



PAI  
02  
02  
02  
02  
02  
02  
02  
02  
02  
02  
02  
02  
02  
02  
02  
02  
02  
02  
02  
02  
02  
02  
02  
02  
02  
02  
02  
03  
03  
03  
03  
03  
03  
03  
03  
03  
03  
03  
03  
03  
03  
03  
03  
03  
03  
03  
03  
03  
03  
03  
03

PPPPPPPP	AAAAAA	DDDDDDDD	RRRRRRRR	IIIIII	VV	VV	EEEEEEEEEE	RRRRRRRR	
PPPPPPPP	AAAAAA	DDDDDDDD	RRRRRRRR	IIIIII	VV	VV	EEEEEEEEEE	RRRRRRRR	
PP	AA	DD	RR	II	VV	VV	EE	RR	RR
PP	AA	DD	RR	II	VV	VV	EE	RR	RR
PP	AA	DD	RR	II	VV	VV	EE	RR	RR
PP	AA	DD	RR	II	VV	VV	EE	RR	RR
PPPPPPPP	AA	DD	RRRRRRRR	II	VV	VV	EEEEEEEEEE	RRRRRRRR	
PPPPPPPP	AA	DD	RRRRRRRR	II	VV	VV	EEEEEEEEEE	RRRRRRRR	
PP	AAAAAAAAAA	DD	RR	II	VV	VV	EE	RR	RR
PP	AAAAAAAAAA	DD	RR	II	VV	VV	EE	RR	RR
PP	AA	DD	RR	II	VV	VV	EE	RR	RR
PP	AA	DD	RR	II	VV	VV	EE	RR	RR
PP	AA	DD	RR	II	VV	VV	EE	RR	RR
PP	AA	DD	RR	II	VV	VV	EE	RR	RR
PP	AA	DD	RR	II	VV	VV	EE	RR	RR
PP	AA	DD	RR	II	VV	VV	EE	RR	RR
PP	AA	DDDDDDDD	RR	IIIIII	VV	VV	EEEEEEEEEE	RR	RR
PP	AA	DDDDDDDD	RR	IIIIII	VV	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
LL	II	SS
LLLLLLLLLLLL	IIIIII	SSSSSSSS
LLLLLLLLLLLL	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 22-FEB-1982
0029
0030 c++
0031 c Functional description:
0032 c
0033 c This module displays entries made by the padriver.
0034 c
0035 c Modified by:
0036 c
0037 c V03-010 EAD0178 Elliott A. Drayton 24-May-1984
0038 c Added code to handle zero length HSC datagram message.
0039 c
0040 c V03-009 EAD0173 Elliott A. Drayton 9-May-1984
0041 c Added code to prevent HSC datagram format overflow.
0042 c
0043 c V03-008 EAD0122 Elliott A. Drayton 24-Mar-1984
0044 c Changed PA error title for subtype 7.
0045 c
0046 c V03-007 EAD0121 Elliott A. Drayton 24-Mar-1984
0047 c Add support for new PA errors subtypes 2,7, and 8.
0048 c
0049 c V03-006 SAR0199 Sharon A. Reynolds, 20-Feb-1984
0050 c Added an SYE update that:
0051 c - Fixed an incorrect path number being reported.
0052 c
0053 c V03-005 SAR0164 Sharon A. Reynolds, 13-Oct-1983
0054 c - Added an SYE update that implements new spec
0055 c changes for PSR/PESR.
0056 c - Fixed a bug in the padriver_attention_error_code
0057 c routine.
```

E 15  
16-Sep-1984 00:11:24  
5-Sep-1984 14:10:51

VAX-11 FORTRAN V3.4-56  
DISK\$VMSMASTER:[ERF.SRC]PADR:VER.FOR;1

```
0058 C
0059 C
0060 C
0061 C
0062 C
0063 C
0064 C
0065 C
0066 C
0067 c
0068 c
0069 c
0070 c
0071 c**
0072 c--
0073
0074
0075
0076
0135
0236
0237
0238
0239
0240
0241
0242
0243
0244
0245
0246
0247
0248
0249
0250
0251
0252
0253
0254
0255
0256
0257
0258
0259
0260
0261
0262
0263
0264
0265
0266
0267
0268
0269
0270
0271
0272
```

```
V03-004 SAR0088 Sharon A. Reynolds, 20-Jun-1983
Changed the carriage control in the 'format' statements
for use with ERF.

V03-003 SAR0057 Sharon A. Reynolds, 15-Jun-1983
Removed brief/cryptic support.

v03-002 BP0002 Brian Porter, 20-AUG-1982
Added ci750.

v03-001 BP0001 Brian Porter, 22-JUL-1982
Corrected 'ppd$b_flags' conversion error.
```

```
Subroutine PADRIVER_ATTENTION780 (lun)
```

```
include 'src$:msghdr.for /nolist'
include 'src$:deverr.for /nolist'
```

```
byte lun

integer*4 padriver_error_type_code
integer*4 pcnfr
integer*4 pmcsr
integer*4 psr
integer*4 pfar
integer*4 pesr
integer*4 ppr
integer*4 pmadr
integer*4 pmdatr
integer*4 correct_control_store_value
integer*4 compress4

logical*1 diagnostic_mode

equivalence (emb$l_dv_regsav(0),padriver_error_type_code)
equivalence (emb$l_dv_regsav(1),pcnfr)
equivalence (emb$l_dv_regsav(2),pmcsr)
equivalence (emb$l_dv_regsav(3),psr)
equivalence (emb$l_dv_regsav(4),pfar)
equivalence (emb$l_dv_regsav(5),pesr)
equivalence (emb$l_dv_regsav(6),ppr)
equivalence (emb$l_dv_regsav(7),pmadr)
equivalence (emb$l_dv_regsav(8),pmdatr)
equivalence (emb$l_dv_regsav(9),correct_control_store_value)

call frctof (lun)
call header (lun)
call logger (lun,'DEVICE ATTENTION')
call padriver_attention_error_code (lun,padriver_error_type_code)
```

```
0273
0274      call padriver_initialization (lun,padriver_error_type_code)
0275
0276      if (lib$extzv(8,7,padriver_error_type_code) .eq. 0) goto 75
0277
0278      c
0279      c      set not diagnostic mode for now
0280      c
0281
0282      diagnostic_mode = .false.
0283
0284      if (.not. diagnostic_mode) then
0285
0286      call ci780_rega (lun,pcnfgr)
0287      else
0288
0289      call linchk (lun,2)
0290
0291      write(lun,5) pcnfgr
0292      5      format('/' ,t8,'CNFGR',t24,z8.8)
0293      endif
0294
0295      call ci_pmcsr (lun,pmcsr,diagnostic_mode)
0296
0297      call ci_psr (lun,psr,diagnostic_mode)
0298
0299      call linchk (lun,1)
0300
0301      write(lun,10) pfar
0302      10     format(' ',t8,'PFAR',t24,z8.8)
0303
0304      call ci_pesr (lun,pesr,psr,diagnostic_mode)
0305
0306      call ci_ppr (lun,ppr,psr,diagnostic_mode)
0307
0308      call ci_control_store_mismatch (lun,pmadr,pmdatr,
0309      1 correct_control_store_value,padriver_error_type_code,diagnostic_mode)
0310
0311      call linchk (lun,1)
0312
0313      write(lun,15)
0314      15     format(' ',:)
0315
0316      call ucb$b_ertcnt (lun,lib$extzv(16,8,padriver_error_type_code))
0317
0318      call ucb$b_ertmax (lun,lib$extzv(24,8,padriver_error_type_code))
0319
0320      call ucb$l_char (lun,emb$l_dv_char)
0321
0322      call ucb$w_sts (lun,emb$w_dv_sts)
0323
0324      call ucb$w_errcnt (lun,emb$w_dv_errcnt)
0325
0326      75     return
0327      End
```

0001  
0002  
0003  
0004  
0005  
0006  
0007  
0008  
0009  
0010  
0011  
0012  
0013  
0014  
0015  
0016  
0017  
0018  
0019  
0020  
0021  
0022  
0023  
0024  
0025  
0026  
0027  
0028  
0029  
0030  
0031  
0032  
0033  
0034  
0035  
0036  
0037  
0038  
0039  
0040  
0041  
0042  
0043  
0044  
0045  
0046  
0047  
0048  
0049  
0050  
0051  
0052  
0053  
0054  
0055  
0056  
0057  
0058  
0059  
0060  
0061  
0062  
0063  
0064  
0065  
0066  
0067  
0068  
0069  
0070  
0071  
0072  
0073  
0074  
0075  
0076  
0077  
0078  
0079  
0080  
0081  
0082  
0083  
0084  
0085  
0086  
0087  
0088  
0089  
0090  
0091  
0092  
0093  
0094  
0095  
0096  
0097  
0098  
0099  
0100  
0101  
0102  
0103  
0104  
0105  
0106  
0107  
0108  
0109  
0110  
0111  
0112  
0113  
0114  
0115  
0116  
0117  
0118  
0119  
0120  
0121  
0122  
0123  
0124  
0125  
0126  
0127  
0128  
0129  
0130  
0131  
0132  
0133  
0134  
0135  
0136  
0137  
0138  
0139  
0140  
0141  
0142  
0143  
0144  
0145  
0146  
0147  
0148  
0149  
0150  
0151  
0152  
0153  
0154  
0155  
0156  
0157  
0158  
0159  
0160  
0161  
0162  
0163  
0164  
0165  
0166  
0167  
0168  
0169  
0170  
0171  
0172  
0173  
0174  
0175  
0176  
0177  
0178  
0179  
0180  
0181  
0182  
0183  
0184  
0185  
0186  
0187  
0188  
0189  
0190  
0191  
0192  
0193  
0194  
0195  
0196  
0197  
0198  
0199  
0200  
0201  
0202  
0203  
0204  
0205  
0206  
0207  
0208  
0209  
0210  
0211  
0212  
0213  
0214  
0215  
0216  
0217  
0218  
0219  
0220  
0221  
0222  
0223  
0224  
0225  
0226  
0227  
0228  
0229  
0230  
0231  
0232  
0233  
0234  
0235  
0236  
0237  
0238  
0239  
0240  
0241  
0242  
0243  
0244  
0245  
0246  
0247  
0248  
0249  
0250  
0251  
0252  
0253  
0254  
0255  
0256  
0257  
0258  
0259  
0260  
0261  
0262  
0263  
0264  
0265  
0266  
0267  
0268  
0269  
0270  
0271  
0272  
0273  
0274  
0275  
0276  
0277  
0278  
0279  
0280  
0281  
0282  
0283  
0284  
0285  
0286  
0287  
0288  
0289  
0290  
0291  
0292  
0293  
0294  
0295  
0296  
0297  
0298  
0299  
0300  
0301  
0302  
0303  
0304  
0305  
0306  
0307  
0308  
0309  
0310  
0311  
0312  
0313  
0314  
0315  
0316  
0317  
0318  
0319  
0320  
0321  
0322  
0323  
0324  
0325  
0326  
0327  
0328  
0329  
0330  
0331  
0332  
0333  
0334  
0335  
0336  
0337  
0338  
0339  
0340  
0341  
0342  
0343  
0344  
0345  
0346  
0347  
0348  
0349  
0350  
0351  
0352  
0353  
0354  
0355  
0356  
0357  
0358  
0359  
0360  
0361  
0362  
0363  
0364  
0365  
0366  
0367  
0368  
0369  
0370  
0371  
0372  
0373  
0374  
0375  
0376  
0377  
0378  
0379  
0380  
0381  
0382  
0383  
0384  
0385  
0386  
0387  
0388  
0389  
0390  
0391  
0392  
0393  
0394  
0395  
0396  
0397  
0398  
0399  
0400  
0401  
0402  
0403  
0404  
0405  
0406  
0407  
0408  
0409  
0410  
0411  
0412  
0413  
0414  
0415  
0416  
0417  
0418  
0419  
0420  
0421  
0422  
0423  
0424  
0425  
0426  
0427  
0428  
0429  
0430  
0431  
0432  
0433  
0434  
0435  
0436  
0437  
0438  
0439  
0440  
0441  
0442  
0443  
0444  
0445  
0446  
0447  
0448  
0449  
0450  
0451  
0452  
0453  
0454  
0455  
0456  
0457  
0458  
0459  
0460  
0461  
0462  
0463  
0464  
0465  
0466  
0467  
0468  
0469  
0470  
0471  
0472  
0473  
0474  
0475  
0476  
0477  
0478  
0479  
0480  
0481  
0482  
0483  
0484  
0485  
0486  
0487  
0488  
0489  
0490  
0491  
0492  
0493  
0494  
0495  
0496  
0497  
0498  
0499  
0500  
0501  
0502  
0503  
0504  
0505  
0506  
0507  
0508  
0509  
0510  
0511  
0512  
0513  
0514  
0515  
0516  
0517  
0518  
0519  
0520  
0521  
0522  
0523  
0524  
0525  
0526  
0527  
0528  
0529  
0530  
0531  
0532  
0533  
0534  
0535  
0536  
0537  
0538  
0539  
0540  
0541  
0542  
0543  
0544  
0545  
0546  
0547  
0548  
0549  
0550  
0551  
0552  
0553  
0554  
0555  
0556  
0557  
0558  
0559  
0560  
0561  
0562  
0563  
0564  
0565  
0566  
0567  
0568  
0569  
0570  
0571  
0572  
0573  
0574  
0575  
0576  
0577  
0578  
0579  
0580  
0581  
0582  
0583  
0584  
0585  
0586  
0587  
0588  
0589  
0590  
0591  
0592  
0593  
0594  
0595  
0596  
0597  
0598  
0599  
0600  
0601  
0602  
0603  
0604  
0605  
0606  
0607  
0608  
0609  
0610  
0611  
0612  
0613  
0614  
0615  
0616  
0617  
0618  
0619  
0620  
0621  
0622  
0623  
0624  
0625  
0626  
0627  
0628  
0629  
0630  
0631  
0632  
0633  
0634  
0635  
0636  
0637  
0638  
0639  
0640  
0641  
0642  
0643  
0644  
0645  
0646  
0647  
0648  
0649  
0650  
0651  
0652  
0653  
0654  
0655  
0656  
0657  
0658  
0659  
0660  
0661  
0662  
0663  
0664  
0665  
0666  
0667  
0668  
0669  
0670  
0671  
0672  
0673  
0674  
0675  
0676  
0677  
0678  
0679  
0680  
0681  
0682  
0683  
0684  
0685  
0686  
0687  
0688  
0689  
0690  
0691  
0692  
0693  
0694  
0695  
0696  
0697  
0698  
0699  
0700  
0701  
0702  
0703  
0704  
0705  
0706  
0707  
0708  
0709  
0710  
0711  
0712  
0713  
0714  
0715  
0716  
0717  
0718  
0719  
0720  
0721  
0722  
0723  
0724  
0725  
0726  
0727  
0728  
0729  
0730  
0731  
0732  
0733  
0734  
0735  
0736  
0737  
0738  
0739  
0740  
0741  
0742  
0743  
0744  
0745  
0746  
0747  
0748  
0749  
0750  
0751  
0752  
0753  
0754  
0755  
0756  
0757  
0758  
0759  
0760  
0761  
0762  
0763  
0764  
0765  
0766  
0767  
0768  
0769  
0770  
0771  
0772  
0773  
0774  
0775  
0776  
0777  
0778  
0779  
0780  
0781  
0782  
0783  
0784  
0785  
0786  
0787  
0788  
0789  
0790  
0791  
0792  
0793  
0794  
0795  
0796  
0797  
0798  
0799  
0800  
0801  
0802  
0803  
0804  
0805  
0806  
0807  
0808  
0809  
0810  
0811  
0812  
0813  
0814  
0815  
0816  
0817  
0818  
0819  
0820  
0821  
0822  
0823  
0824  
0825  
0826  
0827  
0828  
0829  
0830  
0831  
0832  
0833  
0834  
0835  
0836  
0837  
0838  
0839  
0840  
0841  
0842  
0843  
0844  
0845  
0846  
0847  
0848  
0849  
0850  
0851  
0852  
0853  
0854  
0855  
0856  
0857  
0858  
0859  
0860  
0861  
0862  
0863  
0864  
0865  
0866  
0867  
0868  
0869  
0870  
0871  
0872  
0873  
0874  
0875  
0876  
0877  
0878  
0879  
0880  
0881  
0882  
0883  
0884  
0885  
0886  
0887  
0888  
0889  
0890  
0891  
0892  
0893  
0894  
0895  
0896  
0897  
0898  
0899  
0900  
0901  
0902  
0903  
0904  
0905  
0906  
0907  
0908  
0909  
0910  
0911  
0912  
0913  
0914  
0915  
0916  
0917  
0918  
0919  
0920  
0921  
0922  
0923  
0924  
0925  
0926  
0927  
0928  
0929  
0930  
0931  
0932  
0933  
0934  
0935  
0936  
0937  
0938  
0939  
0940  
0941  
0942  
0943  
0944  
0945  
0946  
0947  
0948  
0949  
0950  
0951  
0952  
0953  
0954  
0955  
0956  
0957  
0958  
0959  
0960  
0961  
0962  
0963  
0964  
0965  
0966  
0967  
0968  
0969  
0970  
0971  
0972  
0973  
0974  
0975  
0976  
0977  
0978  
0979  
0980  
0981  
0982  
0983  
0984  
0985  
0986  
0987  
0988  
0989  
0990  
0991  
0992  
0993  
0994  
0995  
0996  
0997  
0998  
0999  
1000

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	434	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	82	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	280	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3 EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
Total Space Allocated		1308

ENTRY POINTS

Address	Type	Name
0-00000000		PADRIVER_ATTENTION780

VARIABLES

Address	Type	Name	Address	Type	Name
2-00000004	I*4	COMPRESS4	3-00000076	I*4	CORRECT_CONTROL_STORE_VALUE
2-00000000	L*1	DIAGNOSTIC_MODE	3-0000001C	L*1	EMBSB_DV_CLASS
3-00000010	L*1	EMBSB_DV_ERTCNT	3-00000011	L*1	EMBSB_DV_ERTMAX
3-0000003E	L*1	EMBSB_DV_NAMLNG	3-0000003A	L*1	EMBSB_DV_SLAVE
3-0000001D	L*1	EMBSB_DV_TYPE	3-00000036	I*4	EMBSL_DV_CHAR
3-00000012	I*4	EMBSL_DV_IOSB1	3-00000016	I*4	EMBSL_DV_IOSB2
3-00000026	I*4	EMBSL_DV_MEDIA	3-0000004E	I*4	EMBSL_DV_NUMREG
3-0000002E	I*4	EMBSL_DV_OPCNT	3-00000032	I*4	EMBSL_DV_OWNUIC
3-0000001E	I*4	EMBSL_DV_RQPID	3-00000000	I*4	EMBSL_HD_SID
3-0000003F	CHAR	EMBST_DV_NAME	3-00000024	I*2	EMBSL_DV_BCNT
3-00000022	I*2	EMBSW_DV_BOFF	3-0000002C	I*2	EMBSW_DV_ERRCNT
3-0000003C	I*2	EMBSW_DV_FUNC	3-0000001A	I*2	EMBSW_DV_STS
3-0000002A	I*2	EMBSW_DV_UNIT	3-00000004	I*2	EMBSW_HD_ENTRY
3-0000000E	I*2	EMBSW_HD_ERRSEQ	AP-00000004a	L*1	LUN
3-00000052	I*4	PADRIVER_ERROR_TYPE_CODE	3-00000056	I*4	PCNFGR
3-00000066	I*4	PESR	3-00000062	I*4	PFAR
3-0000006E	I*4	PMADR	3-0000005A	I*4	PMCSR
3-00000072	I*4	PMDATR	3-0000006A	I*4	PPR
3-0000005E	I*4	PSR			

ARRAYS

Address	Type	Name	Bytes	Dimensions
3-00000000	L*1	EMB	512	(0:511)
3-00000052	I*4	EMBSL_DV_REGSAV	420	(0:104)
3-00000006	I*4	EMBSQ_HD_TIME	8	(2)

PA

PR

EN

VAI

A

LAI

FUI

LABELS

Address	Label	Address	Label	Address	Label	Address	Label
1-00000029	5'	1-0000003C	10'	1-0000004D	15'	0-000001B1	75

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
	CI780 REGA		CI_CONTROL_STORE_MISMATCH		CI_PESR
	CI_PMCSR		CI_PPR		CI_PSR
	FRCTOF		HEADER	I*4	LIB\$EXTZV
	LINCHK		LOGGER		PADRIVER_ATTENTION_ERROR_CODE
	PADRIVER_INITIALIZATION		UCBSB_ERTCNT		UCBSB_ERTMAX
	UCBSL_CHAR		UCBSW_ERRCNT		UCBSW_STS

0001  
 0002  
 0003  
 0004  
 0005  
 0006  
 0007  
 0008  
 0009  
 0010  
 0011  
 0012  
 0013  
 0014  
 0015  
 0016  
 0017  
 0018  
 0019  
 0020  
 0021  
 0022  
 0023  
 0024  
 0025  
 0026  
 0027  
 0028  
 0029  
 0030  
 0031  
 0032  
 0033  
 0034  
 0035  
 0036  
 0037  
 0038  
 0039  
 0040  
 0041  
 0042  
 0043  
 0044  
 0045  
 0046  
 0047  
 0048  
 0049  
 0050  
 0051  
 0052  
 0053  
 0054  
 0055  
 0056  
 0057  
 0058  
 0059  
 0060  
 0061  
 0062  
 0063  
 0064  
 0065  
 0066  
 0067  
 0068  
 0069  
 0070  
 0071  
 0072  
 0073  
 0074  
 0075  
 0076  
 0077  
 0078  
 0079  
 0080  
 0081  
 0082  
 0083  
 0084  
 0085  
 0086  
 0087  
 0088  
 0089  
 0090  
 0091  
 0092  
 0093  
 0094  
 0095  
 0096  
 0097  
 0098  
 0099  
 0100

0001  
0002  
0003  
0004  
0005  
0006  
0065  
0166  
0167  
0168  
0169  
0170  
0171  
0172  
0173  
0174  
0175  
0176  
0177  
0178  
0179  
0180  
0181  
0182  
0183  
0184  
0185  
0186  
0187  
0188  
0189  
0190  
0191  
0192  
0193  
0194  
0195  
0196  
0197  
0198  
0199  
0200  
0201  
0202  
0203  
0204  
0205  
0206  
0207  
0208  
0209  
0210  
0211  
0212  
0213  
0214  
0215

Subroutine PADRIVER\_ATTENTION750 (lun)

include 'src\$:msghdr.for /nolist'  
include 'src\$:deverr.for /nolist'

byte lun

integer\*4 padriver\_error\_type\_code  
integer\*4 pcnfgr  
integer\*4 pmcsr  
integer\*4 psr  
integer\*4 pfar  
integer\*4 pesr  
integer\*4 ppr  
integer\*4 pmadr  
integer\*4 pmdatr  
integer\*4 correct\_control\_store\_value  
integer\*4 compress4

logical\*1 diagnostic\_mode

equivalence (emb\$l\_dv\_regsav(0),padriver\_error\_type\_code)  
equivalence (emb\$l\_dv\_regsav(1),pcnfgr)  
equivalence (emb\$l\_dv\_regsav(2),pmcsr)  
equivalence (emb\$l\_dv\_regsav(3),psr)  
equivalence (emb\$l\_dv\_regsav(4),pfar)  
equivalence (emb\$l\_dv\_regsav(5),pesr)  
equivalence (emb\$l\_dv\_regsav(6),ppr)  
equivalence (emb\$l\_dv\_regsav(7),pmadr)  
equivalence (emb\$l\_dv\_regsav(8),pmdatr)  
equivalence (emb\$l\_dv\_regsav(9),correct\_control\_store\_value)

call frctof (lun)

call header (lun)

call logger (lun,'DEVICE ATTENTION')

call padriver\_attention\_error\_code (lun,padriver\_error\_type\_code)

call padriver\_initialization (lun,padriver\_error\_type\_code)

if (lib\$extzv(8,7,padriver\_error\_type\_code) .eq. 0) goto 20

c  
c  
c  
set not diagnostic\_mode for now

diagnostic\_mode = .false.

If (LIB\$EXTZV(14,1,pcnfgr) .EQ. 1) then

0001  
0002  
0003  
0004  
0005  
0006  
0065  
0166  
0167  
0168  
0169  
0170  
0171  
0172  
0173  
0174  
0175  
0176  
0177  
0178  
0179  
0180  
0181  
0182  
0183  
0184  
0185  
0186  
0187  
0188  
0189  
0190  
0191  
0192  
0193  
0194  
0195  
0196  
0197  
0198  
0199  
0200  
0201  
0202  
0203  
0204  
0205  
0206  
0207  
0208  
0209  
0210  
0211  
0212  
0213  
0214  
0215





PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	454	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	106	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	296	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3 EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
Total Space Allocated		1368

ENTRY POINTS

Address	Type	Name
0-00000000		PADRIVER_ATTENTION750

VARIABLES

Address	Type	Name	Address	Type	Name
2-00000004	I*4	COMPRESS4	3-00000076	I*4	CORRECT_CONTROL_STORE_VALUE
2-00000000	L*1	DIAGNOSTIC_MODE	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_NAMLANG	3-0000003A	L*1	EMB\$B_DV_SLAVE
3-0000001D	L*1	EMB\$B_DV_TYPE	3-00000036	I*4	EMB\$B_DV_CHAR
3-00000012	I*4	EMB\$B_DV_IOSB1	3-00000016	I*4	EMB\$B_DV_IOSB2
3-00000026	I*4	EMB\$B_DV_MEDIA	3-0000004E	I*4	EMB\$B_DV_NUMREG
3-0000002E	I*4	EMB\$B_DV_OPCNT	3-00000032	I*4	EMB\$B_DV_OWNUIC
3-0000001E	I*4	EMB\$B_DV_RQPID	3-00000000	I*4	EMB\$B_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	AP-00000004	L*1	LUN
3-00000052	I*4	PADRIVER_ERROR_TYPE_CODE	3-00000056	I*4	PCNFGR
3-00000066	I*4	PESR	3-00000062	I*4	PFAR
3-0000006E	I*4	PMADR	3-0000005A	I*4	PMCSR
3-00000072	I*4	PMDATR	3-0000006A	I*4	PPR
3-0000005E	I*4	PSR			

ARRAYS

Address	Type	Name	Bytes	Dimensions
3-00000000	L*1	EMB	512	(0:511)
3-00000052	I*4	EMB\$B_DV_REGSAV	420	(0:104)
3-00000006	I*4	EMB\$B_HD_TIME	8	(2)



M 15  
16-Sep-1984 00:11:24  
5-Sep-1984 14:10:51

VAX-11 FORTRAN V3.4-56 Page 10  
DISK\$VMSMASTER:[ERF.SRC]PADRIVER.FOR;1

0001  
0002

CI

PR

EN

VA

AI  
AI

AR

LA

FU

0003  
0004  
0005  
0064  
0165  
0166  
0167  
0168  
0169  
0170  
0171  
0172  
0173  
0174  
0175  
0176  
0177  
0178  
0179  
0180  
0181  
0182  
0183  
0184  
0185  
0186  
0187  
0188  
0189  
0190  
0191  
0192  
0193  
0194  
0195  
0196  
0197  
0198  
0199  
0200  
0201  
0202  
0203  
0204  
0205  
0206  
0207  
0208  
0209  
0210  
0211  
0212  
0213  
0214  
0215  
0216  
0217

Subroutine PADRIVER\_ATTENTION\_ERROR\_CODE (lun,padriver\_error\_type\_code)

include 'src\$:msghdr.for /nolist'  
include 'src\$:deverr.for /nolist'

byte lun  
  
integer\*4 padriver\_error\_type\_code  
integer\*4 error\_type  
integer\*4 error\_subtype  
integer\*4 compress4, Length  
  
Character\*(80) Message  
Character\*(\*) Msg\_free, Gram\_free, Hi, Lo, Prio\_cmd,  
1 Q\_ins\_fail, Q\_rem\_fail, Resp,  
1 Msg1, Msg2, Msg3, Msg4, Msg5,  
1 Msg6, Msg7, Msg8, Msg9, Msg10,  
1 Msg11,Msg12,Msg13

Parameter (  
1 Msg\_free = 'MESSAGE FREE ',  
2 Gram\_free = 'DATAGRAM FREE ',  
3 Hi = 'HIGH ',  
4 Lo = 'LOW ',  
5 Prio\_cmd = 'PRIORITY COMMAND ',  
6 Q\_ins\_fail = 'QUEUE INSERT FAILURE',  
7 Q\_rem\_fail = 'QUEUE REMOVE FAILURE',  
8 Resp = 'RESPONSE ',  
9 Msg1 = 'INSUFFICIENT NON-PAGED POOL FOR INITIALIZATION',  
1 Msg2 = 'FAILED TO LOCATE PORT MICRO-CODE IMAGE',  
2 Msg3 = 'MICRO-CODE VERIFICATION ERROR',  
3 Msg4 = 'NO TRANSITION FROM 'UNINITIALIZED' TO 'DISABLED'',  
4 Msg5 = 'PORT ERROR BIT(S) SET',  
5 Msg6 = 'PORT POWER DOWN',  
6 Msg7 = 'PORT POWER UP',  
7 Msg8 = 'UNEXPECTED INTERRUPT',  
8 Msg9 = 'SCSSYSTEMID MUST BE SET TO A NON-ZERO VALUE.',  
9 Msg10 = 'CI PORT MICROCODE REV NOT ',  
1 Msg11 = 'SUPPORTED',  
2 Msg12 = 'CURRENT, BUT SUPPORTED',  
3 Msg13 = '11/750 CPU MICROCODE NOT ADEQUATE FOR CI')

Error\_subtype = lib\$extzv(0,8,padriver\_error\_type\_code)  
Error\_type = lib\$extzv(8,7,padriver\_error\_type\_code)

Call linchk (lun,2)

Goto ( 100, 200 ) error\_type

If (error\_type .eq. 0) then  
If (error\_subtype .eq. 0) then  
Message = msg1  
Length = len (msg1)  
Goto 990  
Else if (error\_subtype .eq. 1) then  
Message = msg2  
Length = len (msg2)

B  
C  
D  
E  
F  
G  
H  
I  
J  
K  
L  
M  
N  
O  
P  
Q  
R  
S  
T  
U  
V  
W  
X  
Y  
Z

```

0218           Goto 990
0219       Else if (error_subtype .eq. 2) then
0220           Message = msg9
0221           Goto 990
0222       Endif
0223
0224       Else
0225           Write(lun,995) emb$b_dv_name(1:emb$b_dv_namlng),emb$w_dv_unit,
0226           1 ''PADRIVER'' ERROR TYPE-#,error_type,',', ERROR SUB-TYPE-#,
0227           1 error_subtype,',',
0228 995       Format(/' 'C) SUB-SYSTEM, ',a,
0229           1 i<compress4 (lib$extzv(0,16,emb$w_dv_unit))>,': - ',
0230           1 a,i<compress4 (error_type)>,>,a,i<compress4 (error_subtype)>,>,a)
0231       Endif
0232
0233       Return
0234
0235 100       Goto ( 5, 10, 15, 20, 25, 30, 35, 40 ) error_subtype
0236
0237       If (error_subtype .eq. 0) then
0238           Message = msg3
0239           Length = len (msg3)
0240           Goto 990
0241       Endif
0242       Return
0243
0244 5         Message = msg4
0245           Length = len (msg4)
0246           Goto 990
0247
0248 10        Message = msg5
0249           Length = len (msg5)
0250           Goto 990
0251
0252 15        Message = msg6
0253           Length = len (msg6)
0254           Goto 990
0255
0256 20        Message = msg7
0257           Length = len (msg7)
0258           Goto 990
0259
0260 25        Message = msg8
0261           Length = len (msg8)
0262           Goto 990
0263
0264 30        Message = msg10 // msg11
0265           Goto 990
0266
0267 35        Message = msg13
0268           Goto 990
0269
0270 40        Message = msg10 // msg12
0271           Goto 990
0272
0273 200       Goto ( 210,220,230,240,250,260 ) error_subtype
0274

```

```
0275
0276      If (error_subtype .eq. 0) then
0277          Message = msg_free // q_rem_fail
0278          Length = len (msg_free) + len (q_rem_fail)
0279          Goto 990          ! Go Write
0280      Endif
0281      Return
0282
0283 210      Message = gram_free // q_rem_fail
0284          Length = len (gram_free) + len (q_rem_fail)
0285          Goto 990
0286
0287 220      Message = resp // q_rem_fail
0288          Length = len (resp) + len (q_rem_fail)
0289          Goto 990
0290
0291 230      Message = hi // prio_cmd // q_ins_fail
0292          Length = len (hi) + len (prio_cmd) + len (q_ins_fail)
0293          Goto 990
0294
0295 240      Message = lo // prio_cmd // q_ins_fail
0296          Length = len (lo) + len (prio_cmd) + len (q_ins_fail)
0297          Goto 990
0298
0299 250      Message = msg_free // q_ins_fail
0300          Length = len (msg_free) + len (q_ins_fail)
0301          Goto 990
0302
0303 260      Message = gram_free // q_ins_fail
0304          Length = len (gram_free) + len (q_ins_fail)
0305
0306 990      write(lun,991) emb$b_dv_name(1:emb$b_dv_namln),
0307          1 emb$w_dv_unit, Message
0308
0309 991      format(/' ', 'CI SUB-SYSTEM, ', a,
0310          1 i<compress4 (lib$extzv(0,16,emb$w_dv_unit))>, ': - ', a,
0311          1 : i<compress4 (error_subtype)>, : a)
0312
0313      Return
0314      End
```





PADRIVER\_ATTENTION\_ERROR\_CODE

E 16  
16-Sep-1984 00:11:24  
5-Sep-1984 14:10:51

VAX-11 FC:TRAN V3.4-56 Page 15  
DISK\$VMSMASTER:[ERF.SRC]PADRIVER.FOR;1

FUNCTIONS AND SUBROUTINES REFERENCED

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

```
0001 Subroutine PADRIVER_INITIALIZATION (lun,padriver_error_type_code)
0002
0003
0004 byte lun
0005
0006 integer*4 padriver_error_type_code
0007 integer*4 initialization_retry_count
0008 integer*4 initialization_maxtry_count
0009 integer*4 compress4
0010
0011 logical*1 port_reinitialization
0012
0013
0014 port_reinitialization = .false.
0015
0016 if (lib$extzv(15,1,padriver_error_type_code) .eq. 1)
0017 1 port_reinitialization = .true.
0018
0019 initialization_retry_count = lib$extzv(16,8,padriver_error_type_code)
0020 initialization_maxtry_count = lib$extzv(24,8,padriver_error_type_code)
0021
0022 if (port_reinitialization) then
0023
0024 call linchk (lun,2)
0025
0026 if (initialization_retry_count .gt. 0) then
0027
0028 write(lun,10) 'PORT WILL BE RESTARTED, ',
0029 1 initialization_retry_count, '. OF ',initialization_maxtry_count,
0030 1 '. RETRIES REMAINING'
0031 10 format(/' ',t8,a,i<compress4 (initialization_retry_count)>,a,
0032 1 i<compress4 (initialization_maxtry_count)>,a)
0033 else
0034
0035 write(lun,15) '0. RETRIES REMAINING, PORT WILL BE DISABLED'
0036 15 format(/' ',t8,a)
0037
0038 endif
0039
0040 endif
0041
0042 return
0043
0044 end
```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	226	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	145	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	120	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated		491

ENTRY POINTS

Address	Type	Name
0-00000000		PADRIVER_INITIALIZATION

VARIABLES

Address	Type	Name	Address	Type	Name
2-00000008	I*4	INITIALIZATION_MAXTRY_COUNT	2-00000004	I*4	INITIALIZATION_RETRY_COUNT
AP-00000004@	I*1	LUN	AP-00000008@	I*4	PADRIVER_ERROR_TYPE_CODE
2-00000000	L*1	PORT_REINITIALIZATION			

LABELS

Address	Label	Address	Label
1-00000073	10'	1-00000089	15'

FUNCTIONS AND SUBROUTINES REFERENCED

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

```
0001  
0002  
0003  
0004  
0005 Subroutine C1_PESR (lun,pesr,psr,diagnostic_mode)  
0006  
0007 byte lun  
0008  
0009 integer*4 pesr  
0010 integer*4 psr  
0011 integer*4 compress4  
0012 integer*4 pesr_value  
0013  
0014 logical*1 diagnostic_mode  
0015  
0016  
0017 call linchk (lun,1)  
0018  
0019 write(lun,25) pesr  
25 format(' ',t8,'PESR',t24,z8.8)  
0020  
0021 if (.not. diagnostic_mode) then  
0022  
0023 if (lib$extzv(4,1,psr) .eq. 1) then  
0024  
0025 Pesr_value = LIB$EXTZV(0,20,psr)  
0026  
0027 If (pesr_value .NE. 0) then  
0028 Call LINCHK (lun,1)  
0029 Endif  
0030  
0031 IF (pesr_value .EQ. 1) then  
0032  
0033 write(lun,30) 'ILLEGAL SYSTEM VIRT ADDR FORMAT'  
30 format(' ',t40,a,:i<compress4 (pesr_value)>, :a)  
0034  
0035  
0036 else if (pesr_value .eq. 2) then  
0037 write(lun,30) 'NON-EXISTENT SYSTEM VIRTUAL ADDR'  
0038  
0039  
0040 else if (pesr_value .eq. 3) then  
0041 write(lun,30) 'INVALID SYSTEM "PTE"'  
0042  
0043  
0044 else if (pesr_value .eq. 4) then  
0045 write(lun,30) 'INVALID BUFFER "PTE"'  
0046  
0047  
0048 else if (pesr_value .eq. 5) then  
0049 write(lun,30) 'NON-EXISTENT SYSTEM GBL VIRT ADDR'  
0050  
0051  
0052 else if (pesr_value .eq. 6) then  
0053 write(lun,30) 'NON-EXISTENT BUFFER GBL VIRT ADDR'  
0054  
0055  
0056 else if (pesr_value .eq. 7) then  
0057 write(lun,30) 'INVALID SYSTEM GLOBAL "PTE"'
```

```
0058     else if (pesr_value .eq. 9) then
0059     write(lun,30) 'INVALID SYSTEM GBL "PTE" MAPPING'
0060
0061     else if (pesr_value .eq. 10) then
0062     write(lun,30) 'INVALID BUFFER GBL "PTE" MAPPING'
0063
0064     else if (pesr_value .eq. 11) then
0065     write(lun,30) 'QUEUE INTERLOCK RETRY FAILURE'
0066
0067     else if (pesr_value .eq. 12) then
0068     write(lun,30) 'ILLEGAL QUEUE OFFSET ALIGNMENT'
0069
0070     else if (pesr_value .eq. 13) then
0071     write(lun,30) 'ILLEGAL "PQB" FORMAT'
0072
0073     else if (pesr_value .eq. 14) then
0074     write(lun,30) 'REGISTER PROTOCOL VIOLATION'
0075     else
0076
0077     write(lun,30) 'ERROR STATUS CODE #',pesr_value,'.'
0078     endif
0079     endif
0080
0081     If (LIB$EXTZV(7,1,psr) .EQ. 1) then
0082
0083     Pesr_value = LIB$EXTZV(16,5,psr)
0084
0085     If (pesr_value .NE. 0) then
0086     Call LINCHK (lun,1)
0087     Endif
0088
0089     If (pesr_value .EQ. 1) then
0090
0091     write(lun,30) 'RECEIVE BUFFERS EMPTY, FLAG SET'
0092
0093     else if (pesr_value .eq. 2) then
0094
0095     write(lun,30) 'INTERNAL PACKET IN ILLEGAL STATE'
0096
0097     else if (pesr_value .eq. 3) then
0098
0099     write(lun,30) 'PORT STATUS, ENABLED AND DISABLED'
0100
0101     else if (pesr_value .eq. 4) then
0102
0103     write(lun,30) 'COMMAND, COMPLETE AND INCOMPLETE'
0104
0105     else if (pesr_value .eq. 5) then
0106
0107     write(lun,30) 'INTERNAL QUEUE RETRY EXPIRED'
0108
0109     else if (pesr_value .eq. 6) then
0110
0111     write(lun,30) 'INTERNAL TRANSMIT, NO PATH'
0112
0113     else if (pesr_value .eq. 7) then
0114
```

CI\_PESR

J 16  
16-Sep-1984 00:11:24  
5-Sep-1984 14:10:51

VAX-11 FORTRAN V3.4-56 Page 20  
DISK\$VMSMASTER:[ERF.SRC]PADRIVER.FOR;1

```

0115 write(lun,30) 'RECEIVE PACKET, ACK AND NACK'
0116
0117 else if (pesr_value .eq. 8) then
0118
0119 write(lun,30) 'PATH FAILURE, BOTH AVAILABLE'
0120
0121 else if (pesr_value .eq. 9) then
0122
0123 write(lun,30) 'UNKNOWN MAINTENANCE OPCODE'
0124
0125 else if (pesr_value .eq. 10) then
0126
0127 write(lun,30) 'BOTH PATHS BEING FORCED'
0128
0129 else if (pesr_value .eq. 11) then
0130
0131 write(lun,30) 'ILLEGAL CSB STATE'
0132 else
0133
0134 write(lun,30) 'ERROR STATUS CODE #',pesr_value,','
0135 endif
0136 endif
0137 endif
0138
0139 return
0140
0141 End

```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	1357	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	778	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	304	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	2439	

ENTRY POINTS

Address	Type	Name
0-00000000		CI_PESR

VARIABLES

Address	Type	Name	Address	Type	Name
AP-00000010a	L*1	DIAGNOSTIC_MODE	AP-00000004a	L*1	LUN
AP-00000008a	I*4	PESR	2-000000C0	I*4	PESR_VALUE
AP-0000000Ca	I*4	PSR			

CI\_PESR

K 16  
16-Sep-1984 00:11:24  
5-Sep-1984 14:10:51

VAX-11 FORTRAN V3.4-56 Page 21  
DISK\$VMSMASTER:[ERF.SRC]PADRIVER.FOR;1

LABELS

Address	Label	Address	Label
1-000002E9	25'	1-000002FA	30'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
I*4	COMPRESS4	I*4	LIBSEXTZV		LINCHK

0001  
0002  
0003  
0004  
0005  
0006  
0007  
0008  
0009  
0010  
0011  
0012  
0013  
0014  
0015  
0016  
0017  
0018  
0019  
0020  
0021  
0022  
0023  
0024  
0025  
0026  
0027  
0028  
0029  
0030  
0031  
0032  
0033  
0034  
0035  
0036  
0037  
0038  
0039  
0040  
0041  
0042  
0043  
0044  
0045  
0046  
0047

Subroutine CI\_PMCSR (lun,pmcsr,diagnostic\_mode)

byte lun

integer\*4 pmcsr

logical\*1 diagnostic\_mode

character\*29 v1pmcsr(0:4)

data v1pmcsr(0) /\*MAINTENANCE INITIALIZE\*/

data v1pmcsr(1) /\*MAINTENANCE TIMER DISABLE\*/

data v1pmcsr(2) /\*MAINTENANCE INTERRUPT ENABLE\*/

data v1pmcsr(3) /\*MAINTENANCE INTERRUPT FLAG\*/

data v1pmcsr(4) /\*WRONG PARITY\*/

character\*30 v2pmcsr(6:15)

data v2pmcsr(6) /\*PROGRAMMABLE STARTING ADDRESS\*/

data v2pmcsr(7) /\*UNINITIALIZED STATE\*/

data v2pmcsr(8) /\*TRANSMIT BUFFER PARITY ERROR\*/

data v2pmcsr(9) /\*OUTPUT PARITY ERROR\*/

data v2pmcsr(10) /\*INPUT PARITY ERROR\*/

data v2pmcsr(11) /\*TRANSMIT BUFFER PARITY ERROR\*/

data v2pmcsr(12) /\*RECEIVE BUFFER PARITY ERROR\*/

data v2pmcsr(13) /\*LOCAL STORE PARITY ERROR\*/

data v2pmcsr(14) /\*CONTROL STORE PARITY ERROR\*/

data v2pmcsr(15) /\*PARITY ERROR\*/

call linchk (lun,1)

write(lun,5) pmcsr

format(' ',t8,'PMCSR',t24,z8.8)

if (.not. diagnostic\_mode) then

call output (lun,pmcsr,v1pmcsr,0,0,4,'0')

call output (lun,pmcsr,v2pmcsr,6,6,15,'0')

endif

return

End



PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	98	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	40	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	596	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated		734

ENTRY POINTS

Address	Type	Name
0-00000000		CI_PMCSR

VARIABLES

Address	Type	Name	Address	Type	Name
AP-0000000C@	L*1	DIAGNOSTIC_MODE	AP-00000004@	L*1	LUN
AP-00000008@	I*4	PMCSR			

ARRAYS

Address	Type	Name	Bytes	Dimensions
2-00000000	CHAR	V1PMCSR	145	(0:4)
2-00000091	CHAR	V2PMCSR	300	(6:15)

LABELS

Address	Label
1-00000016	5'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name
	LINCHK		OUTPUT

0001  
0002  
0003  
0004  
0005  
0006  
0007  
0008  
0009  
0010  
0011  
0012  
0013  
0014  
0015  
0016  
0017  
0018  
0019  
0020  
0021  
0022  
0023  
0024  
0025  
0026  
0027  
0028  
0029  
0030  
0031  
0032  
0033  
0034  
0035  
0036  
0037  
0038  
0039  
0040  
0041

Subroutine (I\_PSR (lun,psr,diagnostic\_mode)

byte lun

integer\*4 psr

logical\*1 diagnostic\_mode

character\*29 v1psr(0:7)

data v1psr(0) /\*RESPONSE QUEUE AVAILABLE\*/

data v1psr(1) /\*MESSAGE FREE QUEUE EMPTY\*/

data v1psr(2) /\*PORT DISABLE COMPLETE\*/

data v1psr(3) /\*PORT INITIALIZATION COMPLETE\*/

data v1psr(4) /\*DATA STRUCTURE ERROR\*/

data v1psr(5) /\*MEMORY SYSTEM ERROR\*/

data v1psr(6) /\*MAINTENANCE TIMER EXPIRATION\*/

data v1psr(7) /\*MISCELLANEOUS ERROR DETECTED\*/

character\*18 v2psr(31:31)

data v2psr(31) /\*MAINTENANCE ERROR\*/

call linchk (lun,1)

5 write(lun,5) psr  
format(' ',t8,'PSR',t24,z8.8)

if (.not. diagnostic\_mode) then

call output (lun,psr,v1psr,0,0,7,'0')

call output (lun,psr,v2psr,31,31,31,'0')

endif

return

End

PA  
0000  
0001  
0002  
0003  
0004  
0005  
0006  
0007  
0008  
0009  
0010  
0011  
0012  
0013  
0014  
0015  
0016  
0017  
0018  
0019  
0020  
0021  
0022  
0023  
0024  
0025  
0026  
0027  
0028  
0029  
0030  
0031  
0032  
0033  
0034  
0035  
0036  
0037  
0038  
0039  
0040  
0041  
0042  
0043  
0044  
0045  
0046  
0047  
0048  
0049  
0050  
0051  
0052  
0053  
0054  
0055  
0056  
0057  
0058  
0059  
0060  
0061  
0062  
0063  
0064  
0065  
0066  
0067  
0068  
0069  
0070  
0071  
0072  
0073  
0074  
0075  
0076  
0077  
0078  
0079  
0080  
0081  
0082  
0083  
0084  
0085  
0086  
0087  
0088  
0089  
0090  
0091  
0092  
0093  
0094  
0095  
0096  
0097  
0098  
0099  
0100















```
0002 Subroutine PADRIVER_LOGMESSAGE (lun,option)
0003
0004 include 'src$:msghdr.for /nolist'
0063 include 'src$:emblmdef.for /nolist'
0132
0133
0134 byte lun
0135
0136 character*1 option
0137
0138 integer*4 padriver_error_type_code
0139 integer*4 ucb$l_errcnt
0140 integer*4 remote_station_address031
0141 integer*4 remote_system_id031
0142 integer*4 first_68_bytes_of_message(17)
0143 integer*4 error_subtype
0144 integer*4 error_type
0145 integer*4 path
0146 integer*4 remote_node_number
0147 integer*4 operation_code
0148 integer*4 compress4
0149
0150 logical*1 response
0151
0152 integer*2 local_station_address(3)
0153 integer*2 local_system_id(3)
0154 integer*2 remote_station_address(3)
0155 integer*2 remote_system_id(3)
0156 integer*2 remote_station_address3247, hsc$w_msglen
0157 integer*2 remote_system_id3247, hsc$w_errlog_dg
0158
0159 byte ppd$b_port
0160 byte ppd$b_status
0161 byte ppd$b_opc
0162 byte ppd$b_flags
0163
0164 equivalence (remote_station_address(3),remote_station_address3247)
0165 equivalence (remote_station_address,remote_station_address031)
0166 equivalence (remote_system_id,remote_system_id031)
0167 equivalence (remote_system_id(3),remote_system_id3247)
0168
0169 equivalence (emb$b_lm_msgtxt(1),padriver_error_type_code)
0170 equivalence (emb$b_lm_msgtxt(5),ucb$l_errcnt)
0171 equivalence (emb$b_lm_msgtxt(9),local_station_address)
0172 equivalence (emb$b_lm_msgtxt(15),local_system_id)
0173 equivalence (emb$b_lm_msgtxt(21),remote_station_address)
0174 equivalence (emb$b_lm_msgtxt(27),remote_system_id)
0175 equivalence (emb$b_lm_msgtxt(33),ppd$b_port)
0176 equivalence (emb$b_lm_msgtxt(34),ppd$b_status)
0177 equivalence (emb$b_lm_msgtxt(35),ppd$b_opc)
0178 equivalence (emb$b_lm_msgtxt(36),ppd$b_flags)
0179 equivalence (emb$b_lm_msgtxt(37),first_68_bytes_of_message)
0180 equivalence (emb$b_lm_msgtxt(39),hsc$w_errlog_dg)
0181 equivalence (emb$b_lm_msgtxt(49),hsc$t_nodename)
0182 equivalence (emb$b_lm_msgtxt(57),hsc$w_msglen)
0183 equivalence (emb$b_lm_msgtxt(59),hsc$t_message)
0184
```

PA

PR

EN

VA

A

A

AR

```

0185
0186 character*(200) hsc$t_message
0187 character*(8) hsc$t_nodename
0188 character*(50) Message_string
0189 character*(*) Msg1,msg2,msg3,msg4,msg5,msg6,msg7,msg8,msg9,msg10,
0190 1 msg11,msg12,msg13
0191
0192 Integer*4 str$position, start_index, end_loc
0193 Character*1 sub_str
0194 Data sub_str/13/
0195
0196 parameter (
0197 1 msg1 = 'DATA CABLE(S) CHANGE OF STATE',
0198 2 msg2 = 'PATH #0. HAS GONE FROM GOOD TO BAD',
0199 3 msg3 = 'PATH #1. HAS GONE FROM GOOD TO BAD',
0200 4 msg4 = 'PATH #0. HAS GONE FROM BAD TO GOOD',
0201 5 msg5 = 'PATH #1. HAS GONE FROM BAD TO GOOD',
0202 6 msg6 = 'CABLES HAVE GONE FROM UNCROSSED TO CROSSED',
0203 7 msg7 = 'CABLES HAVE GONE FROM CROSSED TO UNCROSSED',
0204 8 msg8 = 'PATH #0. LOOPBACK HAS GONE FROM GOOD TO BAD',
0205 9 msg9 = 'PATH #1. LOOPBACK HAS GONE FROM GOOD TO BAD',
0206 2 msg10 = 'PATH #0. LOOPBACK HAS BECOME GOOD, UNCROSSED',
0207 2 msg11 = 'PATH #1. LOOPBACK HAS BECOME GOOD, UNCROSSED',
0208 2 msg12 = 'PATH #0. HAS BECOME WORKING BUT CROSSED TO PATH #1.',
0209 2 msg13 = 'PATH #1. HAS BECOME WORKING BUT CROSSED TO PATH #0.')
```

```

0210
0211 if (option .eq. 'S') call frctof (lun)
0212
0213 call header (lun)
0214
0215 call logger (lun,'ERL$LOGMESSAGE ENTRY')
0216
0217 error_subtype = lib$extzv(0,8,padriver_error_type_code)
0218 error_type = lib$extzv(8,7,padriver_error_type_code)
0219
0220 call linchk (lun,2)
0221
0222 if (error_type .eq. 64) then
0223
0224     if (error_subtype .eq. 0) then
0225         write(lun,10) emb$t_lm_name(1:emb$b_lm_namlng),
0226 1 emb$w_lm_unit,'UNRECOGNIZED "SCA" PACKET'
0227 10 format(/' ','CI SUB-SYSTEM, 'a,
0228 1 i<compress4 (lib$extzv(0,16,emb$w_lm_unit))>',' - ',
0229 1 a,:i<compress4 (error_subtype)>,:a)
0230
0231     else if (error_subtype .eq. 1) then
0232         write(lun,10) emb$t_lm_name(1:emb$b_lm_namlng),
0233 1 emb$w_lm_unit,'PORT HAS CLOSED "VIRTUAL CIRCUIT"'
0234
0235     else if (error_subtype .eq. 2) then
0236         write(lun,10) emb$t_lm_name(1:emb$b_lm_namlng),
0237 1 emb$w_lm_unit,'SOFTWARE SHUTTING DOWN PORT'
0238
0239     else if (error_subtype .eq. 3) then
0240         write(lun,10) emb$t_lm_name(1:emb$b_lm_namlng),
0241 1 emb$w_lm_unit,'SOFTWARE IS CLOSING "VIRTUAL CIRCUIT"'

```



```

0299 335      message_string = msg8
0300          go to 990
0301
0302 340      message_string = msg9
0303          go to 990
0304
0305 345      message_string = msg10
0306          go to 990
0307
0308 350      message_string = msg11
0309          go to 990
0310
0311 355      message_string = msg12
0312          go to 990
0313
0314 360      message_string = msg13
0315
0316 990      write(lun,12) message_string
0317 12       format(/' ',t8,a)
0318 992      continue
0319
0320      else
0321
0322          write(lun,25) emb$t_lm_name(1:emb$b_lm_namlng),
0323          emb$b_lm_unit,'PADRIVER' ERROR TYPE #',error_type,
0324          '., ERROR SUB-TYPE #',error_subtype,'.'
0325 25       format(/' ',t8,a,3(z4.4),a)
0326          i<compress4 (lib$extzv(0,16,emb$b_lm_unit))>,': - ',
0327          a,i<compress4 (error_type)>,a,i<compress4 (error_subtype)>,a)
0328      endif
0329
0330      call padriver_initialization (lun,padriver_error_type_code)
0331
0332      if (option .eq. 'B') return
0333
0334      call linchk (lun,2)
0335
0336      write(lun,30) 'LOCAL STATION ADDRESS, ',
0337 30       1 (local_station_address(i),i = 3,1,-1)
0338          format(/' ',t8,a,3(z4.4),a) (HEX)'
0339
0340      call linchk (lun,2)
0341
0342      write(lun,30) 'LOCAL SYSTEM ID, ',(local_system_id(i),i = 3,1,-1)
0343
0344      message = .false.
0345
0346      call linchk (lun,2)
0347
0348      if (remote_station_address031 - 0) 35,40,40
0349 35       if (remote_station_address3247 - 0) 45,40,40
0350
0351 40       write(lun,30) 'REMOTE STATION ADDRESS, ',
0352          1 (remote_station_address(i),i = 3,1,-1)
0353
0354      message = .true.
0355

```

```

0356       goto 55
0357
0358 45      write(lun,50) 'REMOTE STATION ADDRESS UNAVAILABLE'
0359 50      format(/' ',t8,a)
0360
0361 55      continue
0362
0363       call linchk (lun,2)
0364
0365       if (remote_system_id031 - 0) 70,65,70
0366
0367 65      if (remote_system_id3247 - 0) 70,75,70
0368
0369 70      write(lun,30) 'REMOTE SYSTEM ID, ',(remote_system_id(i),i = 3,1,-1)
0370
0371       goto 80
0372
0373 75      write(lun,50) 'REMOTE SYSTEM ID UNAVAILABLE'
0374
0375 80      continue
0376
0377       call linchk (lun,1)
0378
0379       write(lun,85)
0380 85      format(' ',:)
0381
0382       call ucb$b_ertcnt (lun,lib$extzv(16,8,padriver_error_type_code))
0383
0384       call ucb$b_ertmax (lun,lib$extzv(24,8,padriver_error_type_code))
0385
0386       call ucb$w_errcnt (lun,ucb$l_errcnt)
0387
0388       if (.NOT. message) return
0389
0390       call linchk (lun,1)
0391
0392       write(lun,90) ppd$b_port
0393 90      format(' ',t8,'PPD$B_PORT',t30,z2.2)
0394
0395       remote_node_number = lib$extzv(0,8,ppd$b_port)
0396
0397       call linchk (lun,1)
0398
0399       write(lun,95) remote_node_number
0400 95      format(' ',t40,'REMOTE NODE #',i<compress4 (remote_node_number)>,
0401 1 '. ')
0402
0403       call linchk (lun,1)
0404
0405       write(lun,97) ppd$b_status
0406 97      format(' ',t8,'PPD$B_STATUS',t30,z2.2)
0407
0408       response = .false.
0409
0410       if (ppd$b_status .ne. 0) response = .true.
0411
0412       if (lib$extzv(5,1,ppd$b_opc) .eq. 1) response = .true.

```

00  
00  
00  
00  
00  
00  
00  
00  
00  
00  
00  
00  
00  
00  
00  
00  
00  
00  
00  
00  
00

PR

EN

VA

A

```
0413
0414     if (response) then
0415
0416     call status (lun,ppd$b_status)
0417     endif
0418
0419     call linchk (lun,1)
0420
0421     write(lun,99) ppd$b_opc
0422 99     format(' ',t8,'PPD$b_OPC',t30,z2.2)
0423
0424     operation_code = lib$extzv(0,8,ppd$b_opc)
0425
0426     call linchk (lun,1)
0427
0428     if (operation_code .eq. 1) then
0429
0430         if (.not. response) then
0431
0432             write(lun,105) 'SNDDG'
0433 105     format(' ',t40,a)
0434             else
0435
0436             write(lun,105) 'DGSNT'
0437             endif
0438
0439     call flags_pf (lun,ppd$b_flags)
0440
0441     else if (operation_code .eq. 2) then
0442
0443         if (.not. response) then
0444
0445             write(lun,105) 'SNDMSG'
0446             else
0447
0448             write(lun,105) 'MSGSENT'
0449             endif
0450
0451     call flags_pf (lun,ppd$b_flags)
0452
0453     else if (operation_code .eq. 3) then
0454
0455         if (.not. response) then
0456
0457             write(lun,105) 'RETCNF'
0458             else
0459
0460             write(lun,105) 'CNFRET'
0461             endif
0462
0463     call flags (lun,ppd$b_flags)
0464
0465     else if (operation_code .eq. 5) then
0466
0467         if (.not. response) then
0468
0469             write(lun,105) 'REQID'
```



```
0527     if (.not. response) then
0528
0529     write(lun,105) 'REQDAT2'
0530     else
0531
0532     write(lun,105) 'DATREQ2'
0533     endif
0534
0535     call flags_p (lun,ppd$b_flags)
0536
0537     else if (operation_code .eq. 13) then
0538
0539     if (.not. response) then
0540
0541     write(lun,105) 'SNDLB'
0542     else
0543
0544     write(lun,105) 'LBSNT'
0545     endif
0546
0547     call flags_pf (lun,ppd$b_flags)
0548
0549     else if (operation_code .eq. 14) then
0550
0551     if (.not. response) then
0552
0553     write(lun,105) 'REQMDAT'
0554     else
0555
0556     write(lun,105) 'MDATREQ'
0557     endif
0558
0559     call flags_p (lun,ppd$b_flags)
0560
0561     else if (operation_code .eq. 16) then
0562
0563     if (.not. response) then
0564
0565     write(lun,105) 'SNDDAT'
0566     else
0567
0568     write(lun,105) 'DATSNT'
0569     endif
0570
0571     call flags_p (lun,ppd$b_flags)
0572
0573     else if (operation_code .eq. 17) then
0574
0575     if (.not. response) then
0576
0577     write(lun,105) 'RETDAT'
0578     else
0579
0580     write(lun,105) 'DATRET'
0581     endif
0582
0583     call flags_p (lun,ppd$b_flags)
```



```
0584  
0585     else if (operation_code .eq. 18) then  
0586  
0587     if (.not. response) then  
0588  
0589     write(lun,105) 'SNDMDAT'  
0590     else  
0591  
0592     write(lun,105) 'MDATSNT'  
0593     endif  
0594  
0595     call flags_p (lun,ppd$b_flags)  
0596  
0597     else if (operation_code .eq. 24) then  
0598  
0599     if (.not. response) then  
0600  
0601     write(lun,105) 'INVTG'  
0602     else  
0603  
0604     write(lun,105) 'TCINV'  
0605     endif  
0606  
0607     call flags (lun,ppd$b_flags)  
0608  
0609     else if (operation_code .eq. 25) then  
0610  
0611     if (.not. response) then  
0612  
0613     write(lun,105) 'SETCKT'  
0614     else  
0615  
0616     write(lun,105) 'CKTSET'  
0617     endif  
0618  
0619     call flags (lun,ppd$b_flags)  
0620  
0621     else if (operation_code .eq. 26) then  
0622  
0623     if (.not. response) then  
0624  
0625     write(lun,105) 'RDCNT'  
0626     else  
0627  
0628     write(lun,105) 'CNTRD'  
0629     endif  
0630  
0631     call flags (lun,ppd$b_flags)  
0632  
0633     else if (operation_code .eq. 33) then  
0634  
0635     write(lun,105) 'DGREC'  
0636  
0637     call flags_pf (lun,ppd$b_flags)  
0638  
0639     else if (operation_code .eq. 34) then  
0640
```

OC  
OC  
OC  
OC  
OC  
OC  
OC  
OC  
OC  
OC  
OC  
OC  
OC  
OC  
OC  
OC  
OC  
OC  
OC  
OC  
OC  
OC

PR

EN

VA

A

LA

```
0641      write(lun,105) 'MSGREC'
0642
0643      call flags_pf (lun,ppd$b_flags)
0644
0645      else if (operation_code .eq. 35) then
0646
0647      write(lun,105) 'CNFREC'
0648
0649      call linchk (lun,1)
0650
0651      write(lun,111) ppd$b_flags
0652
0653      else if (operation_code .eq. 49) then
0654
0655      write(lun,105) 'DATREC'
0656
0657      call linchk (lun,1)
0658
0659      write(lun,111) ppd$b_flags
0660
0661      else if (operation_code .eq. 45) then
0662
0663      write(lun,105) 'LBREC'
0664
0665      call linchk (lun,1)
0666
0667      write(lun,111) ppd$b_flags
0668
0669      path = lib$extzv(1,2,ppd$b_flags)
0670
0671      path = path - 1
0672
0673      if (path .ge. 0) then
0674
0675      call linchk (lun,1)
0676
0677      write(lun,110) 'LOOPBACK RECEIVED ON PATH #',path
0678 110      format(' ',t40,a,i<compress4 (path)>,'.')
0679      endif
0680
0681      else if (operation_code .eq. 43) then
0682
0683      write(lun,105) 'IDREC'
0684
0685      call linchk (lun,1)
0686
0687      write(lun,111) ppd$b_flags
0688 111      format(' ',t8,'PPD$b_FLAGS',t30,z2.2)
0689
0690      path = lib$extzv(1,2,ppd$b_flags)
0691
0692      path = path - 1
0693
0694      if (path .GE. 0) then
0695
0696      call linchk (lun,1)
0697
```

```
0698       write(lun,110) 'RECEIVE PATH #',path
0699
0700       else
0701
0702       call linchk (lun,1)
0703
0704       write(lun,112) 'RECEIVE'
0705  112   format(' ',t40,a,' PATH, INTERNAL LOOPBACK')
0706       endif
0707
0708       path = lib$extzv(4,2,ppd$b_flags)
0709
0710       path = path - 1
0711
0712       if (path .GE. 0) then
0713
0714       call linchk (lun,1)
0715
0716       write(lun,110) 'SEND PATH #',path
0717
0718       else
0719
0720       call linchk (lun,1)
0721
0722       write(lun,112) 'SEND'
0723       endif
0724
0725       else if (operation_code .eq. 36) then
0726
0727       write(lun,105) 'MCNFREC'
0728
0729       call linchk (lun,1)
0730
0731       write(lun,111) ppd$b_flags
0732
0733       else if (operation_code .eq. 51) then
0734
0735       write(lun,105) 'MDATREC'
0736
0737       call linchk (lun,1)
0738
0739       write(lun,111) ppd$b_flags
0740
0741       else if (operation_code .eq. 11) then
0742
0743       write(lun,105) 'ID'
0744
0745       call linchk (lun,1)
0746
0747       write(lun,111) ppd$b_flags
0748
0749       else if (operation_code .eq. 19) then
0750
0751       write(lun,105) 'RETMDAT'
0752
0753       call linchk (lun,1)
0754
```

OC  
OC  
OC  
OC  
OC  
OC  
OC  
OC  
OC  
OC  
OC  
OC  
OC  
OC  
OC  
OC  
OC  
OC

PR

EM

VA

```
0755      write(lun,111) ppd$b_flags
0756
0757      else if (operation_code .eq. 4) then
0758
0759      write(lun,105) 'MCNF'
0760
0761      call linchk (lun,1)
0762
0763      write(lun,111) ppd$b_flags
0764      else
0765
0766      if (.not. response) then
0767
0768      115      write(lun,115) 'COMMAND, ',operation_code,'.'
0769      format(' ',t40,'PORT ',a,i<compress4-(operation_code)>,a)
0770
0771      call linchk (lun,1)
0772
0773      write(lun,111) ppd$b_flags
0774      else
0775
0776      write(lun,115) 'RESPONSE, ',operation_code,'.'
0777
0778      call linchk (lun,1)
0779
0780      write(lun,111) ppd$b_flags
0781      endif
0782      endif
0783
0784      if (message) then
0785
0786      do 123,i = 1,17
0787
0788      if (first_68_bytes_of_message(i) .ne. 0) goto 124
0789
0790      123      continue
0791
0792      goto 140
0793
0794      124      If ((error_subtype .eq. 7) .AND. (error_type .eq. 64)) then
0795
0796      If ((hsc$w_errlog_dg .eq. 5) .AND. (hsc$w_msglen .gt. 2)) then
0797      call linchk (lun,3)
0798      write(lun,85) '! Write a blank line
0799      write(lun,125) ''HSC'' ERROR LOG DATAGRAM'
0800      write(lun,125) hsc$t_nodename(1:8)
0801
0802      start_index = 1
0803      end_loc = hsc$w_msglen - 2
0804
0805      1111      j = STR$POSITION (hsc$t_message, sub_str, start_index)
0806
0807      c      If the search find the sub string past the end of the message
0808      c      then the search failed.
0809
0810      if (j .gt. (hsc$w_msglen - 2) ) then
0811      j = 0
```



PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	6069	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	1877	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	1112	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3 EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
Total Space Allocated		9570

ENTRY POINTS

Address	Type	Name
0-00000000		PADRIVER_LOGMESSAGE

VARIABLES

Address	Type	Name	Address	Type	Name
3-00000010	L*1	EMB\$B_LM_CLASS	3-00000014	L*1	EMB\$B_LM_NAMLANG
3-00000011	L*1	EMB\$B_LM_TYPE	3-00000000	I*4	EMB\$Q_HD_SID
3-00000015	CHAR	EMB\$T_LM_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	2-0000004C	I*4	END_LOC
2-00000034	I*4	ERROR_SUBTYPE	2-00000038	I*4	ERROR_TYPE
3-00000060	CHAR	HSC\$T_MESSAGE	3-00000056	CHAR	HSC\$T_NODENAME
3-0000004C	I*2	HSC\$W_ERRLOG_DG	3-0000005E	I*2	HSC\$W_MSGLN
2-00000050	I*4	I	2-00000058	I*4	J
AP-0000004@	L*1	LUN	2-00000054	I*4	MESSAGE
2-00000001	CHAR	MESSAGE_STRING	2-00000044	I*4	OPERATION_CODE
AP-00000008@	CHAR	OPTION	3-00000026	I*4	PADRIVER_ERROR_TYPE_CODE
2-0000003C	I*4	PATH	3-00000049	L*1	PPD\$B_FLAGS
3-00000048	L*1	PPD\$B_OPC	3-00000046	L*1	PPD\$B_PORT
3-00000047	L*1	PPD\$B_STATUS	2-00000040	I*4	REMOTE_NODE_NUMBER
3-0000003A	I*4	REMOTE_STATION_ADDRESS031	3-0000003E	I*2	REMOTE_STATION_ADDRESS3247
3-00000040	I*4	REMOTE_SYSTEM_ID031	3-00000044	I*2	REMOTE_SYSTEM_ID3247
2-00000000	L*1	RESPONSE	2-00000048	I*4	START_INDEX
2-00000033	CHAR	SUB_STR	3-0000002A	I*4	UCB\$L_ERRCNT

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)
3-0000004A	I*4	FIRST_68_BYTES_OF_MESSAGE	68	(17)
3-0000002E	I*2	LOCAL_STATION_ADDRESS	6	(3)
3-00000034	I*2	LOCAL_SYSTEM_ID	6	(3)
3-0000003A	I*2	REMOTE_STATION_ADDRESS	6	(3)
3-00000040	I*2	REMOTE_SYSTEM_ID	6	(3)

ST  
 00  
 00  
 00  
 00  
 00  
 00  
 00  
 00  
 00  
 00  
 00  
 00  
 00  
 00  
 00  
 PR  
 EN  
 VA  
 A  
 AR

LABELS

Address	Label	Address	Label	Address	Label	Address	Label	Address	Label	Address	Label
1-000003D9	10'	1-00000407	12'	1-0000040F	25'	1-00000442	30'	**	35	0-000005CE	40
0-00000605	45	1-0000045A	50'	0-00000628	55	**	65	0-00000647	70	0-0000067A	75
0-0000069D	80	1-00000462	85'	1-00000467	90'	1-0000047E	95'	1-0000049C	97'	1-000004B5	99'
1-000004CB	105'	1-000004D2	110'	1-000004E2	111'	1-000004FA	112'	1-0000051B	115'	**	123
0-00001527	124	1-00000537	125'	1-0000053F	130'	**	135	0-0000171C	140	0-000003E3	310
0-000003F0	315	0-000003FD	320	0-0000040A	325	0-00000417	330	0-00000424	335	0-00000431	340
0-0000043E	345	0-0000044B	350	0-00000458	355	0-00000463	360	0-0000046C	990	0-0000048E	992
0-000015DB	1111	1-00000530	2126'								

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
I*4	COMPRESS4		FLAGS		FLAGS_DS
	FLAGS_F		FLAGS_P		FLAGS_PF
	FRCTOF		HEADER	I*4	LIB\$EXITV
	LINCHK		LOGGER		PADRIVER_INITIALIZATION
	STATUS	I*4	STR\$POSITION		UCB\$B_ERTCNT
	UCB\$B_ERTMAX		UCB\$W_ERRCNT		

```
0001
0002
0003
0004 Subroutine FLAGS (lun,ppd$b_flags)
0005
0006 byte lun
0007 byte ppd$b_flags
0008
0009 integer*4 path_select
0010
0011
0012 call linchk (lun,1)
0013
0014 write(lun,5) ppd$b_flags
0015 5 format(' ',t8,'PPD$B_FLAGS',t30,z2.2)
0016
0017 if (lib$extzv(0,1,ppd$b_flags) .eq. 1) then
0018
0019 call linchk (lun,1)
0020
0021 write(lun,10) 'RESPONSE QUEUE BIT'
0022 10 format(' ',t40,a)
0023 endif
0024
0025 path_select = lib$extzv(1,2,ppd$b_flags)
0026
0027 call linchk (lun,1)
0028
0029 if (path_select .eq. 1) then
0030 write(lun,10) 'SELECT PATH #0.'
0031
0032 else if (path_select .eq. 2) then
0033 write(lun,10) 'SELECT PATH #1.'
0034
0035 endif
0036
0037 return
0038 End
```





```

0001
0002
0003      Subroutine FLAGS_PF (lun,ppd$b_flags)
0004
0005      byte          lun
0006      byte          ppd$b_flags
0007
0008
0009      call flags (lun,ppd$b_flags)
0010
0011      call linchk (lun,1)
0012
0013      if (lib$extzv(8,1,ppd$b_flags) .eq. 1) then
0014
0015      5      write(lun,5) "'NIBBLE' PACKED'
0016      format(' ',t40,a)
0017      else
0018
0019      write(lun,5) "'LONGWORD' PACKED'
0020      endif
0021
0022      return
0023
0024      End
  
```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	127	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	47	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	56	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	230	

ENTRY POINTS

Address	Type	Name
0-00000000		FLAGS_PF

VARIABLES

Address	Type	Name	Address	Type	Name
AP-00000004@	L*1	LUN	AP-00000008@	L*1	PPD\$B_FLAGS

0000  
0001  
0002  
0003  
0004  
0005  
0006  
0007  
0008  
0009  
0010  
0011  
0012  
0013  
0014  
0015  
0016  
0017  
0018  
0019  
0020  
0021  
0022  
0023  
0024  
0025  
0026  
0027  
0028  
0029  
0030  
0031  
0032  
0033  
0034  
0035  
0036  
0037  
0038  
0039  
0040  
0041  
0042  
0043  
0044  
0045  
0046  
0047  
0048  
0049  
0050  
0051  
0052  
0053  
0054  
0055  
0056  
0057  
0058  
0059  
0060  
0061  
0062  
0063  
0064  
0065  
0066  
0067  
0068  
0069  
0070  
0071  
0072  
0073  
0074  
0075  
0076  
0077  
0078  
0079  
0080  
0081  
0082  
0083  
0084  
0085  
0086  
0087  
0088  
0089  
0090  
0091  
0092  
0093  
0094  
0095  
0096  
0097  
0098  
0099  
0100



0001  
0002  
0003  
0004  
0005  
0006  
0007  
0008  
0009  
0010  
0011  
0012  
0013  
0014  
0015  
0016  
0017  
0018  
0019  
0020  
0021  
0022  
0023  
0024  
0025  
0026  
0027  
0028  
0029  
0030  
0031  
0032  
0033  
0034

Subroutine FLAGS\_P (lun,ppd\$b\_flags)

byte lun  
byte ppd\$b\_flags

integer\*4 packet\_multiple  
integer\*4 packet\_base\_size  
integer\*4 packet\_size  
integer\*4 compress4

call flags (lun,ppd\$b\_flags)

packet\_multiple = lib\$extzv (5,3,ppd\$b\_flags)  
packet\_base\_size = 512

if (lib\$extzv(8,1,ppd\$b\_flags) .eq. 1) packet\_base\_size = 576

packet\_size = packet\_base\_size \* (packet\_multiple + 1)

call linchk (lun,2)

write(lun,5) 'PACKET MULTIPLE ',packet\_multiple,  
1 ' - PACKET SIZE ',packet\_size,' BYTES'  
5 format(' ',t40,a,i<compress4 (packet\_multiple)>,/,  
1 t40,a,i<compress4 (packet\_size)>,a)

return

End



```

0001
0002
0003      Subroutine FLAGS_F (lun,ppd$b_flags)
0004
0005      byte          lun
0006      byte          ppd$b_flags
0007
0008
0009      call flags (lun,ppd$b_flags)
0010
0011      if (lib$extzv(8,1,ppd$b_flags) .eq. 1) then
0012
0013      call linchk (lun,1)
0014
0015      write(lun,5) 'FORCE RESET'
0016      5      format(' ',t40,a)
0017      endif
0018
0019      return
0020
0021      End
  
```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	91	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	26	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	48	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated		165

ENTRY POINTS

Address	Type	Name
0-00000000		FLAGS_F

VARIABLES

Address	Type	Name	Address	Type	Name
AP-00000004@	L*1	LUN	AP-00000008@	L*1	PPD\$b_FLAGS

LABELS

Address	Label
1-00000013	5'

FLAGS\_F

E 3  
16-Sep-1984 00:11:24  
5-Sep-1984 14:10:51

VAX-11 FORTRAN v3.4-56 Page 53  
DISK&VMSMASTER:[ERF.SRC]PADRIVER.FOR,1

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
	FLAGS	I*4	LIBSEXTZV		LINCHK

```

0001
0002
0003
0004      Subroutine FLAGS_DS (lun,ppd$b_flags)
0005
0006      byte          lun
0007      byte          ppd$b_flags
0008
0009
0010      call flags (lun,ppd$b_flags)
0011
0012      if (lib$extzv(8,1,ppd$b_flags) .eq. 1) then
0013
0014      call linchk (lun,1)
0015
0016      write(lun,5) 'DEFAULT STARTING ADDRESS'
0017      5      format(' ',t40,a)
0018      endif
0019
0020      return
0021
0022      End
  
```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	91	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	39	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	48	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
Total Space Allocated	178	

ENTRY POINTS

Address	Type	Name
0-00000000		FLAGS_DS

VARIABLES

Address	Type	Name	Address	Type	Name
AP-00000004@	L*1	LUN	AP-00000008@	L*1	PPDSB_FLAGS

00000000  
 00000001  
 00000002  
 00000003  
 00000004  
 00000005  
 00000006  
 00000007  
 00000008  
 00000009  
 00000010  
 00000011  
 00000012  
 00000013  
 00000014  
 00000015  
 00000016  
 00000017  
 00000018  
 00000019  
 00000020  
 00000021  
 00000022  
 00000023  
 00000024  
 00000025  
 00000026  
 00000027  
 00000028  
 00000029  
 00000030  
 00000031  
 00000032  
 00000033  
 00000034  
 00000035  
 00000036  
 00000037  
 00000038  
 00000039  
 00000040  
 00000041  
 00000042  
 00000043  
 00000044  
 00000045  
 00000046  
 00000047  
 00000048  
 00000049  
 00000050  
 00000051  
 00000052  
 00000053  
 00000054  
 00000055  
 00000056  
 00000057  
 00000058  
 00000059  
 00000060  
 00000061  
 00000062  
 00000063  
 00000064  
 00000065  
 00000066  
 00000067  
 00000068  
 00000069  
 00000070  
 00000071  
 00000072  
 00000073  
 00000074  
 00000075  
 00000076  
 00000077  
 00000078  
 00000079  
 00000080  
 00000081  
 00000082  
 00000083  
 00000084  
 00000085  
 00000086  
 00000087  
 00000088  
 00000089  
 00000090  
 00000091  
 00000092  
 00000093  
 00000094  
 00000095  
 00000096  
 00000097  
 00000098  
 00000099  
 00000100



FLAGS\_DS

6 3  
16-Sep-1984 00:11:24  
5-Sep-1984 14:10:51

VAX-11 FORTRAN V3.4-56 Page 55  
DISK\$VMSMASTER:[ERF.SRC]PADRIVER.FOR;1

LABELS

Address	Label
1-00000020	5'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
	FLAGS	I*4	LIB\$EXTZV		LINCHK

PC

PR

EN

VA

A

```
0001
0002
0003 Subroutine STATUS (lun,ppd$b_status)
0004
0005
0006 byte lun
0007 byte ppd h_status
0008
0009 integer*4 type
0010 integer*4 pth_1
0011 integer*4 pth_0
0012 integer*4 sub_type
0013 integer*4 compressc
0014
0015 character*5 v1status(0:0)
0016 data v1status(0) /*FAIL*/
0017
0018 character*20 path_status(0:3)
0019 data path_status(0) /*'ACK' OR NOT USED*/
0020 data path_status(1) /*'NAK'*/
0021 data path_status(2) /*NO RESPONSE*/
0022 data path_status(3) /*ARBITRATION TIMEOUT*/
0023
0024 character*25 subtype(0:3)
0025 data subtype(0) /*PACKET SIZE VIOLATION*/
0026 data subtype(1) /*UNRECOGNIZED PACKET*/
0027 data subtype(2) /*INVALID DESTINATION PORT*/
0028 data subtype(3) /*UNRECOGNIZED COMMAND*/
0029
0030 character*27 types(0:6)
0031 data types(0) /*NORMAL*/
0032 data types(1) /*VIRTUAL CIRCUIT CLOSED*/
0033 data types(2) /*INVALID BUFFER NAME*/
0034 data types(3) /*BUFFER LENGTH VIOLATION*/
0035 data types(4) /*ACCESS CONTROL VIOLATION*/
0036 data types(5) /*NO PATH*/
0037 data types(6) /*BUFFER MEMORY SYSTEM ERROR*/
0038
0039
0040 type = lib$extzv(5,3,ppd$b_status)
0041 pth_1 = lib$extzv(3,2,ppd$b_status)
0042 pth_0 = lib$extzv(1,2,ppd$b_status)
0043 sub_type = lib$extzv(1,4,ppd$b_status)
0044
0045 call output (lun,ppd$b_status,v1status,0,0,0,'0')
0046
0047 if (type .eq. 7) then
0048
0049 call linchk (lun,1)
0050
0051 write(lun,10) subtype(sub_type)
0052 10 format(' ',t40,a<compressc (subtype(sub_type))>>)
0053 else
0054
0055 call linchk (lun,2)
0056
0057 write(lun,15) '0',path_status(pth_0)
```

STATUS

1 3  
 16-Sep-1984 00:11:24  
 5-Sep-1984 14:10:51

VAX-11 FORTRAN V3.4-56  
 DISK\$VMSMASTER:[ERF.SRC]PADRIVER.FOR;1

```

0058 15  format(' ',t40,'PATH #',a,' ',t40,
0059      1 a<compressc (path_status(pth_0))>>)
0060
0061      pth_0 = pth_1
0062
0063      write(lun,15) '1'.path_status(pth_1)
0064
0065      call linchk (lun,1)
0066
0067      write(lun,20) types(type)
0068 20  format(' ',t40,a<compressc (types(type))>>)
0069      endif
0070
0071      return
0072
0073      End
    
```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	488	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	77	PIC CON REL LCL CHR NOEXE RD NOWRT LONG
2 \$LOCAL	616	PIC CON REL LCL NOSMP NOEXE RD WRT LONG
Total Space Allocated	1181	

ENTRY POINTS

Address	Type	Name
0-00000000		STATUS

VARIABLES

Address	Type	Name	Address	Type	Name	Address	Type	Name	Address	Type	Name
AP-00000004	L*1	LUN	AP-00000008	L*1	PPDSB_STATUS	2-00000180	I*4	PTH_0	2-0000017C	I*4	PTH_1
2-00000184	I*4	SUB_TYPE	2-00000178	I*4	TYPE						

ARRAYS

Address	Type	Name	Bytes	Dimensions
2-00000005	CHAR	PATH STATUS	80	(0:3)
2-00000055	CHAR	SUBTYPE	100	(0:3)
2-00000089	CHAR	TYPES	189	(0:6)
2-00000000	CHAR	VISTATUS	5	(0:0)



The image displays a grid of 100 small terminal window screenshots, arranged in 10 rows and 10 columns. Each window shows a different screen from the VAX/VMS V4.0 software. The screens contain various data, including lists, tables, and graphical elements like bar charts. Some screens are clearly labeled with titles such as 'PADRIVER LIS', 'OUTPUT LIS', 'NEW RTN LIS', 'MT DISMT LIS', 'OPNOUTFIL LIS', and 'NEWFILE LIS'. Each window also displays a date and time stamp, typically '11:11:11'. The overall appearance is that of a dense collection of software output screens.

