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

FILEID**DEVICE

```
DDDDDDDD    EEEEEEEEEE  VV      VV    IIIIII      CCCCCCCC  EEEEEEEEEE
DDDDDDDD    EEEEEEEEEE  VV      VV    IIIIII      CCCCCCCC  EEEEEEEEEE
DD      DD  EE          VV      VV      II      CC          EE
DD      DD  EE          VV      VV      II      CC          EE
DD      DD  EE          VV      VV      II      CC          EE
DD      DD  EE          VV      VV      II      CC          EE
DD      DD  EEEEEEEE    VV      VV      II      CC          EEEEEEEE
DD      DD  EEEEEEEE    VV      VV      II      CC          EEEEEEEE
DD      DD  EE          VV      VV      II      CC          EE
DD      DD  EE          VV      VV      II      CC          EE
DD      DD  EE            VV  VV        II      CC          EE              ....
DD      DD  EE            VV  VV        II      CC          EE              ....
DDDDDDDD    EEEEEEEEEE      VV        IIIIII      CCCCCCCC  EEEEEEEEEE      ....
DDDDDDDD    EEEEEEEEEE      VV        IIIIII      CCCCCCCC  EEEEEEEEEE      ....

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

```
0000     29          .sbttl  Program description
0000     30 ;++
0000     31 ;   Facility
0000     32 ;
0000     33 ;       System Dump Analyzer
0000     34 ;
0000     35 ;   Abstract
0000     36 ;
0000     37 ;       This module contains routines to print device data
0000     38 ;       structures for the i/o subsystem.
0000     39 ;
0000     40 ;   Environment
0000     41 ;
0000     42 ;       Native mode, User mode
0000     43 ;
0000     44 ;   Author
0000     45 ;
0000     46 ;       Tim Halvorsen, July 1978
0000     47 ;
0000     48 ;   Modified by
0000     49 ;
0000     50 ;       V03-011 EMB0110         Ellen M. Batbouta       24-Jul-1984
0000     51 ;               Fix a typo in the SHOW DEVICE display and update the
0000     52 ;               list of devices and device characteristics.
0000     53 ;
0000     54 ;       V03-010 EMB0105         Ellen M. Batbouta       07-Jun-1984
0000     55 ;               Add routines to display the contents of the class
0000     56 ;               driver data blocks (CDDB) when displaying an mscp
0000     57 ;               served device. Also for mscp served devices check
0000     58 ;               2 additional queues before drawing the conclusion
0000     59 ;               that the io request queue is empty.  Fix a minor
0000     60 ;               bug and include the node name in the display in
0000     61 ;               the routine, SHOW_SYSTEM_BLOCK.
0000     62 ;
0000     63 ;       V03-009 EMD0082         Ellen M. Dusseault      12-Apr-1984
0000     64 ;               Print the address of the cddb and the alternate cddb
0000     65 ;               (if the device is mscp served) when displaying the ucb tables
0000     66 ;               and action routines .  Also display the reasons to wait
0000     67 ;               count for mscp served devices.
0000     68 ;
0000     69 ;       V03-008 LMP0221         L. Mark Pilant,         30-Mar-1984  11:53
0000     70 ;               Change UCB$L_OWNUIC to ORB$L_OWNER and UCB$W_VPROT to
0000     71 ;               ORB$W_PROT.
0000     72 ;
0000     73 ;       V03-007 EMD0059         Ellen M. Dusseault      07-Mar-1984
0000     74 ;               Fill in local ucb with zeroes in routine, GET_UCB,
0000     75 ;               just in case next ucb fetched is shorter than the
0000     76 ;               previous one.
0000     77 ;
0000     78 ;       V03-006 WHM0002         Bill Matthews           16 Feb-1984
0000     79 ;               Change IDB$B_COMBO_VECTOR back to IDB$B_VECTOR.
0000     80 ;
0000     81 ;       V03-005 TMK0002         Todd M. Katz            29-Jan-1984
0000     82 ;               Add DT$_NI to the table BUS_TYPE.
0000     83 ;
0000     84 ;       V03-004 WHM0001         Bill Matthews           16-Jan-1984
0000     85 ;               Change IDB$B_VECTOR to IDB$B_COMBO_VECTOR.
```

```
0000     86 ;
0000     87 ;       V03-003 TMK0001         Todd M. Katz            19-Nov-1983
0000     88 ;               Change DT$_UNA11 to DT$_DEUNA in the table SCOM_TYPE and
0000     89 ;               add DT$_DELUA to the same table.
0000     90 ;
0000     91 ;       V03-002 ROW0237         Ralph O. Weber          10-OCT-1983
0000     92 ;               Enhance all displays for latest and greatest I/O database
0000     93 ;               information.  Add support for SHOW DEVICE/ADDR <expr>, where
0000     94 ;               expression is a UCB address.
0000     95 ;
0000     96 ;       V03-001 KTA3041         Kerbey T. Altmann       26-Apr-1983
0000     97 ;               Fix for cluster names.
0000     98 ;--
```

```
                                J 11
0000    100         .sbttl  declarations
0000    101 ;
0000    102 ;       symbol defintions
0000    103 ;
0000    104         $adpdef                 ; Adapter Control Block (ADP)
0000    105         $aqbdef                 ; ACP queue header block (AQB)
0000    106         $cddbdef                ; Class Driver Data Block (CDDB)
0000    107         $cdrpdef                ; Class Driver Request Packet (CDRP)
0000    108         $crbdef                 ; channel request block (CRB)
0000    109         $dcdef                  ; device class/type definitions
0000    110         $ddbdef                 ; device data block (DDB)
0000    111         $ddtdef                 ; Driver dispatch table (DDT)
0000    112         $devdef                 ; Device characteristics definitions
0000    113         $dptdef                 ; Driver prologue table (DPT)
0000    114         $dyndef                 ; Dynamic storage type definitions
0000    115         $idbdef                 ; interrupt dispatch block (IDB)
0000    116         $iodef                  ; I/O function codes
0000    117         $irpdef                 ; I/O request package (IRP)
0000    118         $mscpdef                ; Mass Storage Control Protocol (MSCP)
0000    119         $orbdef                 ; Object's Rights Block (ORB)
0000    120         $pbdef                  ; path block (PB)
0000    121         $pcbdef                 ; Process control block (PCB)
0000    122         $sbdef                  ; System block (SB)
0000    123         $tpadef                 ; TPARSE definitions
0000    124         $ttyucbdef              ; terminal UCB definitions
0000    125         $ucbdef                 ; unit control block (UCB)
0000    126         $vcbdef                 ; Volume control block (VCB)
0000    127         $vecdef                 ; interrupt transfer vector (in IDB)
0000    128
0000    129 ;
0000    130 ; definition of requested device name storage fields
0000    131 ;  (using storage based at parsed_devnam)
0000    132 ;
0000    133         $defini pdvnm
0000    134 $def    pdvnm_t_node    .blkb 16        ; node name
0010    135 $def    pdvnm_t_ddc     .blkb 16        ; device & controller
0020    136 $def    pdvnm_w_unit    .blkw 1         ; unit number
0022    137 $def    pdvnm_b_nodesz  .blkb 1         ; size of real node name
0023    138                                         ;  (use by get_ddb)
00000024 0023   139         .blkb 1
00000024 0024   140 pdvnm_k_length = .              ; size of this structure
0024    141         $defend pdvnm
0000    142
0000    143 ;
0000    144 ; definition of flags bits stored in r8 by display_device
0000    145 ;
0000    146         _vield  flag,0,< -
0000    147                 <one_unit,,m>, - ; a specific unit was specified
0000    148                 <alt_path,,m>, - ; traversing the alternate DDB chain
0000    149                 <fnd_unit,,m>, - ; found at least one unit
0000    150                 >
```

```
                     0000   152          .sbttl  storage definitions
                     0000   153 ;
                     0000   154 ;       storage definitions
                     0000   155 ;
                     0000   156
                 00000000   157          .psect  sdadata,noexe,wrt
                     0000   158
000000CC             0000   159 ucb_size = ucb$k_lcl_disk_length
                     0000   160 .iif gt <ucb$l_2p_cddb+4-ucb_size>, ucb_size = ucb$l_2p_cddb+4
                     0000   161
00000060             0000   162 sb:      .blkb   sb$k_length     ; System block (SB)
                     0060   163 nodnam_2p:
00000071             0060   164          .blkb   sb$s_nodename+1
                     0071   165
000000B5             0071   166 ddb:     .blkb   ddb$k_length    ; device data block (DDB)
000000F9             00B5   167 ddb_2p:  .blkb   ddb$k_length    ; secondary device data block (DDB)
                     00F9   168
000001C5             00F9   169 ucb:     .blkb   ucb_size        ; unit control block (UCB)
                     01C5   170                                  ;  all the interesting stuff
                     01C5   171
00000289             01C5   172 irp:     .blkb   irp$c_length    ; I/O request package (IRP)
                     0289   173
00000331             0289   174 cdrp:    .blkb   cdrp$c_cd_len-cdrp$l_ioqfl      ; Class Driver Request Package (CDRP
000000A8             0331   175 cdrp_length=cdrp$c_cd_len-cdrp$l_ioqfl  ; Total length of cdrp including negative of
                     0331   176
0000041D             0331   177 vcb:     .blkb   vcb$c_length    ; Volume control block (VCB)
                     041D   178
00000439             041D   179 aqb:     .blkb   aqb$c_length    ; ACP queue header block (AQB)
                     0439   180
00000471             0439   181 dpt:     .blkb   dpt$c_length    ; Driver prologue table (DPT)
                     0471   182
000004E1             0471   183 cddb:    .blkb   cddb$k_length   ; Class driver data block (CDDB)
                     04E1   184
00000551             04E1   185 cddb_2p:   .blkb  cddb$k_length ; Secondary CDDB
                     0551   186
                     0551   187 parsed_devnam:
00000575             0551   188          .blkb   pdvnm_k_length
                     0575   189
                     0575   190 flag_2nd_cddb:
0000                 0575   191          .word 0                 ; flag to tell us if the address coming in is the
                     0577   192                                  ; primary or secondary cddb in routine, show_cddb
                     0577   193 queue_notempty:
00                   0577   194          .byte 0                 ; if 1 means item in an io queue to be displayed
                     0578   195                                  ; if 0 the queue is empty
                     0578   196
                 00000000   197          .psect  device,exe,nowrt,long
                     0000   198
                     0000   199          .default displacement,long
```

```
0000    201          .sbttl  read-only data definitions
0000    202
0000    203 ;
0000    204 ;       read-only data definitions
0000    205 ;
0000    206
0000    207 pb_status:
0000    208          table   pb$v_,<tim>
0010    209
0010    210 pb_state:
0010    211          table   pb$c_,<CLOSED,ST_SENT,ST_REC,OPEN>
0038    212
0038    213 pb_rstate:
0038    214          table   pb$c_,<UNINIT,DISAB,ENAB>
0058    215
0058    216 pb_rport_type:
0058    217          table   pb$c_,<CI780,HSC,KL10,CINT,NI,PS>
0090    218
0090    219 ddb_acpclass:
0090    220          table   ddb$k_,<PACK,CART,SLOW,TAPE>
00B8    221
00B8    222 unit_status:
00B8    223          table   ucb$v_,<tim,int,erlogip,cancel,online,power,timout,-
00B8    224                  inttype,bsy,mounting,deadmo,valid,unload,template,-
00B8    225                  mntverip,wrongvol,deleteucb,lcl_valid,supmvmsg,-
00B8    226                  mntverpnd>
0160    227
0160    228 device_char:
0160    229          table   dev$v_,<rec,ccl,trm,dir,sdi,sqd,spl,opr,rct,net,fod,-
0160    230                  dua,shr,gen,avl,mnt,mbx,dmt,elg,all,for,swl,idv,odv,-
0160    231                  rnd,rtm,rck,wck>
0248    232
0248    233 device_char_2:
0248    234          table   dev$v_,<clu,det,rtt,cdp,2p,mscp,ssm,srv,red,nnm>
02A0    235
02A0    236 device_class:
02A0    237          addr_table dc$_,<-
02A0    238                  <disk,disk_type>,-
02A0    239                  <tape,tape_type>,-
02A0    240                  <scom,scom_type>,-
02A0    241                  <card,card_type>,-
02A0    242                  <term,term_type>,-
02A0    243                  <lp,lp_type>,-
02A0    244                  <workstation,workstation_type>,-
02A0    245                  <realtime,realtime_type>,-
02A0    246                  <bus,bus_type>,-
02A0    247                  <mailbox,mailbox_type>,-
02A0    248                  <journal,journal_type>,-
02A0    249                  <misc,misc_type>-
02A0    250                  >
0308    251
0308    252 disk_type:
0308    253          table   dt$_,<RK06,RK07,RP04,RP05,RP06,RM03,RP07,RP07HT,RL01,RL02,-
0308    254                  RX02,RX04,RM80,TU58,RM05,RX01,ML11,RB02,RB80,RA80,RA81,RA60,-
0308    255                  RZ01,RC25,RZF01,RCF25,RD51,RX50,RD52,RD53,RD26,RA82,RC26,-
0308    256                  RCF26,CRX50>
0428    257
```

```
0428    258 tape_type:
0428    259         table   dt$_,<TE16,TU45,TU77,TS11,TU78,TA78,TU80,TU81,TA81,TK50>
0480    260
0480    261 scom_type:
0480    262         table   dt$_,<DMC11,DMR11,XK_3271,XJ_2780,NW_X25,NV_X29,SB_ISB11,-
0480    263                 MX_MUX200,DMP11,DMF32,XV_327T,CI,NI,DEUNA,YN_X25,YO_X25,-
0480    264                 YP_ADCCP,YQ_3271,YR_DDCMP,YS_SDLC,UK_KTC32,DEQNA,DMV11,DELUA>
0548    265
0548    266 card_type:
0548    267         table   dt$_,<CR11>
0558    268
0558    269 term_type:
0558    270         table   dt$_,<TTYUNKN,VT05,FT1,FT2,FT3,FT4,FT5,FT6,FT7,FT8,LAX,-
0558    271                 LA36,LA120,VT5X,VT52,VT55,TQ_BTS,TEK401X,VT100,VK100,-
0558    272                 VT173,LA34,LA38,LA12,LA24,LQP02,VT101,VT102,VT105,VT125,-
0558    273                 VT131,VT132,DZ11,DZ32,DZ730,DMZ32,DHV,DHU>
0690    274
0690    275 lp_type:
0690    276         table   dt$_,<LP11,LA11,LA180>
06B0    277
06B0    278 workstation_type:
06B0    279         table   dt$_,<VS100,VS125,VS300>
06D0    280
06D0    281 realtime_type:
06D0    282         table   dt$_,<LPA11,DR780,DR750,DR11W,PCL11R,PCL11T,DR11C,XI_DR11C,-
06D0    283                 XP_PCL11B,IX_IEX11>
0728    284
0728    285 bus_type:
0728    286         table   dt$_,<CI780,CI750,UQPORT,UDA50,UDA50A,LESI,TU81P,RDRX,NI>
0778    287
0778    288 mailbox_type:
0778    289         table   dt$_,<MBX,SHRMBX,NULL>
0798    290
0798    291 journal_type:
0798    292         table   dt$_,<RUJNL,BIJNL,AIJNL,ATJNL,CLJNL>
07C8    293
07C8    294 misc_type:
07C8    295         table   dt$_,<DN11>
07D8    296
07D8    297 vcb_disk_status:
07D8    298         table   vcb$v_,<write_if,write_sm,homblkbad,idxhdrbad,noalloc,-
07D8    299                 extfid,group,system>
0820    300
0820    301 vcb_disk_status2:
0820    302         table   vcb$v_,<writethru,nocache,mountver,erase,nohighwater>
0850    303
0850    304 vcb_tape_status:
0850    305         table   vcb$v_,<partfile,logiceovs,waimouvol,wairewind,waiusrlbl,-
0850    306                 cancelio,mustclose,nowrite>
0898    307
0898    308 vcb_tape_mode:
0898    309         table   vcb$v_,<ovrexp,ovracc,ovrlbl,ovrsetid,intchg,ebcdic,novol2,-
0898    310                 starfile,enusereot,blank,init,noauto,ovrvolo>
0908    311
0908    312 vcb_journal_char:
0908    313         table   vcb$v_,<jnl_disk,jnl_tape,jnl_tmpfi>
0928    314
```

```
0928   315 cddb_status:
0928   316         table   cddb$v_,<snglstrm,impend,initing,reconnect,resynch,polling,-
0928   317                 alcls_set,noconn,rstrtwait,quorlost,dapbsy,2pbsy>
0990   318
0990   319 cddb_flags:
0990   320         table   mscp$v_,<cf_576,cf_shadw,cf_mlths,cf_this,cf_other,cf_misc,-
0990   321                 cf_attn,cf_replc>
09D8   322
09D8   323 cdrp_dutuflags:
09D8   324         table   cdrp$v_,<cand,canio,erlip,perm,hirt,ivcmd>
0A10   325
0A10   326 request_status:
0A10   327         table   irp$v_,<bufio,func,pagio,complx,virtual,chained,swapio,-
0A10   328                 diagbuf,physio,termio,mbxio,extend,filacp,mvirp>
0A88   329
0A88   330 io_function:
0A88   331         table   io$_,<nop,unload,seek,recal,erasetape,packack,spacerecord,-
0A88   332                 writecheck,writepblk,readpblk,available,dse,setchar,sensechar,-
0A88   333                 writemark,wrttmkr,writelblk,readlblk,rewindoff,setmode,rewind,-
0A88   334                 skipfile,skiprecord,sensemode,writeof,writevblk,readvblk,-
0A88   335                 access,create,deaccess,delete,modify,acpcontrol>
0B98   336
0B98   337 acp_status:
0B98   338         table   aqb$v_,<unique,defclass,defsys,creating>
0BC0   339
0BC0   340 aqb_acptype:
0BC0   341         table   aqb$k_,<undefined,f11v1,f11v2,mta,net,rem,jnl>
0C00   342
0C00   343
```

```
                                0C00    345          .sbttl  display_devbyaddr -- display UCB, etc. given its address
                                0C00    346 ;---
                                0C00    347 ;
                                0C00    348 ;       display_devbyaddr
                                0C00    349 ;
                                0C00    350 ;       This routine takes the address value in TPA$L_NUMBER(AP),
                                0C00    351 ;       attempt to use it as a UCB address, and do a SHOW DEVICE
                                0C00    352 ;       for that UCB.  This is the primary support routine for
                                0C00    353 ;       the SHOW DEVICE/ADDR command.
                                0C00    354 ;
                                0C00    355 ;   Inputs:
                                0C00    356 ;
                                0C00    357 ;       AP = pointer to TPARSE block
                                0C00    358 ;
                                0C00    359 ;   Outputs:
                                0C00    360 ;
                                0C00    361 ;       The i/o data structures for that device are shown.
                                0C00    362 ;
                                0C00    363 ;---
                                0C00    364
                                0C00    365          .enable lsb
                                0C00    366
                        0DFC    0C00    367          .entry  display_devbyaddr, -
                                0C02    368                  ^m<r2,r3,r4,r5,r6,r7,r8,r8,r10,r11>
                                0C02    369
                                0C02    370          subhd   <I/O data structures>
    57  000000F9'EF    9E   0C0F    371          movab   ucb, r7                         ; get local UCB home
        52    1C AC    D0   0C16    372          movl    tpa$l_number(ap), r2            ; get supposed UCB address
                136C   30   0C1A    373          bsbw    get_ucb                         ; pull UCB to local memory
              06 50    E9   0C1D    374          blbc    r0, 900$                        ; if error, exit
         0A A7   10    91   0C20    375          cmpb    #dyn$c_ucb, ucb$b_type(r7)      ; is it really a UCB?
                 4E    13   0C24    376          beql    10$                             ; branch if really a UCB
              1C AC    DD   0C26    377 900$:    pushl   tpa$l_number(ap)                ; else, output a error
                       0C29    378          type    1,<!XL is not the address of a UCB>
                 006D  31   0C71    379          brw     999$                            ; then exit
                       0C74    380
    56  00000071'EF    9E   0C74    381 10$:     movab   ddb, r6                         ; get local DDB home
                       0C7B    382          trymem  @ucb$l_ddb(r7), (r6), #ddb$k_length ; copy the DDB
              96 50    E9   0C8D    383 910$:    blbc    r0, 900$                        ; quit now, if error
         0A A6   06    91   0C90    384          cmpb    #dyn$c_ddb, ddb$b_type(r6)      ; is this a DDB?
                 90    12   0C94    385 911$:    bneq    900$                            ; branch if not a DDB
    5B  00000000'EF    9E   0C96    386          movab   sb, r11                         ; get local SB home
                       0C9D    387          trymem  @ddb$l_sb(r6), (r11), #sb$k_length ; copy the SB
              DB 50    E9   0CAF    388          blbc    r0, 910$                        ; if error, exit
     0A AB    0760 8F    B1   0CB2    389          cmpw    #<dyn$c_scs_sb@8+dyn$c_scs>, -  ; is this really a SB?
                       0CB8    390                  sb$b_type(r11)
                 DA    12   0CB8    391          bneq    911$                            ; branch if no really a SB
                       0CBA    392
       10 38 A7   0E    E1   0CBA    393          bbc     #dev$v_fod, ucb$l_devchar(r7), - ; branch if this device not
                       0CBF    394                  27$                             ;   file oriented?
          50   44 AB    9A   0CBF    395          movzbl  sb$t_nodename(r11), r0          ; else, get node name size
                 0D    13   0CC3    396          beql    30$                             ; branch if no node name
        45 AB40   24    90   0CC5    397          movb    #^a/$/, -                       ; add "$" to node  name
                       0CCA    398                  sb$t_nodename+1(r11)[r0]
              44 AB    96   0CCA    399          incb    sb$t_nodename(r11)              ; increase size of node name
                 03    11   0CCD    400          brb     30$
              44 AB    94   0CCF    401 27$:     clrb    sb$t_nodename(r11)              ; non-fod devices have no node
```

```
                        0CD2    402
                00      DD  0CD2    403 30$:    pushl   #0                          ; setup no flags flags longword
              44 AB     9F  0CD4    404         pushab  sb$t_nodename(r11)          ; setup node name
                52      DD  0CD7    405         pushl   r2                          ; setup UCB VA
             7E 56      7D  0CD9    406         movq    r6, -(sp)                   ; setup local DDB and UCB
1C7B'CF      05      FB  0CDC    407         calls   #5, w^show_ucb              ; display this UCB
                        0CE1    408
                04      0CE1    409 999$:   ret
                        0CE2    410
                        0CE2    411         .disable lsb
```

```
                        0CE2   413          .sbttl  display_device -- display i/o data structures
                        0CE2   414 ;---
                        0CE2   415 ;
                        0CE2   416 ;       display_device
                        0CE2   417 ;
                        0CE2   418 ;       This routine displays all i/o data structures related
                        0CE2   419 ;       to a specified generic device name.
                        0CE2   420 ;
                        0CE2   421 ;   Inputs:
                        0CE2   422 ;
                        0CE2   423 ;       AP = pointer to TPARSE block
                        0CE2   424 ;
                        0CE2   425 ;   Outputs:
                        0CE2   426 ;
                        0CE2   427 ;       The i/o data structures for that device are shown.
                        0CE2   428 ;
                        0CE2   429 ;---
                        0CE2   430          .enabl  lsb
                        0CE2   431
                        0CE2   432 display_device::
                0FFC    0CE2   433          .word   ^m<r2,r3,r4,r5,r6,r7,r8,r9,r10,r11>
                        0CE4   434
                  58    D4    0CE4   435          clrl    r8                      ; init internal flags
                00B4    30    0CE6   436          bsbw    parse_device            ; parse the device into name and unit
                        0CE9   437
                        0CE9   438          subhd   <I/O data structures>
                        0CF6   439
                        0CF6   440          assume  flag_v_one_unit eq 0
              05 58     E8    0CF6   441          blbs    r8, 10$                 ; if explicit unit, skip ddb info
        0DFD'CF   6C    FA    0CF9   442          callg   (ap),w^show_ddbs        ; show DDB summary
                        0CFE   443
                        0CFE   444          ; init iodb scan
                  5B    D4    0CFE   445 10$:     clrl    r11                     ; make get_ddb initialize
                        0D00   446
                        0D00   447          ; loop over all DDBs and both paths
                0202    30    0D00   448 20$:     bsbw    get_ddb                 ; get the next DDB
              3C 50     E9    0D03   449          blbc    r0,45$                  ; leave when done
    57  000000F9'EF     9E    0D06   450          movab   ucb, r7                 ; address UCB in local storage
                58  02  CA    0D0D   451          bicl    #flag_m_alt_path, r8    ; assume not alternate path, yet
              52  04 A6    D0    0D10   452          movl    ddb$l_ucb(r6), r2       ; Address of first UCB
                    06    13    0D14   453          beql    30$                     ; Branch if none
                1270    30    0D16   454          bsbw    get_ucb                 ; Read first UCB
              14 50     E8    0D19   455          blbs    r0, 40$                 ; If got something, go process it
              52  40 A6    D0    0D1C   456 30$:     movl    ddb$l_dp_ucb(r6), r2    ; try looking at the alternate path
                    DE    13    0D20   457          beql    20$                     ; branch if nothing there
                1264    30    0D22   458          bsbw    get_ucb                 ; read first alternate pathed UCB
              F4 50     E9    0D25   459          blbc    r0, 30$                 ; if nothing there, skip this DDB
                58  02  C8    0D28   460          bisl    #flag_m_alt_path, r8    ; now doing the alternate path
                  04 A6    D5    0D2B   461          tstl    ddb$l_ucb(r6)           ; was anything found on primary path?
                    30    12    0D2E   462          bneq    60$                     ; if so, skip the controller info
                        0D30   463
                        0D30   464          ; display controller information if appropriate
                        0D30   465          assume  flag_v_one_unit eq 0
              2D 58     E8    0D30   466 40$:     blbs    r8, 60$                 ; if explicit unit, skip controler info
                  59    DD    0D33   467          pushl   r9                      ; SVA of DDB
                44 AB    9F    0D35   468          pushab  sb$t_nodename(r11)      ; address of nodename
              7E  56    7D    0D38   469          movq    r6,-(sp)                ; address of DDB,UCB blocks
```

```
                OFE1'CF    03   FB  0D3B    470          calls   #3,w^show_controller    ; Display controller info
                           28   11  0D40    471          brb     70$                     ; ...enter loop
                                    0D42    472
                                    0D42    473 ; Intermediate branch to final cleanup/error processing.
                                    0D42    474
                           3A   11  0D42    475 45$:     brb     100$
                                    0D44    476
                                    0D44    477          ; loop over all UCBs on a either DDB chain
                  09 58    01   E1  0D44    478 50$:     bbc     #flag_v_alt_path, r8, - ; branch if using primary chain
                                    0D48    479                  53$
                52   00A4  C7   D0  0D48    480          movl    ucb$l_dp_link(r7), r2   ; else, addr. of next UCB on sec. chain
                           B1   13  0D4D    481          beql    20$                     ; branch if no more
                           06   11  0D4F    482          brb     55$                     ; else, continue processing
                  52   30  A7   D0  0D51    483 53$:     movl    ucb$l_link(r7), r2      ; address of next UCB in primary chain
                           C5   13  0D55    484          beql    30$                     ; branch if no more
                         122F   30  0D57    485 55$:     bsbw    get_ucb                 ; Get local copy of the UCB
                        BF 50   E9  0D5A    486          blbc    r0, 30$                 ; skip rest if chain broken
                                    0D5D    487          assume  flag_v_one_unit eq 0
                        0A 58   E9  0D5D    488          blbc    r8, 70$                 ; branch if displaying all units
                                    0D60    489
00000571'EF     54   A7   B1   0D60    490 60$:     cmpw    ucb$w_unit(r7), -       ; check if request unit
                                    0D68    491                  parsed_devnam+pdvnm_w_unit
                           DA   12  0D68    492          bneq    50$                     ; skip if not
                                    0D6A    493
                           58   DD  0D6A    494 70$:     pushl   r8                      ; flags longword
                        44 AB   9F  0D6C    495          pushab  sb$t_nodename(r11)      ; address of node name
                           52   DD  0D6F    496          pushl   r2                      ; actual address of UCB
                     7E    56   7D  0D71    497          movq    r6,-(sp)                ; address of DDB,UCB blocks
                1C7B'CF    05   FB  0D74    498          calls   #5,w^show_ucb           ; display current UCB
                     58    04   C8  0D79    499          bisl    #flag_m_fnd_unit, r8    ; mark at least 1 UCB was displayed
                           C6   11  0D7C    500          brb     50$                     ; loop thru all UCB's
                                    0D7E    501
                     58    02   E0  0D7E    502 100$:    bbs     #flag_v_fnd_unit, -     ; branch if at least 1 ucb displayed
                           13       0D81    503                  r8, 110$
           50   0000'8F    3C   0D82    504          movzwl  #ss$_nosuchdev,r0       ; signal "no such device"
                                    0D87    505          signal  0
                                    0D95    506 110$:    status  success                 ; exit to tparse w/success
                           04   0D9C    507          ret
                                    0D9D    508
                                    0D9D    509          .dsabl  lsb
```

```
                     0D9D   511          .sbttl  parse_device -- parse device name into name and unit number
                     0D9D   512 ;---
                     0D9D   513 ;         parse the device name into name and unit number
                     0D9D   514 ;
                     0D9D   515 ; Inputs:
                     0D9D   516 ;
                     0D9D   517 ;         r8 = longword of show command status flags
                     0D9D   518 ;         tpa$l_tokencnt(ap) = Descriptor of device name
                     0D9D   519 ;         parsed_devnam = address of a work area into which parsed fragments
                     0D9D   520 ;                         of the device name are stored
                     0D9D   521 ;
                     0D9D   522 ; Outputs:
                     0D9D   523 ;
                     0D9D   524 ;         if x equals parsed_devnam then:
                     0D9D   525 ;             pdvnm_t_node(x) = ASCIC string for parsed node name
                     0D9D   526 ;             pdvnm_t_ddc(x)  = ASCIC string for parsed device and controller
                     0D9D   527 ;             pdvnm_s_unit(x) = converted unit number
                     0D9D   528 ;             (null strings imply item missing from input)
                     0D9D   529 ;         flag_m_one_unit in r8, set if unit number specified
                     0D9D   530 ;         r2-r7 and r9-r11 are destroyed.
                     0D9D   531 ;---
                     0D9D   532
                     0D9D   533 parse_device:
5B   00000551'EF  9E 0D9D   534          movab   parsed_devnam, r11      ; get working area base address
           6B  D4 0DA4   535          clrl    pdvnm_t_node(r11)       ; null the two string values
         10 AB  D4 0DA6   536          clrl    pdvnm_t_ddc(r11)
         20 AB  B4 0DA9   537          clrw    pdvnm_w_unit(r11)       ; zero unit number
      56 10 AC  7D 0DAC   538          movq    tpa$l_tokencnt(ap), r6  ; get descriptor of input string
   67 56   24  3A 0DB0   539          locc    #^a/$/, r6, (r7)        ; scan name for a "$"
           14  13 0DB4   540          beql    10$                     ; branch if none
   59 51   57  C3 0DB6   541          subl3   r7, r1, r9              ; compute size of node name
01 AB 67   59  28 0DBA   542          movc3   r9, (r7), -             ; copy node name string to work area
                0DBF   543                  pdvnm_t_node+1(r11)
      6B   59  90 0DBF   544          movb    r9, pdvnm_t_node(r11)   ; store node name size
           59  D6 0DC2   545          incl    r9                      ; get size of node name incl. "$"
           56  59 C2 0DC4   546          subl    r9, r6                  ; adjust input string descriptor to
           57  59 C0 0DC7   547          addl    r9, r7                  ;  remove node name section
               56 D5 0DCA   548 10$:     tstl    r6                      ; anything left to work with?
               2E 13 0DCC   549          beql    90$                     ; branch if no characters left
   50  67      30 83 0DCE   550 20$:     subb3   #^a/0/, (r7), r0        ; convert next character to a
               12 19 0DD2   551          blss    50$                     ;  a numeric value and branch to
           09  50 91 0DD4   552          cmpb    r0, #9                  ;  50$ if not a numeric digit
               0D 1A 0DD7   553          bgtru   50$
     20 AB     0A A4 0DD9   554          mulw    #10, pdvnm_w_unit(r11)  ; scale unit number by ten
     20 AB     50 A0 0DDD   555          addw    r0, pdvnm_w_unit(r11)   ;  and add new digit
        58     01 C8 0DE1   556          bisl    #flag_m_one_unit, r8    ; set the unit number found flag
               11 11 0DE4   557          brb     66$                     ; go do next digit
                     0DE6   558 50$:     assume  flag_v_one_unit eq 0
         13 58     E8 0DE6   559          blbs    r8, 90$                 ; branch if unit number already found
   50    10 AB     9A 0DE9   560          movzbl  pdvnm_t_ddc(r11), r0    ; get number of characters in dev/ctrl
11 AB40    67     90 0DED   561          movb    (r7), -                 ; move new character into place
                     0DF2   562                  pdvnm_t_ddc+1(r11)[r0]
10 AB   50  01     81 0DF2   563          addb3   #1, r0, pdvnm_t_ddc(r11) ; store new character count
               57  D6 0DF7   564 66$:     incl    r7                      ; move string pointer
         D2 56     F5 0DF9   565          sobgtr  r6, 20$                 ; reduce character count and branch
                     0DFC   566                                          ;  if characters still left to process
                     05 0DFC   567 90$:     rsb
```

```
                        ;DFD   569          .sbttl  show_ddbs -- display device data blocks (DDBs)
                        ;DFD   570 ;---
                        ;DFD   571 ;
                        ;DFD   572 ;       show_ddbs
                        ;DFD   573 ;
                        ;DFD   574 ;       This routine displays all active DDB's associated
                        ;DFD   575 ;       with a specified generic device name.
                        ;DFD   576 ;
                        ;DFD   577 ;   Inputs:
                        ;DFD   578 ;
                        ;DFD   579 ;       AP = pointer to TPARSE block
                        ;DFD   580 ;
                        ;DFD   581 ;---
                        ;DFD   582          .save
                000008D2;      583          .psect literals
                        ;8D2   584
                        ;8D2   585 found_dpt:
000008DA'00000008'      ;8D2   586          .address 8, 10$
                        ;8DA   587 10$:     string  <!_!XL    !10<!AC!AC!>    !6AD!+!+    !10AC !XL  !XW>
                        ;915   588
                        ;915   589 no_dpt:
0000091D'00000006'      ;915   590          .address 6, 10$
                        ;91D   591 10$:     string  <!_!XL    !10<!AC!AC!>    !6AD!+!+    !10AC>
                        ;94F   592
                00000DFD;      593          .restore
                        ;DFD   594
                        ;DFD   595 show_ddbs:
                0FFC    ;DFD   596          .word   ^m<r2,r3,r4,r5,r6,r7,r8,r9,r10,r11>
                        ;DFF   597
                        ;DFF   598          skip    page
                        ;E06   599          print   0,<!_!_!_!_DDB list>
                        ;E13   600          print   0,<!_!_!_!_-------->
                        ;E20   601          skip    1
                        ;E29   602          print   0,<!_ Address    Controller     ACP       Driver      DPT   DPT size
                        ;E36   603          print   0,<!_ -------    ----------     ---       ------      ---   --------
                        ;E43   604          skip    1
                     5B D4      ;E4C   605          clrl    r11                         ; make get_ddb initialize
                        ;E4E   606
                   00B4 30      ;E4E   607 10$:     bsbw    get_ddb                     ; find next DDB
                  62 50 E9      ;E51   608          blbc    r0, 90$                     ; end of DDB list
          54    00000915'EF 7D  ;E54   609          movq    no_dpt, r4                  ; assume no DPT will be found
                     5A 10      ;E5B   610          bsbb    find_dpt                    ; locate dpt; r7 = local dpt; r8 = address
                  0D 50 E9      ;E5D   611          blbc    r0, T7$                     ; branch if not found
          54    000008D2'EF 7D  ;E60   612          movq    found_dpt, r4               ; show that DPT was found
           7E    08 A7 3C       ;E67   613          movzwl  dpt$w_size(r7), -(sp)       ; length of DPT
                     58 DD      ;E6B   614          pushl   r8                          ; address of DPT
                  24 A6 DF      ;E6D   615 17$:     pushal  ddb$t_drvname(r6)           ; address of driver name
                     7E 7C      ;E70   616          clrq    -(sp)                       ; allocate 2 longwords for ACP name
                     6E DF      ;E72   617          pushal  (sp)
                     7E D4      ;E74   618          clrl    -(sp)                       ; assume no ACP name for this DDB
50    10 A6    FF000000 8F CB  ;E76   619          bicl3   #^xff000000, -              ; obtain ACP name for this DDB
                        ;E7F   620                  ddb$l_acpd(r6), r0
                     20 13      ;E7F   621          beql    30$                         ; branch if no ACP name in this DDB
                  08 AE 50 D0   ;E81   622          movl    r0, 8(sp)                   ; put name in the working string
                     6E 06 D0   ;E85   623          movl    #6, (sp)                    ; set length of ACP name
      0B AE    00505158 8F D0   ;E88   624          movl    #^a'XQP', 11(sp)            ; assume ACP is really an XQP
         50    00313146 8F D1   ;E90   625          cmpl    #^a'F11', r0                ; is it an XQP?
```

```
                    08   13  0E97   626          beql    30$                          ; branch if its an XQP
          0B AE 00504341 8F   D0  0E99   627          movl    #^a'ACP', 11(sp)             ; else, change it to an ACP
                    14 A6   DF  0EA1   628 30$:     pushal  ddb$t_name(r6)               ; generic device name for controller
                    44 AB   9F  0EA4   629          pushab  sb$t_nodename(r11)           ; node name
                    59      DD  0EA7   630          pushl   r9                           ; actual address of DDB
                                0EA9   631          printd  r4, (r5)                     ; print a line
                    98      11  0EB4   632          brb     10$                          ; loop till out of DDBs
                                04  0EB6   633 90$:     ret                              ; then return
```

```
                                OEB7   635 ;
                                OEB7   636 ;       Subroutine to locate the DPT corresponding to the current
                                OEB7   637 ;       DDB.
                                OEB7   638 ;
                                OEB7   639 find_dpt:
                          3C    BB    OEB7   640          pushr   #^m<r2,r3,r4,r5>
      57    00000439'EF         9E    OEB9   641          movab   dpt,r7
                                OEC0   642          trymem  @ioc$gl_dptlist,dpt$l_flink(r7) ; set address of first DPT
                       2F 50    E9    OED0   643          blbc    r0,90$                  ; branch if error
                    58    67    D0    OED3   644 10$:     movl    dpt$l_flink(r7),r8      ; skip to next DPT
      00000000'EF   58    D1    OED6   645          cmpl    r8,ioc$gl_dptlist       ; check if back to listhead
                          21    13    OEDD   646          beql    80$                     ; branch if end of list
                                OEDF   647          trymem  (r8),(r7),#dpt$c_length ; read the entire dpt
                       13 50    E9    OEEC   648          blbc    r0,90$                  ; branch if error
                 50    20 A7    9A    OEEF   649          movzbl  dpt$t_name(r7),r0       ; get length of dpt driver name
      25 A6      21 A7    50    29    OEF3   650          cmpc    r0,dpt$t_name+1(r7),ddb$t_drvname+1(r6)
                          D8    12    OEF9   651          bneq    10$                     ; branch if no match yet
                    50    01    D0    OEFB   652 50$:     movl    #1,r0                   ; success
                          02    11    OEFE   653          brb     90$
                          50    D4    0F00   654 80$:     clrl    r0                      ; not found
                          3C    BA    0F02   655 90$:     popr    #^m<r2,r3,r4,r5>
                                05    0F04   656          rsb
```

```
                     0F05   658          .sbttl  get_ddb -- locate the next DDB in the I/O database
                     0F05   659 ;---
                     0F05   660 ;
                     0F05   661 ;       get_ddb
                     0F05   662 ;
                     0F05   663 ;       This routine locates the next DDB in the I/O database.  All
                     0F05   664 ;       available system blocks are searched.  However, if a node name
                     0F05   665 ;       is specified, only the system block whose node name matches
                     0F05   666 ;       actually has DDBs returned.
                     0F05   667 ;
                     0F05   668 ; Inputs:
                     0F05   669 ;
                     0F05   670 ;       r6 -    addr of DDB, local storage
                     0F05   671 ;       r11 -   addr of SB, local storage
                     0F05   672 ;               (zero means initialize scan)
                     0F05   673 ;
                     0F05   674 ; Outputs:
                     0F05   675 ;
                     0F05   676 ;       r0 -    status
                     0F05   677 ;       r6 -    addr of DDB, local storage
                     0F05   678 ;       r9 -    SYS VA of DDB
                     0F05   679 ;       r11-    addr of SB, local storage
                     0F05   680 ;
                     0F05   681 ;---
                     0F05   682
                     0F05   683 get_ddb:
          5B   D5    0F05   684          tstl    r11                     ; must we initialize?
          64   13    0F07   685          beql    1500$                   ; branch if must initialize
                     0F09   686
     59   66   D0    0F09   687 10$:     movl    ddb$l_link(r6),r9       ; skip to next DDB
               61   13  0F0C   688          beql    100$                    ; if end of list, go try next SB
                     0F0E   689          getmem  (r9), (r6), -           ; read entire DDB
                     0F0E   690                  #ddb$c_length
          4A 50    E9  0F1F   691          blbc    r0, 90$                 ; skip if cannot read
 57   00000551'EF    9E  0F22   692          movab   parsed_devnam, r7       ; get parsed device name data base addr.
       51   10 A7    9A  0F29   693          movzbl  pdvnm_t_ddc(r7), r1     ; was generic device specified?
               0E   13  0F2D   694          beql    50$                     ; branch if not
       14 A6   51   91  0F2F   695          cmpb    r1, ddb$t_name(r6)      ; Is device name big enough?
               D4   1A  0F33   696          bgtru   10$                     ; branch if not
 15 A6   11 A7   51   29  0F35   697          cmpc3   r1, pdvnm_t_ddc+1(r7), -
                     0F3B   698                  ddb$t_name+1(r6)
               CC   12  0F3B   699          bneq    10$                     ; loop until end of list
    44 AB   22 A7   90  0F3D   700 50$:     movb    pdvnm_b_nodesz(r7), -   ; assume that the node name is
                     0F42   701                  sb$t_nodename(r11)      ;  required for this DDB
00000000'EF   34 A6   D1  0F42   702          cmpl    ddb$l_sb(r6), -         ; is this the local node?
                     0F4A   703                  scs$ga_localsb
               1D   12  0F4A   704          bneq    70$                     ; no, node name is required
       51   04 A6   D0  0F4C   705          movl    ddb$l_ucb(r6), r1       ; for the local node, we want to
               06   12  0F50   706          bneq    53$                     ;  show a node name if and only if
       51   40 A6   D0  0F52   707          movl    ddb$l_dp_ucb(r6), r1    ;  this is a file oriented device
               11   13  0F56   708          beql    70$                     ; if we cannot tell, show the node name
                     0F58   709 53$:     getmem  ucb$l_devchar(r1)       ; else test for a file oriented device
       03 51   0E   E0  0F62   710          bbs     #dev$v_fod, r1, 70$     ;  using device characteristics flag
           44 AB   94  0F66   711          clrb    sb$t_nodename(r11)      ; if not fod, vanish node name
          50   01   D0  0F69   712 70$:     movl    #1,r0                   ; set success
                     05  0F6C   713 90$:     rsb
               52   11  0F6D   714 1500$:   brb     500$                    ; branch assist
```

```
                                 0F6F    715
                                 0F6F    716 ;
                                 0F6F    717 ; move to next SB
                                 0F6F    718 ;
                                 0F6F    719
                      50   D4    0F6F    720 100$:   clrl    r0                          ; Set for failure
                   5A   6B   D0  0F71    721         movl    sb$l_flink(r11), r10        ; Get next block
         00000000'EF   5A   D1  0F74    722         cmpl    r10, scs$gq_config          ; Reached end of queue?
                      EF   13  0F7B    723         beql    90$                         ; yes
                               0F7D    724         getmem  (r10), (r11), -             ; Pick up system block
                               0F7D    725                 #sb$c_length
                   DB 50   E9  0F8E    726         blbc    r0,90$                      ; exit if broken
                   54 AB   D0  0F91    727         movl    sb$l_ddb(r11),-
                      66       0F94    728                  ddb$l_link(r6)             ; set address of first DDB
      5A   00000551'EF   9E   0F95    729         movab   parsed_devnam, r10          ; get parsed device name data base addr.
            50   44 AB   9A   0F9C    730         movzbl  sb$t_nodename(r11), r0      ; get size of node name
                   0A   13   0FA0    731         beql    120$                        ; branch if no node name
               45 AB40   24   90   0FA2    732         movb    #^a/$/, -               ; append "$" to the node name
                               0FA7    733                 sb$t_nodename+1(r11)[r0]
            22 AA   50   01   81   0FA7    734         addb3   #1, r0, pdvnm_b_nodesz(r10) ; store new node name size
                   55   6A   9A   0FAC    735 120$:   movzbl  pdvnm_t_node(r10), r5    ; pick up requested node name lenght
                   0D   13   0FAF    736         beql    130$                        ; there is none, go scan DDB chain
                   50   55   91   0FB1    737         cmpb    r5, r0                  ; do length match?
                   B9   12   0FB4    738         bneq    100$                        ; no, this cannot be it
         45 AB   01 AA   55   29   0FB6    739         cmpc3   r5, -                   ; do names match?
                               0FBC    740                 pdvnm_t_node+1(r10), -
                               0FBC    741                 sb$t_nodename+1(r11)
                      B1   12   0FBC    742         bneq    100$                        ; no, this cannot be it
                   FF48   31   0FBE    743 130$:   brw     10$                         ; go scan the DDB chain
                               0FC1    744
                               0FC1    745 ;
                               0FC1    746 ; initialize I/O database scan
                               0FC1    747 ;
                               0FC1    748
   5B   00000000'EF   9E   0FC1    749 500$:   movab   sb, r11                     ; pickup local SB storage address
   56   00000071'EF   9E   0FC8    750         movab   ddb, r6                     ; pickup local DDB storage address
                               0FCF    751         getmem  @scs$gq_config, -           ; initialize next SB pointer
                               0FCF    752                 sb$l_flink(r11)
                      8E   11   0FDF    753         brb     100$                        ; link to next SB
```

```
                                0FE1    755          .sbttl  show_controller, Display controller information
                                0FE1    756 ;---
                                0FE1    757 ;
                                0FE1    758 ;       show_controller
                                0FE1    759 ;
                                0FE1    760 ;       Display all information related to the controller
                                0FE1    761 ;       device associated with each generic device name.
                                0FE1    762 ;
                                0FE1    763 ;   Inputs:
                                0FE1    764 ;
                                0FE1    765 ;       4(ap) = Address of DDB in local storage
                                0FE1    766 ;       8(ap) = Address of UCB in local storage
                                0FE1    767 ;       12(ap)= Address of node name in local storage
                                0FE1    768 ;       16(ap)= SVA of DDB
                                0FE1    769 ;
                                0FE1    770 ;---
                                0FE1    771
                                0FE1    772 show_controller:
                        00FC    0FE1    773          .word   ^m<r2,r3,r4,r5,r6,r7>
              52    04 AC      7D    0FE3    774          movq    4(ap),r2                ; get address of DDB,UCB
      54    00000000'EF        9E    0FE7    775          movab   buffer,r4
                                0FEE    776
                                0FEE    777 ; begin with controller heading
                                0FEE    778
                                0FEE    779          skip    page
                   14 A2        DF    0FF5    780          pushal  ddb$t_name(r2)          ; generic controller name
                   0C AC        DD    0FF8    781          pushl   12(ap)
                                0FFB    782          print   2,<Controller: !AC!AC>
                      0C        DD    1008    783          pushl   #12
                6E    14 A2     80    100A    784          addb    ddb$t_name(r2), (sp)
                6E    0C BC     80    100E    785          addb    @12(ap), (sp)
                                1012    786          print   1,<!#*->
                                101F    787          skip    1
                                1028    788
00000000'EF     34 A2     91    1028    789          cmpb    ddb$l_sb(r2), scs$ga_localsb    ; skip this stuff if
                   08     13    1030    790          beql    skip_sb                         ;  this is the local SB
                34 A2     DD    1032    791          pushl   ddb$l_sb(r2)                    ; else, display SB and
          17D8'CF  01     FB    1035    792          calls   #1, w^show_system_block         ;  related information
                                103A    793
                                103A    794 skip_sb:
                                103A    795          getmem  @16(ap), (r4), #ddb$k_length    ; copy DDB to local mem.
                                104C    796          retiferr
                                1050    797          ensure  6
                10 AC     DD    1068    798          pushl   16(ap)
                                106B    799          print   1,<!_!_--- Device Data Block (DDB) !XL --->
                                1078    800          skip    1
                                1081    801          print_columns -
                                1081    802                  buffer, 16(ap), -
                                1081    803                  ddb_column_1, ddb_column_2, ddb_column_3
                                10A3    804          skip    1
                                10AC    805
                                10AC    806          getmem  @ucb$l_crb(r3), (r4), #crb$k_length     ; get primary CRB
                                10BE    807          retiferr
                                10C2    808          ensure  8
                24 A3     DD    10DA    809          pushl   ucb$l_crb(r3)
                                10DD    810          print   1,<!_    ---  Primary Channel Request Block (CRB) !XL --->
                                10EA    811          skip    1
```

```
00000578'EF    40 A3    90  10F3    812          movb    ucb$b_devclass(r3), crb_devclass        ; setup device info.
                        10FB    813          print_columns -
                        10FB    814                  buffer, ucb$l_crb(r3), -                ; output CRB columns
                        10FB    815                  crb_column_1, crb_column_2, crb_column_3
         50    24 A3    24  C1  111D    816          addl3   #crb$l_intd, ucb$l_crb(r3), r0
                        1122    817          print_columns -
                        1122    818                  buffer+crb$l_intd, r0, -                ; output VEC columns
                        1122    819                  vec_column_1, vec_column_2, vec_column_3
                        1143    820          skip    1
                        114C    821
            57    20 A4    D0  114C    822          movl    crb$l_link(r4), r7                      ; link to second. CRB
                  03       12  1150    823          bneq    10$
                0093       31  1152    824          brw     skip_second_crb                         ; branch if none
                        1155    825 10$:     getmem  (r7), (r4), #crb$k_length               ; get secondary CRB
                        1166    826          retiferr
                        116A    827          ensure  8
                  57    DD    1182    828          pushl   r7
                        1184    829          print   1,<!_  ---  Secondary Channel Request Block (CRB) !XL --->
                        1191    830          skip    1
                        119A    831          print_columns -
                        119A    832                  buffer, r7, -                           ; output CRB columns
                        119A    833                  crb_column_1, crb_column_2, crb_column_3
               57    24  C0  11BB    834          addl2   #crb$l_intd, r7
                        11BE    835          print_columns -
                        11BE    836                  buffer+crb$l_intd, r7, -                ; output VEC columns
                        11BE    837                  vec_column_1, vec_column_2, vec_column_3
                        11DF    838          skip    1
                        11E8    839
                        11E8    840 skip_second_crb:
00000000'EF    34 A2    D1  11E8    841          cmpl    ddb$l_sb(r2), scs$ga_localsb            ; is this a local dev.?
                  03    13  11F0    842          beql    10$
                0080    31  11F2    843          brw     display_ddt                             ; if so, skip IDB etc.
      57    24 A3    2C  C1  11F5    844 10$:     addl3   #<crb$l_intd+vec$l_idb>, -              ; locate address of
                        11FA    845                  ucb$l_crb(r3), r7                       ;  primary IDB
                        11FA    846          getmem  (r7)                                    ; get that address
                        1203    847          retiferr
               57    51  D0  1207    848          movl    r1, r7                                  ; save IDB address
                        120A    849          getmem  (r7), (r4), #idb$k_length               ; copy IDB to local mem.
                        1217    850          retiferr
                        121B    851          ensure  4
                  57    DD    1233    852          pushl   r7
                        1235    853          print   1,<!_!_--- Interrupt Data Block (IDB) !XL --->
                        1242    854          skip    1
                        124B    855          print_columns -
                        124B    856                  buffer, r7, -
                        124B    857                  idb_column_1, idb_column_2, idb_column_3
                        126C    858          skip    1
                        1275    859
                        1275    860 display_ddt:
                        1275    861          getmem  @ucb$l_ddt(r3), (r4), #ddt$k_length     ; copy DDT to local mem.
                        1284    862          retiferr
                        1288    863          ensure  6
               0088 C3  DD  12A0    864          pushl   ucb$l_ddt(r3)
                        12A4    865          print   1,<!_!_--- Driver Dispatch Table (DDT) !XL --->
                        12B1    866          skip    1
                        12BA    867          print_columns -
                        12BA    868                  buffer, ucb$l_ddt(r3), -
```

```
                12BA    869                     ddt_column_1, ddt_column_2, ddt_column_3
                12DD    870
 04             12DD    871             ret
```

```
                        873          .sbttl show_controller tables & action routines
                        874
                        875 ;  The following are all PRINT_COLUMNS action routines for the show
                        876 ;  controller displays.
                        877 ;
                        878 ;    Action Routine Inputs:
                        879 ;
                        880 ;        R2              value from the COLUMN_LIST entry
                        881 ;        R5              size of value section for this item
                        882 ;        R7              address of a descriptor for a scratch string in
                        883 ;                        which the FAO converted value is to be returned
                        884 ;        R11             base address of the local UCB copy
                        885 ;
                        886 ;    Action Routine Outputs:
                        887 ;
                        888 ;        R0              status
                        889 ;                           lbs ==> use this entry
                        890 ;                           lbc ==> skip this entry
                        891 ;        R1 - R5         scratch
                        892 ;                        all other registers must be preserved
                        893
                        894 ;
                        895 ; FAO control strings, etc. used by the action routines
                        896 ;
                        897
                        898          .save
                00000B47 899          .psect  literals
                    0B47 900
                    0B47 901 vec_fao_datapath:
                    0B47 902          string  <!UB!AC!AC>
                    0B58 903
                    0B58 904 vec_fao_mapreg:
                    0B58 905          string  <!UB(!UB)!AC>
                    0B6B 906
                    0B6B 907 vec_lwae:
          45 41 57 4C 20 00' 0B6B 908          .ascic  / LWAE/
                       05    0B6B
                    0B71 909
                    0B71 910 vec_locked:
    64 65 6B 63 6F 4C 20 00' 0B71 911          .ascic  / Locked/
                       07    0B71
                    0B79 912
                    0B79 913 ddt_return:
       6E 72 75 74 65 72 00' 0B79 914          .ascic  /return/
                       06    0B79
                    0B80 915
                00001 2DE 916          .restore
                    12DE 917
                    12DE 918 ;
                    12DE 919 ; PRINT_COLUMNS tables for DDB display
                    12DE 920 ;
                    12DE 921
                    12DE 922 ddb_column_1:
                    12DE 923          column_list -
                    12DE 924                  ddb$, 20, 8, 3, <-
                    12DE 925                  <<Driver name>,t_drvname,ac,13,15>, -
                    12DE 926                  <<ACP ident>,ddb_acpd,0,25,3>, -
```

```
                        12DE   927                 <<ACP class>,ddb_acpcls,0>, -
                        12DE   928                 >
                        131E   929
                        131E   930 ddb_column_2:
                        131E   931         column_list -
                        131E   932                 ddb$, 15, 8, 3, <-
                        131E   933                 <<Alloc. class>,l_allocls,ub>, -
                        131E   934                 <<SB address>,l_sb,xl>, -
                        131E   935                 <<UCB address>,l_ucb,xl>, -
                        131E   936                 >
                        135E   937
                        135E   938 ddb_column_3:
                        135E   939         column_list -
                        135E   940                 ddb$, 15, 8, 0, <-
                        135E   941                 <<DDT address>,l_ddt,xl>, -
                        135E   942                 <<CONLINK addr.>,l_conlink,xl_neq>, -
                        135E   943                 <<2p UCB addr.>,l_dp_ucb,xl_neq>, -
                        135E   944                 >
                        139E   945
                        139E   946 ;********
                        139E   947 ddb_acpd:
52   10 AB   FF000000 8F   CB  139E   948         bicl3   #^xff000000, ddb$l_acpd(r11), - ; get ACP descriptor
                               13A7   949                 r2
                     18   13  13A7   950         beql    ddb_no_acp                      ; branch if no ACP info
              52     52   08  9C  13A9   951         rotl    #8, r2, r2                      ; make ACP descriptor into
                     52   03  C0  13AD   952         addl    #3, r2                          ;  an ASCIC string and
                     52   DD  13B0   953         pushl   r2                              ;  push it onto the stack
                     52   5E  D0  13B2   954         movl    sp, r                           ; save ASCIC pointer
                               13B5   955         do_column_entry ac                      ; display ACP type id
                          8E   D5  13BE   956         tstl    (sp)+                           ; cleanup stack
                               05  13C0   957         rsb
                               13C1   958 ddb_no_acp:
                          50   D4  13C1   959         clrl    r0
                               05  13C3   960         rsb
                               13C4   961
                               13C4   962 ;**********
                               13C4   963 ddb_acpcls:
  52    13 AB    9A  13C4   964         movzbl  ddb$b_acpclass(r11), r2          ; get ACP class
           F7   13  13C8   965         beql    ddb_no_acp                       ; branch if none
53    ECC2 CF   9E  13CA   966         movab   ddb_acpclass, r3                 ; get translate table
 00000000'GF   16  13CF   967         jsb     g^translate_address              ; translate ACP class
           0C   13  13D5   968         beql    90$                              ; branch if translate failed
      52   50   D0  13D7   969         movl    r0, r2                           ; setup translated string
                          13DA   970         do_column_entry ac, jmp                  ; display translation
                          13E3   971
  52    13 AB    9E  13E3   972 90$:     movab   ddb$b_acpclass(r11), r2          ; else, get class address
                          13E7   973         do_column_entry ub, jmp                  ;  just display the value
                          13F0   974
                          13F0   975 ;
                          13F0   976 ; PRINT_COLUMNS tables for CRB display
                          13F0   977 ;
                          13F0   978
                          13F0   979 crb_column_1:
                          13F0   980         column_list -
                          13F0   981                 crb$, 16, 8, 4, <-
                          13F0   982                 <<Reference count>,w_refc,uw>, -
                          13F0   983                 <<Due time>,crb_timeout,crb$l_duetime>, -
```

```
                     13F0   984                     >
                     1420   985
                     1420   986 crb_column_2:
                     1420   987             :olumn_list -
                     1420   988                     crb$, 16, 8, 4, <-
                     1420   989                     <<Wait queue>,l_wqfl,q2>, -
                     1420   990                     <<Timeout rout.>,crb_timeout,crb$l_toutrout>, -
                     1420   991                     >
                     1450   992
                     1450   993 crb_column_3:
                     1450   994             column_list -
                     1450   995                     crb$, 16, 8, 0, <-
                     1450   996                     <<Aux. struct.>,l_auxstruc,xl_neq>, -
                     1450   997                     <<Timeout link>,crb_timeout,crb$l_timelink>, -
                     1450   998                     >
                     1480   999
                     1480  1000 ;**********
                     1480  1001 crb_timeout:
00000578'EF    42 8F 91  1480  1002             cmpb    #dc$_term, -                    ; terminals have a different
                     1488  1003                     crb_devclass                    ;  timeout scheme
                  11 13  1488  1004             beql    90$                             ;  so don't do them
                1C AB D5  148A  1005             tstl    crb$l_toutrout(r11)             ; also don't bother unless
                   0C 13  148D  1006             beql    90$                             ;  a time out routine specified
              52    5B C0  148F  1007             addl    r11, r2                         ; get datum address
                     1492  1008             do_column_entry xl, jmp                 ;  and display it
                   50 D4  149B  1009 90$:        clrl    r0                              ; or don't show anything
                      05  149D  1010             rsb
                     149E  1011
                     149E  1012             .save
                 00000578  1013             .psect  sdadata,noexe,wrt
                     0578  1014 crb_devclass:
00000000             0578  1015             .long   0
                 0000149E  1016             .restore
                     149E  1017
                     149E  1018 ;
                     149E  1019 ; PRINT_COLUMNS tables for VEC display
                     149E  1020 ;
                     149E  1021
                     149E  1022 vec_column_1:
                     149E  1023             column_list -
                     149E  1024                     vec$, 16, 8, 4, <-
                     149E  1025                     <<IDB address>,l_idb,xl>, -
                     149E  1026                     <<ADP address>,l_adp,xl_neq>, -
                     149E  1027                     <<Unit start rout.>,l_start,xl_neq>, -
                     149E  1028                     >
                     14DE  1029
                     14DE  1030 vec_column_2:
                     14DE  1031             column_list -
                     14DE  1032                     vec$, 16, 8, 4, <-
                     14DE  1033                     <<Datapath>,vec_datapath,0,10,14>, -
                     14DE  1034                     <<Unit init.>,l_unitinit,xl_neq>, -
                     14DE  1035                     <<Disc. rout.>,l_unitdisc,xl_neq>, -
                     14DE  1036                     >
                     151E  1037
00000004             151E  1038 vec$l_intser = vec$q_dispatch+4
                     151E  1039 vec_column_3:
                     151E  1040             column_list -
```

```
                                 151E  1041                vec$, 16, 8, 0, <-
                                 151E  1042                <<Map reg.>,vec_mapreg,0,11,13>, -
                                 151E  1043                <<Int. service>,l_intser,xl_neq>, -
                                 151E  1044                <<Ctrl. init.>,l_initial,xl_neq>, -
                                 151E  1045                >
                                 155E  1046
                                 155E  1047 ;************
                                 155E  1048 vec_datapath:
                          5E     10  155E  1049          bsbb    vec_test_uba                    ; is this a UNIBUS?
                       5E    18   C2  1560  1050          subl    #<8+16>, sp                     ; make scratch space on stack
                       52    5E   D0  1563  1051          movl    sp, r2                          ; point to string descriptor
                       62    10   D0  1566  1052          movl    #16, (r2)                       ; build string descriptor
                    04 A2    08 A2   9E  1569  1053          movab   8(r2), 4(r2)
                 53   00000E21'EF     9E  156E  1054          movab   null_ascic, r3                  ; assume no LWAE
                    07 13 AB    05   E1  1575  1055          bbc     #vec$v_lwae, -                  ; branch if LWAE not on
                                 157A  1056                  vec$b_datapath(r11), 10$
                 53   00000B6B'EF     9E  157A  1057          movab   vec_lwae, r3                    ; else, change assumption
                 54   00000E21'EF     9E  1581  1058 10$:     movab   null_ascic, r4                  ; assume no pathlock
                    07 13 AB    07   E1  1588  1059          bbc     #vec$v_pathlock, -              ; branch if path not locked
                                 158D  1060                  vec$b_datapath(r11), 20$
                 54   00000B71'EF     9E  158D  1061          movab   vec_locked, r4                  ; else, change assumption
              51   13 AB    05   00   EF  1594  1062 20$:     extzv   #vec$v_datapath, -              ; extract data path number
                                 159A  1063                  #vec$s_datapath, -
                                 159A  1064                  vec$b_datapath(r11), r1
                                 159A  1065          $fao_s  -
                                 159A  1066                  ctrstr = vec_fao_datapath, -    ; convert everything to
                                 159A  1067                  outbuf = (r2), -                ;  to a string
                                 159A  1068                  outlen = (r2), -
                                 159A  1069                  p1 = r1, -
                                 159A  1070                  p2 = r3, -
                                 159A  1071                  p3 = r4
                                 15B1  1072          do_column_entry as                      ; put string in column
                       5E    18   C0  15BA  1073          addl    #<8+16>, sp                     ; cleanup stack
                                 05  15BD  1074          rsb
                                 15BE  1075
                                 15BE  1076
                                 15BE  1077 ;*********
                                 15BE  1078 vec_test_uba:
                    50   14 AB    D0  15BE  1079          movl    vec$l_adp(r11), r0              ; get ADP address
                                 13  15C2  1080          beql    90$                             ; if none, its not a UBA
                                 15C4  1081          getmem  adp$w_adptype(r0)               ; get adapter type
                       06 50     E9  15CE  1082          blbc    r0, 90$                         ; if error, its not a UBA
                       51    01   B1  15D1  1083          cmpw    #at$_uba, r1                    ; is it a UBA?
                                 01  15D4  1084          bneq    90$                             ; branch if not a UBA
                                 05  15D6  1085          rsb                             ; else, return to caller
                                 8E  15D7  1086 90$:     tstl    (sp)+                           ; if not a UBA, return a skip
                                 50  15D9  1087          clrl    r0                              ;  this entry status to the
                                 05  15DB  1088          rsb                             ;  action routines caller
                                 15DC  1089
                                 15DC  1090 ;**********
                                 15DC  1091 vec_mapreg:
                                 E0  10  15DC  1092          bsbb    vec_test_uba                    ; is this a UBA?
                          5E     18  C2  15DE  1093          subl    #<8+16>, sp                     ; make scratch space on stack
                          52     5E  D0  15E1  1094          movl    sp, r2                          ; point to string descriptor
                          62     10  D0  15E4  1095          movl    #16, (r2)                       ; build string descriptor
                       04 A2    08 A2   9E  15E7  1096          movab   8(r2), 4(r2)
                    54   00000E21'EF     9E  15EC  1097          movab   null_ascic, r4                  ; assume no map lock
```

```
                07 10 AB   0F    E1  15F3  1098          bbc     #vec$v_maplock, -                ; branch if no map lock
                                     15F8  1099                  vec$w_mapreg(r11), 10$
             54   00000B71'EF   9E    15F8  1100          movab   vec_locked, r4                   ; else, change assumption
          53   10 AB   0F   00    EF  15FF  1101 10$:     extzv   #vec$v_mapreg, #vec$s_mapreg, - ; extract starting map
                                     1605  1102                  vec$w_mapreg(r11), r3            ;  number
                                     1605  1103          $fao_s  -
                                     1605  1104                  ctrstr = vec_fao_mapreg, -       ; convert whole mess to a
                                     1605  1105                  outbuf = (r2), -                 ;  string
                                     1605  1106                  outlen = (r2), -
                                     1605  1107                  p1 = r3, -
                                     1605  1108                  p2 = vec$b_numreg(r11), -
                                     1605  1109                  p3 = r4
                                     161D  1110          do_column_entry as                       ; put string in column
                      5E   18   C0   1626  1111          addl    #<8+16>, sp                      ; cleanup stack
                                05   1629  1112          rsb
                                     162A  1113
                                     162A  1114 ;
                                     162A  1115 ; PRINT_COLUMNS tables for IDB display
                                     162A  1116 ;
                                     162A  1117
                                     162A  1118 idb_column_1:
                                     162A  1119          column_list -
                                     162A  1120                  idb$, 16, 8, 4, <-
                                     162A  1121                  <<CSR address>,l_csr,xl>, -
                                     162A  1122                  <<Number of units>,w_units,uw>, -
                                     162A  1123                  >
                                     165A  1124
                                     165A  1125 idb_column_2:
                                     165A  1126          column_list -
                                     165A  1127                  idb$, 16, 8, 4, <-
                                     165A  1128                  <<Owner UCB addr.>,l_owner,xl>, -
                                     165A  1129                  <<Interrupt vector>,idb_vector,0,18,6>, -
                                     165A  1130                  >
                                     168A  1131
                                     168A  1132 idb_column_3:
                                     168A  1133          column_list -
                                     168A  1134                  idb$, 16, 8, 0, <-
                                     168A  1135                  <<ADP address>,l_adp,xl>, -
                                     168A  1136                  >
                                     16AA  1137
                                     16AA  1138 ;**********
                                     16AA  1139 idb_vector:
             50   0B AB   9A    16AA  1140          movzbl  idb$b_vector(r11), r0            ; Obtain vector information
                     12   13    16AE  1141          beql    90$                              ; Branch if none present
          7E   50    02   78    16B0  1142          ashl    #2, r0, -(sp)                    ; Convert vector information
               52    5E   D0    16B4  1143          movl    sp, r2                           ; Get converted info. addr.
                                16B7  1144          do_column_entry ow                       ; Display information
                     8E   D5    16C0  1145          tstl    (sp)+                            ; Cleanup stack
                          05    16C2  1146 90$:     rsb                                      ; Return to caller
                                16C3  1147
                                16C3  1148 ;
                                16C3  1149 ; PRINT_COLUMNS tables for DDT display
                                16C3  1150 ;
                                16C3  1151
                                16C3  1152 ddt_column_1:
                                16C3  1153          column_list -
                                16C3  1154                  ddt$, 16, 8, 4, <-
```

```
                        16C3  1155                    <<Errlog buf sz>,w_errorbuf,uw>,-
                        16C3  1156                    <<Start I/O>,ddt_address,ddt$l_start>, -
                        16C3  1157                    <<Alt start I/O>,ddt_address,ddt$l_altstart>, -
                        16C3  1158                    <<Cancel I/O>,ddt_address,ddt$l_cancel>, -
                        16C3  1159                    >
                        1713  1160
                        1713  1161 ddt_column_2:
                        1713  1162            column_list -
                        1713  1163                    ddt$, 16, 8, 4, <-
                        1713  1164                    <<Diag buf sz>,w_diagbuf,uw>, -
                        1713  1165                    <<Register dump>,ddt_address,ddt$l_regdump>, -
                        1713  1166                    <<Unit init>,ddt_address,ddt$l_unitinit>, -
                        1713  1167                    <<Unsol int>,ddt_address,ddt$l_unsolint>, -
                        1713  1168                    >
                        1763  1169
                        1763  1170 ddt_column_3:
                        1763  1171            column_list -
                        1763  1172                    ddt$, 16, 8, 0, <-
                        1763  1173                    <<FDT size>,w_fdtsize,uw>, -
                        1763  1174                    <<FDT address>,l_fdt,xl>, -
                        1763  1175                    <<Mnt verify>,ddt_address,ddt$l_mntver>, -
                        1763  1176                    <<Cloned UCB>,ddt_address,ddt$l_cloneducb>, -
                        1763  1177                    >
                        17B3  1178
                        17B3  1179 ;***********
                        17B3  1180 ddt_address:
          52   5B   C0  17B3  1181            addl    r11, r2                 ; get datum address
00000000'EF    62   D1  17B6  1182            cmpl    (r2), ioc$return        ; is this the RSB routine?
               09   13  17BD  1183            beql    90$                     ; branch if RSB routine
                        17BF  1184            do_column_entry xl, jmp         ; else, output value
52   00000B79'EF    9E  17C8  1185 90$:       movab   ddt_return, r2          ; for RSB routine, display
                        17CF  1186            do_column_entry ac, jmp         ;  "return"
```

```
                                17D8  1188          .sbttl  show_system_block, show system/path blocks (SB/PB)
                                17D8  1189 ;---
                                17D8  1190 ;
                                17D8  1191 ;       show_system_block
                                17D8  1192 ;
                                17D8  1193 ;       This routine displays the system and path blocks given
                                17D8  1194 ;       the address of the system block.
                                17D8  1195 ;
                                17D8  1196 ;       4(ap) = SVA of the system block of interest
                                17D8  1197 ;---
                                17D8  1198
                                17D8  1199 show_system_block::
                         01FC   17D8  1200          .word   ^m<r2,r3,r4,r5,r6,r7,r8>
      54  00000000'EF    9E   17DA  1201          movab   buffer, r4                              ; get working buffer
                                17E1  1202
                                17E1  1203 ; display system block
                                17E1  1204
                                17E1  1205          ensure  12
                                17F9  1206          getmem  @4(ap), (r4), #sb$k_length          ; copy SB to local mem.
                                180B  1207          retiferr
                   04 AC    DD  180F  1208          pushl   4(ap)
                   44 A4    9F  1812  1209          pushab  sb$t_nodename(r4)                   ; node name
                                1815  1210          print   1,<!_!_    --- !AC System Block (SB) !XL --->
                                1822  1211          skip    1
                                182B  1212          print_columns -
                                182B  1213                  buffer, 4(ap), -
                                182B  1214                  sb_column_1, sb_column_2
                                1847  1215          skip    1
                                1850  1216
                                1850  1217
                                1850  1218 ; display each path block
                                1850  1219
                                1850  1220          assume  pb$k_length lt 512
          64    0C A4    D0  1850  1221          movl    sb$l_pbfl(r4), pb$l_flink(r4)       ; init PB scan
                                1854  1222
                                1854  1223 pb_loop:
      50  04 AC    0C    C1  1854  1224          addl3   #sb$l_pbfl, 4(ap), r0               ; is there another PB?
             50    64    D1  1859  1225          cmpl    pb$l_flink(r4), r0
                   03    12  185C  1226          bneq    10$
                 00B2    31  185E  1227          brw     end_pb                              ; branch if no PBs left
             58    64    D0  1861  1228 10$:     movl    pb$l_flink(r4), r8                  ; save new PB addr.
                               1864  1229          getmem  (r8), (r4), #pb$k_length            ; copy PB to local mem.
                               1875  1230          retiferr
                               1879  1231          ensure  12
                   58    DD  1891  1232          pushl   r8
                               1893  1233          print   1,<!_!_    --- Path Block (PB) !XL --->
                               18A0  1234          skip    1
             57    5E    D0  18A9  1235          movl    sp, r7                              ; save stack pointer
                               18AC  1236          alloc   80, r6                              ; allocate scratch
          7E    44 A4    3C  18BE  1237          movzwl  pb$w_sts(r4), -(sp)                 ; push PB STS
             E73A CF    9F  18C2  1238          pushab  pb_status                           ; push bit conv. data
00000000'GF       02    FB  18C6  1239          calls   #2, g^translate_bits                ; translate PB STS
                   56    DD  18CD  1240          pushl   r6                                  ; push result
          7E    44 A4    3C  18CF  1241          movzwl  pb$w_sts(r4), -(sp)                 ; push PB STS
                               18D3  1242          print   2,<!_!_Status:  !XW  !AS>           ; output PB STS
             5E    57    D0  18E0  1243          movl    r7, sp                              ; restore stack
                               18E3  1244          skip    1
```

```
                18EC  1245          print_columns -
                18EC  1246                  buffer, r8, -
                18EC  1247                  pb_column_1, pb_column_2
                1907  1248          skip    1
      FF41   31 1910  1249          brw     pb_loop
                1913  1250
                1913  1251 end_pb:
             04 1913  1252          ret
```

```
                        1914  1254          .sbttl show_system_block tables & action routines
                        1914  1255
                        1914  1256 ;  The following are all PRINT_COLUMNS action routines for the show
                        1914  1257 ;  system/path block displays.
                        1914  1258 ;
                        1914  1259 ;    Action Routine Inputs:
                        1914  1260 ;
                        1914  1261 ;       R2              value from the COLUMN_LIST entry
                        1914  1262 ;       R5              size of value section for this item
                        1914  1263 ;       R7              address of a descriptor for a scratch string in
                        1914  1264 ;                       which the FAO converted value is to be returned
                        1914  1265 ;       R11             base address of the local UCB copy
                        1914  1266 ;
                        1914  1267 ;    Action Routine Outputs:
                        1914  1268 ;
                        1914  1269 ;       R0              status
                        1914  1270 ;                          lbs ==> use this entry
                        1914  1271 ;                          lbc ==> skip this entry
                        1914  1272 ;       R1 - R5         scratch
                        1914  1273 ;                       all other registers must be preserved
                        1914  1274
                        1914  1275 ;
                        1914  1276 ; FAO control strings, etc. used by the action routines
                        1914  1277 ;
                        1914  1278
                        1914  1279          .save
                    0000ODEE  1280          .psect  literals,exe,nowrt
                        ODEE  1281
                        ODEE  1282 sb_fao_6bytes:
                        ODEE  1283          string  <!#* !XW!XL>
                        0E00  1284
                        0E00  1285 sb_fao_ascic:
                        0E00  1286          string  <!#* !#(AC)>
                        0E12  1287
                        0E12  1288 cddb_fao:
                        0E12  1289          string  <!#* !XL>
                        0E21  1290
                        0E21  1291 null_ascic:
                    00000000  0E21  1292          .long   0
                        0E25  1293
                        0E25  1294 maint_ascic:
                5F 54 4E 49 41 4D 00'  0E25  1295          .ascic  /MAINT_/
                              06  0E25
                        0E2C  1296
                        0E2C  1297 cbl_a_ascic:
                        2D 41 00'  0E2C  1298          .ascic  /A-/
                              02  0E2C
                        0E2F  1299
                        0E2F  1300 cbl_b_ascic:
                     2D 42 20 00'  0E2F  1301          .ascic  / B-/
                              03  0E2F
                        0E33  1302
                        0E33  1303 ok_ascic:
                        4B 4F 00'  0E33  1304          .ascic  /OK/
                              02  0E33
                        0E36  1305
                        0E36  1306 bad_ascic:
```

```
                     44 41 42 00'  0E36  1307          .ascic  /BAD/
                        03         0E36
                                   0E3A  1308
                                   0E3A  1309 crossed_ascic:
                  64 65 58 20 00'  0E3A  1310          .ascic  / Xed/
                        04         0E3A
                                   0E3F  1311
                                   1914  1312          .restore
                                   1914  1313
                                   1914  1314 ;
                                   1914  1315 ; PRINT_COLUMNS tables for SB display
                                   1914  1316 ;
                                   1914  1317
                                   1914  1318 sb_column_1:
                                   1914  1319          column_list -
                                   1914  1320                  sb$, 21, 12, 4, < -
                                   1914  1321                  <<System ID>,sb_6bytes,sb$b_systemid>, -
                                   1914  1322                  <<Max message size>,w_maxmsg,uw>, -
                                   1914  1323                  <<Max datagram size>,w_maxdg,uw>, -
                                   1914  1324                  <<Local hardware type>,sb_lwchar,sb$t_hwtype,29,4>, -
                                   1914  1325                  <<Local hardware vers.>,sb_6bytes,sb$b_hwvers>, -
                                   1914  1326                  << >,sb_6bytes,sb$b_hwvers+6>, -
                                   1914  1327                  >
                                   1984  1328
                        00000030   1984  1329 sb$q_swincarn2 = sb$q_swincarn+4
                                   1984  1330 sb_column_2:
                                   1984  1331          column_list -
                                   1984  1332                  sb$, 21, 12, 0, < -
                                   1984  1333                  <<Local software type>,sb_lwchar,sb$t_swtype,29,4>, -
                                   1984  1334                  <<Local software vers.>,sb_lwchar,sb$t_swvers,29,4>, -
                                   1984  1335                  <<Local software incarn.>,q_swincarn,xl,25,8>, -
                                   1984  1336                  << >,q_swincarn2,xl,25,8>, -
                                   1984  1337                  <<SCS poller timeout>,w_timeout,xw>, -
                                   1984  1338                  <<SCS poller enable mask>,b_enbmsk,xb,31,2>, -
                                   1984  1339                  >
                                   19F4  1340
                                   19F4  1341 ;**********
                                   19F4  1342 sb_6bytes:
                  53   5B   52   C1  19F4  1343          addl3   r2, r11, r3                     ; locate storage of interest
                       55   0C   C2  19F8  1344          subl    #12, r5                         ; get size of filler field
                                   19FB  1345          $fao_s  -
                                   19FB  1346                  ctrstr = sb_fao_6bytes, -
                                   19FB  1347                  outbuf = (r7), -
                                   19FB  1348                  outlen = (r7), -
                                   19FB  1349                  p1 = r5, -
                                   19FB  1350                  p2 = 4(r3), -
                                   19FB  1351                  p3 = (r3)
                             05    1A13  1352          rsb
                                   1A14  1353
                                   1A14  1354 ;**********
                                   1A14  1355 sb_lwchar:
                  53   5B   52   C1  1A14  1356          addl3   r2, r11, r3                     ; locate storage of interest
                             7E   ..  1A18  1357          clrl    -(sp)                           ; make scratch ASCIC space
                             63   95  1A1A  1358          tstb    (r3)                            ; check for null string
                             16   13  1A1C  1359          beql    5$                              ; equal, null string
                             04   DD  1A1E  1360          pushl   #4                              ;  of the right size
                       52    5E   D0  1A20  1361 10$:     movl    sp, r2                          ; save ASCIC pointer
```

```
                01 A2    63    D0  1A23  1362          movl    (r3), 1(r2)                     ; put text in ASCIC string
                                   1A27  1363          do_column_entry ac                      ; convert the ASCIC
                   5E    08    C0  1A30  1364          addl    #2*4, sp                        ; cleanup stack
                                05  1A33  1365          rsb
                                      1A34  1366
                         00    DD  1A34  1367 5$:       pushl   #0
                         E8    11  1A36  1368          brb     10$
                                      1A38  1369
                                      1A38  1370 ;
                                      1A38  1371 ; PRINT_COLUMNS tables for PB display
                                      1A38  1372 ;
                                      1A38  1373
                                      1A38  1374 pb_column_1:
                                      1A38  1375          column_list -
                                      1A38  1376                  pb$, 21, 12, 4, < -
                                      1A38  1377                  <<Remote sta. addr.>,sb_6bytes,pb$b_rstation>, -
                                      1A38  1378                  <<Remote state>,pb_rmtstate,0>, -
                                      1A38  1379                  <<Remote hardware rev.>,l_rport_rev,xl>, -
                                      1A38  1380                  <<Remote func. mask>,l_rport_fcn,xl>, -
                                      1A38  1381                  <<Reseting port>,b_rst_port,xb>, -
                                      1A38  1382                  <<Handshake retry cnt.>,w_retry,uw>, -
                                      1A38  1383                  <<Msg. buf. wait queue>,l_waitqfl,q2>, -
                                      1A38  1384                  >
                                      1AB8  1385
                                      1AB8  1386 pb_column_2:
                                      1AB8  1387          column_list -
                                      1AB8  1388                  pb$, 21, 12, 4, < -
                                      1AB8  1389                  <<Remote port type>,pb_rport_typ,0>, -
                                      1AB8  1390                  <<Number of data paths>,pb_dualpath,0>, -
                                      1AB8  1391                  <<Cables state>,pb_cables,0,18,15>,-
                                      1AB8  1392                  <<Local state>,pb_lclstate,0>, -
                                      1AB8  1393                  <<Port dev. name>,sb_lwchar,pb$t_lport_name,29,4>, -
                                      1AB8  1394                  <<SCS MSGBUF address>,l_scsmsg,xl>, -
                                      1AB8  1395                  <<PDT address>,l_pdt,xl>, -
                                      1AB8  1396                  >
                                      1B38  1397
                                      1B38  1398 ;************
                                      1B38  1399 pb_rmtstate:
                   54    00000E21'EF   9E  1B38  1400          movab   null_ascic, r4                  ; assume rport not in maint.
                                      1B3F  1401          assume  pb$v_maint eq 0                 ;  state
                            0A 21 AB  E9  1B3F  1402          blbc    pb$b_rstate(r11), 20$           ; branch if rport not in maint.
                   54    00000E25'EF   9E  1B43  1403          movab   maint_ascic, r4                 ; else, set maintenance flag
                               55    64  A2  1B4A  1404          subw    (r4), r5                        ;  and reduce the fill count
                         53    E4E7 CF   9E  1B4D  1405 20$:    movab   pb_rstate, r3                  ; get remote state tbl. addr.
                52    21 AB    02    01  EF  1B52  1406          extzv   #pb$v_state, #pb$s_state, -     ; extract remote port state
                                      1B58  1407                  pb$b_rstate(r11), r2            ;  information
                            00000000'GF   16  1B58  1408          jsb     g^translate_address             ; convert it to ASCIC pointer
                                      1D  13  1B5E  1409          beql    90$                             ; branch if translation failed
                               55    60  82  1B60  1410          subb    (r0), r5                        ; reduce the fill count
                                      1B63  1411          $fao_s  -
                                      1B63  1412                  ctrstr = sb_fao_ascic, -
                                      1B63  1413                  outbuf = (r7), -
                                      1B63  1414                  outlen = (r7), -
                                      1B63  1415                  p1 = r5, -
                                      1B63  1416                  p2 = #2, -
                                      1B63  1417                  p3 = r4, -
                                      1B63  1418                  p4 = r0
```

```
                                05  1B7C  1419          rsb
                          52    21 AB   9E  1B7D  1420 90$:    movab   pb$b_rstate(r11), r2          ; if cannot convert remote
                                        1B81  1421          do_column_entry xb, jmp               ;  status then display value
                                        1B8A  1422
                                        1B8A  1423 ;*************
                                        1B8A  1424 pb_rport_typ:
                          53    E4CA CF   9E  1B8A  1425          movab   pb_rport_type, r3             ; get port type conversion
                                        1B8F  1426          assume  pb$v_port_typ eq 0
                                        1B8F  1427          assume  pb$s_port_typ eq 31
                52  14 AB    80000000 8F   CB  1B8F  1428          bicl3   #^x80000000, -                ; get remote port type value
                                        1B98  1429                  pb$l_rport_typ(r11), r2
                          00000000'GF   16  1B98  1430          jsb     g^translate_address           ; translate port type
                                    0C  13  1B9E  1431          beql    90$                           ; branch if translation failed
                              52    50  D0  1BA0  1432          movl    r0, r2                        ; setup string for display
                                        1BA3  1433          do_column_entry ac, jmp               ; display translated string
                                        1BAC  1434
                                    52  DD  1BAC  1435 90$:    pushl   r2                            ; else, display just the port
                              52    5E  D0  1BAE  1436          movl    sp, r2                        ;  type value
                                        1BB1  1437          do_column_entry xl
                                    8E  D5  1BBA  1438          tstl    (sp)+                         ; cleanup stack
                                        1BBC  1439          rsb
                                        1BBD  1440
                                        1BBD  1441 ;************
                                        1BBD  1442 pb_dualpath:
                                        1BBD  1443          assume  pb$m_dualpath eq <^x80000000>
                52  14 AB    01  1F   EF  1BBD  1444          extzv   #pb$v_dualpath, #1, -         ; get paths flag for remote port
                                        1BC3  1445                  pb$l_rport_typ(r11), r2
                          7E    52  01   C1  1BC3  1446          addl3   #1, r2, -(sp)                 ; add one (there's at least one)
                                52  5E   D0  1BC7  1447          movl    sp, r2                        ; get value pointer
                                        1BCA  1448          do_column_entry ul                    ; display value
                                    8E  D5  1BD3  1449          tstl    (sp)+                         ; cleanup stack
                                        1BD5  1450          rsb
                                        1BD6  1451
                                        1BD6  1452 ;*********
                                        1BD6  1453 pb_cables:
                                        1BD6  1454          assume  pb$v_cur_ps eq 0
                              54    03  D0  1BD6  1455          movl    #3, r4                        ; assume single path port
                          00000E21'EF   9F  1BD9  1456 10$:    pushab  null_ascic
                                F7 54   F5  1BDF  1457          sobgtr  r4, 10$
                                        1BE2  1458
                              55    04  C2  1BE2  1459          subl    #4, r5                        ; adjust fill for path A
                          00000E33'EF   9F  1BE5  1460          pushab  ok_ascic                      ; assume path A is ok
                            09 29 AB    E8  1BEB  1461          blbs    pb$b_p0_sts(r11), 25$         ; branch if path A is ok
                    6E    00000E36'EF   9E  1BEF  1462          movab   bad_ascic, (sp)               ; else, change path A to bad
                                    55  D7  1BF6  1463          decl    r5                            ; adjust fill for bad path
                          00000E2C'EF   9F  1BF8  1464 25$:    pushab  cbl_a_ascic                   ; insert "A-"
                                        1BFE  1465
                                        1BFE  1466          assume  pb$m_dualpath eq <^x80000000>
                             14 AB    D5  1BFE  1467          tstl    pb$l_rport_typ(r11)           ; is this a dual pathed port?
                                30    18  1C01  1468          bgeq    40$                           ; branch if not dual pathed
                              55    05  C2  1C03  1469          subl    #5, r5                        ; adjust fill for path B
                0C AE     00000E33'EF   9E  1C06  1470          movab   ok_ascic, 12(sp)              ; assume path B is ok
                            0A 2A AB    E8  1C0E  1471          blbs    pb$b_p1_sts(r11), 33$         ; branch if path B is ok
                0C AE     00000E36'EF   9E  1C12  1472          movab   bad_ascic, 12(sp)             ; else, change path B to bad
                                    55  D7  1C1A  1473          decl    r5                            ; adjust fill for bad path
                08 AE     00000E2F'EF   9E  1C1C  1474 33$:    movab   cbl_b_ascic, 8(sp)            ; add " B-"
                                        1C24  1475          assume  pb$v_cur_cbl eq 0
```

```
                 0B 28 AB    E8  1C24  1476          blbs    pb$b_cbl_sts(r11), 40$          ; branch if cables not crossed
      10 AE  00000E3A'EF     9E  1C28  1477          movab   crossed_ascic, 16(sp)           ; else, add crossed cables flag
                   55  04    C2  1C30  1478          subl    #4, r5                          ;  and adjust fill count
                               1C33  1479
                       05    DD  1C33  1480 40$:     pushl   #5                              ; set number of ASCICs
                       55    DD  1C35  1481          pushl   r5                              ; set fill count
                   54  5E    D0  1C37  1482          movl    sp, r4                          ; get parameter list pointer
                               1C3A  1483          $faol_s -
                               1C3A  1484                  ctrstr = sb_fao_ascic, -
                               1C3A  1485                  outbuf = (r7), -
                               1C3A  1486                  outlen = (r7), -
                               1C3A  1487                  prmlst = (r4)
                   5E  1C      C0  1C4D  1488          addl    #<7*4>, sp                      ; cleanup stack
                               05  1C50  1489          rsb
                               1C51  1490
                               1C51  1491 ;************
                               1C51  1492 pb_lclstate:
               53  E3BB CF     9E  1C51  1493          movab   pb_state, r3                    ; get port state conversion
                               1C56  1494          assume  pb$v_port_typ eq 0
                 52  12 AB     3C  1C56  1495          movzwl  pb$w_state(r11), r2             ; get local port state
                00000000'GF     16  1C5A  1496          jsb     g^translate_address             ; translate port state
                       0C       13  1C60  1497          beql    90$                             ; branch if translation failed
                   52  50      D0  1C62  1498          movl    r0, r2                          ; setup string for display
                               1C65  1499          do_column_entry ac, jmp                 ; display trans. string
                               1C6E  1500
                 52  12 AB     9E  1C6E  1501 90$:     movab   pb$w_state(r11), r2             ; else, display just the port
                               1C72  1502          do_column_entry xw, jmp                 ;  state value
```

```
                        1C7B  1504          .sbttl  show_ucb, show unit control block (UCB)
                        1C7B  1505 ;---
                        1C7B  1506 ;
                        1C7B  1507 ;        show_ucb
                        1C7B  1508 ;
                        1C7B  1509 ;        This routine shows the unit control block associated
                        1C7B  1510 ;        with a device.
                        1C7B  1511 ;
                        1C7B  1512 ;
                        1C7B  1513 ;        4(ap) = address of DDB in local storage
                        1C7B  1514 ;        8(ap) = address of UCB in local storage
                        1C7B  1515 ;        12(ap)= actual address of UCB
                        1C7B  1516 ;        16(ap)= address of nodename in local storage
                        1C7B  1517 ;        20(ap)= flags longword
                        1C7B  1518 ;
                        1C7B  1519 ;---
                        1C7B  1520
                        1C7B  1521 show_ucb:
                  0FFC  1C7B  1522          .word   ^m<r2,r3,r4,r5,r6,r7,r8,r9,r10,r11>
                        1C7D  1523
                        1C7D  1524          ensure  24
              0C AC     DD  1C95  1525          pushl   12(ap)                  ; push virtual address of UCB
                        1C98  1526
        54    08 AC     D0  1C98  1527          movl    8(ap), r4               ; get local address of UCB
 00001160'EF             9F  1C9C  1528          pushab  unknown                 ; assume the device will be unknown
        52    40 A4     9A  1CA2  1529          movzbl  ucb$b_devclass(r4), r2  ; get device class value
     53    E5F6 CF     9E  1CA6  1530          movab   device_class, r3        ; get conversion table
 00000000'GF             16  1CAB  1531          jsb     g^translate_address     ; get address of device type table
                 12      13  1CB1  1532          beql    90$                     ; branch if no class match
        52    41 A4     9A  1CB3  1533          movzbl  ucb$b_devtype(r4), r2   ; get device type value
           53    50     D0  1CB7  1534          movl    r0, r3                  ; get table address picked above
 00000000'GF             16  1CBA  1535          jsb     g^translate_address     ; get device type ASCIC address
                 03      13  1CC0  1536          beql    90$                     ; branch if no device type matches
              6E    50     D0  1CC2  1537          movl    r0, (sp)                ; else replace unknown with devtype
        52    04 AC     7D  1CC5  1538 90$:     movq    4(ap), r2               ; get DDB and UCB addresses
                        1CC9  1539
5A   00001065'EF        7D  1CC9  1540          movq    one_path, r10           ; assume a single path device which
                        1CD0  1541                                              ; is not a virtual terminal
                        1CD0  1542
   40 A3    42 8F       91  1CD0  1543          cmpb    #dc$_term, -            ; is this a terminal?
                        1CD5  1544                      ucb$b_devclass(r3)
                 49      12  1CD5  1545          bneq    200$                    ; branch if not a terminal
     54    00A0 C3     D0  1CD7  1546          movl    ucb$l_tl_phyucb(r3), r4 ; is this a virtual terminal?
                 3F      13  1CDC  1547          beql    7777$                   ; branch if not a virtual terminal
              0C AC     54  D1  1CDE  1548          cmpl    r4, 12(ap)              ; does virt. term. equal phy. term.?
                 39      13  1CE2  1549          beql    7777$                   ; if yes, then this not a virtual term.
5A   0000111F'EF        7D  1CE4  1550          movq    virtual_terminal, r10   ; it is a virtual terminal
                        1CEB  1551          getmem  ucb$w_unit(r4)          ; get physical terminal's unit number
              7E    51    3C  1CF5  1552          movzwl  r1, -(sp)               ; push than unit number
55   000000B5'EF        9E  1CF8  1553          movab   ddb_2p, r5              ; get work space for phy. DDB copy
                        1CFF  1554          getmem  ucb$l_ddb(r4)           ; get address of DDB for phy. UCB
                        1D09  1555          getmem  (r1), (r5), -           ; get local copy of physical DDB
                        1D09  1556                  #ddb$k_length
              14 A5     9F  1D1A  1557          pushab  ddb$t_name(r5)          ; push address of phy. device name
               00A4     31  1D1D  1558 7777$:   brw     setup_primary           ; go setup virtual terminal name
                        1D20  1559                                              ; (this is also a branch assist)
                        1D20  1560
```

```
                3C A3    10    D3   1D20  1561 200$:    bitl    #dev$m_2p, -                ; dual path device?
                                   1D24  1562                 ucb$l_devchar2(r3)
                         F7    13   1D24  1563          beql    7777$                       ; branch if not dual path
                                   1D26  1564
5A    0000109B'EF        7D         1D26  1565          movq    this_primary, r10           ; assume this path is primary
59    00000060'EF        9E         1D2D  1566          movab   nodnam_2p, r9               ; get node name workarea address
     54    00A8 C3       D0         1D34  1567          movl    ucb$l_dp_altucb(r3), r4     ; is there a local path?
                         1B    12   1D39  1568          bneq    local_2p_device             ; branch if local path
                                   1D3B  1569
                                   1D3B  1570                                             ; both paths through the class driver
                   7E    54 A3    3C   1D3B  1571          movzwl  ucb$w_unit(r3), -(sp)       ; push secondary unit number
                55    00A0 C3       D0   1D3F  1572          movl    ucb$l_dp_ddb(r3), r5        ; get secondary DDB address
             33 14 AC    01       E1   1D44  1573          bbc     #flag_v_alt_path, -         ; if scanning primary DDB chain,
                                   1D49  1574                 20(ap), process_2p_ddb      ;  go join common code
                   55    28 A3    D0   1D49  1575          movl    ucb$l_ddb(r3), r5           ; else, other DDB is primary DDB
5A    000010DD'EF        7D         1D4D  1576          movq    this_secondary, r10         ;  and this is the secondary path
                         26    11   1D54  1577          brb     process_2p_ddb              ; go to common other path code
                                   1D56  1578
                                   1D56  1579 local_2p_device:                            ; only one path through the class driver
                                   1D56  1580          getmem  ucb$w_unit(r4)              ; get other path unit number
                      7E    51 3C    1D60  1581          movzwl  r1, -(sp)                   ; push other path unit number
                                   1D63  1582          getmem  ucb$l_ddb(r4)               ; get other path ddb address
                      55    51 D0    1D6D  1583          movl    r1, r5                      ; save ddb address in right place
                07 3C A3    03       E1   1D70  1584          bbc     #dev$v_cdp, -               ; branch if the path whose UCB is in
                                   1D75  1585                 ucb$l_devchar2(r3), -       ; r3 is the primary path
                                   1D75  1586                 process_2p_ddb              ; else indicate that first name is
5A    000010DD'EF        7D         1D75  1587          movq    this_secondary, r10         ; the secondary path
                                   1D7C  1588
                                   1D7C  1589 process_2p_ddb:
                   54    000000B5'EF    9E   1D7C  1590          movab   ddb_2p, r4                  ; get workarea address for 2p DDB
                                   1D83  1591          getmem  (r5), (r4), -               ; pickup secondary DDB
                                   1D83  1592                 #ddb$k_length
                         14 A4    9F   1D94  1593          pushab  ddb$t_name(r4)              ; push address of secondary device name
                59    00000060'EF    9E   1D97  1594          movab   nodnam_2p, r9               ; get workarea address fo 2p node name
50    34 A4    00000044 8F    C1   1D9E  1595          addl3   #sb$t_nodename, -           ; locate secondary node name
                                   1DA7  1596                 ddb$l_sb(r4), r0
                                   1DA7  1597          getmem  (r0), (r9), -               ; pickup secondary node name
                                   1DA7  1598                 #sb$s_nodename
                   51    51 9A    1DB4  1599          movzbl  r1, r1                      ; convert byte count to long word
                         0B    13   1DB7  1600          beql    setup_primary               ; don't add "$" to null node name
                         51    D6   1DB9  1601          incl    r1                          ; add one for "$"
                      69    51 90    1DBB  1602          movb    r1, (r9)                    ; store count in ASCIC string
                   6941    24 90    1DBE  1603          movb    #^a/$/, (r9)[r1]            ; store "$" in string
                         59    DD   1DC2  1604          pushl   r9                          ; push node name pointer
                                   1DC4  1605
                                   1DC4  1606 setup_primary:
                      54 A3    DD   1DC4  1607          pushl   ucb$w_unit(r3)              ; unit number
                      14 A2    DF   1DC7  1608          pushal  ddb$t_name(r2)              ; generic controller name
                      10 AC    DD   1DCA  1609          pushl   16(ap)                      ; address of nodename
                                   1DCD  1610          printd  r10, (r11)                  ; print device name and UCB
                                   1DD8  1611          skip    1
                      5B    5E DO    1DE1  1612          movl    sp, r11                     ; save pre-allocation stack pointer
                                   1DE4  1613          alloc   80, r4                      ; allocate an output buffer
                      64 A3    DD   1DF6  1614          pushl   ucb$l_sts(r3)               ; push device status value
                   E2BB CF    9F   1DF9  1615          pushab  unit_status                 ; bit definition table
          00000000'EF    02    FB   1DFD  1616          calls   #2,translate_bits           ; translate bits into string
                         54    DD   1E04  1617          pushl   r4                          ; result string
```

```
                64 A3    DD  1E06  1618          pushl   ucb$l_sts(r3)           ; push device status value
                         1E09  1619          print   2,<Device status:   !XL !AS>
          64     50 8F    9A  1E16  1620          movzbl  #80, (r4)              ; refresh output buffer descriptor
                 38 A3    DD  1E1A  1621          pushl   ucb$l_devchar(r3)      ; push device characteristics one
              E33F CF     9F  1E1D  1622          pushab  device_char            ; setup bit definition table
00000000'EF      02     FB  1E21  1623          calls   #2, translate_bits      ; translate bits into string
                 54     DD  1E28  1624          pushl   r4                     ; push result string
                 38 A3    DD  1E2A  1625          pushl   ucb$l_devchar(r3)      ; push device characteristics one
                         1E2D  1626          print   2,<Characteristics: !XL !AS>
          64     50 8F    9A  1E3A  1627          movzbl  #80, (r4)              ; refresh output buffer descriptor
                 3C A3    DD  1E3E  1628          pushl   ucb$l_devchar2(r3)     ; push device characteristics two
              E403 CF     9F  1E41  1629          pushab  device_char_2          ; setup bit definition table
00000000'EF      02     FB  1E45  1630          calls   #2, translate_bits      ; translate bits into string
                 54     DD  1E4C  1631          pushl   r4                     ; push result string
                 3C A3    DD  1E4E  1632          pushl   ucb$l_devchar2(r3)     ; push device characteristics two
                         1E51  1633          print   2,<                !XL !AS>
             5E    5B    D0  1E5E  1634          movl    r11, sp                ; restore stack pointer
                         1E61  1635          skip    1
                         1E6A  1636
                         1E6A  1637 define_ucb_symbols:
                         1E6A  1638          .enable lsb
                         1E6A  1639          make_symbol UCB, 12(ap)
                         1E80  1640          make_symbol SB, ddb$l_sb(r2)
                         1E96  1641          make_symbol ORB, ucb$l_orb(r3)
                         1EAC  1642          make_symbol DDB, ucb$l_ddb(r3)
                         1EC2  1643          make_symbol DDT, ucb$l_ddt(r3)
                         1ED9  1644          make_symbol CRB, ucb$l_crb(r3)
                 60 A3    D5  1EEF  1645          tstl    ucb$l_amb(r3)
                    16    13  1EF2  1646          beql    10$
                         1EF4  1647          make_symbol AMB, ucb$l_amb(r3)
          16 64 A3    08    E1  1F0A  1648 10$:     bbc     #ucb$v_bsy, ucb$l_sts(r3), 20$
                         1F0F  1649          make_symbol IRP, ucb$l_irp(r3)
                         1F25  1650 20$:
                         1F25  1651          .disable lsb
                         1F25  1652
                         1F25  1653 do_ucb_columns:
0000057C'EF      04 AC    D0  1F25  1654          movl    4(ap), ucb_ddb         ; setup local DDB copy address
                         1F2D  1655          print_columns -
                         1F2D  1656                  @8(ap), 12(ap), -
                         1F2D  1657                  ucb_column_1, ucb_column_2, ucb_column_3
             7E    08 AC    7D  1F4C  1658          movq    8(ap),-(sp)            ; push local,real address of UCB
          25 3C A3    05    E1  1F50  1659          bbc     #dev$v_mscp, ucb$l_devchar2(r3), 30$     ; check to see if mscp ser
    00000575'EF      00    B0  1F55  1660          movw    #0,flag_2nd_cddb       ; initialize flag to zero for primary
                56    00BC C3  D0  1F5C  1661          movl    ucb$l_cddb(r3),r6      ; pass the address of the cddb by reg. 6
    0000313C'EF      63    FA  1F61  1662          callg   (r3),show_cddb         ; Display class driver data block
                00000575'EF    B6  1F68  1663          incw    flag_2nd_cddb          ; set to 1 to indicate secondary
                56    00C0 C3  D0  1F6E  1664          movl    ucb$l_2p_cddb(r3),r6   ; pass the address of the secondary cddb
    0000313C'EF      63    FA  1F73  1665          callg   (r3),show_cddb         ; Display class driver data block
    000024C9'EF      02    FB  1F7A  1666 30$:     calls   #2,show_ioq            ; Display I/O request queue
    00002AB1'EF      63    FA  1F81  1667          callg   (r3),show_vcb          ; Display volume control block
                         04  1F88  1668          ret
```

```
                                1F89  1670          .sbttl get_ucb, copy UCB to local storage
                                1F89  1671
                                1F89  1672 ; This routine knows how to load enough of the UCB into local memory for
                                1F89  1673 ; the operations performed above, but how to avoid trying to load more
                                1F89  1674 ; UCB than there really is.
                                1F89  1675 ;
                                1F89  1676 ; Inputs:
                                1F89  1677 ;
                                1F89  1678 ;       r2      real UCB address
                                1F89  1679 ;       r7      address of the place to copy it to
                                1F89  1680 ;
                                1F89  1681 ; Outputs:
                                1F89  1682 ;
                                1F89  1683 ;       r0      status of the copy operation
                                1F89  1684 ;       r1      first longword of copied UCB
                                1F89  1685
                                1F89  1686
                                1F89  1687 get_ucb:
                          3C    BB  1F89  1688          pushr   #^m<r2,r3,r4,r5>        ; save registers
 67  00CC 8F    00    6E    00  2C  1F8B  1689          movc5   #0,(sp),#0,#ucb_size,(r7)  ; zero out the local ucb
                          3C    BA  1F93  1690          popr    #^m<r2,r3,r4,r5>        ; restore registers
                                1F95  1691          trymem  ucb$w_size(r2)          ; get size of this UCB
                       1E 50    E9  1F9F  1692          blbc    r0, 90$                 ; exit now, if error occured
                    51    51    3C  1FA2  1693          movzwl  r1, r1                  ; extend size to a longword
              000000CC 8F       51  D1  1FA5  1694          cmpl    r1, #ucb_size           ; is UCB bigger than the local space?
                          05    15  1FAC  1695          bleq    10$                     ; branch if not bigger
                51    00CC 8F   3C  1FAE  1696          movzwl  #ucb_size, r1           ; else minimize the size
                                1FB3  1697 10$:    trymem  (r2), (r7), r1          ; copy UCB to local storage
                          05    1FC0  1698 90$:    rsb                             ; return to caller
```

```
                     1FC1  1700          .sbttl show_ucb tables & action routines
                     1FC1  1701
                     1FC1  1702          .save
                00001065   1703          .psect  literals,exe,nowrt
                     1065  1704
                     1065  1705 ;
                     1065  1706 ; FAO control strings for locally generated UCB displays
                     1065  1707 ;
                     1065  1708
                     1065  1709 one_path:
          0000106D'00000005' 1065  1710          .address 5, 10$
                     106D  1711 10$:     string  ^\!40<!AC!AC!UW!>!17AC UCB address:  !XL\
                     109B  1712
                     109B  1713 this_primary:
          000010A3'00000008' 109B  1714          .address 8, 10$
                     10A3  1715 10$:     string  ^\!40<!AC!AC!UW (!AC!AC!UW)!>!17AC UCB address:  !XL\
                     10DD  1716
                     10DD  1717 this_secondary:
          000010E5'00000008' 10DD  1718          .address 8, 10$
                     10E5  1719 10$:     string  ^\!40<(!AC!AC!UW) !AC!AC!UW!>!17AC UCB address:  !XL\
                     111F  1720
                     111F  1721 virtual_terminal:
          00001127'00000007' 111F  1722          .address 7, 10$
                     1127  1723 10$:     string  ^\!40<!AC!AC!UW ==> !AC!UW!>!17AC UCB address:  !XL\
                     1160  1724
                     1160  1725 unknown:
    6E 77 6F 6E 6B 6E 55 00' 1160  1726          .ascic  /Unknown/
                       07    1160
                     1168  1727
                     1168  1728 ;
                     1168  1729 ; FAO control strings used by the action routines
                     1168  1730 ;
                     1168  1731
                     1168  1732 ucb_uic_cstr1:
                     1168  1733          string  <[!60W,!60W]>
                     117B  1734
                     117B  1735 ucb_two_bytes:
                     117B  1736          string  <!5XB/!2XB>
                     118C  1737
                     118C  1738 ucb_retry_fao:
                     118C  1739          string  <!#UB/!UB>
                     119C  1740
                     119C  1741 ucb_test_retry_fao:
                     119C  1742          string  <!UB>
                     11A7  1743
                00001FC1   1744          .restore
                     1FC1  1745
                     1FC1  1746 ;
                     1FC1  1747 ;  PRINT_COLUMNS tables for UCB display
                     1FC1  1748 ;
                     1FC1  1749
                     1FC1  1750 ucb_column_1:
                     1FC1  1751          column_list -                           ; column 1 -- allocation
                     1FC1  1752                  ucb$, 17, 8, 3, < -             ;    and other device status
                     1FC1  1753                  <<Owner UIC>,orb_owner,0,10,15>, -      ; Owner UIC
                     1FC1  1754                  <<      PID>,l_pid,xl>, -               ; Owner PID
                     1FC1  1755                  <<Alloc. lock ID>,ucb_lockid,0>, -      ; Allocation lock ID
```

```
                1FC1  1756                -                                       ; Allocation class
                1FC1  1757                <<Alloc. class>,ucb_alloclass,ucb_ddb>, -
                1FC1  1758                <<Class/Type>,ucb_clstyp,0>, -          ; Device class/type
                1FC1  1759                <<Def. buf. size>,w_devbufsiz,uw>, -    ; default buffer size
                1FC1  1760                <<DEVDEPEND>,l_devdepend,xl>, -         ; Device dependent first
                1FC1  1761                <<DEVDEPND2>,l_devdepnd2,xl>, -         ;                  sec.
                1FC1  1762                <<FIPL/DIPL>,ucb_ipls,0>, -             ; Fork / Device IPL
                1FC1  1763                <<Charge PID>,ucb_cpid,0>, -            ; UCB size charge PID
                1FC1  1764                >                                       ; *** end column 1
                2071  1765
                2071  1766        .save
            0000057C  1767        .psect  sdadata,noexe,wrt
                057C  1768 ucb_ddb:
    00000000    057C  1769        .long   0
            00002071  1770        .restore
                2071  1771
                2071  1772 ucb_column_2:
                2071  1773        column_list -                                   ; column 2 -- device activity
                2071  1774                ucb$, 18, 8, 3, < -                     ;    data
                2071  1775                <<Operation count>,l_opcnt,ul>, -       ; operations completed
                2071  1776                <<Error count>,w_errcnt,uw>, -          ; errors recorded count
                2071  1777                <<Reference count>,w_refc,uw>, -        ; reference count
                2071  1778                <<Online count>,ucb_onlcnt,0>, -        ; online count
                2071  1779                <<Retry cnt/max>,ucb_retry,0>, -        ; error retry count/maximum
                2071  1780                <<BOFF>,w_boff,xw>, -                   ; byte offset
                2071  1781                <<Byte count>,w_bcnt,xw>, -             ; byte count
                2071  1782                <<SVAPTE>,l_svapte,xl>, -               ; system virtual addr. PTE
                2071  1783                <<SVPN>,ucb_svpn,0> -                   ; system virtual page number
                2071  1784                <<DEVSTS>,w_devsts,xw>, -               ; Device dependent status
                2071  1785                <<Master CSID>,ucb_mcsid,0>, -          ; Master node's CSID
                2071  1786                <<Int. due time>,ucb_duetim,0>, -       ; Interrupt due time
                2071  1787                <<RWAITCNT>,ucb_rwaitcnt,0> -           ; Reasons to wait count
                2071  1788                >                                       ; *** end column 2
                2151  1789
                2151  1790 ucb_column_3:
                2151  1791        column_list -                                   ; column 3 -- pointer addresses
                2151  1792                ucb$, 15, 8, 0, < -
                2151  1793                <<ORB address>,l_orb,xl>, -             ; Object's rights block
                2151  1794                <<DDB address>,l_ddb,xl>, -             ; Device data block
                2151  1795                <<DDT address>,l_ddt,xl>, -             ; Driver dispatch table
                2151  1796                <<VCB address>,ucb_vcb,0>, -            ; Volume control block
                2151  1797                <<CRB address>,l_crb,xl>, -             ; Channel request block
                2151  1798                <<LNM address>,ucb_lnm,0>, -            ; MBX LNM pointer
                2151  1799                <<AMB address>,l_amb,xl_neq>, -         ; Associated mailbox
                2151  1800                <<PDT address>,ucb_pdt,0>, -            ; Port descriptor table
                2151  1801                <<CDDB address>,ucb_cddb,0>,-           ; Class driver data block
                2151  1802                <<2P_CDDB addr.>,ucb_2pcddb,0>,-        ; Alternate CDDB
                2151  1803                <<2P_DDB address>,ucb_2pddb,0>, -       ; Secondary path DDB
                2151  1804                <<2P_UCB address>,ucb_altucb,0>,-       ; Alternate UCB
                2151  1805                -                                       ; All of the following appear
                2151  1806                -                                       ;  only when the UCB is busy
                2151  1807                <<IRP address>,ucb_bsy,ucb$l_irp>, -    ; I/O request packet
                2151  1808                <<Fork PC>,ucb_bsy,ucb$l_fpc>, -        ; Fork PC
                2151  1809                <<Fork R3>,ucb_bsy,ucb$l_fr3>, -        ; Fork R3
                2151  1810                <<Fork R4>,ucb_bsy,ucb$l_fr4>, -        ; Fork R4
                2151  1811                <<I/O wait queue>,l_ioqfl,q2>, -        ; Pending I/O queue
                2151  1812                >                                       ; *** end column 3
```

```
                     2271  1813
                     2271  1814 ;  The following are all PRINT_COLUMNS action routines for the UCB
                     2271  1815 ;  display.
                     2271  1816 ;
                     2271  1817 ;    Action Routine Inputs:
                     2271  1818 ;
                     2271  1819 ;        R2              value from the COLUMN_LIST entry
                     2271  1820 ;        R5              size of value section for this item
                     2271  1821 ;        R7              address of a descriptor for a scratch string in
                     2271  1822 ;                        which the FAO converted value is to be returned
                     2271  1823 ;        R11             base address of the local UCB copy
                     2271  1824 ;
                     2271  1825 ;    Action Routine Outputs:
                     2271  1826 ;
                     2271  1827 ;        R0              status
                     2271  1828 ;                           lbs ==> use this entry
                     2271  1829 ;                           lbc ==> skip this entry
                     2271  1830 ;        R1 - R5         scratch
                     2271  1831 ;                        all other registers must be preserved
                     2271  1832
                     2271  1833 ;**************
                     2271  1834 ucb_alloclass:  ; if appropriate, return allocation class
5F 38 AB   0E   E1   2271  1835         bbc     #dev$v_fod, -                   ; branch if not a file oriented
                     2276  1836                 ucb$l_devchar(r11), ucb_act_nop ;  device
 52   62   3C   C1   2276  1837         addl3   #ddb$t_allocls, (r2), r2        ; get allocation class address
                     227A  1838 ucb_act_ub:                                      ; display allocation class
                     227A  1839         do_column_entry ub, jmp
                     2283  1840
                     2283  1841 ;**********
                     2283  1842 ucb_altucb:
4D 3C AB   04   E1   2283  1843         bbc     #dev$v_2p, ucb$l_devchar2(r11), - ; branch if device is not
                     2288  1844                 ucb_act_nop                     ;  dual pathed
 52   00A8 CB   DE   2288  1845         moval   ucb$l_dp_altucb(r11), r2        ; alternate UCB address
                62   D5   228D  1846         tstl    (r2)                    ; is there something there?
                44   13   228F  1847         beql    ucb_act_nop             ; branch if nothing there
                     2291  1848         make_symbol -                           ; else,
                     2291  1849                 2P_UCB, (r2)                    ;  make a symbol and
              0087   31   22A6  1850         brw     ucb_act_xl              ;  display it
                     22A9  1851
                     22A9  1852 ;*******
                     22A9  1853 ucb_bsy:
27 64 AB   08   E1   22A9  1854         bbc     #ucb$v_bsy, ucb$l_sts(r11), -   ; exit doing nothing if the
                     22AE  1855                 ucb_act_nop                     ;  UCB is not busy
       52   5B   C0   22AE  1856         addl    r11, r2                         ; else locate cell to return
                     22B1  1857 ucb_act_xl_neq:
                     22B1  1858         do_column_entry xl_neq, jmp             ; display that entry
                     22BB  1859
                     22BB  1860 ;**********
                     22BB  1861 ucb_clstyp:      ; return device class / type
   52   40 AB   9A   22BB  1862         movzbl  ucb$b_devclass(r11), r2         ; return device class
   53   41 AB   9A   22BF  1863         movzbl  ucb$b_devtype(r11), r3          ;  and device type
                26   11   22C3  1864         brb     ucb_ret_2xbytes         ; go join common code
                     22C5  1865
                     22C5  1866 ;********
                     22C5  1867 ucb_cpid:        ; if appropriate, return PID charged for UCB creation
38 AB   00102000 8F   D3   22C5  1868         bitl    #<dev$m_mbx ! dev$m_net>, -     ; is this a mailbox or a
                     22CD  1869                 ucb$l_devchar(r11)              ;  network device
```

```
                     06   13  22CD  1870          beql    ucb_act_nop                     ; if not, assume no PID charged
                52   20 AB   DE  22CF  1871          moval   ucb$l_cpid(r11), r2             ; else, return charged PID
                     5B   11  22D3  1872          brb     ucb_act_xl                      ; using common code
                              22D5  1873
                              22D5  1874 ucb_act_nop:
                     50   D4  22D5  1875          clrl    r0                              ; make this call a nop
                          05  22D7  1876          rsb                                     ; return
                              22D8  1877
                              22D8  1878 ;**********
                              22D8  1879 ucb_duetim:     ; if appropriate, return interrupt due time
             F8 64 AB    00   E1  22D8  1880          bbc     #ucb$v_tim, -                   ; branch if time-out not
                              22DD  1881                  ucb$l_sts(r11), ucb_act_nop     ;  expected
                52   6C AB   DE  22DD  1882          moval   ucb$l_duetim(r11), r2           ; else return due time
                     4D   11  22E1  1883          brb     ucb_act_xl                      ; join common code
                              22E3  1884
                              22E3  1885 ;********
                              22E3  1886 ucb_ipls:       ; return fork / device IPL
                52   0B AB   9A  22E3  1887          movzbl  ucb$b_fipl(r11), r2             ; return fork IPL
                53   5E AB   9A  22E7  1888          movzbl  ucb$b_dipl(r11), r3             ;  and device IPL
                              22EB  1889 ucb_ret_2xbytes:
                              22EB  1890          $fao_s  -                               ; two values as requested
                              22EB  1891                  ctrstr = ucb_two_bytes,  -
                              22EB  1892                  outbuf = (r7), -
                              22EB  1893                  outlen = (r7), -
                              22EB  1894                  p1 = r2, -
                              22EB  1895                  p2 = r3
                          05  2300  1896          rsb                                     ; return
                              2301  1897
                              2301  1898 ;*******
                              2301  1899 ucb_lnm:
         40 AB    A0 8F   91  2301  1900          cmpb    #dc$_mailbox, -                 ; is this a mailbox?
                              2306  1901                  ucb$b_devclass(r11)
                     CD   12  2306  1902          bneq    ucb_act_nop                     ; branch if not a mailbox
                52   74 AB   DE  2308  1903          moval   ucb$l_logadr(r11), r2           ; get logical name pointer
                     62   D5  230C  1904          tstl    (r2)                            ; is something there?
                     C5   13  230E  1905          beql    ucb_act_nop                     ; branch if nothing there
                              2310  1906          make_symbol -                           ; else,
                              2310  1907                  LNM, (r2)                       ;  make a symbol and
                     09   11  2325  1908          brb     ucb_act_xl                      ;  display it
                              2327  1909
                              2327  1910 ;**********
                              2327  1911 ucb_lockid:     ; if sensible, return allocation lock id
         A9 3C AB    00   E1  2327  1912          bbc     #dev$v_clu, -                   ; branch if not a cluster
                              232C  1913                  ucb$l_devchar2(r11), ucb_act_nop ;  accessible device
                52   20 AB   DE  232C  1914          moval   ucb$l_lockid(r11), r2           ; else return lock id
                              2330  1915 ucb_act_xl:
                              2330  1916          do_column_entry xl, jmp
                              2339  1917
                              2339  1918 ;*********
                              2339  1919 ucb_mcsid:
         40 AB    A1 8F   91  2339  1920          cmpb    #dc$_journal, -                 ; is this a journal device?
                              233E  1921                  ucb$b_devclass(r11)
                     95   12  233E  1922          bneq    ucb_act_nop                     ; branch if not a journal dev.
              52   0084 CB   DE  2340  1923          moval   ucb$l_jnl_mcsid(r11), r2        ; else, return master CSID
                     E9   11  2345  1924          brb     ucb_act_xl                      ;  using common code
                              2347  1925
                              2347  1926 ;**********
```

```
                        2347  1927 ucb_onlcnt:
      40 AB   01   91   2347  1928          cmpb    #dc$_disk, ucb$b_devclass(r11)  ; is this a disk device?
              70   12   234B  1929          bneq    ucb_act_nop_a                   ; branch if not a disk
   52   00AE CB   9E   234D  1930          movab   ucb$b_onlcnt(r11), r2           ; else get online count addr.
        FF25    31   2352  1931          brw     ucb_act_ub                      ;  and display it
                        2355  1932
                        2355  1933 ;**********
                        2355  1934 orb_owner:       ; attempt to format owner UIC
              7E   D4   2355  1935          clrl    -(sp)                           ; storage for the UIC from ORB
           52 5E   D0   2357  1936          movl    sp,r2                           ; save address for later
      51    1C AB   D0   235A  1937          movl    ucb$l_orb(r11),r1               ; get real ORB address
              0F   13   235E  1938          beql    10$                             ; display [0,0] if no ORB
                        2360  1939          getmem  orb$l_owner(r1)                 ; get the owner UIC
           03 50   E9   2369  1940          blbc    r0,10$                          ; display [0,0] if unaccessable
           62 51   D0   236C  1941          movl    r1,(r2)                         ; save for $FAO below
                        236F  1942          ASSUME  ORB$L_OWNER EQ 0
                        236F  1943 10$:     $fao_s  -                               ; convert UIC to octal
                        236F  1944                  ctrstr = ucb_uic_cstr1, -
                        236F  1945                  outbuf = (r7), -
                        236F  1946                  outlen = (r7), -
                        236F  1947                  p1 = orb$w_uicgroup(r2), -
                        236F  1948                  p2 = orb$w_uicmember(r2)
              8E   D5   2385  1949          tstl    (sp)+                           ; clean the stack
              05   05   2387  1950          rsb                                     ; return
                        2388  1951
                        2388  1952 ;*******
                        2388  1953 ucb_pdt:
   53   0084 CB   D0   2388  1954          movl    ucb$l_pdt(r11), r3              ; get possible PDT address
              2E   13   238D  1955          beql    ucb_act_nop_a                   ; branch if none
                        238F  1956          getmem  ucb$b_type(r3)                  ; get type and sub-type of PDT
   51   0560 8F   B1   2399  1957          cmpw    #<dyn$c_scs_pdt@8 -             ; is thing pointed to really
                        239E  1958                   + dyn$c_scs>, r1               ;  a PDT?
              1D   12   239E  1959          bneq    ucb_act_nop_a                   ; branch if not really a PDT
   52   0084 CB   DE   23A0  1960          moval   ucb$l_pdt(r11), r2              ; get address of PDT pointer
                        23A5  1961          make_symbol -
                        23A5  1962                  PDT, (r2)                       ;  make a symbol and
        FF73    31   23BA  1963          brw     ucb_act_xl                      ;  display it
                        23BD  1964
                        23BD  1965
                        23BD  1966 ucb_act_nop_a:
              50   D4   23BD  1967          clrl    r0
                   05   23BF  1968          rsb
                        23C0  1969
                        23C0  1970 ;*********
                        23C0  1971 ucb_cddb:
      3C AB   05   E1   23C0  1972          bbc     #dev$v_mscp,ucb$l_devchar2(r11),-
              F8        23C4  1973                  ucb_act_nop_a                   ; branch if device is not mscp serve
   52   00BC CB   DE   23C5  1974          moval   ucb$l_cddb(r11),r2              ; get address of CDDB pointer
                        23CA  1975          make_symbol -
                        23CA  1976                  CDDB, (r2)                      ; make a symbol and
        FF4E    31   23DF  1977          brw     ucb_act_xl                      ; display it
                        23E2  1978
                        23E2  1979 ;*********
                        23E2  1980 ucb_2pcddb:
      3C AB   05   E1   23E2  1981          bbc     #dev$v_mscp,ucb$l_devchar2(r11),-
              D6        23E6  1982                  ucb_act_nop_a                   ; branch if device is not mscp serve
   52   00C0 CB   DE   23E7  1983          moval   ucb$l_2p_cddb(r11),r2           ; alternate CDDB address
```

```
                62   D5  23EC  1984          tstl    (r2)                                ; is there a secondary cddb
                CD   13  23EE  1985          beql    ucb_act_nop_a                       ; branch if not
                         23F0  1986          make_symbol -
                         23F0  1987                  2P_CDDB, (r2)                       ; make a symbol and
             FF28   31   2405  1988          brw     ucb_act_xl                          ; display it
                         2408  1989
                         2408  1990 ;*********
                         2408  1991 ucb_retry:
           0081 CB   95  2408  1992          tstb    ucb$b_ertmax(r11)                   ; is there a retry max?
                AF   13  240C  1993          beql    ucb_act_nop_a                       ; quit now, if no retry max
                7E   D4  240E  1994          clrl    -(sp)                               ; make a little room on stack
           52   5E   D0  2410  1995          movl    sp, r2                              ; save its address
                         2413  1996          $fao_s  -
                         2413  1997                  ctrstr = ucb_test_retry_fao, -      ; determine size of
                         2413  1998                  outbuf = (r7), -                    ;  retry max
                         2413  1999                  outlen = (r2), -
                         2413  2000                  p1 = ucb$b_ertmax(r11)
                6E   D6  2428  2001          incl    (sp)                                ; add one to retry max size
           55   8E   C2  242A  2002          subl    (sp)+, r5                           ; reduce retry cnt. size by that
                         242D  2003          $fao_s  -
                         242D  2004                  ctrstr = ucb_retry_fao, -           ; now produce the whole value
                         242D  2005                  outbuf = (r7), -
                         242D  2006                  outlen = (r7), -
                         242D  2007                  p1 = r5, -
                         242D  2008                  p2 = ucb$b_ertcnt(r11), -
                         242D  2009                  p3 = ucb$b_ertmax(r11)
                05   2448  2010          rsb                                         ; then return
                         2449  2011
                         2449  2012 ;************
                         2449  2013 ucb_rwaitcnt:
        3C AB     05   E1  2449  2014          bbc     #dev$v_mscp,ucb$l_devchar2(r11),-
                    78       244D  2015                  ucb_act_nop_b                       ; branch if device is not mscp serve
           52   56 AB   DE  244E  2016          moval   ucb$w_rwaitcnt(r11),r2              ; get address of wait count
                         2452  2017          make_symbol -
                         2452  2018                  RWAITCNT,(r2)                       ; make a symbol and
                         2467  2019 ucb_act_xw:
                         2467  2020          do_column_entry xw,jmp
                         2470  2021
                         2470  2022 ;********
                         2470  2023 ucb_svpn:
        40 AB     A0 8F   91  2470  2024          cmpb    #dc$_mailbox, -                     ; is this a mailbox? (they
                         2475  2025                  ucb$b_devclass(r11)                 ;  don't have SVPN's)
                    4F   13  2475  2026          beql    ucb_act_nop_b                       ; branch if mailbox
           52   74 AB   DE  2477  2027          moval   ucb$l_svpn(r11), r2                 ; get SVPN address
                  FE33   31  247B  2028          brw     ucb_act_xl_neq                      ;  display it if non-zero
                         247E  2029
                         247E  2030 ;*******
                         247E  2031 ucb_vcb:
38 AB     00280000 8F   D3  247E  2032          bitl    #<dev$m_mnt ! dev$m_dmt>, -         ; is the device mounted?
                         2486  2033                  ucb$l_devchar(r11)
                    3E   13  2486  2034          beql    ucb_act_nop_b                       ; branch if not mounted
           52   34 AB   DE  2488  2035          moval   ucb$l_vcb(r11), r2
                         248C  2036          make_symbol -                               ; else,
                         248C  2037                  VCB, (r2)                           ;  make a symbol and
                  FE8C   31  24A1  2038          brw     ucb_act_xl                          ;  and display it
                         24A4  2039
                         24A4  2040 ;*********
```

```
                        24A4  2041 ucb_2pddb:
1D 3C AB    04    E1    24A4  2042          bbc     #dev$v_2p, ucb$l_devchar2(r11), - ; branch if device is not
                        24A9  2043                  ucb_act_nop_b                     ;  dual pathed
 52   00A0 CB    DE    24A9  2044          moval   ucb$l_dp_ddb(r11), r2              ; secondary DDB address
                        24AE  2045          make_symbol -
                        24AE  2046                  2P_DDB, (r2)                      ; make a symbol and
          FDEB    31    24C3  2047          brw     ucb_act_xl_neq                    ;  display it
                        24C6  2048
                        24C6  2049 ucb_act_nop_b:
            50    D4    24C6  2050          clrl    r0
                  05    24C8  2051          rsb
                        24C9  2052
```

```
                                24C9  2054          .sbttl  show_ioq, Display I/O queue for device
                                24C9  2055 ;---
                                24C9  2056 ;
                                24C9  2057 ;       show_ioq
                                24C9  2058 ;
                                24C9  2059 ;       Display the IRPs and/or CDRP's (if mscp served) in the I/O queues
                                24C9  2060 ;       associated with a specified device.
                                24C9  2061 ;
                                24C9  2062 ; Inputs:
                                24C9  2063 ;
                                24C9  2064 ;       4(ap) = Address of UCB in local storage
                                24C9  2065 ;       8(ap) = Actual address of UCB
                                24C9  2066 ;
                                24C9  2067 ;---
                                24C9  2068          .enabl  lsb
                                24C9  2069
                                24C9  2070 show_ioq:
                          01FC  24C9  2071          .word   ^m<r2,r3,r4,r5,r6,r7,r8>
                                24CB  2072
                     52   04 AC  D0  24CB  2073          movl    4(ap),r2                ; address of UCB
                 34 3C A2   05  E1  24CF  2074          bbc     #dev$v_mscp,ucb$l_devchar2(r2),5$
                                24D4  2075                                              ; only 1 queue if not mscp served
              57   00000471'EF  9E  24D4  2076          movab   cddb,r7                 ; address of Class Driver Data Block
                                24DB  2077          getmem  @ucb$l_cddb(r2),(r7),#cddb$c_length   ; read CDDB
                       4D 50    E9  24EE  2078          blbc    r0,8$                   ; branch if cannot read entire CDDB
           54   00BC C2   00    C1  24F1  2079          addl3   #cddb$l_cdrpqfl,ucb$l_cddb(r2),r4       ; Get real address of cdrp q
                     54   67    D1  24F7  2080          cmpl    cddb$l_cdrpqfl(r7),r4   ; Empty CDRP queue?
                          45    12  24FA  2081          bneq    10$                     ; branch if not empty
           54   00BC C2   3C    C1  24FC  2082 4$:      addl3   #cddb$l_rstrtqfl,ucb$l_cddb(r2),r4  ; Get real address of restart qu
                     54   3C A7  D1  2502  2083          cmpl    cddb$l_rstrtqfl(r7),r4  ; Empty restart queue
                          50    12  2506  2084          bneq    30$                     ; branch if not empty
54   08 AC    0000004C 8F    C1  2508  2085 5$:      addl3   #ucb$l_ioqfl,8(ap),r4   ; Get real address of queue header
                     54   4C A2  D1  2511  2086          cmpl    ucb$l_ioqfl(r2),r4      ; Empty i/o queue?
                          24    12  2515  2087          bneq    7$                      ; Branch if not
                1F 64 A2    08    E0  2517  2088          bbs     #ucb$v_bsy,ucb$w_sts(r2),7$  ; Branch if have IRP
                  00000577'EF  95  251C  2089          tstb    queue_notempty          ; if 0 all queues are empty
                          1A    12  2522  2090          bneq    8$                      ; if 1 then at least 1 queue was not empty
                                2524  2091          skip    1
                                252D  2092          print   0,<!_*** I/O request queue is empty ***>
                          04    253A  2093          ret
                                253B  2094
                        0033    31  253B  2095 7$:      brw     50$                     ; process io request queue
                        0064    31  253E  2096 8$:      brw     90$                     ; clear queue flag and return
                                2541  2097
                                2541  2098 ;       Queue - Class Driver Request Packet Queue (CDRP)
                                2541  2099 ;
                     53   67    D0  2541  2100 10$:     movl    cddb$l_cdrpqfl(r7),r3   ; Get address of first entry in queue
                     56   01    D0  2544  2101          movl    #1,r6                   ; Set state to current
                58   08 AC    D0  2547  2102          movl    8(ap),r8                ; pass actual address of ucb in r8
                        02D5    30  254B  2103 20$:     bsbw    print_cdrp              ; display the contents of the cdrp
                     53   65    D0  254E  2104          movl    cdrp$l_fqfl(r5),r3      ; advance to next entry in queue
                     54   53    D1  2551  2105          cmpl    r3,r4                   ; check to see if another entry exists
                          A6    13  2554  2106          beql    4$                      ; if points back to beginning no more
                          F3    11  2556  2107          brb     20$                     ; process this entry in queue
                                2558  2108 ;
                                2558  2109 ;       Queue - Restarted Class Driver Request Packet Queue (RSTRTQ)
                                2558  2110 ;
```

```
                53   3C A7    D0  2558  2111 30$:    movl    cddb$l_rstrtqfl(r7),r3  ; Get first entry in queue
                   56   02    D0  255C  2112         movl    #2,r6                   ; State is restart
                58   08 AC    D0  255F  2113         movl    8(ap),r8                ; pass actual address of ucb in r8
                     02BD    30  2563  2114 40$:    bsbw    print_cdrp              ; Call routine to display this cdrp
                      53 65    D0  2566  2115         movl    cdrp$l_fqfl(r5),r3      ; Advance to next entry in queue
                      54 53    D1  2569  2116         cmpl    r3,r4                   ; Check to see if no more entries in queue
                         F5    12  256C  2117         bneq    40$                     ; if eql branch to check next queue
                       FF97    31  256E  2118         brw     5$                      ; otherwise still more entries here to proce
                                   2571  2119 ;
                                   2571  2120 ;       Queue - Standard IO Request Packet Queue (IRP)
                                   2571  2121 ;
          00000577'EF    95    2571  2122 50$:    tstb    queue_notempty          ; Check to see if anyone set this flag
                      0A   12    2577  2123         bneq    55$                     ; if 1 then yes so don't bother with it
                       03E6    30  2579  2124         bsbw    queue_title             ; print header for page (IO Request Queue)
       00000577'EF    01    90  257C  2125         movb    #1,queue_notempty       ; set flag to indicate queue was not empty
          0A 64 A2    08    E1  2583  2126 55$:    bbc     #ucb$v_bsy,ucb$w_sts(r2),60$    ; Branch if not busy
                53   58 A2    D0  2588  2127         movl    ucb$l_irp(r2),r3        ; Address of current IRP
                   56   01    D0  258C  2128         movl    #1,r6                   ; Indicate current IRP
                      043E    30  258F  2129         bsbw    print_irp               ; Print line for current IRP
                                   2592  2130
                53   4C A2    D0  2592  2131 60$:    movl    ucb$l_ioqfl(r2),r3      ; Get address of first IRP in queue
                         56    D4  2596  2132         clrl    r6                      ; Indicate not current IRP
                                   2598  2133
                   54   53    D1  2598  2134 70$:    cmpl    r3,r4                   ; end of queue?
                         08    13  259B  2135         beql    90$                     ; Branch if so
                      0430    30  259D  2136         bsbw    print_irp               ; print IRP line
                   53   65    D0  25A0  2137         movl    irp$l_ioqfl(r5),r3      ; Skip to next IRP in queue
                         F3    11  25A3  2138         brb     70$
                                   25A5  2139
          00000577'EF    94    25A5  2140 90$:    clrb    queue_notempty          ; clear flag before we are called again
                                   25AB  2141         status  success
                         04    25B2  2142         ret
                                   25B3  2143         .dsabl  lsb
```

```
                        25B3  2145          .sbttl  show_acpq, display acp queue
                        25B3  2146 ;---
                        25B3  2147 ;
                        25B3  2148 ;       show_acpq
                        25B3  2149 ;
                        25B3  2150 ;       Display the IRP queue associated with the ACP
                        25B3  2151 ;       on the current volume.
                        25B3  2152 ;
                        25B3  2153 ; Inputs:
                        25B3  2154 ;
                        25B3  2155 ;       ap = address of VCB in local storage
                        25B3  2156 ;
                        25B3  2157 ;---
                        25B3  2158          .enabl  lsb
                        25B3  2159
                        25B3  2160 show_acpq:
                 007C   25B3  2161          .word   ^m<r2,r3,r4,r5,r6>
                        25B5  2162
              10 AC     D5   25B5  2163          tstl    vcb$l_aqb(ap)           ; Is there any AQB?
                 03     12   25B8  2164          bneq    10$                     ; Branch if so
                 0188   31   25BA  2165 90$:     brw     95$                     ; Exit
                        25BD  2166
     52 0000041D'EF     9E   25BD  2167 10$:     movab   aqb,r2
                        25C4  2168          getmem  @vcb$l_aqb(ap),(r2),#aqb$c_length  ; Read entire AQB
              E5 50     E9   25D2  2169          blbc    r0,90$
                        25D5  2170          ensure  11
              10 AC     DD   25ED  2171          pushl   vcb$l_aqb(ap)
                        25F0  2172          skip    1
                        25F9  2173          print   1,<!_!_    --- ACP Queue Block (AQB) !XL --->
                        2606  2174          skip    1
              0C A2     D5   260F  2175          tstl    aqb$l_acppid(r2)        ; Is the XQP servicing this queue?
                 53     13   2612  2176          beql    20$                     ; Branch if XQP
                        2614  2177          getmem  @sch$gl_pcbvec,r3       ; Get address of PCB vector
              4D 50     E9   2624  2178          blbc    r0,30$
           51 0C A2     32   2627  2179          cvtwl   aqb$l_acppid(r2),r1     ; Extract process index
            51  6341    DE   262B  2180          moval   (r3)[r1],r1             ; Point to PCB address entry
                        262F  2181          getmem  (r1)                    ; Read PCB address
              39 50     E9   2638  2182          blbc    r0,30$
     53 00000000'EF     9E   263B  2183          movab   buffer,r3
                        2642  2184          getmem  pcb$t_lname(r1),(r3),#16  ; Read 16-byte process name
              21 50     E9   2650  2185          blbc    r0,30$
              0C A2     DD   2653  2186          pushl   aqb$l_acppid(r2)        ; Process PID
                 53     DD   2656  2187          pushl   r3                      ; Address of ASCIC string
                        2658  2188          print   1,<ACP requests are serviced by process !AC whose PID is !XL>
                 0D     11   2665  2189          brb     30$
                        2667  2190
                        2667  2191 20$:     print   0,<ACP requests are serviced by the eXtended Qio Processor (XQP)>
                        2674  2192
                        2674  2193 30$:     skip    1
                        267D  2194          alloc   80                      ; 80 byte string buffer
           7E   14 A2     9A   268C  2195          movzbl  aqb$b_status(r2),-(sp)  ; ACP status
              E504 CF     9F   2690  2196          pushab  acp_status              ; Bit definition table
00000000'EF    02     FB   2694  2197          calls   #2,translate_bits       ; Translate bits into names
                 5E     DD   269B  2198          pushl   sp                      ; Address of string descriptor
              14 A2     DD   269D  2199          pushl   aqb$b_status(r2)        ; ACP status
                        26A0  2200          print   2,<Status: !XB !AS>
                        26AD  2201          skip    1
```

```
                                26B6  2202          print_columns -
                                26B6  2203                  (r2), vcb$l_aqb(ap), -
                                26B6  2204                  aqb_column_1, aqb_column_2, aqb_column_3
                                26D4  2205
                                26D4  2206          skip    1
          53    62    D0      26DD  2207          movl    aqb$l_acpqfl(r2),r3     ; Get address of first IRP
    54 10 AC    00    C1      26E0  2208          addl3   #aqb$l_acpqfl,vcb$l_aqb(ap),r4  ; Get real address of queuehead
          54    53    D1      26E5  2209          cmpl    r3,r4                   ; Empty ACP queue?
                0E    12      26E8  2210          bneq    70$                     ; Branch if not
                              26EA  2211          print   0,<!_*** ACP request queue is empty ***>
                      04      26F7  2212          ret
                              26F8  2213
                              26F8  2214 70$:     ensure  8
                              2710  2215          print   0,<!_!_!_!_ACP request queue>
                              271D  2216          print   0,<!_!_!_!_----------------->
                              272A  2217          skip    1
                0260  30      2733  2218          bsbw    irp_heading             ; Print heading line
                  56  D4      2736  2219          clrl    r6                      ; Indicate not current IRP
                              2738  2220
          54    53  D1      2738  2221 80$:     cmpl    r3,r4                   ; End of queue?
                08  13      273B  2222          beql    95$                     ; Branch if so
                0290  30      273D  2223          bsbw    print_irp               ; Print IRP line
          53    65  D0      2740  2224          movl    irp$l_ioqfl(r5),r3      ; skip to next IRP
                F3  11      2743  2225          brb     80$
                              2745  2226
                              2745  2227 95$:     status  success
                      04      274C  2228          ret
                              274D  2229          .dsabl  lsb
                              274D  2230          .sbttl volume control block tables & action routines
                              274D  2231
                              274D  2232 ;  The following are all PRINT_COLUMNS action routines for the show_vcb
                              274D  2233 ;  block displays.
                              274D  2234 ;
                              274D  2235 ;    Action Routine Inputs:
                              274D  2236 ;
                              274D  2237 ;       R2              value from the COLUMN_LIST entry
                              274D  2238 ;       R5              size of value section for this item
                              274D  2239 ;       R7              address of a descriptor for a scratch string in
                              274D  2240 ;                       which the FAO converted value is to be returned
                              274D  2241 ;       R11             base address of the local UCB copy
                              274D  2242 ;
                              274D  2243 ;    Action Routine Outputs:
                              274D  2244 ;
                              274D  2245 ;       R0              status
                              274D  2246 ;                          lbs ==> use this entry
                              274D  2247 ;                          lbc ==> skip this entry
                              274D  2248 ;       R1 - R5         scratch
                              274D  2249 ;                       all other registers must be preserved
                              274D  2250
                              274D  2251
                              274D  2252 ;
                              274D  2253 ; PRINT_COLUMNS tables for AQB display
                              274D  2254 ;
                              274D  2255
                              274D  2256 aqb_column_1:
                              274D  2257          column_list -
                              274D  2258                  aqb$, 16, 8, 4, < -
```

```
                     274D  2259                <<Mount count>,b_mntcnt,ub>, -
                     274D  2260                >
                     276D  2261
                     276D  2262 aqb_column_2:
                     276D  2263         column_list -
                     276D  2264                aqb$, 16, 8, 4, < -
                     276D  2265                <<ACP type>,aqb_type,0,14,10>, -
                     276D  2266                <<ACP class>,aqb_class,0>, -
                     276D  2267                >
                     279D  2268
                     279D  2269 aqb_column_3:
                     279D  2270         column_list -
                     279D  2271                aqb$, 16, 8, 0, < -
                     279D  2272                <<Linkage>,l_link,xl_neg>, -
                     279D  2273                <<Request queue>,l_acpqfl,q2>, -
                     279D  2274                >
                     27CD  2275
                     27CD  2276 ;**********
                     27CD  2277 aqb_type:
   52    15 AB   9A  27CD  2278         movzbl  aqb$b_acptype(r11), r2          ; get ACP type
53    E3EB CF   9E  27D1  2279         movab   aqb_acptype, r3                 ; get translate table
 00000000'GF   16  27D6  2280         jsb     g^translate_address             ; translate ACP class
         0C   13  27DC  2281         beql    90$                             ; branch if translate failed
      52   50   D0  27DE  2282         movl    r0, r2                          ; setup translated string
                     27E1  2283         do_column_entry ac, jmp                 ; display translation
                     27EA  2284
   52    15 AB   9E  27EA  2285 90$:     movab   aqb$b_acptype(r11), r2          ; else, get type address
                     27EE  2286         do_column_entry ub, jmp                 ;  just display the value
                     27F7  2287
                     27F7  2288 ;**********
                     27F7  2289 aqb_class:
   52    16 AB   9A  27F7  2290         movzbl  aqb$b_class(r11), r2            ; get ACP class
         19   13  27FB  2291         beql    90$                             ; branch if none
53    D88F CF   9E  27FD  2292         movab   ddb_acpclass, r3                ; get translate table
 00000000'GF   16  2802  2293         jsb     g^translate_address             ; translate ACP class
         0C   13  2808  2294         beql    90$                             ; branch if translate failed
      52   50   D0  280A  2295         movl    r0, r2                          ; setup translated string
                     280D  2296         do_column_entry ac, jmp                 ; display translation
                     2816  2297
   52    13 AB   9E  2816  2298 90$:     movab   ddb$b_acpclass(r11), r2         ; else, get class address
                     281A  2299         do_column_entry ub, jmp                 ;  just display the value
```

```
                                2823  2301          .sbttl  print_cdrp, print a single CDRP block
                                2823  2302 ;---
                                2823  2303
                                2823  2304          .enabl  lsb
                                2823  2305 ;
                                2823  2306 ;       Subroutine to print information for a single CDRP block
                                2823  2307 ;
                                2823  2308 ; Inputs:
                                2823  2309 ;
                                2823  2310 ;       r3 = Dump address of CDRP block
                                2823  2311 ;       r6 = 2, if restarted CDRP, 1 if current CDRP
                                2823  2312 ;       r8 = Actual address of UCB
                                2823  2313 ;
                                2823  2314 ; Outputs:
                                2823  2315 ;
                                2823  2316 ;       r5 = Address of CDRP in local storage
                                2823  2317 ;
                                2823  2318 ;---
                                2823  2319
                                2823  2320 print_cdrp:
                                2823  2321          ensure  3
                          56    DD  283B  2322          pushl   r6                      ; save r6
56  53  FFFFFFA0 8F   C1  283D  2323          addl3   #cdrp$l_ioqfl,r3,r6     ; get start of cdrp at most negative offset
    55  00000289'EF   9E  2845  2324          movab   cdrp,r5                 ; get address of local cdrp
                          284C  2325          getmem  (r6),(r5),#cdrp_length  ; read entire CDRP
                       56 8ED0  285D  2326          popl    r6                      ; restore r6
                    03 50   E8  2860  2327          blbs    r0,5$                   ; check status
                      00FB  31  2863  2328          brw     90$                     ; return
    55  FFFFFFA0 8F   C2  2866  2329 5$:     subl2   #cdrp$l_ioqfl,r5        ; actual start of CDRP
     BC A5   58   D1  286D  2330          cmpl    r8,cdrp$l_ucb(r5)       ; check to see if this request is from this
                    03  13  2871  2331          beql    10$                     ; if equal yes so process it
                    00EB  31  2873  2332          brw     90$                     ; return
          00000577'EF   95  2876  2333 10$:     tstb    queue_notempty          ; Check to see if anyone set this flag
                      0A  12  287C  2334          bneq    15$                     ; If 1 then yes so don't bother with it
                    00E1  30  287E  2335          bsbw    queue_title             ; Otherwise display the header for page
    00000577'EF     01  90  2881  2336          movb    #1,queue_notempty       ; set flag to say this queue was not empty
                 CA A5  DD  2888  2337 15$:     pushl   cdrp$w_sts(r5)          ; request status
                 C4 A5  DD  288B  2338          pushl   cdrp$l_iosb(r5)         ; address of IOSB
                 B0 A5  DD  288E  2339          pushl   cdrp$l_ast(r5)          ; address of AST routine
                 C2 A5  DD  2891  2340          pushl   cdrp$b_efn(r5)          ; Event flag number
                 B8 A5  DD  2894  2341          pushl   cdrp$l_wind(r5)         ; Address of WCB
                 C0 A5  DD  2897  2342          pushl   cdrp$w_func(r5)         ; Function code
                 C8 A5  DD  289A  2343          pushl   cdrp$w_chan(r5)         ; Channel number
50    AB A5   02   00  EF  289D  2344          extzv   #irp$v_mode,#irp$s_mode,cdrp$b_rmod(r5),r0
          5553454B 8F  DD  28A3  2345          pushl   #^a'KESU'               ; Possible user modes
                6E40  9F  28A9  2346          pushab  (sp)[r0]                ; Address of string
                   01  DD  28AC  2347          pushl   #1                      ; Length of string
                 AC A5  DD  28AE  2348          pushl   cdrp$l_pid(r5)          ; Process identification
                    53  DD  28B1  2349          pushl   r3                      ; Address of CDRP
          00000043 8F  DD  28B3  2350          pushl   #^a'C'                  ; String containing space
                    5E  DD  28B9  2351          pushl   sp                      ; Address of string
                    01  DD  28BB  2352          pushl   #1                      ; Length of string
              56    01  D1  28BD  2353          cmpl    #1,r6                   ; check if current CDRP
                    08  13  28C0  2354          beql    20$                     ; branch if not
 08 AE    00000052 8F  D0  28C2  2355          movl    #^a'R',8(sp)            ; Flag current CDRP being done
                           28CA  2356 20$:     print   15,< !AD!+    !XL  !XL  !AD!+  !XW  !XW  !XL  !2UB  !XL  !XL  !XW>
                           28D7  2357
```

```
                                28D7  2358 ; save a few registers now.  Then we will allocate stack space for two output
                                28D7  2359 ; buffers.  Translate the class driver's flags field, the status field of the
                                28D7  2360 ; cdrp, and the function code for the request.  Then display.
                                28D7  2361 ;
                7E    52    7D  28D7  2362          movq    r2, -(sp)                       ; save some registers
                                28DA  2363          alloc   80,r2                           ; 80 byte output buffer for request status
                                28EC  2364          alloc   80,r3                           ; another buffer of 80 bytes
             7E    40 A5    DO  28FE  2365          movl    cdrp$l_dutuflags(r5),-(sp)      ; cdrp flags
                E0D2 CF    9F  2902  2366          pushab  cdrp_dutuflags                  ; bit definition table
       00000000'EF    02    FB  2906  2367          calls   #2,translate_bits               ; translate bits to names
                      5E    DD  290D  2368          pushl   sp                              ; push the address of descriptor
                   04 A2    DD  290F  2369          pushl   4(r2)                           ; push descriptor for request status
                      62    DD  2912  2370          pushl   (r2)                            ; push size of this buffer
             7E    CA A5    3C  2914  2371          movzwl  cdrp$w_sts(r5),-(sp)            ; request status
                E0F4 CF    9F  2918  2372          pushab  request_status                  ; bit definition table
       00000000'EF    02    FB  291C  2373          calls   #2,translate_bits               ; translate bits to names
                      5E    DD  2923  2374          pushl   sp                              ; address of string descriptor
             00000E21'EF    9F  2925  2375          pushab  null_ascic                      ; assume function will not translate
    52    C0 A5    06    00  EF  292B  2376          extzv   #io$v_fcode, #io$s_fcode, -
                                2931  2377                  cdrp$w_func(r5), r2             ; get function code
          53    E153 CF    9E  2931  2378          movab   io_function, r3                 ; get translation table
           00000000'GF    16  2936  2379          jsb     g^translate_address             ; translate function to text
                      03    13  293C  2380          beql    33$                             ; branch if translate failed
                   6E    50  DO  293E  2381          movl    r0, (sp)                        ; setup translated function
                                2941  2382 33$:     print   3,<!_!AC !AS!+!+ !AS>           ; print translated information
                                294E  2383          skip    1                               ; advance
       5E    000000B8 8F    C0  2957  2384          addl    #184,sp                         ; deallocate translate buffers
                   52    8E  7D  295E  2385          movq    (sp)+, r2                       ; restore saved registers
                                2961  2386
                      05        2961  2387 90$:     rsb
```

```
                        2962  2389          .sbttl  print_irp, print a single IRP block
                        2962  2390 ;---
                        2962  2391
                        2962  2392          .enabl  lsb
                        2962  2393 ;
                        2962  2394 ;       Subroutine to print information for a single IRP block
                        2962  2395 ;
                        2962  2396 ; Inputs:
                        2962  2397 ;
                        2962  2398 ;       r3 = Dump address of IRP block
                        2962  2399 ;       r6 = 1 if current IRP, else 0 if pending
                        2962  2400 ;
                        2962  2401 ; Outputs:
                        2962  2402 ;
                        2962  2403 ;       r5 = Address of IRP in local storage
                        2962  2404 ;
                        2962  2405 ;---
                        2962  2406
                        2962  2407 queue_title:
                        2962  2408          ensure  10
                        297A  2409          print   0,<!_!_!_!_I/O request queue>
                        2987  2410          print   0,<!_!_!_!_---------------->
                        2994  2411          skip    1
17 3C A2    05    E0    299D  2412          bbs     #dev$v_mscp,ucb$l_devchar2(r2),cdrp_heading
                        29A2  2413 irp_heading:
                        29A2  2414          print   0,<STATE    IRP      PID   MODE CHAN  FUNC    WCB     EFN    AST
                        29AF  2415          skip    1
                  05    29B8  2416 999$:    rsb
                        29B9  2417
                        29B9  2418 cdrp_heading:
                        29B9  2419          print   0,<STATE CDRP/IRP    PID   MODE CHAN  FUNC    WCB     EFN    AST
                        29C6  2420          skip    1
                  05    29CF  2421          rsb
                        29D0  2422
                        29D0  2423 print_irp:
                        29D0  2424          ensure  3
     55   000001C5'EF   9E    29E8  2425          movab   irp,r5
                        29EF  2426          getmem  (r3),(r5),#irp$c_length ; read entire IRP
               B5 50    E9    2A00  2427          blbc    r0,999$
               2A A5    DD    2A03  2428          pushl   irp$w_sts(r5)           ; request status
               24 A5    DD    2A06  2429          pushl   irp$l_iosb(r5)          ; address of IOSB
               10 A5    DD    2A09  2430          pushl   irp$l_ast(r5)           ; address of AST routine
               22 A5    DD    2A0C  2431          pushl   irp$b_efn(r5)           ; Event flag number
               18 A5    DD    2A0F  2432          pushl   irp$l_wind(r5)          ; Address of WCB
               20 A5    DD    2A12  2433          pushl   irp$w_func(r5)          ; Function code
               28 A5    DD    2A15  2434          pushl   irp$w_chan(r5)          ; Channel number
50   0B A5   02    00    EF    2A18  2435          extzv   #irp$v_mode,#irp$s_mode,irp$b_rmod(r5),r0
          5553454B 8F    DD    2A1E  2436          pushl   #^a'KESU'               ; Possible user modes
              6E40 9F    2A24  2437          pushab  (sp)[r0]                ; Address of string
                 01    DD    2A27  2438          pushl   #1                      ; Length of string
               0C A5    DD    2A29  2439          pushl   irp$l_pid(r5)           ; Process identification
                 53    DD    2A2C  2440          pushl   r3                      ; Address of IRP
          00000050 8F    DD    2A2E  2441          pushl   #^a'P'                  ; String containing space
                 5E    DD    2A34  2442          pushl   sp                      ; Address of string
                 01    DD    2A36  2443          pushl   #1                      ; Length of string
                 56    D5    2A38  2444          tstl    r6                      ; check if current IRP
                 08    13    2A3A  2445          beql    20$                     ; branch if not
```

```
                                2A3C  2446          movl    #^a'C',8(sp)                    ; Flag current IRP being done
                                2A44  2447 20$:     print   15,< !AD!+   !XL  !XL  !AD!+   !XW  !XW  !XL  !2UB  !XL  !XL  !XW>
                                2A51  2448
                     7E   52    7D   2A51  2449          movq    r2, -(sp)                       ; save some registers
                                2A54  2450          alloc   80                              ; 80 byte output buffer
                   7E   2A A5    3C   2A63  2451          movzwl  irp$w_sts(r5),-(sp)     ; request status
                      DFA5 CF    9F   2A67  2452          pushab  request_status          ; bit definition table
          00000000'EF    02    FB   2A6B  2453          calls   #2,translate_bits       ; translate bits to names
                         5E    DD   2A72  2454          pushl   sp                      ; address of string descriptor
                00000E21'EF    9F   2A74  2455          pushab  null_ascic              ; assume function will not translate
     52   20 A5    06   00    EF   2A7A  2456          extzv   #io$v_fcode, #io$s_fcode, -
                                2A80  2457                  irp$w_func(r5), r2      ; get function code
                53   E004 CF    9E   2A80  2458          movab   io_function, r3         ; get translation table
                 00000000'GF    16   2A85  2459          jsb     g^translate_address     ; translate function to text
                         03    13   2A8B  2460          beql    33$                     ; branch if translate failed
                      6E   50    D0   2A8D  2461          movl    r0, (sp)                ; setup translated function
                                2A90  2462 33$:     print   2,<!_!AC !AS>           ; print translated information
                                2A9D  2463          skip    1
          5E    00000058 8F    C0   2AA6  2464          addl    #88,sp                  ; deallocate translate buffer
                      52   8E    7D   2AAD  2465          movq    (sp)+, r2               ; restore saved registers
                                2AB0  2466
                                05   2AB0  2467 90$:     rsb
                                2AB1  2468
                                2AB1  2469          .dsabl  lsb
```

```
                        2AB1  2471          .sbttl  show_vcb, Display Volume Control Block (VCB)
                        2AB1  2472 ;---
                        2AB1  2473 ;
                        2AB1  2474 ;       show_vcb
                        2AB1  2475 ;
                        2AB1  2476 ;       Display the Volume Control Block (VCB)
                        2AB1  2477 ;
                        2AB1  2478 ; Inputs:
                        2AB1  2479 ;
                        2AB1  2480 ;       ap = Address of UCB in local storage
                        2AB1  2481 ;
                        2AB1  2482 ;---
                        2AB1  2483
                        2AB1  2484 show_vcb:
                   083C 2AB1  2485          .word   ^m<r2,r3,r4,r5,r11>
                        2AB3  2486
                 34 AC   D5 2AB3  2487          tstl    ucb$l_vcb(ap)           ; any VCB for this unit?
                    08   12 2AB6  2488          bneq    10$                     ; Branch if so
                        2AB8  2489
                        2AB8  2490 90$:     status  success
                    04  2ABF  2491          ret
                        2AC0  2492
              F3 38 AC  06  E0 2AC0  2493 10$:     bbs     #dev$v_spl,ucb$l_devchar(ap),90$  ; ignore VCB for
                        2AC5  2494                                              ; spooled devices (wrong usage)
           52 00000331'EF   9E 2AC5  2495          movab   vcb,r2
                        2ACC  2496          getmem  @ucb$l_vcb(ap),(r2),#vcb$c_length ; read entire VCB
                 D7 50   E9 2ADE  2497          blbc    r0,90$
              11 0A A2   91 2AE1  2498          cmpb    vcb$b_type(r2),#dyn$c_vcb       ; Check if block valid
                    D1   12 2AE5  2499          bneq    90$                     ; Exit if not valid type
                        2AE7  2500
                        2AE7  2501          ensure  12
                        2AFF  2502          skip    1
                 34 AC   DD 2B08  2503          pushl   ucb$l_vcb(ap)
                        2B0B  2504          print   1,<!_!_--- Volume Control Block (VCB) !XL --->
                        2B18  2505          skip    1
                        2B21  2506          alloc   80                      ; 80 byte output buffer
                 5B   5E   D0 2B30  2507          movl    sp, r11                 ; save descriptor address
                        2B33  2508
                        2B33  2509          ; use different display stratagies for different VCB types
           03 38 AC  0D   E1 2B33  2510          bbc     #dev$v_net, ucb$l_devchar(ap), 20$
                   028D   31 2B38  2511          brw     vcb_net
                        2B3B  2512 20$:     dispatch ucb$b_devclass(ap), type=B, prefix=dc$_, <-
                        2B3B  2513                  <disk,vcb_disk>, -
                        2B3B  2514                  <tape,vcb_tape>, -
                        2B3B  2515                  <journal,vcb_journal> -
                        2B3B  2516                  >
                        2C83  2517          status  success
                    04  2C8A  2518          ret
                        2C8B  2519
                        2C8B  2520 vcb_disk:
           03 38 AC  18   E1 2C8B  2521          bbc     #dev$v_for, -           ; Is this a foreign mounted disk?
                        2C90  2522                  ucb$l_devchar(ap), 20$
                   008A   31 2C90  2523          brw     vcb_foreign             ; Branch if foreign.
                0080 C2   DF 2C93  2524 20$:     pushal  vcb$t_volcknam(r2)      ; Address of volume lock name
                    0C    DD 2C97  2525          pushl   #12                     ; Length of volume lock name
                  14 A2   DF 2C99  2526          pushal  vcb$t_volname(r2)       ; Address of volume name
                    0C    DD 2C9C  2527          pushl   #12                     ; Length of volume name
```

```
                          2C9E  2528          print   2,<Volume: !AD     Lock name: !AF>
          5E    5B    D0  2CAB  2529          movl    r11, sp                         ; Setup scratch area
       7E   0B A2    9A  2CAE  2530          movzbl  vcb$b_status(r2), -(sp) ; Volume status
         DB22 CF    9F  2CB2  2531          pushab  vcb_disk_status         ; Bit definition table
00000000'EF   02    FB  2CB6  2532          calls   #2, translate_bits      ; Translate bits to names
              5E    DD  2CBD  2533          pushl   sp                      ; Address of output descriptor
           0B A2    DD  2CBF  2534          pushl   vcb$b_status(r2)
                        2CC2  2535          print   2,<Status:  !XB !AS>
          5E    5B    D0  2CCF  2536          movl    r11, sp                         ; Setup scratch area
       7E   53 A2    9A  2CD2  2537          movzbl  vcb$b_status2(r2), -(sp); Volume status, second byte
         DB46 CF    9F  2CD6  2538          pushab  vcb_disk_status2        ; Bit definition table
00000000'EF   02    FB  2CDA  2539          calls   #2, translate_bits      ; Translate bits to names
              5E    DD  2CE1  2540          pushl   sp                      ; Address of output descriptor
           53 A2    DD  2CE3  2541          pushl   vcb$b_status2(r2)
                        2CE6  2542          print   2,<Status2: !XB !AS>
                        2CF3  2543          skip    1
                        2CFC  2544          print_columns -
                        2CFC  2545                  (r2), ucb$l_vcb(ap), -
                        2CFC  2546                  vcb_disk_col_1, vcb_disk_col_2, vcb_disk_col_3
            0129    31  2D1A  2547          brw     vcb_show_acpq
                        2D1D  2548
                        2D1D  2549          .enable lsb
                        2D1D  2550 vcb_tape:
                        2D1D  2551 vcb_foreign:
           14 A2    DF  2D1D  2552          pushal  vcb$t_volname(r2)       ; Address of volume name
              0C    DD  2D20  2553          pushl   #12                     ; Length of volume name
                        2D22  2554          print   1,<Volume: !AD>
    22 38 AC    18    E1  2D2F  2555          bbc     #dev$v_for, -           ; Is this a foreign mounted volume?
                        2D34  2556                  ucb$l_devchar(ap), 20$  ; Branch if not foreign.
                        2D34  2557          skip    1
                        2D3D  2558          print   0,<!_!_!_Volume is foreign mounted>
                        2D4A  2559          skip    1
            00F0    31  2D53  2560          brw     vcb_show_acpq           ; Go try to do AQB, ha ha.
          5E    5B    D0  2D56  2561 20$:    movl    r11, sp                         ; Setup scratch area
       7E   0B A2    9A  2D59  2562          movzbl  vcb$b_status(r2), -(sp) ; Volume status
         DAEF CF    9F  2D5D  2563          pushab  vcb_tape_status         ; Bit definition table
00000000'EF   02    FB  2D61  2564          calls   #2, translate_bits      ; Translate bits to names
              5E    DD  2D68  2565          pushl   sp                      ; Address of output descriptor
           0B A2    DD  2D6A  2566          pushl   vcb$b_status(r2)
                        2D6D  2567          print   2,<Status: !4XB !AS>
          5E    5B    D0  2D7A  2568          movl    r11, sp                         ; Setup scratch area
       7E   2C A2    3C  2D7D  2569          movzwl  vcb$w_mode(r2), -(sp)   ; Volume operating mode
         DB13 CF    9F  2D81  2570          pushab  vcb_tape_mode           ; Bit definition table
00000000'EF   02    FB  2D85  2571          calls   #2, translate_bits      ; Translate bits to names
              5E    DD  2D8C  2572          pushl   sp                      ; Address of output descriptor
           2C A2    DD  2D8E  2573          pushl   vcb$w_mode(r2)
                        2D91  2574          print   2,<Mode:   !4XW !AS>
                        2D9E  2575          skip    1
                        2DA7  2576          print_columns -
                        2DA7  2577                  (r2), ucb$l_vcb(ap), -
                        2DA7  2578                  vcb_tape_col_1, vcb_tape_col_2, vcb_tape_col_3
            007E    31  2DC5  2579          brw     vcb_show_acpq
                        2DC8  2580
                        2DC8  2581          .disable lsb
                        2DC8  2582
                        2DC8  2583 vcb_net:
                        2DC8  2584          print_columns -
```

```
                          2DC8  2585                  (r2), ucb$l_vcb(ap), -
                          2DC8  2586                  vcb_net_col_1, vcb_net_col_2, vcb_net_col_3
                 005D  31 2DE6  2587          brw     vcb_show_acpq
                          2DE9  2588
                          2DE9  2589 vcb_journal:
              14 A2  DF   2DE9  2590          pushal  vcb$t_volname(r2)       ; Address of journalname
                 0C  DD   2DEC  2591          pushl   #12                     ; Length of journal name
                          2DEE  2592          print   1,<Journal name: !AD>
           5E  5B  D0   2DFB  2593          movl    r11, sp                 ; Setup scratch area
        7E   24 A2  D0   2DFE  2594          movl    vcb$l_jnl_char(r2),-(sp); Journal characteristics
           DB02 CF  9F   2E02  2595          pushab  vcb_journal_char        ; Bit definition table
00000000'EF    02  FB   2E06  2596          calls   #2, translate_bits      ; Translate bits to names
              5E  DD   2E0D  2597          pushl   sp                      ; Address of output descriptor
              24 A2  DD   2E0F  2598          pushl   vcb$l_jnl_char(r2)
                          2E12  2599          print   2,<Characteristics: !XL !AS>
                          2E1F  2600          skip    1
                          2E28  2601          print_columns -
                          2E28  2602                  (r2), ucb$l_vcb(ap), -
                          2E28  2603                  vcb_jnl_col_1, vcb_jnl_col_2, vcb_jnl_col_3
                          2E46  2604
                          2E46  2605 vcb_show_acpq:
         F768 CF  62   FA 2E46  2606          callg   (r2),show_acpq          ; Display ACP queue (if any)
                    04    2E4B  2607          ret
```

```
2E4C  2609          .sbttl volume control block tables & action routines
2E4C  2610
2E4C  2611 ;  The following are all PRINT_COLUMNS action routines for the show_vcb
2E4C  2612 ;  block displays.
2E4C  2613 ;
2E4C  2614 ;    Action Routine Inputs:
2E4C  2615 ;
2E4C  2616 ;        R2              value from the COLUMN_LIST entry
2E4C  2617 ;        R5              size of value section for this item
2E4C  2618 ;        R7              address of a descriptor for a scratch string in
2E4C  2619 ;                        which the FAO converted value is to be returned
2E4C  2620 ;        R11             base address of the local UCB copy
2E4C  2621 ;
2E4C  2622 ;    Action Routine Outputs:
2E4C  2623 ;
2E4C  2624 ;        R0              status
2E4C  2625 ;                           lbs ==> use this entry
2E4C  2626 ;                           lbc ==> skip this entry
2E4C  2627 ;        R1 - R5         scratch
2E4C  2628 ;                        all other registers must be preserved
2E4C  2629
2E4C  2630
2E4C  2631 ;
2E4C  2632 ; PRINT_COLUMNS tables for disk VCB displays
2E4C  2633 ;
2E4C  2634
2E4C  2635 vcb_disk_col_1:
2E4C  2636          column_list -
2E4C  2637                  vcb$, 16, 8, 4, < -
2E4C  2638                  <<Mount count>,w_mcount,uw>, -
2E4C  2639                  <<Transactions>,w_trans,uw>, -
2E4C  2640                  <<Free blocks>,l_free,ul>, -
2E4C  2641                  <<Window size>,b_window,ub>, -
2E4C  2642                  <<Vol. lock ID>,l_vollkid,xl_neq>, -
2E4C  2643                  <<Block. lock ID>,l_blockid,xl_neq>, -
2E4C  2644                  >
2EBC  2645
2EBC  2646 vcb_disk_col_2:
2EBC  2647          column_list -
2EBC  2648                  vcb$, 16, 8, 4, < -
2EBC  2649                  <<Rel. volume>,w_rvn,uw>, -
2EBC  2650                  <<Max. files>,l_maxfiles,ul>, -
2EBC  2651                  <<Rsvd. files>,b_resfiles,ub>, -
2EBC  2652                  <<Cluster size>,w_cluster,uw>, -
2EBC  2653                  <<Def. extend sz.>,w_extend,uw>, -
2EBC  2654                  <<Record size>,w_recordsz,uw>, -
2EBC  2655                  >
2F2C  2656
2F2C  2657 vcb_disk_col_3:
2F2C  2658          column_list -
2F2C  2659                  vcb$, 16, 8, 0, < -
2F2C  2660                  <<AQB address>,l_aqb,xl>, -
2F2C  2661                  <<RVT address>,l_rvt,xl>, -
2F2C  2662                  <<FCB queue>,l_fcbfl,q2>, -
2F2C  2663                  <<Quota FCB>,l_quotafcb,xl_neq>, -
2F2C  2664                  <<Quota cache>,l_quocache,xl_neq>, -
2F2C  2665                  <<Cache blk.>,l_cache,xl_neq>, -
```

```
2F2C  2666                      >
2F9C  2667
2F9C  2668 ;
2F9C  2669 ; PRINT_COLUMNS tables for tape VCB displays
2F9C  2670 ;
2F9C  2671
2F9C  2672 vcb_tape_col_1:
2F9C  2673          column_list -
2F9C  2674                  vcb$, 16, 8, 4, < -
2F9C  2675                  <<Transactions>,w_trans,uw>, -
2F9C  2676                  <<Start record>,l_st_record,ul>, -
2F9C  2677                  <<Tapemark count>,b_tm,ub>, -
2F9C  2678                  >
2FDC  2679
2FDC  2680 vcb_tape_col_2:
2FDC  2681          column_list -
2FDC  2682                  vcb$, 16, 8, 4, < -
2FDC  2683                  <<Rel. volume>,b_cur_rvn,ub>, -
2FDC  2684                  <<Tape vol. list>,l_mvl,xl_neq>, -
2FDC  2685                  >
300C  2686
300C  2687 vcb_tape_col_3:
300C  2688          column_list -
300C  2689                  vcb$, 16, 8, 0, < -
300C  2690                  <<AQB address>,l_aqb,xl>, -
300C  2691                  <<Virt. pg. queue>,l_vpfl,q2>, -
300C  2692                  <<Blocked queue>,l_blockfl,q2>, -
300C  2693                  >
304C  2694
304C  2695 ;
304C  2696 ; PRINT_COLUMNS tables for network VCB displays
304C  2697 ;
304C  2698
304C  2699 vcb_net_col_1:
304C  2700          column_list -
304C  2701                  vcb$, 16, 8, 4, < -
304C  2702                  <<Transactions>,w_trans,uw>, -
304C  2703                  >
306C  2704
306C  2705 vcb_net_col_2:
306C  2706          column_list -
306C  2707                  vcb$, 16, 8, 4, < -
306C  2708                  <<Mount count>,w_mcount,uw>, -
306C  2709                  >
308C  2710
308C  2711 vcb_net_col_3:
308C  2712          column_list -
308C  2713                  vcb$, 16, 8, 0, < -
308C  2714                  <<AQB address>,l_aqb,xl>, -
308C  2715                  >
30AC  2716
30AC  2717 ;
30AC  2718 ; PRINT_COLUMNS tables for journal VCB displays
30AC  2719 ;
30AC  2720
30AC  2721 vcb_jnl_col_1:
30AC  2722          column_list -
```

```
                                                N 15
30AC  2723                      vcb$, 16, 8, 4, < -
30AC  2724                      <<Copies>,w_jnl_cop,uw>, -
30AC  2725                      <<Mask>,l_jnl_mask,xl>, -
30AC  2726                      >
30DC  2727
30DC  2728 vcb_jnl_col_2:
30DC  2729              column_list -
30DC  2730                      vcb$, 16, 8, 4, < -
30DC  2731                      <<Access mode>,b_jnl_mode,xb>, -
30DC  2732                      <<JFT address>,l_jnl_jfta,xl>, -
30DC  2733                      >
310C  2734
310C  2735 vcb_jnl_col_3:
310C  2736              column_list -
310C  2737                      vcb$, 16, 8, 0, < -
310C  2738                      <<AQB address>,l_aqb,xl>, -
310C  2739                      <<JMT address>,l_jnl_jmt,xl>, -
310C  2740                      >
```

```
                        313C  2742          .sbttl  show_cddb, Display Class Driver Data Block (CDDB)
                        313C  2743 ;---
                        313C  2744 ;
                        313C  2745 ;       show_cddb
                        313C  2746 ;
                        313C  2747 ;       Display the Class Driver Data Block (CDDB)
                        313C  2748 ;
                        313C  2749 ; Inputs:
                        313C  2750 ;
                        313C  2751 ;       ap = Address of UCB in local storage
                        313C  2752 ;       r6 = actual address of cddb
                        313C  2753 ;
                        313C  2754 ;---
                        313C  2755
                        313C  2756 show_cddb:
                   083C 313C  2757          .word   ^m<r2,r3,r4,r5,r11>
                        313E  2758
                56   D5 313E  2759          tstl    r6                      ; is there a cddb
                23   13 3140  2760          beql    5$                      ; no, so exit
     52  00000471'EF   9E 3142  2761          movab   cddb, r2                ; store address of local cddb
                        3149  2762          getmem  (r6),(r2),#cddb$c_length        ; read entire cddb
              08 50   E9 315A  2763          blbc    r0,5$                   ; return if not able to read it
                        315D  2764
    0A A2   0164 8F   B1 315D  2765          cmpw    #<dyn$c_cd_cddb@8+dyn$c_classdrv>,cddb$b_type(r2)
                        3163  2766                                          ; check for valid block type
                08   13 3163  2767          beql    10$                     ; exit if not valid type
                        3165  2768 5$:
                        3165  2769          status  success                 ;
                04   316C  2770          ret
                        316D  2771
                        316D  2772 10$:     ensure  20                      ; need 15 lines for this display
                        3185  2773          skip    1                       ; advance 1 line
                56   DD 318E  2774          pushl   r6                      ; pass address of cddb to print routine
     00000575'EF   B5 3190  2775          tstw    flag_2nd_cddb           ; 0 - primary, 1 - secondary
                0F   12 3196  2776          bneq    second                  ; secondary if branch
                        3198  2777          print   1,<!_!_--- Primary Class Driver Data Block (CDDB) !XL --->
                0D   11 31A5  2778          brb     display
                        31A7  2779 second:
                        31A7  2780          print   1,<!_!_--- Secondary Class Driver Data Block (CDDB) !XL --->
                        31B4  2781 display:
                        31B4  2782          skip    1                       ; advance 1 line
             5B   5E   D0 31BD  2783          movl    sp,r11                  ; save pre-allocation stack pointer
                        31C0  2784          alloc   80,r4                   ; 80 byte output buffer
          7E   12 A2   3C 31D2  2785          movzwl  cddb$w_status(r2),-(sp) ; cddb status field
             D74E CF   9F 31D6  2786          pushab  cddb_status             ; bit definition table
00000000'EF   C2   FB 31DA  2787          calls   #2,translate_bits       ; translate bits to names
                54   DD 31E1  2788          pushl   r4                      ; address of output descriptor
          7E   12 A2   3C 31E3  2789          movzwl  cddb$w_status(r2),-(sp) ; pass value of status field to print
                        31E7  2790          print   2,<Status:              !XW !AS>        ; display status
          64   50 8F   9A 31F4  2791          movzbl  #80,(r4)                ;
          7E   28 A2   3C 31F8  2792          movzwl  cddb$w_cntrlflgs(r2),-(sp)  ; cddb controller flags
             D790 CF   9F 31FC  2793          pushab  cddb_flags              ; bit definition table
00000000'EF   02   FB 3200  2794          calls   #2,translate_bits       ; translate bits to names
                54   DD 3207  2795          pushl   r4                      ; address of output descriptor
          7E   28 A2   3C 3209  2796          movzwl  cddb$w_cntrlflgs(r2),-(sp)      ; pass value of status field to prin
                        320D  2797          print   2,<Controller Flags:    !XW !AS>  ; display status
             5E   5B   D0 321A  2798          movl    r11,sp                  ; restore stack pointer
```

```
                321D  2799          skip    1                               ; advance 1 line
                3226  2800          print_columns   -
                3226  2801                  (r2), r6,-
                3226  2802                  cddb_col_1,cddb_col_2,cddb_col_3        ;display!!!!
                3243  2803          status  success
          04    324A  2804          ret
                324B  2805
                324B  2806
```

```
                        .sbttl class driver data block tables & action routines

                ;  The following are all PRINT_COLUMNS action routines for the show_cddb
                ;  block displays.
                ;
                ;    Action Routine Inputs:
                ;
                ;       R2              value from the COLUMN_LIST entry
                ;       R5              size of value section for this item
                ;       R7              address of a descriptor for a scratch string in
                ;                       which the FAO converted value is to be returned
                ;       R11             base address of the local UCB copy
                ;
                ;    Action Routine Outputs:
                ;
                ;       R0              status
                ;                          lbs ==> use this entry
                ;                          lbc ==> skip this entry
                ;       R1 - R5         scratch
                ;                       all other registers must be preserved


                ;
                ; PRINT_COLUMNS tables for CDDB displays
                ;

                cddb_col_1:
                        column_list -
                                cddb$, 16, 8, 4, < -
                                <<Allocation class>,l_allocls,ul>, -
                                <<System ID>,cddb_4bytes,cddb$b_systemid>, -
                                <<>,cddb_2bytes,cddb$b_systemid+4>,-
                                <<Contrl. ID>,cddb_4bytes,cddb$q_cntrlid>, -
                                <<>,cddb_4bytes,cddb$q_cntrlid+4>,-
                                <<Response ID>,l_oldrspid,xl>, -
                                <<MSCP Cmd status>,l_oldcmdsts,xl>,-
                                >

                cddb_col_2:
                        column_list -
                                cddb$, 16, 8, 4, < -
                                <<CDRP Queue>,l_cdrpqfl,q2>, -
                                <<Restart Queue>,l_rstrtqfl,q2>, -
                                <<Restarted CDRP>,rstrt_cdrp,cddb$l_rstrtcdrp>, -
                                <<CDRP retry cnt.>,retry_cnt,cddb$b_retrycnt>, -
                                <<DAP Count>,b_dapcount,ub>, -
                                <<Contr. timeout>,w_cntrltmo,uw>, -
                                <<Reinit Count>,w_rstrtcnt,uw>, -
                                <<Wait UCB Count>,w_wtucbctr,uw>, -
                                >

                cddb_col_3:
                        column_list -
                                cddb$, 16, 8, 0, < -
                                <<DDB address>,l_ddb,xl>, -
                                <<CRB address>,l_crb,xl>, -
```

```
                     335B  2865                    <<CDDB link>,l_cddblink,xl>, -
                     335B  2866                    <<PDT address>,l_pdt,xl>, -
                     335B  2867                    <<Original UCB>,l_origucb,xl>,-
                     335B  2868                    <<UCB chain>,l_ucbchain,xl>, -
                     335B  2869                    >
                     33CB  2870
                     33CB  2871 ;**********
                     33CB  2872 cddb_4bytes:
53   5B   52   C1      33CB  2873            addl3   r2,r11,r3                      ; locate storage of interest
     55   08   C2      33CF  2874            subl    #8,r5                         ; get size of filler field
                     33D2  2875            $fao_s  -
                     33D2  2876                    ctrstr=cddb_fao,-
                     33D2  2877                    outbuf = (r7),-
                     33D2  2878                    outlen = (r7),-
                     33D2  2879                    p1 = r5,-
                     33D2  2880                    p2 = (r3)
                05   33E7  2881            rsb
                     33E8  2882
                     33E8  2883 ;**********
                     33E8  2884 cddb_2bytes:
53   5B   52   C1      33E8  2885            addl3   r2, r11, r3                   ; locate storage of interest
     55   04   C2      33EC  2886            subl    #4, r5                        ; get size of filler field
                     33EF  2887            $fao_s  -
                     33EF  2888                    ctrstr = sb_fao_6bytes, -
                     33EF  2889                    outbuf = (r7), -
                     33EF  2890                    outlen = (r7), -
                     33EF  2891                    p1 = r5, -
                     33EF  2892                    p2 = (r3)
                05   3404  2893            rsb
                     3405  2894
                     3405  2895 ;**********
                     3405  2896 rstrt_cdrp:
0C 12 AB   00   E1   3405  2897            bbc     #cddb$v_snglstrm,cddb$w_status(r11),cddb_act_nop
                     340A  2898                                                  ; cdrp only exists if single stream
     52   5B   C0      340A  2899            addl    r11,r2                        ; locate cell to return
                     340D  2900            do_column_entry xl,jmp              ; display this entry
                     3416  2901
                     3416  2902 cddb_act_nop:
          50   D4      3416  2903            clrl    r0
               05      3418  2904            rsb
                     3419  2905
                     3419  2906 ;**********
                     3419  2907 retry_cnt:
F8 12 AB   00   E1   3419  2908            bbc     #cddb$v_snglstrm,cddb$w_status(r11),cddb_act_nop
                     341E  2909                                                  ; count is valid if single stream
     52   5B   C0      341E  2910            addl    r11,r2                        ; locate cell to return
                     3421  2911            do_column_entry ub,jmp              ; display this entry
                     342A  2912
```

```
342A  2914          .end
```

```
$$$                             = 00000871 R      04
$$.TMP1                         = 00000001
$$.TMP2                         = 000000EF
$$BASE                          = 00000001
$$DISPL                         = 000000A2
$$GENSW                         = 00000001
$$HIGH                          = 000000A1
$$LIMIT                         = 000000A0
$$LOW                           = 00000001
$$MNSW                          = 00000001
$$MXSW                          = 00000001
$$T2                            = 00000005
ACP_STATUS                        00000B98 R      03
ADD_SYMBOL                        ******** X    03
ADP$W_ADPTYPE                   = 0000000E
AQB                               0000041D R      02
AQB$B_ACPTYPE                   = 00000015
AQB$B_CLASS                     = 00000016
AQB$B_MNTCNT                    = 0000000B
AQB$B_STATUS                    = 00000014
AQB$C_LENGTH                    = 0000001C
AQB$K_F11V1                     = 00000001
AQB$K_F11V2                     = 00000002
AQB$K_JNL                       = 00000006
AQB$K_MTA                       = 00000003
AQB$K_NET                       = 00000004
AQB$K_REM                       = 00000005
AQB$K_UNDEFINED                 = 00000000
AQB$L_ACPPID                    = 0000000C
AQB$L_ACPQFL                    = 00000000
AQB$L_LINK                      = 00000010
AQB$V_CREATING                  = 00000003
AQB$V_DEFCLASS                  = 00000001
AQB$V_DEFSYS                    = 00000002
AQB$V_UNIQUE                    = 00000000
AQB_ACPTYPE                       00000BC0 R      03
AQB_CLASS                         000027F7 R      03
AQB_COLUMN_1                      0000274D R      03
AQB_COLUMN_2                      0000276D R      03
AQB_COLUMN_3                      0000279D R      03
AQB_TYPE                          000027CD R      03
ARGS                            = 00000003
AT$_UBA                         = 00000001
BAD_ASCIC                         00000E36 R      04
BIT...                          = 00000003
BUFFER                            ******** X    03
BUS_TYPE                          00000728 R      03
CARD_TYPE                         00000548 R      03
CBL_A_ASCIC                       00000E2C R      04
CBL_B_ASCIC                       00000E2F R      04
CDDB                              00000471 R      02
CDDB$B_DAPCOUNT                 = 00000039
CDDB$B_RETRYCNT                 = 00000038
CDDB$B_SYSTEMID                 = 0000000C
CDDB$B_TYPE                     = 0000000A
CDDB$C_LENGTH                   = 00000070
CDDB$K_LENGTH                   = 00000070
CDDB$L_ALLOCLS                  = 00000050
CDDB$L_CDDBLINK                 = 00000058
CDDB$L_CDRPQFL                  = 00000000
CDDB$L_CRB                      = 00000018
CDDB$L_DDB                      = 0000001C
CDDB$L_OLDCMDST                 = 00000030
CDDB$L_OLDRSPID                 = 0000002C
CDDB$L_ORIGUCB                  = 0000004C
CDDB$L_PDT                      = 00000014
CDDB$L_RSTRTCDRP                = 00000034
CDDB$L_RSTRTQFL                 = 0000003C
CDDB$L_UCBCHAIN                 = 00000048
CDDB$Q_CNTRLID                  = 00000020
CDDB$V_2PBSY                    = 0000000B
CDDB$V_ALCLS_SET                = 00000006
CDDB$V_DAPBSY                   = 0000000A
CDDB$V_IMPEND                   = 00000001
CDDB$V_INITING                  = 00000002
CDDB$V_NOCONN                   = 00000007
CDDB$V_POLLING                  = 00000005
CDDB$V_QUORLOST                 = 00000009
CDDB$V_RECONNECT                = 00000003
CDDB$V_RESYNCH                  = 00000004
CDDB$V_RSTRTWAIT                = 00000008
CDDB$V_SNGLSTRM                 = 00000000
CDDB$W_CNTRLFLGS                = 00000028
CDDB$W_CNTRLTMO                 = 0000002A
CDDB$W_RSTRTCNT                 = 0000003A
CDDB$W_STATUS                   = 00000012
CDDB$W_WTUCBCTR                 = 0000005E
CDDB_2BYTES                       000033E8 R      03
CDDB_2P                           000004E1 R      02
CDDB_4BYTES                       000033CB R      03
CDDB_ACT_NOP                      00003416 R      03
CDDB_COL_1                        0000324B R      03
CDDB_COL_2                        000032CB R      03
CDDB_COL_3                        0000335B R      03
CDDB_FAO                          00000E12 R      04
CDDB_FLAGS                        00000990 R      03
CDDB_STATUS                       00000928 R      03
CDRP                              00000289 R      02
CDRP$B_EFN                      = FFFFFFC2
CDRP$B_RMOD                     = FFFFFFAB
CDRP$C_CD_LEN                   = 00000048
CDRP$L_AST                      = FFFFFFB0
CDRP$L_DUTUFLAGS                = 00000040
CDRP$L_FQFL                     = 00000000
CDRP$L_IOQFL                    = FFFFFFA0
CDRP$L_IOSB                     = FFFFFFC4
CDRP$L_PID                      = FFFFFFAC
CDRP$L_UCB                      = FFFFFFBC
CDRP$L_WIND                     = FFFFFFB8
CDRP$V_CAND                     = 00000000
CDRP$V_CANIO                    = 00000001
CDRP$V_ERLIP                    = 00000002
CDRP$V_HIRT                     = 00000004
CDRP$V_IVCMD                    = 00000008
```

```
CDRP$V_PERM             =  00000003
CDRP$W_CHAN             =  FFFFFFC8
CDRP$W_FUNC             =  FFFFFFC0
CDRP$W_STS              =  FFFFFFCA
CDRP_DOTUFLAGS             000009D8 R      03
CDRP_HEADING               000029B9 R      03
CDRP_LENGTH             =  000000A8
CMND_BUFFER                ********   X    03
CMND_DESCR                 ********   X    03
COLM$K_FAO_AC           =  00000000
COLM$K_FAO_AS           =  00000001
COLM$K_FAO_OW           =  00000007
COLM$K_FAO_Q2           =  00000011
COLM$K_FAO_UB           =  00000005
COLM$K_FAO_UL           =  0000000F
COLM$K_FAO_UW           =  0000000A
COLM$K_FAO_XB           =  00000003
COLM$K_FAO_XL           =  0000000D
COLM$K_FAO_XL_NEQ       =  0000008D
COLM$K_FAO_XW           =  00000008
COLM$K_LENGTH           =  00000010
CRB$K_LENGTH            =  00000048
CRB$L_AUXSTRUC          =  00000010
CRB$L_DUETIME           =  00000018
CRB$L_INTD              =  00000024
CRB$L_LINK              =  00000020
CRB$L_TIMELINK          =  00000014
CRB$L_TOUTROUT          =  0000001C
CRB$L_WQFL              =  00000000
CRB$W_REFC              =  0000000C
CRB_COLUMN_1               000013F0 R      03
CRB_COLUMN_2               00001420 R      03
CRB_COLUMN_3               00001450 R      03
CRB_DEVCLASS               00000578 R      02
CRB_TIMEOUT                00001480 R      03
CROSSED_ASCIC              00000E3A R      04
DC$_BUS                 =  00000080
DC$_CARD                =  00000041
DC$_DISK                =  00000001
DC$_JOURNAL             =  000000A1
DC$_LP                  =  00000043
DC$_MAILBOX             =  000000A0
DC$_MISC                =  000000C8
DC$_REALTIME            =  00000060
DC$_SCOM                =  00000020
DC$_TAPE                =  00000002
DC$_TERM                =  00000042
DC$_WORKSTATION         =  00000046
DDB                        00000071 R      02
DDB$B_ACPCLASS          =  00000013
DDB$B_TYPE              =  0000000A
DDB$C_LENGTH            =  00000044
DDB$K_CART              =  00000002
DDB$K_LENGTH            =  00000044
DDB$K_PACK              =  00000001
DDB$K_SLOW              =  00000003
DDB$K_TAPE              =  00000004
DDB$L_ACPD              =  00000010
DDB$L_ALLOCLS           =  0000003C
DDB$L_CONLINK           =  00000038
DDB$L_DDT               =  0000000C
DDB$L_DP_UCB            =  00000040
DDB$L_LINK              =  00000000
DDB$L_SB                =  00000034
DDB$L_UCB               =  00000004
DDB$T_DRVNAME           =  00000024
DDB$T_NAME              =  00000014
DDB_2P                     000000B5 R      02
DDB_ACPCLASS               00000090 R      03
DDB_ACPCLS                 000013C4 R      03
DDB_ACPD                   0000139E R      03
DDB_COLUMN_1               000012DE R      03
DDB_COLUMN_2               0000131E R      03
DDB_COLUMN_3               0000135E R      03
DDB_NO_ACP                 000013C1 R      03
DDT$K_LENGTH            =  00000038
DDT$L_ALTSTART          =  0000001C
DDT$L_CANCEL            =  0000000C
DDT$L_CLONEDUCB         =  00000024
DDT$L_FDT               =  00000008
DDT$L_MNTVER            =  00000020
DDT$L_REGDUMP           =  00000010
DDT$L_START             =  00000000
DDT$L_UNITINIT          =  00000018
DDT$L_UNSOLINT          =  00000004
DDT$W_DIAGBUF           =  00000014
DDT$W_ERRORBUF          =  00000016
DDT$W_FDTSIZE           =  00000028
DDT_ADDRESS                000017B3 R      03
DDT_COLUMN_1               000016C3 R      03
DDT_COLUMN_2               00001713 R      03
DDT_COLUMN_3               00001763 R      03
DDT_RETURN                 00000B79 R      04
DEFINE_UCB_SYMBOLS         00001E6A R      03
DEV$M_2P                =  00000010
DEV$M_DMT               =  00200000
DEV$M_MBX               =  00100000
DEV$M_MNT               =  00080000
DEV$M_NET               =  00002000
DEV$V_2P                =  00000004
DEV$V_ALL               =  00000017
DEV$V_AVL               =  00000012
DEV$V_CCL               =  00000001
DEV$V_CDP               =  00000003
DEV$V_CLU               =  00000000
DEV$V_DET               =  00000001
DEV$V_DIR               =  00000003
DEV$V_DMT               =  00000015
DEV$V_DUA               =  0000000F
DEV$V_ELG               =  00000016
DEV$V_FOD               =  0000000E
DEV$V_FOR               =  00000018
DEV$V_GEN               =  00000011
DEV$V_IDV               =  0000001A
```

```
DEV$V_MBX                       =  00000014
DEV$V_MNT                       =  00000013
DEV$V_MSCP                      =  00000005
DEV$V_NET                       =  0000000D
DEV$V_NNM                       =  00000009
DEV$V_ODV                       =  0000001B
DEV$V_OPR                       =  00000007
DEV$V_RCK                       =  0000001E
DEV$V_RCT                       =  00000008
DEV$V_REC                       =  00000000
DEV$V_RED                       =  00000008
DEV$V_RND                       =  0000001C
DEV$V_RTM                       =  0000001D
DEV$V_RTT                       =  00000002
DEV$V_SDI                       =  00000004
DEV$V_SHR                       =  00000010
DEV$V_SPL                       =  00000006
DEV$V_SQD                       =  00000005
DEV$V_SRV                       =  00000007
DEV$V_SSM                       =  00000006
DEV$V_SWL                       =  00000019
DEV$V_TRM                       =  00000002
DEV$V_WCK                       =  0000001F
DEVICE_CHAR                       00000160  R      03
DEVICE_CHAR_2                     00000248  R      03
DEVICE_CLASS                      000002A0  R      03
DISK_TYPE                         00000308  R      03
DISPLAY                           000031B4  R      03
DISPLAY_DDT                       00001275  R      03
DISPLAY_DEVBYADDR                 00000C00  RG     03
DISPLAY_DEVICE                    00000CE2  RG     03
DO_UCB_COLUMNS                    00001F25  R      03
DPT                               00000439  R      02
DPT$C_LENGTH                    =  00000038
DPT$L_FLINK                     =  00000000
DPT$T_NAME                      =  00000020
DPT$W_SIZE                      =  00000008
DT$_ATJNL                       =  00000003
DT$_ATJNL                       =  00000004
DT$_BIJNL                       =  00000002
DT$_CI                          =  0000000C
DT$_CI750                       =  00000002
DT$_CI780                       =  00000001
DT$_CLJNL                       =  00000005
DT$_CR11                        =  00000001
DT$_CRX50                       =  00000021
DT$_DELUA                       =  00000019
DT$_DEQNA                       =  00000016
DT$_DEUNA                       =  0000000E
DT$_DHU                         =  00000047
DT$_DHV                         =  00000046
DT$_DMC11                       =  00000001
DT$_DMF32                       =  0000000A
DT$_DMP11                       =  00000009
DT$_DMR11                       =  00000002
DT$_DMV11                       =  00000017
DT$_DMZ32                       =  00000045
DT$_DN11                        =  00000001
DT$_DR11C                       =  00000007
DT$_DR11W                       =  00000004
DT$_DR750                       =  00000003
DT$_DR780                       =  00000002
DT$_DZ11                        =  00000042
DT$_DZ32                        =  00000043
DT$_DZ730                       =  00000044
DT$_FT1                         =  00000010
DT$_FT2                         =  00000011
DT$_FT3                         =  00000012
DT$_FT4                         =  00000013
DT$_FT5                         =  00000014
DT$_FT6                         =  00000015
DT$_FT7                         =  00000016
DT$_FT8                         =  00000017
DT$_IX_IEX11                    =  0000000A
DT$_LAT1                        =  00000002
DT$_LA12                        =  00000024
DT$_LA120                       =  00000021
DT$_LA180                       =  00000003
DT$_LA24                        =  00000025
DT$_LA34                        =  00000022
DT$_LA36                        =  00000020
DT$_LA38                        =  00000023
DT$_LAX                         =  00000020
DT$_LESI                        =  00000005
DT$_LP11                        =  00000001
DT$_LPA11                       =  00000001
DT$_LQP02                       =  00000026
DT$_MBX                         =  00000001
DT$_ML11                        =  00000011
DT$_MX_MUX200                   =  00000008
DT$_NI                          =  0000000D
DT$_NULL                        =  00000003
DT$_NV_X29                      =  00000006
DT$_NW_X25                      =  00000005
DT$_PCL11R                      =  00000005
DT$_PCL11T                      =  00000006
DT$_RA60                        =  00000016
DT$_RA80                        =  00000014
DT$_RA81                        =  00000015
DT$_RA82                        =  0000001E
DT$_RB02                        =  00000012
DT$_RB80                        =  00000013
DT$_RC25                        =  00000017
DT$_RC26                        =  0000001F
DT$_RCF25                       =  00000018
DT$_RCF26                       =  00000020
DT$_RD26                        =  0000001D
DT$_RD51                        =  00000019
DT$_RD52                        =  0000001B
DT$_RD53                        =  0000001C
DT$_RDRX                        =  00000007
DT$_RK06                        =  00000001
DT$_RK07                        =  00000002
DT$_RL01                        =  00000009
```

```
DT$_RL02                =  0000000A
DT$_RM03                =  00000006
DT$_RM05                =  0000000F
DT$_RM80                =  0000000D
DT$_RP04                =  00000003
DT$_RP05                =  00000004
DT$_RP06                =  00000005
DT$_RP07                =  00000007
DT$_RP07HT              =  00000008
DT$_RUJNL               =  00000001
DT$_RX01                =  00000010
DT$_RX02                =  0000000B
DT$_RX04                =  0000000C
DT$_RX50                =  0000001A
DT$_RZ01                =  00000017
DT$_RZF01               =  00000018
DT$_SB_ISB11            =  00000007
DT$_SHRMBX              =  00000002
DT$_TA78                =  00000006
DT$_TA81                =  00000009
DT$_TE16                =  00000001
DT$_TEK401X             =  0000000A
DT$_TK50                =  0000000A
DT$_TQ_BTS              =  00000004
DT$_TST1                =  00000004
DT$_TTYUNKN             =  00000000
DT$_TU45                =  00000002
DT$_TU58                =  0000000E
DT$_TU77                =  00000003
DT$_TU78                =  00000005
DT$_TU80                =  00000007
DT$_TU81                =  00000008
DT$_TU81P               =  00000006
DT$_UDA50               =  00000003
DT$_UDA50A              =  00000004
DT$_UK_KTC32            =  00000015
DT$_UQPORT              =  00000003
DT$_VK100               =  00000002
DT$_VS100               =  00000001
DT$_VS125               =  00000002
DT$_VS300               =  00000003
DT$_VT05                =  00000001
DT$_VT100               =  00000060
DT$_VT101               =  00000061
DT$_VT102               =  00000062
DT$_VT105               =  00000063
DT$_VT125               =  00000064
DT$_VT131               =  00000065
DT$_VT132               =  00000066
DT$_VT173               =  00000003
DT$_VT52                =  00000040
DT$_VT55                =  00000041
DT$_VT5X                =  00000040
DT$_XI_DR11C            =  0000000D
DT$_XJ_2780             =  00000004
DT$_XK_3271             =  00000003
DT$_XP_PCL11B           =  00000009
DT$_XV_3271             =  0000000B
DT$_YN_X25              =  0000000F
DT$_YO_X25              =  00000010
DT$_YP_ADCCP            =  00000011
DT$_YQ_3271             =  00000012
DT$_YR_DDCMP            =  00000013
DT$_YS_SDLC             =  00000014
DYN$C_CD_CDDB           =  00000001
DYN$C_CLASSDRV          =  00000064
DYN$C_DDB               =  00000006
DYN$C_SCS               =  00000060
DYN$C_SCS_PDT           =  00000005
DYN$C_SCS_SB            =  00000007
DYN$C_UCB               =  00000010
DYN$C_VCB               =  00000011
END_PB                     00001913 R      03
FAB$L_STV                  ******** X    03
FIND_DPT                   00000EB7 R      03
FLAG_2ND_CDDB              00000575 R      02
FLAG_M_ALT_PATH         =  00000002
FLAG_M_FND_UNIT         =  00000004
FLAG_M_ONE_UNIT         =  00000001
FLAG_V_ALT_PATH         =  00000001
FLAG_V_FND_UNIT         =  00000002
FLAG_V_ONE_UNIT         =  00000000
FOUND_DPT                  000008D2 R      04
GETMEM                     ******** X    03
GET_DDB                    00000F05 R      03
GET_UCB                    00001F89 R      03
IDB$B_VECTOR            =  0000000B
IDB$K_LENGTH            =  00000038
IDB$L_ADP               =  00000014
IDB$L_CSR               =  00000000
IDB$L_OWNER             =  00000004
IDB$W_UNITS             =  0000000C
IDB_COLUMN_1               0000162A R      03
IDB_COLUMN_2               0000165A R      03
IDB_COLUMN_3               0000168A R      03
IDB_VECTOR                 000016AA R      03
IO$S_FCODE              =  00000006
IO$V_FCODE              =  00000000
IO$_ACCESS              =  00000032
IO$_ACPCONTROL          =  00000038
IO$_AVAILABLE           =  00000011
IO$_CREATE              =  00000033
IO$_DEACCESS            =  00000034
IO$_DELETE              =  00000035
IO$_DSE                 =  00000015
IO$_ERASETAPE           =  00000006
IO$_MODIFY              =  00000036
IO$_NOP                 =  00000000
IO$_PACKACK             =  00000008
IO$_READLBLK            =  00000021
IO$_READPBLK            =  0000000C
IO$_READVBLK            =  00000031
IO$_RECAL               =  00000003
IO$_REWIND              =  00000024
```

```
IO$_REWINDOFF                   =  00000022
IO$_SEEK                        =  00000002
IO$_SENSECHAR                   =  0000001B
IO$_SENSEMODE                   =  00000027
IO$_SETCHAR                     =  0000001A
IO$_SETMODE                     =  00000023
IO$_SKIPFILE                    =  00000025
IO$_SKIPRECORD                  =  00000026
IO$_SPACERECORD                 =  00000009
IO$_UNLOAD                      =  00000001
IO$_WRITECHECK                  =  0000000A
IO$_WRITELBLK                   =  00000020
IO$_WRITEMARK                   =  0000001C
IO$_WRITEOF                     =  00000028
IO$_WRITEPBLK                   =  0000000B
IO$_WRITEVBLK                   =  00000030
IO$_WRTTMKR                     =  0000001D
IOC$GL_DPTLIST                    ********    X  03
IOC$RETURN                        ********    X  03
IO_FUNCTION                       00000A88 R     03
IRP                               000001C5 R     02
MSCP$V_CF_576                   =  00000000
MSCP$V_CF_ATTN                  =  00000007
MSCP$V_CF_MISC                  =  00000006
MSCP$V_CF_MLTHS                 =  00000002
MSCP$V_CF_OTHER                 =  00000005
MSCP$V_CF_REPLC                 =  0000000F
MSCP$V_CF_SHADW                 =  00000001
MSCP$V_CF_THIS                  =  00000004
MSG$_SUCCESS                      ********    X  03
NEW_PAGE                          ********    X  03
NODNAM_2P                         00000060 R     02
NO_DPT                            00000915 R     04
NULL_ASCIC                        00000E21 R     04
OK_ASCIC                          00000E33 R     04
ONE_PATH                          00001065 R     04
ORB$L_OWNER                     =  00000000
ORB$W_UICGROUP                  =  00000002
ORB$W_UICMEMBER                 =  00000000
ORB_OWNER                         00002355 R     03
OUTPUT                            ********    X  03
PAGE_SIZE                         ********    X  03
```

```
PB$V_MAINT                      = 00000000
PB$V_PORT_TYP                   = 00000000
PB$V_STATE                      = 00000001
PB$V_TIM                        = 00000000
PB$W_RETRY                      = 00000022
PB$W_STATE                      = 00000012
PB$W_STS                        = 00000044
PB_CABLES                         00001BD6 R      03
PB_COLUMN_1                       00001A38 R      03
PB_COLUMN_2                       00001AB8 R      03
PB_DUALPATH                       00001BBD R      03
PB_LCLSTATE                       00001C51 R      03
PB_LOOP                           00001854 R      03
PB_RMTSTATE                       00001B38 R      03
PB_RPORT_TYP                      00001B8A R      03
PB_RPORT_TYPE                     00000058 R      03
PB_RSTATE                         00000038 R      03
PB_STATE                          00000010 R      03
PB_STATUS                         00000000 R      03
PCB$T_LNAME                     = 00000070
PDVNM_B_NODESZ                    00000022
PDVNM_K_LENGTH                  = 00000024
PDVNM_T_DDC                       00000010
PDVNM_T_NODE                      00000000
PDVNM_W_UNIT                      00000020
PRINT                             ********    X   03
PRINT_CDRP                        00002823 R      03
PRINT_COLUMNS                     ********    X   03
PRINT_COLUMN_VALUE                ********    X   03
PRINT_IRP                         000029D0 R      03
PROCESS_2P_DDB                    00001D7C R      03
QUEUE_NOTEMPTY                    00000577 R      02
QUEUE_TITLE                       00002962 R      03
RAB$L_RBF                         ********    X   03
RAB$W_RSZ                         ********    X   03
REALTIME_TYPE                     000006D0 R      03
REQUEST_STATUS                    00000A10 R      03
RETRY_CNT                         00003419 R      03
RSTRT_CDRP                        00003405 R      03
SB                                00000000 R      02
SB$B_ENBMSK                     = 0000005A
SB$B_HWVERS                     = 00000038
SB$B_SYSTEMID                   = 00000018
SB$B_TYPE                       = 0000000A
SB$C_LENGTH                     = 00000060
SB$K_LENGTH                     = 00000060
SB$L_DDB                        = 00000054
SB$L_FLINK                      = 00000000
SB$L_PBFL                       = 0000000C
SB$Q_SWINCARN                   = 0000002C
SB$Q_SWINCARN2                  = 00000030
SB$S_NODENAME                   = 00000010
SB$T_HWTYPE                     = 00000034
SB$T_NODENAME                   = 00000044
SB$T_SWTYPE                     = 00000024
SB$T_SWVERS                     = 00000028
SB$W_MAXDG                      = 00000020
SB$W_MAXMSG                     = 00000022
SB$W_TIMEOUT                    = 00000058
SB_6BYTES                         000019F4 R      03
SB_COLUMN_1                       00001914 R      03
SB_COLUMN_2                       00001984 R      03
SB_FAO_6BYTES                     00000DEE R      04
SB_FAO_ASCIC                      00000E00 R      04
SB_LWCHAR                         00001A14 R      03
SCH$GL_PCBVEC                     ********    X   03
SCOM_TYPE                         00000480 R      03
SCS$GA_LOCALSB                    ********    X   03
SCS$GQ_CONFIG                     ********    X   03
SECOND                            000031A7 R      03
SETUP_PRIMARY                     00001DC4 R      03
SET_HEADING                       ********    X   03
SHOW_ACPQ                         000025B3 R      03
SHOW_CDDB                         0000313C R      03
SHOW_CONTROLLER                   00000FE1 R      03
SHOW_DDBS                         00000DFD R      03
SHOW_IOQ                          000024C9 R      03
SHOW_SYSTEM_BLOCK                 000017D8 RG     03
SHOW_UCB                          00001C7B R      03
SHOW_VCB                          00002AB1 R      03
SIZ...                          = 00000001
SKIP_LINES                        ********    X   03
SKIP_SB                           0000103A R      03
SKIP_SECOND_CRB                   000011E8 R      03
SS$_NOSUCHDEV                     ********    X   03
SYS$FAO                           ********    X   03
SYS$FAOL                          ********   GX   03
SYG$PUT                           ********   GX   03
TAPE_TYPE                         00000428 R      03
TERM_TYPE                         00000558 R      03
THIS_PRIMARY                      0000109B R      04
THIS_SECONDARY                    000010DD R      04
TPA$C_NUMBER                    = 0000001C
TPA$L_TOKENCNT                  = 00000010
TRANSLATE_ADDRESS                 ********    X   03
TRANSLATE_BITS                    ********    X   03
TRYMEM                            ********    X   03
UCB                               000000F9 R      02
UCB$B_DEVCLASS                  = 00000040
UCB$B_DEVTYPE                   = 00000041
UCB$B_DIPL                      = 0000005E
UCB$B_ERTCNT                    = 00000080
UCB$B_ERTMAX                    = 00000081
UCB$B_FIPL                      = 0000000B
UCB$B_ONLCNT                    = 000000AE
UCB$B_TYPE                      = 0000000A
UCB$K_LCL_DISK_LENGTH           = 000000CC
UCB$L_2P_CDDB                   = 000000C0
UCB$L_AMB                       = 00000060
UCB$L_CDDB                      = 000000BC
UCB$L_CPID                      = 00000020
UCB$L_CRB                       = 00000024
UCB$L_DDB                       = 00000028
UCB$L_DDT                       = 00000088
```

```
UCB$L_DEVCHAR           =  00000038
UCB$L_DEVCHAR2          =  0000003C
UCB$L_DEVDEPEND         =  00000044
UCB$L_DEVDEPND2         =  00000048
UCB$L_DP_ALTUCB         =  000000A8
UCB$L_DP_DDB            =  000000A0
UCB$L_DP_LINK           =  000000A4
UCB$L_DUETIM            =  0000006C
UCB$L_FPC               =  0000000C
UCB$L_FR3               =  00000010
UCB$L_FR4               =  00000014
UCB$L_IOQFL             =  0000004C
UCB$L_IRP               =  00000058
UCB$L_JNL_MCSID         =  00000084
UCB$L_LINK              =  00000030
UCB$L_LOCKID            =  00000020
UCB$L_LOGADR            =  00000074
UCB$L_OPCNT             =  00000070
UCB$L_ORB               =  0000001C
UCB$L_PDT               =  00000084
UCB$L_PID               =  0000002C
UCB$L_STS               =  00000064
UCB$L_SVAPTE            =  00000078
UCB$L_SVPN              =  00000074
UCB$L_TL_PHYUCB         =  000000A0
UCB$L_VCB               =  00000034
UCB$V_BSY               =  00000008
UCB$V_CANCEL            =  00000003
UCB$V_DEADMO            =  0000000A
UCB$V_DELETEUCB         =  00000010
UCB$V_ERLOGIP           =  00000002
UCB$V_INT               =  00000001
UCB$V_INTTYPE           =  00000007
UCB$V_LCL_VALID         =  00000011
UCB$V_MNTVERIP          =  0000000E
UCB$V_MNTVERPND         =  00000013
UCB$V_MOUNTING          =  00000009
UCB$V_ONLINE            =  00000004
UCB$V_POWER             =  00000005
UCB$V_SUPMVMSG          =  00000012
UCB$V_TEMPLATE          =  0000000D
UCB$V_TIM               =  00000000
UCB$V_TIMOUT            =  00000006
UCB$V_UNLOAD            =  0000000C
UCB$V_VALID             =  0000000B
UCB$V_WRONGVOL          =  0000000F
UCB$W_BCNT              =  0000007E
UCB$W_BOFF              =  0000007C
UCB$W_DEVBUFSIZ         =  00000042
UCB$W_DEVSTS            =  00000068
UCB$W_ERRCNT            =  00000082
UCB$W_REFC              =  0000005C
UCB$W_RWAITCNT          =  00000056
UCB$W_SIZE              =  00000008
UCB$W_STS               =  00000064
UCB$W_UNIT              =  00000054
UCB_2PCDDB              =  000023E2 R     03
UCB_2PDDB               =  000024A4 R     03
UCB_ACT_NOP             =  000022D5 R     03
UCB_ACT_NOP_A           =  000023BD R     03
UCB_ACT_NOP_B           =  000024C6 R     03
UCB_ACT_UB              =  0000227A R     03
UCB_ACT_XL              =  00002330 R     03
UCB_ACT_XL_NEQ          =  000022B1 R     03
UCB_ACT_XW              =  00002467 R     03
UCB_ALLOCLASS           =  00002271 R     03
UCB_ALTUCB              =  00002283 R     03
UCB_BSY                 =  000022A9 R     03
UCB_CDDB                =  000023C0 R     03
UCB_CLSTYP              =  000022BB R     03
UCB_COLUMN_1            =  00001FC1 R     03
UCB_COLUMN_2            =  00002071 R     03
UCB_COLUMN_3            =  00002151 R     03
UCB_CPID                =  000022C5 R     03
UCB_DDB                 =  0000057C R     02
UCB_DUETIM              =  000022D8 R     03
UCB_IPLS                =  000022E3 R     03
UCB_LNM                 =  00002301 R     03
UCB_LOCKID              =  00002327 R     03
UCB_MCSID               =  00002339 R     03
UCB_ONLCNT              =  00002347 R     03
UCB_PDT                 =  00002388 R     03
UCB_RETRY               =  00002408 R     03
UCB_RETRY_FAO           =  0000118C R     04
UCB_RET_2XBYTES         =  000022EB R     03
UCB_RWAITCNT            =  00002449 R     03
UCB_SIZE                =  000000CC
UCB_SVPN                =  00002470 R     03
UCB_TEST_RETRY_FAO      =  0000119C R     04
UCB_TWO_BYTES           =  0000117B R     04
UCB_UIC_CSTR1           =  00001168 R     04
UCB_VCB                 =  0000247E R     03
UNIT_STATUS             =  000000B8 R     03
UNKNOWN                 =  00001160 R     04
VCB                     =  00000331 R     02
VCB$B_CUR_RVN           =  0000002F
VCB$B_JNL_MODE          =  00000044
VCB$B_RESFILES          =  0000004F
VCB$B_STATUS            =  0000000B
VCB$B_STATUS2           =  00000053
VCB$B_TM                =  0000002E
VCB$B_TYPE              =  0000000A
VCB$B_WINDOW            =  00000048
VCB$C_LENGTH            =  000000EC
VCB$L_AQB               =  00000010
VCB$L_BLOCKFL           =  00000000
VCB$L_BLOCKID           =  0000008C
VCB$L_CACHE             =  00000058
VCB$L_FCBFL             =  00000000
VCB$L_FREE              =  00000040
VCB$L_JNL_CHAR          =  00000024
VCB$L_JNL_JFTA          =  00000028
VCB$L_JNL_JMT           =  00000034
VCB$L_JNL_MASK          =  00000048
```

```
VCB$L_MAXFILES          = 00000044
VCB$L_MVL               = 00000034
VCB$L_QUOCACHE          = 0000005C
VCB$L_QUOTAFCB          = 00000054
VCB$L_RVT               = 00000020
VCB$L_ST_RECORD         = 00000030
VCB$L_VOLLKID           = 0000007C
VCB$L_VPFL              = 0000003C
VCB$T_VOLCKNAM          = 00000080
VCB$T_VOLNAME           = 00000014
VCB$V_BLANK             = 0000000A
VCB$V_CANCELIO          = 00000005
VCB$V_EBCDIC            = 00000005
VCB$V_ENUSEREOT         = 00000009
VCB$V_ERASE             = 00000003
VCB$V_EXTFID            = 00000005
VCB$V_GROUP             = 00000006
VCB$V_HOMBLKBAD         = 00000002
VCB$V_IDXHDRBAD         = 00000003
VCB$V_INIT              = 0000000B
VCB$V_INTCHG            = 00000004
VCB$V_JNL_DISK          = 00000000
VCB$V_JNL_TAPE          = 00000001
VCB$V_JNL_TMPFI         = 00000002
VCB$V_LOGICEOVS         = 00000001
VCB$V_MOUNTVER          = 00000002
VCB$V_MUSTCLOSE         = 00000006
VCB$V_NOALLOC           = 00000004
VCB$V_NOAUTO            = 0000000C
VCB$V_NOCACHE           = 00000001
VCB$V_NOHIGHWATER       = 00000004
VCB$V_NOVOL2            = 00000006
VCB$V_NOWRITE           = 00000007
VCB$V_OVRACC            = 00000001
VCB$V_OVREXP            = 00000000
VCB$V_OVRLBL            = 00000002
VCB$V_OVRSETID          = 00000003
VCB$V_OVRVOLO           = 0000000D
VCB$V_PARTFILE          = 00000000
VCB$V_STARFILE          = 00000008
VCB$V_SYSTEM            = 00000007
VCB$V_WAIMOUVOL         = 00000002
VCB$V_WAIREWIND         = 00000003
VCB$V_WAIUSRLBL         = 00000004
VCB$V_WRITETHRU         = 00000000
VCB$V_WRITE_IF          = 00000000
VCB$V_WRITE_SM          = 00000001
VCB$W_CLUSTER           = 0000003C
VCB$W_EXTEND            = 0000003E
VCB$W_JNL_COP           = 00000045
VCB$W_MCOUNT            = 0000004C
VCB$W_MODE              = 0000002C
VCB$W_RECORDSZ          = 00000050
VCB$W_RVN               = 0000000E
VCB$W_TRANS             = 0000000C
VCB_DISK                  00002C8B R      03
VCB_DISK_COL_1            00002E4C R      03
VCB_DISK_COL_2            00002EBC R      03
VCB_DISK_COL_3            00002F2C R      03
VCB_DISK_STATUS           000007D8 R      03
VCB_DISK_STATUS2          00000820 R      03
VCB_FOREIGN               00002D1D R      03
VCB_JNL_COL_1             000030AC R      03
VCB_JNL_COL_2             000030DC R      03
VCB_JNL_COL_3             0000310C R      03
VCB_JOURNAL               00002DE9 R      03
VCB_JOURNAL_CHAR          00000908 R      03
VCB_NET                   00002DC8 R      03
VCB_NET_COL_1             0000304C R      03
VCB_NET_COL_2             0000306C R      03
VCB_NET_COL_3             0000308C R      03
VCB_SHOW_ACPQ             00002E46 R      03
VCB_TAPE                  00002D1D R      03
VCB_TAPE_COL_1            00002F9C R      03
VCB_TAPE_COL_2            00002FDC R      03
VCB_TAPE_COL_3            0000300C R      03
VCB_TAPE_MODE             00000898 R      03
VCB_TAPE_STATUS           00000850 R      03
VEC$B_DATAPATH          = 00000013
VEC$B_NUMREG            = 00000012
VEC$L_ADP               = 00000014
VEC$L_IDB               = 00000008
VEC$L_INITIAL           = 0000000C
VEC$L_INTSER            = 00000004
VEC$L_START             = 0000001C
VEC$L_UNITDISC          = 00000020
VEC$L_UNITINIT          = 00000018
VEC$Q_DISPATCH          = 00000000
VEC$S_DATAPATH          = 00000005
VEC$S_MAPREG            = 0000000F
VEC$V_DATAPATH          = 00000000
VEC$V_LWAE              = 00000005
VEC$V_MAPLOCK           = 0000000F
VEC$V_MAPREG            = 00000000
VEC$V_PATHLOCK          = 00000007
VEC$W_MAPREG            = 00000010
VEC_COLUMN_1              0000149E R      03
VEC_COLUMN_2              000014DE R      03
VEC_COLUMN_3              0000151E R      03
VEC_DATAPATH              0000155E R      03
VEC_FAO_DATAPATH          00000B47 R      04
VEC_FAO_MAPREG            00000B58 R      04
VEC_LOCKED                00000B71 R      04
VEC_LWAE                  00000B6B R      04
VEC_MAPREG                000015DC R      03
VEC_TEST_UBA              000015BE R      03
VIRTUAL_TERMINAL          0000111F R      04
WORKSTATION_TYPE          000006B0 R      03
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000024 (36.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
SDADATA	00000580 (1408.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC BYTE
DEVICE	0000342A (13354.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC LONG
LITERALS	00001B70 (7024.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.04	00:00:00.99
Command processing	108	00:00:00.41	00:00:03.28
Pass 1	1290	00:00:43.46	00:02:42.73
Symbol table sort	0	00:00:03.73	00:00:14.02
Pass 2	919	00:00:10.33	00:00:34.74
Symbol table output	1	00:00:00.44	00:00:01.57
Psect synopsis output	0	00:00:00.02	00:00:00.01
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	2349	00:00:58.45	00:03:37.38

The working set limit was 3000 pages.
381141 bytes (745 pages) of virtual memory were used to buffer the intermediate code.
There were 190 pages of symbol table space allocated to hold 3185 non-local and 393 local symbols.
2914 source lines were read in Pass 1, producing 108 object records in Pass 2.
69 pages of virtual memory were used to define 64 macros.

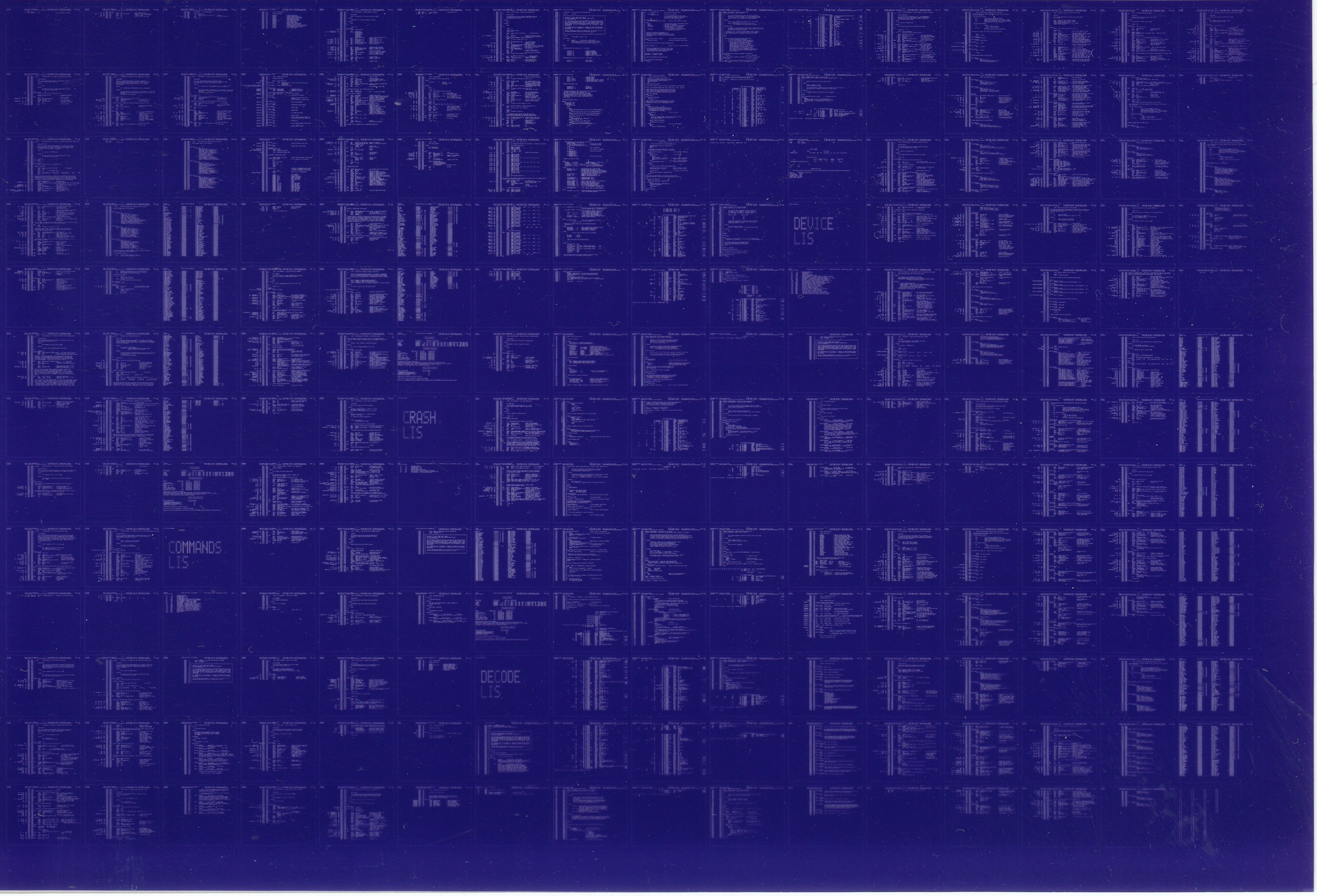
Macro library statistics

Macro library name	Macros defined
_255DUA28:[SDA.OBJ]SDALIB.MLB;1	20
_255DUA28:[SYS.OBJ]LIB.MLB;1	21
_255DUA28:[SYSLIB]STARLET.MLB;2	14
TOTALS (all libraries)	55

3409 GETS were required to define 55 macros.

There were no errors, warnings or information messages.

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



HANDLER
LIS

MAPPING
LIS

DUMP
LIS

MAIN
LIS

EXAMPSL
LIS

MMG
LIS

INDEX
LIS

LOCK
LIS

PARSE
LIS