F 5

```
   SSSSSSSSSSSS  DDDDDDDDDDDD        AAAAAAAAA
   SSSSSSSSSSSS  DDDDDDDDDDDD        AAAAAAAAA
   SSSSSSSSSSSS  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
```

```
**FILE**ID**MAIN                                                    E  8
```

```
MM      MM    AAAAAA      IIIIII    NN      NN
MM      MM    AAAAAA      IIIIII    NN      NN
MMMM  MMMM  AA      AA      II      NN      NN
MMMM  MMMM  AA      AA      II      NN      NN
MM  MM  MM  AA      AA      II      NNNN    NN
MM  MM  MM  AA      AA      II      NNNN    NN
MM      MM  AA      AA      II      NN  NN  NN
MM      MM  AA      AA      II      NN  NN  NN
MM      MM  AAAAAAAAAA      II      NN    NNNN
MM      MM  AAAAAAAAAA      II      NN    NNNN
MM      MM  AA      AA      II      NN      NN      ....
MM      MM  AA      AA      II      NN      NN      ....
MM      MM  AA      AA    IIIIII    NN      NN      ....
MM      MM  AA      AA    IIIIII    NN      NN      ....

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
```

```
0000            .TITLE  MAIN    SYSTEM DUMP ANALYZER MAIN PROGRAM
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          .SBTTL  PROGRAM DESCRIPTION
0000  ;++
0000  ;  FACILITY
0000  ;
0000  ;       SYSTEM DUMP ANALYZER
0000  ;
0000  ;  ABSTRACT
0000  ;
0000  ;       THIS PROGRAM ACCEPTS A DUMP FILE FROM A SYSTEM
0000  ;       CRASH AND THE SYSTEM SYMBOL TABLE CORRESPONDING
0000  ;       TO THE SYSTEM BEING ANALYZED, AND OUTPUTS A
0000  ;       LISTING CONTAINING THE FORMATTED SYSTEM DATA
0000  ;       STRUCTURES AND MEMORY AT THE TIME OF THE CRASH.
0000  ;       THE PROGRAM CAN ALSO BE USED INTERACTIVELY TO
0000  ;       INTERROGATE THE SYSTEM DUMP INFORMATION.
0000  ;
0000  ;  ENVIRONMENT
0000  ;
0000  ;       NATIVE MODE, USER MODE
0000  ;
0000  ;  AUTHOR
0000  ;
0000  ;       TIM HALVORSEN, JULY 1978
0000  ;
0000  ;  MODIFIED BY
0000  ;
0000  ;       V03-012 EMB0104         Ellen M. Batbouta       07-Jun-1984
0000  ;               Increase the size of the LIST_BUFFER from 132 (for a
0000  ;               single line) to 300 since the line of output may overflow
0000  ;               onto the next lines.  Change the version number from v3.0
0000  ;               to v4.0 (which is displayed as part of the heading when
0000  ;               the output is sent to a file).
0000  ;
0000  ;       V03-011 EMD0094         Ellen M. Dusseault      02-May-1984
0000  ;               Save registers to preserve contents at the entrance of
0000  ;               the routine PAGE_WAIT.  The instructions which destroy
0000  ;               the registers are two movc3 instructions.
0000  ;
0000  ;       V03-010 TMK0002         Todd M. Katz            24-Ap-1984
0000  ;               Modify the routine PAGE_WAIT to save the contents of the input
0000  ;               buffer before prompting for a command. If the user simply hits
0000  ;               "RETURN" to the prompt, then the command which  as in progress
0000  ;               when end-of-page was encountered is returned to the input buffer
0000  ;               before the command is allowed to continue. This change will
0000  ;               allow any descriptors of the information in the input buffer to
0000  ;               describe the same information both before and after the
0000  ;               end-of-page was encountered.
0000  ;
0000  ;               This fixes the SHOW POOL/TYPE= problem. This command sets up a
0000  ;               descriptor of the block type requested with the buffer address
0000  ;               pointing into the input buffer. When the first end-of-page is
0000  ;               encountered during the display of block of the specified type,
0000  ;               the retrieval of the users 'RETURN', indicating that the current
0000  ;               SHOW POOL command should be continued, wipes out the block type
0000  ;               within the input buffer, that the descriptor was refering to.
0000  ;               This results in an inability to display more than a screen's
0000  ;               worth of pool whenever a block type is explicitly specified.
```

```
0000   ;
0000   ;       V03-009 JLV0329         Jake VanNoy             27-FEB-1984
0000   ;               Fix bug in ^C handling that resulted in RMS-F-BUSY errors.
0000   ;
0000   ;       V03-008 ROW0237         Ralph O. Weber          22-OCT-1983
0000   ;               Correct sub-heading output to only take character count from
0000   ;               first word of descriptor, not first longword.  Add
0000   ;               PRINT_COLUMNS a table-driven, generalized "produce displays in
0000   ;               columns" routine.
0000   ;
0000   ;       V03-007 JLV0303         Jake VanNoy             22-AUG-1983
0000   ;               Remove one argument from call to SMG$READ_COMPOSED_LINE
0000   ;               to track change to this RTL routine.
0000   ;
0000   ;       V03-006 JLV0281         Jake VanNoy             27-JUL-1983
0000   ;               Change name of init file.
0000   ;
0000   ;       V03-005 JLV0260         Jake VanNoy             23-MAY-1983
0000   ;               Add key input. Remove use of RMS for SYS$INPUT. Replace
0000   ;               use of SCR$SCREEN_INFO with a call to $GETDVI.
0000   ;
0000   ;       V03-004 TMK0001         Todd M. Katz            21-Mar-1983
0000   ;               Add the descriptor LOG_FILE, the RMS control blocks
0000   ;               LOGFAB LOGRAB and LOGNAM, and the action routines OPEN_LOG
0000   ;               and CLOSE_LOG so that interactive sessions maybe logged.
0000   ;               Also modify PUT_LINE so that all lines written to the terminal
0000   ;               are also logged to the log file when logging is enabled.
0000   ;
0000   ;       V03-003 CWH1002         CW Hobbs                13-Mar-1983
0000   ;               Reduce the prompt region at the bottom of a screen to
0000   ;               three lines so that an extra line in SHOW PROCESS can
0000   ;               be displayed without a page wrap.  Also changed a couple
0000   ;               of references to the prompt region to use the symbol
0000   ;               PROMPT_LINES rather than a constant.
0000   ;
0000   ;       V03-002 JLV0223         Jake VanNoy             21-JAN-1983
0000   ;               Add assigning a channel to terminal and establish
0000   ;               a ^C handler to exit current command.
0000   ;
0000   ;       V03-001 KTA0093         Kerbey T. Altmann       05-Apr-1982
0000   ;               Modifications to allow PAGEFILE.SYS to be a dumpfile.
0000   ;               Also use SYS$LP_LINES to calculate page size.
0000   ;
0000   ;******                                                                 ******
0000   ;               If ANALYZE was invoked via DCL;
0000   ;               Then
0000   ;                       If /SYMBOLS is present and nonblank;
0000   ;                       Then
0000   ;                               Use the value of /SYMBOLS (e.g. directory spec);
0000   ;                       Else
0000   ;                               Use a default of SYS$SYSTEM: ;
0000   ;                       Endif;
0000   ;               Else
0000   ;                       Use the directory that the dump file came from;
0000   ;               End;
0000   ;**--**                                                                 **--**
0000   ;
```

```
0000      ;                     Change all CMPW's referencing an MSG$_ symbol to CMPL's.
0000      ;
0000      ;                     Change default addressing to longword.
0000      ;
0000      ;                     Remove references to $SDAMSGDEF macro.
0000      ;
0000      ;                     Remove old Help file FAB and RAB.
0000      ;
0000      ;             V006    TMH0006         Tim Halvorsen   22-May-1981
0000      ;                     Do not show "Dump taken on ..." message if
0000      ;                     analyzing the running system.
0000      ;
0000      ;             V005    TMH0005         Tim Halvorsen   20-May-1981
0000      ;                     Add indirect FABs and RABs.  Do not request upcasing
0000      ;                     from terminal driver, as upcasing will be done by
0000      ;                     command parser now.  Change version number to 3.0.
0000      ;
0000      ;             V004    TMH0004         Tim Halvorsen   03-Feb-1981
0000      ;                     Allow program to be invoked via new ANALZYE/SYSTEM
0000      ;                     or ANALYZE/CRASH_DUMP DCL commands.
0000      ;
0000      ;             V003    TMH0003         Tim Halvorsen   23-Sep-1980
0000      ;                     Change reference to SCR$INFO to SCR$SCREEN_INFO.
0000      ;--
```

```
0000   168        .SBTTL  DECLARATIONS
0000   169 ;
0000   170 ;      SYMBOL DEFINTIONS
0000   171 ;
0000   172        $STSDEF                    ; COMPLETION CODE FIELDS
0000   173        $DSCDEF                    ; DESCRIPTOR DEFINITIONS
0000   174        $DVIDEF                    ; GETDVI DEFINITIONS
0000   175        $COLMDEF                   ; COLUMN LIST definitions
0000   176        $EMBDEF <CR>               ; ERROR LOG DEFINITIONS
0000   177        $DMPDEF                    ; DUMP FILE DEFINITIONS
0000   178        $DIBDEF                    ; DEVICE INFORMATION BUFFER
0000   179        $DCDEF                     ; DEVICE TYPE DEFINITIONS
0000   180        $DEVDEF                    ; DEVICE CHARACTERISTICS
0000   181        $JPIDEF                    ; GETJPI REQUEST DEFINITIONS
0000   182        $SHRDEF                    ; SHARED MESSAGE DEFINITIONS
0000   183        $SCRDEF                    ; SCREEN PACKAGE DEFINITIONS
0000   184        $CHFDEF                    ; CONDITION HANDLING FACILITY DEFINITIONS
0000   185        $CLIDEF                    ; OLD CLI INTERFACE DEFINITIONS
0000   186
0000   187        .DEFAULT DISPLACEMENT,LONG
```

```
              0000                .SBTTL  STORAGE DEFINITIONS
              0000
              0000       ;
              0000       ;        READ/WRITE STORAGE DEFINITIONS
              0000       ;
              0000
          00000000                .PSECT  SDADATA,NOEXE,WRT,LONG
              0000
              0000       VERSION_FLAGS::
00000004      0000                .BLKL   1                        ; DESCRIBES SYSTEM VERSION
              0004
              0004       LOG_FILE::
0000000C      0004                .BLKQ   1                        ; LOG FILE NAME DESCRIPTOR
              000C
              000C       OUTPUT_FILE::
00000014      000C                .BLKQ   1                        ; OUTPUT FILE NAME DESCRIPTOR
              0014
              0014       CURRENT_SYSTEM::
00000018      0014                .BLKL   1                        ; TRUE IF EXAMINING CURRENT SYSTEM
              0018
              0018       PAGE_NUMBER::
0000001C      0018                .BLKL   1                        ; CURRENT PAGE NUMBER
              001C
              001C       LINE_COUNT::
00000000      001C                .LONG   0                        ; LINES FOR CURRENT PAGE
              0020
              0020       HEADING_LINES:
00000024      0020                .BLKL   1                        ; LINES USED FOR PAGE HEADING
              0024
00000003      0024       PROMPT_LINES = 3                          ; NO. LINES USED FOR PAGE_WAIT
              0024
              0024       PAGE_SIZE::
00000028      0024                .BLKL   1                        ; MAX. LINES/PAGE
              0028
              0028       CLR_PAGE:
0000002C      0028                .BLKL   1                        ; zeroed to prevent clearing page
              002C
              002C       CURRENT_TIME:                             ; CURRENT DATE AND TIME
00000018      002C                .LONG   24
00000034'     0030                .LONG   TIME_BUFFER
              0034       TIME_BUFFER:
0000004C      0034                .BLKB   24                       ; ONLY GET FIRST 24 CHARS
              004C
              004C       CTRLC_PENDING:
0000004D      004C                .BLKB   1                        ; ^C pending flag
              004D
              004D       PUT_BUSY:
0000004E      004D                .BLKB   1                        ; $PUT busy flag
              004E
          00000000                .PSECT  BUFFERS,NOEXE,WRT
              0000
              0000       BUFFER::
00000200      0000                .BLKB   512                      ; BUFFER FOR GENERAL USE
              0200
00000050      0200       INPUT_BUF_LEN == 80
              0200       INPUT_BUFFER::
00000250      0200                .BLKB   INPUT_BUF_LEN
```

```
                              0250  SAVE_INPUT_BUFFER:
                    000002A0  0250          .BLKB   INPUT_BUF_LEN
                              02A0  SAVE_INPUT_LEN:
                    00000000  02A0          .LONG   0
                              02A4  INPUT_LEN::
                    00000000  02A4          .LONG   0
                              02A8  INPUT_BUF::
                    00000050  02A8          .LONG           INPUT_BUF_LEN   ; Descriptor for input buffer
                   00000200'  02AC          .ADDRESS        INPUT_BUFFER
                              02B0
                              02B0  DUMP_HEADER::
                    00000600  02B0  DUMP_HEADER_LEN == 3*512            ; 3 BLOCKS
                    000008B0  02B0          .BLKB   DUMP_HEADER_LEN
                              08B0
                              08B0  LINE_DESCR::
                   0000012C'  08B0          .LONG   LIST_BUFFER_LEN
                   000008B8'  08B4          .LONG   LIST_BUFFER
                              08B8  LIST_BUFFER:
                    0000012C  08B8  LIST_BUFFER_LEN = 300
                    000009E4  08B8          .BLKB   LIST_BUFFER_LEN
                              09E4
                              09E4  CMND_DESCR::                        ; OUTPUT BUFFER DESCRIPTOR
                   00000050'  09E4          .LONG   CMND_BUFFER_LEN
                   000009EC'  09E8          .LONG   CMND_BUFFER
                              09EC  CMND_BUFFER::
                    00000050  09EC  CMND_BUFFER_LEN == 80
                    00000A3C  09EC          .BLKB   CMND_BUFFER_LEN
                              0A3C
                    00000000  0A3C  STB_BUFFER==BUFFER                  ; OVERLAP MISC. BUFFER
                    00000200  0A3C  STB_BUFFER_LEN = 512
                              0A3C
                              0A3C  HELP_BUFFER:
                    00000050  0A3C  HELP_BUFFER_LEN = 80
                    00000A8C  0A3C          .BLKB   HELP_BUFFER_LEN
                              0A8C
                              0A8C  REPEAT_KEY::
30 50 4B 00000A94'010E0000'  0A8C          .ASCID  /KP0/               ; Default 'repeat' key
                              0A97  KEYBOARD::
                    00000000  0A97          .LONG   0                   ; Screen input routine storage
                              0A9B  KEYTABLE::
                    00000000  0A9B          .LONG   0
                              0A9F  TT_CHAN::
                    00000000  0A9F          .LONG   0                   ; TERMINAL CHANNEL IF SYS$INPUT IS TRM
                              0AA3  SMG_PROMPT:
                    00000000  0AA3          .LONG   0                   ; address of prompt loaded here
                              0AA7
                              0AA7  DVI_ITMLST:
                   000A 0004  0AA7          .WORD   4,DVI$_DEVDEPEND
          00000000 00000ADB'  0AAB          .LONG   DVI_DEVDEPEND,0     ; GETDVI FOR DEVDEPEND
                              0AB3
                   001C 0004  0AB3          .WORD   4,DVI$_DEVDEPEND2
          00000000 00000ADF'  0AB7          .LONG   DVI_DEVDEPND2,0     ; GETDVI FOR DEVDEPEND 2
                              0ABF
                   0008 0004  0ABF          .WORD   4,DVI$_DEVBUFSIZ
          00000000 00000AE3'  0AC3          .LONG   DVI_DEVBUFSIZ,0
                              0ACB
                   0002 0004  0ACB          .WORD   4,DVI$_DEVCHAR
```

```
00000000 00000AE7'  0ACF            .LONG   DVI_DEVCHAR,0
                    0AD7
          00000000  0AD7            .LONG   0                       ; End of list
                    0ADB
                    0ADB  DVI_DEVDEPEND::
          00000ADE  0ADB            .BLKB   3
                    0ADE  DVI_PAGESIZE:
                00  0ADE            .BYTE   0                       ; High byte of DEVDEPEND is page size
                    0ADF  DVI_DEVDEPND2::
          00000000  0ADF            .LONG   0
                    0AE3  DVI_DEVBUFSIZ:
          00000000  0AE3            .LONG   0
                    0AE7  DVI_DEVCHAR::
          00000000  0AE7            .LONG   0
                    0AEB
                00000000            .PSECT  RMSBLOCKS,NOEXE,WRT,LONG
                    0000
                    0000  DUMPF:: $FAB      DNM=<SYSDUMP.DMP>,-
                    0000                    NAM=DUMPN,-             ; ADDRESS OF NAM BLOCK
                    0000                    FAC=<BIO,GET>           ; BLOCK I/O
                    0050
                    0050  DUMPN:  $NAM      ESA=DUMP_EXPNAME,-      ; EXPANDED NAME STRING BUFFER
                    0050                    ESS=NAM$C_MAXRSS        ; LENGTH OF BUFFER
                    00B0
                    00B0  DUMP_EXPNAME:
          000001AF  00B0            .BLKB   NAM$C_MAXRSS            ; EXPANDED NAME STRING
                    01AF
                    01AF  DUMPR:: $RAB      FAB=DUMPF, -
                    01AF                    ROP=BIO, -              ; BLOCK I/O ACCESS
                    01AF                    BKT=1, -                ; BOZO'S BUCKET #1
                    01AF                    UBF=DUMP_HEADER, -      ; BUFFER ADDRESS
                    01AF                    USZ=DUMP_HEADER_LEN     ; BUFFER LENGTH
                    01F3
                    01F3  SAVDMPF: $FAB     DNM=<.DMP>,-            ; DEFAULT NAME STRING
                    01F3                    FOP=SUP,-               ; SUPERSEDE OLD VERSION ON CREATE
                    01F3                    FAC=<BIO,PUT>,-         ; BLOCK I/O
                    01F3                    RFM=FIX,-               ; RECORD FORMAT IS FIXED
                    01F3                    MRS=512                 ; 512 BYTE RECORDS
                    0243
                    0243  SAVDMP:: $RAB     FAB=SAVDMPF,-           ; ADDRESS OF FAB BLOCK
                    0243                    ROP=BIO                 ; BLOCK I/O ACCESS
                    0287
                    0287  LISTF:  $FAB      FAC=<PUT,UPD>, -        ; PUT/UPDATE
                    0287                    DNM=<SYSDUMP.LIS>,-     ; DEFAULT NAME STRING
                    0287                    MRS=133, -              ; MAXIMUM RECORD SIZE
                    0287                    ORG=SEQ, -              ; SEQUENTIAL ORGANIZATION
                    0287                    RAT=CR, -               ; CR CARRIAGE CONTROL
                    0287                    RFM=VAR,-               ; FIXED LENGTH RECORDS
                    0287                    NAM=LISTN               ; ADDRESS OF NAM BLOCK
                    02D7
                    02D7  LIST::  $RAB      FAB=LISTF, -
                    02D7                    MBF=2, -                ; DOUBLE BUFFERED
                    02D7                    MBC=16, -               ; 16 BLOCKS AT A TIME
                    02D7                    RAC=SEQ, -              ; SEQUENTIAL ACCESS
                    02D7                    RBF=LIST_BUFFER, -      ; BUFFER ADDRESS
                    02D7                    RSZ=0, -                ; EMPTY BUFFER
                    02D7                    UBF=LIST_BUFFER, -      ; DUMMY READ BUFFER
```

```
    02D7                         USZ=LIST_BUFFER_LEN, -   ; BUFFER LENGTH
    02D7                         ROP=WBH                  ; WRITE BEHIND (DOUBLE BUFFER)
    031B
    031B  LISTN:    $NAM         ESS=NAM$C_MAXRSS         ; MAXIMUM EXPANDED SIZE
    037B
    037B  LOGFAB::$FAB           FAC=PUT, -               ; PUT OPERATIONS
    037B                         DNM=<SYSDUMP.LOG>,-      ; DEFAULT NAME STRING
    037B                         MRS=133, -               ; MAXIMUM RECORD SIZE
    037B                         ORG=SEQ, -               ; SEQUENTIAL ORGANIZATION
    037B                         RAT=CR, -                ; CR CARRIAGE CONTROL
    037B                         RFM=VAR,-                ; FIXED LENGTH RECORDS
    037B                         NAM=LOGNAM               ; ADDRESS OF NAM BLOCK
    03CB
    03CB  LOGRAB::$RAB           FAB=LOGFAB, -
    03CB                         MBF=2, -                 ; DOUBLE BUFFERED
    03CB                         MBC=16, -                ; 16 BLOCKS AT A TIME
    03CB                         RAC=SEQ, -               ; SEQUENTIAL ACCESS
    03CB                         ROP=WBH                  ; WRITE BEHIND (DOUBLE BUFFER)
    040F
    040F  LOGNAM: $NAM           ESS=NAM$C_MAXRSS         ; MAXIMUM EXPANDED SIZE
    046F
    046F  INDFAB: $FAB           FAC=GET,-                ; READ OPERATIONS
    046F                         DNM=<.COM>               ; DEFAULT NAME STRING
    04BF
    04BF  INDRAB:: $RAB          FAB=INDFAB,-             ; ADDRESS OF FAB
    04BF                         UBF=INPUT_BUFFER,-       ; ADDRESS OF INPUT BUFFER
    04BF                         USZ=INPUT_BUF_LEN        ; BUFFER LENGTH
    0503
    0503  KEYFAB: $FAB           FAC=GET,-                ; READ OPERATIONS
    0503                         FNM=<SYS$LOGIN:SDA.INIT>,-; FILE NAME
    0503                         DNM=<SDA$INIT>           ; DEFAULT NAME STRING
    0553
    0553  KEYRAB:: $RAB          FAB=KEYFAB,-             ; ADDRESS OF FAB
    0553                         UBF=INPUT_BUFFER,-       ; ADDRESS OF INPUT BUFFER
    0553                         USZ=INPUT_BUF_LEN        ; BUFFER LENGTH
    0597
    0597  OUTPUTF: $FAB          FNM=<SYS$OUTPUT>, -
    0597                         RAT=CR, -                ; EACH LINE NEEDS LF/CR
    0597                         FAC=PUT                  ; PUT OPERATIONS ONLY
    05E7
    05E7  OUTPUT:: $RAB          FAB=OUTPUTF, -
    05E7                         UBF=CMND_BUFFER, -       ; OUTPUT BUFFER
    05E7                         USZ=CMND_BUFFER_LEN      ; OUTPUT BUFFER LENGTH
    062B
    062B  STBF::    $FAB         FNM=<SYS.STB;0>, -
    062B                         FAC=GET                  ; GET OPERATIONS ONLY
    067B
    067B  STB::     $RAB         FAB=STBF, -
    067B                         UBF=STB_BUFFER, -        ; BUFFER ADDRESS
    067B                         USZ=STB_BUFFER_LEN       ; BUFFER LENGTH
    06BF
00000000            .PSECT       MAIN,EXE,NOWRT,LONG
```

```
                                         0000     413         .SBTTL  READ-ONLY DATA DEFINITIONS
                                         0000     414
                                         0000     415 ;
                                         0000     416 ;       READ-ONLY DATA DEFINITIONS
                                         0000     417 ;
                                         0000     418
                                         0000     419 SYSTEM_ENTITY:
4D 45 54 53 59 53 00000008'010E0000'     0000     420         .ASCID  'SYSTEM'
                                         000E     421 CRASH_ENTITY:
5F 48 53 41 52 43 00000016'010E0000'     000E     422         .ASCID  'CRASH_DUMP'
                         50 4D 55 44     001C
                                         0020     423 DUMPFILE_ENTITY:
46 5F 50 4D 55 44 00000028'010E0000'     0020     424         .ASCID  'DUMP_FILE'
                            45 4C 49     002E
                                         0031     425 SYMBOLS_ENTITY:
4C 4F 42 4D 59 53 00000039'010E0000'     0031     426         .ASCID  'SYMBOLS'
                                  53     003F
                                         0040     427
                                         0040     428 DEV_PROMPT:
72 65 74 6E 45 0A 00000048'010E0000'     0040     429         .ASCID  <10>'Enter name of dump file > '
6D 75 64 20 66 6F 20 65 6D 61 6E 20      004E
         20 3E 20 65 6C 69 66 20 70      005A
                                         0063     430
                                         0063     431 SDA_PROMPT:
20 3E 41 44 53 0A 0000006B'010E0000'     0063     432         .ASCID  <10>'SDA> '
                                         0071     433
                                         0071     434 SYS$SYSTEM:
59 53 24 53 59 53 00000079'010E0000'     0071     435         .ASCID  'SYS$SYSTEM:'
                      3A 4D 45 54 53     007F
                                         0084     436
                                         0084     437 STARTUP:
                50 55 54 52 41 54 53     0084     438         .ASCII  'STARTUP'                ; NAME OF STARTUP PROCESS
                            00000007     008B     439 STARTUP_LEN = . - STARTUP
                                         008B     440
                                         008B     441 SYSINPUT:
4E 49 24 53 59 53 00000093'010E0000'     008B     442         .ASCID  /SYS$INPUT/              ;
                            54 55 50     0099
```

```
                              009C               .SBTTL  MAIN PROGRAM
                              009C      ;---
                              009C      ;
                              009C      ;        START - MAIN PROGRAM ENTRY POINT
                              009C      ;
                              009C      ;        CALL INITIALIZATION ROUTINES AND FORMAT THE SYSTEM DUMP
                              009C      ;        BY CALLING THE INDIVIDUAL FORMATTING ROUTINES ONE AT A
                              009C      ;        TIME.
                              009C      ;---
                              009C
                              009C               .ENABL  LSB
                              009C
                        0204  009C               .ENTRY  START,^M<R2,R9>
                              009E
  6D   00000000'EF        9E  009E               MOVAB   HANDLER,(FP)               ; INITIALIZE CONDITION HANDLER
          01E4'CF    00   FB  00A5               CALLS   #0,W^OPEN_FILES            ; OPEN INPUT/OUTPUT FILES
     00000000'EF     00   FB  00AA               CALLS   #0,MAP_DUMP                ; MAP DUMP INTO VIRTUAL MEMORY
     00000000'EF     00   FB  00B1               CALLS   #0,READ_SYMBOLS            ; READ SYSTEM SYMBOL TABLE
     00000000'EF     00   FB  00B8               CALLS   #0,GET_DUMP_INFO           ; GET DUMP FILE INFORMATION
          05B4'CF    00   FB  00BF               CALLS   #0,W^EXIT_IF_OLD           ; IF OLD DUMP AT STARTUP TIME
                              00C4
                              00C4               SKIP    PAGE                       ; ERASE SCREEN
     0F 00000014'EF       E9  00CB               BLBC    CURRENT_SYSTEM,3$          ; BRANCH IF ANALYZING A DUMP
                              00D2               PRINT   0,<VAX/VMS System analyzer>
                              00DF      ;        PUSHL   #0
                              00DF      ;        PRINT   1,<!17%D>
                     70   11  00DF               BRB     8$
                              00E1
                              00E1      3$:      PRINT   0,<VAX/VMS System dump analyzer>
                              00EE               PRINT   0,< >
  59   00000000'EF        D0  00FB               MOVL    ERLPTR,R9                  ; ADDRESS OF ERROR LOG ENTRY
                              0102               $ASCTIM_S TIMADR=EMB$Q_CR_TIME(R9),TIMBUF=CURRENT_TIME
       0000002C'EF        7F  0116               PUSHAQ  CURRENT_TIME
                              011C               PRINT   1,<Dump taken on !AS>
50   00F4 C9    FD 8F     78  0129               ASHL    #-3,EMB$L_CR_CODE(R9),R0          ; MESSAGE NUMBER
                     1F   13  0130               BEQL    8$                         ; BRANCH IF NO MESSAGE
  51   00000000'EF        9E  0132               MOVAB   BUG$T_MESSAGES,R1          ; ADDRESS OF MESSAGES
                              0139      5$:
                52   81   9A  0139               MOVZBL  (R1)+,R2                   ; LENGTH OF MESSAGE
                51   52   C0  013C               ADDL2   R2,R1                      ; SKIP TO NEXT MESSAGE
                  F7 50   F5  013F               SOBGTR  R0,5$                      ; LOOP UNTIL FOUND
                     51   DD  0142               PUSHL   R1                         ; ADDRESS OF BUGCHECK MESSAGE
                              0144               PRINT   1,<!AC>
                              0151      8$:
                              0151               PRINT   0,<>                       ; BLANK LINE
                              015E      ;
                              015E      ;        SET CURRENT PROCESS = PROCESS THAT CRASHED
                              015E      ;
     00000000'EF     00   FB  015E               CALLS   #0,CURPROC                 ; SET TO CURRENT PROCESS
                              0165      ;
                              0165      ;        PROCESS SDAINI FILE
                              0165      ;
       00000000'EF        D5  0165               TSTL    INPUT_RAB                  ; SEE IF SDAINI FOUND
                     16   13  016B               BEQL    20$                        ; NOT IF ZERO
                              016D      10$:
       0000001C'EF        D4  016D               CLRL    LINE_COUNT                 ; AVOID END OF PAGE PROMPTS
     00000000'EF     00   FB  0173               CALLS   #0,GET_COMMANDS            ; ACCEPT AND EXECUTE COMMANDS
                  F0 50   E8  017A               BLBS    R0,10$                     ; CONTINUE UNTIL ERROR
```

```
        00000000'EF       D4  017D    501          CLRL    INPUT_RAB               ; CLEAR SDAINI RAB
                              0183    502 ;
                              0183    503 ;        ACCEPT COMMANDS UNTIL END OF FILE
                              0183    504 ;
                              0183    505 20$:
   00000028'EF    01      D0  0183    506          MOVL    #1,CLR_PAGE             ; CLEAR PAGE ON MSG$_EXITCMD
        0000001C'EF       D4  018A    507          CLRL    LINE_COUNT              ; AVOID END OF PAGE PROMPTS
   00000000'EF    00      FB  0190    508          CALLS   #0,GET_COMMANDS         ; ACCEPT AND EXECUTE COMMANDS
   00000000'8F    50      D1  0197    509          CMPL    R0,#MSG$_EOF            ; CHECK IF END OF FILE
                  26      13  019E    510          BEQL    50$                     ; BRANCH IF SO
   00000000'8F    50      D1  01A0    511          CMPL    R0,#MSG$_EXITCMD        ; DID WE JUST EXIT A LEVEL?
                  DA      12  01A7    512          BNEQ    20$                     ; BRANCH IF NOT
        00000000'EF       7C  01A9    513          CLRQ    SUB_HEADING             ; CLEAR CURRENT HEADING
        00000000'EF       D4  01AF    514          CLRL    HEADING_ROUTINE         ; CLEAR HEADING ROUTINE ADDRESS
        00000028'EF       D5  01B5    515          TSTL    CLR_PAGE                ; IF ZERO, DON'T CLEAR PAGE
                  C6      13  01BB    516          BEQL    20$                     ;
                              01BD    517          SKIP    PAGE                    ; ERASE PREVIOUS JUNK
                  BD      11  01C4    518          BRB     20$
                              01C6    519 50$:
        0000000C'EF       D5  01C6    520          TSTL    OUTPUT_FILE             ; OUTPUT FILE SPECIFIED?
                  07      13  01CC    521          BEQL    90$                     ; BRANCH IF NOT
   00000000'EF    00      FB  01CE    522          CALLS   #0,PRINT_INDEX          ; PRINT TABLE OF CONTENTS
                              01D5    523 90$:
   00000000'EF    00      FB  01D5    524          CALLS   #0,MARK_DUMP            ; MARK DUMP ANALYZED
                              01DC    525          STATUS  SUCCESS
                          04  01E3    526          RET
                              01E4    527
                              01E4    528          .DSABL  LSB
```

```
                                       01E4          .SBTTL  OPEN_FILES - OPEN INPUT/OUTPUT FILES
                                       01E4     ;---
                                       01E4     ;       OPEN_FILES
                                       01E4     ;
                                       01E4     ;       THE FOLLOWING FILES WILL BE OPENED:
                                       01E4     ;
                                       01E4     ;        - SYSTEM DUMP FILE (SYSDUMP.DMP)
                                       01E4     ;        - SYSTEM SYMBOL TABLE (SYS.STB)
                                       01E4     ;        - COMMAND FILE FOR RUN-TIME OPTIONS
                                       01E4     ;
                                       01E4     ;---
                                       01E4
                                007C   01E4          .ENTRY  OPEN_FILES,^M<R2,R3,R4,R5,R6>
                                       01E6          ;
                                       01E6          ; Use screen management routines for input.
                                       01E6          ; RMS will be used by the RTL in SYS$INPUT is a file.
                                       01E6          ;
                 FEA1 CF         9F    01E6          pushab  SYSINPUT                    ; SYS$INPUT
             00000A97'EF         9F    01EA          pushab  keyboard                    ; keyboard id
        00000000'GF    02        FB    01F0          calls   #2,G^SMG$CREATE_VIRTUAL_KEYBOARD ; create (open) keyboard
                                       01F7          SIGNAL
                                       0203
             00000A9B'EF         9F    0203          pushab  keytable                    ; keypad table
        00000000'GF    01        FB    0209          calls   #1,G^SMG$CREATE_KEY_TABLE ; create keypad table
                                       0210          SIGNAL
                                       021C          ;
                                       021C          ; Try to open a file defined by the logical name ??? SDA$KEYPAD ???
                                       021C          ;
                                       021C          $OPEN   KEYFAB                      ; try open
                    1B 50        E9    0229          BLBC    R0,10$                      ; continue if error
                                       022C          $CONNECT KEYRAB                     ; try connect
                    0B 50        E9    0239          BLBC    R0,10$                      ; continue if error
  00000000'EF   00000553'EF      9E    023C          MOVAB   KEYRAB,INPUT_RAB            ; make it look like this
                                       0247                                              ; was an "@filespec"
                                       0247     10$:
                                       0247          $CREATE OUTPUTF                     ; OPEN OUTPUT FILE
                                       0254          SIGNAL  RMS,OUTPUTF
                                       026A          $CONNECT OUTPUT
                                       0277          SIGNAL  RMS,OUTPUT
                                       028D
                                       028D          $GETDVI_S DEVNAM = SYSINPUT,-
                                       028D                  ITMLST = DVI_ITMLST         ; GET DEVICE DEPENDENT INFO
                                       02A9          SIGNAL
                                       02B5
        50    00000ADE'EF        9A    02B5          MOVZBL  DVI_PAGESIZE,R0             ; GET PAGE SIZE
     00000024'EF    50   03      C3    02BC          SUBL3   #PROMPT_LINES,R0,PAGE_SIZE      ; SET PAGE SIZE
                                       02C4
      06 00000AE7'EF   02        E0    02C4          BBS     #DEV$V_TRM,DVI_DEVCHAR,40$
             00000ADB'EF         D4    02CC          CLRL    DVI_DEVDEPEND               ; Clear if not terminal
                                       02D2     40$:
                                       02D2     ;
                                       02D2     ;       If the command line entities SYSTEM or CRASH_DUMP are defined
                                       02D2     ;       and "present", initialize as "current system" or dump file
                                       02D2     ;       respectively.
                                       02D2     ;
                       56        D4    02D2          clrl    r6                          ;set DCL flag = .FALSE.
                5E     1C        C2    02D4          subl    #cli$c_reqdesc,sp           ; Allocate old CLINT request block
```

```
6E   1C   00   6E   00   2C  02D7            movc5   #0,(sp),#0,#cli$c_reqdesc,(sp) ; Zero request block
      53   00000000'EF   9E  02DD            movab   dumpf,r3                 ; Set address of DUMP FAB
                6E   01   90  02E4            movb    #cli$k_getcmd,cli$b_rqtype(sp) ; Set "get command" request
                     5E   DD  02E7            pushl   sp                       ; Push address of request descriptor
      00000000'GF    01   FB  02E9            calls   #1,g^sys$cli             ; Call old CLI interface for verb type
                72   50   E9  02F0            blbc    r0,80$                   ; Branch if not a DCL command
      00'8F    03    AE   91  02F3            cmpb    cli$b_rqstat(sp),#cli$k_verb_fore ; Foreign command?
                     6B   13  02F8            beql    80$                      ; Branch if so
      00'8F    03    AE   91  02FA            cmpb    cli$b_rqstat(sp),#cli$k_verb_mcr  ; or MCR command?
                     64   13  02FF            beql    80$                      ; Branch if so
                              0301
             FCFB    CF   9F  0301            pushab  system_entity            ; address of entity descriptor
      00000000'GF    01   FB  0305            calls   #1,g^cli$present         ; Check if /SYSTEM specified
                54   50   E8  030C            blbs    r0,60$                   ; branch if present
             FCFB    CF   9F  030F            pushab  crash_entity             ; address of entity descriptor
      00000000'GF    01   FB  0313            calls   #1,g^cli$present         ; Check if /CRASH_DUMP specified
                48   50   E9  031A            blbc    r0,80$                   ; branch if absent
             FCFF    CF   9F  031D            pushab  dumpfile_entity          ; address of entity descriptor
      00000000'GF    01   FB  0321            calls   #1,g^cli$present         ; Check if dump filespec present
                3A   50   E9  0328            blbc    r0,80$                   ; Branch if absent
                     7E   7C  032B            clrq    -(sp)                    ; Recieve buffer descriptor
           03 AE     02   90  032D            movb    #dsc$k_class_d,dsc$b_class(sp) ; Set to dynamic desc.
                     5E   DD  0331            pushl   sp                       ; address of return buffer
             FCE9    CF   9F  0333            pushab  dumpfile_entity          ; address of entity descriptor
      00000000'GF    02   FB  0337            calls   #2,g^cli$get_value       ; Get dump filespec from command line
                24   50   E9  033E            blbc    r0,80$                   ; Branch if absent
           34 A3     6E   90  0341            movb    (sp),fab$b_fns(r3)       ; set length of filespec
      2C A3    04    AE   D0  0345            movl    4(sp),fab$l_fna(r3)      ; set address of filespec
                     7E   7C  034A            clrq    -(sp)                    ; Recieve buffer descriptor
           03 AE     02   90  034C            movb    #dsc$k_class_d,dsc$b_class(sp) ; Set to dynamic desc.
                     5E   DD  0350            pushl   sp                       ; address of return buffer
             FCDB    CF   9F  0352            pushab  symbols_entity           ;address of symbols descriptor
      00000000'GF    02   FB  0356            calls   #2,g^cli$get_value       ;Check if /SYMBOLS specified and
                              035D                                             ;   get its value if it was.
                56   01   88  035D            bisb2   #1,r6                    ;set DCL flag = .TRUE.
                   007D   31  0360            brw     200$                     ; and open dump/stb files
                              0363
                     50   11  0363  60$:      brb     140$                     ; analyze current running system
                              0365  ;
                              0365  ;         Attempt to get file name from foreign command line
                              0365  ;
                              0365  80$:
  00000AA3'EF    FCD7  CF   9E  0365            movab   dev_prompt,smg_prompt    ; Prompt descriptor
                              036E
                     7E   7C  036E            clrq    -(sp)                    ; Recieve buffer descriptor
           03 AE     02   90  0370            movb    #dsc$k_class_d,dsc$b_class(sp) ; Set to dynamic desc.
             FCC8    CF   9F  0374            pushab  dev_prompt               ; Address of prompt desc.
                04   AE   9F  0378            pushab  4(sp)                    ; Address of buffer desc.
      00000000'GF    02   FB  037B            calls   #2,G^lib$get_foreign     ; Get the command line
                41   50   E9  0382            blbc    r0,160$                  ; branch if any error
           34 A3     6E   90  0385            movb    (sp),fab$b_fns(r3)       ; set length of filespec
      2C A3    04    AE   D0  0389            movl    4(sp),fab$l_fna(r3)      ; set address of filespec
                     13   11  038E            brb     120$                     ; Process command line
                              0390  100$:
                              0390            ;
                              0390            ; Loop to here to try another dump file prompt
                              0390            ;
```

```
              019D  30 0390   644           bsbw    get_input                 ; Get one line of input
 34 A3   000002A4'EF  90 0393   645           movb    input_len,fab$b_fns(r3)  ; set length of filespec
 2C A3   00000200'EF  9E 0398   646           movab   input_buffer,fab$l_fna(r3); set address of filespec
                        03A3   647 120$:
       00000014'EF  D4 03A3   648           CLRL    CURRENT_SYSTEM            ; PRESET TO NON-CURRENT SYS.
        01    34 A3  91 03A9   649           CMPB    fab$b_fns(R3),#1          ; MUST BE EXACTLY 1 CHAR.
                17  12 03AD   650           BNEQ    160$                      ; BRANCH IF NOT
        2A    2C B3  91 03AF   651           CMPB    @fab$l_fna(R3),#^A'*'     ; SEE CURRENT RUNNING SYSTEM?
                11  12 03B3   652           BNEQ    160$                      ; BRANCH IF NOT
   00000014'EF   01  D0 03B5   653 140$:    MOVL    #1,CURRENT_SYSTEM         ; EXAMINE CURRENT SYSTEM
   54    0000062B'EF  9E 03BC   654           MOVAB   STBF,R4                   ; SETUP R4 FOR STB CODE
              00C1  31 03C3   655           BRW     340$                      ; OPEN STB FILE
                        03C6   656 160$:
   00000000'8F   50  D1 03C6   657           CMPL    R0,#RMS$_EOF              ; CHECK IF END OF FILE
                05  12 03CD   658           BNEQ    180$                      ; BRANCH IF NOT
  50   03   00   04  F0 03CF   659           INSV    #STS$K_SEVERE, -          ; MUST EXIT IMAGE IF EOF
                        03D4   660                   #STS$V_SEVERITY,#STS$S_SEVERITY,R0
                        03D4   661 180$:
                        03D4   662           SIGNAL
                        03E0   663 200$:
   53    00000000'EF  9E 03E0   664           MOVAB   DUMPF,R3                  ; ADDRESS OF FAB
        55    28 A3  D0 03E7   665           MOVL    FAB$L_NAM(R3),R5          ; ADDRESS OF NAM BLOCK
             34 A3  95 03EB   666           TSTB    FAB$B_FNS(R3)             ; TEST SIZE
                0C  12 03EE   667           BNEQ    220$                      ; BRANCH IF NON-EMPTY STRING
    2C A3    FC81 CF  D0 03F0   668           MOVL    SYS$SYSTEM+4,FAB$L_FNA(R3)      ; GET FROM SYS$SYSTEM
    34 A3    FC77 CF  90 03F6   669           MOVB    SYS$SYSTEM,FAB$B_FNS(R3)
                        03FC   670 220$:
                        03FC   671           $OPEN   (R3)                      ; ATTEMPT TO OPEN THE FILE
             06 50  E8 0405   672           BLBS    R0,240$
        51    0C A3  D0 0408   673           MOVL    FAB$L_STV(R3),R1          ; SECONDARY ERROR CODE
                17  11 040C   674           BRB     260$                      ; AND OUTPUT ERROR MESSAGE
                        040E   675 240$:
                        040E   676           $CONNECT DUMPR
             29 50  E8 041B   677           BLBS    R0,300$                   ; BRANCH IF SUCCESSFUL
   51    000001BB'EF  D0 041E   678           MOVL    DUMPR+RAB$L_STV,R1
                        0425   679 260$:
             0C A5  DD 0425   680           PUSHL   NAM$L_ESA(R5)             ; DESCRIPTOR OF FILE NAME
        7E    0B A5  9A 0428   681           MOVZBL  NAM$B_ESL(R5),-(SP)
          7E    50  7D 042C   682           MOVQ    R0,-(SP)                  ; PUSH RMS ERROR CODES
             08 AE  9F 042F   683           PUSHAB  8(SP)                     ; ADDRESS OF DESCRIPTOR
                01  DD 0432   684           PUSHL   #1                        ; NUMBER OF FAO ARGUMENTS
       00000000'8F  DD 0434   685           PUSHL   #MSG$_OPENIN              ; ERROR OPENING INPUT FILE
                        043A   686 ;***     TSTB    RAB$B_PSZ(R2)             ; ARE WE PROMPTING FOR FILESPEC?
                        043A   687 ;***     BEQL    280$                      ; IF NOT, EXIT PROGRAM ON ANY ERROR
                        043A   688 ;***     INSV    #STS$K_WARNING, -         ; CHANGE SEVERITY TO WARNING
                        043A   689 ;***             #STS$V_SEVERITY,#STS$S_SEVERITY,(SP)
                        043A   690
   00000000'GF   05  FB 043A   691 280$:    CALLS   #5,G^LIB$SIGNAL           ; OUTPUT ERROR MESSAGE
            5E   08  C0 0441   692           ADDL    #8,SP                     ; REMOVE DESCRIPTOR FROM STACK
              FF49  31 0444   693           BRW     100$                      ; ALLOW USER TO TRY AGAIN
                        0447   694
                        0447   695 300$:
   54    0000062B'EF  9E 0447   696           MOVAB   STBF,R4
             1A 56  E9 044E   697           BLBC    R6,320$                          ;INVOKED VIA DCL?
                6E  B5 0451   698           TSTW    (SP)                             ;WAS /SYMBOLS NONBLANK?
                32  13 0453   699           BEQL    340$                             ;NO, SO USE SYS$SYSTEM:
    35 A4    34 A4  90 0455   700           MOVB    FAB$B_FNS(R4),FAB$B_DNS(R4) ;MAKE "SYS.STB;0" THE DEFAULT FILE
```

```
        30 A4    2C A4   D0  045A   701           MOVL    FAB$L_FNA(R4),FAB$L_DNA(R4)
        2C A4    04 AE   D0  045F   702           MOVL    4(SP),FAB$L_FNA(R4)      ;USER SUPPLIED DIRECTORY SPEC
           34 A4    6E   90  0464   703           MOVB    (SP),FAB$B_FNS(R4)       ;  TAKEN FROM THE /SYMBOLS QUAL.
                  0028   31  0468   704           BRW     360$                     ;ATTEMPT TO OPEN THE FILE
                             046B   705 320$:
        30 A4    0C A5   D0  046B   706           MOVL    NAM$L_ESA(R5),FAB$L_DNA(R4)  ; SET DEFAULT FROM DUMP
        35 A4    0B A5   90  0470   707           MOVB    NAM$B_ESL(R5),FAB$B_DNS(R4)
                             0475   708           $OPEN   (R4)                     ; OPEN THE STB FILE
     00000000'8F      50 D1  047E   709           CMPL    R0,#RMS$_FNF             ; CHECK IF STB FILE THERE
                      15 12  0485   710           BNEQ    380$                     ; BRANCH IF OK
       30 A4   FBEA CF   D0  0487   711 340$:     MOVL    SYS$SYSTEM+4,FAB$L_DNA(R4)      ; SET TO TRY SYS$SYSTEM
       35 A4   FBE0 CF   90  048D   712           MOVB    SYS$SYSTEM,FAB$B_DNS(R4)
                             0493   713
                             0493   714 360$:     $OPEN   (R4)                     ; OPEN THE STB FILE
                 21 50   E8  049C   715 380$:     BLBS    R0,400$                  ; BRANCH IF SUCCESSFUL
                 2C A4   DD  049F   716           PUSHL   FAB$L_FNA(R4)            ; DESCRIPTOR OF FILE NAME
            7E   34 A4   9A  04A2   717           MOVZBL  FAB$B_FNS(R4),-(SP)
                 0C A4   DD  04A6   718           PUSHL   FAB$L_STV(R4)            ; PUSH RMS SECONDARY STATUS
                    50   DD  04A9   719           PUSHL   R0                       ; PUSH RMS PRIMARY STATUS
                 08 AE   9F  04AB   720           PUSHAB  8(SP)                    ; ADDRESS OF DESCRIPTOR
                    01   DD  04AE   721           PUSHL   #1                       ; NUMBER OF FAO ARGUMENTS
          00000000'8F    DD  04B0   722           PUSHL   #MSG$_OPENIN             ; ERROR OPENING INPUT FILE
     00000000'GF    05   FB  04B6   723           CALLS   #5,G^LIB$SIGNAL          ; OUTPUT ERROR MESSAGE
            5E      08   C0  04BD   724           ADDL    #8,SP                    ; REMOVE DESCRIPTOR FROM STACK
                             04C0   725
                             04C0   726 400$:     $CONNECT STB
                             04CD   727           SIGNAL  RMS,STB
                             04E3   728 ;
                             04E3   729 ; SET UP TERMINAL HANDLING IF SYS$INPUT IS A TERMINAL
                             04E3   730 ;
   3B 00000AE7'EF     02  E1 04E3   731           BBC     #DEV$V_TRM,DVI_DEVCHAR,420$ ; EXIT IF NOT TERMINAL
                             04EB   732
                             04EB   733           $ASSIGN_S CHAN = TT_CHAN -
                             04EB   734                   DEVNAM = SYSINPUT        ; SYS$INPUT
                 25 50    E9 04FE   735           BLBC    R0,420$                  ; BRANCH ON ERROR
                             0501   736           $QIOW_S CHAN = TT_CHAN -
                             0501   737                   FUNC = #IO$_SETMODE!IO$M_CTRLCAST -
                             0501   738                   P1 = CTRL_C_AST          ; AST ROUTINE
                             0526   739 420$:
 00000AA3'EF   FB39 CF    9E 0526   740           movab   sda_prompt,smg_prompt    ; Prompt descriptor
                             052F   741
                          04 052F   742           RET
                             0530   743
```

```
                               0530           .SBTTL  GET_INPUT - Get one line of input using RTL
                               0530    ;---
                               0530    ;
                               0530    ; This routine calls SMG$READ_COMPOSED_LINE and is provided for
                               0530    ; any read to SYS$INPUT.
                               0530    ;
                               0530    ;
                               0530    ; INPUTS:
                               0530    ;
                               0530    ;      SMG_PROMPT - loaded with address of prompt to be used.
                               0530    ;
                               0530    ; OUTPUTS:
                               0530    ;
                               0530    ;      INPUT_BUF - descriptor of input
                               0530    ;      INPUT_LEN - length of input
                               0530    ;
                               0530    ;+++
                               0530
                               0530   GET_INPUT::
                               0530
                   7E     D4   0530           clrl    -(SP)                         ; display ID
          000002A4'EF     9F   0532           pushab  input_len                     ; input length
          00000AA3'EF     DD   0538           pushl   smg_prompt                    ; pre-loaded address of prompt
          000002A8'EF     9F   053E           pushab  input_buf                     ; input buffer descriptor
          00000A9B'EF     9F   0544           pushab  keytable                      ; key definition table
          00000A97'EF     9F   054A           pushab  keyboard                      ; keyboard ID
   00000000'GF    06      FB   0550           calls   #6,G^SMG$READ_COMPOSED_LINE ; read file spec
                          05   0557           RSB
                               0558
```

```
                                0558         .SBTTL  CTRL_C_AST - Handle Control C AST routine
                                0558
                                0558 ;+++
                                0558 ;
                                0558 ;       This routine is called as an ast routine whenever ^C (cancel)
                                0558 ;       is typed. The routine signals MSG$_EXITCMD and exits.
                                0558 ;
                                0558 ;
                                0558 ;       0(AP) is zero if called to fake ^C.
                                0558 ;
                                0558 ;---
                                0558
                                0558 CTRL_C_AST:
                          0000  0558         .WORD   0
                                055A
                      6C    D5  055A         TSTL    (AP)                    ; test for fake ^C
                      31    13  055C         BEQL    10$                     ; no need to re-enable ^C
                                055E ;
                                055E ; It would be nice for this to be an out of band, rather than a
                                055E ; simple ^C enable. This prevents the ^Y window. For right now however,
                                055E ; an out of band cannot cancel I/O, so until that work is done...
                                055E ;
                                055E         $QIOW_S CHAN = TT_CHAN -
                                055E                 FUNC = #IO$_SETMODE!IO$M_CTRLCAST -
                                055E                 P1 = CTRL_C_AST         ; AST ROUTINE
                                0580 ;
                                0580 ; If a $PUT is active, just flag ^C pending
                                0580 ;
        08 0000004D'EF      E9  0580         BLBC    PUT_BUSY,10$            ; branch if ok to signal
   0000004C'EF      01      90  0587         MOVB    #1,CTRLC_PENDING        ; set flag
                            04  058E         RET                             ; and exit
                                058F
           0000004C'EF      94  058F 10$:    CLRB    CTRLC_PENDING           ; CLEAR PENDING FLAG
           0000001C'EF      D4  0595         CLRL    LINE_COUNT              ; CLEAR SO NO CONTINUE PROMPT
           00000028'EF      D4  059B         CLRL    CLR_PAGE                ; CLEAR TO AVOID PAGE ERASE
                                05A1         SIGNAL  0,EXITCMD               ; EXIT MESSAGE
                            04  05B3         RET                             ; RET
                                05B4
```

```
                                      05B4   814                 .SBTTL  EXIT_IF_OLD - EXIT IF OLD DUMP AT STARTUP TIME
                                      05B4   815 ;---
                                      05B4   816 ;
                                      05B4   817 ;               THIS ROUTINE WILL EXIT THE PROGRAM IF WE ARE CALLED
                                      05B4   818 ;               FROM STARTUP.COM AT BOOT TIME AND IF THE DUMP HAS
                                      05B4   819 ;               ALREADY BEEN ANALYZED.  OPERATOR SHUTDOWN CRASHES
                                      05B4   820 ;               ARE ALSO IGNORED AS THEY DO NOT CONSTITUTE A REAL
                                      05B4   821 ;               CRASH.
                                      05B4   822 ;
                                      05B4   823 ;       INPUTS:
                                      05P4   824 ;
                                      05B4   825 ;               DUMP_HEADER CONTAINS THE DUMP HEADER BLOCKS
                                      05B4   826 ;               ERLPTR CONTAINS A POINTER TO THE ERROR LOG ENTRY
                                      05B4   827 ;
                                      05B4   828 ;       OUTPUTS:
                                      05B4   829 ;
                                      05B4   830 ;               NONE
                                      05B4   831 ;
                                      05B4   832 ;---
                                      05B4   833
                                0004  05B4   834                 .ENTRY  EXIT_IF_OLD,^M<R2>
                                      05B6   835
                                      05B6   836 ;
                                      05B6   837 ;               CHECK IF WE ARE IN SYSTEM STARTUP PROCEDURE
                                      05B6   838 ;
                                      05B6   839
                                      05B6   840                 ALLOC   15,R2                   ; ALLOCATE 15 BYTE BUFFER
                             7E   D4  05C0   841                 CLRL    -(SP)                   ; CREATE GETJPI REQUEST LIST
                             52   DD  05C2   842                 PUSHL   R2                      ; ADDRESS TO RECEIVE LENGTH
                          04 A2   DD  05C4   843                 PUSHL   4(R2)                   ; ADDRESS OF OUTPUT BUFFER
                    031C000F 8F   DD  05C7   844                 PUSHL   #<JPI$_PRCNAM@16>!15    ; REQUEST CODE AND BUFLTH
                        51   5E   D0  05CD   845                 MOVL    SP,R1
                                      05D0   846                 $GETJPI_S ITMLST=(R1)           ; GET NAME OF THIS PROCESS
                                      05E3   847                 SIGNAL
 FA8C CF   07   20   04 B2   62   2D  05EF   848                 CMPC5   (R2),@4(R2),#^A' ',#STARTUP_LEN,STARTUP
                             36   12  05F8   849                 BNEQ    90$                     ; BRANCH IF NOT STARTUP TIME
                                      05FA   850
                                      05FA   851 ;
                                      05FA   852 ;               EXIT IF DUMP HAS ALREADY BEEN ANALYZED OR IS EMPTY
                                      05FA   853 ;
                             03   B3  05FA   854                 BITW    #<<1@DMP$V_OLDDUMP> ! <1@DMP$V_EMPTY>>,-
                    000002B4'EF       05FC   855                          DUMP_HEADER+DMP$L_FLAGS
                             16   12  0601   856                 BNEQ    20$
                                      0603   857
                                      0603   858 ;
                                      0603   859 ;               ... OR IF OPERATOR SHUTDOWN
                                      0603   860 ;
               51   00000000'EF   D0  0603   861                 MOVL    ERLPTR,R1               ; ADDRESS OF ERROR LOG ENTRY
              51   00F4 C1   07   CB  060A   862                 BICL3   #7,EMB$L_CR_CODE(R1),R1 ; GET BUGCHECK CODE OF CRASH
               00000000'8F   51   D1  0610   863                 CMPL    R1,#BUG$_OPERATOR       ; CHECK IF OPERATOR SHUTDOWN
                             17   12  0617   864                 BNEQ    90$                     ; BRANCH IF NOT
                                      0619   865 ;
                                      0619   866 ;               EXIT THE IMAGE - FLUSH THE REMAINING INPUT COMMANDS
                                      0619   867 ;
                                      0619   868 20$:
            06 00000AE7'EF   02   E0  0619   869                 BBS     #DEV$V_TRM,DVI_DEVCHAR,40$ ; SKIP IF TERMINAL
                                      0621   870 30$:
```

```
        FF0C   30  0621    871          BSBW    GET_INPUT                ; Get input line
       FA 50   E8  0624    872          BLBS    R0,30$                   ; CONTINUE UNTIL ALL DATA READ
                   0627    873 40$:
                   0627    874          $EXIT_S                          ; EXIT THE IMAGE
                   0630    875 90$:
                   0630    876          STATUS  SUCCESS
               04  0637    877          RET
```

```
                                 0638          .SBTTL  PAGE_WAIT - GIVE END-OF-PAGE PROMPT ON SCREEN
                                 0638  ;---
                                 0638  ;
                                 0638  ;          PAGE_WAIT
                                 0638  ;
                                 0638  ;          THIS ROUTINE CAUSES AN END-OF-PAGE PROMPT TO BE GIVEN
                                 0638  ;          ON THE BOTTOM OF THE SCREEN.  IF THE USER SIMPLY HITS
                                 0638  ;          RETURN, HE WILL CONTINUE OUT OF THIS ROUTINE TO PRINT
                                 0638  ;          THE NEXT PAGE.  IF HE ENTERS SOME OTHER COMMAND, THE
                                 0638  ;          CURRENT COMMAND WILL BE ABORTED.
                                 0638  ;
                                 0638  ; INPUTS:
                                 0638  ;
                                 0638  ;          IF LINE_COUNT = 0, NO PROMPT WILL BE ISSUED.
                                 0638  ;
                                 0638  ;---
                                 0638
                                 0638          .ENABL  LSB
                                 0638
                           003C  0638          .ENTRY  PAGE_WAIT,^M<R2,R3,R4,R5>
                                 063A
              0000000C'EF    D5  063A          TSTL    OUTPUT_FILE               ; CHECK IF ANY OUTPUT FILE
                       14    12  0640          BNEQ    5$                        ; SKIP IF LISTING FILE
08 00000ADB'EF   00000000'8F    E1  0642          BBC     #TT$V_SCOPE,DVI_DEVDEPEND,5$ ; SKIP SCROLLING
              0000001C'EF    D5  064E          TSTL    LINE_COUNT                ; 0 FORCES NO PROMPT
                       01    12  0654          BNEQ    10$                       ; BRANCH IF PROMPT WANTED
                             04  0656  5$:     RET
                                 0657  10$:
              0000001C'EF    D4  0657          CLRL    LINE_COUNT                ; CLEAR BEFORE ANYTHING ELSE
                                 065D          SKIP    <PROMPT_LINES-1>          ; MOVE UP IF SCROLLING
                       01    DD  0666          PUSHL   #1                        ; COLUMN 1
         50   00000ADE'EF    9A  0668          MOVZBL  DVI_PAGESIZE,R0           ; GET PAGE SIZE
              7E   50  02    C3  066F          SUBL3   #<PROMPT_LINES-1>,R0,-(SP)      ; 2ND FROM BCTTOM LINE
          00000000'GF  02    FB  0673          CALLS   #2,G^SCR$SET_CURSOR       ; SET CURSOR POSITION
                                 067A          PRINT   0,<    Press RETURN for more.>
                                 0687
000002A0'EF   000002A4'EF    D0  0687          MOVL    INPUT_LEN,SAVE_INPUT_LEN; SAVE CURRENT COMMAND LINE LENGTH
00000200'EF   000002A4'EF    28  0692          MOVC3   INPUT_LEN,INPUT_BUFFER,-; SAVE THE CURRENT COMMAND LINE
              00000250'EF        069D                  SAVE_INPUT_BUFFER         ; BEFORE PROMPTING FOR COMMAND
          00000000'EF  00    FB  06A2          CALLS   #0,GET_COMMANDS           ; PROMPT FOR COMMAND
                    12 50    E9  06A9          BLBC    R0,14$                    ; BRANCH IF EMPTY LINE
                                 06AC  13$:
                                 06AC          SIGNAL  0,BACKUP                  ; SILENTLY ABORT COMMAND
                                 06BE  14$:
          00000000'8F  50    D1  06BE          CMPL    R0,#MSG$_BACKUP           ; ARE WE BACKING UP?
                       E5    13  06C5          BEQL    13$                       ; CONTINUE BACK TO MAIN LEVEL
          00000000'8F  50    D1  06C7          CMPL    R0,#MSG$_EOF              ; CHECK IF END OF FILE (EXIT)
                       25    13  06CE          BEQL    15$                       ; BRANCH IF SO
          00000000'8F  50    D1  06D0          CMPL    R0,#MSG$_EXITCMD          ; ARE WE EXITING COMMAND?
                       1C    13  06D7          BEQL    15$                       ; BRANCH IF SO
000002A4'EF   000002A0'EF    D0  06D9          MOVL    SAVE_INPUT_LEN,INPUT_LEN; RESTORE CURRENT COMMAND LINE LENGTH
              000002A4'EF    28  06E4          MOVC3   INPUT_LEN,-               ; WE ARE CONTINUING WITH THE CURRENT
              00000250'EF        06EA                  SAVE_INPUT_BUFFER,-       ; COMMAND SO RESTORE THE CONTENTS OF
              00000200'EF        06EF                  INPUT_BUFFER              ; THE INPUT BUFFER TO ITS PRIOR STATE
                                 06F4
                             04  06F4          RET
                                 06F5  15$:
```

```
    06F5   936          SIGNAL  0,EXITCMD               ; EXIT AND ERASE SCREEN
```

```
                                  0707             .SBTTL  NEW_PAGE - BEGIN A NEW PAGE ON THE LISTING
                                  0707      ;---
                                  0707      ;
                                  0707      ;      NEW_PAGE
                                  0707      ;
                                  0707      ;      THIS ROUTINE WILL CAUSE A NEW PAGE TO BE WRITTEN AND
                                  0707      ;      WILL OUTPUT THE PAGE HEADING AND CURRENT SUB-HEADING.
                                  0707      ;
                                  0707      ;  INPUTS:
                                  0707      ;
                                  0707      ;      PAGE_NUMBER = CURRENT PAGE NUMBER
                                  0707      ;
                                  0707      ;  OUTPUTS:
                                  0707      ;
                                  0707      ;      PAGE_NUMBER IS UPDATED
                                  0707      ;      LINE_COUNT IS INITIALIZED
                                  0707      ;
                                  0707      ;---
                                  0707
                                  0707             .ENABL  LSB
                                  0707
                            0000  0707             .ENTRY  NEW_PAGE,^M<>
                                  0709
             00000000'EF    D5    0709             TSTL    SUB_HEADING               ; ANY SUB-HEADING?
                      0D    13    070F             BEQL    10$                       ; SKIP CHECK IF NOT
 00000020'EF 0000001C'EF    D1    0711             CMPL    LINE_COUNT,HEADING_LINES ; ANY NEW LINES BESIDES TITLE?
                      6A    13    071C             BEQL    90$                       ; IF NOT, SKIP PAGE EJECT
                                  071E      10$:
              FF15 CF   00  FB    071E             CALLS   #0,PAGE_WAIT              ; GIVE BOTTOM OF PAGE PROMPT
             0000001C'EF    D4    0723             CLRL    LINE_COUNT                ; CLEAR BEFORE ANYTHING ELSE
             00000018'EF    D6    0729             INCL    PAGE_NUMBER               ; INCREMENT PAGE NUMBER
             0000000C'EF    D5    072F             TSTL    OUTPUT_FILE               ; CHECK IF LISTING FILE
                      5D    13    0735             BEQL    50$                       ; NO HEADINGS IF NOT
                                  0737             PRINT   0,<!^>                    ; PRINT FORM FEED
             00000018'EF    DD    0744             PUSHL   PAGE_NUMBER
             0000002C'EF    7F    074A             PUSHAQ  CURRENT_TIME
                                  0750             PRINT   2,<VAX/VMS 4.0 -- System Dump Analysis!_!_!_!_!AS!_!_!_Page !UL>
             00000000'EF    7F    075D             PUSHAQ  SUB_HEADING               ; SECTION HEADING
                                  0763             PRINT   1,<!AS>
                                  0770             SKIP    3                         ; 3 BLANK LINES
                                  0779      60$:
             00000000'EF    D5    0779             TSTL    HEADING_ROUTINE           ; ANY HEADING ROUTINE?
                      07    13    077F             BEQL    90$                       ; BRANCH IF NOT
        00000000'FF   00    FB    0781             CALLS   #0,@HEADING_ROUTINE       ; CALL THE ROUTINE
                                  0788      90$:
 00000020'EF 0000001C'EF    D0    0788             MOVL    LINE_COUNT,HEADING_LINES ; REMEMBER # HEADING LINES
                            04    0793             RET
                                  0794      50$:
                                  0794      ;      SKIP SUB-HEADING IF NOT SCREEN ORIENTED DEVICE
E8 00000ADB'EF 00000000'8F  E1    0794             BBC     #TT$V_SCOPE,DVI_DEVDEPEND,90$ ;
                      01    DD    07A0             PUSHL   #1                        ; FROM COLUMN 1
                      01    DD    07A2             PUSHL   #1                        ; LINE 1
        00000000'GF   02    FB    07A4             CALLS   #2,G^SCR$ERASE_PAGE       ; ERASE ENTIRE SCREEN
             00000000'EF    7F    07AB             PUSHAQ  SUB_HEADING               ; SECTION HEADING
                                  07B1             PRINT   1,<!AS>
          7E 00000000'EF    3C    07BE             MOVZWL  SUB_HEADING,-(SP)         ; CHARACTERS IN HEADING
                                  07C5             PRINT   1,<!#*->
```

```
A5   11  07D2    995         BRB     60$
         07D4    996
         07D4    997         .DSABL  LSB
```

```
                                07D4          .SBTTL  PRINT -- FORMAT AND PRINT A SINGLE LINE
                                07D4    ;---
                                07D4    ;
                                07D4    ;       PRINT
                                07D4    ;
                                07D4    ;       THIS ROUTINE IS INVOKED FROM THE PRINT MACRO TO FORMAT
                                07D4    ;       AND PRINT A SINGLE LINE.
                                07D4    ;
                                07D4    ; INPUTS:
                                07D4    ;
                                07D4    ;        4(AP) = ADDRESS OF CONTROL STRING
                                07D4    ;        8(AP) = FAO PARAMETERS (AS MANY AS NEEDED)
                                07D4    ;
                                07D4    ; OUTPUTS:
                                07D4    ;
                                07D4    ;       NONE
                                07D4    ;
                                07D4    ;---
                                07D4
                          0000  07D4          .ENTRY  PRINT,^M<>
                                07D6
              08 AC       DF    07D6          PUSHAL  8(AP)                   ; ADDRESS OF PARAMETER LIST
         000008B0'EF      7F    07D9          PUSHAQ  LINE_DESCR              ; BUFFER DESCRIPTOR
         000002F9'EF      DF    07DF          PUSHAL  LIST+RAB$W_RSZ          ; TO RECEIVE LENGTH OF LINE
              04 AC       DD    07E5          PUSHL   4(AP)                   ; ADDRESS OF CONTROL STRING
   00000000'GF    04      FB    07E8          CALLS   #4,G^SYS$FAOL           ; FORMAT LINE
         F4'AF    00      FB    07EF          CALLS   #0,B^PUT_LINE           ; OUTPUT LINE
                          04    07F3          RET
```

```
                                     07F4               .SBTTL  PUT_LINE - OUTPUT A LINE TO THE LISTING FILE
                                     07F4       ;---
                                     07F4       ;
                                     07F4       ;       PUT_LINE
                                     07F4       ;
                                     07F4       ;       THIS ROUTINE OUTPUTS A SINGLE PRINT LINE TO THE LISTING
                                     07F4       ;       FILE.  THE NUMBER OF LINES ON THE PAGE IS ACCOUNTED FOR
                                     07F4       ;       AND A NEW PAGE WILL BE ISSUED WHEN THE PAGE IS FULL.
                                     07F4       ;
                                     07F4       ;   INPUTS:
                                     07F4       ;
                                     07F4       ;       LINE_COUNT      = NUMBER OF LINES ON CURRENT PAGE
                                     07F4       ;
                                     07F4       ;   OUTPUTS:
                                     07F4       ;
                                     07F4       ;       LINE_COUNT IS UPDATED
                                     07F4       ;
                                     07F4       ;---
                                     07F4
                                     07F4               .ENABL  LSB
                                     07F4
                               01FC  07F4               .ENTRY  PUT_LINE,^M<R2,R3,R4,R5,R6,R7,R8>
                                     07F6
       0000004D'EF    01         90  07F6               MOVB    #1,PUT_BUSY                     ; flag "Put Busy"
       56   0000001C'EF          DE  07FD               MOVAL   LINE_COUNT,R6                   ; ADDRESS OF LINE COUNT
       58   000002D7'EF          DE  0804               MOVAL   LIST,R8                         ; ADDRESS OF LIST RAB
                      57         D4  080B               CLRL    R7                              ; R7=0 IF TERMINAL OUTPUT
            0000000C'EF          D5  080D               TSTL    OUTPUT_FILE                     ; OUTPUT FILE SPECIFIED?
                      03         13  0813               BEQL    10$                             ; BRANCH IF NOT
                 57   01         D0  0815               MOVL    #1,R7                           ; R7=1 IF LISTING OUTPUT
                                     0818       10$:
                      66         D6  0818               INCL    (R6)                            ; ASSUME 1 LINE PRINTED
                   0C 57         E8  081A               BLBS    R7,5$                           ; BRANCH IF LISTING FILE
    00000AE3'EF    22 A8         B1  081D               CMPW    RAB$W_RSZ(R8),DVI_DEVBUFSIZ
                                     0825                                                       ; CHECK IF OVER SIZE OF SCREEN
                      02         1B  0825               BLEQU   5$                              ; BRANCH IF OK
                      66         D6  0827               INCL    (R6)                            ; ACCOUNT FOR 2 LINES
                                     0829       5$:
       00000024'EF    66         D1  0829               CMPL    (R6),PAGE_SIZE
                      3A         15  0830               BLEQ    20$                             ; BRANCH IF STILL ROOM
            00000024'EF          D5  0832               TSTL    PAGE_SIZE                       ; CHECK IF VALID PAGE SIZE
                      32         15  0838               BLEQ    20$                             ; BRANCH IF NO PAGE SIZE (FILE)
              7E   22 A8         B0  083A               MOVW    RAB$W_RSZ(R8),-(SP)             ; SAVE LINE LENGTH
       5E   0000012C 8F          C2  083E               SUBL2   #LIST_BUFFER_LEN,SP             ; ALLOCATE SPACE FOR LINE
 6E   000008B8'EF   012C 8F      28  0845               MOVC3   #LIST_BUFFER_LEN,LIST_BUFFER,(SP)  ; SAVE LINE
                                     084F               SKIP    PAGE
 000008B8'EF   6E   012C 8F      28  0854               MOVC3   #LIST_BUFFER_LEN,(SP),LIST_BUFFER  ; RESTORE LINE
       5E   0000012C 8F          C0  085E               ADDL2   #LIST_BUFFER_LEN,SP             ; DEALLOCATE SPACE
            22 A8   8E           B0  0865               MOVW    (SP)+,RAB$W_RSZ(R8)             ; RESTORE LINE LENGTH
                    FFAC         31  0869               BRW     10$                             ; TRY AGAIN
                                     086C       20$:
                   6C 57         E8  086C               BLBS    R7,50$                          ; BRANCH IF LISTING FILE
       52   000005E7'EF          9E  086F               MOVAB   OUTPUT,R2
          22 A2    22 A8         B0  0876               MOVW    RAB$W_RSZ(R8),RAB$W_RSZ(R2)
    28 A2   000008B4'EF          D0  087B               MOVL    LINE_DESCR+4,RAB$L_RBF(R2)
                                     0883               $PUT    (R2)                            ; OUTPUT TO TERMINAL
                                     088C               SIGNAL  RMS,(R2)
```

```
                                089E
            6A 08 A2        E9  089E            BLBC    RAB$L_STS(R2),100$          ; OUTPUT TO LOGFILE IF LOGGING ENABLED
          00000004'EF       95  08A2            TSTB    LOG_FILE                    ; AND OUTPUT TO TERMINAL SUCCEEDED
                  62        13  08A8            BEQL    100$
     52   000003CB'EF       9E  08AA            MOVAB   LOGRAB,R2
       22 A2    22 A8       B0  08B1            MOVW    RAB$W_RSZ(R8),RAB$W_RSZ(R2)
 28 A2    000008B4'EF       D0  08B6            MOVL    LINE_DESCR+4,RAB$L_RBF(R2)
                                08BE            $PUT    (R2)
                                08C7            SIGNAL  RMS,(R2)
                  31        11  08D9            BRB     100$
                                08DB       50$:
                                08DB            $PUT    (R8)                        ; OUTPUT RECORD TO THE FILE
                                08E4            SIGNAL  RMS,(R8)
012C 8F    20    6E   00    2C  08F6            MOVC5   #0,(SP),#^A' ',#LIST_BUFFER_LEN,LIST_BUFFER
          000008B8'EF           08FD
               22 A8        B4  0902            CLRW    RAB$W_RSZ(R8)               ; RESET TO EMPTY LINE
                                0905            STATUS  SUCCESS
                                090C      100$:
                                090C            ;
                                090C            ; clear "Put Busy" and check for ^C pending flag
                                090C            ;
          0000004D'EF       94  090C            CLRB    PUT_BUSY
       01 0000004C'EF       E8  0912            BLBS    CTRLC_PENDING,110$
                            04  0919            RET
                                091A      110$:
         FC39 CF     00     FB  091A            CALLS   #0,CTRL_C_AST               ; fake ^C operation
                            04  091F            RET
                                0920
                                0920            .DSABL  LSB
```

```
                    0920          .SBTTL  SKIP_LINES - SKIP ANY NUMBER OF BLANK LINES
                    0920  ;---
                    0920  ;
                    0920  ;       SKIP_LINES
                    0920  ;
                    0920  ;       THIS ROUTINE WILL OUTPUT A SPECIFIED NUMBER OF BLANK
                    0920  ;       LINES TO THE LISTING FILE.
                    0920  ;
                    0920  ;   INPUTS:
                    0920  ;
                    0920  ;       4(AP)   = THE NUMBER OF LINES TO SKIP
                    0920  ;
                    0920  ;   OUTPUTS:
                    0920  ;
                    0920  ;       THE BLANK LINES ARE OUTPUT
                    0920  ;
                    0920  ;---
                    0920
             0000   0920          .ENTRY  SKIP_LINES,^M<>
                    0922
       04 AC    D5  0922          TSTL    4(AP)                    ; CHECK IF ALREADY DONE
          0F    13  0925          BEQL    90$
                    0927  10$:
 000002F9'EF    B4  0927          CLRW    LIST+RAB$W_RSZ           ; EMPTY LINE
FEC2 CF   00    FB  092D          CALLS   #0,PUT_LINE              ; OUTPUT A BLANK LINE
    F1 04 AC    F5  0932          SOBGTR  4(AP),10$
                    0936  90$:
                04  0936          RET
```

```
0937            .SBTTL  PRINT_COLUMNS -- PRODUCE COLUMNAR OUTPUT
0937      ;---
0937      ;
0937      ;     PRINT_COLUMNS
0937      ;
0937      ;     Based upon input parameters, tables, and action routine outputs, this
0937      ;     routine produces multi-column displays.  This routine has the
0937      ;     following major features:
0937      ;
0937      ;       o it is entirely input driven
0937      ;
0937      ;       o an action routine my signal that the entry it is preparing is not
0937      ;         to be included in this display.  This will result in the
0937      ;         successive column entries in that column being moved up one row.
0937      ;
0937      ;       o ragged bottoms of columns are properly handled.
0937      ;
0937      ;     It is assumed that each column is to contain three sections, a text
0937      ;     description of a value followed by the value followed by a spacer to
0937      ;     the next column.
0937      ;
0937      ; INPUTS:
0937      ;
0937      ;          (AP)       number of arguments [ (AP)-((COLLS1/4)-1  gives the
0937      ;                     number of columns]
0937      ;     DATBAS(AP)      base address for data structure against which offsets apply
0937      ;     DATSVA(AP)      system virtual address of data structure
0937      ;                     (used only on queue header processing)
0937      ;     COLLS1(AP)      base address of the COLUMN_LIST for column 1
0937      ;     COLLS1+4(AP)    base address of the COLUMN_LIST for column 2
0937      ;     COLLS1+8(AP)    base address of the COLUMN_LIST for column 3
0937      ;       .                 .
0937      ;       .                 .
0937      ;       .                 .
0937      ;
0937      ; OUTPUTS:
0937      ;
0937      ;     NONE
0937      ;
0937      ; Description of the COLUMN_LIST macro:
0937      ;
0937      ; Format:
0937      ;
0937      ; BASE: COLUMN_LIST -
0937      ;               prefix, df-desc-size, df-val-size, df-sep-size, < -
0937      ;               -                                ; row 1 description this column
0937      ;               <<string>,offset,type,desc-size,val-size,sep-size>, -
0937      ;               -                                ; row 2 description this column
0937      ;               <<string>,action,value,desc-size,val-size,sep-size>, -
0937      ;               -                                ; row 3 description this column
0937      ;               <<string>,offset,type,desc-size,val-size,sep-size>, -
0937      ;                     .                                       .
0937      ;                     .                                       .
0937      ;                     .                                       .
0937      ;               >
0937      ;
0937      ;  Where:
```

```
0937  1203 ;
0937  1204 ;          prefix            is the data structure prefix
0937  1205 ;          df-desc-size      is the default description string size for this column
0937  1206 ;          df-val-size       is the default value string size for this column
0937  1207 ;          df-sep-size       is the default seperator size for this column
0937  1208 ;          string            is the description string for an entry
0937  1209 ;          offset            is the data structure offset name for the value
0937  1210 ;                                (w/o the prefix)
0937  1211 ;          type              is one of the following FAO directives w/o the "!"
0937  1212 ;                                AC, AS, OB, OW, OL, XB, XW, XL, ZB, ZW, ZL,
0937  1213 ;                                UB, UW, UL, SB, SW, or SL
0937  1214 ;                            the following types cause conversion only if the
0937  1215 ;                                value is not zero:
0937  1216 ;                                OB_NEQ, OW_NEQ, OL_NEQ, XB_NEQ, XW_NEQ, XL_NEQ,
0937  1217 ;                                ZB_NEQ, ZW_NEQ, ZL_NEQ, UB_NEQ, UW_NEQ, UL_NEQ,
0937  1218 ;                                SB_NEQ, SW_NEQ, and SL_NEQ
0937  1219 ;                            the following special codes are also available:
0937  1220 ;                                Q2  - doubly-linked queue header (does not work
0937  1221 ;                                    with DO_COLUMN_ENTRY macro)
0937  1222 ;          action            is an action routine name
0937  1223 ;          value             is a longword value to be passed to the action routine
0937  1224 ;          desc-size         over-rides df-desc-size on this entry
0937  1225 ;          val-size          over-rides df-val-size on this entry
0937  1226 ;          sep-size          over-rides df-sep-size on this entry
0937  1227 ;
0937  1228 ;       Action Routine Inputs:
0937  1229 ;
0937  1230 ;          R2                value from the COLUMN_LIST entry
0937  1231 ;          R5                size of the value field for this entry
0937  1232 ;          R7                address of a descriptor for the scratch string in
0937  1233 ;                            which the FAO converted value is to be returned
0937  1234 ;          R11               base address of the data structure from DATBAS(AP)
0937  1235 ;
0937  1236 ;       Action Routine Outputs:
0937  1237 ;
0937  1238 ;          R0                status
0937  1239 ;                              lbs ==> use this entry
0937  1240 ;                              lbc ==> skip this entry
0937  1241 ;          R1 - R5           scratch
0937  1242 ;                            all other registers must be preserved
0937  1243 ;
0937  1244 ;       Action routines may also use the DO_COLUMN_ENTRY macro to access any
0937  1245 ;       of the conversion services available through the COLUMN_LIST type
0937  1246 ;       parameter.
0937  1247 ;
0937  1248 ;       Invocation:
0937  1249 ;
0937  1250 ;          DO_COLUMN_ENTRY type [,jump]
0937  1251 ;
0937  1252 ; Parameters:
0937  1253 ;
0937  1254 ;          type     FAO type (anything valid in the COLUMN_LIST macro, except Q2,
0937  1255 ;                   is valid here)
0937  1256 ;          jump     JMP or JSB controlling transfer to subroutine
0937  1257 ;                   (JSB is the default: if JMP is used control does not return
0937  1258 ;                   to the action routine)
0937  1259 ;
```

```
                        0937  ; Inputs:
                        0937  ;
                        0937  ;       R2      address of the datum or its descriptor
                        0937  ;       R5      field size  (as input to the action routine)
                        0937  ;---
                        0937
                        0937          $OFFSET 4,POSITIVE,< -                  ; Argument list offsets:
                        0937                  DATBAS, -
                        0937                  DATSVA, -
                        0937                  COLLS1 -
                        0937                  >
                        0004  DATBAS:
                        0008  DATSVA:
                        000C  COLLS1:
                        0937
                        0937          .SAVE
                    0000010C          .PSECT  LITERALS,EXE,NOWRT
                        010C
                        010C  ONE_COL:
                        010C          STRING  <!#AC!#AS!#* >
                        0120
                        0120  NULL_ASCID:                             ; A null .ASCID string
                        0120  NULL_ASCIC:                             ; A null .ASCIC string
   00000000 00000000    0120          .LONG   0, 0
                        0128
                    00000937          .RESTORE
                        0937
                        0937          $OFFSET -4,NEGATIVE,< -                 ; FP offsets for stack scratch:
                        0937                  <LINE_CTRSTR, 8>, -             ;  1 line FAO CTRSTR descriptor
                        0937                  NUMCOL, -                       ;  number of columns
                        0937                  SCRATCH_SIZE, -                 ;  size of one scratch string
                        0937                  COLLST_BASE, -                  ;  base of COLUMN_LIST pointers
                        0937                  COLSCRATCH_BASE, -              ;  base of per-column scratch
                        0937                  -                               ;      pointers
                        0937                  FLAGS, -                        ;  flags
                        0937                  <STACK_LEN, 0> -                ; end of stack storage
                        0937                  >
                        FFF4  LINE_CTRSTR:
                        FFF0  NUMCOL:
                        FFEC  SCRATCH_SIZE:
                        FFE8  COLLST_BASE:
                        FFE4  COLSCRATCH_BASE:
                        FFE0  FLAGS:
                        FFE0  STACK_LEN:
                        0937
                        0937          _VIELD  FLAGS,0,< -                     ; fields in FLAGS above:
                        0937                  <NO_ENTRIES,,M> -               ;  no entries on this line
                        0937                  >
                        0937
                        0937  ;
                        0937  ; Out-of-line code used during PRINT_COLUMNS setup
                        0937  ;
                        0937
                  04    0937  PC_XIT: RET                             ; Zero columns - so exit.
                        0938
                        0938  ;
                        0938  ; PRINT_COLUMNS entry point
```

```
                                 0938      ;
                                 0938
                                 0938      PRINT_COLUMNS::
                           0FFC  0938              .WORD   ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                                 093A
          5B   6C   02     A3    093A              SUBW3   #<<COLLS1/4>-1>, (AP), R11     ; Compute number of columns.
                    F7     15    093E              BLEQ    PC_XIT                         ; Branch if no columns.
                                 0940
          5E    E0 AE      9E    0940              MOVAB   STACK_LEN(SP), SP              ; Allocate space on the stack.
             5B     5B     3C    0944              MOVZWL  R11, R11                       ; Zero extend columns count.
          F0 AD     5B     D0    0947              MOVL    R11, NUMCOL(FP)                ; Save column count.
                                 094B
   52   5B   0000010C'EF   A5    094B              MULW3   ONE_COL, R11, R2               ; Compute size needed for the
          51   52   03     A1    0953              ADDW3   #3, R2, R1                     ;  line FAO control string
       51   FFFF0003 8F    CA    0957              BICL    #^XFFFF0003, R1                ;  (rounded to a longword).
             5E     51     C2    095E              SUBL    R1, SP                         ; Allocate space on the stack.
          F4 AD     52     3C    0961              MOVZWL  R2, LINE_CTRSTR(FP)            ; Setup size in descriptor.
          F8 AD     5E     D0    0965              MOVL    SP, LINE_CTRSTR+4(FP)          ; Setup starting address too.
                                 0969
       59    0000010C'EF   7D    0969              MOVQ    ONE_COL, R9                    ; Get 1 column FAO descriptor.
             53     5E     D0    0970              MOVL    SP, R3                         ; Initialize output str. addr.
          63   6A   59     28    0973      10$:    MOVC3   R9, (R10), (R3)                ; Make one copy of the single
                 F9 5B     F5    0977              SOBGTR  R11, 10$                       ;  col. string for every col.
                                 097A
                                 097A      SETUP_COL_SCRATCH:
                                 097A
          5B    F0 AD      D0    097A              MOVL    NUMCOL(FP), R11                ; Get number of columns.
          5A    0C AC      D0    097E              MOVL    COLLS1(AP), R10                ; Get a COLUMN_LIST entry.
     59   0D AA   0C AA    81    0982              ADDB3   COLM$B_DESC_SIZE(R10), -       ; Assume that value field can
                                 0988                      COLM$B_VAL_SIZE(R10), R9       ;  never be bigger than total
          59    0E AA      80    0988              ADDB    COLM$B_SEP_SIZE(R10), R9       ;  column size.
             59     03     80    098C              ADDB    #3, R9                         ; Round that to a longword.
       59   FFFFFF03 8F    CA    098F              BICL    #^XFFFFFF03, R9                ; That times number of columns
          58   59   5B     C5    0996              MULL3   R11, R9, R8                    ;  is the size of scratch.
          EC AD     59     D0    099A              MOVL    R9, SCRATCH_SIZE(FP)           ; Save per-col. scratch size.
             5E     58     C2    099E              SUBL    R8, SP                         ; Allocate scratch on stack.
             5A     5E     D0    09A1              MOVL    SP, R10                        ; Save starting scr. address.
             7E     59     7D    09A4      10$:    MOVQ    R9, -(SP)                      ; Make a descriptor.
             5A     59     C0    09A7              ADDL    R9, R10                        ; Move to next slot.
                 F7 5B     F5    09AA              SOBGTR  R11, 10$                       ; Loop through all columns.
                                 09AD
          E4 AD     5E     D0    09AD              MOVL    SP, COLSCRATCH_BASE(FP)        ; Save scratch descriptors
                                 09B1                                                     ;  pointer.
                                 09B1
                                 09B1      SETUP_COL_INFO:
                                 09B1
          50    08 AC      9E    09B1              MOVAB   COLLS1-4(AP), R0               ; Get indexable base for
                                 09B5                                                     ; COLUMN_LIST addresses in
                                 09B5                                                     ; argument list.
          51    F0 AD      D0    09B5              MOVL    NUMCOL(FP), R1                 ; Get columns count.
                  6041     DD    09B9      10$:    PUSHL   (R0)[R1]                       ; Init column list pointers on
                 FA 51     F5    09BC              SOBGTR  R1, 10$                        ; the stack.
          E8 AD     5E     D0    09BF              MOVL    SP, COLLST_BASE(FP)            ; Save base addr. for pointers.
                                 09C3
          5B    04 AC      D0    09C3              MOVL    DATBAS(AP), R11                ; Get data structure base addr.
                                 09C7
                                 09C7      LINES_LOOP:
```

```
       5A    F0 AD     01    C3  09C7   1364          SUBL3   #1, NUMCOL(FP), R10             ; Initial columns counter.
             E0 AD     01    C8  09CC   1365          BISL    #FLAGS_M_NO_ENTRIES, FLAGS(FP)  ; Initially, no entries made.
                                 09D0   1366
                                 09D0   1367 COLUMN_LOOP:
        59     E8 BD4A       D0  09D0   1368          MOVL    @COLLST_BASE(FP)[R10], R9       ; Get current COLUMN_LIST ptr.
         7E       0E A9      9A  09D5   1369          MOVZBL  COLM$B_SEP_SIZE(R9), -(SP)      ; Assume a null entry
         00000120'EF         9F  09D9   1370          PUSHAB  NULL_ASCID                      ;  for this column.
         55       0D A9      9A  09DF   1371          MOVZBL  COLM$B_VAL_SIZE(R9), R5         ; Also, setup val. size for
                     55      DD  09E3   1372          PUSHL   R5                              ;  action routine.
         00000120'EF         9F  09E5   1373          PUSHAB  NULL_ASCIC
         7E       0C A9      9A  09EB   1374          MOVZBL  COLM$B_DESC_SIZE(R9), -(SP)
                                 09EF   1375
                                 09EF   1376 DO_COLLST_ENTRY:
                     69      D5  09EF   1377          TSTL    COLM$L_STRING(R9)               ; Is this list exhausted?
                     44      13  09F1   1378          BEQL    NEXT_COLUMN                     ; Branch if list exhausted.
        57     E4 BD4A       7E  09F3   1379          MOVAQ   @COLSCRATCH_BASE(FP)[R10], R7   ; Get scratch descriptor for
                                 09F8   1380                                                  ;  this column.
                                 09F8   1381
         58       04 A9      D0  09F8   1382          MOVL    COLM$L_SOURCE(R9), R8           ; Get data source descriptor.
                     08      14  09FC   1383          BGTR    200$                            ; Branch if its an action rout.
                                 09FE   1384
                   0064      30  09FE   1385          BSBW    DO_ONE_COLUMN                   ; Convert data to ASCID.
                  21 50      E8  0A01   1386          BLBS    R0, 400$                        ; Branch if value returned.
                     09      11  0A04   1387          BRB     300$                            ; Else, skip this entry.
                                 0A06   1388
         52       08 A9      D0  0A06   1389 200$:    MOVL    COLM$L_ACTION_VALUE(R9), R2     ; Get supplied value.
                     68      16  0A0A   1390          JSB     (R8)                            ; Call action routine.
                  16 50      E8  0A0C   1391          BLBS    R0, 400$                        ; Branch if value returned
                                 0A0F   1392
           59        10      C0  0A0F   1393 300$:    ADDL    #COLM$K_LENGTH, R9              ; Else, move to the next entry.
      10 AE       0E A9      9A  0A12   1394          MOVZBL  COLM$B_SEP_SIZE(R9), 16(SP)     ; Setup new separation, value,
         55       0D A9      9A  0A17   1395          MOVZBL  COLM$B_VAL_SIZE(R9), R5         ;  and descriptor sizes.
         08 AE       55      D0  0A1B   1396          MOVL    R5, 8(SP)
         6E       0C A9      9A  0A1F   1397          MOVZBL  COLM$B_DESC_SIZE(R9), (SP)
                     CA      11  0A23   1398          BRB     DO_COLLST_ENTRY                 ; Try processing this entry.
                                 0A25   1399
         0C AE       57      D0  0A25   1400 400$:    MOVL    R7, 12(SP)                      ; Replace null ASCID for value.
         04 AE       69      D0  0A29   1401          MOVL    COLM$L_STRING(R9), 4(SP)        ; Replace descriptive text too.
         E0 AD       01      CA  0A2D   1402          BICL    #FLAGS_M_NO_ENTRIES, FLAGS(FP)  ; Indicate an entry was made.
     E8 BD4A     59  10      C1  0A31   1403          ADDL3   #COLM$K_LENGTH, R9, -           ; Setup next COLUMN_LIST entry
                                 0A37   1404                  @COLLST_BASE(FP)[R10]           ;  in pointer table.
                                 0A37   1405
                                 0A37   1406 NEXT_COLUMN:
                  96 5A      F4  0A37   1407          SOBGEQ  R10, COLUMN_LOOP                ; Loop till all columns done.
                                 0A3A   1408
                                 0A3A   1409 PRINT_A_LINE:
                                 0A3A   1410          ASSUME  FLAGS_V_NO_ENTRIES EQ 0
               23 E0 AD      E8  0A3A   1411          BLBS    FLAGS(FP), ALL_DONE             ; If no entries made, all done.
       52    F0 AD     05    C5  0A3E   1412          MULL3   #5, NUMCOL(FP), R2              ; Compute number of FAO args.
                                 0A43   1413          PRINTD  R2, LINE_CTRSTR(FP)             ; Print the line.
                                 0A4D   1414
                                 0A4D   1415 RESTORE_SCRATCH_DESCRIPTORS:
         51       F0 AD      D0  0A4D   1416          MOVL    NUMCOL(FP), R1                  ; Get number of columns.
         50       E4 AD      D0  0A51   1417          MOVL    COLSCRATCH_BASE(FP), R0         ; Get base of scratch desc.
         80       EC AD      D0  0A55   1418 10$:     MOVL    SCRATCH_SIZE(FP), (R0)+         ; Restore a size value.
                     80      D5  0A59   1419          TSTL    (R0)+                           ; Skip the address.
                  F7 51      F5  0A5B   1420          SOBGTR  R1, 10$                         ; Do all scratch descriptors.
```

```
                        0A5E   1421
        FF66    31      0A5E   1422          BRW     LINES_LOOP                  ; Try for another line.
                        0A61   1423
                        0A61   1424 ALL_DONE:
                04      0A61   1425          RET                                 ; All done, so return.
```

```
MA
Sy

ON
OP
OP
OP
OU
OU
OU
PA
PA
PA
PC
PR
PR
PR
PR
PR
PR
PU
PU
QH
QU
RA
RA
RA
RA
RA
RA
RA
RA
RA
RA
RA
RA
RA
RE
RE
RE
RM
RM
SA
SA
SA
SA
SA
SC
SC
SC
SD
SE
SE
SI
SK
SM
SM
SM
SM
ST
```

```
0A62  ;         Internal routine for PRINT_COLUMNS
0A62  ;
0A62  ;---
0A62  ;
0A62  ;         DO_ONE_COLUMN -- process a single -- non-action-routine -- column entry
0A62  ;
0A62  ;         This routine is the single column entry action routine used by
0A62  ;         PRINT_COLUMNS when a data structure offset is specified in place of an
0A62  ;         action routine.  The specified data is located, converted to ASCII,
0A62  ;         and the resulting string descriptor is returned.
0A62  ;
0A62  ;         PRINT_COLUMN_VALUE -- action routine callback which processes one value
0A62  ;
0A62  ;         This is the target of the DO_COLUMN_ENTRY macro.
0A62  ;
0A62  ;     INPUTS:
0A62  ;
0A62  ;         R2                  address of datum or its descriptor
0A62  ;                             (PRINT_COLUMN_VALUE only)
0A62  ;         R4                  byte COLM$K_FAO_xxx value
0A62  ;                             (PRINT_COLUMN_VALUE only)
0A62  ;
0A62  ;         R5                  size of the value section for this item
0A62  ;         R7                  address of descriptor for a scratch string
0A62  ;
0A62  ;         R8                  minus offset in data structure which locates data
0A62  ;                             (DO_ONE_COLUMN only)
0A62  ;         R9                  address of the current COLUMN_LIST entry
0A62  ;                             (DO_ONE_COLUMN only)
0A62  ;         R11                 data structure base
0A62  ;                             (DO_ONE_COLUMN only)
0A62  ;
0A62  ;     OUTPUTS:
0A62  ;
0A62  ;         R0 - R4             scratch
0A62  ;                             all other registers preserved
0A62  ;
0A62  ;---
0A62
0A62            ASSUME COLM$K_FAO_AC EQ 0       ; Besure type dispatching and
0A62            ASSUME COLM$K_FAO_AS EQ 1       ; the FAO string table will work
0A62            ASSUME COLM$K_FAO_OB EQ 2
0A62            ASSUME COLM$K_FAO_XB EQ 3
0A62            ASSUME COLM$K_FAO_ZB EQ 4
0A62            ASSUME COLM$K_FAO_UB EQ 5
0A62            ASSUME COLM$K_FAO_SB EQ 6
0A62            ASSUME COLM$K_FAO_OW EQ 7
0A62            ASSUME COLM$K_FAO_XW EQ 8
0A62            ASSUME COLM$K_FAO_ZW EQ 9
0A62            ASSUME COLM$K_FAO_UW EQ 10
0A62            ASSUME COLM$K_FAO_SW EQ 11
0A62            ASSUME COLM$K_FAO_OL EQ 12
0A62            ASSUME COLM$K_FAO_XL EQ 13
0A62            ASSUME COLM$K_FAO_ZL EQ 14
0A62            ASSUME COLM$K_FAO_UL EQ 15
0A62            ASSUME COLM$K_FAO_SL EQ 16
0A62            ASSUME COLM$K_FAO_OB_NEQ EQ <COLM$K_FAO_OB + ^X80>
```

```
                                    0A62  1484          ASSUME COLM$K_FAO_XB_NEQ EQ <COLM$K_FAO_XB + ^X80>
                                    0A62  1485          ASSUME COLM$K_FAO_ZB_NEQ EQ <COLM$K_FAO_ZB + ^X80>
                                    0A62  1486          ASSUME COLM$K_FAO_UB_NEQ EQ <COLM$K_FAO_UB + ^X80>
                                    0A62  1487          ASSUME COLM$K_FAO_SB_NEQ EQ <COLM$K_FAO_SB + ^X80>
                                    0A62  1488          ASSUME COLM$K_FAO_OW_NEQ EQ <COLM$K_FAO_OW + ^X80>
                                    0A62  1489          ASSUME COLM$K_FAO_XW_NEQ EQ <COLM$K_FAO_XW + ^X80>
                                    0A62  1490          ASSUME COLM$K_FAO_ZW_NEQ EQ <COLM$K_FAO_ZW + ^X80>
                                    0A62  1491          ASSUME COLM$K_FAO_UW_NEQ EQ <COLM$K_FAO_UW + ^X80>
                                    0A62  1492          ASSUME COLM$K_FAO_SW_NEQ EQ <COLM$K_FAO_SW + ^X80>
                                    0A62  1493          ASSUME COLM$K_FAO_OL_NEQ EQ <COLM$K_FAO_OL + ^X80>
                                    0A62  1494          ASSUME COLM$K_FAO_XL_NEQ EQ <COLM$K_FAO_XL + ^X80>
                                    0A62  1495          ASSUME COLM$K_FAO_ZL_NEQ EQ <COLM$K_FAO_ZL + ^X80>
                                    0A62  1496          ASSUME COLM$K_FAO_UL_NEQ EQ <COLM$K_FAO_UL + ^X80>
                                    0A62  1497          ASSUME COLM$K_FAO_SL_NEQ EQ <COLM$K_FAO_SL + ^X80>
                                    0A62  1498
                                    0A62  1499          .SAVE
                                00000128  1500          .PSECT  LITERALS,EXE,NOWRT
                                    0128  1501
            43 41 21 20 2A 23 21    0128  1502 FAO_AC:  .ASCII  /!#* !AC/
            53 41 21 20 2A 23 21    012F  1503 FAO_AS:  .ASCII  /!#* !AS/
                     42 4F 23 21    0136  1504 FAO_OB:  .ASCII  /!#OB/
                     42 58 23 21    013A  1505 FAO_XB:  .ASCII  /!#XB/
                     42 5A 23 21    013E  1506 FAO_ZB:  .ASCII  /!#ZB/
                     42 55 23 21    0142  1507 FAO_UB:  .ASCII  /!#UB/
                     42 53 23 21    0146  1508 FAO_SB:  .ASCII  /!#SB/
                     57 4F 23 21    014A  1509 FAO_OW:  .ASCII  /!#OW/
                     57 58 23 21    014E  1510 FAO_XW:  .ASCII  /!#XW/
                     57 5A 23 21    0152  1511 FAO_ZW:  .ASCII  /!#ZW/
                     57 55 23 21    0156  1512 FAO_UW:  .ASCII  /!#UW/
                     57 53 23 21    015A  1513 FAO_SW:  .ASCII  /!#SW/
                     4C 4F 23 21    015E  1514 FAO_OL:  .ASCII  /!#OL/
                     4C 58 23 21    0162  1515 FAO_XL:  .ASCII  /!#XL/
                     4C 5A 23 21    0166  1516 FAO_ZL:  .ASCII  /!#ZL/
                     4C 55 23 21    016A  1517 FAO_UL:  .ASCII  /!#UL/
                     4C 53 23 21    016E  1518 FAO_SL:  .ASCII  /!#SL/
                                    0172  1519
                                    0172  1520 FAO_TABLE:
          00000128'00000007'        0172  1521          .ADDRESS 7,FAO_AC
          0000012F'00000007'        017A  1522          .ADDRESS 7,FAO_AS
          00000136'00000004'        0182  1523          .ADDRESS 4,FAO_OB
          0000013A'00000004'        018A  1524          .ADDRESS 4,FAO_XB
          0000013E'00000004'        0192  1525          .ADDRESS 4,FAO_ZB
          00000142'00000004'        019A  1526          .ADDRESS 4,FAO_UB
          00000146'00000004'        01A2  1527          .ADDRESS 4,FAO_SB
          0000014A'00000004'        01AA  1528          .ADDRESS 4,FAO_OW
          0000014E'00000004'        01B2  1529          .ADDRESS 4,FAO_XW
          00000152'00000004'        01BA  1530          .ADDRESS 4,FAO_ZW
          00000156'00000004'        01C2  1531          .ADDRESS 4,FAO_UW
          0000015A'00000004'        01CA  1532          .ADDRESS 4,FAO_SW
          0000015E'00000004'        01D2  1533          .ADDRESS 4,FAO_OL
          00000162'00000004'        01DA  1534          .ADDRESS 4,FAO_XL
          00000166'00000004'        01E2  1535          .ADDRESS 4,FAO_ZL
          0000016A'00000004'        01EA  1536          .ADDRESS 4,FAO_UL
          0000016E'00000004'        01F2  1537          .ADDRESS 4,FAO_SL
                                    01FA  1538
                                00000A62  1539          .RESTORE
                                    0A62  1540
```

```
                   0098  31  0A62  1541 QHDR:   BRW     DO_QUEUE_HEADER                  ; Branch assist
                             0A65  1542
                             0A65  1543 DO_ONE_COLUMN:
                             0A65  1544
              52   5B   58  C3  0A65  1545         SUBL3   R8, R11, R2                      ; Compute data value address.
                 54   08 A9  90  0A69  1546         MOVB    COLM$B_SRC_FAO(R9), R4           ; Get FAO type for data.
                             0A6D  1547
                             0A6D  1548 PRINT_COLUMN_VALUE::
   53   54   FFFFFF80 8F  CB  0A6D  1549         BICL3   #^XFFFFFF80, R4, R3              ; Strip _NEQ from FAO type.
                   11   53  91  0A75  1550         CMPB    R3, #COLM$K_FAO_Q2               ; Is this a queue header?
                        E8  13  0A78  1551         BEQL    QHDR                             ; Branch if queue header.
                   01   53  91  0A7A  1552         CMPB    R3, #COLM$K_FAO_AS               ; Is data a value?
                        10  14  0A7D  1553         BGTR    40$                              ; Branch if data is value.
                        05  19  0A7F  1554         BLSS    20$                              ; Branch if data is ASCIC.
                   51   62  3C  0A81  1555         MOVZWL  (R2), R1                         ; Must be ASCID, get its size.
                        03  11  0A84  1556         BRB     30$                              ; Go adjust fill size.
                   51   62  9A  0A86  1557 20$:    MOVZBL  (R2), R1                         ; Get ASCIC size.
                   55   51  C2  0A89  1558 30$:    SUBL    R1, R5                           ; Adjust string filler size.
                      003E  31  0A8C  1559         BRW     70$                              ; Go convert the data.
                             0A8F  1560 40$:    DISPATCH R4, type=B, prefix=COLM$K_FAO_, <-
                             0A8F  1561                 <OB_NEQ,41$>, -                  ; Dispatch byte values for
                             0A8F  1562                 <XB_NEQ,41$>, -                  ;  zero test
                             0A8F  1563                 <ZB_NEQ,41$>, -
                             0A8F  1564                 <UB_NEQ,41$>, -
                             0A8F  1565                 <SB_NEQ,41$>, -
                             0A8F  1566                 <OW_NEQ,43$>, -                  ; Dispatch word values for
                             0A8F  1567                 <XW_NEQ,43$>, -                  ;  zero test
                             0A8F  1568                 <ZW_NEQ,43$>, -
                             0A8F  1569                 <UW_NEQ,43$>, -
                             0A8F  1570                 <SW_NEQ,43$>, -
                             0A8F  1571                 <OL_NEQ,45$>, -                  ; Dispatch longword values for
                             0A8F  1572                 <XL_NEQ,45$>, -                  ;  zero test
                             0A8F  1573                 <ZL_NEQ,45$>, -
                             0A8F  1574                 <UL_NEQ,45$>, -
                             CA8F  1575                 <SL_NEQ,45$> -
                             0A8F  1576                 >
                   52   62  D0  0AB2  1577         MOVL    (R2), R2                         ; Everything else doesn't get
                        16  11  0AB5  1578         BRB     70$                              ;  tested.
                   52   62  9A  0AB7  1579 41$:    MOVZBL  (R2), R2                         ; Perform byte sized zero
                        11  12  0ABA  1580         BNEQ    70$                              ;  test.
                        0C  11  0ABC  1581         BRB     49$                              ; Branch if zero.
                   52   62  3C  0ABE  1582 43$:    MOVZWL  (R2), R2                         ; Perform word sized zero
                        0A  12  0AC1  1583         BNEQ    70$                              ;  test.
                        05  11  0AC3  1584         BRB     49$                              ; Branch if zero.
                   52   62  D0  0AC5  1585 45$:    MOVL    (R2), R2                         ; Perform longword zero
                        03  12  0AC8  1586         BNEQ    70$                              ;  test.
                            50  D4  0ACA  1587 49$:    CLRL    R0                               ; For zero, indicate that
                                05  0ACC  1588         RSB                                      ;  entry is to be skipped.
                             0ACD  1589 70$:    $FAO_S  ctrstr = FAO_TABLE[R3], -        ; Convert the data.
                             0ACD  1590                 outbuf = (R7), -
                             0ACD  1591                 outlen = (R7), -
                             0ACD  1592                 p1 = R5, -
                             0ACD  1593                 p2 = R2
                             0AE3  1594         SIGNAL
                            05  0AEF  1595         RSB                                      ; Return
                             0AF0  1596
                             0AF0  1597
```

```
                          0AF0 ;---
                          0AF0 ;    DO_QUEUE_HEADER - Action routine for queue headers
                          0AF0 ;
                          0AF0 ;    This routine tests a doubly linked queue header to see if the queue
                          0AF0 ;    is empty.  If the queue is not empty, the address of the first entry
                          0AF0 ;    in the queue is displayed (PRINT_COLUMNS style).  If the queue is
                          0AF0 ;    empty, the word "empty" is displayed.
                          0AF0 ;
                          0AF0 ;    Inputs:
                          0AF0 ;        R2              address of queue header in local storage
                          0AF0 ;        R5              size of the value section for this item
                          0AF0 ;        R7              address of descriptor for a scratch string
                          0AF0 ;        R8              minus offset in data structure which locates data
                          0AF0 ;        DATSVA(AP)      SVA of real data structure base
                          0AF0 ;
                          0AF0 ;    Outputs:
                          0AF0 ;        see PRINT_COLUMN_VALUE
                          0AF0 ;
                          0AF0 ;    Implicit outputs:
                          0AF0 ;        column entry made in PRINT_COLUMNS table
                          0AF0 ;---
                          0AF0
                          0AF0 QUEUE_EMPTY:
                          0AF0         STRING  <empty>
                          0AFD
                          0AFD DO_QUEUE_HEADER:
          54    0D    90  0AFD         MOVB    #COLM$K_FAO_XL, R4              ; Assume queue is not empty.
  53   08 AC    58    C3  0B00         SUBL3   R8, DATSVA(AP), R3              ; Get SVA of queue header
          53    62    D1  0B05         CMPL    (R2), R3                        ; Is the queue empty?
                07    12  0B08         BNEQ    90$                             ; Branch if not empty.
       52   E3 AF     9E  0B0A         MOVAB   QUEUE_EMPTY, R2                 ; Else, flag queue as empty
          54    01    90  0B0E         MOVB    #COLM$K_FAO_AS, R4              ;  and change output type.
              FF59    31  0B11 90$:    BRW     PRINT_COLUMN_VALUE              ; Go output information.
```

```
                               0B14  1633         .SBTTL  OPEN_OUTPUT -- OPEN THE OUTPUT LISTING FILE
                               0B14  1634 ;---
                               0B14  1635 ;
                               0B14  1636 ;       OPEN_OUTPUT
                               0B14  1637 ;
                               0B14  1638 ;       OPEN THE OUTPUT LISTING FILE AND SETUP TO
                               0B14  1639 ;       BEGIN LISTING OUTPUT.
                               0B14  1640 ;
                               0B14  1641 ;   INPUTS:
                               0B14  1642 ;
                               0B14  1643 ;       OUTPUT_FILE = DESCRIPTOR OF FILE NAME
                               0B14  1644 ;
                               0B14  1645 ;   OUTPUTS:
                               0B14  1646 ;
                               0B14  1647 ;       NONE
                               0B14  1648 ;
                               0B14  1649 ;---
                               0B14  1650
                         007C  0B14  1651         .ENTRY  OPEN_OUTPUT,^M<R2,R3,R4,R5,R6>
                               0B16  1652
      53   000002D7'EF    9E   0B16  1653         MOVAB   LIST,R3                 ; ADDRESS THE RAB
          52     3C A3    D0   0B1D  1654         MOVL    RAB$L_FAB(R3),R2        ; ADDRESS THE FAB
          54     28 A2    D0   0B21  1655         MOVL    FAB$L_NAM(R2),R4        ; ADDRESS THE NAM BLOCK
                               0B25  1656 ;
                               0B25  1657 ;       CLOSE THE PREVIOUS LISTING FILE, IF ANY
                               0B25  1658 ;
                 34 A2    95   0B25  1659         TSTB    FAB$B_FNS(R2)           ; WAS FILE ALREADY OPEN?
                    22    13   0B28  1660         BEQL    20$                     ; BRANCH IF NOT
     00000000'EF    00    FB   0B2A  1661         CALLS   #0,PRINT_INDEX          ; PRINT TABLE OF CONTENTS
                               0B31  1662         $CLOSE  (R2)                    ; CLOSE LISTING FILE
                               0B3A  1663         SIGNAL  RMS,(R2)
                               0B4C  1664 ;
                               0B4C  1665 ;       DETERMINE IF PARAMETER GIVEN IS A TERMINAL OR LISTING DEVICE
                               0B4C  1666 ;
  34 A2    0000000C'EF    90   0B4C  1667 20$:    MOVB    OUTPUT_FILE,FAB$B_FNS(R2)       ; SET FILE NAME
  2C A2    00000010'EF    D0   0B54  1668         MOVL    OUTPUT_FILE+4,FAB$L_FNA(R2)
                               0B5C  1669         ALLOC   NAM$C_MAXRSS,R5         ; ALLOCATE STRING BUFFER
       0C A4     04 A5    D0   0B6E  1670         MOVL    4(R5),NAM$L_ESA(R4)     ; SET ADDRESS OF BUFFER
                               0B73  1671         $PARSE  (R2)                    ; GET EXPANDED FILE NAME
                               0B7C  1672         SIGNAL  RMS,(R2)
          65     0B A4    9A   0B8E  1673         MOVZBL  NAM$B_ESL(R4),(R5)      ; SET LENGTH OF STRING
                               0B92  1674         ALLOC   DIB$C_LENGTH,R6         ; ALLOCATE GETDEV BUFFER
                               0BA4  1675         $GETDEV_S DEVNAM=(R5),PRIBUF=(R6)
                               0BB5  1676         SIGNAL
          51     04 A6    D0   0BC1  1677         MOVL    4(R6),R1                ; ADDRESS THE BUFFER
       42 8F     04 A1    91   0BC5  1678         CMPB    DIB$B_DEVCLASS(R1),#DC$_TERM     ; TERMINAL?
                    6C    13   0BCA  1679         BEQL    50$                     ; BRANCH IF SO
                               0BCC  1680 ;
                               0BCC  1681 ;       OPEN LISTING FILE AND SET PAGE SIZES
                               0BCC  1682 ;
                               0BCC  1683         $CREATE (R2)                    ; OPEN LISTING FILE
                               0BD5  1684         SIGNAL  RMS,(R2)
                               0BE7  1685         $CONNECT (R3)                   ; CONNECT TO OUTPUT STREAM
                               0BF0  1686         SIGNAL  RMS,(R3)
     00000000'GF    00    FB   0C02  1687         CALLS   #0,G^LIB$LP_LINES       ; FIND THE CURRENT PAGE SIZE
00000024'EF    50   06    C3   0C09  1688         SUBL3   #6,R0,PAGE_SIZE         ; INITIALIZE PAGE SIZE
        00000018'EF       D4   0C11  1689         CLRL    PAGE_NUMBER             ; START AT PAGE 1
```

```
         0000001C'EF   D4  0C17  1690          CLRL    LINE_COUNT                  ; NEW PAGE
                           0C1D  1691          $ASCTIM_S TIMBUF=CURRENT_TIME       ; GET CURRENT DATE/TIME
     00000000'EF   00  FB  0C30  1692          CALLS   #0,DUMMY_INDEX              ; PRINT DUMMY TABLE OF CONTENTS
                       04  0C37  1693          RET
                           0C38  1694 ;
                           0C38  1695 ;        OPEN TERMINAL FOR OUTPUT AND SET PAGE SIZES
                           0C38  1696 ;
         0000000C'EF   D4  0C38  1697 50$:     CLRL    OUTPUT_FILE                 ; SIGNAL TO USE TERMINAL
               34 A2   94  0C3E  1698          CLRB    FAB$B_FNS(R2)               ; MARK NO LISTING FILE OPEN
         50    0B A1   9A  0C41  1699          MOVZBL  DIB$L_DEVDEPEND+3(R1),R0           ; GET PAGE SIZE
 00000024'EF   50  03  C3  0C45  1700          SUBL3   #PROMPT_LINES,R0,PAGE_SIZE         ; SET PAGE SIZE
         0000001C'EF   D4  0C4D  1701          CLRL    LINE_COUNT                  ; NEW PAGE
             50    01  D0  0C53  1702          MOVL    #1,R0                       ; SUCCESS
                       04  0C56  1703          RET
```

```
                             0C57             .SBTTL  OPEN_LOG -- OPEN THE LOGGING FILE
                             0C57      ;---
                             0C57      ;
                             0C57      ;      OPEN_LOG
                             0C57      ;
                             0C57      ;      OPEN THE LOGGING FILE AND SETUP TO BEGIN LOGGING.
                             0C57      ;
                             0C57      ; INPUTS:
                             0C57      ;
                             0C57      ;      LOG_FILE = DESCRIPTOR OF FILE NAME
                             0C57      ;
                             0C57      ; OUTPUTS:
                             0C57      ;
                             0C57      ;      NONE
                             0C57      ;
                             0C57      ;---
                             0C57
                       007C  0C57             .ENTRY  OPEN_LOG,^M<R2,R3,R4,R5,R6>
                             0C59
 53   000003CB'EF        9E  0C59             MOVAB   LOGRAB,R3                ; ADDRESS THE RAB
        52   3C A3       D0  0C60             MOVL    RAB$L_FAB(R3),R2         ; ADDRESS THE FAB
        54   28 A2       D0  0C64             MOVL    FAB$L_NAM(R2),R4         ; ADDRESS THE NAM BLOCK
                             0C68      ;
                             0C68      ;      CLOSE THE PREVIOUS LOGGING FILE, IF ANY
                             0C68      ;
             34 A2       95  0C68             TSTB    FAB$B_FNS(R2)            ; WAS FILE ALREADY OPEN?
                1B       13  0C6B             BEQL    20$                      ; BRANCH IF NOT
                             0C6D             $CLOSE  (R2)                     ; CLOSE LOGGING FILE
                             0C76             SIGNAL  RMS,(R2)
                             0C88      ;
                             0C88      ;      DETERMINE IF PARAMETER GIVEN IS A TERMINAL OR LOGGING DEVICF
                             0C88      ;
34 A2   00000004'EF      90  0C88 20$:        MOVB    LOG_FILE,FAB$B_FNS(R2)   ; SET FILE NAME
2C A2   00000008'EF      D0  0C90             MOVL    LOG_FILE+4,FAB$L_FNA(R2)
                             0C98             ALLOC   NAM$C_MAXRSS,R5          ; ALLOCATE STRING BUFFER
     0C A4   04 A5       D0  0CAA             MOVL    4(R5),NAM$L_ESA(R4)      ; SET ADDRESS OF BUFFER
                             0CAF             $PARSE  (R2)                     ; GET EXPANDED FILE NAME
                             0CB8             SIGNAL  RMS,(R2)
        65   0B A4       9A  0CCA             MOVZBL  NAM$B_ESL(R4),(R5)       ; SET LENGTH OF STRING
                             0CCE             ALLOC   DIB$C_LENGTH,R6          ; ALLOCATE GETDEV BUFFER
                             0CE0             $GETDEV_S DEVNAM=(R5),PRIBUF=(R6)
                             0CF1             SIGNAL
        51   04 A6       D0  0CFD             MOVL    4(R6),R1                 ; ADDRESS THE BUFFER
     42 8F   04 A1       91  0D01             CMPB    DIB$B_DEVCLASS(R1),#DC$_TERM    ; TERMINAL?
                37       13  0D06             BEQL    50$                      ; BRANCH IF SO
                             0D08      ;
                             0D08      ;      OPEN LOGGING FILE
                             0D08      ;
                             0D08 30$:        $CREATE (R2)                     ; OPEN LOGGING FILE
                             0D11             SIGNAL  RMS,(R2)
                             0D23             $CONNECT (R3)                    ; CONNECT TO STREAM
                             0D2C             SIGNAL  RMS,(R3)
                         04  0D3E             RET
                             0D3F      ;
                             0D3F      ;      LOGGING AT THE TERMINAL IS NOT ALLOWED SINCE THIS IS BEING DONE ANYWAY
                             0D3F      ;
             34 A2       94  0D3F 50$:        CLRB    FAB$B_FNS(R2)            ; MARK NO LOGGING FILE OPEN
```

```
00000004'EF  D4  0D42  1763         CLRL    LOG_FILE                     ; MARKING NO LOGGING ENABLED
     50  01  D0  0D48  1764         MOVL    #1,R0                        ; SUCCESS
             04  0D4B  1765         RET
```

```
                                   0D4C              .SBTTL  CLOSE_LOG -- CLOSE THE LOGGING FILE
                                   0D4C      ;---
                                   0D4C      ;
                                   0D4C      ;       CLOSE_LOG
                                   0D4C      ;
                                   0D4C      ;       CLOSE THE LOGGING FILE IF ONE IS OPEN.
                                   0D4C      ;
                                   0D4C      ;   INPUTS:
                                   0D4C      ;
                                   0D4C      ;       LOGFAB = LOGGING FAB
                                   0D4C      ;
                                   0D4C      ;   OUTPUTS:
                                   0D4C      ;
                                   0D4C      ;       NONE
                                   0D4C      ;
                                   0D4C      ;---
                                   0D4C
                             0004  0D4C              .ENTRY  CLOSE_LOG,^M<R2>
                                   0D4E
  52    0000037B'EF     9E         0D4E              MOVAB   LOGFAB,R2               ; ADDRESS THE FAB
            50   01     D0         0D55              MOVL    #1,R0
                                   0D58
             34 A2      95         0D58              TSTB    FAB$B_FNS(R2)           ; WAS FILE OPEN?
                24      13         0D5B              BEQL    20$                     ; BRANCH IF NOT
                                   0D5D              $CLOSE  (R2)                    ; CLOSE LOGGING FILE
                                   0D66              SIGNAL  RMS,(R2)
             34 A2      94         0D78              CLRB    FAB$B_FNS(R2)           ; CLEAR INDICATERS THAT THERE
        00000004'EF     D4         0D7B              CLRL    LOG_FILE                ; IS A LOGGING FILE AND THAT LOGGING IS
                        04         0D81      20$:    RET                             ; ENABLED
                0D82  1798
                0D82  1799          .END    START
```

```
$$.TAB                          = 0000067B R       04      COLM$K_FAO_UB_NEQ               = 00000085
$$.TABEND                       = 000006BF R       04      COLM$K_FAO_UL                   = 0000000F
$$.TMP                          = 00000000                 COLM$K_FAO_UL_NEQ               = 0000008F
$$.TMP1                         = 00000001                 COLM$K_FAO_UW                   = 0000000A
$$.TMP2                         = 00000062                 COLM$K_FAO_UW_NEQ               = 0000008A
$$.TMPX                         = 0000004D R       05      COLM$K_FAO_XB                   = 00000003
$$.TMPX1                        = 00000009                 COLM$K_FAO_XB_NEQ               = 00000083
$$BASE                          = 00000082                 COLM$K_FAO_XL                   = 0000000D
$$DISPL                         = 00000091                 COLM$K_FAO_XL_NEQ               = 0000008D
$$GENSW                         = 00000001                 COLM$K_FAO_XW                   = 00000008
$$HIGH                          = 00000090                 COLM$K_FAO_XW_NEQ               = 00000088
$$LIMIT                         = 0000000E                 COLM$K_FAO_ZB                   = 00000004
$$LOW                           = 00000082                 COLM$K_FAO_ZB_NEQ               = 00000084
$$MNSW                          = 00000001                 COLM$K_FAO_ZL                   = 0000000E
$$MXSW                          = 00000001                 COLM$K_FAO_ZL_NEQ               = 0000008E
$$T1                            = 00000001                 COLM$K_FAO_ZW                   = 00000009
$$T2                            = 00000005                 COLM$K_FAO_ZW_NEQ               = 00000089
ALL_DONE                          00000A61 R       06      COLM$K_LENGTH                   = 00000010
BIT...                          = 00000001                 COLM$L_ACTION_VALUE             = 00000008
BUFFER                            00000000 RG      03      COLM$L_SOURCE                   = 00000004
BUG$T_MESSAGES                    ********    X    06      COLM$L_STRING                   = 00000000
BUG$_OPERATOR                     ********    X    06      COLSCRATCH_BASE                   FFFFFFE4
CLI$B_RQSTAT                    = 00000003                 COLUMN_LOOP                       000009D0 R       06
CLI$B_RQTYPE                    = 00000000                 CRASH_ENTITY                      0000000E R       06
CLI$C_REQDESC                   = 0000001C                 CTRLC_PENDING                     0000004C R       02
CLI$GET_VALUE                     ********    X    06      CTRL_C_AST                        00000558 R       06
CLI$K_GETCMD                    = 00000001                 CURPROC                           ********    X    06
CLI$K_VERB_FORE                   ********    X    06      CURRENT_SYSTEM                    00000014 RG      02
CLI$K_VERB_MCR                    ********    X    06      CURRENT_TIME                      0000002C R       02
CLI$PRESENT                       ********    X    06      DATBAS                            00000004
CLOSE_LOG                         00000D4C RG      06      DATSVA                            00000008
CLR_PAGE                          00000028 R       02      DC$_TERM                        = 00000042
CMND_BUFFER                       000009EC RG      03      DEV$V_TRM                       = 00000002
CMND_BUFFER_LEN                 = 00000050  G              DEV_PROMPT                        00000040 R       06
CMND_DESCR                        000009E4 RG      03      DIB$B_DEVCLASS                  = 00000004
COLLS1                            0000000C                 DIB$C_LENGTH                    = 00000074
COLLST_BASE                       FFFFFFE8                 DIB$L_DEVDEPEND                 = 00000008
COLM$B_DESC_SIZE                = 0000000C                 DIR...                          = FFFFFFFF
COLM$B_SEP_SIZE                 = 0000000E                 DMP$L_FLAGS                     = 00000004
COLM$B_SRC_FAO                  = 00000008                 DMP$V_EMPTY                     = 00000001
COLM$B_VAL_SIZE                 = 0000000D                 DMP$V_OLDDUMP                   = 00000000
COLM$K_FAO_AC                   = 00000000                 DO_COLLST_ENTRY                   000009EF R       06
COLM$K_FAO_AS                   = 00000001                 DO_ONE_COLUMN                     00000A65 R       06
COLM$K_FAO_OB                   = 00000002                 DO_QUEUE_HEADER                   00000AFD R       06
COLM$K_FAO_OB_NEQ               = 00000082                 DSC$B_CLASS                     = 00000003
COLM$K_FAO_OL                   = 0000000C                 DSC$K_CLASS_D                   = 00000002
COLM$K_FAO_OL_NEQ               = 0000008C                 DUMMY_INDEX                       ********    X    06
COLM$K_FAO_OW                   = 00000007                 DUMPF                             00000000 RG      04
COLM$K_FAO_OW_NEQ               = 00000087                 DUMPFILE_ENTITY                   00000020 R       06
COLM$K_FAO_Q2                   = 00000011                 DUMPN                             00000050 R       04
COLM$K_FAO_SB                   = 00000006                 DUMPR                             000001AF RG      04
COLM$K_FAO_SB_NEQ               = 00000086                 DUMP_EXPNAME                      000000B0 R       04
COLM$K_FAO_SL                   = 00000010                 DUMP_HEADER                       000002B0 RG      03
COLM$K_FAO_SL_NEQ               = 00000090                 DUMP_HEADER_LEN                 = 00000600  G
COLM$K_FAO_SW                   = 0000000B                 DVI$_DEVBUFSIZ                  = 00000008
COLM$K_FAO_SW_NEQ               = 0000008B                 DVI$_DEVCHAR                    = 00000002
COLM$K_FAO_UB                   = 00000005                 DVI$_DEVDEPEND                  = 0000000A
```

```
DVI$_DEVDEPEND2       = 0000001C                  GET_INPUT               00000530 RG     06
DVI_DEVBUFSIZ           00000AE3 R      03        HANDLER                 ********    X   06
DVI_DEVCHAR             00000AE7 RG     03        HEADING_LINES           00000020 R      02
DVI_DEVDEPEND           00000ADB RG     03        HEADING_ROUTINE         ********    X   06
DVI_DEVDEPND2           00000ADF RG     03        HELP_BUFFER             00000A3C R      03
DVI_ITMLST              00000AA7 R      03        HELP_BUFFER_LEN       = 00000050
DVI_PAGESIZE            00000ADE R      03        INDFAB                  0000046F R      04
EMB$L_CR_CODE         = 000000F4                  INDRAB                  000004BF RG     04
EMB$Q_CR_TIME         = 00000006                  INPUT_BUF               000002A8 RG     03
ERLPTR                  ********    X   06        INPUT_BUFFER            00000200 RG     03
EXIT_IF_OLD             000005B4 RG     06        INPUT_BUF_LEN         = 00000050  G
FAB$B_DNS             = 00000035                  INPUT_LEN               000002A4 RG     03
FAB$B_FNS             = 00000034                  INPUT_RAB               ********    X   06
FAB$C_BID             = 00000003                  IO$M_CTRLCAST           ********    X   06
FAB$C_BLN             = 00000050                  IO$_SETMODE             ********    X   06
FAB$C_FIX             = 00000001                  JPI$_PRCNAM           = 0000031C
FAB$C_SEQ             = 00000000                  KEYBOARD                00000A97 RG     03
FAB$C_VAR             = 00000002                  KEYFAB                  00000503 R      04
FAB$L_ALQ             = 00000010                  KEYRAB                  00000553 RG     04
FAB$L_DNA             = 00000030                  KEYTABLE                00000A9B RG     03
FAB$L_FNA             = 0000002C                  LIB$GET_FOREIGN         ********    X   06
FAB$L_FOP             = 00000004                  LIB$LP_LINES            ********    X   06
FAB$L_NAM             = 00000028                  LIB$SIGNAL              ********    X   06
FAB$L_STV             = 0000000C                  LINES_LOOP              000009C7 R      06
FAB$V_BIO             = 00000005                  LINE_COUNT              0000001C RG     02
FAB$V_CHAN_MODE       = 00000002                  LINE_CTRSTR             FFFFFFF4
FAB$V_CR              = 00000001                  LINE_DESCR              000008B0 RG     03
FAB$V_FILE_MODE       = 00000004                  LIST                    000002D7 RG     04
FAB$V_GET             = 00000001                  LISTF                   00000287 R      04
FAB$V_LNM_MODE        = 00000000                  LISTN                   0000031B R      04
FAB$V_PUT             = 00000000                  LIST_BUFFER             000008B8 R      03
FAB$V_SUP             = 00000002                  LIST_BUFFER_LEN       = 0000012C
FAB$V_UPD             = 00000003                  LOGFAB                  0000037B RG     04
FAB$W_GBC             = 00000048                  LOGNAM                  0000040F R      04
FAO_AC                  00000128 R      07        LOGRAB                  000003CB RG     04
FAO_AS                  0000012F R      07        LOG_FILE                00000004 RG     02
FAO_OB                  00000136 R      07        MAP_DUMP                ********    X   06
FAO_OL                  0000015E R      07        MARK_DUMP               ********    X   06
FAO_OW                  0000014A R      07        MSG$_BACKUP             ********    X   06
FAO_SB                  00000146 R      07        MSG$_EOF                ********    X   06
FAO_SL                  0000016E R      07        MSG$_EXITCMD            ********    X   06
FAO_SW                  0000015A R      07        MSG$_OPENIN             ********    X   06
FAO_TABLE               00000172 R      07        MSG$_SUCCESS            ********    X   06
FAO_UB                  00000142 R      07        NAM$B_ESL             = 0000000B
FAO_UL                  0000016A R      07        NAM$B_ESS             = 0000000A
FAO_UW                  00000156 R      07        NAM$B_NOP             = 00000008
FAO_XB                  0000013A R      07        NAM$B_RSS             = 00000002
FAO_XL                  00000162 R      07        NAM$C_BID             = 00000002
FAO_XW                  0000014E R      07        NAM$C_BLN             = 00000060
FAO_ZB                  0000013E R      07        NAM$C_MAXRSS          = 000000FF
FAO_ZL                  00000166 R      07        NAM$L_ESA             = 0000000C
FAO_ZW                  00000152 R      07        NAM$L_RSA             = 00000004
FLAGS                   FFFFFFE0                  NEW_PAGE                00000707 RG     06
FLAGS_M_NO_ENTRIES    = 00000001                  NEXT_COLUMN             00000A37 R      06
FLAGS_V_NO_ENTRIES    = 00000000                  NULL_ASCIC              00000120 R      07
GET_COMMANDS            ********    X   06        NULL_ASCID              00000120 R      07
GET_DUMP_INFO           ********    X   06        NUMCOL                  FFFFFFF0
```

```
ONE_COL                          0000010C R      07     START                           0000009C RG     06
OPEN_FILES                       000001E4 RG     06     STARTUP                         00000084 R      06
OPEN_LOG                         00000C57 RG     06     STARTUP_LEN                   = 00000007
OPEN_OUTPUT                      00000B14 RG     06     STB                             0000067B RG     04
OUTPUT                           000005E7 RG     04     STBF                            0000062B RG     04
OUTPUTF                          00000597 R      04     STB_BUFFER                    = 00000000 RG     03
OUTPUT_FILE                      0000000C RG     02     STB_BUFFER_LEN                = 00000200
PAGE_NUMBER                      00000018 RG     02     STS$K_SEVERE                  = 00000004
PAGE_SIZE                        00000024 RG     02     STS$S_SEVERITY                = 00000003
PAGE_WAIT                        00000638 RG     06     STS$V_SEVERITY                = 00000000
PC_XIT                           00000937 R      06     SUB_HEADING                     ********    X   06
PRINT                            000007D4 RG     06     SYMBOLS_ENTITY                  00000031 R      06
PRINT_A_LINE                     00000A3A R      06     SYS$ASCTIM                      ********   GX   06
PRINT_COLUMNS                    00000938 RG     06     SYS$ASSIGN                      ********   GX   06
PRINT_COLUMN_VALUE               00000A6D RG     06     SYS$CLI                         ********    X   06
PRINT_INDEX                      ********    X   06     SYS$CLOSE                       ********   GX   06
PROMPT_LINES                   = 00000003               SYS$CONNECT                     ********   GX   06
PUT_BUSY                         0000004D R      02     SYS$CREATE                      ********   GX   06
PUT_LINE                         000007F4 RG     06     SYS$EXIT                        ********   GX   06
QHDR                             00000A62 R      06     SYS$FAO                         ********    X   06
QUEUE_EMPTY                      00000AF0 R      06     SYS$FAOL                        ********    X   06
RAB$B_RAC                      = 0000001E               SYS$GETDEV                      ********   GX   06
RAB$C_BID                      = 00000001               SYS$GETDVI                      ********   GX   06
RAB$C_BLN                      = 00000044               SYS$GETJPI                      ********   GX   06
RAB$C_SEQ                      = 00000000               SYS$OPEN                        ********   GX   06
RAB$L_CTX                      = 00000018               SYS$PARSE                       ********   GX   06
RAB$L_FAB                      = 0000003C               SYS$PUT                         ********   GX   06
RAB$L_RBF                      = 00000028               SYS$QIOW                        ********   GX   06
RAB$L_ROP                      = 00000004               SYS$SYSTEM                      00000071 R      06
RAB$L_STS                      = 00000008               SYSINPUT                        0000008B R      06
RAB$L_STV                      = 0000000C               SYSTEM_ENTITY                   00000000 R      06
RAB$V_BIO                      = 0000000B               TIME_BUFFER                     00000034 R      02
RAB$V_WBH                      = 0000000A               TT$V_SCOPE                      ********    X   06
RAB$W_RSZ                      = 00000022               TT_CHAN                         00000A9F RG     03
READ_SYMBOLS                     ********    X   06     VERSION_FLAGS                   00000000 RG     02
REPEAT_KEY                       00000A8C RG     03
RESTORE_SCRATCH_DESCRIPTORS      00000A4D R      06
RMS$_EOF                         ********    X   06
RMS$_FNF                         ********    X   06
SAVABS...                      = FFFFFFE0
SAVDMP                           00000243 RG     04
SAVDMPF                          000001F3 R      04
SAVE_INPUT_BUFFER                00000250 R      03
SAVE_INPUT_LEN                   000002A0 R      03
SCR$ERASE_PAGE                   ********    X   06
SCR$SET_CURSOR                   ********    X   06
SCRATCH_SIZE                     FFFFFFEC
SDA_PROMPT                       00000063 R      06
SETUP_COL_INFO                   000009B1 R      06
SETUP_COL_SCRATCH                0000097A R      06
SIZ...                         = 00000001
SKIP_LINES                       00000920 RG     06
SMG$CREATE_KEY_TABLE             ********    X   06
SMG$CREATE_VIRTUAL_KEYBOARD      ********    X   06
SMG$READ_COMPOSED_LINE           ********    X   06
SMG_PROMPT                       00000AA3 R      03
STACK_LEN                        FFFFFFE0
```

```
                                                        +----------------+
                                                        ! Psect synopsis !
                                                        +----------------+

PSECT name                          Allocation          PSECT No.  Attributes
----------                          ----------          ---------  ----------
.  ABS  .                           00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
$ABS$                               FFFFFFFC  (    0.)  01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
SDADATA                             0000004E  (   78.)  02 (  2.)  NOPIC   USR   CON   REL   LCL NOSHR NOEXE   RD    WRT NOVEC LONG
BUFFERS                             00000AEB  ( 2795.)  03 (  3.)  NOPIC   USR   CON   REL   LCL NOSHR NOEXE   RD    WRT NOVEC BYTE
RMSBLOCKS                           000006BF  ( 1727.)  04 (  4.)  NOPIC   USR   CON   REL   LCL NOSHR NOEXE   RD    WRT NOVEC LONG
$RMSNAM                             00000056  (   86.)  05 (  5.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
MAIN                                00000D82  ( 3458.)  06 (  6.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD  NOWRT NOVEC LONG
LITERALS                            000001FA  (  506.)  07 (  7.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD  NOWRT NOVEC BYTE

                                                     +------------------------+
                                                     ! Performance indicators !
                                                     +------------------------+

Phase                       Page faults    CPU Time       Elapsed Time
-----                       -----------    --------       ------------
Initialization                       36    00:00:00.06    00:00:01.06
Command processing                  138    00:00:00.48    00:00:02.76
Pass 1                              646    00:00:21.13    00:01:15.00
Symbol table sort                     0    00:00:01.97    00:00:07.61
Pass 2                              327    00:00:04.84    00:00:18.22
Symbol table output                  37    00:00:00.20    00:00:00.64
Psect synopsis output                 3    00:00:00.03    00:00:00.03
Cross-reference output                0    00:00:00.00    00:00:00.00
Assembler run totals               1189    00:00:28.71    00:01:45.33

The working set limit was 2400 pages.
170200 bytes (333 pages) of virtual memory were used to buffer the intermediate code.
There were 100 pages of symbol table space allocated to hold 1921 non-local and 111 local symbols.
1799 source lines were read in Pass 1, producing 70 object records in Pass 2.
74 pages of virtual memory were used to define 65 macros.

                                                    +--------------------------+
                                                    ! Macro library statistics !
                                                    +--------------------------+

Macro library name                                  Macros defined
------------------                                  --------------
_$255$DUA28:[SDA.OBJ]SDALIB.MLB;1                                9
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                                   5
_$255$DUA28:[SYSLIB]STARLET.MLB;2                               45
TOTALS (all libraries)                                          59

2353 GETS were required to define 59 macros.

There were no errors, warnings or information messages.

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

0352 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

HANDLER
LIS

MAPPING
LIS

DUMP
LIS

MAIN
LIS

EXAMPSL
LIS

MMG
LIS

INDEX
LIS

LOCK
LIS

PARSE
LIS