

[illegible][illegible]

[illegible]



(1) 156

START I/O OPERATION

```
0000 1      .TITLE DXPDRIVER - VAX-11/780 RX01 CONSOLE DRIVER
0000 2      .IDENT 'V04-000'
0000 3
0000 4
0000 5 *****
0000 6
0000 7 *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 *  ALL RIGHTS RESERVED.
0000 10
0000 11 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 *  TRANSFERRED.
0000 17
0000 18 *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 *  CORPORATION.
0000 21
0000 22 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24
0000 25 *****
0000 26
0000 27
0000 28 C. A. MONIA 13-JUL-77
0000 29
0000 30 MODIFIED BY:
0000 31
0000 32 V03-001 RAS0300 Ron Schaefer 19-Jun-1984
0000 33 Add DEV$M_NNM characteristic to DECHAR2 so that these
0000 34 devices will have the 'node$' prefix.
0000 35
0000 36 V02-005 ACG0179 Andrew C. Goldstein, 23-Jul-1980 18:32
0000 37 Fix ACP class code in DPT
0000 38
0000 39 **
0000 40
0000 41
0000 42 STAR CONSOLE FLOPPY DISK DRIVER
0000 43
0000 44 MACRO LIBRARY CALLS
0000 45
0000 46
0000 47 $ADPDEF ;DEFINE ADP OFFSETS
0000 48 $CRBDEF ;DEFINE CRB OFFSETS
0000 49 $DDBDEF ;DEFINE DDB OFFSETS
0000 50 $DPTDEF ;DEFINE DPT OFFSETS
0000 51 $DYNDEF ;DEFINE DATA STRUCTURE TYPES
0000 52 $EMBDEF ;DEFINE EMB OFFSETS
0000 53 $IDBDEF ;DEFINE IDB OFFSETS
0000 54 $IODEF ;DEFINE I/O FUNCTION CODES
0000 55 $IRPDEF ;DEFINE IRP OFFSETS
0000 56 $UBADEF ;DEFINE UBA REGISTER OFFSETS
0000 57 $UCBDEF ;DEFINE UCB OFFSETS
```



```
0000 58          $VECDEF                      ;DEFINE INTERRUPT DISPATCH VECTOR OFFSETS
0000 59
0000 60 :
0000 61 : LOCAL SYMBOLS
0000 62 :
0000 63 : CONSOLE FLOPPY STATUS BIT DEFINITIONS
0000 64 :
0000 65
0000 66          VIELD DXP_RXDB,0,<-          ;RECEIVER DATA BUFFER FIELD DEFINITIONS
0000 67          ZCRC,,M>,-                  ;CRC ERROR
0000 68          <PAR,,M>,-                  ;PARITY
0000 69          <INI,,M>-                    ;INIT. COMPLETE
0000 70 >
0000 71          VIELD DXP_RXDB,6,<-          ;
0000 72          ZDEL,,M>,-                  ;DELETED DATA MARK READ
0000 73          <ERR,,M>-                    ;ERROR DETECTED
0000 74 >
0000 75
0000 76 :
0000 77 : CONSOLE COMMAND FIELD DEFINITION
0000 78 :
0000 79
00000800 0000 80 DXP_M_CMD = 808             ;COMMAND BIT
0000 81
0000 82 :
0000 83 : CONSOLE DATA BUFFER VALUES
0000 84 :
0000 85
00000900 0000 86 DXP_RXDB_K_CMD = 908         ;INITIATE FLOPPY FUNCTION
00000200 0000 87 DXP_RXDB_K_FCMP = 208        ;FLOPPY FUNCTION COMPLETE
00000005 0000 88 DXP_RXDB_K_PRTC = 5          ;PROTOCOL ERROR STATUS
0000 89
0000 90 :
0000 91 : HARDWARE FUNCTION CODES
0000 92 :
0000 93
00000800 0000 94 F_READSECTOR=808!0          ;READ SECTOR
00000801 0000 95 F_WRITESECTOR=808!1        ;WRITE SECTOR
00000802 0000 96 F_READSTATUS=808!2         ;READ STATUS
00000803 0000 97 F_WRITEDELETEDATA=808!3    ;WRITE DELETED DATA SECTOR
00000804 0000 98 F_CANCEL=808!4             ;CANCEL FLOPPY FUNCTION
0000 99
0000 100 :
0000 101 : LOCAL DATA
0000 102 :
0000 103 : DRIVER PROLOGUE TABLE
0000 104 :
0000 105
0000 106 DXP$DPT::
0000 107          DPTAB -                      ;DEFINE DRIVER PROLOGUE TABLE
0000 108          END=DXP_END,-                  ;END OF DRIVER
0000 109          ADAPTER=NULL,-                ;ADAPTER TYPE
0000 110          FLAGS=DPT$M_SVP,-            ;SYSTEM PAGE TABLE ENTRY REQUIRED
0000 111          UCBSIZE=UCB$B_DX_SCTCNT+2,-  ;UCB SIZE
0000 112          NAME=DXDRIVER                  ;DRIVER NAME
0038 113          DPT_STORE INIT              ;CONTROL BLOCK INIT VALUES
0038 114          DPT_STORE DDB,DB$$_ACPD,,L,<^A\F11> ;DEFAULT ACP NAME
```



```
003F 115 DPT_STORE DDB, DDB$$_ACPD+3, B, DDB$$_SLOW ; ACP CLASS
0043 116 DPT_STORE UCB, UCB$$_FIPL, B, 8 ; FORK IPL
0047 117 DPT_STORE UCB, UCB$$_DEVCHAR, L, - ; DEVICE CHARACTERISTICS
0047 118 <DEV$$_FOD- ; FILES ORIENTED
0047 119 :DEV$$_DIR- ; DIRECTORY STRUCTURED
0047 120 :DEV$$_AVL- ; AVAILABLE
0047 121 :DEV$$_SHR- ; SHAREABLE
0047 122 :DEV$$_IDV- ; INPUT DEVICE
0047 123 :DEV$$_ODV- ; OUTPUT DEVICE
0047 124 :DEV$$_RND> ; RANDOM ACCESS
004E 125 DPT_STORE UCB, UCB$$_DEVCHAR2, L, - ; DEVICE CHARACTERISTICS
004E 126 <DEV$$_NM> ; PREFIX NAME WITH 'node$'
0055 127 DPT_STORE UCB, UCB$$_DEVCLASS, B, DC$_DISK ; DEVICE CLASS
0059 128 DPT_STORE UCB, UCB$$_DEVTYPE, B, DT$_RX01 ; DEVICE TYPE
005D 129 DPT_STORE UCB, UCB$$_DEVBUFSIZ, W, 512 ; DEFAULT BUFFER SIZE
0062 130 DPT_STORE UCB, UCB$$_SECTORS, B, 26 ; NUMBER OF SECTORS PER TRACK
0066 131 DPT_STORE UCB, UCB$$_TRACKS, B, 1 ; NUMBER OF TRACKS PER CYLINDER
006A 132 DPT_STORE UCB, UCB$$_CYLINDERS, W, 77 ; NUMBER OF CYLINDERS
006F 133 DPT_STORE UCB, UCB$$_DIPL, B, 20 ; DEVICE IPL
0073 134 DPT_STORE UCB, UCB$$_DEVSTS, W, UCB$$_NOCNVRT ; NO LBN TO MEDIA ADDR. CONV.
0078 135 DPT_STORE UCB, UCB$$_ERTCNT, B, 8 ; ERROR RETRY COUNT
007C 136 DPT_STORE UCB, UCB$$_ERTMAX, B, 8 ; MAX ERROR RETRY COUNT
0080 137 DPT_STORE UCB, UCB$$_MAXBLOCK, L, <<76*26>/4> ; MAX. NUMBER OF BLOCKS
0087 138 DPT_STORE REINIT ; CONTROL BLOCK RE-INIT VALUES
0087 139 DPT_STORE DDB, DDB$$_DDT, D, DXP$$_DDT ; DDT ADDRESS
008C 140 DPT_STORE END ;
0000 141
0000 142 :
0000 143 : DRIVER DISPATCH TABLE
0000 144 :
0000 145
0000 146 DDTAB DXP, - ; DRIVER DISPATCH TABLE
0000 147 STARTIO, - ; INITIATE I/O OPERATION
0000 148 UNSOLNT, - ; UNSOLICITED INTERRUPT
0000 149 DX$$_FUNCTABLE, - ; FUNCTION DECISION TABLE
0000 150 UNSOLNT, - ; DEASSIGN PROCESSING ROUTINE
0000 151 REGDUMP, - ; REGISTER DUMP ROUTINE
0000 152 <<6>*4+<3>*4>, - ; SIZE OF DIAGNOSTIC BUFFER
0000 153 0, - ; SIZE OF ERROR BUFFER (0=NONE)
0000 154 DX$$_UNITINIT ; UNIT INITIALIZATION
```



```
0038 156 .SBTTL START I/O OPERATION
0038 157
0038 158
0038 159 :+ STARTIO - START I/O OPERATION ON DEVICE UNIT
0038 160
0038 161 : THIS ENTRY POINT IS ENTERED TO START AN I/O OPERATION ON THE CONSOLE
0038 162 : FLOPPY. CONTROL ALTERNATES BETWEEN THE FLOPPY DRIVER AND THE FLOPPY
0038 163 : UTILITY ROUTINES IN DXUTILITY THAT ARE SHARED BY ALL FLOPPY DEVICES.
0038 164 : THE DRIVER PERFORMS ALL DEVICE-DEPENDANT PROCESSING WHILE THE UTILI-
0038 165 : TY ROUTINES HANDLE THE INTERFACE BETWEEN THE DRIVE AND THE SYSTEM OR
0038 166 : USER. THE PROTOCOL FOR EACH DATA TRANSFER IS AS FOLLOWS:
0038 167
0038 168 : 1. THE SYSTEM DETERMINES THE LEGALITY OF A FUNCTION AND ANY PRE-
0038 169 : PROCESSING TO BE PERFORMED BY SCANNING THE COMMON FUNCTION DE-
0038 170 : CISION TABLE.
0038 171
0038 172 : 2. A REQUEST IS DEQUEUED AND THE DRIVER IS ENTERED AT ITS START-I/O
0038 173 : ENTRY POINT.
0038 174
0038 175 : 3. A SUBROUTINE CALL TO COMMON CODE IS EXECUTED TO COMPUTE THE INI-
0038 176 : TIAL MEDIA ADDRESS AND SETUP THE I/O DATA BASE FOR THE TRANSFER.
0038 177
0038 178 : 4. A CO-ROUTINE CALL TO THE DRIVER IS EXECUTED TO POSITION THE MEDIA
0038 179
0038 180 : 5. A CO-ROUTINE CALL TO THE DRIVER IS PERFORMED TO TRANSFER ONE BYTE
0038 181 : OF DATA
0038 182
0038 183 : 6. A CO-ROUTINE CALL TO THE COMMON PROCESSOR IS PERFORMED TO UPDATE
0038 184 : THE ADDRESS AND BYTE COUNT AND CHECK FOR ENDPOINT CONDITIONS
0038 185
0038 186 : STEPS 4 - 6 ARE EXECUTED AT HARDWARE IPL LEVEL. A TRANSFER TO FORK LEVEL
0038 187 : MUST BE EXECUTED WHENEVER:
0038 188
0038 189 : . ONE SECTOR OF DATA HAS BEEN TRANSFERRED
0038 190
0038 191 : . AN ERROR CONDITION IS DETECTED.
0038 192
0038 193 : ON OCCURANCE OF AN ERROR, THE DRIVER PERFORMS ANY HARDWARE DEPENDANT
0038 194 : FUNCTIONS THEN TRANSFERS CONTROL TO THE COMMON EXCEPTION ENTRY POINT
0038 195 : TO RESET MEDIA AND USER ADDRESSES AND REISSUE THE REQUEST.
0038 196
0038 197 : THE FLAGS DESCRIBED BELOW ARE USED TO SIGNAL EXCEPTION CONDITIONS:
0038 198
0038 199 : FLAGS SET BY DRIVER:
0038 200
0038 201 : WHEN EXCEPTION RETURN IS TAKEN:
0038 202
0038 203 : R3 LBS = RETRIABLE HARDWARE ERROR
0038 204 : R3 LBC = FATAL HARDWARE ERROR
0038 205
0038 206 : R0 CONTAINS THE STATUS CODE REFLECTING THE TYPE OF ERROR
0038 207 : DETECTED.
0038 208
0038 209 : FLAGS SET BY UTILITY:
0038 210
0038 211 : R3 LBC = TRANSFER OF ONE SECTOR COMPLETED
0038 212 :
```



```
0038 213 : R3 LBS = PERFORM NORMAL DRIVER FUNCTION
0038 214 :
0038 215 :
0038 216 : INPUTS:
0038 217 :
0038 218 : R3 = ADDRESS OF I/O PACKET.
0038 219 : R5 = ADDRESS OF DEVICE UNIT CONTROL BLOCK
0038 220 :
0038 221 : OUTPUTS:
0038 222 :
0038 223 : *****OUTPUTS*****
0038 224 :
0038 225 :
0038 226 : .ENABL LSB
0038 227 :
0038 228 : STARTIO:
0038 229 : JSB DX$STARTIO ;CALL UTILITY ROUTINE TO SETUP PHYSICAL ADDR
003E 230 : RESTART:
003E 231 : MOVZWL #F_READSECTOR,R3 ;ASSUME READ PHYSICAL SECTOR
0043 232 : BBC #UCB$V_DX_WRITE,UCB$W_DEVSTS(R5),20$ ;BRANCH IF READ
0048 233 : ADDB2 #<F_WRITESECTOR-F_READSECTOR>,R3 ;CONVERT FUNCTION CODE TO WRITE SEC
004B 234 : 20$:
004B 235 : BSBB DXPOUT ;SEND COMMAND TO FLOPPY
004D 236 : MOVZBL UCB$L_MEDIA(R5),R3 ;GET SECTOR NUMBER
0052 237 : BSBB DXPOUT ;OUTPUT SECTOR
0054 238 : MOVZBL UCB$L_MEDIA+2(R5),R3 ;GET CYLINDER NUMBER
0059 239 : BSBB DXPOUT ;SEND CYLINDER NUMBER
005B 240 : BBS #UCB$V_DX_WRITE,UCB$W_DEVSTS(R5),DXPWRITE ;BRANCH IF WRITE
0060 241 : BSBB DXPINP ;WAIT FOR FLOPPY TO FINISH
0062 242 : 30$:
0062 243 : BSBB DXPINP ;WAIT FOR INPUT DATA
0064 244 : MOVB R3,@UCB$L_DX_BFPNT(R5) ;STORE BYTE
0069 245 : JSB @($P)+ ;RETURN TO CALLER
006B 246 : BLBS R3,30$ ;IF LBS CONTINUE TRANSFER
006E 247 : BRB 40$ ;NO MORE DATA
0070 248 :
0070 249 : +
0070 250 : DXPWRITE - OUTPUT TO CONSOLE FLOPPY
0070 251 :
0070 252 : THIS ROUTINE IS ENTERED TO WRITE ONE BYTE OF DATA ON THE CONSOLE FLOPPY.
0070 253 : IF AN ERROR OCCURS, A COROUTINE CALL IS MADE TO THE EXCEPTION ENTRY POINT
0070 254 : SPECIFIED BY CALLER'S CALLER. THE STACK CONTAINS THE RETURN TO IN-LINE DRIVER
0070 255 : CODE. IN THIS CASE, R3 INDICATES ERROR SEVERITY AS FOLLOWS:
0070 256 :
0070 257 : R3 LBS = RETRIABLE ERROR
0070 258 : R3 LBC = FATAL ERROR
0070 259 :
0070 260 : THE ERROR CODE IS IN R0.
0070 261 :
0070 262 : IF NO ERROR OCCURS, A RETURN TO THE DRIVER IS EXECUTED.
0070 263 :
0070 264 : INPUTS:
0070 265 :
0070 266 : R3 LBS = REQUEST TO WRITE ONE BYTE OF DATA
0070 267 : R5 = ADDRESS OF UCB
0070 268 : ($P) = RETURN ADDRESS
0070 269 :
```

|             |    |      |     |
|-------------|----|------|-----|
| 00000000'EF | 16 | 0038 | 229 |
| 53 0800 8F  | 3C | 003E | 230 |
| 03 68 A5 03 | E1 | 0043 | 232 |
| 53 01       | 80 | 0048 | 233 |
|             |    | 004B | 234 |
| 53 00BC C5  | 9A | 004D | 236 |
|             | 5C | 0052 | 237 |
| 53 00BE C5  | 9A | 0054 | 238 |
|             | 55 | 0059 | 239 |
| 10 68 A5 03 | E0 | 005B | 240 |
|             | 29 | 0060 | 241 |
|             |    | 0062 | 242 |
| 00D0 D5     | 27 | 0062 | 243 |
|             | 53 | 0064 | 244 |
|             | 9E | 0069 | 245 |
| F4 53       | 16 | 0069 | 245 |
|             | E8 | 006B | 246 |
| OE          | 11 | 006E | 247 |



```
0070 270 : R3 LBC = SECTOR TRANSFER COMPLETE
0070 271 :
0070 272 : OUTPUTS:
0070 273 :
0070 274 : A RETURN TO INLINE CODE IS EXECUTED FOR A SUCCESSFUL TRANSFER.
0070 275 : IF SECTOR TRANSFER IS COMPLETE, A FORK IS EXECUTED BEFORE CALLING
0070 276 : THE CALLER.
0070 277 :
0070 278 : A RETURN TO THE EXCEPTION ENTRY POINT IS TAKEN IF AN ERROR OCCURS.
0070 279 : IN THIS CASE, R3 INDICATES THE SEVERITY AND R0 CONTAINS THE ERROR
0070 280 : CODE.
0070 281 :
0070 282 : -
0070 283 :
0070 284 : DXPWRITE:
53 00D0 D5 9A 0070 285 MOVZBL @UCBSL_DX_BFPNT(R5),R3 : GET OUTPUT DATA
39 10 0075 286 BSBB DXPOUT : SEND TO DEVICE
9E 16 0077 287 JSB @ (SP)+ : CALL THE CALLER
F4 53 E8 0079 288 BLBS R3,DXPWRITE : IF LBS WRITE ANOTHER BYTE
OD 10 007C 289 BSBB DXPINP : WAIT FOR FLOPPY FUNCTION COMPLETE
007E 290 40$:
54 8E D0 007E 291 MOVL (SP)+,R4 : RETRIEVE RETURN ADDRESS
OC81 292 IOFORK : DROP TO FORK LEVEL
64 16 0087 293 JSB (R4) : CALL THE CALLER
B3 11 0089 294 BRB RESTART : GO AGAIN
008B 295
008B 296 .DSABL LSB
008B 297
008B 298 : +
008B 299 : DXPINP - WAIT FOR FLOPPY INPUT INTERRUPT
008B 300 :
008B 301 : THIS ROUTINE IS ENTERED VIA A BSBB TO WAIT FOR AN INPUT INTERRUPT.
008B 302 : IT ENTERS COMMON CODE TO EXECUTE A WAIT FOR INTERRUPT. ON RECEIPT
008B 303 : OF THE INTERRUPT A RETURN TO THE CALLER IS EXECUTED AT ISR LEVEL.
008B 304 :
008B 305 : INPUTS:
008B 306 :
008B 307 : (SP) = RETURN TO CALLER
008B 308 : 4(SP) = RETURN TO CALLERS CALLER (DXUTILITY CO-ROUTINE)
008B 309 : 8(SP) = RETURN TO EXECUTIVE
008B 310 :
008B 311 : OUTPUTS:
008B 312 :
008B 313 : R3 = INPUT DATA
008B 314 : R5 = UCB ADDRESS
008B 315 :
008B 316 : (SP) = RETURN TO CO-ROUTINE IN DXUTILITY
008B 317 :
008B 318 : -
008B 319 :
008B 320 : .ENABL LSB
008B 321 :
008B 322 : DXPINP:
25 64 A5 05 E0 0091 323 DSBINT #31 : DISABLE ALL INTERRUPTS
1C BA 0096 324 BBS #UCBSV_POWER,UCBSW_STS(R5) 5$ : BRANCH IF POWER FAILURE
52 DD 0098 325 POPR #M<R2,R3,R4> : GET IPL IN R2, RETURNS IN R3, R4
326 326 PUSHL R2 : RESTORE IPL TO TOP OF STACK
```

```
35 64 A5 07 E4 009A 327 WFIKPC 40$, #10 :WAIT FOR INTERRUPT
64 A5 03 A8 00A4 328 BBSC #UCBSV_INTTYPE,UCBSW_STS(R5),25$ :IF SET, RECEIVED INPUT INT.
00B5 31 00A9 329 BLSW #<UCBSM_INT!UCBSM_TIM>,UCBSW_STS(R5) ;ENABLE INTERRUPTS AND TIMEOUTS
00B0 330 BRW UNSOLNT ;
00B0 331
00B0 332
00B0 333 :+
00B0 334 : DXPOUT - PERFORM OUTPUT TO THE CONSOLE FLOPPY DISK
00B0 335 : THIS ROUTINE IS ENTERED VIA A BSB TO SEND DATA TO THE CONSOLE FLOPPY.
00B0 336 : IT ENTERS THE COMMON CONSOLE INTERRUPT HANDLER TO TRANSFER THE DATA.
00B0 337 :
00B0 338 : INPUTS:
00B0 339 :
00B0 340 : R3 = FLOPPY DATA (BIT 11 MUST BE SET FOR A FLOPPY CONSOLE COMMAND).
00B0 341 : R5 = ADDRESS OF UCB
00B0 342 : (SP) = RETURN TO CALLER
00B0 343 : 4(SP) = RETURN TO CALLERS CALLER (DXUTILITY CO-ROUTINE)
00B0 344 : 8(SP) = RETURN TO EXECUTIVE
00B0 345 :
00B0 346 : OUTPUTS:
00B0 347 :
00B0 348 : R5 = UCB ADDRESS
00B0 349 :
00B0 350 : (SP) = RETURN TO CO-ROUTINE IN DXUTILITY
00B0 351 :
00B0 352 : -
00B0 353 :
00B0 354 : DXPOUT:
00B0 355 : DSBINT #31 :DISABLE ALL DEVICE INTERRUPTS
00B6 356 BBC #UCBSV_POWER,UCBSW_STS(R5),10$ ;BRANCH IF NO POWER FAILURE
00BB 357 5$:
00BB 358 ENBINT :ENABLE INTERRUPTS
00BE 359 POPR #^M<R3,R4> :REMOVE RETURNS FROM STACK
00C0 360 CLRL R3 :SET TO FLAG NONFATAL ERROR
00C2 361 BRW 50$
00C5 362 10$:
00C5 363 JSB G^CON$STARTIO :STARTUP THE DEVICE
00CB 364 20$:
00CB 365 POPR #^M<R2,R3,R4> :GET IPL IN R2, RETURNS IN R3, R4
00CD 366 PUSHL R2 :RESTORE IPL TO TOP OF STACK
00CF 367 WFIKPC 40$, #10 :WAIT FOR INTERRUPT, KEEP CHANNEL
00D9 368 BBCC #UCBSV_INTTYPE,UCBSW_STS(R5),30$ :IF CLEAR, RECEIVED OUTPUT INTERRUPT
00DE 369 25$:
00DE 370 MOVL R3,UCBSL_DX_RXDB(R5) :SAVE CONTENTS OF RXDB
00E3 371 EXTZV #8,#4,R3,-(SP) :EXTRACT COMMAND FIELD
00E8 372 CMPL (SP)+,#<DXP_RXDB_K_FCMPO-8> :TEST COMPLETION CODE
00EB 373 BLSS 30$ :IF LSS PROCESS NEXT CHARACTER
00ED 374 BNEQ DXPERR :IF NEQ, ERROR
00EF 375 TSTB R3 :TEST STATUS
00F1 376 BNEQ DXPERR :IF NEQ, EXCEPTION CONDITION
00F3 377 30$:
00F3 378 PUSHL R4 :SET RETURN TO CALLER
00F5 379 JMP @UCBSL_FR3(R5) :RETURN TO CALLER
00F8 380 40$:
00F8 381 DSBINT #31 :LOCKOUT ALL INTERRUPTS
00FE 382 BBS #UCBSV_POWER,UCBSW_STS(R5),43$ :DO NOT REINITIALIZE ON POWER FAIL
0103 383 JSB G^CON$INITIAL :INITIALIZE DEVICE AND OUTPUT QUEUE
```



```

0109 384 43$:
0109 385 ENBINT ;ENABLE INTERRUPTS
010C 386 CLRL R3 ;SET TO FLAG FATAL ERROR
010E 387 MOVZWL #SS$_TIMEOUT,R0 ;GET FINAL STATUS
0113 388 BRB 60$
0115 389
0115 390
0115 391 :+
0115 392 : DXPERR - HARDWARE DEPENDANT ERROR PROCESSING
0115 393 : THIS ROUTINE IS ENTERED WHENEVER AN ERROR INDICATION IS RECEIVED FROM
0115 394 : THE FLOPPY INTERFACE. IF THE PROBLEM WAS NOT CAUSED BY DEVICE TIMEOUT
0115 395 : OR POWER FAIL THEN THE DEVICE REGISTERS ARE SAVED AND A FORK IS EXE-
0115 396 : CUTED TO PERFORM ERROR ANALYSES.
0115 397 :
0115 398 : IF THE ERROR IS CAUSED BY HARDWARE, THE SEVERITY (FATAL OR NON-FATAL)
0115 399 : AND STANDARD ERROR CODE ARE SETUP.
0115 400 :
0115 401 : INPUTS:
0115 402 :
0115 403 : R3 = CONTENTS OF RXDB (IF ENTRY IS FROM CONSOLE INTERRUPT DISPATCHER)
0115 404 : R4 = CO-ROUTINE ENTRY POINT
0115 405 : R5 = ADDRESS OF UCB
0115 406 :
0115 407 : OUTPUTS:
0115 408 :
0115 409 : R0 CONTAINS ERROR CODE
0115 410 : R3 LSB = 1, RETRIABLE ERROR
0115 411 : R3 LSB = 0, FATAL ERROR
0115 412 :
0115 413 :-
0115 414
0115 415 DXPERR:
0115 416 IOFORK
0115 417 MOVZWL #SS$_CTRLERR,R0 ;FORK
0115 418 MOVL R3,R1 ;ASSUME CONTROLLER ERROR
0115 419 CLRL R3 ;COPY RXDB CONTENTS
0115 420 EXTZV #8,R1,R2 ;ASSUME ERROR IS FATAL
0115 421 CMPB R2,#<DXP_RXDB_K_FCMDB-8> ;EXTRACT COMMAND FIELD
0115 422 BEQL 45$ ;FUNCTION COMPLETE?
0115 423 CMPB R2,#<DXP_RXDB_K_CMDB-8> ;IF EQL YES, CHECK ERRORS
0115 424 BNEQ 60$ ;PROTOCOL ERROR?
0115 425 CMPB R1,#DXP_RXDB_K_PRTC ;IF NEQ GARBAGE IN SELECT FIELD
0115 426 BEQL 50$ ;CHECK DATA BYTE
0115 427 BRB 60$ ;IF EQL, PROTOCOL ERROR
0115 428 45$: ;ELSE GARBAGE IN DATA BYTE
0115 429 MOVZWL #SS$_PARITY,R0 ;ASSUME DATA ERROR
0115 430 TSTB R1 ;TEST FOR DATA ERROR
0115 431 BLSS 50$ ;IF LSS DATA ERROR
0115 432 MOVZWL #SS$_FORMAT,R0 ;ASSUME READ DELETED DATA
0115 433 BBS #DXP_RXDB_V_DEL,R1,60$ ;IF BIT SET, FATAL FORMAT ERROR
0115 434 50$:
0115 435 INCL R3 ;NONFATAL ERROR
0115 436 60$:
0115 437 JSB DX$ERR ;CALL COMMON EXCEPTION CODE
0115 438 BRW RESTART ;RESTART TRANSFER
0115 439
0115 440 .DSABL LSB
```

```
0158 441
0158 442 :+
0158 443 : REGDUMP - CONSOLE FLOPPY REGISTER DUMP ROUTINE
0158 444 :
0158 445 : THIS ROUTINE IS ENTERED TO COPY THE CONSOLE FLOPPY STATUS REGISTER
0158 446 : CONTENTS (RXDB) TO THE SPECIFIED BUFFER. IT IS CALLED FROM THE DE-
0158 447 : VICE ERROR LOGGING ROUTINE AND FROM THE DIAGNOSTIC BUFFER FILL ROU-
0158 448 : TIME.
0158 449 :
0158 450 : INPUTS:
0158 451 :
0158 452 :     R0 = ADDRESS OF REGISTER SAVE BUFFER
0158 453 :     R5 = ADDRESS OF UCB
0158 454 :
0158 455 : OUTPUTS:
0158 456 :
0158 457 :     THE COPY OF RXDB RECORDED IN THE UCB IS SAVED IN THE SPECIFIED BUFFER.
0158 458 :
0158 459 :-
0158 460 :
0158 461 REGDUMP:
0158 462     MOVL    #2,(R0)+           ; SET NUMBER OF DEVICE REGISTERS
0158 463     MOVL    UCB$$_DX_RXDB(R5),(R0)+ ; COPY DEVICE REGISTER
0158 464     MOVL    UCB$$_MEDIA(R5),(R0) ; COPY LAST DISK ADDRESS
0165 465 UNSOLNT:
0165 466     RSB
0166 467 :
0166 468 :+
0166 469 : DX$UNITINIT - UNIT INITIALIZATION
0166 470 :
0166 471 : THIS ROUTINE IS CALLED ON INITIAL DRIVER LOAD AND ON POWER RECOVERY
0166 472 : TO INITIALIZE THE UNIT. ON INITIAL DRIVER LOAD, IT ALLOCATES A
0166 473 : 128 BYTE SECTOR BUFFER FROM NON-PAGED POOL AND LINKS IT ONTO THE UCB.
0166 474 : IT THEN PUTS THE UCB ADDRESS INTO THE SLOT FOR UNIT 2 IN THE UCB LIST
0166 475 : IN THE IDB.
0166 476 : ON POWER RECOVERY IT SIMPLY RETURNS.
0166 477 :
0166 478 : INPUTS:
0166 479 :
0166 480 :     R5 = ADDRESS OF UCB
0166 481 :
0166 482 : OUTPUTS:
0166 483 :
0166 484 :     UCB$$_DX_BUF = ADDRESS OF SECTOR BUFFER
0166 485 :-
0166 486 :
0166 487 DX$UNITINIT::
0166 488     BBS      #UCB$$_POWER,UCB$$_STS(R5),30$ ; RETURN IF POWER RECOVERY
0166 489     TSTL     UCB$$_DX_BUF(R5) ; IS THERE ALREADY A SECTOR BUFFER?
0166 490     BNEQ     30$ ; YES, RETURN
0171 491 :
0171 492     MOVAL     G^EXE$GL_NONPAGED,R4 ; PUT ADDRESS OF NON-PAGED POOL
0178 493 :     LIST HEAD IN R4
0178 494     PUSHL     (R4) ; SAVE IPL IN POOL LIST HEAD
017A 495     MFR      #PR$$_IPL,(R4) ; SET ALLOCATION IPL TO 31
0181 496 :
0181 497     MOVZBL    #140,R1 ; SIZE OF BLOCK TO ALLOCATE
```

80 80 02 D0 0158 462  
00D4 C5 D0 0158 463  
00BC C5 D0 0160 464  
05 0165 465  
0165 466  
0166 467  
0166 468  
0166 469  
0166 470  
0166 471  
0166 472  
0166 473  
0166 474  
0166 475  
0166 476  
0166 477  
0166 478  
0166 479  
0166 480  
0166 481  
0166 482  
0166 483  
0166 484  
0166 485  
0166 486  
44 64 A5 05 E0 0166 488  
00CC C5 D5 0168 489  
3E 12 016F 490  
0171 491  
54 00000000'GF DE 0171 492  
0178 493  
64 DD 0178 494  
00000000'8F DB 017A 495  
0181 496  
51 8C 8F 9A 0181 497



```
00000000'GF 16 0185 498 JSB G^EXESALONONPAGED ; ALLOCATE MEMORY
      12 50 E9 018B 499 BLBC R0,20$ ; BR. IF FAILURE
08 A2 51 B0 018E 500 MOVW R1,8(R2) ; STORE SIZE OF BLOCK IN BLOCK
0A A2 13 90 0192 501 MOVW #DYN$C_BUF10,10(R2) ; STORE TYPE OF BLOCK IN BLOCK
00CC C5 52 0C C1 0196 502 ADDL3 #12,R2,UCB$L_DX_BUF(R5) ; SAVE ADDRESS OF BLOCK
      64 A5 10 A8 019C 503 B1SW #UCB$M_ONLINE,UCB$W_STS(R5) ; SET DEVICE ONLINE
      64 8ED0 01A0 504 POPL (R4) ; RESTORE IPL IN LISTHEAD
      01A3 506
50 24 A5 D0 01A3 507 MOVL UCB$L_CRB(R5),R0 ; GET ADDRESS OF CRB
51 2C A0 D0 01A7 508 MOVL CRB$L_INTD+VE($L_IDB(R0),R1 ; GET ADDRESS OF IDB
20 A1 55 D0 01AB 509 MOVL R5,IDB$L_UCBLST+8(R1) ; STORE UCB ADDRESS IN SLOT FOR UNIT 2
      01AF 510 ; (THIS IS BECAUSE CONSOLE FLOPPY CAN
      01AF 511 ; INTERRUPT AS UNIT 2)
      01AF 512
      05 01AF 513 30$: RSB
      01B0 514
      01B0 515 .END
```

DXPDRIVER  
Symbol table

- VAX-11/780 RX01 CONSOLE DRIVER K 11

15-SEP-1984 23:55:49 VAX/VMS Macro V04-00 Page 11  
5-SEP-1984 00:14:13 [DRIVER.SRC]DXPDRIVER.MAR;1 (1)

```

SSS = 00000020 R 03
SSOP = 00000002
ATS_NULL ***** X 03
BIT... = 00000008 ***** X 04
CONSINITIAL ***** X 04
CONSSTARTIO ***** X 04
CRBSL_INTD = 00000024 ***** X 03
DCS_DISK ***** X 03
DDBSK_SLOW = 00000003 ***** X 03
DDBSL_ACPD = 00000010 ***** X 03
DDBSL_DDT = 0000000C ***** X 03
DEVSM_AVL ***** X 03
DEVSM_DIR ***** X 03
DEVSM_FOD ***** X 03
DEVSM_IDV ***** X 03
DEVSM_NNM ***** X 03
DEVSM_ODV ***** X 03
DEVSM_RND ***** X 03
DEVSM_SHR ***** X 03
DPTSC_LENGTH = 00000038 ***** X 03
DPTSC_VERSION = 00000004 ***** X 03
DPTSINITAB = 00000038 R 03
DPTSM_SVP = 00000002 ***** R 03
DPTSREINITAB = 00000087 R 03
DPTSTAB = 00000000 R 03
DTS_RX01 ***** X 03
DX$ERR ***** X 04
DX$FUNCTIONAL ***** X 04
DX$STARTIO ***** X 04
DX$UNITINIT 00000166 RG 04
DXPSDDT 00000000 RG 04
DXPSDPT 00000000 RG 01
DXPERR 00000115 R 04
DXPINP 0000008B R 04
DXPOUT 00000080 R 04
DXPWRITE 00000070 R 04
DXP_END ***** X 03
DXP_M_CMD = 00000800 ***** X 03
DXP_RXDB_K_CMD = 00000900 ***** X 03
DXP_RXDB_K_FCMF = 00000200 ***** X 03
DXP_RXDB_K_PRTC = 00000005 ***** X 03
DXP_RXDB_M_CRC = 00000001 ***** X 03
DXP_RXDB_M_DEL = 00000040 ***** X 03
DXP_RXDB_M_ERR = 00000080 ***** X 03
DXP_RXDB_M_INI = 00000004 ***** X 03
DXP_RXDB_M_PAR = 00000002 ***** X 03
DXP_RXDB_V_CRC = 00000000 ***** X 03
DXP_RXDB_V_DEL = 00000006 ***** X 03
DXP_RXDB_V_ERR = 00000007 ***** X 03
DXP_RXDB_V_INI = 00000002 ***** X 03
DXP_RXDB_V_PAR = 00000001 ***** X 03
DYN$C_BUFID = 00000013 ***** X 04
DYN$C_DDB = 00000006 ***** X 04
DYN$C_DPT = 0000001E ***** X 04
DYN$C_UCB = 00000010 ***** X 04
EXESA[ONONPAGED ***** X 04
EXESGL_NONPAGED ***** X 04

```

```

EXESIOFORK ***** X 04
FUNCTAB_LEN = 00000000 ***** X 04
F_CANCEL = 00000804 ***** X 04
F_READSECTOR = 00000800 ***** X 04
F_READSTATUS = 00000802 ***** X 04
F_WRITEDELDATA = 00000803 ***** X 04
F_WRITESECTOR = 00000801 ***** X 04
IDBSL_UCBLST = 00000018 ***** X 04
IOCSMTVER ***** X 04
IOCSRETURN ***** X 04
IOCSWFIKPCB ***** X 04
PR$ IPL ***** X 04
REGDUMP 00000158 R 04
RESTART 0000003E R 04
SIZ... = 00000001 ***** X 04
SSS_CTRLERR ***** X 04
SSS_FORMAT ***** X 04
SSS_PARITY ***** X 04
SSS_TIMEOUT ***** X 04
STARTIO 00000038 R 04
UCBSB_DEVCLASS = 00000040 ***** X 04
UCBSB_DEVTYPE = 00000041 ***** X 04
UCBSB_DIPL = 0000005E ***** X 04
UCBSB_DX_SCTCNT = 000000DA ***** X 04
UCBSB_ERTCNT = 00000080 ***** X 04
UCBSB_ERTMAX = 00000081 ***** X 04
UCBSB_FIPL = 0000000B ***** X 04
UCBSB_SECTORS = 00000044 ***** X 04
UCBSB_TRACKS = 00000045 ***** X 04
UCBSL_CRB = 00000024 ***** X 04
UCBSL_DEVCHAR = 00000038 ***** X 04
UCBSL_DEVCHAR2 = 0000003C ***** X 04
UCBSL_DX_BFPNT = 000000D0 ***** X 04
UCBSL_DX_BUF = 000000CC ***** X 04
UCBSL_DX_RXDB = 000000D4 ***** X 04
UCBSL_FR3 = 00000010 ***** X 04
UCBSL_MAXBLOCK = 000000B0 ***** X 04
UCBSL_MEDIA = 000000BC ***** X 04
UCBSM_INT = 00000002 ***** X 04
UCBSM_NOCNVRT = 00000004 ***** X 04
UCBSM_ONLINE = 00000010 ***** X 04
UCBSM_TIM = 00000001 ***** X 04
UCBSV_DX_WRITE = 00000003 ***** X 04
UCBSV_INTTYPE = 00000007 ***** X 04
UCBSV_POWER = 00000005 ***** X 04
UCBSW_CYLINDERS = 00000046 ***** X 04
UCBSW_DEVBUFSIZ = 00000042 ***** X 04
UCBSW_DEVSTS = 00000068 ***** X 04
UCBSW_STS = 00000064 ***** X 04
UNSOLNT 00000165 R 04
VECSL_IDB = 00000008 ***** X 04

```



+-----+  
! Psect synopsis !  
+-----+

| PSECT name         | Allocation       | PSECT No. | Attributes                                              |
|--------------------|------------------|-----------|---------------------------------------------------------|
| . ABS :            | 00000000 ( 0.)   | 00 ( 0.)  | NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE |
| . BLANK :          | 00000000 ( 0.)   | 01 ( 1.)  | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE       |
| \$AB\$\$           | 00000000 ( 0.)   | 02 ( 2.)  | NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE       |
| \$\$\$105_PROLOGUE | 0000008D ( 141.) | 03 ( 3.)  | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE       |
| \$\$\$115_DRIVER   | 000001B0 ( 432.) | 04 ( 4.)  | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG       |

+-----+  
! Performance indicators !  
+-----+

| Phase                  | Page faults | CPU Time    | Elapsed Time |
|------------------------|-------------|-------------|--------------|
| Initialization         | 31          | 00:00:00.06 | 00:00:02.12  |
| Command processing     | 120         | 00:00:00.39 | 00:00:03.27  |
| Pass 1                 | 391         | 00:00:10.84 | 00:01:15.70  |
| Symbol table sort      | 0           | 00:00:01.55 | 00:00:12.68  |
| Pass 2                 | 102         | 00:00:02.06 | 00:00:16.53  |
| Symbol table output    | 14          | 00:00:00.07 | 00:00:00.07  |
| Psect synopsis output  | 1           | 00:00:00.02 | 00:00:00.02  |
| Cross-reference output | 0           | 00:00:00.00 | 00:00:00.00  |
| Assembler run totals   | 661         | 00:00:15.00 | 00:01:50.40  |

The working set limit was 1500 pages.  
84630 bytes (166 pages) of virtual memory were used to buffer the intermediate code.  
There were 80 pages of symbol table space allocated to hold 1480 non-local and 15 local symbols.  
515 source lines were read in Pass 1, producing 17 object records in Pass 2.  
34 pages of virtual memory were used to define 31 macros.

+-----+  
! Macro library statistics !  
+-----+

| Macro library name                  | Macros defined |
|-------------------------------------|----------------|
| _\$255\$DUA28:[SYS.OBJ]LIB.MLB;1    | 23             |
| _\$255\$DUA28:[SYSLIB]STARLET.MLB;2 | 6              |
| TOTALS (all libraries)              | 29             |

1723 GETS were required to define 29 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:DXPDRIVER/OBJ=OBJ\$:DXPDRIVER MSRC\$:DXPDRIVER/UPDATE=(ENH\$:DXPDRIVER)+EXECMLS/LIB



0111 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

