

(1)	136	DECLARATIONS
(1)	287	REGISTER DEFINITIONS
(1)	346	CONTROLLER INITIALIZATION
(1)	427	UNIT INITIALIZATION
(1)	512	MAINTENANCE ROUTINES
(1)	577	OUTPUT MODEM CONTROL
(1)	631	DZ-11 MODEM POLLER
(1)	717	RECEIVER INTERRUPT SERVICE
(1)	891	START I/O ROUTINE
(1)	943	PORT ROUTINES STOP, RESUME, XON, XOFF
(1)	1085	OUTPUT INTERRUPT SERVICE
(1)	1348	SET SPEED, PARITY PARAMETERS
(1)	1400	INITIALIZE DZ-11 MODEM POLLING

```

00000001 0000 1 DZV = 1 ; CREATE DZVDRIVER (Q-BUS VERSION)
0000 1 .if df DZV
0000 2 .TITLE DZVDRIVER - Port Driver for DZV-11 support
0000 3 .iff
0000 4 .TITLE DZDRIVER - Port Driver for DZ-11 support
0000 5 .endc
0000 6 .IDENT 'V04-000'
0000 7
0000 8
0000 9 :*****
0000 10 :*
0000 11 :* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 12 :* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 13 :* ALL RIGHTS RESERVED.
0000 14 :*
0000 15 :* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 16 :* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 17 :* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 18 :* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 19 :* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 20 :* TRANSFERRED.
0000 21 :*
0000 22 :* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 23 :* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 24 :* CORPORATION.
0000 25 :*
0000 26 :* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 27 :* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 28 :*
0000 29 :*
0000 30 :*****
0000 31 :
0000 32 :++
0000 33 : FACILITY:
0000 34 :
0000 35 : VAX/VMS TERMINAL DRIVER
0000 36 :
0000 37 : ABSTRACT:
0000 38 :
0000 39 : DZ PORT DRIVER
0000 40 : This module functions as a port driver for DZ11 and DZ32 terminal
0000 41 : controllers. It contains hardware specific port level service
0000 42 : routines.
0000 43 :
0000 44 : AUTHOR:
0000 45 :
0000 46 : RICK SPITZ
0000 47 :
0000 48 : Revision history:
0000 49 :
0000 50 : V03-028 MIRO480 Michael I. Rosenblum 8-Aug-1984
0000 51 : Fix bugs found in testing the DZ-32 and reported in
0000 52 : QAR 1126 (FT1).
0000 53 :
0000 54 : V03-027 LMP0275 L. Mark Pilant, 12-Jul-1984 21:01
0000 55 : Initialize the ACL info in the ORB to be a null descriptor
0000 56 : list rather than an empty queue. This avoids the overhead

```

```

0000 57 : of locking and unlocking the ACL mutex, only to find out
0000 58 : that the ACL was empty.
0000 59 :
0000 60 : V03-026 EMD0090 Ellen M. Dusseault 30-Apr-1984
0000 61 : Add DEVSM_NNM characteristic to DEVCHAR2 so that these
0000 62 : devices will have the 'node$' prefix.
0000 63 :
0000 64 : V03-025 LMP0221 L. Mark Pilant, 7-Apr-1984 13:37
0000 65 : Change UCBSL_OWNUIC to ORBSL_OWNER and UCBSW_VPROT to
0000 66 : ORBSW_PROT.
0000 67 :
0000 68 : V03-024 JLV0321 Jake VanNoy 5-JAN-1984
0000 69 : Minor enhancements to DZV11 support. Comment out
0000 70 : DPT_STORE for parity, must fix a restriction.
0000 71 :
0000 72 : V03-023 MIR0055 Michael I. Rosenblum 30-June-1983
0000 73 : Remove code from unit and controller-init routines
0000 74 : and make insert calls to class driver macros
0000 75 : Add DZV11 support.
0000 76 :
0000 77 : V03-022 RKS0022 RICK SPITZ 14-MAR-1983
0000 78 : ADD ENHANCEMENTS TO SUPPORT LOGICAL UCB.
0000 79 :
0000 80 : V03-021 MIR0022 Michael I. Rosenblum 19-Jan-1982
0000 81 : Change references to UCBSB_ERTCNT to use UCBSW_IT_UNITBIT
0000 82 : to be more maintainable.
0000 83 : Replace old vector table with new vector table macro.
0000 84 : Remove references to UCBSL_DEVDEPEND and UCBSQ_IT_STATE
0000 85 : move these references into the class driver jacket routines
0000 86 :
0000 87 : V03-020 MIR0021 Michael I. Rosenblum 17-Jan-1983
0000 88 : Fix DZ32 DZ_SET table entry.
0000 89 :
0000 90 : V03-019 RKS0019 RICK SPITZ 13-JAN-1983
0000 91 : Repair problem with port vector macro
0000 92 :
0000 93 : V03-018 MIR0019 Michael I. Rosenblum 11-Jan-1982
0000 94 : Fix undefined symbol created by MIR0018.
0000 95 :
0000 96 : V03-017 MIR0018 Michael I. Rosenblum 07-Jan-1983
0000 97 : Change the port vector table to use the $VEC macros
0000 98 :
0000 99 : V03-016 MIR0017 Michael I. Rosenblum 05-Jan-1983
0000 100 : Add powerfail check in the Unit init routine to allow the
0000 101 : terminal class driver to take positive action on powerfail.
0000 102 : Change code to accept a byte value as returns from the
0000 103 : GETNXT and PUTNXT class services, this will remove this
0000 104 : information from the condition codes.
0000 105 :
0000 106 : V03-015 MIR0016 Michael I. Rosenblum 29-Dec-1982
0000 107 : Replace time calculation code with TIMSET macro call
0000 108 :
0000 109 : V03-014 MIR0015 Michael I. Rosenblum 20-Dec-1982
0000 110 : Remove entry point to reflect the redefinition of the PORT_DISCONN
0000 111 : entry point.
0000 112 :
0000 113 : V03-013 MIR0014 Michael I. Rosenblum 17-Dec-1982

```

0000 114 :
0000 115 :
0000 116 :
0000 117 :
0000 118 :
0000 119 :
0000 120 :
0000 121 :
0000 122 :
0000 123 :
0000 124 :
0000 125 :
0000 126 :
0000 127 :
0000 128 :
0000 129 :
0000 130 :
0000 131 :
0000 132 :
0000 133 :
0000 134 :--

Remove code to calculate flow control characters from
port XON and XOFF routines and move that code into
the class driver.

V03-012 RKS0012 RICK SPITZ 16-SEP-1982
Check reference count in unit init to determine if
modem control should be initialized or hungup. This
is needed to insure that a hangup ^Y is posted on
powerfail for modem lines.

-011 JLV0211 Jake VanNoy 2-JUL-1982
Remove check for powerfail in unit init that prevents
SETUP_UCB from being called. This insures that UCB fields
are initialized correctly when Unit init is called for
use with CSS unibus switch.

V03-010 KDM0002 Kathleen D. Morse 28-Jun-1982
Added \$DEVDEF, \$IPLDEF, \$PRDEF, and \$\$SDEF.

```

0000 136          .SBTTL  DECLARATIONS
0000 137
0000 138 :
0000 139 : EXTERNAL DEFINITIONS:
0000 140 :
0000 141          $ACBDEF          : DEFINE ACB
0000 142          $CRBDEF          : DEFINE CRB
0000 143          $DCDEF           : DEVICE DEFINITIONS
0000 144          $DDBDEF          : DEFINE DDB
0000 145          $DEVDEF          : DEFINE DEVICE TYPES
0000 146          $DYNDEF          : DYNAMIC STRUCTURE DEFINITONS
0000 147          $IDBDEF          : DEFINE IDB OFFSETS
0000 148          $IODEF           : DEFINE I/O FUNCTION CODES
0000 149          $IPLDEF          : DEFINE INTERRUPT PRIORITY LEVELS
0000 150          $IRPDEF          : IRP DEFINITIONS
0000 151          $ORBDEF          : DEFINE OBJECT'S RIGHTS BLOCK OFFSETS
0000 152          $PRDEF           : DEFINE PROCESSOR REGISTERS
0000 153          $$SDEF           : DEFINE SYSTEM STATUS CODES
0000 154          $TTYDEF          : DEFINE TERMINAL DRIVER SYMBOLS
0000 155          $TTDEF           : DEFINE TERMINAL TYPES
0000 156          $TT2DEF          : DEFINE EXTENDED DEFINITIONS
0000 157          $TQDEF           : DEFINE TIMER QUEUE OFFSETS
0000 158          $UCBDEF          : DEFINE UCB
0000 159          $UBADEF          : DEFINE UBA
0000 160          $VECDEF          : DEFINE VECTOR FOR CRB
0000 161          $TTYMACS        : DEFINE TERMINAL DRIVER MACROS
0000 162          $TTYDEFS        : DEFINE TERMINAL DRIVER SYMBOLS
0000 163          $TTYMODEM       : DEFINE MODEM DEFINITIONS
0000 164
0000 165
0000 166 :
0000 167 : LOCAL STORAGE
0000 168 :
00000000 169          .PSECT $$$105_PROLOGUE
0000 170
0000 171 :
0000 172 : Driver prologue table:
0000 173 :
0000 174
0000 175 DZ$DPT::          : DRIVER START
0000 176 .IF DF DZV
0000 177          DPTAB          -          : DRIVER PROLOGUE TABLE
0000 178          END=DZ$END,-          : End and offset to INIT's vectors
0000 179          UCBSIZE=UCB$C TT LENGTH,- : SIZE OF UCB
0000 180          FLAGS=DPT$M_NOUNLOAD,-   : DO NOT ALLOW UNLOAD
0000 181          ADAPTER=UBA,-            : ADAPTER TYPE
0000 182          DEFUNITS=4,-            : DZV has 4 units
0000 183          NAME=DZDRIVER,-          : NAME OF DRIVER
0000 184          VECTOR=PORT_VECTOR      : PORT DRIVER VECTOR TABLE
0038 185 .IFF
0038 186          DPTAB          -          : DRIVER PROLOGUE TABLE
0038 187          END=DZ$END,-          : End and offset to INIT's vectors
0038 188          UCBSIZE=UCB$C TT LENGTH,- : SIZE OF UCB
0038 189          FLAGS=DPT$M_NOUNLOAD,-   : DO NOT ALLOW UNLOAD
0038 190          ADAPTER=UBA,-            : ADAPTER TYPE
0038 191          DEFUNITS=8,-            : Number of units to create
0038 192          NAME=DZDRIVER,-          : NAME OF DRIVER

```

```

0038 193 VECTOR=PORT_VECTOR ; PORT DRIVER VECTOR TABLE
0038 194 .ENDC
0038 195 DPT_STORE INIT
0038 196 DPT_STORE UCB,UCBSB_FIPL,B,8 ; FORK IPL
003C 197 DPT_STORE UCB,UCBSL_DEVCHAR,L,<- ; CHARACTERISTICS
003C 198 DEVSM_REC!-
003C 199 DEVSM_AVL!-
003C 200 DEVSM_IDV!-
003C 201 DEVSM_ODV!-
003C 202 DEVSM_TRM!-
003C 203 DEVSM_CCL>
0043 204 DPT_STORE UCB,UCBSL_DEVCHAR2,L,- ; Device Characteristics
0043 205 <DEVSM_NNM> ; prefix with 'node$'
004A 206 DPT_STORE UCB,UCBSB_DEVCLASS,B,DC$ TERM;
004E 207 DPT_STORE UCB,UCBSB_TT_DETYPE,B,TT$ UNKNOWN ; TYPE
0052 208 DPT_STORE UCB,UCBSW_TT_DESIZE,@W,TTY$GW_DEFBUF ; BUFFER SIZE
0059 209 DPT_STORE UCB,UCBSL_TT_DECHAR,@L,TTY$GL_DEFCHAR ; DEFAULT CHARACTERS
0060 210 DPT_STORE UCB,UCBSL_TT_DECHA1,@L,TTY$GL_DEFCHAR2; DEFAULT CHARACTERS
0067 211 DPT_STORE UCB,UCBSW_TT_DESPEE,@B,TTY$GB_DEFSPEED; DEFAULT SPEED
006E 212 DPT_STORE UCB,UCBSW_TT_DESPEE+1,@B,TTY$GB_RSPEED; DEFAULT SPEED
0075 213 DPT_STORE UCB,UCBSB_TT_DEPARI,@B,TTY$GB_PARITY ; DEFAULT PARITY
007C 214 ; DPT_STORE UCB,UCBSB_TT_PARITY,@B,TTY$GB_PARITY ; DEFAULT PARITY
007C 215 DPT_STORE UCB,UCBSB_DEVTYPE,B,TT$ UNKNOWN ; TYPE
0080 216 DPT_STORE UCB,UCBSW_DEVBUFSIZ,@W,TTY$GW_DEFBUF ; BUFFER SIZE
0087 217 DPT_STORE UCB,UCBSL_DEVDEPEND,@L,TTY$GL_DEFCHAR ; DEFAULT CHARACTERS
008E 218 DPT_STORE UCB,UCBSL_DEVDEPND2,@L,TTY$GL_DEFCHAR2; DEFAULT CHARACTERS
0095 219 DPT_STORE UCB,UCBSW_TT_SPEED,@B,TTY$GB_DEFSPEED ; DEFAULT SPEED
009C 220 DPT_STORE UCB,UCBSW_TT_SPEED+1,@B,TTY$GB_RSPEED ; DEFAULT SPEED
00A3 221 DPT_STORE UCB,UCBSB_DIPL,B,21 ; DEVICE IPL
00A7 222 DPT_STORE UCB,UCBSL_TT_WFLINK,L,0 ; Zero write queue.
00AE 223 DPT_STORE UCB,UCBSL_TT_WBLINK,L,0 ; Zero write queue.
00B5 224 DPT_STORE UCB,UCBSL_TT_RTIMOU,L,0 ; Zero read timed out disp.
00BC 225 DPT_STORE ORB,ORBSB_FLAGS,B,- ; Protection block flags
00BC 226 ; ZORBSM PROT 16> ; SOGW protection word
00C0 227 DPT_STORE ORB,ORBSW_PROT,@W,TTY$GW_PROT ; Default allocation protection
00C7 228 DPT_STORE ORB,ORBSL_OWNER,@L,TTY$GL_OWNUIC ; Default owner UIC
00CE 229 DPT_STORE DDB,DDBSL_DDT,D,DZ$DDT
00D3 230
00D3 231 DPT_STORE REINIT
00D3 232 DPT_STORE CRB,CRBSL_INTD+VECSL_INITIAL,D,DZ$INITIAL ; CONTROLLER INIT
00D8 233 DPT_STORE CRB,CRBSL_INTD+VECSL_UNITINIT,D,DZ$INITLINE; UNIT INIT
00DD 234 DPT_STORE END
0000 235
0000 236 DDTAB DEVNAM = DZ,- ; DUMMY DZ PORT DRIVER DISPATCH TABLE
0000 237 START = 0,-
0000 238 FUNCTB = 0
0038 239
0000038 240 .PSECT $$$115_DRIVER
0038 241
0038 242 ;
0038 243 ; THE ASSOCIATED CLASS DRIVER USES THIS TABLE TO COMMAND THE PORT DRIVER.
0038 244 ; THE ADDRESS OF THIS TABLE IS CONTAINED IN THE TERMINAL UCB EXTENSION AREA.
0038 245 ; THE OFFSET DEFINITIONS ARE DEFINED BY TTYDEFS.
0038 246 ;
0038 247
0038 248 PORT_VECTOR:
0038 249

```



```

0038 250 :
0038 251 : DZ-11 SPECIFIC DISPATCH TABLE
0038 252 :
0038 253 $VECINI DZ11,DZ$NULL
0070 254 $VEC STARTIO,DZ11$STARTIO
003C 255 $VEC SET_LINE,DZ$SET_LINE
0044 256 $VEC DS_SET,DZ11$DS_SET
0048 257 $VEC XON,DZ11$XON
004C 258 $VEC XOFF,DZ11$XOFF
0050 259 $VEC STOP,DZ$STOP
0054 260 $VEC ABORT,DZ$ABORT
005C 261 $VEC RESUME,DZ11$RESUME
0060 262 $VEC SET_MODEM,DZ11$SET_MODEM
0064 263 $VEC MAINT,DZ11$MAINT
006C 264 .IF NDF DZV
006C 265 $VECEND END=NO
006C 266 :
006C 267 : DZ-32 SPECIFIC DISPATCH TABLE
006C 268 :
006C 269 :
006C 270 $VECINI DZ32,DZ$NULL
006C 271 $VEC STARTIO,DZ32$STARTIO ; START NEW OUTPUT
006C 272 $VEC SET_LINE,DZ$SET_LINE ; SET NEW PARITY/SPEED
006C 273 $VEC DS_SET,DZ32$DS_SET ; SET NEW OUTPUT MODEM SIGNALS
006C 274 $VEC XON,DZ32$XON ; SEND XON
006C 275 $VEC XOFF,DZ32$XOFF ; SEND XOFF
006C 276 $VEC STOP,DZ$STOP ; STOP CURRENT OUTPUT
006C 277 $VEC ABORT,DZ$ABORT ; ABORT CURRENT OUTPUT
006C 278 $VEC RESUME,DZ32$RESUME ; RESUME STOPPED OUTPUT
006C 279 $VEC MAINT,DZ32$MAINT ; INVOKE MAINTENANCE FUNCTIONS
006C 280 .ENDC
006C 281 $VECEND
05 0074 282 DZ$NULL: ; NULL PORT ROUTINE
0074 283 RSB
0075 284
0075 285

```

```

0075 287      .SBTTL REGISTER DEFINITIONS
0075 288
0075 289      :
0075 290      : CSR BIT DEFINITIONS ( CSR ) ( READ/WRITE )
0075 291      :
0075 292      $VIELD DZCSR,0,<-
0075 293      <MODE,1,M>,-      : DZ32 - MODE/ DZ11 - UNUSED
0075 294      <DS_ENAB,1,M>,- : DZ32 - DATA SET INTERRUPT ENABLE
0075 295      <1>,-          : UNUSED
0075 296      <MAINT,1,M>,-   : LINE TURNAROUND
0075 297      <CLEAR,1,M>,-  : MASTER RESET
0075 298      <MASTENAB,1,M>,- : MASTER SCAN ENABLE
0075 299      <RCVINT,1,M>,-  : RECEIVER INTERRUPT ENABLE
0075 300      <RCVRDY,1,M>,- : RECEIVER READY
0075 301      <LINE,3,M>,-   : LINE NUMBAE ( 0 - 7)
0075 302      <DS_CHG,1,M>,- : DZ32 - DATA SET INTERRUPT
0075 303      <2>,-          : UNUSED
0075 304      <SNDINT,1,M>,- : TRANSMIT INTERRUPT ENABLE
0075 305      <SNDRDY,1,M>,- : TRANSMITTER READY
0075 306      >
0075 307      :
0075 308      : RECEIVER BUFFER ( CSR+2 ) ( READ ONLY )
0075 309      :
0075 310      $VIELD DZRCV,0,<-
0075 311      <BUF,8,M>,-      : RECEIVER DATA
0075 312      <LINE,3,M>,-   : LINE NUMBER ( 0 - 7)
0075 313      <1>,-          :
0075 314      <PARERR,1,M>,- : PARITY ERROR
0075 315      <FRAMER,1,M>,- : FRAME ERROR
0075 316      <OVERRUN,1,M>,- : OVERRUN ERROR
0075 317      <VALID,1,M>,-  : DATA VALID
0075 318      >
0075 319      :
0075 320      : LINE PARAMETER REGISTER ( CSR+2 ) ( WRITE ONLY )
0075 321      :
0075 322      $VIELD DZLPR,0,<-
0075 323      <LINE,3,M>,-   : LINE NUMBER (0-7)
0075 324      <SIZE,2,M>,-  : CHARACTER SIZE
0075 325      <STOP,1,M>,-  : NUMBER STOP BITS
0075 326      <PARITY,1,M>,- : PARITY ENABLE
0075 327      <ODD,1,M>,-   : ODD PARITY
0075 328      <SPEED,4,M>,- : LINE SPEED
0075 329      <CLOCK,1,M>,- : RECEIVER CLOCK
0075 330      <SPLIT,1,M>,- : DZ32 - SPLIT SPEED
0075 331      >
0075 332      :
0075 333      :
0075 334      : DZ-32 SPECIFIC MODEM CONTROL
0075 335      :
0075 336      :
0075 337      $VIELD DZLCS1,8,<-
0075 338      <7>,-          :
0075 339      <ACK,1,M>,-    : READY FOR COMMAND/ UPDATE OUTPUT MODEM
0075 340      >
0075 341
0075 342
0075 343

```



```

0075 346 .SBTTL CONTROLLER INITIALIZATION
0075 347
0075 348
0075 349
0075 350 :++
0075 351 : DZ$INITIAL - INITIALIZE INTERFACE
0075 352 :
0075 353 : FUNCTIONAL DESCRIPTION:
0075 354 :
0075 355 : THIS ROUTINE IS ENTERED AT SYSTEM STARTUP AND POWER RECOVERY.
0075 356 :
0075 357 : INPUTS:
0075 358 :
0075 359 : R4 = ADDRESS OF THE UNIT CSR
0075 360 : R5 = IDB OF UNIT
0075 361 : R8 = ADDRESS OF THE UNIT CRB
0075 362 :
0075 363 : OUTPUTS:
0075 364 :
0075 365 : R2 is destroyed.
0075 366 :
0075 367 : IMPLICIT INPUTS:
0075 368 :
0075 369 : IPL = IPL$ POWER
0075 370 :
0075 371 :--
0075 372 DZ$INITIAL:: ; INITIALIZE DZ UNIT
0075 373
0075 374 :
0075 375 : SET UP CONTROLLER
0075 376 :
0075 377 class_ctrl_init dz$dpt,port_vector
00A1 378
64 10 B0 00A1 379 25$: MOVW #DZCSR$M_CLEAR,(R4) ; INIT CONTROLLER RESET
00A4 380
00A4 381 :
00A4 382 : WAIT TILL CONTROLLER INITIALIZATION IS COMPLETE
00A4 383 :
00A4 384 : TIMEWAIT #500,#DZCSR$M_CLEAR,(R4),W,.FALSE.
00CB 385
64 4063 8F B0 00CB 386 MOVW #<<DZCSR$M_MASTENAB>!-- ;
00D0 387 <DZCSR$M_RCVINT>!-- ; ENABLE RECEIVER INTERRUPTS
00D0 388 <DZCSR$M_SNDINT>!-- ; ENABLE TRANSMITTER INTERRUPTS
00D0 389 <DZCSR$M_DS_ENAB>!-- ; ENABLE DZ-32 DATA SET INTERRUPTS
00D0 390 <DZCSR$M_MODE>>,(R4) ; ENABLE ENHANCED MODE ON DZ-32
21 50 E9 00D0 391 BLBC R0,DZ$CTRL_ERROR
00D3 392
19 52 64 B0 00D3 393 MOVW (R4),R2 ; GET NEW STATUS
00D6 394 BBS #DZCSR$V_MODE,R2,110$ ; BRANCH IF DZ-32 CONTROLLER
00DA 395
00DA 396 100$: MOVB #DZ1_DZ1!,CRB$B_TT_TYPE(R8); CONTROLLER IS DZ11
00DF 397
00DF 398 :
00DF 399 : INIT DZ-11 INTERRUPT VECTORS
00DF 400 : THIS IS DONE HERE TO ALLOW THE DRIVER TO SERVICE INTERUPTS
00DF 401 : FOR BOTH THE DZ-11 AND DZ-32 BETWEEN CONTROLER AN UNIT INIT.
00DF 402 :

```

```
28 A8 0000028E'EF DE 00DF 403 MOVAL DZ11$INTINP,CRB$SL_INTD+4(R8) ; INIT RECEIVER VECTOR
4C A8 000003FA'EF DE 00E7 404 MOVAL DZ11$INTOUT,CRB$SL_INTD2+4(R8) ; INIT TRANSMITTER VECTOR
      00EF 405
      1C A8 B4 00EF 406 CLRW CRB$B_DZ_RING(R8) ; RESET CURRENT DZ-11 MODEM STATE
      05 00F2 407 RSB
      00F3 408
      00F3 409 110$: ;DZ-32 CONTROLLER SPECIFIC INIT
      00F3 410 .if ndf DZV
      00F3 411
      00F3 412 MOVB #DTS DZ32,CRB$B_TT_TYPE(R8) ; CONTROLLER IS DZ-32
      00F3 413 CLRW IDB$B_TT_ENABLE(R5) ; RESET DZ-32 LINE ENABLE
      00F3 414 MOVB IDB$B_TT_ENABLE(R5),7(R4) ; RESET TRANSMIT LINE ENABLES
      00F3 415 :
      00F3 416 :: INIT DZ-32 ALTERNATE INTERRUPT VECTORS
      00F3 417 :
      00F3 418 MOVAL DZ32$INTINP,CRB$SL_INTD+4(R8) ; INIT RECEIVER VECTOR
      00F3 419 MOVAL DZ32$INTOUT,CRB$SL_INTD2+4(R8) ; INIT TRANSMITTER VECTOR
      00F3 420 .endc
      05 00F3 421 RSB
      00F4 422
      00F4 423 DZ$CTRL_ERROR:
      05 00F4 424 RSB
      00F5 425
```

```

00F5 427      .SBTTL UNIT INITIALIZATION
00F5 428      :++
00F5 429      : DZ$INITLINE - UNIT INITIALIZATION
00F5 430      :
00F5 431      : FUNCTIONAL DESCRIPTION:
00F5 432      :
00F5 433      : THIS ROUTINE PERFORMS A SIMPLE UNIT INITIALIZATION.
00F5 434      :
00F5 435      : INPUTS:
00F5 436      :
00F5 437      :     R5 = UCB ADDRESS
00F5 438      :
00F5 439      : OUTPUTS:
00F5 440      :
00F5 441      :     R2,R5 ARE PRESERVED.
00F5 442      :--
00F5 443
00F5 444 DZ$INITLINE::
00F5 445
54 24 A5 D0 00F5 446      MOVL    UCBSL_CRB(R5),R4      ; GET CRB ADDRESS
00F9 447
00F9 448 .IF NDF DZV      ; IF NOT DZV
00F9 449
00F9 450      MOVAL   DZ32$VEC,R0      ; SET DZ-32 PORT VECTOR TABLE
00F9 451      CMPB   #DTS_DZ32,CRBSB_TT_TYPE(R4) ; IS IT DZ-32 ?
00F9 452      BEQL   5$      ; YES
00F9 453
00F9 454 .ENDC ; END OF DZ32 CODE
00F9 455
50 FF3B CF DE 00F9 456      MOVAL   DZ11$VEC,R0      ; SET DZ-11 PORT VECTOR TABLE
00FE 457 5$: CLASS_UNIT_INIT
53 64 A5 10 AB 0147 458      BISW   #UCBSM_ONLINE,UCBSW_STS(R5); SET ONLINE
0106 C5 53 B0 014B 459 10$: ASHL   UCBSW_UNIT(R5),#1,R3 ; BUILD UNIT'S BIT MASK
51 0114 C5 D0 0150 460      MOVW   R3,UCBSW_TT_UNITBIT(R5) ; SAVE IT
000004A7'EF 16 0155 461      MOVL   UCBSL_TT_CLASS(R5),R1 ; ADDRESS CLASS VECTOR TABLE
54 24 A5 D0 015A 462      JSB    @CLASS_SETUP_UCB(R1) ; INIT UCB FIELDS
015D 463 20$:
0163 464      JSB    DZ$SET_LINE      ; INIT SPEED/PARITY
0163 465
0163 466      MOVL   UCBSL_CRB(R5),R4      ; GET CRB ADDRESS
0167 467
0167 468 .IF NDF DZV      ; IF NOT DZV
0167 469
0167 470      CMPB   #DTS_DZ11,CRBSB_TT_TYPE(R4) ; CONTROLLER DZ11?
0167 471      BEQL   25$      ; YES
0167 472
0167 473
0167 474 :
0167 475 : INIT RECEIVER MODEM STATUS FOR DZ-32
0167 476 :
0167 477      MOVL   @CRBSL_INTD+VEC$L_IDB(R4),R4 ; GET CSR ADDRESS
0167 478
0167 479 ; WAIT TILL MODEM CONTROL READY FOR COMMAD
0167 480
0167 481      TIMEWAIT #500,#DZLCS1$M_ACK,4(R4),W,.TRUE.
0167 482      BLBC   R0,DZ$UNIT_ERROR
0167 483      MOVW   UCBSW_UNIT(R5),4(R4) ; REQUEST STATUS ON LINE

```

```

0167 484
0167 485 ; WAIT FOR COMPLETION
0167 486
0167 487 TIMEWAIT #500,#DZLCS1$M ACK,4(R4),W,.TRUE.
0167 488 BLBC R0,DZ$UNIT_ERROR
0167 489 MOVB 4(R4),UCB$B_TT_DS_RCV(R5) ; UPDATE RECEIVER MODEM STATUS
0167 490
0167 491 25$:
0167 492 .ENDC ; END OF DZ32 CODE
0167 493
0167 494 PUSHL R2
50 51 00 9A 0169 495 MOVZBL #MODEM$C_INIT,R1 ; ASSUME INIT MODEM PROTOCOL
0114 C5 D0 016C 496 MOVL UCB$L_TT_CLASS(R5),R0 ; ADDRESS CLASS VECTOR TABLE
OC B0 16 0171 497 JSB @CLASS_DS_TRAN(R0) ; INVOKE TO INIT MODEM PROTOCOL
52 8ED0 0174 498 POPL R2
08 64 A5 05 E1 0177 499 30$:
50 0114 C5 D0 017C 500 BBC #UCB$V_POWER,UCB$W_STS(R5),40$; DID WE DETECT A POWER FAIL
20 B0 17 0181 501 MOVL UCB$L_TT_CLASS(R5),R0 ; GET THE CLASS VECTOR TABLE ADDRESS
05 0184 502 JMP @CLASS_POWERFAIL(R0) ; AND GOTO THE POWERFAIL CODE
0185 503 40$:
0185 504 RSB
0185 505 ; ERROR DETECTED DURING INITIALIZATION
0185 506 ;
0185 507
0185 508 DZ$UNIT_ERROR:
64 A5 10 AA 0185 509 BICW #UCB$M_ONLINE,UCB$W_STS(R5) ; UNIT NOT ON LINE
05 0189 510 RSB

```

```

018A 512      .SBTTL MAINTENANCE ROUTINES
018A 513      :++
018A 514      : DZ$MAINT - MAINTENANCE FUNCTIONS
018A 515      :
018A 516      : FUNCTIONAL DESCRIPTION:
018A 517      :
018A 518      : THIS ROUTINE PERFORMS MAINTENANCE FUNCTIONS FOR THE DZ .
018A 519      :
018A 520      :
018A 521      : INPUTS:
018A 522      :
018A 523      :     R5 = UBC ADDRESS
018A 524      :     UCBSB_TT_MAINT = FUNCTION TO BE PERFORMED
018A 525      :
018A 526      : OUTPUTS:
018A 527      :
018A 528      :--
018A 529      .IF NDF DZV
018A 530 DZ32$MAINT:
018A 531      BITB  #IOSM_LOOPa-7,-          ; LOOPBACK FUNCTION
018A 532      UCBSB_TT_MAINT(R5)
018A 533      BEQL  5$                          ; NO
018A 534      MOVZBL #^X40,R2                ; SPECIFY LOOPBACK CODE
018A 535      BRB   10$
018A 536 5$:
018A 537      BITB  #IOSM_UNLOOPa-7,-        ; RESET LOOPBACK FUNCTION
018A 538      UCBSB_TT_MAINT(R5)
018A 539      BEQL  15$                          NO
018A 540      MOVZWL #^X7200,R2              ; SPECIFY UNLOOP CODE (BOTH)
018A 541 10$:
018A 542      JSB   DZ32$DS_SET              ; UPDATE CONTROLLER
018A 543      MOVZBL #1,R0                  ; INDICATE SUCCESS
018A 544      RSB
018A 545 15$:
018A 546      BITB  #IOSM_LOOP_EXTa-7,-     ; LOOPBACK FUNCTION
018A 547      UCBSB_TT_MAINT(R5)
018A 548      BEQL  20$                          ; NO
018A 549      MOVZBL #^X72,R2              ; SPECIFY LOOPBACK CODE
018A 550      BRB   10$
018A 551
018A 552 20$:
018A 553      .ENDC
018A 554 DZ11$MAINT:
018A 555      BITB  #IOSM_LINE_OFFa-7,-      ; LINE OFF
018A 556      UCBSB_TT_MAINT(R5)
018A 557      BEQL  10$                          ; NO
018A 558      BISB  #UCBSM_TT_DSBL,-        ; DISABLE LINE
018A 559      UCBSB_TT_MAINT(R5)
018A 560      BRB   20$
018A 561 10$:
018A 562      BITB  #IOSM_LINE_ONa-7,-       ; LINE ON
018A 563      UCBSB_TT_MAINT(R5)
018A 564      BEQL  30$                          ; NO
018A 565      BICB  #UCBSM_TT_DSBL,-        ; REENABLE LINE
018A 566      UCBSB_TT_MAINT(R5)
018A 567 20$:
018A 568      JSB   DZ$SET_LINE              ; IMPLEMENT FUNCTION
012A C5 04 93 018A 555
      08 13 018C 556
80 8F 88 018F 557
012A C5 0191 558
      0D 11 0194 559
      10 93 0197 560
012A C5 0199 561
      10 13 0199 562
      80 8F 8A 019B 563
012A C5 019E 564
      01A0 565
      01A3 566
000004A7'EF 16 01A6 567
      01A6 568

```


DZVDRIVER
V04-000

- Port Driver for DZV-11 support G 16
MAINTENANCE ROUTINES

16-SEP-1984 02:23:45 VAX/VMS Macro V04-00
5-SEP-1984 04:15:55 [TTDPVR.SRC]DZDRIVER.MAR;1

Page 14
(1)

50	01	9A	01AC	569		MOVZBL	#1,R0
		05	01AF	570		RSB	
			01B0	571	30\$:		
50	D4		01B0	572		CLRL	R0
		05	01B2	573		RSB	
			01B3	574			
			01B3	575			

```

01B3 577      .SBTTL OUTPUT MODEM CONTROL
01B3 578      :++
01B3 579      : DZ$DS_SET - SET OUTPUT MODEM SIGNALS
01B3 580      :
01B3 581      : FUNCTIONAL DESCRIPTION:
01B3 582      :
01B3 583      : THIS ROUTINE OUTPUTS THE OUTPUT MODEM SIGNALS FOR THE SPECIFIED UNIT
01B3 584      :
01B3 585      : INPUTS:
01B3 586      :
01B3 587      :     R2 = LOW BYTE - SIGNALS TO ACTIVATE
01B3 588      :     HIGH BYTE- SIGNALS TO DEACTIVATE
01B3 589      :
01B3 590      :     R5 = UBC ADDRESS
01B3 591      :
01B3 592      : OUTPUTS:
01B3 593      :
01B3 594      :     R0-R3 ARE USED.
01B3 595      :--
01B3 596
01B3 597 DZ11$DS_SET:
01B3 598     PUSHL R4           : SAVE
52 0125 C5 54 DD 01B5 599     BISB R2,UCB$B TT_DS_TX(R5) : SET NEW OUTPUT SIGNALS
   52 52 F8 8F 78 01BA 600     ASHL #-8,R2,R2           : ACCESS SIGNALS TO RESET
52 0125 C5 52 8A 01BF 601     BICB R2,UCB$B TT_DS_TX(R5) : RESET THEM
   54 24 A5 DO 01C4 602     MOVL UCB$C_CRB(R5),R4      : GET CRB ADDRESS
   53 2C B4 DO 01C8 603     MOVL @CRB$C_INTD+VEC$C_IDB(R4),R3
01C0 604     : GET CSR ADDRESS
51 0125 C5 01 01 EF 01CC 605     EXTZV #TTSV_DS_DTR,#1,UCB$B TT_DS_TX(R5),R1
01D3 606     : GET CURRENT DTR FOR LINE
51 51 54 A5 78 01D3 607     ASHL UCB$W_UNIT(R5),R1,R1 : SHIFT TO RELATIVE LINE POSITION
1E A4 0106 C5 8A 01D8 608     BICB UCB$W_TT_UNITBIT(R5),CRB$B DZ_DTR(R4)
01DE 609     : RESET CURRENT DTR FOR THAT LINE
   1E A4 51 88 01DE 610     BISB R1,CRB$B DZ_DTR(R4)   : SET IT IF NEED BE
05 A3 1E A4 90 01E2 611     MOVB CRB$B_DZ_DTR(R4),5(R3) : UPDATE DTR STATUS FOR LINES
   54 8ED0 01E7 612     POPL R4
01EA 613     RSB
01EB 614     .IF NDF DZV
01EB 615 DZ32$DS_SET:
01EB 616     PUSHL R4           : SAVE
01EB 617     BISB R2,UCB$B TT_DS_TX(R5) : SET NEW OUTPUT SIGNALS
01EB 618     ASHL #-8,R2,R2           : ACCESS SIGNALS TO RESET
01EB 619     BICB R2,UCB$B TT_DS_TX(R5) : RESET THEM
01EB 620     MOVL UCB$C_CRB(R5),R4      : GET CRB ADDRESS
01EB 621     MOVL @CRB$C_INTD+VEC$C_IDB(R4),R3
01EB 622     TIMEWAIT #100,#DZLCS1$M_ACK,4(R3),W,.TRUE. : WAIT FOR READY
01EB 623     MOVZWL UCB$B TT_DS_TX-T(R5),-(SP) : CREATE TEMP LOCATION
01EB 624     MOVB UCB$W_UNIT(R5),(SP) : SET UNIT NUMBER
01EB 625     BISW #DZLCS1$M_ACK,(SP) : ENABLE NEW OUTPUT SIGNALS
01EB 626     CVTLW (SP)+,4(R3) : SET NEW OUPUT MODEM SIGNALS
01EB 627     POPL R4
01EB 628     RSB
01EB 629 .ENDC

```

```

01EB 631 .SBTTL DZ-11 MODEM POLLER
01EB 632 :++
01EB 633 : DZ$TIMER - POLL FOR DZ-11 MODEM TRANSITIONS
01EB 634 :
01EB 635 : FUNCTIONAL DESCRIPTION:
01EB 636 :
01EB 637 : THIS ROUTINE CHECKS FOR DZ-11 CONTROLLER MODEM
01EB 638 : TRANSITION. IT UPDATES THE INPUT MODEM STATUS FOR EACH
01EB 639 : LINE AND CALLS THE CLASS TRANSITION ROUTINE FOR EACH LINE WITH
01EB 640 : A CHANGE.
01EB 641 :
01EB 642 : INPUTS:
01EB 643 :
01EB 644 : R5 - TQE ADDRESS
01EB 645 :
01EB 646 : OUTPUTS:
01EB 647 :
01EB 648 : R0 - R4 DESTROYED
01EB 649 :
01EB 650 :--
01EB 651 :
01EB 652 DZ$TIMER:
54 0060 8F BB 01EB 653 PUSHR #*M<R5,R6>
00000000'EF DE 01EF 654 MOVAL DZ$L_DIALUP,R4 ; GET DZ TIMER LIST HEAD
54 64 DO 01F6 655 5$:
05 12 01F6 656 MOVL (R4),R4 ; GET NEXT CRB ADDRESS
0060 8F BA 01FB 657 BNEQ 15$ ; PROCESS LINES FOR THIS CRB
05 01FF 658 POPR #*M<R5,R6> ; RESTORE REGISTERS
0200 659 RSB ; RETURN FROM TIMER INTERRUPT
0200 660 :
0200 661 : TEST LINES ON THIS CONTROLLER FOR A TRANSITION
0200 662 :
0200 663 :
0200 664 15$:
06 53 54 DD 0200 665 PUSHL R4 ; SAVE TIMER THREAD
A3 18 C2 0202 666 SUBL #CRB$L_DZ_MODEM,R4 ; GET ACTUAL CRB ADDRESS
2C B4 DO 0205 667 MOVL @CRB$L_INTD+VEC$L_IDB(R4),R3 ; GET CSR ADDRESS
1C A4 B1 0209 668 CMPW CRB$B_DZ_RING(R4),6(R3) ; ANY TRANSITIONS
78 13 020E 669 BEQL 60$ ; NONE
0210 670 :
0210 671 : FIND WHICH SIGNALS CHANGED AND UPDATE THEM
0210 672 :
50 52 06 A3 90 0210 673 MOVB 6(R3),R2 ; GET NEW RING
1C A4 52 8D 0214 674 XORB3 R2,CRB$B_DZ_RING(R4),R0 ; FIND TRANSITIONED LINES
1C A4 52 90 0219 675 MOVB R2,CRB$B_DZ_RING(R4) ; UPDATE CURRENT RING
52 07 A3 90 021D 676 MOVB 7(R3),R2 ; GET NEW CARRIER
56 1D A4 52 8D 0221 677 XORB3 R2,CRB$B_DZ_CARRIER(R4),R6
50 56 88 0226 678 BISB R6,R0 ; FLAG LINES WITH TRANSITIONED CARRIER
1D A4 52 90 0229 679 MOVB R2,CRB$B_DZ_CARRIER(R4) ; UPDATE CURRENT CARRIER
022D 680 :
022D 681 : PROCESS TRANSITIONED LINES
022D 682 :
022D 683 :
51 50 08 00 EA 022D 684 50$: FFS #0,#8,R0,R1 ; FIND NEXT LINE NEEDING SERVICE
00 50 54 13 0232 685 BEQL 60$ ; DONE
00 50 51 E5 0234 686 BBCC R1,R0,55$ ; RESET ATTENTION BIT FOR THIS LINE
0238 687 55$:

```

56	2C	A4	D0	0238	688	MOVL	CRBSL_INTD+VECSL_IDB(R4),R6			
55	18	A641	D0	023C	689	MOVL	IDBSL_UCBLST(R6)[R1],R5	; GET UCB FOR THAT LINE		
		EA	13	0241	690	BEQL	50\$; NONE		
E5	44	A5	E1	0243	691	BBC	#TTSV_MODEM,UCBSL_DEVDEPEND(R5),50\$			
				0248	692			; SKIP IF NOT MODEM LINE		
				0248	693	DSBINT	UCBSB_DIPL(R5)	; RAISE TO DEVICE IPL		
56	1C	A4	01	51	EF	024F	694	EXTZV	R1,#1,CRBSB_DZ_RING(R4),R6	; GET RING FOR THAT LINE
		01	06	56	FO	0255	695	INSV	R6,#TTSV_DS_RING,#1,-	; UPDATE IT IN UCB
			0124	C5		0259	696		UCBSB_TT_DS_RCV(R5)	
56	1D	A4	01	51	EF	025C	697	EXTZV	R1,#1,CRBSB_DZ_CARRIER(R4),R6	; GET CD FOR THAT LINE
		01	05	56	FO	0262	698	INSV	R6,#TTSV_DS_CARRIER,#1,-	; UPDATE IT IN UCB
			0124	C5		0266	699		UCBSB_TT_DS_RCV(R5)	
			90	8F	88	0269	700	BISB	#<TTSM_DS_DSR!TTSM_DS_CTS>,-	; ASSUME CTS AND DSR ALWAYS SET
			0124	C5		026C	701		UCBSB_TT_DS_RCV(R5)	
52			0124	C5	90	026F	702	MOVB	UCBSB_TT_DS_RCV(R5),R2	; GET CURRENT RECV MODEM STATUS
			51	03	9A	0274	703	MOVZBL	#MODEMSC_DATASET,R1	; SIGNAL DATASET TRANSITION
56			0114	C5	D0	0277	704	MOVL	UCBSL_TT_CLASS(R5),R6	; GET CLASS VECTOR TABLE
				1F	BB	027C	705	PUSHR	#*M<R0,RT,R2,R3,R4>	; SAVE VOLITAL REGISTERS
			0C	B6	16	027E	706	JSB	@CLASS_DS_TRAN(R6)	; SIGNAL TRANSITION
				1F	BA	0281	707	POPR	#*M<R0,R1,R2,R3,R4>	; RESTORE REGISTERS
						0283	708	ENBINT		; RESTORE IPL
			A5	11		0286	709	BRB	50\$	
						0288	710			
						0288	711			
			54	8ED0		0288	712	POPL	R4	; RESTORE TIMER THREAD
			FF68	31		028B	713	BRW	5\$	
						028E	714			
						028E	715			

```

028E 717 .SBTTL RECEIVER INTERRUPT SERVICE
028E 718 :++
028E 719 : DZ$INTINP - DZ RECEIVER READY INTERRUPTS
028E 720 :
028E 721 : FUNCTIONAL DESCRIPTION:
028E 722 :
028E 723 : THIS ROUTINE IS ENTERED WHEN A CHARACTER IS AVAILABLE IN THE UNIT'S
028E 724 : SILO. THE CHARACTER IS EXTRACTED AND IS PASSED TO THE ASSOCIATED
028E 725 : CLASS DRIVER. IF THE CLASS DRIVER RETURNS CHARACTER(S) THEN NEW
028E 726 : OUTPUT IT INITIATED (NORMALLY ECHO).
028E 727 :
028E 728 : INPUTS:
028E 729 :
028E 730 :      OO(SP) = ADDRESS OF IDB
028E 731 :
028E 732 : IMPLICIT INPUTS:
028E 733 :
028E 734 :      R0,R1,R2,R3,R4,R5 ARE SAVED ON STACK.
028E 735 :
028E 736 : OUTPUTS:
028E 737 :
028E 738 :      THE INTERRUPT IS DISMISSED WHEN THE SILO IS EMPTY.
028E 739 :
028E 740 :--
028E 741 DZ11$INTINP:: : DZ-11 INPUT INTERRUPTS
028E 742 :
028E 743 : GET THE CSR ADDRESS
028E 744 :
028E 745 :      MOVL @ (SP)+,R4 : GET THE IDB ADDRESS
0291 746 :      PUSHL R4 : SAVE IDB ADDRESS
0293 747 :      MOVL (R4),R0 : GET THE CSR ADDRESS
0296 748 :
0296 749 : GET THE CHARACTER FROM THE INTERFACE
0296 750 :
0296 751 25$: MOVW 2(R0),R3 : GET THE CHARACTER, ERRORS AND LINE NUMBER
029A 752 : BGEQ 100$ : SILO EMPTY
029C 753 : BITW #<DZRCVSM_PARRR>!-
02A1 754 : <DZRCVSM_OVERRUN>!-
02A1 755 : <DZRCVSM_FRAMER>,R3 : ERRORS?
02A1 756 : BNEQ 50$ : YES, PROCESS THEM
52 53 F8 8F 78 02A3 757 27$: ASHL #-8,R3,R2 : GET THE LINE NUMBER
52 FFFFFFF8 8F CA 02A8 758 : BICL #^C<7>,R2 :
02AF 759 : MOVZBL R3,R3 : CLEAR THE HIGH BYTES OF CHARACTER
55 18 A442 D0 02B2 760 : MOVL IDB$L_UCBLST(R4)[R2],R5 : GET THE UCB FOR THAT LINE
02B7 761 : BEQL 25$ : IF EQL THEN NOT THERE
0110 D5 16 02B9 762 : JSB @UCB$L_TT_PUTNXT(R5) : BUFFER THE CHARACTER
010B C5 95 02BD 763 : TSTB UCB$B_TT_OUTYPE(R5) : DID HE RETURN ANYTHING TO OUTPUT
02C1 764 : BLEQ 40$ : NONE OR STRING OUTPUT
0108 C5 53 90 02C3 765 : MOVW R3,UCB$W_TT_HOLD(R5) : SAVE THE CHARACTER IN TANK
0400 8F A8 02C8 766 : BISW #TTY$M_TANK_HOLD,- : SIGNAL CHARACTER IN TANK
0108 C5 02CC 767 : UCB$W_TT_HOLD(R5)
04 A0 0106 C5 A8 02CF 768 : BISW UCB$W_TT_UNITBIT(R5),4(R0) : ENABLE LINE
54 6E D0 02D5 769 30$: MOVL (SP),R4 : GET IDB ADDRESS
BC 11 02DB 770 : BRB 25$ : CONTINUE
02DA 771 40$:
02DA 772 : BEQL 30$ : NO CHARACTER
0800 8F A8 02DC 773 : BISW #TTY$M_TANK_BURST,- : SIGNAL BURST

```

```

04 A0 0108 C5 02E0 774 UCBSW_TT_HOLD(R5)
0106 C5 A8 02E3 775 BISW UCBSW_TT_UNITBIT(R5),4(R0) ; ENABLE LINE
EA 11 02E9 776 BRB 30$
02EB 777
02EB 778 :: SILO EMPTY OR CHARACTER IN ERROR
02EB 779
02EB 780 50$:
02EB 781
02EB 782 :: PROCESS PARITY, FRAME OR OVERRUN ERROR
02EB 783
52 53 F8 8F 78 02EB 784 ASHL #-8,R3,R2 ; GET LINE NUMBER
52 FFFFFFF8 8F CA 02F0 785 BICL #^C<7>,R2
55 18 A442 D0 02F7 786 MOVL IDB$_UCBLST(R4)[R2],R5; GET UCB ADDRESS
0A 13 02FC 787 BEQL 70$ ; IF EQL THEN NO UCB
52 0114 C5 D0 02FE 788 MOVL UCBS$_TT_CLASS(R5),R2 ; GET CLASS DISPATCH
0303 789
14 B2 16 0303 790 60$: JSB @CLASS_READERROR(R2) ; SIGNAL ERROR
98 12 0306 791 BNEQ 27$ ; BRANCH WITH CHARACTER TO MAIN PATH
60 0080 8F B3 0308 792 70$: BITW #^X080,(R0) ; VALID CHARACTER IN SILO NOW?
C6 12 030D 793 BNEQ 30$ ; IF NEQ THEN YES
5E 04 C0 030F 794 100$: ADDL #4,SP ; REMOVE IDB ADDRESS
50 8E 7D 0312 795 MOVQ (SP)+,R0 ; RESTORE REGISTERS
52 8E 7D 0315 796 MOVQ (SP)+,R2
54 8E 7D 0318 797 MOVQ (SP)+,R4
02 031B 798 REI
031C 799

```

```

031C 801 .IF NDF DZV
031C 802 :
031C 803 : DZ-32 INPUT INTERRUPT SERVICE
031C 804 :
031C 805 :
031C 806 DZ32$INTINP:: : DZ-32 INPUT INTERRUPTS
031C 807 :
031C 808 : GET THE CSR ADDRESS
031C 809 :
031C 810 :     MOVL    @ (SP)+,R4 : GET THE IDB ADDRESS
031C 811 :     PUSHL   R4         : SAVE IDB ADDRESS
031C 812 :     MOVL    (R4),R0    : GET THE CSR ADDRESS
031C 813 :     BICW    #<<DZCSR$M_RCVINT>!-- : DISABLE RECEIVER INTERRUPTS
031C 814 :     <DZCSR$M_DS_ENAB>>,- : DISABLE DZ-32 DATA SET INTERRUPTS
031C 815 :     (R0)             : DZ-32 during the interrupt service routine
031C 816 :
031C 817 : GET THE CHARACTER FROM THE INTERFACE
031C 818 :
031C 819 25$:  MOVW    2(R0),R3 : GET THE CHARACTER, ERRORS AND LINE NUMBER
031C 820 :     BGEQ    100$        : SILO EMPTY
031C 821 :     BITW    #<DZRCV$M_PARERR>!-- :
031C 822 :     <DZRCV$M_OVERRUN>!-- : ERRORS?
031C 823 :     <DZRCV$M_FRAMER>,R3 : ERRORS?
031C 824 :     BNEQ    50$        : YES, PROCESS THEM
031C 825 27$:  ASHL    #-8,R3,R2 : GET THE LINE NUMBER
031C 826 :     BICL    #^C<7>,R2  :
031C 827 :     MOVZBL  R3,R3      : CLEAR THE HIGH BYTES OF CHARACTER
031C 828 :     MOVL    IDB$L_UCBLST(R4)[R2],R5 : GET THE UCB FOR THAT LINE
031C 829 :     BEQL    25$        : IF EQL THEN NOT THERE
031C 830 :     JSB    @UCB$L_TT_PUTNXT(R5) : BUFFER THE CHARACTER
031C 831 :     TSTB   UCBSB_TT_OUTYPE(R5) : DID HE RETURN ANYTHING TO OUTPUT
031C 832 :     BLEQ    40$        : NONE OR STRING OUTPUT
031C 833 :     MOVB   R3,UCBSW_TT_HOLD(R5) : SAVE THE CHARACTER IN TANK
031C 834 :     BISW   #TTY$M_TANK_HOLD,- : SIGNAL CHARACTER IN TANK
031C 835 :     UCBSW_TT_HOLD(R5)
031C 836 28$:  MOVL    (SP),R4 : RESTORE IDB ADDRESS
031C 837 :     BISB   UCBSW_TT_UNITBIT(R5),- : ENABLE LINE
031C 838 :     IDB$B_TT_ENABLE(R4)
031C 839 :     MOVB   IDB$B_TT_ENABLE(R4),7(R0)
031C 840 :
031C 841 :
031C 842 30$:  MOVL    (SP),R4 : GET IDB ADDRESS
031C 843 :     BRB    25$        : CONTINUE
031C 844 40$:  BEQL    30$        : NO CHARACTER
031C 845 :     BISW   #TTY$M_TANK_BURST,- : SIGNAL BURST
031C 846 :     UCBSW_TT_HOLD(R5)
031C 847 :     BRB    28$        :
031C 848 :
031C 849 :
031C 850 : SILO EMPTY OR CHARACTER IN ERROR
031C 851 :
031C 852 50$:  :
031C 853 :
031C 854 : PROCESS PARITY, FRAME OR OVERRUN ERROR
031C 855 :
031C 856 :     ASHL    #-8,R3,R2 : GET LINE NUMBER
031C 857 :     BICL    #^C<7>,R2 :

```



```

031C 891 .SBTTL START I/O ROUTINE
031C 892 :++
031C 893 : DZ$STARTIO - START I/O OPERATION ON DZ
031C 894 :
031C 895 : FUNCTIONAL DESCRIPTION:
031C 896 :
031C 897 : THIS ROUTINE IS ENTERED FROM THE DEVICE INDEPENDENT TERMINAL STARTIO
031C 898 : ROUTINE TO ENABLE OUTPUT INTERRUPTS ON AN IDLE DZ UNIT.
031C 899 :
031C 900 : INPUTS:
031C 901 :
031C 902 : R3 = CHARACTER AND CC = PLUS
031C 903 : ADDRESS AND CC = NEGATIVE
031C 904 :
031C 905 : R5 = UCB ADDRESS
031C 906 :
031C 907 : OUTPUTS:
031C 908 :
031C 909 : R5 = UCB ADDRESS
031C 910 :--
031C 911 DZ11$STARTIO:: : START I/O ON UNIT
031C 912 BGEQ 20$ : SINGLE CHARACTER
031E 913 BISW #TTY$M_TANK_BURST,- : SIGNAL BURST ACTIVE
0322 914 UCBSW_TT_HOLD(R5)
0325 915 10$: MOVL UCBSL_CRB(R5),R1 : GET CRB OF UNIT
0329 916 MOVL @CRB$[INTD+VEC$[IDB(R1),R1] : GET CSR
032D 917 BISW UCBSW_TT_UNITBIT(R5),4(R1) : ENABLE LINE
0333 918 RSB : RETURN TO CALLER
0334 919 20$:
0334 920 MOVB R3,UCBSW_TT_HOLD(R5) : SAVE OUTPUT CHARACTER
0339 921 BISW #TTY$M_TANK_HOLD,- : SIGNAL CHARACTER IN TANK
033D 922 UCBSW_TT_HOLD(R5)
0340 923 BRB 10$
0342 924
0342 925 .IF NDF DZV
0342 926 DZ32$STARTIO:: : START I/O ON UNIT
0342 927 BGEQ 20$ : SINGLE CHARACTER SPECIFIED
0342 928 BISW #TTY$M_TANK_BURST,- : SIGNAL BURST ACTIVE
0342 929 UCBSW_TT_HOLD(R5)
0342 930 10$: MOVL UCBSL_CRB(R5),R1 : GET CRB OF UNIT
0342 931 MOVL CRB$[INTD+VEC$[IDB(R1),R4] : GET IDB ADDRESS
0342 932 MOVL (R4),R1 : GET CSR ADDRESS
0342 933 BISB UCBSW_TT_UNITBIT(R5),IDB$B_TT_ENABLE(R4)
0342 934 MOVB IDB$B_TT_ENABLE(R4),7(R1)
0342 935 RSB : RETURN TO CALLER
0342 936 20$:
0342 937 MOVB R3,UCBSW_TT_HOLD(R5) : SAVE OUTPUT CHARACTER
0342 938 BISW #TTY$M_TANK_HOLD,- : SIGNAL CHARACTER IN TANK
0342 939 UCBSW_TT_HOLD(R5)
0342 940 BRB 10$
0342 941 .ENDC

```

```

          16 18
0800 8F A8
0108 C5
51 24 A5 D0
51 2C B1 D0
04 A1 0106 C5 A8
          05
0108 C5 53 90
0400 8F A8
0108 C5
          E3 11

```

```

0342 943 .SBTTL PORT ROUTINES STOP,RESUME,XON,XOFF
0342 944 :++
0342 945 : DZ$XOFF - SEND XOFF
0342 946 : DZ$XON - SEND XON
0342 947 : DZ$STOP - STOP OUTPUT
0342 948 : DZ$ABORT - ABORT CURRENT OUTPUT
0342 949 : DZ$RESUME - RESUME STOPPED OUTPUT
0342 950 :
0342 951 : FUNCTIONAL DESCRIPTION:
0342 952 :
0342 953 : THESE ROUTINES ARE USED BY THE THE TERMINAL CLASS DRIVER TO
0342 954 : CONTROL OUTPUT ON THE PORT
0342 955 :
0342 956 : INPUTS:
0342 957 :
0342 958 : R5 = UCB ADDRESS
0342 959 :
0342 960 : OUTPUTS:
0342 961 :
0342 962 : R5 = UCB ADDRESS
0342 963 :--
0342 964 : .ENABLE LSB
0342 965 :
0342 966 : SCHEDULE XOFF TO BE SEND
0342 967 :
0342 968 : INPUTS:
0342 969 : R3 - CHARACTER TO BE SENT AS FLOW CONTROL
0342 970 :
0342 971 DZ11$XOFF:
0342 972 :
0342 973 : SCHEDULE XON TO BE SENT
0342 974 :
0342 975 DZ11$XON:
0108 C5 0100 8F A8 0342 976 BISW #TTY$M_TANK_PREMPT,UCB$W_TT_HOLD(R5) ; SCHEDULE XON
010A C5 53 90 0349 977 MOVB R3,UCB$B_TT_PREMPT(R5) ; SAVE THE CHARACTER
18 64 A5 01 E0 034E 978
034E 979 BBS #UCB$V_INT,UCB$W_STS(R5),10$ ; IF OUTPUT ACTIVE,
0353 980 ; FINISHED
51 24 A5 DD 0353 981 PUSHL R1 ; SAVE A REGISTER
51 2C B1 D0 0355 982 MOVL UCB$L_CRB(R5),R1 ; ACCESS CRB ADDRESS
04 A1 0106 C5 A8 0359 983 MOVL @CRB$C_INTD+VEC$L_IDB(R1),R1 ; GET CSR ADDRESS
00 64 A5 01 8ED0 035D 984 BISW UCB$W_TT_UNITBIT(R5),4(R1) ; ENABLE LINE
0363 985 POPL R1
0366 986 BBSS #UCB$V_INT,UCB$W_STS(R5),10$ ; SHOW OUTPUT ACTIVE
036B 987 10$:
036B 988 RSB
036C 989 .DISABLE LSB
036C 990 :
036C 991 : STOP PORT OUTPUT
036C 992 :
036C 993 DZ$STOP:
0200 8F A8 036C 994 BISW #TTY$M_TANK_STOP - ; SCHEDULE STOP
0108 C5 05 0370 995 UCBSW_TT_HOLD(R5)
0373 996 RSB
0374 997 :
0374 998 : ABORT ANY CURRENT PORT OUTPUT ACTIVITY
0374 999 :

```



```
03ED 1057  
03ED 1058 :  
03ED 1059 : RESUME STOPPED OUTPUT  
03ED 1060 :  
03ED 1061 DZ32$RESUME:  
03ED 1062     PUSHR    #^M<R1,R4>                ; SAVE REGISTERS  
03ED 1063 5$:     BICW    #TTY$M_TANK_STOP-      ;  
03ED 1064         ,UCB$W_TT_HOLD(R5)           ; RESET STOP CONDITIONS  
03ED 1065         BBS     #TTY$V_TANK_BURST,UCB$W_TT_HOLD(R5),20$ ; BURST IN PROGRESS  
03ED 1066 10$:    TIMSET  1                       ; CHAR IN TANK OR OTHER  
03ED 1067         BRB     30$                     ; TIME OUT  
03ED 1068  
03ED 1069 20$:    MOVZWL  UCB$W_TT_OUTLEN(R5),R1  
03ED 1070         TIMSET  R1,R1-                   ; SET THE TIMER  
03ED 1071 30$:    BBS     #UCB$V_INT,UCB$W_STS(R5),40$ ; SKIP IF OUTPUT ON  
03ED 1072         MOVL   UCB$L_CRB(R5),R1-         ; ACCESS CRB ADDRESS  
03ED 1073         MOVL   CRB$L_INTD+VEC$SL_IDB(R1),R4 ; GET IDB ADDRESS  
03ED 1074         MOVL   (R4),R1                   ; GET CSR ADDRESS  
03ED 1075         BISB   UCB$W_TT_UNITBIT(R5),IDB$B_TT_ENABLE(R4) ; ENABLE LINE  
03ED 1076         MOVB   IDB$B_TT_ENABLE(R4),? (R1)  
03ED 1077         BBSS   #UCB$V_INT,UCB$W_STS(R5),40$ ; SHOW OUTPUT ACTIVE  
03ED 1078  
03ED 1079 40$:    POPR    #^M<R1,R4>  
03ED 1080         RSB  
03ED 1081  
03ED 1082  
03ED 1083 .ENDC
```

```

03ED 1085      .SBTTL OUTPUT INTERRUPT SERVICE
03ED 1086      :++
03ED 1087      : DZ$INTOUT - DZ-11 OUTPUT INTERRUPT SERVICE
03ED 1088      :
03ED 1089      : FUNCTIONAL DESCRIPTION:
03ED 1090      :
03ED 1091      : THIS ROUTINE IS ENTERED WHEN THE DZ-11 FINDS A LINE ENABLED
03ED 1092      : AND AN EMPTY UART. THE CORRESPONDING UCB IS FOUND AND
03ED 1093      : ANY OUTSTANDING PORT OUTPUT IS DONE. WHEN ALL OUTSTANDING PORT
03ED 1094      : OUTPUT IS COMPLETED, THE CLASS DRIVER IS CALLED TO RETURN THE NEXT
03ED 1095      : CHARACTER OR STRING TO BE OUTPUT. IF NO MORE OUTPUT IS FOUND, THEN
03ED 1096      : THE LINE IS DISABLED.
03ED 1097      :
03ED 1098      : INPUTS:
03ED 1099      :
03ED 1100      :     SP(00) = ADDRESS OF THE IDB
03ED 1101      :
03ED 1102      : IMPLICIT INPUTS:
03ED 1103      :
03ED 1104      :     R0,R1,R2,R3,R4,R5 SAVED ON THE STACK.
03ED 1105      :
03ED 1106      : OUTPUTS:
03ED 1107      :
03ED 1108      :     THE INTERRUPT IS DISMISSED.
03ED 1109      :
03ED 1110      :--
03ED 1111      DZ11_OUT_EXIT:
03ED 1112      ADDL    #4,SP          ; EXIT OUTPUT INTERRUPT
03ED 1113      MOVQ   (SP)+,R0      ; REMOVE IDB ADDRESS
03ED 1114      MOVQ   (SP)+,R2      ; RESTORE REGISTERS
03ED 1115      MOVQ   (SP)+,R4
03ED 1116      REI
03ED 1117      :
03ED 1118      DZ11$INTOUT:
03ED 1119      :
03ED 1120      DZ11_OUT_LOOP:
03ED 1121      MOVL   @ (SP),R4    ; GET THE IDB ADDRESS
03ED 1122      MOVL   (R4),R0      ; GET THE CSR ADDRESS
03ED 1123      :
03ED 1124      : GET THE LINE INFO FROM THE CSR
03ED 1125      :
03ED 1126      :
03ED 1127      MOVW   (R0),R2        ; GET THE CSR VALUE
03ED 1128      BGEQ   DZ11_OUT_EXIT  ; NO MORE LINES
03ED 1129      ASHL   #-8,R2,R2      ; GET THE LINE NUMBER
03ED 1130      BICL   #^C<7>,R2
03ED 1131      MOVL   IDB$UCBLST(R4)[R2],R5 ; GET THE UCB ADDRESS
03ED 1132      BEQL   DZ11_OUT_LOOP ; IF EQL THEN DISMISS
03ED 1133      :
03ED 1134      : CHECK FOR BURST ACTIVE ON LINE
03ED 1135      :
03ED 1136      CMPB   #TTY$M_TANK_BURST@-8,- ; ONLY BURST ACTIVE?
03ED 1137      UCBSW TT HOCD+1(R5)
03ED 1138      BEQL   DZ11_BURST      ; YES, CONTINUE BURST
03ED 1139      :
03ED 1140      : LOOK FOR NEXT OUTPUT STATE IN TANK
03ED 1141      :

```

```

53 0109 C5 06 00 EA 0420 1142
0420 1143 FFS #0,#6,UCBSW_TT_HOLD+1(R5),R3
0427 1144 CASE R3,TYPE=B,<= ; DISPATCH
0427 1145 DZ11_PREMPT,- ; SEND PREMPT CHARACTER
0427 1146 DZ11_STOP,- ; STOP OUTPUT
0427 1147 DZ11_CHAR,- ; CHAR IN TANK
0427 1148 DZ11_BURST,- ; BURST IN PROGRESS
0427 1149 >
0433 1150 :
0433 1151 : NO PENDING DATA - LOOK FOR NEXT CHARACTER
0433 1152 :
64 A5 03 8A 0433 1153 BICB #UCBSM_TIM!UCBSM_INT,UCBSW_STS(R5); CLEAR TIMEOUT AND EXPECTED
0437 1154 :
0437 1155 : CALL CLASS DRIVER FOR MORE OUTPUT
0437 1156 :
01 FF 8F 010C D5 16 0437 1157 JSB @UCBSL_TT_GETNXT(R5) ; GET THE NEXT CHARACTER
01 FF 8F 010B C5 8F 043B 1158 CASEB UCBSB_TT_OUTTYPE(R5),#-1,#1 ; OPTIMIZE FOR THE SINGLE
0442 1159 ; CHARACTER CASE BY SETTING THE
0442 1160 ; LIMIT TO 1
0017' 0442 1161 1$: .WORD DZ11_START_BURST-1$ ; BURST SPECIFIED
000A' 0444 1162 .WORD 50$-1$ ; NONE
0446 1163 :
0446 1164 : OUTPUT A CHARACTER TO THE DZ-11
0446 1165 :
06 A0 53 9B 0446 1166 20$: MOVZBW R3,6(R0) ; OUTPUT CHARACTER
AE 11 044A 1167 BRB DZ11_OUT_LOOP
044C 1168 :
044C 1169 : DISABLE OUTPUT ON THIS LINE
044C 1170 :
044C 1171 50$:
A9 64 A5 01 E0 044C 1172 BBS #UCBSV_INT,- ; IF INT EXP, THEN DON'T RESET,
044E 1173 UCBSW_STS(R5),DZ11_OUT_LOOP ; COULD HAVE BEEN SET DURING CALLBACK
0451 1174
0451 1175
04 A0 0106 C5 AA 0451 1176 BICW UCBSW_TT_UNITBIT(R5),4(R0) ; RESET THE OUTPUT ENABLE
A1 11 0457 1177 BRB DZ11_OUT_LOOP
0459 1178
0459 1179
0800 8F AB 0459 1180 DZ11_START_BURST:
0108 C5 0459 1181 BISW #TTY$M_TANK_BURST,- ; SIGNAL BURST ACTIVE
045D 1182 UCBSW_TT_HOLD(R5)
0460 1183 :
0460 1184 : CONTINUE BURST OUTPUT
0460 1185 :
0460 1186 DZ11_BURST:
011C D5 90 0460 1187 MOVB @UCBSL_TT_OUTADR(R5),- ; OUTPUT NEXT BYTE
06 A0 0464 1188 6(R0)
011C C5 D6 0466 1189 INCL UCBSL_TT_OUTADR(R5) ; UPDATE POINTER
0120 C5 B7 046A 1190 DECW UCBSW_TT_OUTLEN(R5) ; UPDATE COUNT
8A 12 046E 1191 BNEQ DZ11_OUT_LOOP ; NOT LAST CHARACTER
0800 8F AA 0470 1192 BICW #TTY$M_TANK_BURST,- ; RESET BURST ACTIVE
0108 C5 0474 1193 UCBSW_TT_HOLD(R5)
FF80 31 0477 1194 BRW DZ11_OUT_LOOP
047A 1195 :
047A 1196 : OUTPUT SINGLE CHARACTER
047A 1197 :
047A 1198 DZ11_CHAR:

```

```

06 A0 0108 C5 90 047A 1199      MOVB   UCBSW_TT_HOLD(R5),6(R0) ; OUTPUT CHAR IN TANK
      0400 8F AA 0480 1200      BICW   #TTY$M_TANK_HOLD,-      ; SHOW TANK EMPTY
      0108 C5      0484 1201      UCBSW_TT_HOLD(R5)
      FF70 31 0487 1202      BRW   DZ11_OUT_LOOP
      048A 1203      ;
      048A 1204      ; STOP THE OUTPUT
      048A 1205      ;
      048A 1206      DZ11_STOP:
04 A0 0106 C5 8A 048A 1207      BICB   #UCBSM_INT!UCBSM_TIM,-
      64 A5 AA 048C 1208      UCBSW_STS(R5) ; RESET OUTPUT ACTIVE
      FF63 31 048E 1209      BICW   UCBSW_TT_UNITBIT(R5),4(R0) ; RESET THE OUTPUT ENABLE
      FF63 31 0494 1210      BRW   DZ11_OUT_LOOP
      0497 1211      .ENABLE LSB
      0497 1212      ;
      0497 1213      ; SEND XON OR XOFF CHARACTER
      0497 1214      ;
      0497 1215      ;
      0497 1216      ;
      0497 1217      DZ11_PREMPT:
06 A0 0100 8F AA 0497 1218      BICW   #TTY$M_TANK_PREMPT,-      ; RESET XOFF STATE
      0108 C5      049B 1219      UCBSW_TT_HOLD(R5)
      010A C5 90 049E 1220      MOVB   UCBSW_TT_PREMPT(R5),6(R0); OUTPUT CHARACTER
      FF53 31 04A4 1221      BRW   DZ11_OUT_LOOP
      04A7 1222      .DISABLE [SB
      04A7 1223

```

```

04A7 1225 .IF NDF DZV
04A7 1226 :
04A7 1227 : DZ-32 OUTPUT INTERRUPT SERVICE CODE
04A7 1228 :
04A7 1229 :
04A7 1230 DZ32_OUT_EXIT: ; EXIT OUTPUT INTERRUPT
04A7 1231 ADDL #4,SP ; REMOVE IDB ADDRESS
04A7 1232 MOVQ (SP)+,R0 ; RESTORE REGISTERS
04A7 1233 MOVQ (SP)+,R2 ;
04A7 1234 MOVQ (SP)+,R4 ;
04A7 1235 REI ; DISMISS INTERRUPT
04A7 1236 :
04A7 1237 DZ32$INTOUT:: ; DZ-32 OUTPUT INTERRUPT SERVICE
04A7 1238 DZ32_OUT_LOOP:
04A7 1239 MOVL @ (SP),R4 ; GET THE IDB ADDRESS
04A7 1240 MOVL (R4),R0 ; GET THE CSR ADDRESS
04A7 1241 :
04A7 1242 : GET THE LINE INFO FROM THE CSR
04A7 1243 :
04A7 1244 :
04A7 1245 MOVW (R0),R2 ; GET THE CSR VALUE
04A7 1246 BGEQ DZ32_OUT_EXIT
04A7 1247 ASHL #-8,R2,R2 ; GET THE LINE NUMBER
04A7 1248 BICL #^C<7>,R2 ;
04A7 1249 MOVL IDB$L UCBLST(R4)[R2],R5 ; GET THE UCB ADDRESS
04A7 1250 BEQL DZ32_OUT_LOOP ; IF EQL THEN DISMISS
04A7 1251 :
04A7 1252 : CHECK FOR BURST ACTIVE ON LINE
04A7 1253 :
04A7 1254 CMPB #TTYSM_TANK_BURST@-8,- ; ONLY BURST ACTIVE?
04A7 1255 UCBSW TT_HOLD+1(R5)
04A7 1256 BEQL DZ32_BURST ; YES, CONTINUE BURST
04A7 1257 :
04A7 1258 : LOOK FOR NEXT OUTPUT STATE IN TANK
04A7 1259 :
04A7 1260 :
04A7 1261 FFS #0,#6,UCBSW_TT_HOLD+1(R5),R3
04A7 1262 CASE R3,TYPE=B,<= ; DISPATCH
04A7 1263 DZ32_PREMPT,- ; SEND PREMPT CHARACTERS
04A7 1264 DZ32_STOP,- ; STOP OUTPUT
04A7 1265 DZ32_CHAR,- ; CHAR IN TANK
04A7 1266 DZ32_BURST,- ; BURST IN PROGRESS
04A7 1267 >
04A7 1268 :
04A7 1269 : NO PENDING DATA - LOOK FOR NEXT CHARACTER
04A7 1270 :
04A7 1271 BICB #UCBSM_TIM!UCBSM_INT,UCBSW_STS(R5); CLEAR TIMEOUT AND EXPECTED
04A7 1272 :
04A7 1273 : CALL CLASS DRIVER FOR MORE OUTPUT
04A7 1274 :
04A7 1275 JSB @UCBSL_TT_GETNXT(R5) ; GET THE NEXT CHARACTER
04A7 1276 CASEB UCBSB_TT_OUTYPE(R5),#-1,#1 ; OPTIMIZE FOR THE SINGLE
04A7 1277 : CHARACTER CASE BY SETTING THE
04A7 1278 : LIMIT TO 1
04A7 1279 1$: .WORD DZ32_START_BURST-1$ ; BURST SPECIFIED
04A7 1280 .WORD 50$-T$ ; NONE
04A7 1281 ;

```



```

04A7 1282 : OUTPUT A CHARACTER TO THE DZ-32
04A7 1283 :
04A7 1284 20$:   MOVB   R3,6(R0)           ; OUTPUT CHARACTER
04A7 1285       BRW   DZ32_OUT_LOOP
04A7 1286 :
04A7 1287 :
04A7 1288 : DISABLE OUTPUT ON THIS LINE
04A7 1289 :
04A7 1290 50$:   BBS    #UCBSV_INT,-           ; IF INT EXP, THEN DON'T RESET,
04A7 1291       UCBSW_STS(R5),DZ32_OUT_LOOP ; COULD HAVE BEEN SET DURING CALLBACK
04A7 1292 :
04A7 1293 :
04A7 1294 :
04A7 1295       MOVL  @ (SP),R4           ; GET IDB ADDRESS
04A7 1296       BICB  UCBSW_TT_UNITBIT(R5),- ; RESET THE OUTPUT ENABLE
04A7 1297       IDBSB_TT_ENABLE(R4)
04A7 1298       MOVB  IDBSB_TT_ENABLE(R4),7(R0)
04A7 1299       BRW   DZ32_OUT_LOOP
04A7 1300 :
04A7 1301 :
04A7 1302 DZ32_START_BURST:
04A7 1303       BISW  #TTY$M_TANK_BURST,-     ; SIGNAL BURST ACTIVE
04A7 1304       UCBSW_TT_HOLD(R5)
04A7 1305 DZ32_BURST:
04A7 1306       MOVB  @UCBSL_TT_OUTADR(R5),- ; OUTPUT NEXT BYTE
04A7 1307       6(R0)
04A7 1308       INCL  UCBSL_TT_OUTADR(R5)    ; UPDATE POINTER
04A7 1309       DECW  UCBSW_TT_OUTLEN(R5)   ; UPDATE COUNT
04A7 1310       BNEQ  60$                   ; NOT LAST CHARACTER
04A7 1311       BICW  #TTY$M_TANK_BURST,-     ; RESET BURST ACTIVE
04A7 1312       UCBSW_TT_HOLD(R5)
04A7 1313 60$:   BRW   DZ32_OUT_LOOP
04A7 1314 :
04A7 1315 : OUTPUT SINGLE CHARACTER
04A7 1316 :
04A7 1317 DZ32_CHAR:
04A7 1318       MOVB  UCBSW_TT_HOLD(R5),6(R0) ; OUTPUT CHAR IN TANK
04A7 1319       BICW  #TTY$M_TANK_HOLD,-     ; SHOW TANK EMPTY
04A7 1320       UCBSW_TT_HOLD(R5)
04A7 1321       BRW   DZ32_OUT_LOOP
04A7 1322 :
04A7 1323 : STOP OUTPUT
04A7 1324 :
04A7 1325 DZ32_STOP:
04A7 1326       BICB  #UCBSM_INT!UCBSM_TIM,-   ; RESET OUTPUT ACTIVE
04A7 1327       UCBSW_STS(R5)
04A7 1328       MOVL  @ (SP),R4           ; GET IDB ADDRESS
04A7 1329       BICB  UCBSW_TT_UNITBIT(R5),- ; RESET THE OUTPUT ENABLE
04A7 1330       IDBSB_TT_ENABLE(R4)
04A7 1331       MOVB  IDBSB_TT_ENABLE(R4),7(R0)
04A7 1332       BRW   DZ32_OUT_LOOP
04A7 1333 :
04A7 1334       .ENABLE LSB
04A7 1335 :
04A7 1336 : SEND XON OR XOFF
04A7 1337 :
04A7 1338 DZ32_PREMPT:

```

DZVDRIVER
V04-000

- Port Driver for DZV-11 support L 1
OUTPUT INTERRUPT SERVICE

16-SEP-1984 02:23:45 VAX/VMS Macro V04-00
5-SEP-1984 04:15:55 [TTDRVR.SRC]DZDRIVER.MAR;1

Page 31
(1)

```
04A7 1339      BICW  #TTY$M_TANK_PREMPT,-      ; RESET XOFF STATE
04A7 1340      UCBSW TT_HOLD(R5)
04A7 1341      MOVW  UCBSB_TT_PREMPT(R5),6(R0); OUTPUT CHARACTER
04A7 1342      BRW   DZ32_OUT_LOOP
04A7 1343
04A7 1344
04A7 1345      .DISABLE      LSB
04A7 1346      .ENDC
```

```

04A7 1348          .SBTTL SET SPEED, PARITY PARAMETERS
04A7 1349
04A7 1350 :++
04A7 1351 : DZ$SET_LINE - RESET SPEED, PARITY
04A7 1352 :
04A7 1353 : FUNCTIONAL DESCRIPTION:
04A7 1354 :
04A7 1355 : INPUTS:
04A7 1356 :
04A7 1357 :     R5 - UCB ADDRESS
04A7 1358 :
04A7 1359 : OUTPUTS:
04A7 1360 :
04A7 1361 :     R4 USED
04A7 1362 :--
04A7 1363
04A7 1364 DZ$SET_LINE:
01 AE 54 24 A5 D0 04A7 1365      MOVL      UCBSL_CRB(R5),R4          ; ADDRESS CRB
04AB 1366 :
04AB 1367 :     SET UP LINE SPEED AND PARITY
04AB 1368 :
04AB 1369      MOVL      @CRBSL_INTD+VECSL_IDB(R4),R4      ; GET THE CSR ADDRESS VIA CRB
04AF 1370      CLRL      -(SP)                                ; RESET A TEMPORARY LOCATION
04B1 1371      SUBB3     #1,UCBSW_TT_SPEED(R5),1(SP)        ; ADJUST DATA BASE SPEED
04B8 1372      MOVB     UCBSB_TT_PARITY(R5),(SP)          ; SET PARITY, STOP, CHARACTER SIZE
04BD 1373      BICW     #^XF007,(SP)                        ; CLEAR SPECIAL FIELDS
04C2 1374      BBS      #UCBSV_TT_DSBL,-                  ; SKIP CLOCK ENABLE IF LINE DISABLED
04C4 1375      UCBSB_TT_MAINT(R5),3$
04C8 1376      BISW     #<DZLPRSM_CLOCK>,(SP)
04CD 1377 3$:
04CD 1378      BISW     UCBSW_UNIT(R5),(SP)                ; SET LINE NUMBER
04D1 1379      BITW     #DZCSRSM_MODE,(R4)                ; DZ32 CONTROLLER?
04D4 1380      BNEQ    10$
04D6 1381 5$:
04D6 1382      CVTLW   (SP)+,2(R4)                        ; INSERT AS LINE PARAMETER
04DA 1383      RSB
04DB 1384 :
04DB 1385 :     HANDLE DZ-32 SPECIFIC FUNCTIONS
04DB 1386 10$:
04DB 1387      CMPB     UCBSW_TT_SPEED(R5) -              ; TRANSMIT/RECEIVE THE SAME
04DF 1388      UCBSW_TT_SPEED+1(R5)
04E2 1389      BEQL    5$
04E4 1390      TSTB    UCBSW_TT_SPEED+1(R5)              ; YES, NO SPLIT SPEED
04E8 1391      BEQL    5$
04EA 1392 :
04EA 1393 :     SET SPLIT SPEED
04EA 1394 :
04EA 1395      BISW     #DZLPRSM_SPLIT,(SP)                ; SET SPLIT SPEED BIT
04EF 1396      BRB     5$
04F1 1397      ; COMPLETE SETUP

```

```

04F1 1399
04F1 1400 .SBTTL INITIALIZE DZ-11 MODEM POLLING
04F1 1401
04F1 1402 :++
04F1 1403 : DZ$SET_MODEM - INIT MODEM POLLING
04F1 1404 :
04F1 1405 : FUNCTIONAL DESCRIPTION:
04F1 1406 :
04F1 1407 : INIT DZ-11 MODEM TRANSITION POLLING IF NOT ALREADY ACTIVE. LINK CRB
04F1 1408 : FOR CURRENT LINE INTO MODEM TRANSITION POLLING LIST
04F1 1409 :
04F1 1410 : INPUTS:
04F1 1411 :
04F1 1412 : R5 - UCB ADDRESS
04F1 1413 :
04F1 1414 : OUTPUTS:
04F1 1415 :
04F1 1416 : R0-R4 USED
04F1 1417 :--
04F1 1418
04F1 1419 DZ11$SET MODEM:
54 24 A5 DO 04F1 1420 MOVL UCBSL_CRB(R5),R4 ; ADDRESS CRB
00000000'EF D5 04F5 1421 TSTL DZ$DIALUP ; DZ-11 POLLING ALREADY ACTIVE?
18 12 04FB 1422 BNEQ 5$ ; YES, SKIP STARTUP
38 BB 04FD 1423 PUSHR #*M<R3,R4,R5>
55 00000004'EF DE 04FF 1424 MOVAL DZ$TIMQUENT,R5 ; ADDRESS OF TIMER ENTRY
0B A5 06 90 0506 1425 MOVB #IPL$_QUEUEAST,TQESB_RQTYPE(R5) ; SET FORK IPL
00000513'EF 9F 050A 1426 PUSHAB 4$ ; RETURN ADDRESS
0021 31 0510 1427 BRW 30$ ; QUEUE FORK
38 BA 0513 1428 4$: POPR #*M<R3,R4,R5>
51 53 18 A4 DE 0515 1429 5$:
00000000'EF DE 0519 1430 MOVAL CRBSL_DZ_MODEM(R4),R3 ; ADDRESS OF DZ CRB THREAD
52 51 DO 0520 1431 MOVAL DZ$DIALUP,R1 ; ADDRESS OF DZ TIMER LIST HEAD
0523 1432 MOVL R1,R2
0523 1433 :
0523 1434 : LINK CRB INTO DZ-11 MODEM POLLER LIST IF NEEDED
0523 1435 :
0523 1436 10$:
53 62 D1 0523 1437 CMPL (R2),R3 ; IS CRB ON LIST
0B 13 0526 1438 BEQL 20$ ; YES, DONE
52 62 DU 0528 1439 MOVL (R2),R2 ; POINT TO NEXT CRB
F6 12 052B 1440 BNEQ 10$ ; LOOK FOR NEXT
63 61 DO 052D 1441 MOVL (R1),(R3) ; LINK CRB AT LIST HEAD
61 53 DO 0530 1442 MOVL R3,(R1)
05 0533 1443 20$: RSB
0534 1444 30$:
00000000'GF 16 0534 1445 JSB G^EXES$FORK ; FORK TO QUEUE TIMER ENTRY
053A 1447 DSBINT #IPL$ SYNCH
OC A5 FCA7 CF 9E 0540 1448 MOVAB W^DZ$TIMER,TQESL_FPC(R5) ; ADDRESS OF TIMER SERVICE ROUTINE
20 A5 00000000'GF DO 0546 1449 MOVL G^TTY$GL_DELTA,TQESQ_DELTA(R5) ; INTERVAL IS SYSGEN PARAMETER
054E 1450
50 0B A5 05 90 054E 1451 MOVB #TQESC_SSREPT,TQESB_RQTYPE(R5)
50 00000000'GF 7D 0552 1452 MOVQ G^EXES$GQ_SYSTIME,R0
00000000'GF CO 0559 1453 ADDL G^TTY$GL_DELTA,R0
51 00 0560 1454 ADWC #0,R1
00000000'GF 16 0563 1455 JSB G^EXES$INSTIMQ ; INSERT INTO TIMER QUEUE

```

DZVDRIIVER
V04-000

- Port Driver for DZV-11 support B 2
INITIALIZE DZ-11 MODEM POLLING

05 0569 1456 ENBINT
056C 1457 RSB
056D 1458

16-SEP-1984 02:23:45 VAX/VMS Macro V04-00
5-SEP-1984 04:15:55 [TTDRVR.SRC]DZVDRIIVER.MAR;1

Page 34
(1)

; RESTORE IPL

TT
VO

```
056D 1460
00000000 1461 .PSECT $$$117_DATA,QUAD
0000 1462 :
0000 1463 : DZ-11 TIMER DATA STRUCTURES FOR DETECTION OF DZ-11 MODEM TRANSITIONS
0000 1464 :
0000 1465 :
00000000 0000 1466 DZ$SL_DIALUP:
0000 1467 .LONG 0 ; LINKED LIST OF DZ-11 CRB'S
0004 1468 ; USING MODEM CONTROL
0004 1469 DZ$TIMQUENT: ; TIMER QUE ENTRY USED TO
0004 1470 ; SAMPLE DZ-11 MODEM SIGNALS
0004 1471 ; ON PERIODIC BASIS
00000038 0004 1472 .ALIGN QUAD
0008 1473 .BLKB TQESC_LENGTH
0038 1474 STO_TQE TQESW_SIZE,WORD,TQESC_LENGTH,DZ$TIMQUENT
0038 1475 STO_TQE TQESB_TYPE,BYTE,DYN$C_TQE,DZ$TIMQUENT
0038 1476 STO_TQE TQESB_RQTYPE,BYTE,TQESC_SSREPT,DZ$TIMQUENT
0038 1477
0038 1478
0038 1479 DZ$END: ; End of driver
0038 1480
0038 1481 .END
```

DZVDRIVER
Symbol table

- Port Driver for DZV-11 support

D 2

16-SEP-1984 02:23:45 VAX/VMS Macro V04-00
5-SEP-1984 04:15:55 [TTDRVR.SRC]DZDRIVER.MAR;1

Page 36
(1)

SSS	= 00000020	R	02	DZ11\$INTINP	0000028E	RG	03
SSSSSS	= 00000038	R	04	DZ11\$INTOUT	000003FA	RG	03
SSOP	= 00000002			DZ11\$MAINT	0000018A	R	03
ATS_LIBA	= 00000001			DZ11\$RESUME	0000038E	R	03
BIT...	= 00000010			DZ11\$SET MODEM	000004F1	R	03
CLASS_DDT	= 00000010			DZ11\$STARTIO	0000031C	RG	03
CLASS_DS_TRAN	= 0000000C			DZ11\$VEC	00000038	R	03
CLASS_GETNXT	= 00000000			DZ11\$VECEND	00000070	R	03
CLASS_POWERFAIL	= 00000020			DZ11\$XOFF	00000342	R	03
CLASS_PUTNXT	= 00000004			DZ11\$XON	00000342	R	03
CLASS_READERROR	= 00000014			DZ11_BURST	00000460	R	03
CLASS_SETUP_UCB	= 00000008			DZ11_CHAR	0000047A	R	03
CRBSB_DZ_CARRIER	= 0000001D			DZ11_OUT_EXIT	000003ED	R	03
CRBSB_DZ_DTR	= 0000001E			DZ11_OUT_LOOP	000003FA	R	03
CRBSB_DZ_RING	= 0000001C			DZ11_PREMPT	00000497	R	03
CRBSB_TT_TYPE	= 0000000B			DZ11_START_BURST	00000459	R	03
CRBSL_DZ_MODEM	= 00000018			DZ11_STOP	0000048A	R	03
CRBSL_INTD	= 00000024			DZCSRSM_CLEAR	= 00000010		
CRBSL_INTD2	= 00000048			DZCSRSM_DS_CHG	= 00000800		
DCS_TERM	= 00000042			DZCSRSM_DS_ENAB	= 00000002		
DDBSL_DDT	= 0000000C			DZCSRSM_LINE	= 00000700		
DEVSM_AVL	= 00040000			DZCSRSM_MAINT	= 00000008		
DEVSM_CCL	= 00000002			DZCSRSM_MASTENAB	= 00000020		
DEVSM_IDV	= 04000000			DZCSRSM_MODE	= 00000001		
DEVSM_NNM	= 00000200			DZCSRSM_RCVINT	= 00000040		
DEVSM_ODV	= 08000000			DZCSRSM_RCVRDY	= 00000080		
DEVSM_REC	= 00000001			DZCSRSM_SNDINT	= 00004000		
DEVSM_TRM	= 00000004			DZCSRSM_SNDRDY	= 00008000		
DPTSC_LENGTH	= 00000038			DZCSRSS_CLEAR	= 00000001		
DPTSC_VERSION	= 00000004			DZCSRSS_DS_CHG	= 00000001		
DPTSINITAB	= 00000038	R	02	DZCSRSS_DS_ENAB	= 00000001		
DPTSM_NOUNLOAD	= 00000004			DZCSRSS_LINE	= 00000003		
DPTSREINITAB	= 000000D3	R	02	DZCSRSS_MAINT	= 00000001		
DPTSTAB	= 00000000	R	02	DZCSRSS_MASTENAB	= 00000001		
DPTSW_VECTOR	= 0000001E			DZCSRSS_MODE	= 00000001		
DTS_DZ11	= 00000042			DZCSRSS_RCVINT	= 00000001		
DYN\$C_CRB	= 00000005			DZCSRSS_RCVRDY	= 00000001		
DYN\$C_DDB	= 00000006			DZCSRSS_SNDINT	= 00000001		
DYN\$C_DPT	= 0000001E			DZCSRSS_SNDRDY	= 00000001		
DYN\$C_ORB	= 00000049			DZCSR\$V_CLEAR	= 00000004		
DYN\$C_TQE	= 0000000F			DZCSR\$V_DS_CHG	= 0000000B		
DYN\$C_UCB	= 00000010			DZCSR\$V_DS_ENAB	= 00000001		
DZ\$ABORT	= 00000374	R	03	DZCSR\$V_LINE	= 00000008		
DZ\$CTRL_ERROR	= 000000F4	R	03	DZCSR\$V_MAINT	= 00000003		
DZ\$DDT	= 00000000	RG	03	DZCSR\$V_MASTENAB	= 00000005		
DZ\$DPT	= 00000000	RG	02	DZCSR\$V_MODE	= 00000000		
DZ\$END	= 00000038	R	04	DZCSR\$V_RCVINT	= 00000006		
DZ\$INITIAL	= 00000075	RG	03	DZCSR\$V_RCVRDY	= 00000007		
DZ\$INITLINE	= 000000F5	RG	03	DZCSR\$V_SNDINT	= 0000000E		
DZ\$L_DIALUP	= 00000000	R	04	DZCSR\$V_SNDRDY	= 0000000F		
DZ\$NOLL	= 00000074	R	03	DZLCS1\$M_ACK	= 00008000		
DZ\$SET_LINE	= 000004A7	R	03	DZLCS1\$S_ACK	= 00000001		
DZ\$STOP	= 0000036C	R	03	DZLCS1\$V_ACK	= 0000000F		
DZ\$TIMER	= 000001EB	R	03	DZLPR\$M_CLOCK	= 00001000		
DZ\$TIMQUENT	= 00000004	R	04	DZLPR\$M_LINE	= 00000007		
DZ\$UNIT_ERROR	= 00000185	R	03	DZLPR\$M_ODD	= 00000080		
DZ11\$DS_SET	= 00000183	R	03	DZLPR\$M_PARITY	= 00000040		

DZVDRIVER
Symbol table

- Port Driver for DZV-11 support E 2

16-SEP-1984 02:23:45 VAX/VMS Macro V04-00
5-SEP-1984 04:15:55 [TTDRVR.SRC]DZDRIVER.MAR;1

Page 37
(1)

DZLPRSM_SIZE = 00000018
 DZLPRSM_SPEED = 00000F00
 DZLPRSM_SPLIT = 00002000
 DZLPRSM_STOP = 00000020
 DZLPRSS_CLOCK = 00000001
 DZLPRSS_LINE = 00000003
 DZLPRSS_ODD = 00000001
 DZLPRSS_PARITY = 00000001
 DZLPRSS_SIZE = 00000002
 DZLPRSS_SPEED = 00000004
 DZLPRSS_SPLIT = 00C00001
 DZLPRSS_STOP = 00000001
 DZLPRSV_CLOCK = 0000000C
 DZLPRSV_LINE = 00000000
 DZLPRSV_ODD = 00000007
 DZLPRSV_PARITY = 00000006
 DZLPRSV_SIZE = 00000003
 DZLPRSV_SPEED = 00000008
 DZLPRSV_SPLIT = 0000000D
 DZLPRSV_STOP = 00000005
 DZRCVSM_BUF = 000000FF
 DZRCVSM_FRAMER = 00002000
 DZRCVSM_LINE = 00000700
 DZRCVSM_OVERRUN = 00004000
 DZRCVSM_PARERR = 00001000
 DZRCVSM_VALID = 00008000
 DZRCVSS_BUF = 00000008
 DZRCVSS_FRAMER = 00000001
 DZRCVSS_LINE = 00000003
 DZRCVSS_OVERRUN = 00000001
 DZRCVSS_PARERR = 00000001
 DZRCVSS_VALID = 00000001
 DZRCVSV_BUF = 00000000
 DZRCVSV_FRAMER = 0000000D
 DZRCVSV_LINE = 00000008
 DZRCVSV_OVERRUN = 0000000E
 DZRCVSV_PARERR = 0000000C
 DZRCVSV_VALID = 0000000F
 DZV = 00000001
 EX\$FORK ***** X 03
 EX\$GL_ABSTIM ***** X 03
 EX\$GL_TENUSEC ***** X 03
 EX\$GL_UBDELAY ***** X 03
 EX\$GQ_SYSTIME ***** X 03
 EX\$INSTIMQ ***** X 03
 FUNCTAB_LEN = 00000000
 IDBSL_UCBLST = 00000018
 IOSM_LINE_OFF = 00000200
 IOSM_LINE_ON = 00000800
 IOCSANTVER ***** X 03
 IOCSRETURN ***** X 03
 IPL\$_QUEUEAST = 00000006
 IPL\$_SYNCH = 00000008
 MODEMSC_DATASET = 00000003
 MODEMSC_INIT = 00000000
 ORBSB_FLAGS = 0000000B
 ORBSL_OWNER = 00000000

ORBSM_PROT_16 = 00000001
 ORBSW_PROT = 00000018
 PORT_ABORT = 00000020
 PORT_DS_SET = 0000000C
 PORT_LENGTH = 00000038
 PORT_MAINT = 00000030
 PORT_RESUME = 00C00024
 PORT_SET_LINE = 00000008
 PORT_SET_MODEM = 00000028
 PORT_STARTIO = 00000000
 PORT_STOP = 00000018
 PORT_VECTOR = 00000038
 PORT_XOFF = 00000014
 PORT_XON = 00000010
 PR\$_IPL = 00000012
 SIZ... = 00000001
 SSS_NORMAL = 00000001
 TQ\$B_RQTYPE = 0000000B
 TQ\$B_TYPE = 0000000A
 TQ\$C_LENGTH = 00000030
 TQ\$C_SSREPT = 00000005
 TQ\$C_FPC = 0000000C
 TQ\$Q_DELTA = 00000020
 TQ\$W_SIZE = 00000008
 TTSM_DS_CTS = 00000010
 TTSM_DS_DSR = 00000080
 TTSV_DS_CARRIER = 00000005
 TTSV_DS_DTR = 00000001
 TTSV_DS_RING = 00000006
 TTSV_MODEM = 00000015
 TTS_UNKNOWN = 00000000
 TTY\$GB_DEFSPEED ***** X 02
 TTY\$GB_PARITY ***** X 02
 TTY\$GB_RSPEED ***** X 02
 TTY\$GL_DEFCHAR ***** X 02
 TTY\$GL_DEFCHAR2 ***** X 02
 TTY\$GL_DELTA ***** X 03
 TTY\$GL_DPT ***** X 03
 TTY\$GL_OWNUIC ***** X 02
 TTY\$GW_DEFBUF ***** X 02
 TTY\$GW_PROT ***** X 02
 TTYSM_TANK_BURST = 00000800
 TTYSM_TANK_HOLD = 00000400
 TTYSM_TANK_PREMPT = 00000100
 TTYSM_TANK_STOP = 00000200
 TTYSV_PC_NOTIME = 00000000
 TTYSV_TANK_BURST = 0000000B
 UCBSB_DEVCLASS = 00000040
 UCBSB_DEVTYPE = 00000041
 UCBSB_DIPL = 0000005E
 UCBSB_FIPL = 0000000B
 UCBSB_TT_DEPARI = 000000EC
 UCBSB_TT_DETYPE = 000000F0
 UCBSB_TT_DS_RCV = 00000124
 UCBSB_TT_DS_TX = 00000125
 UCBSB_TT_MAINT = 0000012A
 UCBSB_TT_OUTTYPE = 0000010B

R 03

X 02
 X 02
 X 02
 X 02
 X 02
 X 03
 X 03
 X 02
 X 02
 X 02

DZVDRIVER
Symbol table

- Port Driver for DZV-11 support F 2

16-SEP-1984 02:23:45 VAX/VMS Macro V04-00
5-SEP-1984 04:15:55 [TTDRVR.SRC]DZDRIVER.MAR;1

Page 38
(1)

```
UCBSB_TT_PARITY      = 000000F8
UCBSB_TT_PREMPT     = 0000010A
UCBSC_TT_LENGTH     = 00000134
UCBSL_CRB           = 00000024
UCBSL_DDB           = 00000028
UCBSL_DDT           = 00000088
UCBSL_DEVCHAR       = 00000038
UCBSL_DEVCHAR2      = 0000003C
UCBSL_DEVDEPEND     = 00000044
UCBSL_DEVDEPN2      = 00000048
UCBSL_DUETIM        = 0000006C
UCBSL_TT_CLASS      = 00000114
UCBSL_TT_DECHA1     = 000000C8
UCBSL_TT_DECHAR     = 000000C4
UCBSL_TT_GETNXT     = 0000010C
UCBSL_TT_OUTADR     = 0000011C
UCBSL_TT_PORT       = 00000118
UCBSL_TT_PUTNXT     = 00000110
UCBSL_TT_RTIMOU     = 000000B4
UCBSL_TT_WBLINK     = 000000D0
UCBSL_TT_WFLINK     = 000000CC
UCBSM_INT           = 00000002
UCBSM_ONLINE        = 00000010
UCBSM_TIM           = 00000001
UCBSM_TT_DSBL       = 00000080
UCBSV_INT           = 00000001
UCBSV_POWER         = 00000005
UCBSV_TT_DSBL       = 00000007
UCBSW_DEVBUFSIZ     = 00000042
UCBSW_STS           = 00000064
UCBSW_TT_DESIZE     = 000000F1
UCBSW_TT_DESPEE     = 000000E8
UCBSW_TT_HOLD       = 00000108
UCBSW_TT_OUTLEN     = 00000120
UCBSW_TT_PRTCTL     = 00000122
UCBSW_TT_SPEED      = 000000F4
UCBSW_TT_UNITBIT    = 00000106
UCBSW_UNIT          = 00000054
VECSL_IDB           = 00000008
VECSL_INITIAL       = 0000000C
VECSL_UNITINIT      = 00000018
```

! Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABS\$	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
\$\$\$105_PROLOGUE	000000DE (222.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
\$\$\$115_DRIVER	0000056D (1389.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
\$\$\$117_DATA	00000038 (56.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC QUAD

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
-----	-----	-----	-----
Initialization	35	00:00:00.05	00:00:00.24
Command processing	116	00:00:00.46	00:00:01.39
Pass 1	696	00:00:22.09	00:00:44.74
Symbol table sort	0	00:00:03.25	00:00:07.03
Pass 2	249	00:00:04.45	00:00:09.34
Symbol table output	31	00:00:00.21	00:00:00.43
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1131	00:00:30.53	00:01:03.19

The working set limit was 2250 pages.
178134 bytes (348 pages) of virtual memory were used to buffer the intermediate code.
There were 160 pages of symbol table space allocated to hold 3025 non-local and 60 local symbols.
1482 source lines were read in Pass 1, producing 22 object records in Pass 2.
72 pages of virtual memory were used to define 67 macros.

! Macro library statistics !

Macro library name	Macros defined
-----	-----
-\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	33
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2	12
TOTALS (all libraries)	45

3540 GETS were required to define 45 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:DZVDRIVER/OBJ=OBJ\$:DZVDRIVER MSRCS\$:DZV/UPDATE=(ENHS\$:DZV)+MSRCS\$:DZDRIVER/UPDATE=(ENHS\$:DZDRIVER)+EXECMLS/LIB

The image displays a grid of 15 columns and 12 rows of small, illegible text fragments. These fragments appear to be individual pages or sections of a document, possibly a manual or a data set, arranged in a grid. The text is too small to read clearly, but some fragments are more legible than others, showing headers like "TTYFOT LIS", "TTYCHARO LIS", "TTYCHARI LIS", and "TTYDRUDAT LIS". The overall appearance is that of a large document or data set, possibly a manual or a data set, arranged in a grid.