```
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
```

```
RRRRRRRR    XX      XX  BBBBBBBB    TTTTTTTTTT  DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  RRRRRRRR
RRRRRRRR    XX      XX  BBBBBBBB    TTTTTTTTTT  DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  RRRRRRRR
RR      RR  XX      XX  BB      BB      TT      DD      DD  RR      RR      II      VV      VV  RR      RR
RR      RR  XX      XX  BB      BB      TT      DD      DD  RR      RR      II      VV      VV  RR      RR
RR      RR    XX  XX    BB      BB      TT      DD      DD  RR      RR      II      VV      VV  RR      RR
RR      RR    XX  XX    BB      BB      TT      DD      DD  RR      RR      II      VV      VV  RR      RR
RRRRRRRR        XX      BBBBBBBB        TT      DD      DD  RRRRRRRR        II      VV      VV  RRRRRRRR
RRRRRRRR        XX      BBBBBBBB        TT      DD      DD  RRRRRRRR        II      VV      VV  RRRRRRRR
RR  RR          XX  XX  BB      BB      TT      DD      DD  RR  RR          II      VV      VV  RR  RR
RR  RR          XX  XX  BB      BB      TT      DD      DD  RR  RR          II      VV      VV  RR  RR
RR    RR    XX      XX  BB      BB      TT      DD      DD  RR    RR        II        VV  VV    RR    RR      ....
RR    RR    XX      XX  BB      BB      TT      DD      DD  RR    RR        II        VV  VV    RR    RR      ....
RR      RR  XX      XX  BBBBBBBB        TT      DDDDDDDD    RR      RR    IIIIII        VV      RR      RR    ....
RR      RR  XX      XX  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 contents

```
0000     1          .TITLE  RXBTDRIVR - 11/8SS CONSOLE RX50 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 the
0000    34 ;       RX50 console floppy.
0000    35 ;
0000    36 ; ENVIRONMENT:  IPL 31, kernel mode, code must be PIC
0000    37 ;
0000    38 ; AUTHOR:  Robert L. Rappaport, CREATION DATE:   14-Oct-1983
0000    39 ;
0000    40 ; MODIFIED BY:
0000    41 ;
0000    42 ;       V03-001 RLR0001         Robert L. Rappaport     2-Apr-1984
0000    43 ;               Subroutine LOAD_RX50 returns in R0; 0 implies success and
0000    44 ;               1 implies error.  Correct sense of BLBx instructions of
0000    45 ;               callers of this routine.
0000    46 ;
0000    47 ;--
```

```
                     0000     49          .SBTTL  DECLARATIONS
                     0000     50 ;
                     0000     51 ; INCLUDE FILES:
                     0000     52 ;
                     0000     53
                     0000     54          $BTDDEF                         ; Boot device types
                     0000     55          $IODEF                          ; I/O function codes
                     0000     56          $PRDEF                          ; Processor registers
                     0000     57          $PTEDEF                         ; PTE definitions
                     0000     58          $RPBDEF                         ; RPB offsets
                     0000     59          $SSDEF                          ; Status codes
                     0000     60          $VADEF                          ; Virtual address fields
                     0000     61
                     0000     62 ;
                     0000     63 ; MACROS:
                     0000     64 ;
                     0000     65
                     0000     66 ;
                     0000     67 ; EQUATED SYMBOLS:
                     0000     68 ;
                     0000     69
                     0000     70 ;
                     0000     71 ; 11/8SS CONSOLE FLOPPY DEFINITIONS
                     0000     72 ;
                     0000     73
                     0000     74          $DEFINI RX                      ;START OF REGISTER DEFINITIONS
                     0000     75
                     0000     76 $DEF     RX5ID           .BLKB   1       ; ID Register (RO register, value is 04)
00000002             0001     77                          .BLKB   1
00000003             0002     78                          .BLKB   1
00000004             0003     79                          .BLKB   1
                     0004     80 $DEF     RX5CS0          .BLKB   1       ; Dual purpose register. Used for
                     0005     81                                          ;  entering commands and also for
                     0005     82                                          ;  reading status.
                     0005     83          _VIELD  RX5CS0,0,<-             ; Command Function bit definitions.
                     0005     84                  <SIDSEL,,M>,-           ; Side Select
                     0005     85                  <DSKSEL,,M>,-           ; Disk Select
                     0005     86                  <DRVSEL,,M>,-           ; Drive Select
                     0005     87                  <EXTMOT,,M>,-           ; Extended Motor Timeout
                     0005     88                  <FUNC0,,M>,-            ; Function Bit 0
                     0005     89                  <FUNC1,,M>,-            ; Function Bit 1
                     0005     90                  <FUNC2,,M>,-            ; Function Bit 2
                     0005     91          >
                     0005     92          _VIELD  RX5CS0,3,<-             ; R/W Status bit definitions.
                     0005     93                  <DONE,,M>,-             ; Done
                     0005     94                  <,3>,-
                     0005     95                  <ERROR,,M>,-            ; Error summary
                     0005     96          >
00000006             0005     97                          .BLKB   1
                     0006     98
                     0006     99
                     0006    100 $DEF     RX5CS1          .BLKB   1       ; Command Function is to accept target
                     0007    101                                          ;  track.
00000008             0007    102                          .BLKB   1
                     0008    103 $DEF     RX5CS2          .BLKB   1       ; Command Function is to accept target
                     0009    104                                          ;  sector.
0000000A             0009    105                          .BLKB   1
```

```
                     000A   106 $DEF     RX5CS3          .BLKB   1       ; Current Sector - This register contains th
                     000B   107                                          ;  sector number that was last specified
                     000B   108                                          ;  in the input sector register.
0000000C             000B   109                          .BLKB   1
                     000C   110
                     000C   111 $DEF     RX5CS4          .BLKB   1       ; Incorrect track register - The contents
                     000D   112                                          ;  of this register will be valid only
                     000D   113                                          ;  when a SEEK error occurs while attempting
                     000D   114                                          ;  to execute a command.  Then this
                     000D   115                                          ;  register will contain the track address
                     000D   116                                          ;  where the R/W head finally located.
                     000D   117                                          ;  If no SEEK error occurred, will be 0.
0000000E             000D   118                          .BLKB   1
                     000E   119
                     000E   120 $DEF     RX5CS5          .BLKB   1       ; Extended Command Register
                     000F   121                                          ;
                     000F   122                                          ;
00000010             000F   123                          .BLKB   1
                     0010   124
                     0010   125 $DEF     RX5EB           .BLKB   1       ; Empty Sector Buffer Register - Sequential
                     0011   126                                          ;  byte reads from this register empty
                     0011   127                                          ;  the 512 byte data buffer.
00000012             0011   128                          .BLKB   1
                     0012   129
                     0012   130 $DEF     RX5CA           .BLKB   1       ; Clear Address Register - Any access to
                     0013   131                                          ;  this register results in clearing
                     0013   132                                          ;  Data Buffer address to zero.
00000014             0013   133                          .BLKB   1
                     0014   134
                     0014   135 $DEF     RX5GO           .BLKB   1       ; Start Command Register - Any access
                     0015   136                                          ;  to this register instructs the
                     0015   137                                          ;  controller to execute the function
                     0015   138                                          ;  specified in the command registers.
00000016             0015   139                          .BLKB   1
                     0016   140
                     0016   141 $DEF     RX5FB           .BLKB   1       ; Fill Sector Buffer - Write only
                     0017   142                                          ;  register.  Successive byte writes
                     0017   143                                          ;  to this register fill 512 byte buffer.
00000018             0017   144                          .BLKB   1
                     0018   145
00000040             0018   146 RX_READSECTOR = ^x40                     ; FUNCTION READ SECTOR
00000070             0018   147 RX_WRITESECTOR = ^x70                    ; FUNCTION WRITE SECTOR
                     0018   148
                     0018   149         $DEFEND RX                      ;END OF 11/8SS FLOPPY DEFINITIONS
                     0000   150
200B0000             0000   151 RCX50_REGS = ^x200B0000                 ; Physical address of RCX50 CSRs.
                     0000   152
                     0000   153 ;
                     0000   154 ; OWN STORAGE:
                     0000   155 ;
                     0000   156
                     0000   157 ;
                     0000   158 ; Boot driver table entry
                     0000   159 ;
                     0000   160
                     0000   161         $BOOT_DRIVER    DEVTYPE = BTD$K_CONSOLE,-  ; Device type (console)
                     0000   162                         CPUTYPE = PR$_STD_TYP8SS,- ; Cpu type (11/8SS)
```

```
                        ;
                        ; SIZE = CONSRX_DRVSIZ,-     ; Driver size
                        ; ADDR = CONSRX_DRIVER,-     ; Driver address
                        ; DRIVRNAME = RXNAME         ; Driver file name
```

```
                     0000   167          .SBTTL  Console Floppy Bootstrap Driver Code
                     0000   168
                     0000   169 ;++
                     0000   170 ;
                     0000   171 ; Inputs:
                     0000   172 ;
                     0000   173 ;       R1      Address of page table for virtual -> physical mapping
                     0000   174 ;       R2      Base VPN of transfer (Bits 29:9 of R10)
                     0000   175 ;       R5      LBN for current piece of transfer
                     0000   176 ;       R8      Size of transfer in bytes
                     0000   177 ;       R9      Address of the RPB
                     0000   178 ;       R10     Starting address of transfer
                     0000   179 ;
                     0000   180 ;       FUNC(AP)  I/O operation (IO$_READLBLK or IO$_WRITELBLK only)
                     0000   181 ;       MODE(AP)  Address interpretation mode:
                     0000   182 ;                    0 -> Physical,  1 -> Virtual
                     0000   183 ;
                     0000   184 ; Outputs:
                     0000   185 ;
                     0000   186 ;       R0      Status code:
                     0000   187 ;
                     0000   188 ;               SS$_NORMAL      Successful transfer
                     0000   189 ;               SS$_CTRLERR     Fatal controller error
                     0000   190 ;--
                     0000   191
           00000010  0000   192 FUNC = 16
           00000014  0000   193 MODE = 20
                     0000   194
                     0000   195 CONSRX_DRIVER:
                 0D06 8F    BB  0000   196         PUSHR   #^M<R1,R2,R8,R10,R11>   ; Save input registers
     54  200B0000 8F    D0  0004   197         MOVL    #RCX50_REGS,R4          ; R4 => CSR base (physical address).
                     000B   198
                     000B   199 ;
                     000B   200 ; Perform initialization:  Set up a mapping switch in R11,
                     000B   201 ; There are 4 possibilities concerning mapping:  the I/O can be
                     000B   202 ; done virtual or physical (MODE(AP)) and we can be executing virtual
                     000B   203 ; or physical (contents of processor register PR$_MAPEN).  If both
                     000B   204 ; modes match, then we can just copy data to/from the user buffer.
                     000B   205 ; If the I/O is to be done virtual and we are executing physical
                     000B   206 ; then the buffer address has to be translated using the page table
                     000B   207 ; pointed to by R1.  If the I/O is to be done physical and we are
                     000B   208 ; executing virtual then we have to double map the buffer using a spare
                     000B   209 ; PTE.  At this point, we just compute a mapping switch in R11 as follows:
                     000B   210 ;
                     000B   211 ;       0       Both modes match, just copy the data
                     000B   212 ;       1       Do virtual -> physical translation using page table
                     000B   213 ;       -1      Do physical -> virtual mapping using a spare PTE
                     000B   214 ;
                     000B   215
                   5B    38  DB  000B   216         MFPR    #PR$_MAPEN,R11          ; Get mapping enabled switch
                         04  13  000E   217         BEQL    10$                     ; EQL implies mapping NOT enabled so
                             0010   218                                             ;  the physical CSR address in R4 is
                             0010   219                                             ;  valid.
                   54    58 A9  D0  0010   220         MOVL    RPB$L_CSRVIR(R9),R4     ; Replace with virtual CSR address. NOTE
                             0014   221                                             ;  that this implies that the console
                             0014   222                                             ;  RX50 is the real boot device.  That
                             0014   223                                             ;  is we rely on RPB$L_CSRVIR pointing
```

```
                     0014   224                                             ;  to this device's CSR.
                     0014   225 10$:
   5B   5B   CE      0014   226          MNEGL   R11,R11                    ; Negate it
5B   14 AC   C0      0017   227          ADDL    MODE(AP),R11               ; Add I/O mode switch
                     001B   228
   56   51   7D      001B   229          MOVQ    R1,R6                      ; R6 = Addr. of page tbl, R7 = Base VPN
                     001E   230
                     001E   231 ;
                     001E   232 ; Set up mapping if required
                     001E   233 ;
        00B4   30    001E   234          BSBW    DO_MAPPING
                     0021   235
                     0021   236 ;
                     0021   237 ; This is the main loop to read or write to the floppy and to get
                     0021   238 ; or store in memory.  Register usage is:
                     0021   239 ;
                     0021   240 ;       R0 - R3         Scratch
                     0021   241 ;       R4              Address of CSR base
                     0021   242 ;       R5              LBN
                     0021   243 ;       R6              Address of page table
                     0021   244 ;       R7              Virtual page number of buffer
                     0021   245 ;       R8              Size of remaining buffer (in bytes)
                     0021   246 ;       R9              Address of RPB
                     0021   247 ;       R10             Address of current spot in buffer
                     0021   248 ;       R11             Mapping switch
                     0021   249 ;
                     0021   250 ; First convert LBN to physical sector and
                     0021   251 ; cylinder.  Then send command, sector, and cylinder to floppy.  Then
                     0021   252 ; read or write 512 bytes of data.  Repeat until byte count goes to zero.
                     0021   253 ;
                     0021   254
                     0021   255
                     0021   256 MAIN_LOOP:
                     0021   257
        5E   10      0021   258          BSBB    TRKSEC                     ; Returns: R0 = side select,
                     0023   259                                             ; R1 = track and R2 = sector
53   0200 8F   3C    0023   260          MOVZWL  #512,R3                    ; Assume full sector I/O.
   58   53   D1      0028   261          CMPL    R3,R8                      ; Compare to what's left of buffer.
        03   15      002B   262          BLEQ    10$                        ; LEQ implies full sector of I/O.
   53   58   D0      002D   263          MOVL    R8,R3                      ; User lesser of R3 and R8.
                     0030   264 10$:
 21   10 AC   D1      0030   265          CMPL    FUNC(AP),#IO$_READLBLK     ; Is it read?
        25   13      0034   266          BEQL    READ                       ; Yes
                     0036   267
                     0036   268 ;
                     0036   269 ; Do a Write to the floppy
                     0036   270 ;
                     0036   271
 50   70 8F   88      0036   272          BISB    #RX_WRITESECTOR,R0         ; Or in write command.
   12 A4   95      003A   273          TSTB    RX5CA(R4)                  ; Clear hardware buffer (silo) address.
        50   DD      003D   274          PUSHL   R0                         ; Save register value.
                     003F   275 20$:
        75   10      003F   276          BSBB    GETBYTE                    ; Get byte from memory into R0.
16 A4   50   90      0041   277          MOVB    R0,RX5FB(R4)               ; Fill hardware (silo) buffer.
   F7 53   F5      0045   278          SOBGTR  R3,20$                     ; Repeat
        50 8ED0    0048   279          POPL    R0                         ; Restore register.
      009E   30      004B   280          BSBW    LOAD_RX50                  ; Load CSRs. (R0=0 return is success.)
```

```
                20 50   E9  004E  281          BLBC    R0,SECTOR_SUCCESS       ; If success, branch around.
                            0051  282
                            0051  283 FLOPPY_ERROR:                           ; Error from floppy
                            0051  284
            0D06 8F     BA  0051  285          POPR    #^M<R1,R2,R8,R10,R11>   ; Restore registers
      50    0054 8F     3C  0055  286          MOVZWL  #SS$_CTRLERR,R0         ; Set failure status
                        05  005A  287          RSB                             ; Return and retry
                            005B  288
                            005B  289 ;
                            005B  290 ; Do a Read from floppy
                            005B  291 ;
                            005B  292
                            005B  293 READ:
      50    40 8F     88  005B  294          BISB    #RX_READSECTOR,R0       ; Or in read command.
               008A   30  005F  295          BSBW    LOAD_RX50               ; Load CSRs. (R0=1 return is failure.)
            EC 50     E8  0062  296          BLBS    R0,FLOPPY_ERROR         ; If ERROR, branch back.
            12 A4     95  0065  297          TSTB    RX5CA(R4)               ; Clear hardware buffer (silo) address.
                            0068  298 30$:
      50    10 A4     90  0068  299          MOVB    RX5EB(R4),R0            ; Empty hardware (silo) buffer.
                  53    10  006C  300          BSBB    PUTBYTE                 ; Put byte into memory from R0.
            F7 53     F5  006E  301          SOBGTR  R3,30$                  ; Repeat
                            0071  302
                            0071  303 ;
                            0071  304 ; Done with this sector.  Repeat loop if byte count is non-zero
                            0071  305 ;
                            0071  306 SECTOR_SUCCESS:
                            0071  307
                  58    D5  0071  308          TSTL    R8                      ; Test remaining byte count
                  04    13  0073  309          BEQL    DONE_SUCCESS            ; Done
                  55    D6  0075  310          INCL    R5                      ; Increment LBN
                  A8    11  0077  311          BRB     MAIN_LOOP               ; Do next sector
                            0079  312
                            0079  313 DONE_SUCCESS:
                            0079  314
      50    01        3C  007A  315          MOVZWL  #SS$_NORMAL,R0          ; Successful completion
      0D06 8F         BA  007C  316          POPR    #^M<R1,R2,R8,R10,R11>   ; Restore registers
                        05  0080  317          RSB
```

```
                                0081   319 ;++
                                0081   320 ; TRKSEC - Subroutine to convert LBN to physical
                                0081   321 ;           sector/cylinder on floppy by applying sector interleave
                                0081   322 ;           and track-to-track skew (2 sectors).
                                0081   323 ;
                                0081   324 ; Inputs:
                                0081   325 ;       R5      LBN
                                0081   326 ;
                                0081   327 ; Outputs:
                                0081   328 ;       OUTPUTS:        R0      Side select data
                                0081   329 ;                       R1      track number
                                0081   330 ;                       R2      Sector number
                                0081   331 ;
                                0081   332 ;  NOTES: SIDE = LBN DIV 800      BN = LBN MOD 800
                                0081   333 ;           TRK  = BN DIV 10       BNA = BN MOD 10
                                0081   334 ;
                                0081   335 ; Credits:
                                0081   336 ;       Thanks to Dick Vignoni for this one.
                                0081   337 ;--
                                0081   338
                                0081   339 TRKSEC:
                                0081   340
                                0081   341 ; Convert LBN in R5 to physical device address
                                0081   342
                     50   55    D0  0081   343         MOVL    R5,R0           ; Copy LBN to R0
                          51    D4  0084   344         CLRL    R1              ; Clear for EDIV
     52   50   50  00000320 8F    7B  0086   345         EDIV    #800.,R0,R0,R2  ; R2<=BN   R0<=SURFACE
                          53    D4  008F   346         CLRL    R3              ; Clear for EDIV
               53   51   52    0A    7B  0091   347         EDIV    #10.,R2,R1,R3   ; R1<=TRK  R3<=BNA
                          52    51  C0  0096   348         ADDL2   R1,R2           ; R2<== BN + TRK
                          52    02  C4  0099   349         MULL2   #2,R2           ; R2<== (BN+TRK)*2
                          53    05  C6  009C   350         DIVL2   #5,R3           ; R3<==  BNA/5
                          52    53  C0  009F   351         ADDL2   R3,R2           ; R2<==(BNA/5 + ((BN+TRK)*2))
                                53    D4  00A2   352         CLRL    R3              ; Clear for EDIV
     52   53   52    0A    7B  00A4   353         EDIV    #10.,R2,R3,R2   ; R2 <==R2 MOD 10
                                52    D6  00A9   354         INCL    R2              ; Make tracks start at 1
                                51    D6  00AB   355         INCL    R1              ; Make sectors start at 1
               51   50  8F    91  00AD   356         CMPB    #80.,R1         ; is this the last track on surface ?
                                02    12  00B1   357         BNEQ    10$             ; Br if no
                                51    D4  00B3   358         CLRL    R1              ; set track to 0 (last track=0)
                                          00B5   359 10$:
                                05    00B5   360         RSB
```

```
                     00B6   362 ;++
                     00B6   363 ; GETBYTE - Subroutine to get a byte from memory
                     00B6   364 ; PUTBYTE - Subroutine to store a byte in memory
                     00B6   365 ;
                     00B6   366 ;       These two subroutines do two things special:
                     00B6   367 ;
                     00B6   368 ;               1)  Since the floppy always reads or writes 128 bytes
                     00B6   369 ;                   these routines simply return if the byte count is zero.
                     00B6   370 ;               2)  These routines take care of page boundaries if
                     00B6   371 ;                   mapping is required.
                     00B6   372 ;
                     00B6   373 ; Inputs:
                     00B6   374 ;       R0      Byte to store (PUTBYTE)
                     00B6   375 ;       R6      Address of page table
                     00B6   376 ;       R7      Virtual page number of buffer
                     00B6   377 ;       R8      Size of remaining buffer (in bytes)
                     00B6   378 ;       R10     Address of current spot in buffer
                     00B6   379 ;       R11     Mapping switch:
                     00B6   380 ;                       -1   Do physical -> virtual map
                     00B6   381 ;                       0    No mapping required
                     00B6   382 ;                       1    Do virtual -> physical translation
                     00B6   383 ;
                     00B6   384 ; Outputs:
                     00B6   385 ;       R0      Byte fetched from memory (GETBYTE)
                     00B6   386 ;--
                     00B6   387
                     00B6   388         .ENABL  LSB
                     00B6   389
                     00B6   390 GETBYTE:
      50   D4        00B6   391         CLRL    R0                      ; Return 0 if byte count = 0
      58   D5        00B8   392         TSTL    R8                      ; Is byte count 0?
      2E   13        00BA   393         BEQL    90$                     ; Yes
50    8A   9A        00BC   394         MOVZBL  (R10)+,R0               ; Get byte
      07   11        00BF   395         BRB     10$                     ; Branch to common code
                     00C1   396
                     00C1   397
                     00C1   398 PUTBYTE:
      58   D5        00C1   399         TSTL    R8                      ; Is byte count 0?
      25   13        00C3   400         BEQL    90$                     ; Yes
8A    50   90        00C5   401         MOVB    R0,(R10)+               ; Store byte
                     00C8   402
                     00C8   403
      58   D7        00C8   404 10$:    DECL    R8                      ; Decr. byte count
      1E   13        00CA   405         BEQL    90$                     ; Reached zero
5A 01FF 8F   B3      00CC   406         BITW    #VA$M_BYTE,R10          ; Did address overflow onto new page?
      17   12        00D1   407         BNEQ    90$                     ; No
      57   D6        00D3   408         INCL    R7                      ; Yes, increment page number
                     00D5   409
                     00D5   410 ;
                     00D5   411 ; Fall through to ...
                     00D5   412 ;
                     00D5   413
                     00D5   414
                     00D5   415 ;++
                     00D5   416 ; DO_MAPPING - Subroutine to perform necessary mapping
                     00D5   417 ;
                     00D5   418 ; Inputs:
```

```
                        00D5   419 ;       R6      Address of page table
                        00D5   420 ;       R7      Page number of buffer
                        00D5   421 ;       R10     Address to map
                        00D5   422 ;       R11     Mapping switch:
                        00D5   423 ;                       -1      Do physical -> virtual map
                        00D5   424 ;                       0       No mapping required
                        00D5   425 ;                       1       Do virtual -> physical translation
                        00D5   426 ;
                        00D5   427 ; Outputs:
                        00D5   428 ;       R10     Address to use
                        00D5   429 ;--
                        00D5   430
                        00D5   431 DO_MAPPING:
                     5B 0D5    432         TSTL    R11                     ; Any mapping required?
                     11 13     433         BEQL    90$                     ; No
                     10 19     434         BLSS    100$                    ; Yes, map physical to virtual
              5A FFFFFE00 8F   CA 00DB    435         BICL    #^C<VA$M_BYTE>,R10      ; Yes, translate virtual to physical
                        00E2   436                                         ; Clear everything but byte offset
                   6647 DD     00E2    437         PUSHL   (R6)[R7]                ; Get PFN on top of stack
           5A 15 09 8E F0      00E5    438         INSV    (SP)+,#VA$V_VPN,#PTE$S_PFN,R10   ; Insert PFN after byte offset
                     05 00EA   439 90$:    RSB
                        00EB   440
                        00EB   441
                        00EB   442 ;
                        00EB   443 ; Map physical to virtual
                        00EB   444 ;
                     00 00EB   445 100$:   HALT                            ; Not implemented yet
                        00EC   446
                        00EC   447         .DSABL  LSB
```

```
                        00EC    449 ;++
                        00EC    450 ; LOAD_RX50 - subroutine to load RX50 CSR's
                        00EC    451 ;
                        00EC    452 ; Inputs:
                        00EC    453 ;       R0 = Command or'ed with side select
                        00EC    454 ;       R1 = track
                        00EC    455 ;       R2 = sector
                        00EC    456 ;       R4 => CSR base
                        00EC    457 ;
                        00EC    458 ; Outputs:
                        00EC    459 ;       R0 = 0 if successful
                        00EC    460 ;       R0 = 1 if error      *****NOTE, inverted success/failure indication*****
                        00EC    461 ;--
                        00EC    462
                        00EC    463 LOAD_RX50:
                        00EC    464
                04 A4   50   90 00EC    465         MOVB    R0,RX5CS0(R4)           ; Load command and side select.
                06 A4   51   90 00F0    466         MOVB    R1,RX5CS1(R4)           ; Load track.
                08 A4   52   90 00F4    467         MOVB    R2,RX5CS2(R4)           ; Load sector.
                     14 A4   94 00F8    468         CLRB    RX5GO(R4)               ; Start function.
                        00FB    469 10$:
                50   04 A4   9A 00FB    470         MOVZBL  RX5CS0(R4),R0           ; Read status register.
                   50   08   93 00FF    471         BITB    #RX5CS0_M_DONE,R0       ; See if function completed.
                        F7   13 0102    472         BEQL    10$                     ; EQL implies not yet done.
                        0104    473         ASSUME  RX5CS0_V_ERROR  EQ      ;
          50    50   F9 8F   78 0104    474         ASHL    #-7,R0,R0               ; Shift error bit into low bit of R0.
                             05 0109    475         RSB                             ; Return to caller. R0 has status.
                        010A    476
58 45 2E 52 45 56 49 52 44 58 52 00' 010A    477 RXNAME: .ASCIC  /RXDRIVER.EXE/
                             45 0116
                             0C 010A
                        0117    478
00000117                0117    479 CONSRX_DRVSIZ=.-CONSRX_DRIVER
                        0117    480
                        0117    481         .END
```

```
$TABLE                          = 00000000 R      02
BTD$K_CONSOLE                   = 00000040
CONSRX_DRIVER                     00000000 R      03
CONSRX_DRVSIZ                   = 00000117
DONE_SUCCESS                      00000079 R      03
DO_MAPPING                        000000D5 R      03
FLOPPY_ERROR                      00000051 R      03
FUNC                            = 00000010
GETBYTE                           000000B6 R      03
IO$_READLBLK                    = 00000021
LOAD_RX50                         000000EC R      03
MAIN_LOOP                         00000021 R      03
MODE                            = 00000014
PR$_MAPEN                       = 00000038
PR$_SID_TYP8SS                  = 00000005
PTE$S_PFN                       = 00000015
PUTBYTE                           000000C1 R      03
RCX50_REGS                      = 200B0000
READ                              0000005B R      03
RPB$L_CSRVIR                    = 00000058
RX5CA                             00000012
RX5CS0                            00000004
RX5CS0_M_DONE                   = 00000008
RX5CS0_V_ERROR                  = 00000007
RX5CS1                            00000006
RX5CS2                            00000008
RX5CS3                            0000000A
RX5CS4                            0000000C
RX5CS5                            0000000E
RX5EB                             00000010
RX5FB                             00000016
RX5GO                             00000014
RX5ID                             00000000
RXNAME                            0000010A R      03
RX_READSECTOR                   = 00000040
RX_WRITESECTOR                  = 00000070
SECTOR_SUCCESS                    00000071 R      03
SIZ...                          = 00000001
SS$_CTRLERR                     = 00000054
SS$_NORMAL                      = 00000001
TRKSEC                            00000081 R      03
VA$M_BYTE                       = 000001FF
VA$V_VPN                        = 00000009
```

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	00000018	(24.)	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	00000117	(279.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	36	00:00:00.09	00:00:00.53
Command processing	138	00:00:00.66	00:00:01.63
Pass 1	330	00:00:10.10	00:00:23.14
Symbol table sort	0	00:00:01.65	00:00:03.45
Pass 2	96	00:00:01.99	00:00:04.02
Symbol table output	5	00:00:00.10	00:00:00.56
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	609	00:00:14.62	00:00:33.36

The working set limit was 1500 pages.
57074 bytes (112 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 1076 non-local and 9 local symbols.
481 source lines were read in Pass 1, producing 14 object records in Pass 2.
19 pages of virtual memory were used to define 17 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[BOOTS.OBJ]BOOTS.MLB;1	1
_255DUA28:[SYS.OBJ]LIB.MLB;1	4
_255DUA28:[SYSLIB]STARLET.MLB;2	8
TOTALS (all libraries)	13

1169 GETS were required to define 13 macros.

There were no errors, warnings or information messages.

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

READLBN
LIS

RTFILREAD
LIS

SHARE
LIS

QUSS
LIS

RWBTDRIVR
LIS

MBBTDRIVR
LIS

SCSLOADER
LIS

RMSCONIO
LIS

READDRIV
LIS

PABTDRIVR
LIS

PUBTDRIVR
LIS

PUTERROR
LIS

READPRMPT
LIS