R4                       ; address of buffer descriptor
             18 A2   DD  3A2D          PUSHL   TRC$L_RETURN1(R2)        ; address of return
  00000000'EF   02   FB  3A30          CALLS   #2,SYMBOLIZE             ; get string for this symbol
                51   DD  3A37          PUSHL   R1                       ; address of descriptor
             18 A2   DD  3A39          PUSHL   TRC$L_RETURN1(R2)        ; address itself
                         3A3C          PRINT   2,<-
                         3A3C                                    RETURN=!XL !AS>
  6E     00000087 8F   D0  3A49          MOVL    #ALLOCSZ,(SP)
                54   DD  3A50          PUSHL   R4
             1C A2   DD  3A52          PUSHL   TRC$L_RETURN2(R2)
  00000000'EF   02   FB  3A55          CALLS   #2,SYMBOLIZE
```

```
                51  DD  3A5C                  PUSHL   R1
             1C A2  DD  3A5E                  PUSHL   TRC$L_RETURN2(R2)
                        3A61                  PRINT   2,<-
                        3A61                                                  !XL !AS>
                        3A6E
                        3A6E        ;
                        3A6E        ; If this element is proper, then dump the BDB information held within
                        3A6E        ; the element.  Allow TRC$L_BDB_ADDR = 0 to signal no BDB information.
                        3A6E        ;
                        3A6E
             24 A2  D5  3A6E  35$:    TSTL    TRC$L_BDB_ADDR(R2)        ; if the address = 0
                49  13  3A71          BEQL    40$                       ; if eql yes, skip BDB
             2E A2  DD  3A73          PUSHL   TRC$L_BDB_SEQ(R2)         ; buffer sequence number
      7E     2A A2  3C  3A76          MOVZWL  TRC$W_BDB_BUFF(R2),-(SP)
      7E     2C A2  9A  3A7A          MOVZBL  TRC$B_BDB_CACHE(R2),-(SP)
      7E     28 A2  3C  3A7E          MOVZWL  TRC$W_BDB_USERS(R2),-(SP)
             24 A2  DD  3A82          PUSHL   TRC$L_BDB_ADDR(R2)
                        3A85          PRINT   5,<-
                        3A85                                    BDB=!XL U=!4UW C=!XB B=!XW S=!XL>
                        3A92
                        3A92        ;
                        3A92        ; Interpret and dump flags, if any.
                        3A92        ;
                        3A92
             2D A2  95  3A92          TSTB    TRC$B_BDB_FLAGS(R2)
                25  13  3A95          BEQL    40$
  6E    00000087 8F  D0  3A97          MOVL    #ALLOCSZ,(SP)
      7E     2D A2  9A  3A9E          MOVZBL  TRC$B_BDB_FLAGS(R2),-(SP)
           C9DA CF  9F  3AA2          PUSHAB  BDB_CHR
 00000000'EF    02  FB  3AA6          CALLS   #2,TRANSLATE_BITS
                54  DD  3AAD          PUSHL   R4
                        3AAF          PRINT   1,<-
                        3AAF                                        F=!AS>
                        3ABC
                        3ABC        ;
                        3ABC        ; Print the BLB trace information if the TRC$L_BLB_ADDR doesn't equal zero.
                        3ABC        ;
                        3ABC
             34 A2  D5  3ABC  40$:    TSTL    TRC$L_BLB_ADDR(R2)
                51  13  3ABF          BEQL    50$
             3C A2  DD  3AC1          PUSHL   TRC$L_BLB_SEQ(R2)         ; blb sequence number
             38 A2  DD  3AC4          PUSHL   TRC$L_BLB_LOCK(R2)
      53     32 A2  9A  3AC7          MOVZBL  TRC$B_BLB_MODE(R2),R3     ; get the mode held byte
           53    06  91  3ACB          CMPB    #6,R3                     ; is it less then 6
                03  1A  3ACE          BGTRU   42$                       ; if gtru then yes
           53    06  9A  3AD0          MOVZBL  #6,R3                     ; set to 6 (i.e. ??)
      CE38 CF43  DD  3AD3  42$:    PUSHL   BLB_TBL[R3]               ; set translation into args
             34 A2  DD  3AD8          PUSHL   TRC$L_BLB_ADDR(R2)
                        3ADB          PRINT   4,<-
                        3ADB                                    BLB=!XL M=!AC L=!XL    S=!XL>
                        3AE8
                        3AE8        ;
                        3AE8        ; Interpret flags (if any).
                        3AE8        ;
                        3AE8
             33 A2  95  3AE8          TSTB    TRC$B_BLB_FLAGS(R2)
                25  13  3AEB          BEQL    50$
```

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

```
                              3B23          .SBTTL  SHOW_NAM
                              3B23
                              3B23  ;+++
                              3B23  ; SHOW_NAM -- Show valuable information in NAM block.
                              3B23  ;
                              3B23  ; Inputs:
                              3B23  ;
                              3B23  ;       4(AP) = Address of NAM block in "Block Buffer".
                              3B23  ;
                              3B23  ; Outputs:
                              3B23  ;
                              3B23  ;       None.
                              3B23  ;---
                              3B23
                              3B23  SHOW_NAM:
                      001C    3B23          .WORD   ^M<R2,R3,R4>
        52   04 AC     D0     3B25          MOVL    4(AP),R2                ; Get address of buffer in R2
             14 A2     DF     3B29          PUSHAL  NAM$T_DVI(R2)           ; Push address of Device ID Field
             0C A2     DD     3B2C          PUSHL   NAM$L_ESA(R2)           ; Expanded string address
        7E   0B A2     9A     3B2F          MOVZBL  NAM$B_ESL(R2),-(SP)     ; Expanded string length
             04 A2     DD     3B33          PUSHL   NAM$L_RSA(R2)           ; Resultant string address
        7E   03 A2     9A     3B36          MOVZBL  NAM$B_RSL(R2),-(SP)     ; Resultant string length
                              3B3A          PRINT   5,<-
                              3B3A  Resultant String Name:   !AF!/-
                              3B3A  Expanded String Name:    !AF!/-
                              3B3A  Device Identification:   !AC>
                              3B47          PRINT   0,<-
                              3B47                                          RVN     File #     File Seq. #!/-
                              3B47                                          ---     ------     ----------->
                              3B54          ALLOC   80,R4                   ; Allocate buffer for status bits
             34 A2     DD     3B66          PUSHL   NAM$L_FNB(R2)           ; Set up args to TRANSLATE_BITS
           C55B CF     9F     3B69          PUSHAB  NAM_CHR                 ; Character translation table
  00000000'EF    02    FB     3B6D          CALLS   #2,TRANSLATE_BITS
                54     DD     3B74          PUSHL   R4                      ; Send buffer to print routine
             34 A2     DD     3B76          PUSHL   NAM$L_FNB(R2)           ; Send bare bits too
                              3B79          PRINT   2,<-
                              3B79  Status Bits (FNB):        !XL     !AS>
             30 A2     DD     3B86          PUSHL   NAM$L_WCC(R2)           ; Set up wildcard context bits
                              3B89          PRINT   1,<-
                              3B89  Wildcard Context:         !XL>
                      04      3B96          RET
```

```
                         3B97           .SBTTL  GET_IFAB
                         3B97
                         3B97  ;+++
                         3B97  ; GET_IFAB -- Get and Verify IFAB from Dump.
                         3B97  ;
                         3B97  ; Inputs:
                         3B97  ;
                         3B97  ;        4(AP) = Block Buffer.
                         3B97  ;
                         3B97  ; Outputs:
                         3B97  ;
                         3B97  ;        Buffer contains IFAB if success.
                         3B97  ;        If failure, BUFFER-4 = 0.
                         3B97  ;---
                         3B97
                         3B97  GET_IFAB:
                   001C  3B97           .WORD   ^M<R2,R3,R4>
   53     04 AC      D0  3B99           MOVL    4(AP),R3                    ; Get address of buffer
                         3B9D           GETMEM  @-4(R3),(R3),#IFB$C_BLN
          15 50      E9  3BAF           BLBC    R0,100$
   08 A3     0B      91  3BB2           CMPB    #IFB$C_BID,IFB$B_BID(R3)
             08      12  3BB6           BNEQ    99$
                         3BB8  90$:     STATUS  SUCCESS
                     04  3BBF           RET
                         3BC0  99$:     STATUS  INVBLKTYP
                     04  3BC7  100$:    RET
```

```
                          3BC8          .SBTTL  GET_IDX
                          3BC8   ;+++
                          3BC8   ; GET_IDX -- Get and verify Index descriptor.
                          3BC8   ;
                          3BC8   ; Inputs:
                          3BC8   ;
                          3BC8   ;       4(AP) = Address of block buffer.
                          3BC8   ;
                          3BC8   ; Outputs:
                          3BC8   ;
                          3BC8   ;       Buffer contains index descriptor.
                          3BC8   ;---
                          3BC8
                    001C  3BC8   GET_IDX:.WORD  ^M<R2,R3,R4>
 53      04 AC      D0    3BCA          MOVL    4(AP),R3                ; address of buffer
                          3BCE          GETMEM  @-4(R3),(R3),#IDX$C_FIXED_BLN+32
         15 50      E9    3BE0          BLBC    R0,100$
 08 A3      0F      91    3BE3          CMPB    #IDX$C_BID,IDX$B_BID(R3)
            08      12    3BE7          BNEQ    99$
                          3BE9   90$:   STATUS  SUCCESS
                    04    3BF0          RET
                          3BF1   99$:   STATUS  INVBLKTYP
                    04    3BF8   100$:  RET
```

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

```
                          3C2C             .SBTTL  GET_CCB
                          3C2C      ;+++
                          3C2C      ; GET_CCB -- Get and verify Channel Control Block from dump.
                          3C2C      ;
                          3C2C      ; Inputs:
                          3C2C      ;
                          3C2C      ;        4(AP) = Address of block buffer.
                          3C2C      ;
                          3C2C      ; Outputs:
                          3C2C      ;
                          3C2C      ;        Block buffer contains CCB.
                          3C2C      ;---
                          3C2C
                          3C2C      GET_CCB:
                   001C   3C2C             .WORD   ^M<R2,R3,R4>
       53    04 AC   D0   3C2E             MOVL    4(AP),R3
                          3C32             GETMEM  @CTL$GL_CCBBASE,(R3),#4
FC A3  63    FC A3   C3   3C42             SUBL3   -4(R3),(R3),-4(R3)
                          3C48             GETMEM  @-4(R3),(R3),#CCB$C_LENGTH
             13 50   E9   3C56             BLBC    R0,100$
                          3C59             STATUS  INVBLKTYP
             09 A3   95   3C60             TSTB    CCB$B_AMOD(R3)
                07   13   3C63             BEQL    100$
                          3C65             STATUS  SUCCESS
                     04   3C6C      100$:  RET
```

```
                     3C6D          .SBTTL  GET_WCB
                     3C6D  ;+++
                     3C6D  ; GET_WCB -- Get and verify Window Contol Block from dump.
                     3C6D  ;
                     3C6D  ; Inputs:
                     3C6D  ;
                     3C6D  ;       4(AP) = Address of block buffer.
                     3C6D  ;
                     3C6D  ; Outputs:
                     3C6D  ;
                     3C6D  ;       Block buffer contains WCB.
                     3C6D  ;---
                     3C6D
                     3C6D  GET_WCB:
               001C  3C6D          .WORD   ^M<R2,R3,R4>
     53   04 AC   D0  3C6F          MOVL    4(AP),R3
                     3C73          GETMEM  @-4(R3),(R3),#WCB$C_LENGTH+512
        15 50   E9  3C85          BLBC    R0,100$
     0A A3   12   91  3C88          CMPB    #DYN$C_WCB,WCB$B_TYPE(R3)
           08   12  3C8C          BNEQ    99$
                     3C8E  90$:    STATUS  SUCCESS
                04  3C95          RET
                     3C96  99$:    STATUS  INVBLKTYP
                04  3C9D  100$:   RET
```

```
                         3C9E         .SBTTL  GET_FCB
                         3C9E  ;+++
                         3C9E  ; GET_FCB -- Get and verify File Control Block from dump.
                         3C9E  ;
                         3C9E  ; Inputs:
                         3C9E  ;
                         3C9E  ;      4(AP) = Address of block buffer.
                         3C9E  ;
                         3C9E  ; Outputs:
                         3C9E  ;
                         3C9E  ;      Block buffer contains FCB.
                         3C9E  ;---
                         3C9E
                         3C9E  GET_FCB:
                   001C  3C9E         .WORD   ^M<R2,R3,R4>
      53    04 AC    D0  3CA0         MOVL    4(AP),R3
                         3CA4         GETMEM  @-4(R3),(R3),#FCB$C_LENGTH
            15 50    E9  3CB6         BLBC    R0,100$
   0A A3    07       91  3CB9         CMPB    #DYN$C_FCB,FCB$B_TYPE(R3)
            08       12  3CBD         BNEQ    99$
                         3CBF  90$:   STATUS  SUCCESS
                     04  3CC6         RET
                         3CC7  99$:   STATUS  INVBLKTYP
                     04  3CCE  100$:  RET
```

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

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

```
                        3D40         .SBTTL  GET_BLB
                        3D40  ;+++
                        3D40  ; GET_BLB -- Get and verify BLB.
                        3D40  ;
                        3D40  ; Inputs:
                        3D40  ;
                        3D40  ;        4(AP) = Address of buffer to fill with BLB.
                        3D40  ;
                        3D40  ; Ouputs:
                        3D40  ;
                        3D40  ;        None.
                        3D40  ;---
                        3D40
                 001C   3D40  GET_BLB:.WORD  ^M<R2,R3,R4>
53    04 AC      D0     3D42          MOVL   4(AP),R3                    ; address of buffer for BLB
                        3D46          GETMEM @-4(R3),(R3),#BLB$C_BLN
      15 50      E9     3D54          BLBC   R0,100$                     ; if error in memory access
08 A3    10      91     3D57          CMPB   #BLB$C_BID,BLB$B_BID(R3)    ; is this valid BLB?
         08      12     3D5B          BNEQ   99$
                        3D5D  90$:    STATUS SUCCESS
                 04     3D64          RET
                        3D65  99$:    STATUS INVBLKTYP
                 04     3D6C  100$:   RET
```

```
                           3D6D                 .SBTTL  GET_GBD
                           3D6D        ;+++
                           3D6D        ; GET_GBD -- Get and verify GBD.
                           3D6D        ;
                           3D6D        ;  Inputs:
                           3D6D        ;
                           3D6D        ;        4(AP) = Address of buffer to fill with GBD.
                           3D6D        ;
                           3D6D        ;  Ouputs:
                           3D6D        ;
                           3D6D        ;        Valid GBD.
                           3D6D        ;---
                           3D6D
                     001C  3D6D        GET_GBD:.WORD   ^M<R2,R3,R4>
53    04 AC          D0    3D6F                MOVL    4(AP),R3                      ; address of buffer for GBD
                           3D73                GETMEM  @-4(R3),(R3),#GBD$C_BLN
         15 50       E9    3D81                BLBC    R0,100$                       ; if error in memory access
08 A3       13       91    3D84                CMPB    #GBD$C_BID,GBD$B_BID(R3) ; is this valid GBD?
            08       12    3D88                BNEQ    99$
                           3D8A        90$:    STATUS  SUCCESS
                     04    3D91                RET
                           3D92        99$:    STATUS  INVBLKTYP
                     04    3D99        100$:   RET
```

```
                         3D9A        .SBTTL  GET_TRC
                         3D9A  ;+++
                         3D9A  ; GET_TRC -- Get and verify TRC.
                         3D9A  ;
                         3D9A  ; Inputs:
                         3D9A  ;
                         3D9A  ;         4(AP) = Address of buffer to fill with TRC.
                         3D9A  ;
                         3D9A  ; Ouputs:
                         3D9A  ;
                         3D9A  ;         None.
                         3D9A  ;---
                         3D9A
                  001C   3D9A  GET_TRC:.WORD   ^M<R2,R3,R4>
53     04 AC      D0     3D9C          MOVL    4(AP),R3                     ; address of buffer for TRC
                         3DA0          GETMEM  @-4(R3),(R3),#TRC$C_BLN
          15 50   E9     3DB2          BLBC    R0,100$                      ; if error in memory access
08 A3     12      91     3DB5          CMPB    #TRC$C_BID,TRC$B_BID(R3) ; is this valid TRC?
             08   12     3DB9          BNEQ    99$
                         3DBB  90$:    STATUS  SUCCESS
                  04     3DC2          RET
                         3DC3  99$:    STATUS  INVBLKTYP
                  04     3DCA  100$:   RET
```

```
                          3DCB         .SBTTL  GET_GBH
                          3DCB  ;+++
                          3DCB  ; GET_GBH -- Get and verify GBH.
                          3DCB  ;
                          3DCB  ; Inputs:
                          3DCB  ;
                          3DCB  ;       4(AP) = Address of buffer to fill with GBH.
                          3DCB  ;
                          3DCB  ; Ouputs:
                          3DCB  ;
                          3DCB  ;       None.
                          3DCB  ;---
                          3DCB
                    001C  3DCB  GET_GBH:.WORD   ^M<R2,R3,R4>
     53    04 AC     D0   3DCD          MOVL    4(AP),R3                        ; address of buffer for GBH
                          3DD1          GETMEM  @-4(R3),(R3),#GBH$C_BLN
           15 50     E9   3DE3          BLBC    R0,100$                         ; if error in memory access
     08 A3    11     91   3DE6          CMPB    #GBH$C_BID,GBH$B_BID(R3)        ; is this valid GBH?
              08     12   3DEA          BNEQ    99$
                          3DEC  90$:    STATUS  SUCCESS
                     04   3DF3          RET
                          3DF4  99$:    STATUS  INVBLKTYP
                     04   3DFB  100$:   RET
```

```
                            3DFC          .SBTTL  GET_RLB
                            3DFC  ;+++
                            3DFC  ; GET_RLB -- Get an RLB from the dump file.
                            3DFC  ;
                            3DFC  ; Inputs:
                            3DFC  ;
                            3DFC  ;         4(AP) = Address of block buffer for RLB.
                            3DFC  ;
                            3DFC  ; Outputs:
                            3DFC  ;
                            3DFC  ;         RLB in block buffer.
                            3DFC  ;---
                            3DFC
                      001C  3DFC  GET_RLB:.WORD   ^M<R2,R3,R4>
        53    04 AC   D0    3DFE          MOVL    4(AP),R3                      ; address of block buffer
                            3E02          GETMEM  @-4(R3),(R3),#RLB$C_BLN ; get memory from dump for this rlb
             15 50    E9    3E10          BLBC    R0,100$                       ; if error pass it back up
     08 A3      0E    91    3E13          CMPB    #RLB$C_BID,RLB$B_BID(R3) ; is this rlb?
                08    12    3E17          BNEQ    99$                           ; if neq no -- error
                            3E19          STATUS  SUCCESS
                      04    3E20          RET
                            3E21  99$:    STATUS  INVBLKTYP
                      04    3E28  100$:   RET
                            3E29
```

```
                          3E29        .SBTTL  GET_ASB
                          3E29   ;+++
                          3E29   ; GET_ASB -- Get an ASB from Dump.
                          3E29   ;
                          3E29   ; Inputs:
                          3E29   ;
                          3E29   ;       4(AP) = Address of block buffer for ASB.
                          3E29   ;
                          3E29   ; Outputs:
                          3E29   ;
                          3E29   ;       ASB in block buffer.
                          3E29   ;---
                          3E29
                    001C  3E29   GET_ASB:.WORD   ^M<R2,R3,R4>
       53   04 AC     D0  3E2B           MOVL    4(AP),R3
                          3E2F           GETMEM  @-4(R3),(R3),#ASB$C_BLN_IDX
            07 50     E9  3E41           BLBC    R0,100$
                          3E44           STATUS  SUCCESS
                      04  3E4B   100$:   RET
```

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

```
                   3E79         .SBTTL  GET_MJB
                   3E79
                   3E79  ;+++
                   3E79  ; GET_MJB -- Get an MJB from the dump file.
                   3E79  ;
                   3E79  ; Inputs:
                   3E79  ;
                   3E79  ;         4(AP) = Address of block buffer for MJB.
                   3E79  ;
                   3E79  ; Outputs:
                   3E79  ;
                   3E79  ;         MJB in block buffer.
                   3E79  ;---
                   3E79
              001C 3E79  GET_MJB:.WORD   ^M<R2,R3,R4>
53    04 AC   D0   3E7B          MOVL    4(AP),R3                ; address of block buffer
                   3E7F          GETMEM  @-4(R3),(R3),#MJB$C_BLN ; get memory from dump for this MJB
      15 50   E9   3E8D          BLBC    R0,100$                 ; if error pass it back up
08 A3    18   91   3E90          CMPB    #MJB$C_BID,MJB$B_BID(R3) ; is this an MJB?
         08   12   3E94          BNEQ    99$                     ; if neq no -- error
                   3E96          STATUS  SUCCESS
              04   3E9D          RET
                   3E9E  99$:    STATUS  INVBLKTYP
              04   3EA5  100$:   RET
                   3EA6
```

```
                        3EA6          .SBTTL  GET_FAB
                        3EA6  ;+++
                        3EA6  ; GET_FAB -- Get and Verify FAB from Dump.
                        3EA6  ;
                        3EA6  ; Inputs:
                        3EA6  ;
                        3EA6  ;       4(AP) = Address of block buffer for FAB.
                        3EA6  ;       IFI   = Contains the IFI of the current IFAB.
                        3EA6  ;
                        3EA6  ; Outputs:
                        3EA6  ;
                        3EA6  ;       FAB in block buffer if success, R0 contains
                        3EA6  ;       code if failure.  BUFFER-4 = 0 if failure.
                        3EA6  ;---
                        3EA6
                        3EA6  GET_FAB:
                  001C  3EA6          .WORD   ^M<R2,R3,R4>
       53    04 AC  D0  3EA8          MOVL    4(AP),R3
                        3EAC          GETMEM  @-4(R3),(R3),#FAB$C_BLN
          1E 50     E9  3EBE          BLBC    R0,100$
         63   03    91  3EC1          CMPB    #FAB$C_BID,FAB$B_BID(R3)
              12    12  3EC4          BNEQ    99$
 02 A3   00000016'EF B1 3EC6          CMPW    IFI,FAB$W_IFI(R3)          ; Is this the FAB for this IFAB
              08    12  3ECE          BNEQ    99$
                        3ED0  90$:    STATUS  SUCCESS
                    04  3ED7          RET
                        3ED8  99$:    STATUS  INVBLKTYP
                    04  3EDF  100$:   RET
```

SDA

```
                          3EE0          .SBTTL  GET_NAM
                          3EE0  ;+++
                          3EE0  ; GET_NAM -- Get and Verify NAM block from Dump.
                          3EE0  ;
                          3EE0  ; Inputs:
                          3EE0  ;
                          3EE0  ;       4(AP) = Address of block buffer for NAM block.
                          3EE0  ;
                          3EE0  ; Outputs:
                          3EE0  ;
                          3EE0  ;       Block buffer contains NAM block if success,
                          3EE0  ;       BUFFER-4 = 0 if failure, R0 = status.
                          3EE0  ;---
                          3EE0
                    001C  3EE0  GET_NAM:.WORD   ^M<R2,R3,R4>
       53    04 AC  D0    3EE2          MOVL    4(AP),R3
                          3EE6          GETMEM  @-4(R3),(R3),#NAM$C_BLN
             14 50  E9    3EF8          BLBC    R0,100$
          63    02  91    3EFB          CMPB    #NAM$C_BID,NAM$B_BID(R3)
                08  12    3EFE          BNEQ    99$
                          3F00  90$:    STATUS  SUCCESS
                    04    3F07          RET
                          3F08  99$:    STATUS  INVBLKTYP
                    04    3F0F  100$:   RET
                          3F10
                          3F10          .END
```

```
$$$                          = 00000570 R     04     BLB$C_BLN                    = 00000038
ALLOCSZ                      = 00000087              BLB$L_BDB_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                    = 0000000D
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
```

SDA

```
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
```

SDA
The
MES

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

**F

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

STA
V04-

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$L_PUTUP_VBN	= 00000078	IRB$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_SRCHGE	= 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_PUTUP_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_ROVHDSZ	= 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$W_SPLIT_2          = 0000004E                 NAM$V_WILD_SFD2        = 0000001A
IRB$W_SRCHFLAGS        = 00000042                 NAM$V_WILD_SFD3        = 0000001B
IRB$W_UDR_ID           = 000000BC                 NAM$V_WILD_SFD4        = 0000001C
IRBBKP_CHR               00000370 R     03        NAM$V_WILD_SFD5        = 0000001D
JNLFLG2_CHR              00000BA4 R     03        NAM$V_WILD_SFD6        = 0000001E
JNLFLG3_CHR              00000BD4 R     03        NAM$V_WILD_SFD7        = 0000001F
JNLFLG_CHR               00000338 R     03        NAM$V_WILD_TYPE        = 00000004
LIB$SIGNAL               ********   X   03        NAM$V_WILD_UFD         = 00000018
LINE_COUNT               ********   X   03        NAM$V_WILD_VER         = 00000003
LN_CHR                   00000ADC R     03        NAM_CHR                  000000C8 R     03
MJB                      000014A2 R     02        NEW_PAGE                 ********   X   03
MJB$B_BID              = 00000008                 OPT$M_ASB              = 00000040
MJB$B_BLN              = 00000009                 OPT$M_BDB              = 00000010
MJB$B_JNL              = 0000000C                 OPT$M_BLB              = 00008000
MJB$C_BID              = 00000018                 OPT$M_CCB              = 00000080
MJB$C_BLN              = 00000020                 OPT$M_FCB              = 00000200
MJB$Q_DESC             = 00000010                 OPT$M_GBD              = 00020000
MJB$Q_IOSB             = 00000018                 OPT$M_GBDSUM           = 00200000
MJB$V_FILE             = 00000002                 OPT$M_GBH              = 00040000
MJB$V_FORCE            = 00000001                 OPT$M_IDX              = 00000008
MJB$V_INIT             = 00000000                 OPT$M_IFB              = 00000002
MJB$V_SYNCH_SHARE      = 00000003                 OPT$M_IRB              = 00000004
MJB$W_FLAGS            = 0000000A                 OPT$M_RJB              = 00400000
MJB_CHR                  00000B7C R     03        OPT$M_RLB              = 00004000
MSG$_INVBLKTYP           ********   X   03        OPT$M_RMSALL           = FFFFFFFE
MSG$_NOIMGRMS            ********   X   03        OPT$M_TRC              = 00080000
MSG$_NOTVALID            ********   X   03        OPT$M_WCB              = 00000100
MSG$_RMSTERM             ********   X   03        OPT$V_ASB              = 00000006
MSG$_SUCCESS             ********   X   03        OPT$V_BDB              = 00000004
NAM                      00000CD6 R     02        OPT$V_BDBSUM           = 00000005
NAM$B_BID              = 00000000                 OPT$V_BLB              = 0000000F
NAM$B_ESL              = 0000000B                 OPT$V_BLBSUM           = 00000010
NAM$B_RSL              = 00000003                 OPT$V_CCB              = 00000007
NAM$C_BID              = 00000002                 OPT$V_FAB              = 0000000A
NAM$C_BLN              = 00000060                 OPT$V_FCB              = 00000009
NAM$L_ESA              = 0000000C                 OPT$V_FWA              = 00000014
NAM$L_FNB              = 00000034                 OPT$V_GBD              = 00000011
NAM$L_RSA              = 00000004                 OPT$V_GBDSUM           = 00000015
NAM$L_WCC              = 00000030                 OPT$V_GBH              = 00000012
NAM$T_DVI              = 00000014                 OPT$V_IDX              = 00000003
NAM$V_EXP_DEV          = 00000007                 OPT$V_IFB              = 00000001
NAM$V_EXP_DIR          = 00000006                 OPT$V_IRB              = 00000002
NAM$V_EXP_NAME         = 00000002                 OPT$V_NAM              = 0000000C
NAM$V_EXP_TYPE         = 00000001                 OPT$V_RAB              = 0000000B
NAM$V_EXP_VER          = 00000000                 OPT$V_RJB              = 00000016
NAM$V_GRP_MBR          = 00000013                 OPT$V_RLB              = 0000000E
NAM$V_HIGHVER          = 0000000F                 OPT$V_TRC              = 00000013
NAM$V_LOWVER           = 0000000E                 OPT$V_WCB              = 00000008
NAM$V_NODE             = 00000011                 OPT$V_XAB              = 0000000D
NAM$V_PPF              = 00000010                 OPT_CHR                  00000818 R     03
NAM$V_QUOTED           = 00000012                 ORG_TBL                  000004C0 R     03
NAM$V_WILDCARD         = 00000008                 PAGE_SIZE                ********   X   03
NAM$V_WILD_DIR         = 00000014                 PARSE_CHR                00000A7C R     03
NAM$V_WILD_GRP         = 00000018                 PASS_CHR                 000009C4 R     03
NAM$V_WILD_MBR         = 00000019                 PIO$GW_IIOIMPA           ********   X   03
NAM$V_WILD_NAME        = 00000005                 PIOSTRING                00000BE4 R     03
NAM$V_WILD_SFD1        = 00000019                 PRINT                    ********   X   03
```

STA(
V04-

```
PRO_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_RFA0                     = 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
```

```
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                             654    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
------------------                        --------------
_$255$DUA28:[SHRLIB]RMS.MLB;1                         32
_$255$DUA28:[SDA.OBJ]SDALIB.MLB;1                     10
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                         7
_$255$DUA28:[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
```

0353 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

PROCESS
LIS

RMS
LIS

QAST
LIS

RELEASE
LIS

POOL
LIS

0354 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

SYMBOLS
LIS

SMGRTL

SMGBLDTRM
MAP

SDAMSG
LIS

VAXINST
LIS

SMGMAPTRM
MAP

SMGKCB
SDL

VALIDATE
LIS

STACKS
LIS

SMGDEF
SDL

SMGKDE
SDL

SMGSHR
MAP