

```

EEEEEEEEEEEEEEEEEE RRRRRRRRRRRR FFFFFFFF
EEEEEEEEEEEEEEEEEE RRRRRRRRRRRR FFFFFFFF
EEEEEEEEEEEEEEEEEE RRRRRRRRRRRR FFFFFFFF
EEE RRR RRR FFF
EEE RRR RRR FFF
EEE RRR RRR FFF
EEE RRR RRR FFF
EEE RRR RRR FFF
EEE RRR RRR FFF
EEEEEEEEEEEEEEEEEE RRRRRRRRRRRR FFFFFFFF
EEEEEEEEEEEEEEEEEE RRRRRRRRRRRR FFFFFFFF
EEEEEEEEEEEEEEEEEE RRRRRRRRRRRR FFFFFFFF
EEE RRR RRR FFF
EEE RRR RRR FFF
EEE RRR RRR FFF
EEE RRR RRR FFF
EEE RRR RRR FFF
EEE RRR RRR FFF
EEEEEEEEEEEEEEEEEE RRRRRRRRRRRR FFFF
EEEEEEEEEEEEEEEEEE RRRRRRRRRRRR FFFF
EEEEEEEEEEEEEEEEEE RRRRRRRRRRRR FFFF

```

[illegible]

```

MM      MM      AAAAAA      SSSSSSSS      SSSSSSSS      DDDDDDDD      IIIIII      SSSSSSSS      KK      KK
MM      MM      AAAAAA      SSSSSSSS      SSSSSSSS      DDDDDDDD      IIIIII      SSSSSSSS      KK      KK
MMM     MMM     AA      AA      SS      SS      DD      DD      II      SS      KK      KK
MMM     MMM     AA      AA      SS      SS      DD      DD      II      SS      KK      KK
MM      MM      AA      AA      SS      SS      DD      DD      II      SS      KK      KK
MM      MM      AA      AA      SS      SS      DD      DD      II      SS      KK      KK
MM      MM      AA      AA      SS      SS      DD      DD      II      SS      KK      KK
MM      MM      AA      AA      SSSSSS      SSSSSS      DD      DD      II      SSSSSS      KKKKKK
MM      MM      AA      AA      SSSSSS      SSSSSS      DD      DD      II      SSSSSS      KKKKKK
MM      MM      AAAAAAAAAA      SS      SS      DD      DD      II      SS      KK      KK
MM      MM      AAAAAAAAAA      SS      SS      DD      DD      II      SS      KK      KK
MM      MM      AA      AA      SS      SS      DD      DD      II      SS      KK      KK
MM      MM      AA      AA      SSSSSSSS      SSSSSSSS      DDDDDDDD      IIIIII      SSSSSSSS      KK      KK
MM      MM      AA      AA      SSSSSSSS      SSSSSSSS      DDDDDDDD      IIIIII      SSSSSSSS      KK      KK

```

.....
.....
.....
.....

```

LL      IIIIII      SSSSSSSS
LL      IIIIII      SSSSSSSS
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
LLLLLLLLLLLL      IIIIII      SSSSSSSS
LLLLLLLLLLLL      IIIIII      SSSSSSSS

```

```
C
C Version:      'v04-000'
```

```

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*****

```

AUTHOR BRIAN PORTER CREATION DATE 19-FEB-1979

Functional description:

This module displays error log entries for all massbus disks.

Modified by:

V03-003 SAR0224 Sharon A. Reynolds, 28-Mar-1984
Changed the call to UCBSL_OWNER to ORBSL_OWNER.

V03-002 SAR0078 Sharon A. Reynolds, 20-Jun-1983
Changed the carriage control in the 'format' statements
for use with ERF.

V03-001 SAR0047 Sharon A. Reynolds, 9-Jun-1983
Removed brief/cryptic support.

v02-004 BP0004 Brian Porter, 16-NOV-1981
Added different mba handling.

v02-003 BP0003 Brian Porter, 06-NOV-1981
Added 'device attention' support. Corrected ac lo, dc lo
reversal in er3.

v02-002 BP0002 Brian Porter, 29-JUN-1981
Added call to LOGGER and DHEAD. Added DIAGNOSTIC_MODE.

C 8
16-Sep-1984 00:06:01
5-Sep-1984 14:00:29

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

Page 2

0058 C v02-001 BP0001 Brian Porter, 31-AUG-1980
0059 C Removed call to UNUSED_BITS. Changed format statements
0060 C so that 32 bit registers are printed.
0061 C**

0063 INCLUDE 'SRC\$:MSGHDR.FOR /NOLIST'
0122 INCLUDE 'SRC\$:DEVERR.FOR /NOLIST'

0223
0224
0225 BYTE LUN
0226
0227 integer*4 mba_control_register(7)
0228
0229 INTEGER*4 DRIVE_CS1
0230 INTEGER*4 DRIVE_DS
0231 INTEGER*4 DRIVE_ER1
0232 INTEGER*4 DRIVE_MR1
0233 INTEGER*4 DRIVE_AS
0234 INTEGER*4 DRIVE_LA
0235 INTEGER*4 DRIVE_DA
0236 INTEGER*4 DRIVE_DT
0237 INTEGER*4 DRIVE_SN
0238 INTEGER*4 DRIVE_OF
0239 INTEGER*4 DRIVE_DC
0240 INTEGER*4 RP_CC
0241 INTEGER*4 RM_ER2
0242 INTEGER*4 RP_ER2
0243 INTEGER*4 RM_MR2
0244 INTEGER*4 RP_ER3
0245 INTEGER*4 RM_HR
0246 INTEGER*4 DRIVE_EC1
0247 INTEGER*4 DRIVE_FC2
0248 INTEGER*4 DRIVE_FUNC
0249
0250 logical*1 diagnostic_mode
0251
0252 INTEGER*4 FUNCN(0:31)
0253 INTEGER*4 DESCR(2)
0254 INTEGER*4 FIELD
0255 INTEGER*4 COMPRESS4
0256 INTEGER*4 COMPRESSC
0257 integer*4 selected_mapping_register
0258
0259 EQUIVALENCE (RM03_DS(0), RP07_DS(0))
0260 EQUIVALENCE (mba_control_register, emb\$1_dv_reg\$av(0))
0261 EQUIVALENCE (DRIVE_CS1, emb\$1_dv_reg\$av(7))
0262 EQUIVALENCE (DRIVE_DS, emb\$1_dv_reg\$av(8))
0263 EQUIVALENCE (DRIVE_ER1, emb\$1_dv_reg\$av(9))
0264 EQUIVALENCE (DRIVE_MR1, emb\$1_dv_reg\$av(10))
0265 EQUIVALENCE (DRIVE_AS, emb\$1_dv_reg\$av(11))
0266 EQUIVALENCE (DRIVE_DA, emb\$1_dv_reg\$av(12))
0267 EQUIVALENCE (DRIVE_DT, emb\$1_dv_reg\$av(13))
0268 EQUIVALENCE (DRIVE_LA, emb\$1_dv_reg\$av(14))
0269 EQUIVALENCE (DRIVE_SN, emb\$1_dv_reg\$av(15))
0270 EQUIVALENCE (DRIVE_OF, emb\$1_dv_reg\$av(16))
0271 EQUIVALENCE (DRIVE_DC, emb\$1_dv_reg\$av(17))
0272 EQUIVALENCE (RP_CC, RM_HR, emb\$1_dv_reg\$av(18))

```
0273 EQUIVALENCE (RP_ER2, RM_MR2, EMB$$_DV_REGS$V(19))
0274 EQUIVALENCE (RP_ER3, RM_ER2, EMB$$_DV_REGS$V(20))
0275 EQUIVALENCE (DRIVE_EC1, EMB$$_DV_REGS$V(21))
0276 EQUIVALENCE (DRIVE_EC2, EMB$$_DV_REGS$V(22))
0277
0278 PARAMETER DT_RP04 = 16
0279 PARAMETER DT_RP05 = 17
0280 PARAMETER DT_RP06 = 18
0281 PARAMETER DT_RP07 = 34
0282 PARAMETER DT_RM03 = 20
0283 PARAMETER DT_RM05 = 23
0284 PARAMETER DT_RM80 = 22
0285 PARAMETER RM03 = 6
0286 PARAMETER RP06 = 5
0287 PARAMETER RP05 = 4
0288 PARAMETER RP04 = 3
0289 PARAMETER RP07 = 7
0290 PARAMETER RM80 = 13
0291 PARAMETER RM05 = 15
0292 PARAMETER XFER_CMD = 20
0293 PARAMETER TIMEOUT = 96
0294 PARAMETER NO_MOL = 1
0295
0296 CHARACTER*12 RM03_DS(0:0)
0297
0298 CHARACTER*7 MASS1_CS1(0:0)
0299 DATA MASS1_CS1(0) /'GO BIT*'/
0300
0301 CHARACTER*16 MASS2_CS1(11:11)
0302 DATA MASS2_CS1(11) /'DRIVE AVAILABLE*'/
0303
0304 CHARACTER*20 RP07_DS(0:2)
0305 DATA RP07_DS(0) /'OFFSET MODE*'/
0306 DATA RP07_DS(1) /'EARLY WARNING*'/
0307 DATA RP07_DS(2) /'INTERLEAVED SECTORS*'/
0308
0309 CHARACTER*24 MASS_DS(6:15)
0310 DATA MASS_DS(6) /'VOLUME VALID*'/
0311 DATA MASS_DS(7) /'DRIVE READY*'/
0312 DATA MASS_DS(8) /'DRIVE PRESENT*'/
0313 DATA MASS_DS(9) /'PROGRAMMABLE*'/
0314 DATA MASS_DS(10) /'LAST SECTOR TRANSFERRED*'/
0315 DATA MASS_DS(11) /'WRITE PROTECTED*'/
0316 DATA MASS_DS(12) /'MEDIUM ON-LINE*'/
0317 DATA MASS_DS(13) /'POSITIONING IN PROGRESS*'/
0318 DATA MASS_DS(14) /'COMPOSITE ERROR*'/
0319 DATA MASS_DS(15) /'ATTENTION ACTIVE*'/
0320
0321 CHARACTER*30 MASS_ER1(0:15)
0322 DATA MASS_ER1(0) /'ILLEGAL FUNCTION*'/
0323 DATA MASS_ER1(1) /'ILLEGAL REGISTER*'/
0324 DATA MASS_ER1(2) /'REGISTER MODIFICATION REFUSED*'/
0325 DATA MASS_ER1(3) /'PARITY ERROR*'/
0326 DATA MASS_ER1(4) /'FORMAT ERROR*'/
0327 DATA MASS_ER1(5) /'WRITE CLOCK FAIL*'/
0328 DATA MASS_ER1(6) /'ECC HARD ERROR*'/
0329 DATA MASS_ER1(7) /'HDR COMPARE ERROR*'/
```

```
0330 DATA MASS_ER1(8) /*HDR CRC ERROR*/
0331 DATA MASS_ER1(9) /*ADDRESS OVERFLOW ERROR*/
0332 DATA MASS_ER1(10) /*INVALID ADDRESS ERROR*/
0333 DATA MASS_ER1(11) /*WRITE LOCK ERROR*/
0334 DATA MASS_ER1(12) /*DRIVE TIMING ERROR*/
0335 DATA MASS_ER1(13) /*OPERATION INCOMPLETE*/
0336 DATA MASS_ER1(14) /*DRIVE UNSAFE*/
0337 DATA MASS_ER1(15) /*DATA CHECK*/
0338
0339 CHARACTER*29 V1RP056_ER2(0:11)
0340 DATA V1RP056_ER2(0) /*WRITE CURRENT UNSAFE*/
0341 DATA V1RP056_ER2(1) /*CURRENT SINK FAILURE*/
0342 DATA V1RP056_ER2(2) /*WRITE SELECT UNSAFE*/
0343 DATA V1RP056_ER2(3) /*CURRENT SWITCH UNSAFE*/
0344 DATA V1RP056_ER2(4) /*READ AND WRITE*/
0345 DATA V1RP056_ER2(5) /*TRANSITIONS DETECTED FAILURE*/
0346 DATA V1RP056_ER2(6) /*TRANSITIONS UNSAFE*/
0347 DATA V1RP056_ER2(7) /*ABNORMAL STOP*/
0348 DATA V1RP056_ER2(8) /*WRITE READY UNSAFE*/
0349 DATA V1RP056_ER2(9) /*MULTIPLE HEAD SELECT*/
0350 DATA V1RP056_ER2(10) /*NO HEAD SELECT*/
0351 DATA V1RP056_ER2(11) /*INDEX ERROR*/
0352
0353 CHARACTER*11 V2RP056_ER2(13:13)
0354 DATA V2RP056_ER2(13) /*PLO UNSAFE*/
0355
0356 CHARACTER*29 V1RP04_ER2(0:13)
0357 DATA V1RP04_ER2(0) /*WRITE CURRENT UNSAFE*/
0358 DATA V1RP04_ER2(1) /*CURRENT SINK FAILURE*/
0359 DATA V1RP04_ER2(2) /*WRITE SELECT UNSAFE*/
0360 DATA V1RP04_ER2(3) /*CURRENT SWITCH UNSAFE*/
0361 DATA V1RP04_ER2(4) /*MOTOR SEQUENCE ERROR*/
0362 DATA V1RP04_ER2(5) /*TRANSITIONS DETECTED FAILURE*/
0363 DATA V1RP04_ER2(6) /*TRANSITIONS UNSAFE*/
0364 DATA V1RP04_ER2(7) /*FAILSAFE ENABLE*/
0365 DATA V1RP04_ER2(8) /*WRITE READY UNSAFE*/
0366 DATA V1RP04_ER2(9) /*MULTIPLE HEAD SELECT*/
0367 DATA V1RP04_ER2(10) /*NO HEAD SELECT*/
0368 DATA V1RP04_ER2(11) /*INDEX ERROR*/
0369 DATA V1RP04_ER2(12) /*30V UNSAFE*/
0370 DATA V1RP04_ER2(13) /*PLO UNSAFE*/
0371
0372 CHARACTER*10 V2RP04_ER2(15:15)
0373 DATA V2RP04_ER2(15) /*AC UNSAFE*/
0374
0375 CHARACTER*21 V1RP07_ER2(8:13)
0376 DATA V1RP07_ER2(8) /*WRITE READY UNSAFE*/
0377 DATA V1RP07_ER2(9) /*WRITE OVERRUN*/
0378 DATA V1RP07_ER2(10) /*READ WRITE UNSAFE #1*/
0379 DATA V1RP07_ER2(11) /*READ WRITE UNSAFE #2*/
0380 DATA V1RP07_ER2(12) /*READ WRITE UNSAFE #3*/
0381 DATA V1RP07_ER2(13) /*8080 UNSAFE*/
0382
0383 CHARACTER*14 V2RP07_ER2(15:15)
0384 DATA V2RP07_ER2(15) /*PROGRAM ERROR*/
0385
0386 CHARACTER*17 V1RP056_ER3(0:1)
```

```
0387 DATA V1RP056_ER3(0) /*DC UNSAFE*/
0388 DATA V1RP056_ER3(1) /*WRITE AND OFFSET*/
0389
0390 CHARACTER*22 V2RP056_ER3(4:6)
0391 DATA V2RP056_ER3(4) /*35V REGULATOR FAILURE*/
0392 DATA V2RP056_ER3(5) /*DC LO*/
0393 DATA V2RP056_ER3(6) /*AC LO*/
0394
0395 CHARACTER*20 V3RP056_ER3(13:15)
0396 DATA V3RP056_ER3(13) /*OPERATOR PLUG ERROR*/
0397 DATA V3RP056_ER3(14) /*SEEK INCOMPLETE*/
0398 DATA V3RP056_ER3(15) /*OFF CYLINDER*/
0399
0400 CHARACTER*18 V1RP04_ER3(0:1)
0401 DATA V1RP04_ER3(0) /*PACK SPEED UNSAFE*/
0402 DATA V1RP04_ER3(1) /*VELOCITY UNSAFE*/
0403
0404 CHARACTER*22 V2RP04_ER3(3:3)
0405 DATA V2RP04_ER3(3) /*ANY UNSAFE EXCEPT R/W*/
0406
0407 CHARACTER*6 V3RP04_ER3(5:6)
0408 DATA V3RP04_ER3(5) /*DC LO*/
0409 DATA V3RP04_ER3(6) /*AC LO*/
0410
0411 CHARACTER*16 V4RP04_ER3(14:15)
0412 DATA V4RP04_ER3(14) /*SEEK INCOMPLETE*/
0413 DATA V4RP04_ER3(15) /*OFF CYLINDER*/
0414
0415 CHARACTER*23 V1RP07_ER3(0:11)
0416 DATA V1RP07_ER3(0) /*RUN TIMEOUT ERROR*/
0417 DATA V1RP07_ER3(1) /*SYNC CLOCK FAILURE*/
0418 DATA V1RP07_ER3(2) /*SYNC BYTE ERROR*/
0419 DATA V1RP07_ER3(3) /*DATA PARITY ERROR*/
0420 DATA V1RP07_ER3(4) /*SERDES DATA FAILURE*/
0421 DATA V1RP07_ER3(5) /*DC UNSAFE*/
0422 DATA V1RP07_ER3(6) /*INDEX UNSAFE*/
0423 DATA V1RP07_ER3(7) /*DEVICE CHECK*/
0424 DATA V1RP07_ER3(8) /*8080 HANDSHAKE FAILURE*/
0425 DATA V1RP07_ER3(9) /*LOSS OF CYLINDER ERROR*/
0426 DATA V1RP07_ER3(10) /*LOSS OF BIT CLOCK*/
0427 DATA V1RP07_ER3(11) /*CONTROL LOGIC FAILURE*/
0428
0429 CHARACTER*18 V2RP07_ER3(13:15)
0430 DATA V2RP07_ER3(13) /*DEFECT SKIP ERROR*/
0431 DATA V2RP07_ER3(14) /*SEEK INCOMPLETE*/
0432 DATA V2RP07_ER3(15) /*BAD SECTOR ERROR*/
0433
0434 CHARACTER*18 V1RM03_ER2(3:3)
0435 DATA V1RM03_ER2(3) /*DATA PARITY ERROR*/
0436
0437 CHARACTER*13 V2RM03_ER2(7:7)
0438 DATA V2RM03_ER2(7) /*DEVICE CHECK*/
0439
0440 CHARACTER*21 V3RM03_ER2(10:15)
0441 DATA V3RM03_ER2(10) /*LOSS OF BIT CLOCK*/
0442 DATA V3RM03_ER2(11) /*LOSS OF SYSTEM CLOCK*/
0443 DATA V3RM03_ER2(12) /*INVALID COMMAND*/
```

```
0444 DATA V3RM03_ER2(13) /*OPERATOR PLUG ERROR*/
0445 DATA V3RM03_ER2(14) /*SEEK INCOMPLETE*/
0446 DATA V3RM03_ER2(15) /*BAD SECTOR ERROR*/
0447
0448 CHARACTER*18 V1RM80_ER2(5:5)
0449 DATA V1RM80_ER2(5) /*SKIP SECTOR ERROR*/
0450
0451 CHARACTER*15 RP0456_LA(0:3)
0452 DATA RP0456_LA(0) /*FIRST QUARTER*/
0453 DATA RP0456_LA(1) /*SECOND QUARTER*/
0454 DATA RP0456_LA(2) /*THIRD QUARTER*/
0455 DATA RP0456_LA(3) /*FOURTH QUARTER*/
0456
0457 CHARACTER*29 V1RP0456_OF(7:8)
0458 DATA V1RP0456_OF(7) /*OFFSET DIRECTION BIT SET*/
0459 DATA V1RP0456_OF(8) /*DATA CHECK IN PROGRESS (VMS)*/
0460
0461 CHARACTER*12 V2RP0456_OF(15:15)
0462 DATA V2RP0456_OF(15) /*SIGN CHANGE*/
0463
0464 CHARACTER*23 MASS_OF(10:12)
0465 DATA MASS_OF(10) /*HDR COMPARE INHIBIT*/
0466 DATA MASS_OF(11) /*ECC INHIBIT*/
0467 DATA MASS_OF(12) /*16-BIT FORMAT*/
0468
0469 CHARACTER*5 RP0456_OF(1:6)
0470 DATA RP0456_OF(1) /*200*/
0471 DATA RP0456_OF(2) /*400*/
0472 DATA RP0456_OF(3) /*600*/
0473 DATA RP0456_OF(4) /*800*/
0474 DATA RP0456_OF(6) /*1200*/
0475
0476 CHARACTER*22 RP07_OF(14:15)
0477 DATA RP07_OF(14) /*MOVE TRACK DESCRIPTOR*/
0478 DATA RP07_OF(15) /*COMMAND MODIFIER BIT*/
0479
0480 CHARACTER*10 V1MASS_DT(11:11)
0481 DATA V1MASS_DT(11) /*DUAL PORT*/
0482
0483 CHARACTER*12 V2MASS_DT(13:13)
0484 DATA V2MASS_DT(13) /*MOVING HEAD*/
0485
0486 CHARACTER*8 OFFSET_DIR(0:1)
0487 DATA OFFSET_DIR(0) /*REVERSE*/
0488 DATA OFFSET_DIR(1) /*FORWARD*/
0489
0490 CHARACTER*25 V1RM03_OFF(7:7)
0491 DATA V1RM03_OFF(7) /*OFFSET DIRECTION BIT SET*/
0492
0493 CHARACTER*26 V1RM80_OFF(9:9)
0494 DATA V1RM80_OFF(9) /*SKIP SECTOR ERROR INHIBIT*/
0495
0496 CHARACTER*13 NO_OPERATION
0497 DATA NO_OPERATION /*NO-OPERATION*/
0498
0499 CHARACTER*7 UNLOAD
0500 DATA UNLOAD /*UNLOAD*/
```

```
0501  
0502 CHARACTER*5 SEEK  
0503 DATA SEEK /'SEEK*'/  
0504  
0505 CHARACTER*12 RECALIBRATE  
0506 DATA RECALIBRATE /'RECALIBRATE*'/  
0507  
0508 CHARACTER*12 DRIVE_CLEAR  
0509 DATA DRIVE_CLEAR /'DRIVE CLEAR*'/  
0510  
0511 CHARACTER*13 RELEASE_PORT  
0512 DATA RELEASE_PORT /'RELEASE PORT*'/  
0513  
0514 CHARACTER*7 OFFSET  
0515 DATA OFFSET /'OFFSET*'/  
0516  
0517 CHARACTER*21 RTZ  
0518 DATA RTZ /'RETURN TO CENTRELINE*'/  
0519  
0520 CHARACTER*15 READ_IN_PRESET  
0521 DATA READ_IN_PRESET /'READ IN PRESET*'/  
0522  
0523 CHARACTER*17 PACK_ACKNOWLEDGE  
0524 DATA PACK_ACKNOWLEDGE /'PACK ACKNOWLEDGE*'/  
0525  
0526 CHARACTER*7 SEARCH  
0527 DATA SEARCH /'SEARCH*'/  
0528  
0529 CHARACTER*17 WRITE_CHECKD  
0530 DATA WRITE_CHECKD /'WRITE CHECK DATA*'/  
0531  
0532 CHARACTER*28 WRITE_CHECKH  
0533 DATA WRITE_CHECKH /'WRITE CHECK HEADER AND DATA*'/  
0534  
0535 CHARACTER*11 WRITE_DATA  
0536 DATA WRITE_DATA /'WRITE DATA*'/  
0537  
0538 CHARACTER*22 WRITEH_DATA  
0539 DATA WRITEH_DATA /'WRITE HEADER AND DATA*'/  
0540  
0541 CHARACTER*10 READ_DATA  
0542 DATA READ_DATA /'READ DATA*'/  
0543  
0544 CHARACTER*21 READH_DATA  
0545 DATA READH_DATA /'READ HEADER AND DATA*'/  
0546  
0547 CHARACTER*17 ILLEGAL_FUNCTION  
0548 DATA ILLEGAL_FUNCTION /'ILLEGAL FUNCTION*'/  
0549  
0550 CHARACTER*11 DIAGNOSTIC  
0551 DATA DIAGNOSTIC /'DIAGNOSTIC*'/  
0552  
0553 CHARACTER*13 FORMAT_TRACK  
0554 DATA FORMAT_TRACK /'FORMAT TRACK*'/  
0555  
0556 CHARACTER*23 WRITE_TRACKD  
0557 DATA WRITE_TRACKD /'WRITE TRACK DESCRIPTOR*'/
```

```
0558  
0559 CHARACTER*22 READ_TRACKD  
0560 DATA READ_TRACKD /'READ TRACK DESCRIPTOR*'/  
0561  
0562 CHARACTER*11 DRIVE_TYPE  
0563 DATA DRIVE_TYPE /'DRIVE TYPE '/  
0564  
0565  
0566  
0567  
0568 CALL FRCTOF(LUN)  
0569  
0570 call dhead1 (lun,'MASSBUS')  
0571  
0572 DRIVE_FUNC=LIB$EXTZV(1,5,DRIVE_CS1)  
0573  
0574 if (  
0575 1 emb$w_dv_bcmt .ne. 0  
0576 1 .and.  
0577 1 drive_func .ge. xfer_cmd  
0578 1 .and.  
0579 1 emb$w_hd_entry .ne. 98  
0580 1 ) then  
0581  
0582 call mba_control_registers (lun,5,mba_control_register,  
0583 1 selected_mapping_register)  
0584  
0585 call mba_mapping_register (lun,selected_mapping_register,  
0586 1 mba_control_register(6))  
0587  
0588 if (selected_mapping_register .gt. 0) then  
0589  
0590 call mba_mapping_register (lun,(selected_mapping_register - 1),  
0591 1 mba_control_register(7))  
0592 endif  
0593 endif  
0594  
0595 DO 250,I = 0,31  
0596  
0597 FUNCN(I) = %LOC(ILLEGAL_FUNCTION)  
0598  
0599 250 CONTINUE  
0600  
0601 FUNCN(0) = %LOC(NO_OPERATION)  
0602  
0603 FUNCN(2) = %LOC(SEEK)  
0604  
0605 FUNCN(3) = %LOC(RECALIBRATE)  
0606  
0607 FUNCN(4) = %LOC(DRIVE_CLEAR)  
0608  
0609 FUNCN(5) = %LOC(RELEASE_PORT)  
0610  
0611 FUNCN(6) = %LOC(OFFSET)  
0612  
0613 FUNCN(7) = %LOC(RTZ)  
0614
```

```
0615      FUNCN(8) = %LOC(READ_IN_PRESET)
0616
0617      FUNCN(9) = %LOC(PACK_ACKNOWLEDGE)
0618
0619      FUNCN(12) = %LOC(SEARCH)
0620
0621      FUNCN(20) = %LOC(WRITE_CHECKD)
0622
0623      FUNCN(21) = %LOC(WRITE_CHECKH)
0624
0625      FUNCN(24) = %LOC(WRITE_DATA)
0626
0627      FUNCN(25) = %LOC(WRITEH_DATA)
0628
0629      FUNCN(28) = %LOC(READ_DATA)
0630
0631      FUNCN(29) = %LOC(READH_DATA)
0632
0633      diagnostic_mode = .false.
0634
0635      IF (EMB$B_DV_TYPE .EQ. RP04
0636 1 .OR.
0637 2 EMB$B_DV_TYPE .EQ. RP05
0638 3 .OR.
0639 4 EMB$B_DV_TYPE .EQ. RP06
0640 5 .OR.
0641 6 EMB$B_DV_TYPE .EQ. RP07) THEN
0642
0643      if (lib$extzv(0,1,drive_mr1) .eq. 1) diagnostic_mode = .true.
0644
0645      IF (EMB$B_DV_TYPE .EQ. RP07) THEN
0646
0647      diagnostic_mode = .false.
0648
0649      if (lib$extzv(15,1,drive_mr1) .eq. 1) diagnostic_mode = .true.
0650
0651      FUNCN(14) = %LOC(DIAGNOSTIC)
0652
0653      FUNCN(25) = %LOC(FORMAT_TRACK)
0654
0655      FUNCN(26) = %LOC(WRITE_TRACKD)
0656
0657      FUNCN(30) = %LOC(READ_TRACKD)
0658      ELSE
0659
0660      FUNCN(1) = %LOC(UNLOAD)
0661      ENDIF
0662
0663      C
0664      C      RP04/5/6/7 DISK DRIVE SUPPORT
0665      C
0666
0667      350      CALL LINCHK (LUN,2)
0668
0669      400      WRITE(LUN,400) DRIVE_CS1
0670      FORMAT('/',T8,'RPCS',T24,Z8.8)
0671
```

```
0672      if (.not. diagnostic_mode) then
0673
0674      call mba_status_register16_31 (lun,drive_cs1,drive_cs1,0)
0675
0676      CALL OUTPUT (LUN,DRIVE_CS1,MASS1_CS1,0,0,0,'0')
0677
0678      FIELD = LIB$EXTZV(1,5,DRIVE_CS1)
0679
0680      DESCR(2) = FUNCN(FIELD)
0681
0682      DESCR(1) = 33
0683
0684      DESCR(1) = LIB$LOCC('*',DESCR)
0685
0686      CALL OUTPUTD (LUN,DESCR)
0687
0688      CALL OUTPUT (LUN,DRIVE_CS1,MASS2_CS1,11,11,11,'0')
0689      endif
0690
0691      C
0692      C      'DS' REGISTER
0693      C
0694
0695      CALL LINCHK (LUN,1)
0696
0697      505 WRITE(LUN,505) DRIVE_DS
0698      FORMAT(' ',T8,'RPDS',T24,Z8.8)
0699
0700      if (.not. diagnostic_mode) then
0701
0702      call mba_status_register16_31 (lun,drive_cs1,drive_ds,1)
0703
0704      IF (EMBSB_DV_TYPE .EQ. RP07) THEN
0705
0706      CALL OUTPUT (LUN,DRIVE_DS,RP07_DS,0,0,2,'0')
0707      ENDIF
0708
0709      CALL OUTPUT (LUN,DRIVE_DS,MASS_DS,6,6,15,'0')
0710      endif
0711
0712      C
0713      C      'ER1' REGISTER
0714      C
0715
0716      CALL LINCHK (LUN,1)
0717
0718      520 WRITE(LUN,520) DRIVE_ER1
0719      FORMAT(' ',T8,'RPER1',T24,Z8.8)
0720
0721      if (.not. diagnostic_mode) then
0722
0723      call mba_status_register16_31 (lun,drive_ds,drive_er1,1)
0724
0725      CALL OUTPUT (LUN,DRIVE_ER1,MASS_ER1,0,0,15,'0')
0726      endif
0727
0728      C
```

```
0729 C      'MR' REGISTER
0730 C
0731
0732 CALL LINCHK (LUN,1)
0733
0734 525 WRITE(LUN,525) DRIVE_MR1
0735     FORMAT(' ',T8,'RPMR',T24,Z8.8)
0736
0737     if (.not. diagnostic_mode) then
0738
0739     call mba_status_register16_31 (lun,drive_er1,drive_mr1,1)
0740     else
0741
0742     CALL LINCHK (LUN,1)
0743
0744 526 WRITE(LUN,526)
0745     FORMAT(' ',T40,'DIAGNOSTIC MODE')
0746     endif
0747
0748 C      'AS' REGISTER
0749 C
0750 C
0751
0752 CALL LINCHK (LUN,1)
0753
0754 527 WRITE(LUN,527) DRIVE_AS
0755     FORMAT(' ',T8,'RPAS',T24,Z8.8)
0756
0757     if (.not. diagnostic_mode) then
0758
0759     call mba_status_register16_31 (lun,drive_mr1,drive_as,1)
0760
0761     DO 529 I=0,7
0762
0763     if (lib$extzv(i,1,drive_as) .eq. 1) then
0764
0765     CALL LINCHK (LUN,1)
0766
0767 528 WRITE(LUN,528) I
0768     FORMAT(' ',T40,'ATTENTION UNIT ',I1,'.')
0769     endif
0770
0771 529 CONTINUE
0772     endif
0773
0774 C      'DA' REGISTER
0775 C
0776 C
0777
0778 CALL LINCHK (LUN,1)
0779
0780 530 WRITE(LUN,530) DRIVE_DA
0781     FORMAT(' ',T8,'RPDA',T24,Z8.8)
0782
0783     if (.not. diagnostic_mode) then
0784
0785     call mba_status_register16_31 (lun,drive_as,drive_da,1)
```

```
0786  
0787 CALL LINCHK (LUN,2)  
0788  
0789 IF (EMBSB_DV_TYPE .EQ. RP07) THEN  
0790  
0791 FIELD=LIB$EXTZV(0,6,DRIVE_DA)  
0792 ELSE  
0793  
0794 FIELD=LIB$EXTZV(0,5,DRIVE_DA)  
0795 ENDIF  
0796  
0797 WRITE(LUN,535) FIELD  
0798 535 FORMAT(' ',T40,'SECTOR = ',I<COMPRESS4 (FIELD)>,'.')  
0799  
0800 FIELD=LIB$EXTZV(8,5,DRIVE_DA)  
0801  
0802 WRITE(LUN,537) FIELD  
0803 537 FORMAT(' ',T40,'TRACK = ',I<COMPRESS4 (FIELD)>,'.')  
0804 endif  
0805  
0806 C  
0807 C 'DT' REGISTER  
0808 C  
0809  
0810 CALL LINCHK (LUN,1)  
0811  
0812 WRITE(LUN,541) DRIVE_DT  
0813 541 FORMAT(' ',T8,'RPDT',T24,Z8.8)  
0814  
0815 if (.not. diagnostic_mode) then  
0816  
0817 call mba_status_register16_31 (lun,drive_da,drive_dt,1)  
0818  
0819 CALL LINCHK (LUN,1)  
0820  
0821 FIELD=LIB$EXTZV(0,9,DRIVE_DT)  
0822  
0823 IF (FIELD .EQ. DT_RP04) THEN  
0824  
0825 WRITE(LUN,544) DRIVE_TYPE  
0826 544 FORMAT(' ',T40,A11,'RP04')  
0827  
0828 ELSE IF (FIELD .EQ. DT_RP05) THEN  
0829  
0830 WRITE(LUN,545) DRIVE_TYPE  
0831 545 FORMAT(' ',T40,A11,'RP05')  
0832  
0833 ELSE IF (FIELD .EQ. DT_RP06) THEN  
0834  
0835 WRITE(LUN,546) DRIVE_TYPE  
0836 546 FORMAT(' ',T40,A11,'RP06')  
0837  
0838 ELSE IF (FIELD .EQ. DT_RP07) THEN  
0839  
0840 WRITE(LUN,547) DRIVE_TYPE  
0841 547 FORMAT(' ',T40,A11,'RP07')  
0842 ENDIF
```

```
0843
0844 554 CALL OUTPUT (LUN,DRIVE_DT,V1MASS_DT,11,11,11,'0')
0845
0846 CALL OUTPUT (LUN,DRIVE_DT,V2MASS_DT,13,13,13,'0')
0847 endif
0848
0849 C
0850 C
0851 C
0852
0853 CALL LINCHK (LUN,1)
0854
0855 WRITE(LUN,560) DRIVE_LA
0856 560 FORMAT(' ',T8,'RPLA',T24,Z8.8)
0857
0858 if (.not. diagnostic_mode) then
0859
0860 call mba_status_register16_31 (lun,drive_dt,drive_la,1)
0861
0862 IF (EMBSB_DV_TYPE .EQ. RP07) THEN
0863
0864 FIELD=LIB$EXTZV(5,7,DRIVE_LA)
0865 ELSE
0866
0867 FIELD=LIB$EXTZV(4,2,DRIVE_LA)
0868
0869 CALL LINCHK (LUN,1)
0870
0871 WRITE(LUN,563) RP0456 LA(FIELD)
0872 563 FORMAT(' ',T40,A<COMPRESSC (RP0456_LA(FIELD))>>)
0873
0874 FIELD=LIB$EXTZV(6,5,DRIVE_LA)
0875 ENDIF
0876
0877 CALL LINCHK (LUN,1)
0878
0879 WRITE(LUN,565) FIELD
0880 565 FORMAT(' ',T40,'SECTOR COUNTER = ',I<COMPRESS4 (FIELD)>,".")
0881 endif
0882
0883 C
0884 C
0885 C
0886
0887 CALL LINCHK (LUN,1)
0888
0889 WRITE(LUN,570) RP_ER2
0890 570 FORMAT(' ',T8,'RPER2',T24,Z8.8)
0891
0892 if (.not. diagnostic_mode) then
0893
0894 call mba_status_register16_31 (lun,drive_la,rp_er2,1)
0895
0896 IF (EMBSB_DV_TYPE .EQ. RP04) THEN
0897
0898 CALL OUTPUT (LUN,RP_ER2,V1RP04_ER2,0,0,13,'0')
0899
```

```
0900      CALL OUTPUT (LUN,RP_ER2,V2RP04_ER2,15,15,15,'0')
0901
0902      ELSE IF (
0903      1 EMB$B_DV_TYPE .EQ. RP05
0904      1 .OR.
0905      2 EMB$B_DV_TYPE .EQ. RP06
0906      1 ) THEN
0907
0908      CALL OUTPUT (LUN,RP_ER2,V1RP056_ER2,0,0,11,'0')
0909
0910      CALL OUTPUT (LUN,RP_ER2,V2RP056_ER2,13,13,13,'0')
0911
0912      ELSE IF (EMB$B_DV_TYPE .EQ. RP07) THEN
0913
0914      FIELD = LIB$EXTZV(0,8,RP_ER2)
0915
0916      IF (FIELD .NE. 0) THEN
0917
0918      CALL LINCHK (LUN,1)
0919
0920      575      WRITE(LUN,575) FIELD
0921      FORMAT(' ',T40,'ERROR CODE = ',Z2,' (HEX)')
0922      ENDIF
0923
0924      CALL OUTPUT (LUN,RP_ER2,V1RP07_ER2,8,8,13,'0')
0925
0926      CALL OUTPUT (LUN,RP_ER2,V2RP07_ER2,15,15,15,'0')
0927      ENDIF
0928      endif
0929
0930      C
0931      C      'OF' REGISTER
0932      C
0933
0934      CALL LINCHK (LUN,1)
0935
0936      580      WRITE(LUN,580) DRIVE_OF
0937      FORMAT(' ',T8,'RPOF',T24,Z8.8)
0938
0939      if (.not. diagnostic_mode) then
0940
0941      call mba_status_register16_31 (lun,rp_er2,drive_of,1)
0942
0943      IF (EMB$B_DV_TYPE .NE. RP07) THEN
0944
0945      FIELD=LIB$EXTZV(3,3,DRIVE_OF)
0946
0947      IF (FIELD .EQ. 0) THEN
0948
0949      CALL OUTPUT (LUN,DRIVE_OF,V1RP0456_OF,7,7,8,'0')
0950
0951      ELSE IF (
0952      1 ((FIELD .EQ. 2
0953      1 .OR.
0954      2 FIELD .EQ. 4
0955      3 .OR.
0956      4 FIELD .EQ. 6)
```

```
0957      5 .AND.  
0958      6 EMB$B_DV_TYPE .EQ. RP04)  
0959      7 .OR.  
0960      8 (FIELD .GE. 1  
0961      9 .AND.  
0962      1 FIELD .LE. 3  
0963      2 .AND.  
0964      3 (EMB$B_DV_TYPE .EQ. RP05  
0965      4 .OR.  
0966      5 EMB$B_DV_TYPE .EQ. RP06))  
0967      1 ) THEN  
0968  
0969      CALL LINCHK (LUN,1)  
0970  
0971      581 WRITE(LUN,581) RP0456_OF(FIELD)  
0972      FORMAT(' ',T40,'OFFSET = ',A<COMPRESSC (RP0456_OF(FIELD))>,  
0973      1 ' MICRO INCHES')  
0974  
0975      FIELD=LIB$EXTZV(7,1,DRIVE_OF)  
0976  
0977      CALL LINCHK (LUN,1)  
0978  
0979      582 WRITE(LUN,582) OFFSET_DIR(FIELD)  
0980      FORMAT(' ',T40,'OFFSET DIRECTION = ',  
0981      1 A<COMPRESSC (OFFSET_DIR(FIELD))>)  
0982  
0983      CALL OUTPUT (LUN,DRIVE_OF,V1RP0456_OF,7,8,8,'0')  
0984      ELSE  
0985  
0986      CALL OUTPUT (LUN,DRIVE_OF,V1RP0456_OF,7,8,8,'0')  
0987      ENDIF  
0988      ENDIF  
0989  
0990      CALL OUTPUT (LUN,DRIVE_OF,MASS_OF,10,10,12,'0')  
0991  
0992      IF (EMB$B_DV_TYPE .EQ. RP07) THEN  
0993  
0994      CALL OUTPUT (LUN,DRIVE_OF,RP07_OF,14,14,15,'0')  
0995      ELSE  
0996  
0997      CALL OUTPUT (LUN,DRIVE_OF,V2RP0456_OF,15,15,15,'0')  
0998      ENDIF  
0999      endif  
1000  
1001      C  
1002      C  
1003      C  
1004  
1005      CALL LINCHK (LUN,1)  
1006  
1007      590 WRITE(LUN,590) DRIVE_DC  
1008      FORMAT(' ',T8,'RPDC',T24,Z8.8)  
1009  
1010      if (.not. diagnostic_mode) then  
1011  
1012      call mba_status_register16_31 (lun,drive_of,drive_dc,1)  
1013
```

```
1014      IF (
1015      1 EMBSB_DV_TYPE .EQ. RP06
1016      1 .OR.
1017      1 EMBSB_DV_TYPE .EQ. RP07
1018      1 ) THEN
1019
1020      FIELD=LIB$EXTZV(0,10,DRIVE_DC)
1021      ELSE
1022
1023      FIELD=LIB$EXTZV(0,9,DRIVE_DC)
1024      ENDIF
1025
1026      CALL LINCHK (LUN,1)
1027
1028      WRITE(LUN,595) FIELD
1029 595      FORMAT(' ',T40,'DESIRED CYLINDER = ',I<COMPRESS4 (FIELD)>,'.')
1030      endif
1031
1032  C
1033  C      'CC' REGISTER
1034  C
1035
1036      CALL LINCHK (LUN,1)
1037
1038      WRITE(LUN,600) RP_CC
1039 600      FORMAT(' ',T8,'RPCC',T24,Z8.8)
1040
1041      if (.not. diagnostic_mode) then
1042
1043      call mba_status_register16_31 (lun,drive_dc,rp_cc,1)
1044
1045      IF (
1046      1 EMBSB_DV_TYPE .EQ. RP06
1047      1 .OR.
1048      1 EMBSB_DV_TYPE .EQ. RP07
1049      1 ) THEN
1050
1051      FIELD=LIB$EXTZV(0,10,RP_CC)
1052      ELSE
1053
1054      FIELD=LIB$EXTZV(0,9,RP_CC)
1055      ENDIF
1056
1057      CALL LINCHK (LUN,1)
1058
1059      WRITE(LUN,605) FIELD
1060 605      FORMAT(' ',T40,'CURRENT CYLINDER = ',I<COMPRESS4 (FIELD)>,'.')
1061      endif
1062
1063  C
1064  C      'SN' REGISTER
1065  C
1066
1067      CALL LINCHK (LUN,1)
1068
1069      WRITE(LUN,610) DRIVE_SN
1070 610      FORMAT(' ',T8,'RPSN',T24,Z8.8)
```

```

1071      if (.not. diagnostic_mode) then
1072
1073
1074      call mba_status_register16_31 (lun,rp_cc,drive_sn,1)
1075      endif
1076
1077      C
1078      C      'ER3' REGISTER
1079      C
1080
1081      614      CALL LINCHK (LUN,1)
1082
1083      WRITE(LUN,615) RP_ER3
1084      615      FORMAT(' ',T8,'RPER3',T24,Z8.8)
1085
1086      if (.not. diagnostic_mode) then
1087
1088      call mba_status_register16_31 (lun,drive_sn,rp_er3,1)
1089
1090      IF (EMBSB_DV_TYPE .EQ. RP04) THEN
1091
1092      CALL OUTPUT (LUN,RP_ER3,V1RP04_ER3,0,0,1,'0')
1093
1094      CALL OUTPUT (LUN,RP_ER3,V2RP04_ER3,3,3,3,'0')
1095
1096      CALL OUTPUT (LUN,RP_ER3,V3RP04_ER3,5,5,6,'0')
1097
1098      CALL OUTPUT (LUN,RP_ER3,V4RP04_ER3,14,14,15,'0')
1099
1100      ELSE IF (
1101      1 EMBSB_DV_TYPE .EQ. RP05
1102      1 .OR.
1103      1 EMBSB_DV_TYPE .EQ. RP06
1104      1 ) THEN
1105
1106      CALL OUTPUT (LUN,RP_ER3,V1RP056_ER3,0,0,1,'0')
1107
1108      CALL OUTPUT (LUN,RP_ER3,V2RP056_ER3,4,4,6,'0')
1109
1110      CALL OUTPUT (LUN,RP_ER3,V3RP056_ER3,13,13,15,'0')
1111
1112      ELSE IF (EMBSB_DV_TYPE .EQ. RP07) THEN
1113
1114      CALL OUTPUT (LUN,RP_ER3,V1RP07_ER3,0,0,11,'0')
1115
1116      CALL OUTPUT (LUN,RP_ER3,V2RP07_ER3,13,13,15,'0')
1117      ENDIF
1118      endif
1119
1120      C
1121      C      'EC1' REGISTER
1122      C
1123
1124      CALL LINCHK (LUN,1)
1125
1126      WRITE(LUN,645) DRIVE_EC1
1127      645      FORMAT(' ',T8,'RPEC1',T24,Z8.8)

```

```
1128
1129     if (.not. diagnostic_mode) then
1130
1131     call mba_status_register16_31 (lun,rp_er3,drive_ec1,1)
1132     endif
1133
1134     C
1135     C      'EC2' REGISTER
1136     C
1137
1138     CALL LINCHK (LUN,1)
1139
1140     WRITE(LUN,650) DRIVE_EC2
1141 650    FORMAT(' ',T8,'RPEC2',T24,Z8.8)
1142
1143     if (.not. diagnostic_mode) then
1144
1145     call mba_status_register16_31 (lun,drive_ec1,drive_ec2,1)
1146     endif
1147
1148     call linchk (lun,1)
1149
1150     write(lun,655)
1151 655    format(' ',:)
1152
1153     if (emb$w_hd_entry .ne. 98) then
1154
1155     call ucb$b_ertcnt (lun,emb$b_dv_ertcnt)
1156
1157     call ucb$b_ertmax (lun,emb$b_dv_ertmax)
1158     endif
1159
1160     call orb$l_owner (lun,emb$l_dv_ownuic)
1161
1162     call ucb$l_char (lun,emb$l_dv_char)
1163
1164     call ucb$w_sts (lun,emb$w_dv_sts)
1165
1166     call ucb$l_opcnt (lun,emb$l_dv_opcnt)
1167
1168     call ucb$w_errcnt (lun,emb$w_dv_errcnt)
1169
1170     if (emb$w_hd_entry .ne. 98) then
1171
1172     call ucb$l_media (lun,emb$l_dv_media)
1173
1174     call linchk (lun,1)
1175
1176     write(lun,655)
1177
1178     call dbdriver_qio (lun,emb$w_dv_func)
1179
1180     call irp$w_bcnc (lun,emb$w_dv_bcnc)
1181
1182     call irp$w_boff (lun,emb$w_dv_boff)
1183
1184     call irp$l_pid (lun,emb$l_dv_rqpid)
```

```
1185
1186      call irp$q_iosb (lun,emb$l_dv_iosb1)
1187    endif
1188
1189    C
1190    C      START OF RM03, RM05 AND RM80 DISK DRIVE SUPPORT
1191    C
1192
1193    ELSE IF (
1194      1 EMB$B_DV_TYPE .EQ. RM03
1195      1 .OR.
1196      2 EMB$B_DV_TYPE .EQ. RM05
1197      3 .OR.
1198      4 EMB$B_DV_TYPE .EQ. RM80
1199      1 ) THEN
1200
1201      if (lib$extzv(0,1,drive_mr1) .eq. 1) diagnostic_mode = .true.
1202
1203      CALL LINCHK (LUN,2)
1204
1205      1005 WRITE(LUN,1005) DRIVE_CS1
1206          FORMAT(/' ',T8,'RMCS',T24,Z8.8)
1207
1208      if (.not. diagnostic_mode) then
1209
1210      call mba_status_register16_31 (lun,drive_cs1,drive_cs1,0)
1211
1212      CALL OUTPUT (LUN,DRIVE_CS1,MASS1_CS1,0,0,0,'0')
1213
1214      FIELD = LIB$EXTZV(1,5,DRIVE_CS1)
1215
1216      DESCR(2) = FUNCN(FIELD)
1217
1218      DESCR(1) = 33
1219
1220      DESCR(1) = LIB$LOCC('* ',DESCR)
1221
1222      CALL OUTPUTD (LUN,DESCR)
1223
1224      CALL OUTPUT (LUN,DRIVE_CS1,MASS2_CS1,11,11,11,'0')
1225    endif
1226
1227    C
1228    C      'DS' REGISTER
1229    C
1230
1231      CALL LINCHK (LUN,1)
1232
1233      1030 WRITE(LUN,1030) DRIVE_DS
1234          FORMAT(' ',T8,'RMDS',T24,Z8.8)
1235
1236      if (.not. diagnostic_mode) then
1237
1238      call mba_status_register16_31 (lun,drive_cs1,drive_ds,1)
1239
1240      CALL OUTPUT (LUN,DRIVE_DS,RM03_DS,0,0,0,'0')
1241
```

```
1242      CALL OUTPUT (LUN,DRIVE_DS,MASS_DS,6,6,15,'0')
1243      endif
1244
1245      C
1246      C      'ER1' REGISTER
1247      C
1248
1249      CALL LINCHK (LUN,1)
1250
1251      1035 WRITE(LUN,1035) DRIVE_ER1
1252      FORMAT(' ',T8,'RMER1',T24,Z8.8)
1253
1254      if (.not. diagnostic_mode) then
1255
1256      call mba_status_register16_31 (lun,drive_ds,drive_er1,1)
1257
1258      CALL OUTPUT (LUN,DRIVE_ER1,MASS_ER1,0,0,15,'0')
1259      endif
1260
1261      C
1262      C      'MR' REGISTER
1263      C
1264
1265      CALL LINCHK (LUN,1)
1266
1267      1040 WRITE(LUN,1040) DRIVE_MR1
1268      FORMAT(' ',T8,'RMMR1',T24,Z8.8)
1269
1270      if (.not. diagnostic_mode) then
1271
1272      call mba_status_register16_31 (lun,drive_er1,drive_mr1,1)
1273      else
1274
1275      call linchk (lun,1)
1276
1277      1042 WRITE(LUN,1042)
1278      FORMAT(' ',T40,'DIAGNOSTIC MODE')
1279      endif
1280
1281      C
1282      C      'AS' REGISTER
1283      C
1284
1285      CALL LINCHK (LUN,1)
1286
1287      1045 WRITE(LUN,1045) DRIVE_AS
1288      FORMAT(' ',T8,'RMAS',T24,Z8.8)
1289
1290      if (.not. diagnostic_mode) then
1291
1292      call mba_status_register16_31 (lun,drive_mr1,drive_as,1)
1293
1294      DO 1048 I=0,7
1295
1296      if (lib$extzv(i,1,drive_as) .eq. 1) then
1297
1298      CALL LINCHK (LUN,1)
```

```
1299
1300      WRITE(LUN,1046) I
1301 1046  FORMAT(' ',T40,'ATTENTION UNIT ',I1,'.')
1302      endif
1303
1304 1048  CONTINUE
1305      endif
1306
1307  C
1308  C      'DA' REGISTER
1309  C
1310
1311      CALL LINCHK (LUN,1)
1312
1313      WRITE(LUN,1050) DRIVE_DA
1314 1050  FORMAT(' ',T8,'RMDA',T24,Z8.8)
1315
1316      if (.not. diagnostic_mode) then
1317
1318      call mba_status_register16_31 (lun,drive_as,drive_da,1)
1319
1320      CALL LINCHK (LUN,2)
1321
1322      FIELD=LIB$EXTZV(0,5,DRIVE_DA)
1323
1324 1055  WRITE(LUN,1055) FIELD
1325      FORMAT(' ',T40,'SECTOR = ',I<COMPRESS4 (FIELD)>,'.')
1326
1327      IF (EMBSB_DV_TYPE .EQ. RM03) THEN
1328
1329      FIELD = LIB$EXTZV(8,3,DRIVE_DA)
1330
1331      ELSE IF (EMBSB_DV_TYPE .EQ. RM80) THEN
1332
1333      FIELD = LIB$EXTZV(8,4,DRIVE_DA)
1334
1335      ELSE IF (EMBSB_DV_TYPE .EQ. RM05) THEN
1336
1337      FIELD = LIB$EXTZV(8,5,DRIVE_DA)
1338      ENDIF
1339
1340 1060  WRITE(LUN,1060) FIELD
1341      FORMAT(' ',T40,'TRACK = ',I<COMPRESS4 (FIELD)>,'.')
1342      endif
1343
1344  C
1345  C      'DT' REGISTER
1346  C
1347
1348      CALL LINCHK (LUN,1)
1349
1350      WRITE(LUN,1065) DRIVE_DT
1351 1065  FORMAT(' ',T8,'RMDT',T24,Z8.8)
1352
1353      if (.not. diagnostic_mode) then
1354
1355      call mba_status_register16_31 (lun,drive_da,drive_dt,1)
```

```
1356 FIELD=LIB$EXTZV(0,9,DRIVE_DT)
1357
1358 CALL LINCHK (LUN,1)
1359
1360 IF (FIELD .EQ. DT_RM03) THEN
1361
1362     1067 WRITE(LUN,1067) DRIVE_TYPE
1363         FORMAT(' ',T40,A11,'RM03')
1364
1365 ELSE IF (FIELD .EQ. DT_RM80) THEN
1366
1367     1068 WRITE(LUN,1068) DRIVE_TYPE
1368         FORMAT(' ',T40,A11,'RM80')
1369
1370 ELSE IF (FIELD .EQ. DT_RM05) THEN
1371
1372     1069 WRITE(LUN,1069) DRIVE_TYPE
1373         FORMAT(' ',T40,A11,'RM05')
1374
1375     ENDIF
1376
1377 CALL OUTPUT (LUN,DRIVE_DT,V1MASS_DT,11,11,11,'0')
1378
1379 CALL OUTPUT (LUN,DRIVE_DT,V2MASS_DT,13,13,13,'0')
1380 endif
1381
1382 C
1383 C 'LA' REGISTER
1384 C
1385
1386 CALL LINCHK (LUN,1)
1387
1388     1075 WRITE(LUN,1075) DRIVE_LA
1389         FORMAT(' ',T8,'RMLA',T24,Z8.8)
1390
1391     if (.not. diagnostic_mode) then
1392
1393         call mba_status_register16_31 (lun,drive_dt,drive_la,1)
1394
1395         FIELD=LIB$EXTZV(6,5,DRIVE_LA)
1396
1397         CALL LINCHK (LUN,1)
1398
1399     1080 WRITE(LUN,1080) FIELD
1400         FORMAT(' ',T40,'SECTOR COUNTER = ',I<COMPRESS4 (FIELD)>,'.')
1401
1402     endif
1403
1404 C
1405 C 'SN' REGISTER
1406 C
1407
1408 CALL LINCHK (LUN,1)
1409
1410     1085 WRITE(LUN,1085) DRIVE_SN
1411         FORMAT(' ',T8,'RMSN',T24,Z8.8)
1412
1413     if (.not. diagnostic_mode) then
```

```
1413
1414      call mba_status_register16_31 (lun,drive_la,drive_sn,1)
1415      endif
1416
1417      C
1418      C      'OF' REGISTER
1419      C
1420
1421      CALL LINCHK (LUN,1)
1422
1423      1090  WRITE(LUN,1090) DRIVE_OF
1424      FORMAT(' ',T8,'RMOF',T24,Z8.8)
1425
1426      if (.not. diagnostic_mode) then
1427
1428      call mba_status_register16_31 (lun,drive_sn,drive_of,1)
1429
1430      IF (JIAND(DRIVE_DS,1) .EQ. 1) THEN
1431
1432      FIELD=LIB$EXTZV(1,1,DRIVE_OF)
1433
1434      CALL LINCHK (LUN,1)
1435
1436      1095  WRITE(LUN,1095) OFFSET_DIR(FIELD)
1437      FORMAT(' ',T40,'OFFSET DIRECTION = ',
1438      1 A<COMPRESSC (OFFSET_DIR(FIELD))>>)
1439      ELSE
1440
1441      CALL OUTPUT (LUN,DRIVE_OF,V1RM03_OFF,7,7,7,'0')
1442      ENDIF
1443
1444      IF (EMBSB_DV_TYPE .EQ. RM80) THEN
1445
1446      CALL OUTPUT (LUN,DRIVE_OF,V1RM80_OFF,9,9,9,'0')
1447      ENDIF
1448
1449      CALL OUTPUT (LUN,DRIVE_OF,MASS_OF,10,10,12,'0')
1450      endif
1451
1452      C
1453      C      'DC' REGISTER
1454      C
1455
1456      CALL LINCHK (LUN,1)
1457
1458      1100  WRITE(LUN,1100) DRIVE_DC
1459      FORMAT(' ',T8,'RMDC',T24,Z8.8)
1460
1461      if (.not. diagnostic_mode) then
1462
1463      call mba_status_register16_31 (lun,drive_of,drive_dc,1)
1464
1465      FIELD=LIB$EXTZV(0,10,DRIVE_DC)
1466
1467      CALL LINCHK (LUN,1)
1468      WRITE(LUN,1105) FIELD
1469
```

```
1470 1105 FORMAT(' ',T40,'DESIRED CYLINDER = ',I<COMPRESS4 (FIELD)>,'.')
1471      endif
1472
1473      C
1474      C      'HR' REGISTER
1475      C
1476
1477      CALL LINCHK (LUN,1)
1478
1479      1110 WRITE(LUN,1110) RM_HR
1480      FORMAT(' ',T8,'RMHR',T24,Z8.8)
1481
1482      if (.not. diagnostic_mode) then
1483
1484      call mba_status_register16_31 (lun,drive_dc,rm_hr,1)
1485      endif
1486
1487      C
1488      C      'MR2' REGISTER
1489      C
1490
1491      CALL LINCHK (LUN,1)
1492
1493      1115 WRITE(LUN,1115) RM_MR2
1494      FORMAT(' ',T8,'RMMR2',T24,Z8.8)
1495
1496      if (.not. diagnostic_mode) then
1497
1498      call mba_status_register16_31 (lun,rm_hr,rm_mr2,1)
1499      endif
1500
1501      C
1502      C      'ER2' REGISTER
1503      C
1504
1505      CALL LINCHK (LUN,1)
1506
1507      1120 WRITE(LUN,1120) RM_ER2
1508      FORMAT(' ',T8,'RMER2',T24,Z8.8)
1509
1510      if (.not. diagnostic_mode) then
1511
1512      call mba_status_register16_31 (lun,rm_mr2,rm_er2,1)
1513
1514      CALL OUTPUT (LUN,RM_ER2,V1RM03_ER2,3,3,3,'0')
1515
1516      IF (EMBSB_DV_TYPE .EQ. RM80) THEN
1517
1518      CALL OUTPUT (LUN,RM_ER2,V1RM80_ER2,5,5,5,'0')
1519      ENDIF
1520
1521      CALL OUTPUT (LUN,RM_ER2,V2RM03_ER2,7,7,7,'0')
1522
1523      CALL OUTPUT (LUN,RM_ER2,V3RM03_ER2,10,10,15,'0')
1524      endif
1525
1526      C
```

```
1527 C      'EC1' REGISTER
1528 C
1529
1530      CALL LINCHK (LUN,1)
1531
1532      WRITE(LUN,1125) DRIVE_EC1
1533 1125  FORMAT(' ',T8,'RMEC1',T24,Z8.8)
1534
1535      if (.not. diagnostic_mode) then
1536
1537      call mba_status_register16_31 (lun,rm_er2,drive_ec1,1)
1538      endif
1539
1540 C
1541 C      'EC2' REGISTER
1542 C
1543
1544      CALL LINCHK (LUN,1)
1545
1546      WRITE(LUN,1130) DRIVE_EC2
1547 1130  FORMAT(' ',T8,'RMEC2',T24,Z8.8)
1548
1549      if (.not. diagnostic_mode) then
1550
1551      call mba_status_register16_31 (lun,drive_ec1,drive_ec2,1)
1552      endif
1553
1554      call linchk (lun,1)
1555
1556      write(lun,1135)
1557 1135  format(' ',:)
1558
1559      if (emb$w_hd_entry .ne. 98) then
1560
1561      call ucb$b_ertcnt (lun,emb$b_dv_ertcnt)
1562
1563      call ucb$b_ertmax (lun,emb$b_dv_ertmax)
1564      endif
1565
1566      call orb$l_owner (lun,emb$l_dv_ownuic)
1567
1568      call ucb$l_char (lun,emb$l_dv_char)
1569
1570      call ucb$w_sts (lun,emb$w_dv_sts)
1571
1572      call ucb$l_opcnt (lun,emb$l_dv_opcnt)
1573
1574      call ucb$w_errcnt (lun,emb$w_dv_errcnt)
1575
1576      if (emb$w_hd_entry .ne. 98) then
1577
1578      call ucb$l_media (lun,emb$l_dv_media)
1579
1580      call linchk (lun,1)
1581
1582      write(lun,1135)
1583
```

```
1584      call drdriver_qio (lun,emb$w_dv_func)
1585
1586      call irp$w_bcmt (lun,emb$w_dv_bcmt)
1587
1588      call irp$w_boff (lun,emb$w_dv_boff)
1589
1590      call irp$l_pid (lun,emb$l_dv_rapid)
1591
1592      call irp$q_iosb (lun,emb$l_dv_iosb1)
1593      endif
1594      ENDIF
1595
1596      RETURN
1597      END
```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	6065	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	1270	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	6896	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3 EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
Total Space Allocated	14743	

ENTRY POINTS

Address	Type	Name
0-00000000		MASSDISK

VARIABLES

Address	Type	Name	Address	Type	Name
2-00000CA7	CHAR	DIAGNOSTIC	2-00000BA7	L*1	DIAGNOSTIC_MODE
3-0000007E	I*4	DRIVE_AS	2-00000BCD	CHAR	DRIVE_CLEAR
3-0000006E	I*4	DRIVE_CS1	3-00000082	I*4	DRIVE_DA
3-00000096	I*4	DRIVE_DC	3-00000072	I*4	DRIVE_DS
3-00000086	I*4	DRIVE_DT	3-000000A6	I*4	DRIVE_EC1
3-000000AA	I*4	DRIVE_EC2	3-00000076	I*4	DRIVE_ER1
2-00000CF8	I*4	DRIVE_FUNC	3-0000008A	I*4	DRIVE_LA
3-0000007A	I*4	DRIVE_MR1	3-00000092	I*4	DRIVE_OF
3-0000008E	I*4	DRIVE_SN	2-00000CEC	CHAR	DRIVE_TYPE
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\$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$LD-HD-SID
3-00000024 I*2 EMB$W-DV-BCNT
3-0000002C I*2 EMB$W-DV-ERRCNT
3-0000001A I*2 EMB$W-DV-STS
3-00000004 I*2 EMB$W-HD-ENTRY
2-00000CFC I*4 FIELD
2-00000D04 I*4 I
AP-00000004 L*1 LUN
2-00000BE6 CHAR OFFSET
2-00000C81 CHAR READH_DATA
2-00000C02 CHAR READ-IN-PRESET
2-00000BC1 CHAR RECALIBRATE
3-000000A2 I*4 RM-ER2
3-0000009E I*4 RM-MR2
3-0000009E I*4 RP-ER2
2-00000BED CHAR RTZ
2-00000BBC CHAR SEEK
2-00000BB5 CHAR UNLOAD
2-00000C29 CHAR WRITE-CHECKD
2-00000C56 CHAR WRITE_DATA

```

```

3-0000003F CHAR EMB$T-DV-NAME
3-00000022 I*2 EMB$W-DV-BOFF
3-0000003C I*2 EMB$W-DV-FUNC
3-0000002A I*2 EMB$W-DV-UNIT
3-0000000E I*2 EMB$W-HD-ERRSEQ
2-00000CB2 CHAR FORMAT_TRACK
2-00000C96 CHAR ILLEGAL_FUNCTION
2-00000BA8 CHAR NO_OPERATION
2-00000C11 CHAR PACK_ACKNOWLEDGE
2-00000C77 CHAR READ_DATA
2-00000CD6 CHAR READ_TRACKD
2-00000BD9 CHAR RELEASE_PORT
3-0000009A I*4 RM-HR
3-0000009A I*4 RP-CC
3-000000A2 I*4 RP-ER3
2-00000C22 CHAR SEARCH
2-00000D00 I*4 SELECTED_MAPPING_REGISTER
2-00000C61 CHAR WRITEH_DATA
2-00000C3A CHAR WRITE_CHECKH
2-00000CBF CHAR WRITE_TRACKD

```

ARRAYS

Address	Type	Name	Bytes	Dimensions
2-000000BC	I*4	DESCR	8	(2)
3-00000000	L*1	EMB	512	(0:511)
3-00000052	I*4	EMB\$LD-DV-REGSAV	420	(0:104)
3-00000006	I*4	EMB\$Q-HD-TIME	8	(2)
2-0000003C	I*4	FUNCN	128	(0:31)
2-000000C4	CHAR	MASS1-CS1	7	(0:0)
2-000000CB	CHAR	MASS2-CS1	16	(11:11)
2-000000DB	CHAR	MASS-DS	240	(6:15)
2-000001CB	CHAR	MASS-ER1	480	(0:15)
2-00000ABF	CHAR	MASS-OF	69	(10:12)
3-00000052	I*4	MBA-CONTROL-REGISTER	28	(7)
2-00000B64	CHAR	OFFSET-DIR	16	(0:1)
2-00000000	CHAR	RM03-DS	12	(0:0)
2-00000A3D	CHAR	RP0456-LA	60	(0:3)
2-00000B04	CHAR	RP0456-OF	30	(6)
2-00000000	CHAR	RP07-DS	60	(0:2)
2-00000B22	CHAR	RP07-OF	44	(14:15)
2-00000B4E	CHAR	V1MASS-DT	10	(11:11)
2-0000098E	CHAR	V1RM03-ER2	18	(3:3)
2-00000B74	CHAR	V1RM03-OFF	25	(7:7)
2-00000A2B	CHAR	V1RM80-ER2	18	(5:5)
2-00000B8D	CHAR	V1RM80-OFF	26	(9:9)
2-00000A79	CHAR	V1RP0456-OF	58	(7:8)
2-00000512	CHAR	V1RP04-ER2	406	(0:13)
2-000007DE	CHAR	V1RP04-ER3	36	(0:1)
2-000003AB	CHAR	V1RP056-ER2	348	(0:11)
2-0000073E	CHAR	V1RP056-ER3	34	(0:1)
2-000006B2	CHAR	V1RP07-ER2	126	(8:13)
2-00000844	CHAR	V1RP07-ER3	276	(0:11)
2-00000B58	CHAR	V2MASS-DT	12	(13:13)
2-000009A0	CHAR	V2RM03-ER2	13	(7:7)

2-00000AB3	CHAR V2RP0456_OF	12	(15:15)
2-000006A8	CHAR V2RP04_ER2	10	(15:15)
2-00000802	CHAR V2RP04_ER3	22	(3:3)
2-00000507	CHAR V2RP056_ER2	11	(13:13)
2-00000760	CHAR V2RP056_ER3	66	(4:6)
2-00000730	CHAR V2RP07_ER2	14	(15:15)
2-00000958	CHAR V2RP07_ER3	54	(13:15)
2-000009AD	CHAR V3RM03_ER2	126	(10:15)
2-00000818	CHAR V3RP04_ER3	12	(5:6)
2-000007A2	CHAR V3RP056_ER3	60	(13:15)
2-00000824	CHAR V4RP04_ER3	32	(14:15)

LABELS

Address	Label	Address	Label	Address	Label	Address	Label	Address	Label	Address	Label
**	250	**	350	1-0000004C	400'	1-0000005E	505'	1-0000006F	520'	1-00000081	525'
1-00000092	526'	1-000000A9	527'	1-000000BA	528'	**	529	1-000000D6	530'	1-000000E7	535'
1-00000101	537'	1-0000011B	541'	1-0000012C	544'	1-0000013A	545'	1-00000148	546'	1-00000156	547'
**	554	1-00000164	560'	1-00000175	563'	1-00000181	565'	1-000001A3	570'	1-000001B5	575'
1-000001D4	580'	1-000001E5	581'	1-0000020B	582'	1-0000022C	590'	1-0000023D	595'	1-00000261	600'
1-00000272	605'	1-00000296	610'	**	614	1-000002A7	615'	1-000002B9	645'	1-000002CB	650'
1-000002DD	655'	1-000002E2	1005'	1-000002F4	1030'	1-00000305	1035'	1-00000317	1040'	1-00000329	1042'
1-00000340	1045'	1-00000351	1046'	**	1048	1-0000036D	1050'	1-0000037E	1055'	1-00000398	1060'
1-000003B2	1065'	1-000003C3	1067'	1-000003D1	1068'	1-000003DF	1069'	1-000003ED	1075'	1-000003FE	1080'
1-00000420	1085'	1-00000431	1090'	1-00000442	1095'	1-00000463	1100'	1-00000474	1105'	1-00000498	1110'
1-000004A9	1115'	1-000004BB	1120'	1-000004CD	1125'	1-000004DF	1130'	1-000004F1	1135'		

FUNCTIONS AND SUBROUTINES REFERENCED

Type Name

I*4 COMPRESS4
DHEAD1
IRPSL_PID
IRPSW_BOFF
LINCHR
MBA_STATUS_REGISTER16_31
OUTPUTD
UCBSL_CHAR
UCBSW_ERRCNT

Type Name

I*4 COMPRESSC
DRDRIVER_QIO
IRPSQ_IOSB
I*4 LIBSEXTZV
MBA_CONTROL_REGISTERS
ORBSL_OWNER
UCBSB_ERTCNT
UCBSL_MEDIA
UCBSW_STS

Type Name

DBDRIVER_QIO
FRCTOF
IRPSW_BCNT
I*4 LIB\$LOCC
MBA_MAPPING_REGISTER
OUTPUT
UCBSB_ERTMAX
UCBSL_OPCNT

```
0001
0002
0003
0004      Subroutine DBDRIVER_QIO (lun,emb$w_dv_func)
0005
0006
0007      include 'src$:qiocommon.for /nolist'
0008
0009
0010
0011      byte          lun
0012
0013      integer*2      emb$w_dv_func
0014
0015      integer*4      qiocode(0:1,0:63)
0016
0017
0018      if (qiocode(0,0) .eq. 0) then
0019
0020      qiocode(1,00) = %loc(io$_nop)
0021
0022      qiocode(1,01) = %loc(io$_unload)
0023
0024      qiocode(1,02) = %loc(io$_seek)
0025
0026      qiocode(1,03) = %loc(io$_recal)
0027
0028      qiocode(1,04) = %loc(io$_drvclr)
0029
0030      qiocode(1,05) = %loc(io$_release)
0031
0032      qiocode(1,06) = %loc(io$_offset)
0033
0034      qiocode(1,07) = %loc(io$_retcenter)
0035
0036      qiocode(1,08) = %loc(io$_packack)
0037
0038      qiocode(1,09) = %loc(io$_search)
0039
0040      qiocode(1,10) = %loc(io$_writecheck)
0041
0042      qiocode(1,11) = %loc(io$_writepblk)
0043
0044      qiocode(1,12) = %loc(io$_readpblk)
0045
0046      qiocode(1,13) = %loc(io$_writehead)
0047
0048      qiocode(1,14) = %loc(io$_readhead)
0049
0050      qiocode(1,24) = %loc(io$_writecheckh)
0051
0052      qiocode(1,25) = %loc(io$_readpreset)
0053
0054      qiocode(1,26) = %loc(io$_setchar)
0055
0056      qiocode(1,27) = %loc(io$_sensechar)
0057
0058      qiocode(1,32) = %loc(io$_writelblk)
0059
0060
```

```
0321      qiocode(1,33) = %loc(io$_readblk)
0322
0323      qiocode(1,35) = %loc(io$_setmode)
0324
0325      qiocode(1,39) = %loc(io$_sensemode)
0326
0327      qiocode(1,48) = %loc(io$_writevblk)
0328
0329      qiocode(1,49) = %loc(io$_readvblk)
0330
0331      qiocode(1,50) = %loc(io$_access)
0332
0333      qiocode(1,51) = %loc(io$_create)
0334
0335      qiocode(1,52) = %loc(io$_deaccess)
0336
0337      qiocode(1,53) = %loc(io$_delete)
0338
0339      qiocode(1,54) = %loc(io$_modify)
0340
0341      qiocode(1,56) = %loc(io$_acpcontrol)
0342
0343      qiocode(1,57) = %loc(io$_mount)
0344
0345      do 10,i = 0,63
0346
0347      qiocode(0,i) = 33
0348
0349      if (qiocode(1,i) .eq. 0) then
0350
0351      qiocode(1,i) = %loc(qio_string)
0352      endif
0353
0354      10 continue
0355      endif
0356
0357      call irp$w_func (lun,emb$w_dv_func,
0358      1 qiocode(0,lib$extzv(0,6,emb$w_dv_func)))
0359
0360      return
0361
0362      end
0363
```

PROGRAM SECTIONS

Name	Bytes	Attributes								
0 \$CODE	309	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 \$IOCOMMON	1247	PIC	OVR	REL	GBL	SHR	NOEXE	RD	WRT	LONG
Total Space Allocated		2112								

ENTRY POINTS

Address	Type	Name
0-00000000		DBDRIVER_Q10

VARIABLES

Address	Type	Name	Address	Type	Name
AP-00000008a	I*2	EMBSW DV FUNC	2-00000200	I*4	I
3-00000442	CHAR	IOS_ABORT	3-00000340	CHAR	IOS_ACCESS
3-000003C2	CHAR	IOS_ACPCONTROL	3-000004B3	CHAR	IOS_AVAILABLE
3-00000297	CHAR	IOS_CLEAN	3-00000369	CHAR	IOS_CREATE
3-00000385	CHAR	IOS_DEACCESS	3-00000393	CHAR	IOS_DELETE
3-0000026D	CHAR	IOS_DIAGNOSE	3-00000065	CHAR	IOS_DRVCLR
3-000004CB	CHAR	IOS_DSE	3-000000A9	CHAR	IOS_ERASETAPE
3-00000276	CHAR	IOS_FORMAT	3-00000071	CHAR	IOS_INITIALIZE
3-00000014	CHAR	IOS_LOADMCODE	3-000003A1	CHAR	IOS_MODIFY
3-000003E2	CHAR	IOS_MOUNT	3-00000000	CHAR	IOS_NOP
3-0000009D	CHAR	IOS_OFFSET	3-000000EB	CHAR	IOS_PACKACK
3-000000E0	CHAR	IOS_QSTOP	3-000003EF	CHAR	IOS_RDSTATS
3-00000421	CHAR	IOS_READCSR	3-00000169	CHAR	IOS_READHEAD
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
3-00000037	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

DBDRIVER_QIO

6 10
16-Sep-1984 00:06:01
5-Sep-1984 14:00:29

VAX-11 FORTRAN V3.4-56
DISK\$VMSMASTER:[ERF.SRC]MASSDISK.FOR;1 Page 32

3-0000017E CHAR IOS_WRIETRACKD
3-00000448 CHAR IOS_WRITEWITHBUF
AP-00000004 L*1 LUN

3-00000326 CHAR IOS_WRITEVBLK
3-00000257 CHAR IOS_WRIETMKR
3-000004A1 CHAR QIO_STRING

ARRAYS

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

LABELS

Address	Label
**	10

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name
	IRPSW_FUNC	1*4	LIB\$EXTZV

```
0001
0002
0003
0004      subroutine drdriver_qio (lun,emb$w_dv_func)
0005
0006
0007
0008      include 'src$:qiocommon.for /nolist'
0009
0010
0011
0012
0013
0014
0015
0016
0017
0018
0019
0020
0021
0022
0023
0024
0025      byte          lun
0026
0027      integer*2      emb$w_dv_func
0028
0029      integer*4      qiocode(0:1,0:63)
0030
0031
0032
0033
0034      if (qiocode(0,0) .eq. 0) then
0035
0036      qiocode(1,00) = %loc(io$_nop)
0037
0038      qiocode(1,02) = %loc(io$_seek)
0039
0040      qiocode(1,03) = %loc(io$_recal)
0041
0042      qiocode(1,04) = %loc(io$_drvclr)
0043
0044      qiocode(1,05) = %loc(io$_release)
0045
0046      qiocode(1,06) = %loc(io$_offset)
0047
0048      qiocode(1,07) = %loc(io$_retcenter)
0049
0050      qiocode(1,08) = %loc(io$_packack)
0051
0052      qiocode(1,09) = %loc(io$_search)
0053
0054      qiocode(1,10) = %loc(io$_writecheck)
0055
0056      qiocode(1,11) = %loc(io$_writepblk)
0057
0058      qiocode(1,12) = %loc(io$_readpblk)
0059
0060      qiocode(1,13) = %loc(io$_writehead)
0061
0062      qiocode(1,14) = %loc(io$_readhead)
0063
0064      qiocode(1,15) = %loc(io$_writetrackd)
0065
0066      qiocode(1,16) = %loc(io$_readtrackd)
0067
0068      qiocode(1,24) = %loc(io$_writecheckh)
0069
0070      qiocode(1,25) = %loc(io$_readpreset)
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
```

```
0321      qiocode(1,26) = %loc(io$_setchar)
0322
0323      qiocode(1,27) = %loc(io$_sensechar)
0324
0325      qiocode(1,29) = %loc(io$_diagnose)
0326
0327      qiocode(1,32) = %loc(io$_writelblk)
0328
0329      qiocode(1,33) = %loc(io$_readlblk)
0330
0331      qiocode(1,48) = %loc(io$_writevblk)
0332
0333      qiocode(1,49) = %loc(io$_readvblk)
0334
0335      qiocode(1,50) = %loc(io$_access)
0336
0337      qiocode(1,51) = %loc(io$_create)
0338
0339      qiocode(1,52) = %loc(io$_deaccess)
0340
0341      qiocode(1,53) = %loc(io$_delete)
0342
0343      qiocode(1,54) = %loc(io$_modify)
0344
0345      qiocode(1,56) = %loc(io$_acpcontrol)
0346
0347      qiocode(1,57) = %loc(io$_mount)
0348
0349      do 10,i = 0,63
0350
0351      qiocode(0,i) = 33
0352
0353      if (qiocode(1,i) .eq. 0) then
0354
0355      qiocode(1,i) = %loc(qio_string)
0356      endif
0357
0358      10 continue
0359      endif
0360
0361      call irp$w_func (lun,emb$w_dv_func,
0362      1 qiocode(0,lib$extzv(0,6,emb$w_dv_func)))
0363
0364      return
0365
0366      end
0367
```

PROGRAM SECTIONS

Name	Bytes	Attributes									
0 \$CODE	310	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		2113									

ENTRY POINTS

Address	Type	Name
0-00000000		DRDRIVER_QIO

VARIABLES

Address	Type	Name	Address	Type	Name
AP-00000008a	I*2	EMBSW DV FUNC	2-00000200	I*4	I
3-00000442	CHAR	IOS_ABORT	3-0000034D	CHAR	IOS_ACCESS
3-000003C2	CHAR	IOS_ACPCONTROL	3-000004B3	CHAR	IOS_AVAILABLE
3-00000297	CHAR	IOS_CLEAN	3-00000369	CHAR	IOS_CREATE
3-00000385	CHAR	IOS_DEACCESS	3-00000393	CHAR	IOS_DELETE
3-0000026D	CHAR	IOS_DIAGNOSE	3-00000065	CHAR	IOS_DRVCLR
3-000004CB	CHAR	IOS_DSE	3-000000A9	CHAR	IOS_ERASETAPE
3-00000276	CHAR	IOS_FORMAT	3-00000071	CHAR	IOS_INITIALIZE
3-00000014	CHAR	IOS_LOADMCODE	3-000003A1	CHAR	IOS_MODIFY
3-000003E2	CHAR	IOS_MOUNT	3-00000000	CHAR	IOS_NOP
3-0000009D	CHAR	IOS_OFFSET	3-000000EB	CHAR	IOS_PACKACK
3-000000E0	CHAR	IOS_QSTOP	3-000003EF	CHAR	IOS_RDSTATS
3-00000421	CHAR	IOS_READCSR	3-00000169	CHAR	IOS_READHEAD
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
3-00000037	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_WRITEBLK
3-00000247	CHAR	IOS_WRITEMARK	3-00000314	CHAR	IOS_WRITEOF
3-0000012A	CHAR	IOS_WRITEPBLK	3-000001C9	CHAR	IOS_WRITERET

DRDRIVER_QIO

K 10
16-Sep-1984 00:06:01
5-Sep-1984 14:00:29

VAX-11 FORTRAN V3.4-56
DISK\$VMSMASTER:[ERF.SRC]MASSDISK.FOR;1 Page 36

3-0000017E CHAR IOS_WRIETRACKD
3-00000448 CHAR IOS_WRIETWTHBUF
AP-00000004a L*1 LUN

3-00000326 CHAR IOS_WRITEVBLK
3-00000257 CHAR IOS_WRTTMKR
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=LISS:MASSDISK/OBJ=OBJ\$:MASSDISK MSRC\$:MASSDISK

/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:	22.69 seconds
Elapsed Time:	49.69 seconds
Page Faults:	366
Dynamic Memory:	361 pages

