K 6

```
EEEEEEEEEEEEEEE  RRRRRRRRRRRR     FFFFFFFFFFFFFFF
EEEEEEEEEEEEEEE  RRRRRRRRRRRR     FFFFFFFFFFFFFFF
EEEEEEEEEEEEEEE  RRRRRRRRRRRR     FFFFFFFFFFFFFFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEEEEEEEEEEE     RRRRRRRRRRRR     FFFFFFFFFFFF
EEEEEEEEEEEE     RRRRRRRRRRRR     FFFFFFFFFFFF
EEEEEEEEEEEE     RRRRRRRRRRRR     FFFFFFFFFFFF
EEE              RRR   RRR        FFF
EEE              RRR   RRR        FFF
EEE              RRR   RRR        FFF
EEE              RRR      RRR     FFF
EEE              RRR      RRR     FFF
EEE              RRR      RRR     FFF
EEEEEEEEEEEEEEE  RRR         RRR  FFF
EEEEEEEEEEEEEEE  RRR         RRR  FFF
EEEEEEEEEEEEEEE  RRR         RRR  FFF
```

```
PPPPPPPP    UU      UU  DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
PPPPPPPP    UU      UU  DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
PP      PP  UU      UU  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
PP      PP  UU      UU  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
PP      PP  UU      UU  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
PP      PP  UU      UU  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
PPPPPPPP    UU      UU  DD      DD  RRRRRRRR      II      VV      VV  EEEEEEEE    RRRRRRRR
PPPPPPPP    UU      UU  DD      DD  RRRRRRRR      II      VV      VV  EEEEEEEE    RRRRRRRR
PP          UU      UU  DD      DD  RR  RR        II      VV      VV  EE          RR  RR
PP          UU      UU  DD      DD  RR  RR        II      VV      VV  EE          RR  RR
PP          UU      UU  DD      DD  RR    RR      II        VV  VV    EE          RR    RR        ....
PP          UU      UU  DD      DD  RR    RR      II        VV  VV    EE          RR    RR        ....
PP          UUUUUUUUUU  DDDDDDDD    RR      RR    IIIIII      VV      EEEEEEEEEE  RR      RR      ....
PP          UUUUUUUUUU  DDDDDDDD    RR      RR    IIIIII      VV      EEEEEEEEEE  RR      RR      ....

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
```

```
0001    C
0002    C Version:      'V04-000'
0003    C
0004    C*****************************************************************************
0005    C*                                                                           *
0006    C*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0007    C*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0008    C*  ALL RIGHTS RESERVED.                                                     *
0009    C*                                                                           *
0010    C*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0011    C*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0012    C*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0013    C*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0014    C*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0015    C*  TRANSFERRED.                                                             *
0016    C*                                                                           *
0017    C*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0018    C*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0019    C*  CORPORATION.                                                             *
0020    C*                                                                           *
0021    C*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0022    C*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0023    C*                                                                           *
0024    C*                                                                           *
0025    C*****************************************************************************
0026    C
0027
0028    c       Author  Brian Porter                    Creation date   10-FEB-1982
0029
0030    c++
0031    c       Functional description:
0032    c
0033    c       This module displays entries logged by pudriver.
0034    c
0035    c       Modified by:
0036    c
0037    C       V03-008 SAR0276         Sharon A. Reynolds      20-Jun-1984
0038    C               Added TMSCP message types.
0039    C
0040    C       V03-007 SAR0230         Sharon A. Reynolds,     28-Mar-1984
0041    C               Changed the call to UCB$L_OWNUIC to ORB$L_OWNER.
0042    C
0043    C       V03-006 SAR0198         Sharon A. Reynolds,     20-Feb-1984
0044    C               Added an SYE update that:
0045    C               - Adds additional AZTEC 'sa' error codes.
0046    C               - Adds RDRX support.
0047    C
0048    C       V03-005 SAR0148         Sharon A. Reynolds,     5-Oct-1983
0049    C               Added an SYE update that:
0050    C               - corrects a Fortran conversion error for micro-code rev.
0051    C               - corrects text descriptions and lengths.
0052    C               - adds AZTEC and TU81P(partial) support.
0053    C
0054    C       V03-004 SAR0091         Sharon A. Reynolds,     20-Jun-1983
0055    C               Changed the carriage control in the 'format' statements
0056    C               for use with ERF.
0057    C
```

```
0058    c       v03-003 BP0003          Brian Porter,           08-FEB-1983
0059    c               Corrected argument list for erllogmsg2.
0060    c
0061    c       v03-002 BP0002          Brian Porter,           25-MAY-1982
0062    c               Added 'option' filtering.
0063    c
0064    c       v03-001 BP0001          Brian Porter,           12-APR-1982
0065    c               Added more message types.
0066    c**
0067    c--
0068
0069            Subroutine PUDRIVER (lun)
0070
0071            include 'src$:msghdr.for /nolist'
0130            include 'src$:deverr.for /nolist'
0231
0232
0233
0234            byte            lun
0235
0236            integer*2       code_word
0237            integer*2       initialization_count
0238
0239            integer*4       vec$l_mapreg
0240
0241            integer*2       reserved
0242            integer*2       uda_sa
0243            integer*2       initialization_handshake(8)
0244
0245            equivalence     (emb(82),code_word)
0246            equivalence     (emb(84),initialization_count)
0247            equivalence     (emb(86),vec$l_mapreg)
0248            equivalence     (emb(90),uda_sa)
0249            equivalence     (emb(94),initialization_handshake)
0250
0251            character*33    vlstep1_sa_to_host(6:10)
0252
0253            Data            vlstep1_sa_to_host(6)
0254            1 /'PORT SUPPORTS ADDRESS MAPPING*'/
0255
0256            Data            vlstep1_sa_to_host(7)
0257            1 /'PORT ALLOWS HOST ODD ADDRESSES*'/
0258
0259            data            vlstep1_sa_to_host(8)
0260            1 /'ENHANCED DIAGNOSTICS IMPLEMENTED*'/
0261
0262            data            vlstep1_sa_to_host(9)
0263            1 /'22-BIT HOST ADDRESSING SUPPORTED*'/
0264
0265            data            vlstep1_sa_to_host(10)
0266            1 /'HOST-SETTABLE VECTOR UNSUPPORTED*'/
0267
0268            character*17    vlstep1_host_to_sa(7:7)
0269
0270            data            vlstep1_host_to_sa(7)
0271            1 /'INTERRUPT ENABLE*'/
0272
```

```
0273            character*21    v2step1_host_to_sa(14:14)
0274
0275            data            v2step1_host_to_sa(14)
0276            1 /'DIAGNOSTIC WRAP MODE*'/
0277
0278            character*21    v1step2_sa_to_host(6:6)
0279
0280            equivalence     (v1step2_sa_to_host,v2step1_host_to_sa)
0281
0282            character*6     v2step2_sa_to_host(15:15)
0283
0284            equivalence     (v2step2_sa_to_host,v1sa(15))
0285
0286            character*33    v1step2_host_to_sa(0:0)
0287
0288            data            v1step2_host_to_sa(0)
0289            1 /'HOST REQUESTS "PURGE" INTERRUPTS*'/
0290
0291            character*17    v1step3_sa_to_host(7:7)
0292
0293            equivalence     (v1step3_sa_to_host,v1step1_host_to_sa)
0294
0295            character*6     v2step3_sa_to_host(15:15)
0296
0297            equivalence     (v2step3_sa_to_host,v1sa(15))
0298
0299            character*31    v1step3_host_to_sa(15:15)
0300
0301            data            v1step3_host_to_sa(15)
0302            1 /'HOST REQUESTS POLL/PURGE TESTS*'/
0303
0304            character*6     v1step4_sa_to_host(15:15)
0305
0306            equivalence     (v1step4_sa_to_host,v1sa(15))
0307
0308            character*26    v1step4_host_to_sa(0:1)
0309
0310            data            v1step4_host_to_sa(0)
0311            1 /'GO*'/
0312
0313            data            v1step4_host_to_sa(1)
0314            1 /'HOST REQUESTS "LAST FAIL"*'/
0315
0316            common          sa
0317
0318            character*7     v1sa(11:15)
0319            common  /sa/    v1sa
0320
0321            data            v1sa(11)
0322            1 /'STEP 1*'/
0323
0324            data            v1sa(12)
0325            1 /'STEP 2*'/
0326
0327            data            v1sa(13)
0328            1 /'STEP 3*'/
0329
```

```
0330            data            vlsa(14)
0331            1 /'STEP 4*'/
0332
0333            data            vlsa(15)
0334            1 /'ERROR*'/
0335
0336            integer*4       compress4
0337            integer*4       compressc
0338            integer*4       ringbase_low
0339            integer*4       ringbase_high
0340            integer*4       burst
0341            integer*4       r_rng_lng
0342            integer*4       c_rng_lng
0343            integer*4       port_type
0344            integer*4       interrupt_vector
0345
0346
0347            call frctof (lun)
0348
0349            call header (lun)
0350
0351            call logger (lun,'DEVICE ATTENTION')
0352
0353            call linchk (lun,2)
0354
0355            if (code_word .eq. 1) then
0356
0357            write(lun,10) emb$t_dv_name(1:emb$b_dv_namlng),emb$w_dv_unit,
0358            1 ', INIT SEQUENCE COMPLETED'
0359    10      format('/' ','"DSA" PORT SUB-SYSTEM, UNIT ',a,
0360            1 i<compress4 (lib$extzv(0,16,emb$w_dv_unit))>,':',:a,
0361            1 :i<compress4 (lib$extzv(0,16,code_word))>,:a)
0362
0363            else if (code_word .eq. 2) then
0364
0365            write(lun,10) emb$t_dv_name(1:emb$b_dv_namlng),emb$w_dv_unit,
0366            1 ', INIT SEQUENCE FAILURE'
0367
0368            else if (code_word .eq. 3) then
0369
0370            write(lun,10) emb$t_dv_name(1:emb$b_dv_namlng),emb$w_dv_unit,
0371            1 ', "SA" ERROR BIT SET'
0372
0373            else if (code_word .eq. 4) then
0374
0375            write(lun,10) emb$t_dv_name(1:emb$b_dv_namlng),emb$w_dv_unit,
0376            1 ', UBA DATAPATH PURGE ERROR'
0377
0378            else if (code_word .eq. 5) then
0379
0380            write(lun,10) emb$t_dv_name(1:emb$b_dv_namlng),emb$w_dv_unit,
0381            1 ', UCODE REV AND "PUDRIVER" MIS-MATCH'
0382            else
0383
0384            write(lun,10) emb$t_dv_name(1:emb$b_dv_namlng),emb$w_dv_unit,
0385            1 ', "PUDRIVER" CODE #',code_word,'.'
0386            endif
```

```
0387
0388            call linchk (lun,2)
0389
0390            write(lun,15) uda_sa
0391    15      format(/' ',t8,'SA',t28,z4.4)
0392
0393            if (uda_sa .ne. 0) then
0394
0395            if (lib$extzv(15,1,uda_sa) .eq. 0) then
0396
0397            Call SA_NOERROR (lun,uda_sa)
0398            else
0399
0400            call sa_error (lun,uda_sa)
0401            endif
0402            endif
0403
0404            call linchk (lun,2)
0405
0406            write(lun,30)
0407    30      format(/' ','INIT SEQUENCE')
0408
0409            call linchk (lun,2)
0410
0411            write(lun,35) initialization_handshake(1)
0412    35      format(/' ',t8,'UCB$W_PORTSTEP1',t28,z4.4)
0413
0414            if (initialization_handshake(1) .ne. 0) then
0415
0416            if (lib$extzv(15,1,initialization_handshake(1)) .eq. 0) then
0417
0418            call output (lun,initialization_handshake(1),v1step1_sa_to_host,6,6,
0419            1 10,'0')
0420
0421            call output (lun,initialization_handshake(1),v1sa,11,11,15,'0')
0422            else
0423
0424            call sa_error (lun,initialization_handshake(1))
0425            endif
0426            endif
0427
0428            call linchk (lun,1)
0429
0430            write(lun,40) initialization_handshake(2)
0431    40      format(' ',t8,'UCB$W_HOSTSTEP1',t28,z4.4)
0432
0433            if (initialization_handshake(2) .ne. 0) then
0434
0435            interrupt_vector = lib$extzv(0,7,initialization_handshake(2))*4
0436
0437            call linchk (lun,1)
0438
0439            write(lun,45) interrupt_vector
0440    45      format(' ',t40,'INTERRUPT VECTOR ',o<compress4 (interrupt_vector)>,
0441            1 ' (OCTAL)')
0442
0443            call output (lun,initialization_handshake(2),v1step1_host_to_sa,7,7,7,
```

```
0444            1 '0')
0445
0446            r_rng_lng = 2**lib$extzv(8,3,initialization_handshake(2))
0447
0448            call linchk (lun,1)
0449
0450            write(lun,50) r_rng_lng
0451    50      format(' ',t40,i<compress4 (r_rng_lng)>,'. RING RESPONSE SLOTS')
0452
0453            c_rng_lng = 2**lib$extzv(11,3,initialization_handshake(2))
0454
0455            call linchk (lun,1)
0456
0457            write(lun,55) c_rng_lng
0458    55      format(' ',t40,i<compress4 (c_rng_lng)>,'. COMMAND RING SLOTS')
0459
0460            call output (lun,initialization_handshake(2),v2step1_host_to_sa,14,14,
0461            1 14,'0')
0462            endif
0463
0464            call linchk (lun,1)
0465
0466            write(lun,60) initialization_handshake(3)
0467    60      format(' ',t8,'UCB$W_PORTSTE^2',t28,z4.4)
0468
0469            if (initialization_handshake(3) .ne. 0) then
0470
0471            if (lib$extzv(15,1,initialization_handshake(3)) .eq. 0) then
0472
0473            r_rng_lng = 2**lib$extzv(0,3,initialization_handshake(3))
0474
0475            call linchk (lun,1)
0476
0477            write(lun,50) r_rng_lng
0478
0479            c_rng_lng = 2**lib$extzv(3,3,initialization_handshake(3))
0480
0481            call linchk (lun,1)
0482
0483            write(lun,55) c_rng_lng
0484
0485            call output (lun,initialization_handshake(3),v1step2_sa_to_host,
0486            1 6,6,6,'0')
0487
0488            port_type = lib$extzv(8,3,initialization_handshake(3))
0489
0490            call linchk (lun,1)
0491
0492            if (port_type .eq. 0) then
0493
0494            write(lun,65) 'UNIBUS/Q BUS STORAGE SYSTEMS PORT'
0495    65      format(' ',t40,a,:i<compress4 (port_type)>,:a)
0496            else
0497
0498            write(lun,65) 'PORT TYPE #',port_type,'.'
0499            endif
0500
```

```
0501            call output (lun,initialization_handshake(3),v1sa,11,11,15,'0')
0502            else
0503
0504            call sa_error (lun,initialization_handshake(3))
0505            endif
0506            endif
0507
0508            ringbase_low = 0
0509
0510            ringbase_high = 0
0511
0512            call linchk (lun,1)
0513
0514            write(lun,70) initialization_handshake(4)
0515    70      format(' ',t8,'UCB$W_HOSTSTEP2',t28,z4.4)
0516
0517            if (initialization_handshake(4) .ne. 0) then
0518
0519            call output (lun,initialization_handshake(4),v1step2_host_to_sa,0,0,
0520            1 0,'0')
0521
0522            ringbase_low = lib$extzv(1,15,initialization_handshake(4))*2
0523            endif
0524
0525            call linchk (lun,1)
0526
0527            write(lun,75) initialization_handshake(5)
0528    75      format(' ',t8,'UCB$W_PORTSTEP3',t28,z4.4)
0529
0530            if (initialization_handshake(5) .ne. 0) then
0531
0532            if (lib$extzv(15,1,initialization_handshake(5)) .eq. 0) then
0533
0534            interrupt_vector = lib$extzv(0,7,initialization_handshake(5))*4
0535
0536            call linchk (lun,1)
0537
0538            write(lun,45) interrupt_vector
0539
0540            call output (lun,initialization_handshake(5),v1step3_sa_to_host,7,7,
0541            1 7,'0')
0542
0543            call output (lun,initialization_handshake(5),v1sa,11,11,15,'0')
0544            else
0545
0546            call sa_error (lun,initialization_handshake(5))
0547            endif
0548            endif
0549
0550            call linchk (lun,1)
0551
0552            write(lun,80) initialization_handshake(6)
0553    80      format(' ',t8,'UCB$W_HOSTSTEP3',t28,z4.4)
0554
0555            if (initialization_handshake(6) .ne. 0) then
0556
0557            If (LIB$EXTZV(6,1,initialization_handshake(1)) .EQ. 0) then
```

```
0558            ringbase_high = lib$extzv(0,2,initialization_handshake(6))
0559            Endif
0560
0561            call calc_map (lun,0,ringbase_high,ringbase_low)
0562
0563            call output (lun,initialization_handshake(6),v1step3_host_to_sa,15,15,
0564            1 15,'0')
0565            endif
0566
0567            call linchk (lun,1)
0568
0569            write(lun,85) initialization_handshake(7)
0570    85      format(' ',t8,'UCB$W_PORTSTEP4',t28,z4.4)
0571
0572            if (initialization_handshake(7) .ne. 0) then
0573
0574            if (lib$extzv(15,1,initialization_handshake(7)) .eq. 0) then
0575
0576            Call SA_NOERROR (lun,initialization_handshake(7))
0577            else
0578
0579            call sa_error (lun,initialization_handshake(7))
0580            endif
0581            endif
0582
0583            call linchk (lun,1)
0584
0585            write(lun,90) initialization_handshake(8)
0586    90      format(' ',t8,'UCB$W_HOSTSTEP4',t28,z4.4)
0587
0588            if (initialization_handshake(8) .ne. 0) then
0589
0590            if (lib$extzv(15,1,initialization_handshake(8)) .eq. 0) then
0591
0592            call output (lun,initialization_handshake(8),v1step4_host_to_sa,0,0,
0593            1 1,'0')
0594
0595            burst = (lib$extzv(2,6,initialization_handshake(8)) + 1)*2
0596
0597            call linchk (lun,1)
0598
0599            if (burst .eq. 0) then
0600
0601            write(lun,95) 'CONTROLLER USING DEFAULT "BURST"'
0602    95      format(' ',t40,a,:i<compress4 (burst)>,:a)
0603            else
0604
0605            write(lun,95) '"BURST", ',burst,'. 16-BIT TRANSFER(S)'
0606            endif
0607            endif
0608            endif
0609
0610            call vecmapreg (lun,vec$l_mapreg)
0611
0612            call orb$l_owner (lun,emb$l_dv_ownuic)
0613
0614            call ucb$l_char (lun,emb$l_dv_char)
```

```
0615
0616            call ucb$w_sts (lun,emb$w_dv_sts)
0617
0618            call ucb$l_opcnt (lun,emb$l_dv_opcnt)
0619
0620            call ucb$w_errcnt (lun,emb$w_dv_errcnt)
0621
0622            call linchk (lun,2)
0623
0624            write(lun,100) (initialization_count,i = 1,2)
0625    100     format(' ',t8,'UCB$W_NUMBINITS',t28,z4.4,/,
0626            1 t40,i<compress4 (lib$extzv(0,16,initialization_count))>,
0627            1 '. INIT SEQUENCE(S)')
0628
0629            return
0630
0631
0632
0633            entry b_pudriver (lun)
0634
0635
0636
0637
0638            call header (lun)
0639
0640            call logger (lun,'DEVICE ATTENTION')
0641
0642            call linchk (lun,6)
0643
0644            if (code_word .eq. 1) then
0645
0646            write(lun,110) emb$t_dv_name(1:emb$b_dv_namlng),emb$w_dv_unit,
0647            1 ', INIT SEQUENCE COMPLETED'
0648    110     format(/' ',''DSA'' PORT SUB-SYSTEM, UNIT ',a,
0649            1 i<compress4 (lib$extzv(0,16,emb$w_dv_unit))>,':',:a,
0650            1 :i<compress4 (lib$extzv(0,16,code_word))>,:a)
0651
0652            else if (code_word .eq. 2) then
0653
0654            write(lun,10) emb$t_dv_name(1:emb$b_dv_namlng),emb$w_dv_unit,
0655            1 ', INIT SEQUENCE FAILURE'
0656
0657            else if (code_word .eq. 3) then
0658
0659            write(lun,10) emb$t_dv_name(1:emb$b_dv_namlng),emb$w_dv_unit,
0660            1 ', ''SA'' ERROR BIT SET'
0661
0662            else if (code_word .eq. 4) then
0663
0664            write(lun,10) emb$t_dv_name(1:emb$b_dv_namlng),emb$w_dv_unit,
0665            1 ', UBA DATAPATH PURGE ERROR'
0666
0667            else if (code_word .eq. 5) then
0668
0669            write(lun,10) emb$t_dv_name(1:emb$b_dv_namlng),emb$w_dv_unit,
0670            1 ', UCODE REV AND ''PUDRIVER'' MIS-MATCH'
0671            else
```

```
0672
0673            write(lun,10) emb$t_dv_name(1:emb$b_dv_naml ),emb$w_dv_unit,
0674            1 ' "PUDRIVER" CODE #',code_word,'.'
0675            endif
0676
0677            write(lun,115) 'SA','PSTEP1','HSTEP1','PSTEP2','HSTEP2','PSTEP3',
0678            1 'HSTEP3','PSTEP4','HSTEP4'
0679    115     format(/' ',t8,a,t15,a,t22,a,t29,a,t36,a,t43,a,t50,a,t57,a,t64,a)
0680
0681            write(lun,120) uda_sa,(initialization_handshake(i),i = 1,8)
0682    120     format(/' ',t8,z4.4,8(' ',z4.4))
0683
0684            return
0685
0686            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	3301	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	1001	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	1780	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
4	$BLANK	4	PIC OVR REL GBL SHR NOEXE RD WRT LONG
5	SA	35	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	6633	

ENTRY POINTS

Address	Type	Name	Address	Type	Name
0-00000919		B_PUDRIVER	0-00000000		PUDRIVER

VARIABLES

Address	Type	Name	Address	Type	Name
2-00000150	I*4	BURST	3-00000052	I*2	CODE_WORD
2-00000144	I*4	COMPRESSC	2-00000158	I*4	C_RNG_LNG
3-0000001C	L*1	EMB$B_DV_CLASS	3-00000010	L*1	EMB$B_DV_ERTCNT
3-00000011	L*1	EMB$B_DV_ERTMAX	3-0000003E	L*1	EMB$B_DV_NAMLNG
3-0000003A	L*1	EMB$B_DV_SLAVE	3-0000001D	L*1	EMB$B_DV_TYPE
3-00000036	I*4	EMB$L_DV_CHAR	3-00000012	I*4	EMB$L_DV_IOSB1
3-00000016	I*4	EMB$L_DV_IOSB2	3-00000026	I*4	EMB$L_DV_MEDIA
3-0000004E	I*4	EMB$L_DV_NUMREG	3-0000002E	I*4	EMB$L_DV_OPCNT
3-00000032	I*4	EMB$L_DV_OWNUIC	3-0000001E	I*4	EMB$L_DV_RQPID
3-00000000	I*4	EMB$L_HD_SID	3-0000003F	CHAR	EMB$T_DV_NAME
3-00000024	I*2	EMB$W_DV_BCNT	3-00000022	I*2	EMB$W_DV_BOFF
3-0000002C	I*2	EMB$W_DV_ERRCNT	3-0000003C	I*2	EMB$W_DV_FUNC
3-0000001A	I*2	EMB$W_DV_STS	3-0000002A	I*2	EMB$W_DV_UNIT

3-00000004	I*2	EMB$W_HD_ENTRY	3-0000000E	I*2	EMB$W_HD_ERRSEQ
2-00000164	I*4	I	3-00000054	I*2	INITIALIZATION_COUNT
2-00000160	I*4	INTERRUPT_VECTOR	AP-00000004@	L*1	LUN
2-0000015C	I*4	PORT_TYPE	2-00000140	I*2	RESERVED
2-0000014C	I*4	RINGBASE_HIGH	2-00000148	I*4	RINGBASE_LOW
2-00000154	I*4	R_RNG_LNG	4-00000000	R*4	SA
3-0000005A	I*2	UDA_SA	3-00000056	I*4	VEC$L_MAPREG

ARRAYS

Address	Type	Name	Bytes	Dimensions
3-000C0000	L*1	EMB	512	(0:511)
3-00000052	I*4	EMB$L_DV_REGSAV	420	(0:104)
3-00000006	I*4	EMB$Q_HD_TIME	8	(2)
3-0000005E	I*2	INITIALIZATION_HANDSHAKE	16	(8)
5-00000000	CHAR	V1SA	35	(11:15)
2-00000000	CHAR	V1STEP1_HOST_TO_SA	17	(7:7)
2-00000026	CHAR	V1STEP1_SA_TO_HOST	165	(6:10)
2-000000CB	CHAR	V1STEP2_HOST_TO_SA	33	(0:0)
2-00000011	CHAR	V1STEP2_SA_TO_HOST	21	(6:6)
2-000000EC	CHAR	V1STEP3_HOST_TO_SA	31	(15:15)
2-00000000	CHAR	V1STEP3_SA_TO_HOST	17	(7:7)
2-0000010B	CHAR	V1STEP4_HOST_TO_SA	52	(0:1)
5-0000001C	CHAR	V1STEP4_SA_TO_HOST	6	(15:15)
2-00000011	CHAR	V2STEP1_HOST_TO_SA	21	(14:14)
5-0000001C	CHAR	V2STEP2_SA_TO_HOST	6	(15:15)
5-0000001C	CHAR	V2STEP3_SA_TO_HOST	6	(15:15)

LABELS

Address	Label	Address	Label	Address	Label	Address	Label	Address	Label	Address	Label
1-00000174	10'	1-000001AD	15'	1-000001BD	30'	1-000001D1	35'	1-000001EE	40'	1-0000020A	45'
1-00000233	50'	1-00000256	55'	1-00000278	60'	1-00000294	65'	1-000002A4	70'	1-000002C0	75'
1-000002DC	80'	1-000002F8	85'	1-00000314	90'	1-00000330	95'	1-00000340	100'	1-00000379	110'
1-000003B2	115'	1-000003D2	120'								

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name	Type	Name	Type	Name	Type	Name
	CALC_MAP	I*4	COMPRESS4		FRCTOF		HEADER	I*4	LIB$EXTZV		LINCHK
	LOGGER		ORB$L_OWNER		OUTPUT		SA_ERROR		SA_NOERROR		UCB$L_CHAR
	UCB$L_OPCNT		UCB$W_ERRCNT		UCB$W_STS		VECMAPREG				

```
0001
0002
0003            Subroutine SA_NOERROR (lun,sa_register)
0004
0005
0006            implicit        none
0007
0008            byte            lun
0009
0010            integer*2       sa_register
0011
0012            integer*4       micro_code_revision
0013            integer*4       port_type
0014            integer*4       lib$extzv
0015            integer*4       compress4
0016
0017            character*7     vlsa(11:15)
0018            common  /sa/    vlsa
0019
0020
0021
0022            micro_code_revision = lib$extzv(0,4,sa_register)
0023
0024            call linchk (lun,1)
0025
0026            write(lun,10) micro_code_revision
0027    10      format(' ',t40,'CONTROLLER MICRO-CODE #',
0028            1 i<compress4 (micro_code_revision)>,'.')
0029
0030            port_type = lib$extzv(4,4,sa_register)
0031
0032            call linchk (lun,1)
0033
0034            if (port_type .eq. 0) then
0035
0036            write(lun,15) 'UDA50'
0037    15      format(' ',t40,'PORT IS ',a)
0038
0039            else if (port_type .eq. 1) then
0040
0041            write(lun,15) 'RC25'
0042
0043            else if (port_type .eq. 5) then
0044
0045            write(lun,15) 'TU81P'
0046
0047            else if (port_type .eq. 6) then
0048
0049            write(lun,15) 'UDA50A'
0050
0051            Else if (port_type .EQ. 7) then
0052
0053            Write(lun,15) 'RDRX'
0054
0055            else
0056            write(lun,20) 'PORT TYPE #',port_type
0057    20      format(' ',t40,a,i<compress4 (port_type)>,'.')
```

```
0058            endif
0059
0060            call output (lun,sa_register,v1sa,11,11,15,'0')
0061
0062            return
0063
0064            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	408	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	130	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	188	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	SA	35	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	761	

ENTRY POINTS

Address	Type	Name
0-00000000		SA_NOERROR

VARIABLES

Address	Type	Name	Address	Type	Name
AP-00000004@	L*1	LUN	2-00000000	I*4	MICRO_CODE_REVISION
2-00000004	I*4	PORT_TYPE	AP-00000008@	I*2	SA_REGISTER

ARRAYS

Address	Type	Name	Bytes	Dimensions
3-00000000	CHAR	V1SA	35	(11:15)

LABELS

Address	Label	Address	Label	Address	Label
1-00000039	10'	1-00000061	15'	1-00000072	20'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name	Type	Name
I*4	COMPRESS4	I*4	LIB$EXTZV		LINCHK		OUTPUT

```
0001
0002
0003
0004          Subroutine SA_ERROR (lun,sa_register)
0005
0006
0007          Implicit        None
0008
0009          byte            lun
0010
0011          integer*2       sa_register
0012
0013          Integer*4       lib$extzv
0014
0015          character*34    port_generic_sa_error_code(0:21)
0016
0017          data            port_generic_sa_error_code(0)
0018          1 /'UDA IDLE*'/
0019
0020          data            port_generic_sa_error_code(1)
0021          1 /'PACKET READ, PE/TIMEOUT*'/
0022
0023          data            port_generic_sa_error_code(2)
0024          1 /'PACKET WRITE, PE/TIMEOUT*'/
0025
0026          data            port_generic_sa_error_code(3)
0027          1 /'UDA ''ROM'' OR ''RAM'' PARITY ERROR*'7
0028
0029          data            port_generic_sa_error_code(4)
0030          1 /'UDA ''RAM'' PARITY ERROR*'7
0031
0032          data            port_generic_sa_error_code(5)
0033          1 /'UDA ''ROM'' PARITY ERROR*'7
0034
0035          data            port_generic_sa_error_code(6)
0036          1 /'RING READ PARITY ERROR/TIMEOUT*'/
0037
0038          data            port_generic_sa_error_code(7)
0039          1 /'RING WRITE PARITY ERROR/TIMEOUT*'7
0040
0041          data            port_generic_sa_error_code(8)
0042          1 /'INTERRUPT MASTER ERROR*'7
0043
0044          data            port_generic_sa_error_code(9)
0045          1 /'HOST ACCESS TIMEOUT*'/
0046
0047          data            port_generic_sa_error_code(10)
0048          1 /'CREDIT LIMIT EXCEEDED*'/
0049
0050          data            port_generic_sa_error_code(11)
0051          1 /'UNIBUS MASTER ERROR*'/
0052
0053          data            port_generic_sa_error_code(12)
0054          1 /'DIAGNOSTIC FATAL ERROR*'7
0055
0056          data            port_generic_sa_error_code(13)
0057          1 /'INSTRUCTION LOOP TIMEOUT*'/
```

```
0058
0059            data            port_generic_sa_error_code(14)
0060            1 /'INVALID CONNECTION IDENTIFIER*'/
0061
0062            data            port_generic_sa_error_code(15)
0063            1 /'INTERRUPT WRITE*'/
0064
0065            data            port_generic_sa_error_code(16)
0066            1 /'MAINTENANCE READ/WRITE FAILURE*'/
0067
0068            data            port_generic_sa_error_code(17)
0069            1 /'MAINTENANCE WRITE FAILURE*'/
0070
0071            data            port_generic_sa_error_code(18)
0072            1 /'CONTROLLER ''RAM'' FAILURE*'/
0073
0074            data            port_generic_sa_error_code(19)
0075            1 /'INITIALIZATION SEQUENCE FAILURE*'/
0076
0077            data            port_generic_sa_error_code(20)
0078            1 /'PROTOCOL INCOMPATIBILITY ERROR*'/
0079
0080            data            port_generic_sa_error_code(21)
0081            1 /'PURGE/POLL HARDWARE FAILURE*'/
0082
0083
0084            character*35    aztec_sa_error_code('310'o:'356'o)
0085
0086            data            aztec_sa_error_code('310'o)
0087            1 /'READ/WRITE ERROR ON INTERRUPT*'/
0088
0089            data            aztec_sa_error_code('311'o)
0090            1 /'INCONSISTENCY AT ^U.BFIL''*'/
0091
0092            data            aztec_sa_error_code('312'o)
0093            1 /'INCONSISTENCY AT ^U.BMTY''*'/
0094
0095            data            aztec_sa_error_code('313'o)
0096            1 /'INCONSISTENCY AT ^U.ALOC''*'/
0097
0098            data            aztec_sa_error_code('314'o)
0099            1 /'INVALID SERVO ENTRY (PIP SET)*'/
0100
0101            data            aztec_sa_error_code('315'o)
0102            1 /'INVALID AT SERVO ENTRY (ERROR SET)*'/
0103
0104            data            aztec_sa_error_code('316'o)
0105            1 /'INCONSISTENCY AT ^U.SEND''*'/
0106
0107            data            aztec_sa_error_code('317'o)
0108            1 /'INCONSISTENCY AT ^U.RECV''*'/
0109
0110            data            aztec_sa_error_code('320'o)
0111            1 /'INCONSISTENCY AT ^U.ATTN''*'/
0112
0113            data            aztec_sa_error_code('321'o)
0114            1 /'INCONSISTENCY AT ^U.ONLN''*'/
```

```
0115
0116            data                aztec_sa_error_code('322'o)
0117            1 /'ILLEGAL "D" REQUEST (U.QDRQ)*'/
0118
0119            data                aztec_sa_error_code('323'o)
0120            1 /'FENCE-POST ERROR AT ^PROTAB"*'/
0121
0122            data                aztec_sa_error_code('324'o)
0123            1 /'BAD PACKET DEQUEUED AT 'U.DONE"*'/
0124
0125            data                aztec_sa_error_code('325'o)
0126            1 /'"DM" PROGRAM ILLEGAL MEMORY STORE*'/
0127
0128            data                aztec_sa_error_code('326'o)
0129            1 /'"DUP" D-Q FAILED (XFC 34/35)*'/
0130
0131            data                aztec_sa_error_code('327'o)
0132            1 /'INCONSISTENCY AT ^U.ATST"*'/
0133
0134            data                aztec_sa_error_code('330'o)
0135            1 /'INCONSISTENCY AT ^U.SEKO"*'/
0136
0137            data                aztec_sa_error_code('331'o)
0138            1 /'INCONSISTENCY AT ^U.CKSV"*'/
0139
0140            data                aztec_sa_error_code('332'o)
0141            1 /'"D.OPCD" FOUND ILLEGAL OPCODE*'/
0142
0143            data                aztec_sa_error_code('333'o)
0144            1 /'"D.CSF" FOUND ILLEGAL OPCODE*'/
0145
0146            data                aztec_sa_error_code('334'o)
0147            1 /'UNKNOWN BAD DRIVE STATUS, ^D.DSTS"*'/
0148
0149            data                aztec_sa_error_code('335'o)
0150            1 /'ILLEGAL "XFC" EXECUTED BY ^DM"*'/
0151
0152            data                aztec_sa_error_code('336'o)
0153            1 /'"D" PICKED UP A ZERO ^SCB.DB"*'/
0154
0155            data                aztec_sa_error_code('337'o)
0156            1 /'INCONSISTENCY AT ^D" IDLE LOOP*'/
0157
0158            data                aztec_sa_error_code('340'o)
0159            1 /'"DM" WORD COUNT ERROR*'/
0160
0161            data                aztec_sa_error_code('341'o)
0162            1 /'UNKNOWN DISPLAY FAULT, "D.DFLT"*'/
0163
0164            data                aztec_sa_error_code('342'o)
0165            1 /'DRIVE NOT FAULTING, ^P.OFLN" STATE*'/
0166
0167            data                aztec_sa_error_code('343'o)
0168            1 /'"U" POWER-UP DIAGNOSTICS FAILED*'/
0169
0170            data                aztec_sa_error_code('344'o)
0171            1 /'"D" POWER-UP DIAGNOSTICS FAILED*'/
```

```
0172
0173            data                aztec_sa_error_code('345'o)
0174            1 /'ADAPTER CARD FAILURE*'/
0175
0176            data                aztec_sa_error_code('346'o)
0177            1 /''EC.TMR'' TIMED OUT*'7
0178
0179            data                aztec_sa_error_code('347'o)
0180            1 /''U.SEND/U.RECV'' RING READ TIMEOUT*'/
0181
0182            data                aztec_sa_error_code('350'o)
0183            1 /''WAITRV'' REASON AT ''D.RVCT''*'/
0184
0185            data                aztec_sa_error_code('351'o)
0186            1 /''D.ARCS'', CLOSEST UNDONE ZONE LOST*'/
0187
0188            data                aztec_sa_error_code('352'o)
0189            1 /''U.SEEK'', SEEK TO ILLEGAL TRACK*'/
0190
0191            data                aztec_sa_error_code('353'o)
0192            1 /''U.HTST'', INIT DIAG WRITE FAILED*'/
0193
0194            data                aztec_sa_error_code('354'o)
0195            1 /''U.HTST'', INIT DIAG DMA FAILED*'/
0196
0197            data                aztec_sa_error_code('355'o)
0198            1 /''U.SYDR'' - ''SS.DER'' T, ''SS.SPN'' 0*'/
0199
0200            data                aztec_sa_error_code('356'o)
0201            1 /'MASTER DRIVE ACLO ASSERTED*'/
0202
0203            character*7         vlsa(11:15)
0204            common  /sa/        vlsa
0205
0206            integer*4           error_code
0207
0208            integer*2           lastfail_code
0209
0210            integer*4           compress4
0211
0212            integer*4           compressc
0213
0214
0215
0216            error_code = lib$extzv(0,11,sa_register)
0217
0218            call linchk (lun,1)
0219
0220            If (error_code .LE. 99) then
0221
0222            if (
0223            1 error_code .gt. 0
0224            1 .and.
0225            1 error_code .lt. 22
0226            1 ) then
0227
0228            write(lun,20) port_generic_sa_error_code(error_code)
```

```
0229     20       format(' ',t40,a<compressc (port_generic_sa_error_code(error_code))>)
0230              Endif
0231
0232     C
0233     C        AZTEC
0234     C
0235              Else if (
0236              1 error_code .GE. '310'o
0237              1 .AND.
0238              1 error_code .LE. '356'o
0239              1 ) then
0240
0241              Write (lun,40) aztec_sa_error_code(error_code)
0242     40       Format(' ',t40,A<COMPRESSC (aztec_sa_error_code(error_code))>)
0243              Else
0244
0245              write(lun,100) error_code
0246     100      format(' ',t40,'ERROR CODE #',i<compress4 (error_code)>,'.')
0247              endif
0248
0249              call output (lun,sa_register,v1sa,11,11,15,'0')
0250
0251              return
0252
0253
0254
0255              Entry   UDA_LASTFAIL_ERROR (lun,lastfail_code)
0256
0257
0258              call linchk (lun,2)
0259
0260              write(lun,27) '"LASTFAIL" CODE',lastfail_code
0261     27       format(' ',t8,a,t28,z4.4)
0262
0263              error_code = lib$extzv (0,16,lastfail_code)
0264
0265              if (
0266              1 lastfail_code .ge. 0
0267              1 .and.
0268              1 lastfail_code .le. 22
0269              1 ) then
0270
0271              write(lun,20) port_generic_sa_error_code(error_code)
0272              else
0273
0274              write(lun,30) error_code
0275     30       format(' ',t40,'ERROR CODE #',i<compress4 (error_code)>,'.')
0276              endif
0277
0278              return
0279
0280              end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	508	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	135	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	2304	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	SA	35	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	2982	

ENTRY POINTS

Address	Type	Name	Address	Type	Name
0-00000000		SA_ERROR	0-000000E5		UDA_LASTFAIL_ERROR

VARIABLES

Address	Type	Name	Address	Type	Name	Address	Type	Name	Address	Type	Name
2-00000844	I*4	ERROR_CODE	AP-00000008@	I*2	LASTFAIL_CODE	AP-00000004@	L*1	LUN	AP-00000008@	I*2	SA_REGISTER

ARRAYS

Address	Type	Name	Bytes	Dimensions
2-000002EC	CHAR	AZTEC_SA_ERROR_CODE	1365	(200:238)
2-00000000	CHAR	PORT_GENERIC_SA_ERROR_CODE	748	(0:21)
3-00000000	CHAR	V1SA	35	(11:15)

LABELS

Address	Label	Address	Label	Address	Label	Address	Label	Address	Label
1-00000029	20'	1-0000005E	27'	1-0000006A	30'	1-00000035	40'	1-00000041	100'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name	Type	Name	Type	Name
I*4	COMPRESS4	I*4	COMPRESSC	I*4	LIB$EXTZV		LINCHK		OUTPUT

```
0001
0002
0003
0004          Subroutine PUDRIVER_MSCP_DISPATCHER (lun,option,reccnt,
0005          1 record_length)
0006
0007
0008          include 'src$:msghdr.for /nolist'
0067          include 'src$:emblmdef.for /nolist'
0136          include 'src$:embspdef.for /nolist'
0249
0250
0251          byte            lun
0252
0253          character*1     option
0254
0255          integer*4       reccnt
0256          integer*4       packet_length
0257          integer*4       record_length
0258
0259          byte            mslg$b_format
0260          equivalence     (emb(46),mslg$b_format)
0261
0262          if (
0263          1 option .eq. 'S'
0264          1 .or.
0265          1 option .eq. 'B'
0266          1 ) then
0267
0268          if (emb$w_hd_entry .eq. 100) then       ! Logmessage entry
0269
0270          call frctof (lun)
0271          call header2 (lun,reccnt)
0272          call logger (lun,'ERL$LOGMESSAGE ENTRY')
0273
0274          call dhead2 (lun,''DSA'' PORT',
0275          1 emb$b_lm_namlng,emb$t_lm_name,emb$w_lm_unit)
0276
0277          Packet_length = record_length - 39
0278
0279          if (mslg$b_format .eq. 0) then          ! Controller error
0280
0281          if (option .eq. 'S') then
0282          call mslg$k_cnt_err (lun,packet_length)
0283          endif
0284
0285          else if (mslg$b_format .eq. 1) then     ! Memory access error
0286
0287          if (option .eq. 'S') then
0288          call mslg$k_bus_addr (lun,packet_length)
0289          endif
0290
0291          else if (
0292          1 mslg$b_format .eq. 2  ! Disk transfer error - mslg$k_disk_trn
0293          1 .OR.
0294          1 mslg$b_format .EQ. 5  ! Tape transfer error - mslg$k_tape_trn
0295          1 ) then
```

```
0296
0297            if (option .eq. 'S') then
0298            call DISK_TAPE_TRANSFER_ERRORS (lun,packet_length)
0299            endif
0300
0301            else if (               ! SDI/STI errors
0302            1 mslg$b_format .eq. 3  ! SDI comm error (disk) - mslg$k_sdi
0303            1 .OR.
0304            1 mslg$b_format .EQ. 6  ! STI comm or cmd failure - mslg$k_sti_err
0305            1 .OR.
0306            1 mslg$b_format .EQ. 7  ! STI drive error - mslg$k_sti_del
0307            1 .OR.
0308            1 mslg$b_format .EQ. 8  ! STI formatter error - mslg$k_sti_fel
0309            1 ) then
0310
0311            if (option .eq. 'S') then
0312            call SDI_STI_ERRORS (lun,packet_length)
0313            endif
0314
0315            else if (mslg$b_format .eq. 4) then     ! Small disk error
0316
0317            if (option .eq. 'S') then
0318            call mslg$k_sml_dsk (lun,packet_length)
0319            endif
0320            else
0321
0322            call erllogmsg2 (lun,record_length)
0323            endif
0324
0325            else if (emb$w_hd_entry .eq. 99) then   ! Logstatus entry
0326
0327            call frctof (lun)
0328            call header2 (lun,reccnt)
0329            call logger (lun,'ERL$LOGSTATUS ENTRY')
0330
0331            call dhead2 (lun,''DSA'' PORT',
0332            1 emb$b_sp_namlng,emb$t_sp_name,emb$w_sp_unit)
0333
0334            call erllogsts2 (lun)
0335            endif
0336            endif
0337
0338            return
0339            end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	379	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	52	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	156	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	1099	

ENTRY POINTS

Address	Type	Name
0-00000000		PUDRIVER_MSCP_DISPATCHER

VARIABLES

Address	Type	Name	Address	Type	Name
3-00000010	L*1	EMB$B_LM_CLASS	3-00000014	L*1	EMB$B_LM_NAMLNG
3-00000011	L*1	EMB$B_LM_TYPE	3-00000010	L*1	EMB$B_SP_CLASS
3-00000040	L*1	EMB$B_SP_NAMLNG	3-00000011	L*1	EMB$B_SP_TYPE
3-00000000	I*4	EMB$L_HD_SID	3-00000014	I*4	EMB$L_SP_BCNT
3-00000038	I*4	EMB$L_SP_CHAR	3-0000003C	I*4	EMB$L_SP_CMDREF
3-00000020	I*4	EMB$L_SP_IOSB1	3-00000024	I*4	EMB$L_SP_IOSB2
3-00000018	I*4	EMB$L_SP_MEDIA	3-0000002C	I*4	EMB$L_SP_OPCNT
3-00000034	I*4	EMB$L_SP_OWNUIC	3-00000C1C	I*4	EMB$L_SP_RQPID
3-00000015	CHAR	EMB$T_LM_NAME	3-00000041	CHAR	EMB$T_SP_NAME
3-00000004	I*2	EMB$W_HD_ENTRY	3-0000000E	I*2	EMB$W_HD_ERRSEQ
3-00000024	I*2	EMB$W_LM_MSGTYP	3-00000012	I*2	EMB$W_LM_UNIT
3-00000012	I*2	EMB$W_SP_BOFF	3-00000030	I*2	EMB$W_SP_ERRCNT
3-00000028	I*2	EMB$W_SP_FUNC	3-00000032	I*2	EMB$W_SP_STS
3-0000002A	I*2	EMB$W_SP_UNIT	AP-0000004@	L*1	LUN
3-0000002E	L*1	MSLG$B_FORMAT	AP-00000008@	CHAR	OPTION
2-00000000	I*4	PACKET_LENGTH	AP-0000000C@	I*4	RECCNT
AP-00000010@	I*4	RECORD_LENGTH			

ARRAYS

Address	Type	Name	Bytes	Dimensions
3-00000000	L*1	EMB	512	(0:511)
3-00000026	L*1	EMB$B_LM_MSGTXT	460	(460)
3-00000006	I*4	EMB$Q_HD_TIME	8	(2)

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
	DHEAD2		DISK_TAPE_TRANSFER_ERRORS		ERLLOGMSG2
	ERLLOGSTS2		FRCTOF		HEADER2
	LOGGER		MSLG$K_BUS_ADDR		MSLG$K_CNT_ERR
	MSLG$K_SML_DSK		SDI_STI_ERRORS		

COMMAND QUALIFIERS

```
  FORTRAN /LIS=LIS$:PUDRIVER/OBJ=OBJ$:PUDRIVER MSRC$:PUDRIVER

  /CHECK=(NOBOUNDS,OVERFLOW,NOUNDERFLOW)
  /DEBUG=(NOSYMBOLS,TRACEBACK)
  /STANDARD=(NOSYNTAX,NOSOURCE_FORM)
  /SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP)
  /F77  /NOG_FLOATING  /I4  /OPTIMIZE  /WARNINGS  /NOD_LINES  /NOCROSS_REFERENCE  /NOMACHINE_CODE  /CONTINUATIONS=19
```

COMPILATION STATISTICS

```
  Run Time:           14.08 seconds
  Elapsed Time:       32.70 seconds
  Page Faults:        280
  Dynamic Memory:     250 pages
```

0153 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

RECSELECT
LIS

RLDISK
LIS

PUDRIVER
LIS

PCL11T
LIS

ROLLUP
LIS

RXDISK
LIS

PCL11R
LIS

SB11
LIS

RKDISK
LIS