```
BBBBBBBBBBBB        OOOOOOOOO        OOOOOOOOO    TTTTTTTTTTTTTTT     SSSSSSSSSSS
BBBBBBBBBBBB        OOOOOOOOO        OOOOOOOOO    TTTTTTTTTTTTTTT     SSSSSSSSSSS
BBBBBBBBBBBB        OOOOOOOOO        OOOOOOOOO    TTTTTTTTTTTTTTT     SSSSSSSSSSS
BBB         BBB  OOO         OOO  OOO         OOO       TTT         SSS
BBB         BBB  OOO         OOO  OOO         OOO       TTT         SSS
BBB         BBB  OOO         OOO  OOO         OOO       TTT         SSS
BBB         BBB  OOO         OOO  OOO         OOO       TTT         SSS
BBB         BBB  OOO         OOO  OOO         OOO       TTT         SSS
BBB         BBB  OOO         OOO  OOO         OOO       TTT         SSS
BBBBBBBBBBBB     OOO         OOO  OOO         OOO       TTT            SSSSSSSSS
BBBBBBBBBBBB     OOO         OOO  OOO         OOO       TTT            SSSSSSSSS
BBBBBBBBBBBB     OOO         OOO  OOO         OOO       TTT            SSSSSSSSS
BBB         BBB  OOO         OOO  OOO         OOO       TTT                     SSS
BBB         BBB  OOO         OOO  OOO         OOO       TTT                     SSS
BBB         BBB  OOO         OOO  OOO         OOO       TTT                     SSS
BBB         BBB  OOO         OOO  OOO         OOO       TTT                     SSS
BBB         BBB  OOO         OOO  OOO         OOO       TTT                     SSS
BBB         BBB  OOO         OOO  OOO         OOO       TTT                     SSS
BBB         BBB  OOO         OOO  OOO         OOO       TTT                     SSS
BBBBBBBBBBBB        OOOOOOOOO        OOOOOOOOO         TTT         SSSSSSSSSSS
BBBBBBBBBBBB        OOOOOOOOO        OOOOOOOOO         TTT         SSSSSSSSSSS
BBBBBBBBBBBB        OOOOOOOOO        OOOOOOOOO         TTT         SSSSSSSSSSS
```


DDDDDDDD LL BBBBBBBB TTTTTTTTTT DDDDDDDD RRRRRRRR IIIIII VV VV RRRRRRRR
DDDDDDDD LL BBBBBBBB TTTTTTTTTT DDDDDDDD RRRRRRRR IIIIII VV VV RRRRRRRR
DD DD LL BB BB TT DD DD RR RR II VV VV RR RR
DD DD LL BB BB TT DD DD RR RR II VV VV RR RR
DD DD LL BB BB TT DD DD RR RR II VV VV RR RR
DD DD LL BB BB TT DD DD RR RR II VV VV RR RR
DD DD LL BBBBBBBB TT DD DD RRRRRRRR II VV VV RRRRRRRR
DD DD LL BBBBBBBB TT DD DD RRRRRRRR II VV VV RRRRRRRR
DD DD LL BB BB TT DD DD RR RR II VV VV RR RR
DD DD LL BB BB TT DD DD RR RR II VV VV RR RR
DD DD LL BB BB TT DD DD RR RR II VV VV RR RR
DD DD LL BB BB TT DD DD RR RR II VV VV RR RR
DDDDDDDD LLLLLLLLLL BBBBBBBB TT DDDDDDDD RR RR IIIIII VV RR RR
DDDDDDDD LLLLLLLLLL BBBBBBBB TT DDDDDDDD RR RR IIIIII VV RR RR
LL IIIIII SSSSSSSS
LL IIIIII SSSSSSSS
LL II SS
LL II SS
LL II SS
LL II SS
LL II SSSSSS
LL II SSSSSS
LL II SS
LL II SS
LL II SS
LL II SS
LLLLLLLLLL IIIIII SSSSSSSS
LLLLLLLLLL IIIIII SSSSSSSS

Table of conterts

```
0000     1          .TITLE  DLBTDRIVR - RL01/2 BOOT 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
0000    29 ;++
0000    30 ; FACILITY:     BOOTS
0000    31 ;
0000    32 ; ABSTRACT:
0000    33 ;       This module contains the bootstrap device driver for
0000    34 ;       the RL01/2  disks.
0000    35 ;
0000    36 ; ENVIRONMENT:  IPL 31, kernel mode, code must be PIC
0000    37 ;
0000    38 ; AUTHOR:  Steve Beckhardt,     CREATION DATE:  31-Oct-1979
0000    39 ;  (Original author:  Charlie Franks)
0000    40 ;
0000    41 ; MODIFIED BY:
0000    42 ;
0000    43 ;       02-03   GRR2003         G. R. Robert            11-JUN-1981
0000    44 ;               Fixed get status code to test status bits properly
0000    45 ;
0000    46 ;       02-02   CAS0001         C.A. Samuelson          30-Apr-1980
0000    47 ;               Change interface to BOOTDRIVR for purge of UBA datapath
0000    48 ;
0000    49 ;--
```

```
0000     51          .SBTTL  DECLARATIONS
0000     52 ;
0000     53 ; INCLUDE FILES:
0000     54 ;
0000     55
0000     56          $BTDDEF                         ; Boot device types
0000     57          $IODEF                          ; I/O function codes
0000     58          $RPBDEF                         ; RPB offsets
0000     59          $SSDEF                          ; Status codes
0000     60          $UBADEF                         ; UBA definitions
0000     61          $UBIDEF                         ; 11/750 UBA definitions
0000     62
0000     63 ;
0000     64 ; MACROS:
0000     65 ;
0000     66
0000     67 ;
0000     68 ; EQUATED SYMBOLS:
0000     69 ;
0000     70 ;
0000     71 ; RL11/RL02 CONTROLLER REGISTER OFFSETS
0000     72 ;
0000     73
0000     74          $DEFINI RL                      ;START OF REGISTER DEFINITIONS
0000     75
0000     76 $DEF     RL_CS           .BLKW   1       ;CONTROL STATUS REGISTER (CSR)
0002     77          _VIELD  RL_CS,0,<-              ;START OF CSR BIT DEFINITIONS
0002     78                  <DRDY,,M>,-             ; DRIVER READY
0002     79                  <FCODE,3>,-             ; FUNCTION CODE
0002     80                  <XBA,2>,-               ; BUS ADDRESS EXTENSION BITS
0002     81                  <IE,,M>,-               ; INTERRUPT ENABLE
0002     82                  <CRDY,,M>,-             ; CONTROLLER READY
0002     83                  <DS,2>,-                ; DRIVE SELECT
0002     84                  <OPI,,M>,-              ; OPERATION INCOMPLETE
0002     85                  <CRC,,M>,-              ; DATA CRC OR HEADER CRC
0002     86                  <DLT,,M>,-              ; DATA LATE OR HEADER NOT FOUND
0002     87                  <NXM,,M>,-              ; NON-EXISTENT MEMORY
0002     88                  <DE,,M>,-               ; DRIVE ERROR
0002     89                  <CE,,M>-                ; COMPOSITE ERROR
0002     90          >                               ;END CSR BIT DEFINITIONS
0002     91
0002     92 $DEF     RL_BA           .BLKW   1       ;BUS ADDRESS REGISTER (BAR)
0004     93
0004     94 $DEF     RL_DA           .BLKW   1       ;DISK ADDRESS REGISTER (DAR)
0006     95          _VIELD  RL_DA,0,<-              ;START OF DAR BIT DEFINITIONS
0006     96                  <MRK,,M>,-              ; MARK (ALWAYS 1)
0006     97                  <STS,,M>,-              ; GET STATUS
0006     98                  <,1>,-                  ; RESERVED BIT
0006     99                  <RST,,M>,-              ; RESET
0006    100                  <,12>-                  ; RESERVED BITS
0006    101          >                               ;END OF DAR BIT DEFINITIONS
0006    102
0006    103 $DEF     RL_MP           .BLKW   1       ;MULTIPURPOSE REGISTER (MPR)
0008    104          _VIELD  RL_MP,0,<-              ;START OF MPR BIT DEFINITIONS
0008    105                  <STA,3>,-               ; DRIVE STATE
0008    106                  <BH,,M>,-               ; BRUSH HOME
0008    107                  <HO,,M>,-               ; HEADS OUT
```

```
0008    108                 <CO,,1>,-                   ; COVER OPEN
0008    109                 <HS,,M>,-                   ; HEAD SELECT
0008    110                 <TYP,,M>,-                  ; DRIVE TYPE
0008    111                 <DSE,,M>,-                  ; DRIVE SELECT ERROR
0008    112                 <VC,,M>,-                   ; VOLUME CHECK
0008    113                 <WGE,,M>,-                  ; WRITE GATE ERROR
0008    114                 <SPE,,M>,-                  ; SPIN ERROR
0008    115                 <SKTO,,M>,-                 ; SEEK TIME OUT
0008    116                 <WL,,M>,-                   ; WRITE LOCK
0008    117                 <CHE,,M>,-                  ; CURRENT HEAD ERROR
0008    118                 <WDE,,M>-                   ; WRITE DATA ERROR
0008    119             >                               ;END OF MPR BIT DEFINITIONS
0008    120
0008    121             $DEFEND RL                      ;END RL11/RL02 REGISTER DEFINITIONS
0000    122
0000    123
0000    124 ;
0000    125 ; OWN STORAGE:
0000    126 ;
0000    127
0000    128 ;
0000    129 ; Boot driver table entry
0000    130 ;
0000    131
0000    132             $BOOT_DRIVER    DEVTYPE = BTD$K_DL,-    ; Device type (DL)
0000    133                             SIZE = DL_DRVSIZ,-      ; Driver size
0000    134                             ADDR = DL_DRIVER,-      ; Driver address
0000    135                             DRIVRNAME = DLNAME      ; Driver file name
```

```
                                0000   137          .SBTTL  RL01/2 Bootstrap driver code
                                0000   138
                                0000   139 ;++
                                0000   140 ;
                                0000   141 ; Inputs:
                                0000   142 ;
                                0000   143 ;       R3      - base address of adapter's register space
                                0000   144 ;       R5      - LBN FOR CURRENT PIECE OF TRANSFER
                                0000   145 ;       R6      - contains 0
                                0000   146 ;       R7      - address of device's CSR
                                0000   147 ;       R8      - SIZE OF TRANSFER IN BYTES
                                0000   148 ;       R9      - address of the RPB
                                0000   149 ;       R10     - starting address of transfer (byte offset in first
                                0000   150 ;                 page ORed with starting map register number)
                                0000   151 ;
                                0000   152 ;       FUNC(AP)- I/O operation (IO$_READLBLK or IO$_WRITELBLK only)
                                0000   153 ;       BUF(AP) - Buffer address
                                0000   154 ;       SIZE(AP)- Size of transfer
                                0000   155 ;
                                0000   156 ; Implicit inputs:
                                0000   157 ;
                                0000   158 ;       RPB$W_UNIT      - RPB field containing boot device unit number
                                0000   159 ;
                                0000   160 ; Outputs:
                                0000   161 ;
                                0000   162 ;       R0 - status code
                                0000   163 ;
                                0000   164 ;               SS$_NORMAL      - successful transfer
                                0000   165 ;               SS$_CTRLERR     - fatal controller error
                                0000   166 ;
                                0000   167 ;       R3 - must be preserved
                                0000   168 ;
                                0000   169 ;       This routine destroys R1, R2, R4, R5, R6. Within the
                                0000   170 ;       routine, register usage is as follows:
                                0000   171
                                0000   172 ;--
                                0000   173
                   00000004     0000   174 BUF = 4
                   00000008     0000   175 SIZE = 8
                   00000010     0000   176 FUNC = 16
                                0000   177
                                0000   178 DL_DRIVER:
                                0000   179
                                0000   180 ;
                                0000   181 ; RESET DRIVE AND WAIT FOR IT TO SPIN UP.
                                0000   182 ;
                                0000   183
                          50    0000   184          CLRL    R0                              ; CLEAR R0
50   02    08    64 A9   F0    0002   185          INSV    RPB$W_UNIT(R9),#8,#2,R0         ; GET UNIT NUMBER
            04 A7   0B   B0    0008   186          MOVW    #RL_DA_M_RST!-                  ; PUT RESET & GET STATUS IN DAR
                                000C   187                  RL_DA_M_STS!-
                                000C   188                  RL_DA_M_MRK,RL_DA(R7)
         67   50    04   A9    000C   189 10$:     BISW3   #4,R0,RL_CS(R7)                 ; EXECUTE DRIVE RESET
                   00FD   30    0010   190          BSBW    READY                           ; WAIT FOR CONTROLLER READY
           56    06 A7   3C    0013   191          MOVZWL  RL_MP(R7),R6                    ; FETCH STATUS WORD
   1D      56    05   00   ED    0017   192          CMPZV   #0,#5,R6,-                      ; TEST STATUS BITS 04:00
                                001C   193                  #RL_MP_M_HO!-                   ; HEADS,BRUSHES,STATE OK?
```

```
                                001C   194                     RL_MP_M_BH!5                    ; ... (5 = SEEK LINEAR MODE STATE)
                          EE    001C   195             BNEQ    10$                             ; BRANCH IF NOT: WAIT FOR DRIVE TO SPIN UP
                    67    01    B3     001E   196             BITW    #RL_CS_M_DRDY,RL_CS(R7) ; IS DRIVE READY?
                          E9    13     0021   197             BEQL    10$                             ; IF NOT, BRANCH TO WAIT FOR DRIVE READY
                                0023   198
                                0023   199 ;
                                0023   200 ; FIND CURRENT DISK ADDRESS, CALCULATE CYLINDER DIFFERENCE, AND SEEK
                                0023   201 ; DESIRED CYLINDER
                                0023   202 ;
                                0023   203
                67    50  08    A9     0023   204 20$:     BISW3   #8,R0,RL_CS(R7)                 ; EXECUTE READ HEADER
                          00E6  30     0027   205             BSBW    READY                           ; WAIT FOR CONTROLLER READY
                          03    18     002A   206             BGEQ    30$                             ; BRANCH IF NO ERROR READING HEADER
                          00C8  31     002C   207             BRW     100$                            ; OTHERWISE, BRANCH TO ERROR HANDLING
          51    06 A7   3F    AB     002F   208 30$:     BICW3   #^O77,RL_MP(R7),R1              ; GET CURRENT CYL & SURFACE
                                0034   209
                                0034   210 ;
                                0034   211 ; NOW CONVERT LOGICAL TO PHYSICAL
                                0034   212 ;
                                0034   213
                    55    02    C4     0034   214             MULL    #2,R5                           ; CONVERT LOGICAL BLOCKS TO SECTORS
                          56    D4     0037   215             CLRL    R6                              ; CLEAR HIGH PART OF DIVIDEND
54  56  55    00000050  8F    7B     0039   216             EDIV    #80,R5,R6,R4                    ; R6 = DESIRED CYL = LBN/(SECTORS/CYL)
                                0042   217                                                     ; R4 = REMAINING SECTORS
                          55    D4     0042   218             CLRL    R5                              ; CLEAR HIGH PART OF DIVIDEND
          54    55  54    28    7B     0044   219             EDIV    #40,R4,R5,R4                    ; R5 = DESIRED SURFACE = R5/(SECT/SUR)
                                0049   220                                                     ; R4 = DESIRED SECTOR
          56    56  07    78    0049   221             ASHL    #7,R6,R6                        ; SHIFT DESIRED CYLINDER INTO R6<15:7>
          56    01  06    55    F0     004D   222             INSV    R5,#6,#1,R6                     ; INSERT DESIRED SURFACE BIT INTO R6<6>
                    51    56    B1     0052   223             CMPW    R6,R1                           ; IS A SEEK NEEDED?
                          2E    13     0055   224             BEQL    50$                             ; BRANCH IF NOT.
                                0057   225
                                0057   226 ;
                                0057   227 ; NEED TO PERFORM A SEEK.
                                0057   228 ;
                                0057   229
      51    007F  8F    AA     0057   230             BICW    #^O177,R1                       ; ISOLATE CURRENT CYLINDER IN R1
52    56    007F  8F    AB     005C   231 35$:     BICW3   #^O177,R6,R2                    ; ISOLATE DESIRED CYLINDER IN R2
            51    52    A2     0062   232             SUBW    R2,R1                           ; SUBTRACT DESIRED FROM ACTUAL
                  08    13     0065   233             BEQL    40$                             ; BRANCH IF CURRENT = DESIRED CYLINDER
                  06    1E     0067   234             BCC     40$                             ; BRANCH IF CURRENT >= DESIRED CYLINDER
            51    51    AE     0069   235             MNEGW   R1,R1                           ; ACTUAL<DESIRED, MAKE POSITIVE DIFF
            51    04    A8     006C   236             BISW    #4,R1                           ; SET SIGN FOR MOVE TO CENTER OF DISK
51    01    04    55    F0     006F   237 40$:     INSV    R5,#4,#1,R1                     ; INSERT SURFACE BIT IN R1<4>
  04  A7    51    01    A9     0074   238             BISW3   #RL_DA_M_MRK,R1,-               ; SET MARKER AND LOAD DIFFERENCE
                                0079   239                     RL_DA(R7)                       ; WORD.
      67    50    06    A9     0079   240             BISW3   #6,R0,RL_CS(R7)                 ; EXECUTE SEEK FUNCTION
                0090  30     007D   241             BSBW    READY                           ; WAIT FOR CONTROLLER READY OR ERROR
                  03    18     0080   242             BGEQ    50$                             ; BRANCH IF NO ERROR DURING SEEK
                0072  31     0082   243             BRW     100$                            ; OTHERWISE, BRANCH DUE TO ERROR
                                0085   244
                                0085   245 ;
                                0085   246 ; SEEK, IF ANY, IS COMPLETE. EXECUTE TRANSFER FUNCTION
                                0085   247 ;
                                0085   248
56    06    00    54    F0     0085   249 50$:     INSV    R4,#0,#6,R6                     ; MERGE SECTOR WITH CYLINDER AND SURFACE
                                008A   250                                                     ; CYL=R6<15:7> SUR=R6<6> SEC=R6<5:0>
```

```
                04 A7    56   B0   008A   251          MOVW    R6,RL_DA(R7)            ; SET DESIRED DISK ADDRESS
                02 A7    5A   B0   008E   252          MOVW    R10,RL_BA(R7)           ; SET BUFFER ADDRESS
                   52    58   B0   0092   253          MOVW    R8,R2                   ; GET WORKING COPY OF BYTES LEFT TO XFER
                                   0095   254                                          ; AND ASSUME THIS IS LAST TRANSFER
             51    28    54   A3   0095   255          SUBW3   R4,#40,R1               ; R1 = SECTORS LEFT ON SURFACE
             51  0100  8F   A4   0099   256          MULW    #256,R1                 ; CONVERT TO BYTES LEFT ON SURFACE
                   51    52   B1   009E   257          CMPW    R2,R1                   ; ARE ADDITIONAL TRANSFERS REQUIRED?
                         03   1B   00A1   258          BLEQU   60$                     ; BRANCH IF ANSWER IS NO.
                   52    51   B0   00A3   259          MOVW    R1,R2                   ; SET BYTE COUNT FOR THIS TRANSFER
                   52    02   A6   00A6   260 60$:     DIVW    #2,R2                   ; CALCULATE TRANSFER WORD COUNT
                06 A7    52   AE   00A9   261          MNEGW   R2,RL_MP(R7)            ; SET NEG TRANSFER WORD COUNT
                   51    0C   B0   00AD   262          MOVW    #^XC,R1                 ; ASSUME READ FUNCTION
                20   10 AC   D1   00B0   263          CMPL    FUNC(AP),#IO$_WRITELBLK ; IS IT A WRITE FUNCTION?
                         03   12   00B4   264          BNEQ    70$                     ; BRANCH IF NOT
                   51    0A   B0   00B6   265          MOVW    #^XA,R1                 ; SET WRITE FUNCTION CODE
             67    51    50   A9   00B9   266 70$:     BISW3   R0,R1,RL_CS(R7)         ; MERGE UNIT # WITH FUNCTION AND EXECUTE
                     0050   30   00BD   267          BSBW    READY                   ; WAIT FOR CONTROLLER READY OR ERROR
                         35   19   00C0   268          BLSS    100$                    ; BRANCH ON ERROR
                   52    02   A4   00C2   269 80$:     MULW    #2,R2                   ; FIND BYTES TRANSFERRED THIS TIME
                   58    52   A2   00C5   270          SUBW    R2,R8                   ; UPDATE BYTES LEFT TO TRANSFER
                         1C   13   00C8   271          BEQL    90$                     ; BRANCH IF TRANSFER IS COMPLETE
                                   00CA   272
                                   00CA   273 ;
                                   00CA   274 ; UPDATE PARAMETERS FOR NEXT TRANSFER
                                   00CA   275 ;
                                   00CA   276
          51    04 A7   007F 8F   AB   00CA   277          BICW3   #^O177,RL_DA(R7),R1     ; UPDATE CURRENT CYLINDER IN R1<15:7>
             56    04 A7    3F   A9   00D1   278          BISW3   #^O77,RL_DA(R7),R6      ; SET SECTOR BITS TO 1'S AND -
                            56   B6   00D6   279          INCW    R6                      ; UPDATE DESIRED DISK ADDRESS IN R6
          55    56    01    06   EF   00D8   280          EXTZV   #6,#1,R6,R5             ; UPDATE DESIRED SURFACE IN R5
                            54   D4   00DD   281          CLRL    R4                      ; UPDATE DESIRED SECTOR IN R4
                5A    02 A7   B0   00DF   282          MOVW    RL_BA(R7),R10           ; UPDATE DESIRED BUFFER ADDRESS
                      FF76   31   00E3   283          BRW     35$                     ; LOOP FOR NEXT TRANSFER
                                   00E6   284
                                   00E6   285 ;
                                   00E6   286 ; TRANSFER COMPLETE - RETURN
                                   00E6   287 ;
                                   00E6   288
                   51    08 AC   3C   00E6   289 90$:     MOVZWL  SIZE(AP),R1             ; SET TOTAL BYTES TRANSFERRED
                         07   12   00EA   290          BNEQ    95$                     ; BRANCH IF ORIGINAL SIZE WAS TRANSFERRED
             51  00008000 8F   D0   00EC   291          MOVL    #^X8000,R1              ; ELSE SIZE WAS FORCED TO 64K
                   50    01   3C   00F3   292 95$:     MOVZWL  #SS$_NORMAL,R0          ; SET COMPLETION CODE
                         05   05   00F6   293          RSB                             ; AND RETURN
                                   00F7   294
                                   00F7   295 ;
                                   00F7   296 ; RETRY ERROR
                                   00F7   297 ;
                                   00F7   298
                   58    08 AC   3C   00F7   299 100$:    MOVZWL  SIZE(AP),R8             ; RESTORE BYTE SIZE OF TRANSFER IN R8
                         07   12   00FB   300          BNEQ    110$                    ; BRANCH IF SIZE WAS LEGAL
             58  00001F40 8F   D0   00FD   301          MOVL    #8000,R8                ; ELSE FORCE TO 64K SIZE
       5A    04 AC    09   00   EF   0104   302 110$:    EXTZV   #0,#9,BUF(AP),R10       ; RESTORE BYTE OFFSET IN R10
                   50    0054 8F   3C   010A   303          MOVZWL  #SS$_CTRLERR,R0         ; SET FATAL CONTROLLER ERROR
                         05   05   010F   304          RSB                             ; AND ATTEMPT RETRY
                                   0110   305
                                   0110   306
                                   0110   307 ;
```

```
                        0110   308 ; SUBROUTINE TO WAIT FOR CONTROLLER READY OR ERROR
                        0110   309 ;
                        0110   310
                        0110   311 READY:                                       ;
                67 8080 8F  B3  0110   312         BITW    #^X8080,(R7)         ;CONTROLLER READY OR ERROR?
                        F9  13  0115   313         BEQL    READY                ;IF EQL NO
                            05  0117   314         RSB                          ;
                        0118   315
58 45 2E 52 45 56 49 52 44 4C 44 00' 0118   316 DLNAME: .ASCIC  /DLDRIVER.EXE/   ; Driver file name
                            45  0124
                            0C  0118
                        0125   317
                00000125    0125   318 DL_DRVSIZ=.-DL_DRIVER
                        0125   319
                        0125   320         .END
```

```
$TABLE                          = 00000000 R      02
BTD$K_DL                        = 00000002
BUF                             = 00000004
DLNAME                            00000118 R      03
DL_DRIVER                         00000000 R      03
DL_DRVSIZ                       = 00000125
FUNC                            = 00000010
IO$_WRITELBLK                   = 00000020
READY                             00000110 R      03
RL_BA                             00000002
RL_CS                             00000000
RL_CS_M_DRDY                    = 00000001
RL_DA                             00000004
RL_DA_M_MRK                     = 00000001
RL_DA_M_RST                     = 00000008
RL_DA_M_STS                     = 00000002
RL_MP                             00000006
RL_MP_M_BH                      = 00000008
RL_MP_M_HO                      = 00000010
RPB$W_UNIT                      = 00000064
SIZ...                          = 00000001
SIZE                            = 00000008
SS$_CTRLERR                     = 00000054
SS$_NORMAL                      = 00000001
```

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	00000008 (8.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
BOOTDRIVR_4	00000028 (40.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
BOOTDRIVR_2	00000125 (293.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	31	00:00:00.08	00:00:00.45
Command processing	127	00:00:00.72	00:00:02.50
Pass 1	320	00:00:10.28	00:00:21.07
Symbol table sort	0	00:00:01.74	00:00:03.06
Pass 2	70	00:00:01.80	00:00:03.28
Symbol table output	4	00:00:00.05	00:00:00.05
Psect synopsis output	2	00:00:00.03	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	556	00:00:14.70	00:00:30.64

The working set limit was 1350 pages.
58295 bytes (114 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 1101 non-local and 13 local symbols.
320 source lines were read in Pass 1, producing 14 object records in Pass 2.
17 pages of virtual memory were used to define 15 macros.

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

Macro library name                                  Macros defined
------------------                                  --------------
_$255$DUA28:[BOOTS.OBJ]BOOTS.MLB;1                           1
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                               4
_$255$DUA28:[SYSLIB]STARLET.MLB;2                            6
TOTALS (all libraries)                                      11

1162 GETS were required to define 11 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:DLBTDRIVR/OBJ=OBJ$:DLBTDRIVR MSRC$:DLBTDRIVR/UPDATE=(ENH$:DLBTDRIVR)+EXECML$/LIB+LIB$:BOOTS.MLB/LIB
```

CVBTDRIVR
LIS

DDBTDRIVR
LIS

DMBTDRIVR
LIS

DQBTDRIVR
LIS

CONIO
LIS

LOADERSUB
LIS

CONFIGUTL
LIS

DLBTDRIVR
LIS

DXBTDRIVR
LIS

INITPGFIL
LIS

LOCKDATA
LIS

LOADDRIV
LIS

LOADER
LIS