





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

```

SUBROUTINE TUTAPE (LUN)
C
C Version:      'V04-000'
C
C*****
C*
C*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
C*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
C*  ALL RIGHTS RESERVED.
C*
C*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
C*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
C*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
C*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
C*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
C*  TRANSFERRED.
C*
C*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
C*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
C*  CORPORATION.
C*
C*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
C*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
C*
C*****
C
C
C          AUTHOR  BRIAN PORTER             CREATION DATE  5-OCT-1979
C
C          Fucntional description:
C
C          This module displays TUS8 errors.
C
C          Modified by:
C
C          V03-004 SAR0236      Sharon A. Reynolds,  28-Mar-1984
C                        Changed the call to UCBSL_OWNUIC to ORBSL_OWNER.
C
C          V03-003 SAR0130      Sharon A. Reynolds,   8-Sep-1983
C                        Changed the overprint carriage control in the 'format'
C                        statements for use with ERF.
C
C          V03-002 SAR0100      Sharon A. Reynolds,  20-Jun-1983
C                        Changed the carriage control in the 'format' statements
C                        for use with ERF.
C
C          V03-001 SAR0052      Sharon A. Reynolds,  13-Jun-1983
C                        Removed brief/cryptic support.
C
C          v02-004 BP0004       Brian Porter,         23-NOV-1981
C                        Minor edit.
C
C          v02-003 BP0003       Brian Porter,         05-NOV-1981
C                        Added 'device attention' support.
C

```

TUT

032  
032  
032  
032  
032  
032  
032  
032  
032  
032  
033  
033  
033  
033  
033  
033  
033  
033  
033  
033  
033

PRO

0  
1  
2  
3

ENT

C

VAR

AF

0058  
0059  
0060  
0061  
0062  
0063  
0064  
0065  
0066  
0125  
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

C  
C  
C  
C  
C  
C\*\*

v02-002 BP0002 Brian Porter, 01-JUL-1981  
Added call to LOGGER and DHEAD. Added DIAGNOSTIC\_MODE.  
v02-001 BP0001 Brian Porter, 19-JUL-1980  
Corrected Fortran format error when end packet has illegal  
byte count. Removed call to UNUSED\_BITS.

INCLUDE 'SRC\$:MSGHDR.FOR /NOLIST'  
INCLUDE 'SRC\$:DEVERR.FOR /NOLIST'

BYTE LUN  
BYTE CMD\_PKT(0:13)  
BYTE END\_PKT(0:13)  
  
INTEGER\*4 FIELD  
INTEGER\*4 RCSR  
INTEGER\*4 RBUF  
INTEGER\*4 XCSR  
INTEGER\*4 XBUF  
INTEGER\*4 COMPRESS4  
INTEGER\*4 COMPRESSC  
  
logical\*1 diagnostic\_mode  
  
PARAMETER TIMEOUT = 96  
PARAMETER READ = 2  
PARAMETER WRITE = 3  
PARAMETER POSITION = 5  
  
EQUIVALENCE (EMB\$\$\_DV\_REGS AV(0),XCSR)  
EQUIVALENCE (EMB\$\$\_DV\_REGS AV(1),XBUF)  
EQUIVALENCE (EMB\$\$\_DV\_REGS AV(2),RCSR)  
EQUIVALENCE (EMB\$\$\_DV\_REGS AV(3),RBUF)  
EQUIVALENCE (EMB(98),CMD\_PKT(0))  
EQUIVALENCE (EMB(112),END\_PKT(0))  
  
CHARACTER\*14 UNIT\_NUMBER  
DATA UNIT\_NUMBER /'UNIT NUMBER = '/  
  
CHARACTER\*11 UNUSED\_BYTE  
DATA UNUSED\_BYTE /'UNUSED BYTE'/  
  
CHARACTER\*19 CONTROL\_PACKET  
DATA CONTROL\_PACKET /'TU58 CONTROL PACKET'/  
  
CHARACTER\*14 COMMAND\_PACKET  
DATA COMMAND\_PACKET /'COMMAND PACKET'/  
  
CHARACTER\*7 OP\_CODE  
DATA OP\_CODE /'OP CODE'/  
  
CHARACTER\*17 PBYTE\_COUNT  
DATA PBYTE\_COUNT /'PACKET BYTE COUNT'/

TUT

AP

ARR

2

LAB

FUN

T

COM

F

/

/

/

/

```

0273
0274 CHARACTER*15 SEQUENCE_LO
0275 DATA SEQUENCE_LO 7'SEQUENCE LO, HI'/
0276
0277 CHARACTER*17 BYTECOUNT_LO
0278 DATA BYTECOUNT_LO 7'BYTE COUNT LO, HI'/
0279
0280 CHARACTER*19 BLOCK_LO
0281 DATA BLOCK_LO 7'BLOCK NUMBER LO, HI'/
0282
0283 CHARACTER*15 CHECKSUM_LO
0284 DATA CHECKSUM_LO 7'CHECKSUM LO, HI'/
0285
0286 CHARACTER*17 V1RCSR(6:7)
0287 DATA V1RCSR(6) 7'INTERRUPT ENABLE*'/
0288 DATA V1RCSR(7) 7'RECEIVER DONE*'/
0289
0290 CHARACTER*16 V2RCSR(11:11)
0291 DATA V2RCSR(11) 7'RECEIVER ACTIVE*'/
0292
0293 CHARACTER*15 V1RBUF(12:15)
0294 DATA V1RBUF(12) 7'PARITY ERROR*'/
0295 DATA V1RBUF(13) 7'FRAMING ERROR*'/
0296 DATA V1RBUF(14) 7'OVER-RUN ERROR*'/
0297 DATA V1RBUF(15) 7'ERROR*'/
0298
0299 CHARACTER*6 V1XCSR(0:0)
0300 DATA V1XCSR(0) 7'BREAK*'/
0301
0302 CHARACTER*18 V2XCSR(6:7)
0303 DATA V2XCSR(6) 7'INTERRUPT ENABLE*'/
0304 DATA V2XCSR(7) 7'TRANSMITTER READY*'/
0305
0306 CHARACTER*20 V1OP_CODE(-1:12)
0307 DATA V1OP_CODE(-1) 7'ILLEGAL FUNCTION*'/
0308 DATA V1OP_CODE(0) 7'NO OPERATION*'/
0309 DATA V1OP_CODE(1) 7'INITIALIZE*'/
0310 DATA V1OP_CODE(2) 7'READ*'/
0311 DATA V1OP_CODE(3) 7'WRITE*'/
0312 DATA V1OP_CODE(4) 7'COMPARE*'/
0313 DATA V1OP_CODE(5) 7'POSITION*'/
0314 DATA V1OP_CODE(6) 7'ABORT*'/
0315 DATA V1OP_CODE(7) 7'DIAGNOSE*'/
0316 DATA V1OP_CODE(8) 7'GET STATUS*'/
0317 DATA V1OP_CODE(9) 7'SET STATUS*'/
0318 DATA V1OP_CODE(10) 7'GET CHARACTERISTICS*'/
0319 DATA V1OP_CODE(11) 7'SET CHARACTERISTICS*'/
0320 DATA V1OP_CODE(12) 7'ILLEGAL FUNCTION*'/
0321
0322 CHARACTER*16 V2OP_CODE(63:65)
0323 DATA V2OP_CODE(63) 7'ILLEGAL OP CODE*'/
0324 DATA V2OP_CODE(64) 7'SEND END PACKET*'/
0325 DATA V2OP_CODE(65) 7'ILLEGAL OP CODE*'/
0326
0327 CHARACTER*10 MODIFIER(0:0)
0328 DATA MODIFIER(0) 7'DATACHECK*'/
0329

```

```

0330
0331 CHARACTER*18 V1SUM STATUS(4:7)
0332 DATA V1SUM_STATUS(4)/'LOGIC ERROR*'/
0333 DATA V1SUM_STATUS(5)/'MOTION ERROR*'/
0334 DATA V1SUM_STATUS(6)/'TRANSFER ERROR*'/
0335 DATA V1SUM_STATUS(7)/'SPECIAL CONDITION*'/
0336
0337 BYTE V1SUCCESS_CODE(0:11)
0338 DATA V1SUCCESS_CODE(0)'00'X/
0339 DATA V1SUCCESS_CODE(1)'01'X/
0340 DATA V1SUCCESS_CODE(2)'FF'X/
0341 DATA V1SUCCESS_CODE(3)'FE'X/
0342 DATA V1SUCCESS_CODE(4)'F8'X/
0343 DATA V1SUCCESS_CODE(5)'F7'X/
0344 DATA V1SUCCESS_CODE(6)'F5'X/
0345 DATA V1SUCCESS_CODE(7)'EF'X/
0346 DATA V1SUCCESS_CODE(8)'E0'X/
0347 DATA V1SUCCESS_CODE(9)'DF'X/
0348 DATA V1SUCCESS_CODE(10)'D0'X/
0349 DATA V1SUCCESS_CODE(11)'C9'X/
0350
0351 CHARACTER*22 V2SUCCESS_CODE(0:12)
0352 DATA V2SUCCESS_CODE(0)'NORMAL COMPLETION*'/
0353 DATA V2SUCCESS_CODE(1)'RETRY COMPLETION*'/
0354 DATA V2SUCCESS_CODE(2)'SELF TEST FAILURE*'/
0355 DATA V2SUCCESS_CODE(3)'PARTIAL OPERATION*'/
0356 DATA V2SUCCESS_CODE(4)'INVALID UNIT NUMBER*'/
0357 DATA V2SUCCESS_CODE(5)'NO CARTRIDGE PRESENT*'/
0358 DATA V2SUCCESS_CODE(6)'WRITE PROTECTED*'/
0359 DATA V2SUCCESS_CODE(7)'DATA CHECK*'/
0360 DATA V2SUCCESS_CODE(8)'BLOCK NOT FOUND*'/
0361 DATA V2SUCCESS_CODE(9)'MOTOR STOPPED*'/
0362 DATA V2SUCCESS_CODE(10)'INVALID OP CODE*'/
0363 DATA V2SUCCESS_CODE(11)'INVALID RECORD NUMBER*'/
0364 DATA V2SUCCESS_CODE(12)'ILLEGAL SUCCESS CODE*'/
0365
0366 CHARACTER*9 BYTE_COUNT(9:11)
0367 DATA BYTE_COUNT(9) /' ILLEGAL*'/
0368 DATA BYTE_COUNT(10) /' = 10.*'/
0369 DATA BYTE_COUNT(11) /' ILLEGAL*'/
0370
0371
0372 CALL FRCTOF (LUN)
0373
0374 if (emb$t_dv_name(1:3) .eq. 'CSA') then
0375
0376 call dhead1 (lun,'CONSOLE TU58')
0377 else
0378
0379 call dhead1 (lun,'UBA TU58')
0380 endif
0381
0382 diagnostic_mode = .false.
0383
0384 if (lib$extzv(2,1,xcsr) .eq. 1) diagnostic_mode = .true.
0385
0386 CALL LINCHK (LUN,2)

```



```

0444
0445     IF (CMD_PKT(0) .NE. 2) THEN
0446
0447     WRITE(LUN,40)
0448 40    FORMAT(/' ','COMMAND PACKET HAS INVALID FLAG',/)
0449
0450     DO 44,I = 0,14
0451
0452     CALL LINCHK (LUN,1)
0453
0454     WRITE(LUN,42) CMD_PKT(1)
0455 42    FORMAT(' ',T30,Z2.2)
0456
0457 44    CONTINUE
0458     ELSE
0459
0460     WRITE(LUN,45)
0461 45    FORMAT(/' ','COMMAND MESSAGE PACKET',/)
0462
0463     CALL LINCHK (LUN,2)
0464
0465     WRITE(LUN,50) CMD_PKT(0),CONTROL_PACKET
0466 50    FORMAT(' ',T8,'FLAG',T30,Z2.2,/,T40,A19)
0467
0468     CALL LINCHK (LUN,2)
0469
0470     FIELD = LIB$EXTZV(0.8,CMD_PKT(1))
0471
0472     WRITE(LUN,55) PBYTE_COUNT,CMD_PKT(1),PBYTE_COUNT,
0473 1 BYTE_COUNT(MAX(9,MIN(11,FIELD)))
0474 55    FORMAT(' ',T8,A17,T30,Z2.2,/,
0475 1 T40,A17,A<COMPRESSC (BYTE_COUNT(MAX(9,MIN(11,FIELD))))>>)
0476
0477     CALL LINCHK (LUN,2)
0478
0479     FIELD = LIB$EXTZV(0.8,CMD_PKT(2))
0480
0481     WRITE(LUN,60) OP_CODE,CMD_PKT(2),V1OP_CODE(MAX(-1,MIN(11,FIELD)))
0482 60    FORMAT(' ',T8,A7,T30,Z2.2,/,T40,
0483 1 'FUNCTION = ',A<COMPRESSC (V1OP_CODE(MAX(-1,MIN(11,FIELD))))>>)
0484
0485     CALL LINCHK (LUN,1)
0486
0487     WRITE(LUN,65) CMD_PKT(3)
0488 65    FORMAT(' ',T8,'OP CODE MODIFIER',T30,Z2.2)
0489
0490     CALL OUTPUT (LUN,CMD_PKT(3),MODIFIER,0,0,0,'0')
0491
0492     CALL LINCHK (LUN,2)
0493
0494     FIELD = LIB$EXTZV(0.8,CMD_PKT(4))
0495
0496     WRITE(LUN,68) CMD_PKT(4),UNIT_NUMBER,FIELD
0497 68    FORMAT(' ',T8,'UNIT',T30,Z2.2,/,
0498 1 T40,A14,I<COMPRESSC (FIELD)>,'.')
0499
0500     CALL LINCHK (LUN,1)

```

```

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

```



```

0501
0502 70 WRITE(LUN,70) UNUSED_BYTE,CMD_PKT(5)
0503   FORMAT(' ',T8,A11,T30,Z2.2)
0504
0505   CALL LINCHK (LUN,2)
0506
0507 75 WRITE(LUN,75) SEQUENCE_LO,CMD_PKT(6),CMD_PKT(7)
0508   FORMAT(' ',T8,A15,T30,Z2.2,/,T30,Z2.2)
0509
0510   CALL LINCHK (LUN,2)
0511
0512 80 WRITE(LUN,80) BYTECOUNT_LO,CMD_PKT(8)
0513   FORMAT(' ',T8,A17,T30,Z2.2)
0514
0515   Field = ' '
0516
0517   IF (CMD_PKT(2) .EQ. READ
0518       1 .OR.
0519       2 CMD_PKT(2) .EQ. WRITE) THEN
0520
0521   FIELD = LIB$EXTZV(0,16,CMD_PKT(8))
0522   ENDIF
0523
0524 85 WRITE(LUN,85) CMD_PKT(9),FIELD
0525   FORMAT(' ',T30,Z2.2,/,T40,
0526       1 'TRANSFER BYTE COUNT = ',I<COMPRESS4 (FIELD)>,'.')
0527
0528   CALL LINCHK (LUN,2)
0529
0530 90 WRITE(LUN,90) BLOCK_LO,CMD_PKT(10)
0531   FORMAT(' ',T8,A19,T30,Z2.2)
0532
0533   Field = ' '
0534
0535   IF (CMD_PKT(2) .EQ. POSITION
0536       1 .OR.
0537       2 CMD_PKT(2) .EQ. READ
0538       3 .OR.
0539       4 CMD_PKT(2) .EQ. WRITE) THEN
0540
0541   FIELD = LIB$EXTZV(0,16,CMD_PKT(10))
0542   ENDIF
0543
0544 95 WRITE(LUN,95) CMD_PKT(11),FIELD
0545   FORMAT(' ',T30,Z2.2,/,T40,
0546       1 'REQUESTED BLOCK = ',I<COMPRESS4 (FIELD)>,'.')
0547
0548   CALL LINCHK (LUN,2)
0549
0550 100 WRITE(LUN,100) CHECKSUM_LO,CMD_PKT(12),CMD_PKT(13)
0551   FORMAT(' ',T8,A15,T30,Z2.2,/,T30,Z2.2)
0552   ENDIF
0553
0554   CALL LINCHK (LUN,3)
0555
0556   IF (END_PKT(0) .NE. 2) THEN
0557

```

```
0558  
0559  
0560 105 WRITE(LUN,105)  
0561 FORMAT(/' ', 'END PACKET HAS INVALID FLAG',/)  
0562  
0563 DO 115,I = 0,13  
0564 CALL LINCHK (LUN,1)  
0565  
0566 WRITE(LUN,110) END_PKT(I)  
0567 110 FORMAT(' ',T30,Z2.2)  
0568  
0569 115 CONTINUE  
0570 ELSE  
0571  
0572 WRITE(LUN,120)  
0573 120 FORMAT(/' ', 'END MESSAGE PACKET',/)  
0574  
0575 CALL LINCHK (LUN,2)  
0576  
0577 WRITE(LUN,125) END_PKT(0),CONTROL_PACKET  
0578 125 FORMAT(' ',T8,'FLAG',T30,Z2.2,/,T40,A19)  
0579  
0580 CALL LINCHK (LUN,2)  
0581  
0582 FIELD = LIB$EXTZV(0,8,END_PKT(1))  
0583  
0584 WRITE(LUN,130) PBYTE_COUNT,END_PKT(1),PBYTE_COUNT,  
0585 1 BYTE_COUNT(MAX(9,MIN(11,FIELD)))  
0586 130 FORMAT(' ',T8,A17,T30,Z2.2,/  
0587 1 T40,A17,A<COMPRESSC (BYTE_COUNT(MAX(9,MIN(11,FIELD))))>>  
0588  
0589 CALL LINCHK (LUN,2)  
0590  
0591 FIELD = LIB$EXTZV(0,8,END_PKT(2))  
0592  
0593 WRITE(LUN,135) OP_CODE,END_PKT(2),V2OP_CODE(MAX(63,MIN(65,FIELD)))  
0594 135 FORMAT(' ',T8,A7,T30,Z2.2,/  
0595 1 T40,A<COMPRESSC (V2OP_CODE(MAX(63,MIN(65,FIELD))))>>  
0596  
0597 CALL LINCHK (LUN,2)  
0598  
0599 WRITE(LUN,140) END_PKT(3)  
0600 140 FORMAT(' ',T8,'SUCCESS CODE',T30,Z2.2)  
0601  
0602 J = 12  
0603  
0604 DO 155,I = 0,11  
0605  
0606 IF (END_PKT(3) .EQ. V1SUCCESS_CODE(I)) J = I  
0607  
0608 155 CONTINUE  
0609  
0610 WRITE(LUN,160) V2SUCCESS_CODE(J)  
0611 160 FORMAT(' ',T40,A<COMPRESSC (V2SUCCESS_CODE(J))>>  
0612  
0613 CALL LINCHK (LUN,2)  
0614
```



```

0672
0673      call ucb$w_errcnt (lun,emb$w_dv_errcnt)
0674
0675      if (emb$w_hd_entry .ne. 98) then
0676
0677      call ucb$l_media (lun,emb$l_dv_media)
0678
0679      call linchk (lun,1)
0680
0681      write(lun,200)
0682
0683      call tutape_qio (lun,emb$w_dv_func)
0684
0685      call irp$w_bcnc (lun,emb$w_dv_bcnc)
0686
0687      call irp$w_boff (lun,emb$w_dv_boff)
0688
0689      call irp$l_pid (lun,emb$l_dv_rpid)
0690
0691      call irp$q_iosb (lun,emb$l_dv_iosb1)
0692      endif
0693
0694      RETURN
0695      END

```

## PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	3247	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	906	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	2008	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3 EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
Total Space Allocated		6673

## ENTRY POINTS

Address	Type	Name
0-00000000		TUTAPE

## VARIABLES

Address	Type	Name	Address	Type	Name
2-000003EA	CHAR	BLOCK_LO	2-000003D9	CHAR	BYTECOUNT_LO
2-000003FD	CHAR	CHECKSUM_LO	2-000003A4	CHAR	COMMAND_PACKET
2-00000391	CHAR	CONTROL_PACKET	2-00000377	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_NAMLANG
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	EMBSL_DV_NAME
3-00000024	I*2	EMBSW_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
2-0000040C	I*4	FIELD	2-00000410	I*4	I
2-00000414	I*4	J	AP-00000004	L*1	LUN
2-000003B2	CHAR	OP_CODE	2-000003B9	CHAR	PBYTE_COUNT
3-0000005E	I*4	RBOF	3-0000005A	I*4	RCSR
2-000003CA	CHAR	SEQUENCE_LO	2-00000378	CHAR	UNIT_NUMBER
2-00000386	CHAR	UNUSED_BYTE	3-00000056	I*4	XBUF
3-00000052	I*4	XCSR			

ARRAYS

Address	Type	Name	Bytes	Dimensions
2-0000035C	CHAR	BYTE_COUNT	27	(9:11)
3-00000062	L*1	CMD_PKT	14	(0:13)
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)
3-00000070	L*1	END_PRT	14	(0:13)
2-000001E0	CHAR	MODIFIER	10	(0:0)
2-00000098	CHAR	V1OP_CODE	280	(-1:12)
2-00000032	CHAR	V1RBOF	60	(12:15)
2-00000000	CHAR	V1RCSR	34	(6:7)
2-00000232	L*1	V1SUCCESS_CODE	12	(0:11)
2-000001EA	CHAR	V1SUM_STATUS	72	(4:7)
2-0000006E	CHAR	V1XCSR	6	(0:0)
2-000001B0	CHAR	V2OP_CODE	48	(63:65)
2-00000022	CHAR	V2RCSR	16	(11:11)
2-0000023E	CHAR	V2SUCCESS_CODE	286	(0:12)
2-00000074	CHAR	V2XCSR	36	(6:7)

LABELS

Address	Label	Address	Label	Address	Label	Address	Label	Address	Label
1-00000040	10'	1-00000052	15'	1-00000063	18'	**	19	1-0000007E	20'
1-000000A6	30'	1-000000B7	40'	1-000000DE	42'	**	44	1-000000E7	45'
1-00000118	55'	1-00000133	60'	1-00000156	65'	1-00000173	68'	1-00000192	70'
1-000001B2	80'	1-000001BF	85'	1-000001EC	90'	1-000001F9	95'	1-00000222	100'
1-00000258	110'	**	115	1-00000261	120'	1-0000027B	125'	1-00000291	130'
1-000002BF	140'	**	155	1-000002D8	160'	1-000002E4	170'	1-00000303	175'
1-00000323	185'	1-00000356	190'	1-0000036F	195'	1-00000382	200'	1-00000310	180'



0001  
0002  
0003  
0004  
0005  
0006  
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

```
Subroutine TUTAPE_QIO (lun,emb$w_dv_func)
include 'src$:qiocommon.for /nolist'
byte          lun
integer*2     emb$w_dv_func
integer*4     qiocode(0:1,0:63)

if (qiocode(0,0) .eq. 0) then
qiocode(1,09) = %loc(io$_search)
qiocode(1,11) = %loc(io$_writepblk)
qiocode(1,12) = %loc(io$_readpblk)
qiocode(1,26) = %loc(io$_setchar)
qiocode(1,27) = %loc(io$_sensechar)
qiocode(1,29) = %loc(io$_diagnose)
qiocode(1,32) = %loc(io$_writelblk)
qiocode(1,33) = %loc(io$_readlblk)
qiocode(1,35) = %loc(io$_setmode)
qiocode(1,39) = %loc(io$_sensemode)
qiocode(1,48) = %loc(io$_writevblk)
qiocode(1,49) = %loc(io$_readvblk)
qiocode(1,50) = %loc(io$_acress)
qiocode(1,51) = %loc(io$_create)
qiocode(1,52) = %loc(io$_deaccess)
qiocode(1,53) = %loc(io$_delete)
qiocode(1,54) = %loc(io$_modify)
qiocode(1,56) = %loc(io$_acpcontrol)
qiocode(1,57) = %loc(io$_mount)
do 10,i = 0,63
```

DW7  
PRO  
0  
1  
2  
ENT  
0  
VAR  
2  
AP  
ARR  
2  
2  
LAB  
1  
FUN  
T

```

0321      qiocode(0,i) = 33
0322
0323      if (qiocode(1,i) .eq. 0) then
0324
0325      qiocode(1,i) = %loc(qio_string)
0326      endif
0327
0328      10      continue
0329      endif
0330
0331      call irp$w_func (lun,emb$w_dv_func,
0332      1 qiocode(0,lib$extzv(0,6,emb$w_dv_func)))
0333
0334      return
0335
0336      end
0337

```

## PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	225	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	8	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	548	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3 QIOCOMMON	1247	PIC OVR REL GBL SHR NOEXE RD WRT LONG
Total Space Allocated		2028

## ENTRY POINTS

Address	Type	Name
0-00000000		TUTAPE_QIO

## VARIABLES

Address	Type	Name	Address	Type	Name
AP-00000008a	I*2	EMBSW DV FUNC	2-00000200	I*4	I
3-00000442	CHAR	IO\$_ABORT	3-0000034D	CHAR	IO\$_ACCESS
3-000003C2	CHAR	IO\$_ACPCONTROL	3-000004B3	CHAR	IO\$_AVAILABLE
3-00000297	CHAR	IO\$_CLEAN	3-00000369	CHAR	IO\$_CREATE
3-00000385	CHAR	IO\$_DEACCESS	3-00000393	CHAR	IO\$_DELETE
3-0000026D	CHAR	IO\$_DIAGNOSE	3-00000065	CHAR	IO\$_DRVCLR
3-000004CB	CHAR	IO\$_DSE	3-000000A9	CHAR	IO\$_ERASETAPE
3-00000276	CHAR	IO\$_FORMAT	3-00000071	CHAR	IO\$_INITIALIZE
3-00000014	CHAR	IO\$_LOADMCODE	3-000003A1	CHAR	IO\$_MODIFY
3-000003E2	CHAR	IO\$_MOUNT	3-00000000	CHAR	IO\$_NOP
3-0000009D	CHAR	IO\$_OFFSET	3-000000EB	CHAR	IO\$_PACKACK
3-000000E0	CHAR	IO\$_QSTOP	3-000003EF	CHAR	IO\$_RDSTATS
3-00000421	CHAR	IO\$_READCSR	3-00000169	CHAR	IO\$_READHEAD

000  
000  
000  
000  
000  
000  
000  
000  
000  
000  
001  
001  
001  
001  
001  
001  
001  
001  
001  
001  
002  
002  
002  
002  
002  
002  
002

PRO

0  
1  
2

ENT

VAR

2



3-000002B6	CHAR	IOS_READLBLK	3-0000013F	CHAR	IOS_READPBLK
3-00000200	CHAR	IOS_READPRESET	3-00000195	CHAR	IOS_READTRACKD
3-0000033A	CHAR	IOS_READVBLK	3-0000045A	CHAR	IOS_READWTHBUF
3-00000484	CHAR	IOS_READWTHXBUF	3-0000004D	CHAR	IOS_RECAL
3-0000007C	CHAR	IOS_RELEASE	3-000001AB	CHAR	IOS_REREADN
3-000001B8	CHAR	IOS_REREADP	3-000000CA	CHAR	IOS_RETCENTER
3-000002E6	CHAR	IOS_REWIND	3-000002C9	CHAR	IOS_REWINDOFF
3-000000FC	CHAR	IOS_SEARCH	3-00000024	CHAR	IOS_SEEK
3-00000231	CHAR	IOS_SENSECHAR	3-00000309	CHAR	IOS_SENSEMODE
3-0000021D	CHAR	IOS_SETCHAR	3-000003B8	CHAR	IOS_SETCLOCK
3-00000088	CHAR	IOS_SETCLOCKP	3-000002DD	CHAR	IOS_SETMODE
3-000002ED	CHAR	IOS_SKIPFILE	3-000002FA	CHAR	IOS_SKIPRECORD
3-00000029	CHAR	IOS_SPACEFILE	3-0000010E	CHAR	IOS_SPACERECORD
3-000003D7	CHAR	IOS_STARTDATA	3-000000B4	CHAR	IOS_STARTDATAP
3-00000077	CHAR	IOS_STARTMPROC	3-0000020F	CHAR	IOS_STARTSPNDL
3-00000059	CHAR	IOS_STOP	3-0000000D	CHAR	IOS_UNLOAD
3-0000046B	CHAR	IOS_WRITEBUFNCRC	3-0000011E	CHAR	IOS_WRITECHECK
3-000001E4	CHAR	IOS_WRITECHECKH	3-000003FF	CHAR	IOS_WRITECSR
3-00000153	CHAR	IOS_WRITEHEAD	3-000002A2	CHAR	IOS_WRITELBLK
3-00000247	CHAR	IOS_WRITEMARK	3-00000314	CHAR	IOS_WRITEOF
3-0000012A	CHAR	IOS_WRITEPBLK	3-000001C9	CHAR	IOS_WRITERET
3-0000017E	CHAR	IOS_WRIETRACKD	3-00000326	CHAR	IOS_WRITEVBLK
3-00000448	CHAR	IOS_WRITEWTHBUF	3-00000257	CHAR	IOS_WRTTMKR
AP-00000004@	L*1	LUN	3-000004A1	CHAR	QIO_STRING

ARRAYS

Address	Type	Name	Bytes	Dimensions
2-00000000	I*4	QIOCODE	512	(0:1, 0:63)

LABELS

Address	Label
**	10

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name
	IRPSW_FUNC	I*4	LIB\$EXTZV

COMMAND QUALIFIERS

```

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

TUTAPE\_Q10

E 12  
16-Sep-1984 00:16:59  
5-Sep-1984 14:24:09

VAX-11 FORTRAN V3.4-56  
DISK\$VMSMASTER:[ERF.SRC]TUTAPE.FOR;1 Page 16

COMPILATION STATISTICS

Run Time: 11.14 seconds  
Elapsed Time: 29.09 seconds  
Page Faults: 266  
Dynamic Memory: 251 pages

000  
000  
000  
000  
000  
000  
000  
000  
001  
001  
001  
001  
001  
001  
001  
001  
001  
007  
007  
007  
007  
008  
008  
008  
008  
008  
008  
008  
008  
008  
008  
008  
008  
009  
009  
009  
009  
009  
009  
009  
009  
009  
009  
010  
010  
010  
010  
010  
010  
010  
010  
010  
010  
011  
011  
011  
011  
011  
011

This page contains a grid of 120 small, faded screenshots of VAX/VMS system logs, arranged in 10 rows and 12 columns. Each screenshot displays a different log file, with the filename prominently displayed in the center of the screen. The logs show various system events, error messages, and operational data. The visible log titles include:

- TIMCPL LIS
- TUTAPE LIS
- SBI LIS
- STSEVENT LIS
- TRNS BITS LIS
- LBAERR LIS
- TIMRB LIS
- LBA LIS
- TU8 SENSE LIS
- UNDEFINED LIS
- SYSWRFL LIS
- SUMMARY LIS
- UNKN DISP LIS
- SHRVECTOR LIS
- SYSTARTUP LIS
- LIBAINT LIS
- TOF LIS
- UNKNOWN LIS
- TSTAPE LIS

The text within each screenshot is significantly faded and difficult to read, but the overall structure of each log is consistent, showing headers, timestamps, and lines of system output.