```
************************************************
************************************************
***                                          ***
***  Name: Tovar                             ***
***                                          ***
***  Project:    1       Programmer:  TVR    ***
***                                          ***
***  File Name: XBOOT.LST[  3,TVR]           ***
***                                          ***
***  File Last Written:  15:49 19 Aug 1986   ***
***                                          ***
***  Time: 16:04         Date: 19 Aug 1986   ***
***                                          ***
***            Center for Computer           ***
***        Research in Music and Acoustics   ***
***             Music Department             ***
***           Stanford,  California          ***
***                                          ***
************************************************
************************************************
```



```
                                                                                                    11111111111111111111
         11111111112222222222333333333344444444445555555555666666666677777777778888888888999999999900000000001111111112
12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
```

```
01 0001 0       OBP = 0
01 0002 0       OSQ = 0
01 0003 0       fast = 0
01 0004
01 0005         .INSERT F4UBOT.NEW
```

```
01 0001          ;<PEFMIC>F4UBOT.SLO;4 14-May-82 15:08:33, Edit by FRENCH
01 0002          ;ADDED MEM IMG BOOT OPTION
01 0003
01 0004          XLIST
01 0007                                                  LIST
01 0008
01 0009        1 WAITS = 1
01 0010
01 0011        17000     UBLOC = 17000            ;WHERE TO ASSEMBLE THIS STUFF FOR NOW
01 0012
01 0013
01 0014
```

```
02 0015
02 0016
02 0017                                                 ;TITLE:  uBOOT FOR F4
02 0018                                                 ;AUTHOR:  PHIL FRENCH@OFFICE (TYMSHARE)
02 0019
02 0020                                                 ;DISK AND TAPE MICROCODE BOOTER
02 0021
02 0022                                                 ;DATA SWITCHES ARE READ AS FOLLOWS:
02 0023                                                 ;
02 0024                                                 ;<UHB>B0!<UNUSED>B8!<DC>B17!<FILN>B26!<UNIT>B35
02 0025
02 0026                                                 ;WHERE DC IS DEVICE CODE:
02 0027                                                 ;       0:  DISK
02 0028                                                 ;       1:  TAPE
02 0029                                                 ;WHERE FILN IS FILE #:
02 0030                                                 ;       FOR DISK:  0 FOR MICROCODE
02 0031                                                 ;                  1 FOR MICRODIAG
02 0032                                                 ;                  2 FOR MEMIMG
02 0033                                                 ;       FOR TAPE:  # OF FILE (0 IS FIRST) TO BOOT
02 0034                                                 ;WHERE UNIT IS:
02 0035                                                 ;       FOR DISK:  PACK #
02 0036                                                 ;       FOR TAPE:  MEANINGLESS NOW
02 0037                                                 ;WHERE UHB IS:
02 0038                                                 ;       FOR DISK:  0 FOR USE PRIMARY HOME BLOCK ON UCP
02 0039                                                 ;                  1 FOR USE SECONDARY HOME BLOCK ON UCP
02 0040                                                 ;       FOR TAPE:  MEANINGLESS FOR NOW
02 0041
02 0042
02 0043
02 0044                                                         ;ASSEMBLY SWITCHES AND PARAMETERS
02 0045
02 0046         1 RCVERR = 1                    ;EXACTLY 1 TO ASSEMBLE ERROR RECOVERY ROUTINES
02 0047
02 0048         17777     TRMADR = 17777                ;ADR WHICH WHEN SEEN INDICATES END OF FILE
02 0049                                                                         ;CONTAINS XFER INSTRUCTION
02 0050                                                                         ;LAST INSTRUCTION LOADED INTO UMEM
02 0051
02 0052                                                         ;MICRO MEM USAGE
02 0053
02 0054
02 0055         72        UBOK = 72                     ;SUCCESS CODE FOUND IN AR
02 0056         73        UBBAD = 73                    ;FAILURE CODE FOUND IN AR
02 0057
02 0058
02 0059                                                         ;MAIN MEMORY USAGE
02 0060
02 0061         3 HOMPAG = 3                    ;MAIN MEM PAGE TO READ HOME BLOCK
02 0062         4 XBPAG = 4                     ;MAIN MEM PAGE TO READ XB
02 0063         5 DATPAG = 5                    ;MAIN MEM PAGE TO READ FILE PAGE
02 0064
02 0065
02 0066                                                         ;PTR FOR ILDB OF 1ST DATA BYTE
```

```
02 0067
02 0068        41000,,5000        BYTPTR = 441000000000 + (DATPAG * 1000)
02 0069
02 0070
02 0071                                                ;AMEM USAGE COMMON TO BOTH DISK AND TAPE OPERATION
02 0072
02 0073        51          GBL-GT2 = D%AMEM11       ;TEMP USED BY GT2BYT
02 0074        52          GBL-GT3 = D%AMEM12       ;TEMP USED BY GT3BYT
02 0075        53          GBL-GT4 = D%AMEM13       ;TEMP USED BY GT4BYT
02 0076        54          GBL-UAD = D%AMEM14       ;HOLDS MICRO MEM ADR FOR DEPOSITING
02 0077        55          GBL-BCT = D%AMEM15       ;HOLDS # BYTES LEFT BEFORE IO NEEDED
02 0078        56          GBL-BPT = D%AMEM16       ;HOLDS BYTE PTR FOR EXTRACTING 8 BIT DATA BYTES
02 0079        57          OLD-MMB-FLG = D%AMEM17   ;Flag indicating an old-style MMB board in
02 0079                                                 machine.
02 0080
02 0081
```

```
03 0082
03 0083
03 0084                                                 ;STUFF IMPORTANT DURING DISK OPERATION
03 0085
03 0086
03 0087                                                 ;DISK AMEM USAGE (0:2 USED FOR DISK CONTROLLER OPERATION)
03 0088                                                 ;                (4:7 USED IN THIS MODULE ONLY)
03 0089
03 0090        40        DSK-DSP = D%AMEM0       ;HOLDS DISPATCH ADR FOR INTS
03 0091        41        DSK-PI  = D%AMEM1       ;HOLDS PI ASSIGNMENT AND FLAGS
03 0092        42        DSK-LCM = D%AMEM2       ;HOLDS LAST COMMAND SENT TO CONTROLLER
03 0093
03 0094        44        DSK-HDA = D%AMEM4       ;HOLDS HARDWARE DISK ADR
03 0095        45        DSK-MEM = D%AMEM5       ;HOLDS MEMORY ADR FOR DISK READS
03 0096        46        DSK-SDA = D%AMEM6       ;HOLDS SOFT DISK ADR
03 0097        47        DSK-RTY = D%AMEM7       ;(FLAGS)B1!(RETRY COUNTER)B5!(TIMEOUT COUNTER)B35
03 0097
03 0098                                                                 ;FLAGS:  B0:  0 FOR NORMAL DISK OPERATION
03 0099                                                                 ;             1 FOR RECAL'ING
03 0100                                                                 ;        B1:  0 TO JUMP TO RDSKE0 ERROR RT
N ON
03 0100                                                  ERROR
03 0101                                                                 ;             1 TO JUMP TO RDSKE1 ERROR RT
N ON
03 0101                                                  ERROR
03 0102        50        DSK-AOB = D%AMEM10      ;HOLDS AOBJN PTR FOR SCAN OF XB
03 0103
03 0104                                                         ;SEE GLOBAL AMEM DEFS FOR AMEM > 10
03 0105
03 0106                                                         ;INTERESTING HOMEBLOCK PARAMETERS FOR DISK OPERATION
03 0107
03 0108        1 HM1SRF = 1            ;SURF OF PRIMARY HOME BLOCK
03 0109        0 HM2SRF = 0            ;SURF OF SECONDARY HOME BLOCK
03 0110        3 HMNSEC = 3            ;SECTOR OF HOME BLOCKS
03 0111
03 0112        5 HOMHW2 = 5            ;INDEX FOR (NSECS,,NWSEC)
03 0113        6 HOMHW3 = 6            ;INDEX FOR (NTKUN*NSURFS*NSECS),,(NSURFS*NSECS)
03 0114        12        HOMMDA = 12           ;INDEX FOR HARDWARE DISK ADR OF ALTERNATE FILE XB
03 0114
03 0115                                                                 ; Filename is MICRODIAG.PACK-0;1
03 0116        14        HOMMCA = 14           ;INDEX FOR HARDWARE DISK ADR OF MICROCODE FILE XB
03 0116
03 0117                                                                 ; Filename is MICROCODE.PACK-0;1
03 0118        22        HOMINA = 22           ;INDEX FOR HARDWARE DISK ADR OF MEMIMG FILE XB
03 0119                                                                 ; Filename is MEMIMG.PACK-0;1
03 0120
03 0121
03 0122
```

```
04 0123
04 0124
04 0125                                                 ;STUFF IMPORTANT DURING TAPE OPERATION
04 0126
04 0127                                                 ;AMEM USEAGE:
04 0128
04 0129        40       TAP-DSP = D%AMEM0       ;DISPATCH ADDR.
04 0130        40       TAP-DPM = D%AMEM0       ;DATA PACKING MODE
04 0131                                                                 ; BIT 0: 0=PDP-10 CORE-DUMP, 1=INDUSTRY
(32-bit
04 0131                                                  mode)
04 0132                                                                 ; BIT 1: NRZI Kluge Mode (to rd old CCRM
A & SCI
04 0132                                                  tapes)
04 0133        41       TAP-TIM = D%AMEM1       ;Timeout values for NCNTWT and FMNBWT.
04 0134        42       TAP-LWT = D%AMEM2       ;Copy of last writ to TP.WC,
04 0135                                                                 ;Status at ENDX of tape operation.
04 0136        43       TAP-XXX = D%AMEM3       ;DON'T USE... Storing into it clobbers IR left !

04 0136
04 0137        44       TAP-MEM = D%AMEM4       ;Next mem adr of
 xfer (STRTDC) -- 0 during
04 0137                                                  non-data ops
04 0138        45       TAP-CNT = D%AMEM5       ;Remaining word count in current WCMA (STRTDC)
04 0139        46       TAP-FLN = D%AMEM6       ;# OF FILES TO SKIP (IE FILE # TO READ)
04 0140        47       TAP-STA = D%AMEM7       ;Interrupt state and transfer flags:
04 0141                                                                 ;1b0 = read, 1b1 = waiting for formatter
 to
04 0141                                                  become idle
04 0142                                                                 ;1b2 = mtop interrupt (set with read)
04 0143                                                                 ;18-35 CONI bits:
04 0144                                                                 ;7b35 = pi assignment. 10 = interrupt fl
ag.
04 0145                                                                 ;10 USED AS BOOT FLAG WHILE BOOTING
04 0146                                                                 ;01 USED AS "EOF" FLAG WHILE BOOTING
04 0147
04 0148                                                 ;SEE GLOBAL AMEM DEFS FOR AMEM ) 7 DEFINITIONS AS WELL...
04 0149
04 0150
04 0151                                                 ;MAPF values
04 0152
04 0153        4 TP.RS = 4          ;read status (from formatter)
04 0154        2 TP.RC = 2          ;read control (controller status and un-fifo'd read data)
04 0154
04 0155
04 0156        2 TP.WF = 2          ;write formatter (send ctrl bits to formatter)
04 0157        4 TP.WM = 4          ;write mode control reg.
04 0158        1 TP.WC = 1          ;write control reg.
04 0159        5 TP.MR = 5          ;give Master Reset
04 0160        3 TP.WMA = 3         ;write (load) the CNTMA reg. (count and MA)
04 0161
04 0162
```

```
05 0163
05 0164
05 0165                                                 ;ENTRY FOR UCODE BOOTER
05 0166
05m0167                                                           .use[UBLOC]     [ xlist
05 0167                                                    list ]
05 0168
05 0169 17000 010731170040000014161620254114560000               UBOOTA: CLR-DEV-FROM-INTR JUMP[UBTA1] $
05 0170 17001 010731170000360034161620254114560000               HALTED: JUMP[.] $       ;Hang here when done.
05 0171 17002 010730170000000034442362367011416000               UBTA1:  D[AMEM-ABS APR-DATASW] ROT[18.] MASK[9.] DEST[Q]
 LONG $
05 0172                                                                           ;GET DEVICE CODE IN Q, CLEAR DEV FROM IN
T
05 0173 17003 010661002000000014001620255514560000               D[CONST 0] ALU[D#Q] COND[ZERO] JUMP[UBOOTD] LONG $
05 0174                                                                   ;DISK
05 0175 17004 010661002000000014003620255514560000               D[CONST 1] ALU[D#Q] COND[ZERO] JUMP[UBOOTT] LONG $
05 0176                                                                   ;TAPE
05 0177 17005 010731170000000014161620254114560000               JUMP[XUBOOT] LONG $
05 0178                                                                   ;LOSER
05 0179
05 0180
05 0181                                                 ;TAPE OPERATION
05 0182
05 0183 17006 010731170040060554016463655514170000       UBOOTT: D[CONST 7] DEST[DEV-ADR] LONG $
05 0184                                                                   ;DEV-ADR IS TAPE FROM NOW ON
05 0185 17007 010731370000000354161623670114160000               D[AMEM-ABS APR-DATASW] DEST[AR] LONG $
05 0186                                                                   ;GET AROUND AMEM-ABS AND JADR CONFLICT VIA AR
05 0187 17010 010731170000060540361363654114160000               D[AR] ROT[1] DEST[OLD-MMB-FLG]  LONG $
05 0188 17011 010731170000000340361363670114160000               D[AMEM-ABS APR-DATASW] ROT[1] DEST[OLD-MMB-FLG]  LONG $
05 0189 17012 010731171000000000001003655114160000               D[NLIT 4000000000] DEST[TAP-DPM] LONG $
05 0190                                                                   ;SET 32 BIT PACKING MODE
05 0191 17013 010241170000060554161163654114160000               ALU[0] DEST[TAP-STA] LONG $
05 0192                                                                   ;CLEAR FLAGS
05 0193 17014 010730170000000034662362367011416000               D[AMEM-ABS APR-DATASW] ROT[27.] MASK[9.] DEST[Q] LONG $
05 0194 17015 010231170000000014161140254114560000               ALU[Q] DEST[TAP-FLN] JUMP[UMERGE] LONG $
05 0195                                                                   ;PUT FILE # IN AMEM6 AND MERGE
05 0196
05 0197
05 0198                                                 ;DISK OPERATION
05 0199
05 0200                                                 .REPEAT 1 - WAITS [
05 0201
05 0202                                                 UBOOTD: D[CONST 10] DEST[DEV-ADR] LONG $
05 0203                                                                   ;DEV-ADR IS DISK FROM NOW ON
05 0204                                                         D[CONST HM1SRF] ROT[8.] DEST[Q] LONG $
05 0205                                                                   ;ASSUME PRIMARY HOMEBLOCK, GET MORE STACK
05 0206                                                         D[AMEM-ABS APR-DATASW] DEST[AR] LONG $
05 0207                                                                   ;GET AROUND AMEM-ABS AND JADR CONFLICT VIA AR
05 0208                                                         D[AR] ROT[1] DEST[OLD-MMB-FLG]  LONG $
05 0209                                                         D[AR] COND[SIGNOFF] JUMP[PRIMHB] LONG $
05 0210                                                                   ;1B0 IN SWITCHES MEANS SECONDARY, JUMP IF PRIMAR
Y
```

```
05 0211
05 0212
05 0213
05 0214
05 0215
05 0216
05 0217
05 0218
05 0219
05 0220
 HOME
05 0220
05 0221
05 0222
05 0223
05 0224
05 0225
05 0226
05 0227
05 0228
05 0229
05 0230
05 0231
05 0232
05 0233
05 0234
05 0235
05 0236
05 0237
05 0238
05 0239
05 0240
05 0241
05 0242
05 0243
05 0244
)
05 0245
05 0246
05 0247
05 0248
05 0249
05 0250
05 0251
05 0252
ORDS IN
05 0252
05 0253
05 0254
05 0255
05 0256
05 0257
05 0258
05 0259
```

```
        D[CONST HM2SRF] ROT[8.] DEST[Q] LONG $
                ;WANTS SECONDARY
PRIMHB: D[AMEM-ABS APR-DATASW] MASK[9.] DEST[AR] LONG $
                ;GET XB PACK IN AR
        D[AR] ROT[29.] ALU[DORQ] DEST[Q] $
                ;POSITION PACK AND SRF TOGETHER
        D[CONST HMNSEC] ALU[DORQ] DEST[DSK-HDA] LONG $
                ;INCLUDE SECTOR, PUT DA IN DSK-HDA
        D[CONST HOMPAG] ROT[9.] DEST[DSK-MEM] PUSHJ[RDSK] LONG $
                ;PUT MEM START ADR IN DSK-MEM AND DO THE READ OF

 BLOCK
        D[AMEM-ABS APR-DATASW] ROT[27.] MASK[9.] DEST[Q] LONG $
                ;GET FILE # IN Q
        D[CONST 0] ALU[D#Q] COND[ZERO] JUMP[UCODE] LONG $
                ;MICROCODE.ETC
        D[CONST 1] ALU[D#Q] COND[ZERO] JUMP[UDIAG] LONG $
                ;MICRODIAG.ETC
        D[CONST 2] ALU[D#Q] COND[ZERO] JUMP[UIMAGE] LONG $
                ;MEMIMG.ETC
        JUMP[XUBOOT] LONG $
                ;LOSER

UCODE:  D[CONST HOMMCA] DEST[Q] JUMP[UDFILE] LONG $
                ;MICROCODE FILE HB INDEX

UDIAG:  D[CONST HOMMDA] DEST[Q] JUMP[UDFILE] LONG $
                ;MICRODIAG FILE HB INDEX

UIMAGE: D[CONST HOMINA] DEST[Q] JUMP[UDFILE] LONG $
                ;MICRODIAG FILE HB INDEX

UDFILE: D[CONST HOMPAG] ROT[9.] ALU[D+Q] DEST[MA] LONG $
                ;LOAD MA WITH LOC OF XB HDWR ADR
        DF/WT D[MEM] MASK[29.] DEST[Q]  LONG $
                ;CONSIDER IT PACK RELATIVE ONLY (MASK OFF PACK #

        D[AMEM-ABS APR-DATASW] MASK[9.] DEST[AR] LONG $
                ;GET PACK # FOR XB (SAME AS HOME BLOCK)
        D[AR] ROT[29.] ALU[DORQ] DEST[DSK-HDA] LONG $
                ;PUT DA IN DSK-HDA
        D[CONST XBPAG] ROT[9.] DEST[DSK-MEM] PUSHJ[RDSK] LONG $
                ;PUT MEM START IN DSK-MEM AND READ XB TO IT
        D[NLIT 777000000000] DEST[DSK-AOB] LONG $
                ;LOAD DSK-AOB WITH -1000,,0 FOR LOOP OVER 1000 W

 XB
        D[AMEM-ABS APR-DATASW] ROT[27.] MASK[9.] DEST[Q] LONG $
                ;GET FILE # IN Q
        D[CONST 2] ALU[D#Q] COND[-ZERO] JUMP[UMERGE] C550 $
                ;JUMP IF NOT DOING MAIN MEMORY IMAGE BOOT


;HERE WHEN WANT TO BOOT MAIN MEMORY IMAGE OFF DISK
```

```
05 0260
05 0261
05 0262
05 0263
05 0264
05 0265
05 0266
05 0267
05 0268
05 0269
 PAGES
05 0270
05 0271
05 0272
05 0273
K ADR
05 0274
```

```
;START FILLING MEMORY AT DATPAG
;TERMINATE WHEN SEE NON EXISTANT PAGE OR DID 512  - DATPAG PAGES
;AOBJN PTR FOR XB ALREADY SETUP FOR 512 PAGES, MUST ADJUST

        D[DSK-AOB] DEST[Q] LONG $
                ;GET -1000,,0 INITIAL AOBJN PTR IN Q
        D[CONST DATPAG] ROT[18.] ALU[D+Q] DEST[DSK-AOB] LONG $
                ;ADJUST FOR RELOCATION OF FILE PAGES IN MEMORY
IMGLUP: D[DSK-AOB] COND[-NEGATIVE] JUMP[IMGDUN] LONG $
                ;IF PAGE COUNTER AOBJN WENT POSITIVE, RAN OUT OF

        D[DSK-AOB] MASK[18.] DEST[Q] LONG $
                ;GET PAGE # IN Q
        D[CONST XBPAG] ROT[9.] ALU[D+Q] DEST[MA] LONG $
                ;LOAD MA WITH ADR OF XB WORD CONTAINING NEXT DIS

        DF/WT D[MEM] DEST[Q] COND[ZERO] JUMP[IMGDUN] LONG $
```

```
05 0275                                                                 ;GET SOFT ADR IN DSK-SDA BUT JUMP IF IT IS ZERO
05 0276                                                         ALU[Q] DEST[DSK-SDA] PUSHJ[CVDSK] LONG $
05 0277                                                                 ;CONVERT TO HARD DA IN DSK-HDA
05 0278                                                         D[DSK-AOB] ROT[9.] DEST[Q] LONG $
05 0279                                                                 ;GET PAGE ADR OF FILE WE ARE READING
05 0280                                                         D[MASK 9.] ROT[9.] ALU[D&Q] DEST[Q] NORM $
05 0281                                                         D[CONST DATPAG] ROT[9.] ALU[D+Q] DEST[DSK-MEM] PUSHJ[RDS
K] LONG $
05 0281
05 0282                                                                 ;RELOCATE THAT PAGE VIA BASE ADR
05 0283                                                                 ;DSK-HDA SETUP, DSK-MEM GET MEM ADR, READ DATA P
G
05 0284                                                         D[DSK-AOB] DEST[Q] LONG $
05 0285                                                                 ;GET AOBJN PTR IN Q
05 0286                                                         D[1,,1] ALU[D+Q] DEST[DSK-AOB] JUMP[IMGLUP] LONG $
05 0287                                                                 ;UPDATE AOBJN PTR AND LUP
05 0288
05 0289                                                 ];.REPEAT 1 - WAITS
05 0290
05 0291                                                 ;HERE IF RAN OUT OF PAGES OR SAW NON-EXISTANT PAGE (TERMINATING
05 0291                                                  CONDITIONS)
05 0292
05 0293 17016 010731370000060554165623655514160000     IMGDUN: D[CONST UBOK] DEST[AR] LONG $
05 0294                                                                 ;SUCCESS CODE
05 0295 17017 010731170000360034161620254114560000             JUMP[HALTED] LONG $
05 0296                                                                 ;MAY NOT BE ANYTHING THERE??
05 0297
05 0298
```

```
06 0299
06 0300
06 0301                                                 ;MERGE POINT FOR DISK AND TAPE MICROCODE BOOTS
06 0302                                                 ;RESPECTIVE AMEMS ETC SET UP
06 0303
06 0304                                                 UMERGE:
06 0305                                                 .REPEAT 1 - WAITS [
06 0306                                                         ALU[0] DEST[GBL-BCT] LONG $
06 0307                                                                 ;INIT TO NO BYTES LEFT BEFORE IO NEEDED
06 0308                                                 ];.REPEAT 1 - WAITS
06 0309 17020 010731170000000014161622254114560000     ULOADL: PUSHJ[GET2BY] LONG $
06 0310                                                                 ;GET 2 BYTES (ADR) LEFT JUSTIFIED IN AR
06 0311 17021 010731170000060544041303654114160000             D[AR] ROT[16.] MASK[16.] DEST[GBL-UAD] LONG $
06 0312                                                                 ;SAVE RIGHT JUSTIFIED 16 BIT ADR FOR LATER
06 0313 17022 010731170000000014161622254114560000             PUSHJ[GET4BY] LONG $
06 0314                                                                 ;Get 4 bytes (00:31) left justified in AR
06 0315 17023 010731170002060540160043666114170000             D[GBL-UAD] ROT[MUA-ROT] DEST[JMEM] MU-PUSH LONG $
06 0316                                                                 ;Left justified adr on stack
06 0317 17024 010731170000060554161763654114160000             DEST[MM-PRE-WRT] $
06 0318                                                                 ;Get the MM write started -- it will happen NEXT
 cycle !

06 0318
06 0319 17025 010731042104060554160004254114160000
D[AR] DEST[MM-A] SPEC[MM-ACCESS] COND[-MM-ACC-COND] POPJ LONG $
06 0320                                                                 ;DEPOSIT BITS 00:31 FROM AR[00:31] VIA JADR ON S
TACK
06 0321                                                                 ;POPJ NEVER GETS A CHANCE
06 0322 17026 010731170000000014161622254114560000             PUSHJ[GET4BY] LONG $
06 0323                                                                 ;GET 4 BYTES (32:63) LEFT JUSTIFIED IN AR
06 0324 17027 010730170000060543361623667514160000             D[OLD-MMB-FLG] ROT[13.] DEST[Q] $
06 0325                                                                 ;See if we need to permute fields.
06 0326 17030 010641000000000014161622266114560000             D[GBL-UAD] ALU[D&Q] C550 COND[-ZERO] PUSHJ[OLD-MMB-FIX]
$
06 0327 17031 010731170002060540160043666114170000             D[GBL-UAD] ROT[MUA-ROT] DEST[JMEM] MU-PUSH LONG $
06 0328                                                                 ;LEFT JUSTIFIED ADR ON STACK
06 0329 17032 010731170000060554161763654114160000             DEST[MM-PRE-WRT] $
06 0330                                                                 ;Get the MM write started -- it will happen NEXT
 cycle !

06 0330
06 0331 17033 010731042204060554160004254114160000
D[AR] DEST[MM-B] SPEC[MM-ACCESS] COND[-MM-ACC-COND] POPJ LONG $
06 0332                                                                 ;DEPOSIT BITS 32:63 FROM AR[00:31] VIA JADR
06 0333                                                                 ;POPJ NEVER GETS A CHANCE
06 0334 17034 010731170000000014161622254114560000             PUSHJ[GET3BY] LONG $
06 0335                                                                 ;GET 3 BYTES (64:87) LEFT JUSTIFIED IN AR
06 0336 17035 010731170002060540160043666114170000             D[GBL-UAD] ROT[MUA-ROT] DEST[JMEM] MU-PUSH LONG $
06 0337                                                                 ;LEFT JUSTIFIED ADR ON STACK
06 0338 17036 010731170000060554161763654114160000             DEST[MM-PRE-WRT] $
06 0339                                                                 ;Get the MM write started -- it will happen NEXT
 cycle !
```

```
06 0339
06 0340 17037 010731042304060554160004254114160000
D[AR] DEST[MM-C] SPEC[MM-ACCESS] COND[-MM-ACC-COND] POPJ LONG $
06 0341                                                         ;DEPOSIT BITS 64:87 FROM AR[00:23] VIA JADR
06 0342                                                         ;POPJ NEVER GETS A CHANCE
06 0343 17040 010731170000000014161622254114560000             PUSHJ[GET2BY] LONG $
06 0344                                                         ;THROW AWAY GHOST BYTES
06 0345 17041 010730170000000037777623655114160000             D[LIT TRMADR] DEST[Q] LONG $
06 0346                                                         ;GET TRMADR IN Q
06 0347 17042 010661000000360414161620266114560000             D[GBL-UAD] ALU[D#Q] COND[-ZERO] JUMP[ULOADL] LONG $
06 0348                                                         ;IF ADR JUST LOADED .NE. TRMADR THEN JUMP TO CON
TINUE
06 0349 17043 010731370000060554165623655514160000             D[CONST UBOK] DEST[AR] LONG $
06 0350                                                         ;SUCCESS CODE
06 0351 17044 010731170000000000160007266114560000             D[GBL-UAD] ROT[MUA-ROT] ODISP LONG $
06 0352                                                         ;JUMP TO IT (XFER INSTRUCTION)
06 0353
06 0354                                                 OLD-MMB-FIX:
06 0355 17045 010730170000060554161623654114160000             D[AR] DEST[Q] $
06 0356                                                          ;Prepare to permute fields.
06 0357 17046 010650170000060541465623647114160000             D[MASK 26.] ROT[35. - 29.] ALU[-D&Q] DEST[Q] $
06 0358                                                          ;Clear space for JADR, ROT, and MASK fields.
06 0359 17047 010731170000060544435503654114160000             D[AR] ROT[17. + 1] MASK[14.] DEST[HOLD] $
06 0360                                                          ;Put JADR in new place.
06 0361 17050 010630170000060541561623654515160000             D[MEM] ROT[35. - 29.] ALU[DORQ] DEST[Q] $
06 0362 17051 010731170000060547431503654114160000             D[AR] ROT[29. + 1] MASK[12.] DEST[HOLD] $
06 0363                                                          ;Put ROT and MASK in place.
06 0364 17052 010631370000060545161624254515160000             D[MEM] ROT[35. - 15.] ALU[DORQ] DEST[AR] POPJ $
06 0365
06 0366 17053 010731370021060554167623655514160000     XUBOOT: D[CONST UBBAD] DEST[AR] SPEC[MHS-DISABLE] $
06 0367                                                          ;Failure code... turn off Macro History.
06 0368 17054 010731170017060554161623654114160000             SPEC[UHS-DISABLE] $JUMP[HALTED] LONG $
06 0368 17055 010731170000360034161620254114560000
06 0369                                                                 ;Turn off Micro History, hang in magic spot.
06 0370
06 0371
06 0372
06 0373
```

```
07 0374
07 0375
07 0376                                                 ;GET 1 BYTE LEFT JUSTIFIED IN AR
07 0377
07 0378 17056 010731170000000014161622254114560000     GET1BY: PUSHJ[ILDB] LONG $
07 0379                                                                 ;GET RIGHT JUSTIFED
07 0380 17057 010731370000060547161624254114160000             D[AR] ROT[36. - 8.] DEST[AR] POPJ LONG $
07 0381                                                                 ;LEFT JUSTIFY IT AND RETURN
07 0382
07 0383                                                 ;GET 2 BYTES LEFT JUSTIFIED IN AR
07 0384
07 0385 17060 010731170000361354161622254114560000     GET2BY: PUSHJ[GET1BY] LONG $
07 0386                                                                 ;GET 1ST LEFT JUSTIFIED IN AR
07 0387 17061 010731170000060554161223654114160000             D[AR] DEST[GBL-GT2] LONG $
07 0388                                                                 ;SAVE 1ST IN TEMP
07 0389 17062 010731170000361354161622254114560000             PUSHJ[GET1BY] LONG $
07 0390                                                                 ;GET 2ND LEFT JUSTIFIED IN AR
07 0391 17063 010730170000060547161623654114160000             D[AR] ROT[36. - 8.] DEST[Q] LONG $
07 0392                                                                 ;POSITION 2ND IN Q
07 0393 17064 010631370000060554161623664514160000             D[GBL-GT2] ALU[IORQ] DEST[AR] LONG $
07 0394                                                                 ;IOR 1ST AND 2ND INTO AR
07 0395 17065 010731170000060554161624254114160000             POPJ LONG $
07 0396                                                                 ;RETURN
07 0397
07 0398                                                 ;GET 3 BYTES LEFT JUSTIFIED IN AR
07 0399
07 0400 17066 010731170000361414161622254114560000     GET3BY: PUSHJ[GET2BY] LONG $
07 0401                                                                 ;GET 1ST 2 LEFT JUSTIFIED IN AR
07 0402 17067 010731170000060554161243654114160000             D[AR] DEST[GBL-GT3] LONG $
07 0403                                                                 ;SAVE 1ST 2 IN TEMP
07 0404 17070 010731170000361354161622254114560000             PUSHJ[GET1BY] LONG $
07 0405                                                                 ;GET 3RD LEFT JUSTIFIED IN AR
07 0406 17071 010730170000060545161623654114160000             D[AR] ROT[36. - 16.] DEST[Q] LONG $
07 0407                                                                 ;POSITION 3RD IN Q
07 0408 17072 010631370000060554161623665114160000             D[GBL-GT3] ALU[IORQ] DEST[AR] LONG $
07 0409                                                                 ;IOR 1ST 2 AND 3RD INTO AR
07 0410 17073 010731170000060554161624254114160000             POPJ LONG $
07 0411
07 0412                                                 ;GET 4 BYTES LEFT JUSTIFIED IN AR
07 0413
07 0414 17074 010731170000361414161622254114560000     GET4BY: PUSHJ[GET2BY] LONG $
07 0415                                                                 ;GET 1ST 2 LEFT JUSTIFIED IN AR
07 0416 17075 010731170000060554161263654114160000             D[AR] DEST[GBL-GT4] LONG $
07 0417                                                                 ;SAVE 1ST 2 IN TEMP
07 0418 17076 010731170000361414161622254114560000             PUSHJ[GET2BY] LONG $
07 0419                                                                 ;GET 2ND 2 LEFT JUSTIFIED IN AR
07 0420 17077 010730170000060545161623654114160000             D[AR] ROT[36. - 16.] DEST[Q] LONG $
07 0421                                                                 ;POSITION 2ND 2 IN Q
07 0422 17100 010631370000060554161623665514160000             D[GBL-GT4] ALU[IORQ] DEST[AR] LONG $
07 0423                                                                 ;IOR 1ST 2 AND 2ND 2 INTO AR
07 0424 17101 010731170000060554161624254114160000             POPJ LONG $
07 0425
```

07 0426

```
08 0427
08 0428
08 0429                                                 ;ACCEPTS PTR IN GBL-BPT.
08 0430                                                 ;ACCEPTS # BYTES LEFT IN GBL-BCT.
08 0431                                                 ;IF NO BYTES LEFT, DO IO AND THEN GET BYTE.
08 0432                                                 ;ALWAYS RETURNS WITH DESIRED BYTE UNLESS ERROR.
08 0433
08 0434 17102 0107200040000000141616202665145600       ILDB:   D[GBL-BCT] ALU[D-1] DEST[Q] COND[NEGATIVE] JUMP[NEEDIO]
LONG $
08 0435                                                                 ;DEC BYTES LEFT, JUMP TO DO IO IF NECESSARY
08 0436 17103 0102311700000605541613236541141600                       ALU[Q] DEST[GBL-BCT] LONG $
08 0437                                                                 ;UPDATE AMEM
08 0438 17104 0107303700000605541616236671141600                       D[GBL-BPT] DEST[Q AR] LONG $
08 0439                                                                 ;SETUP FROM PTR
08 0440 17105 0107313700000000030145022541145700                       D[AR] ROT[12.] MASK[6.] DEST[AR MASKR] PUSHJ[IBP] LONG $
08 0441                                                                 ; Get S field and call common code
08 0442 17106 7107313700000605541616236671141600                       D[GBL-BPT] DEST[AR MA] LONG $
08 0443                                                                 ;AR GETS NEW PTR, MA GETS ADR OF DATA
08 0444 17107 0107301700000605541116236555141600                       D[CONST 36.] DEST[Q] LONG $
08 0445                                                                 ;Q GET 36.
08 0446 17110 0116111700000605414144436541121700                       DFRQ D[AR] ROT[6.] MASK[6.] ALU[Q-D] DEST[ROTR] LONG $
08 0447                                                                  ;36.-P to ROTR, fetch data
08 0448 17111 0107313700000605521216242545151600                       D[MEM] ROT[R] MASK[R] DEST[AR] POPJ LONG $
08 0449                                                                  ; Get and store byte IN AR, RETURN
08 0450
08 0451                                                 ;ACCEPTS PTR IN Q
08 0452                                                 ;S IN AR
08 0453                                                 ;INCREMENTS AND STORES IN GBL-BPT
08 0454
08 0455 17112 0116110300000605475613442541141600       IBP:    D[AR] ROT[30.] ALU[Q-D] DEST[GBL-BPT] COND[CRYO] POPJ LO
NG $
08 0456                                                                 ;Subtract from P (in place) and store, checking
for
08 0456                                                  overflow
08 0457 17113 0106401700000605540756236471141600                       D[MASK 30.] ALU[D&Q] DEST[Q] LONG $
08 0458                                                                 ; ptr will overflow, fix it up
08 0459 17114 0116001700003622475116202555145600                       D[CONST 44] ROT[30.] ALU[D+Q+1] DEST[Q] JUMP[IBP] LONG $
08 0460                                                                 ;New P of 44, add 1 to adr
08 0461
08 0462
08 0463
08 0464 17115 0107301700000003444236236701141600       NEEDIO: D[AMEM-ABS APR-DATASW] ROT[18.] MASK[9.] DEST[Q] LONG $
08 0465                                                                 ;GET ALREADY CHECKED DEVICE CODE IN Q.
08 0466 17116 0106610020000000140016202555145600                       D[CONST 0] ALU[D#Q] COND[ZERO] JUMP[IODSK] LONG $
08 0467                                                                 ;JUMP FOR DISK
08 0468 17117 0107311700000000141616222541145600       IOTAP:  PUSHJ[TAPIO] LONG $
08 0469                                                                 ;SUCK IN NEXT RECORD, SETUP GBL-BCT
08 0470 17120 0107311700000000141616202541145600                       JUMP[IOBOTH] LONG $
08 0471                                                                 ;SETUP PTR AND CONTINUE
08 0472
08 0473                                                 IODSK:
08 0474                                                 .REPEAT 1 - WAITS [
```

```
08 0475                                                                 
08 0476                                                                 
08 0477                                                                 
08 0478                                                                 
08 0479                                                                 
t no
08 0479                                                                 
08m0480                                                                 
08m0480 17121 010731170000361274161620254114560000                     
t no
08m0480                                                                 
08 0480                                                                 
08 0481 17122 010731171102000012001343655114160000                     
08 0482                                                                 
08 0483 17123 010731170000362054161620254114560000                     
08 0484                                                                 
08 0485                                                                 
```

```
        PUSHJ[DSKIO] LONG $
                ;SUCK IN NEXT PAGE, SETUP GBL-BCT
];.REPEAT 1 - WAITS
.REPEAT WAITS [
        JUMP[XUBOOT] $          ;That's all there is, there ain'

 more!
];[
        JUMP[XUBOOT] $          ;That's all there is, there ain'

 more!
].REPEAT
IOBOTH: DI[NLIT BYTPTR] DEST[GBL-BPT] LONG $
                ;INIT BYTE PTR FOR ILDB OF OTH BYTE OF DATA PAGE
        JUMP[ILDB] LONG $
                ;DO ILDB AGAIN
```

```
09 0486
09 0487
09 0488                                                 .REPEAT WAITS [
09 0489                                                         PROM = 1
09 0490                                                         .INSERT DSKBT.NEW[1,111]
09m0491                                                 ];[
09m0491         1               PROM = 1
09m0491                                                         .INSERT DSKBT.NEW[1,111]
```

```
01m0001                                                 DSKBT-BEG:
01m0001                                                 .REPEAT 1 + (DSKBT-BEG ) 0) [
01m0001
01m0001                                                 .INSERT SWINIT.SLO[1,114]
01m0001                                                 .REPEAT 1 - F4SW [
01m0001                                                 NEWMAP = 1
01m0001                                                 XUCODE = 1
01m0001                                                 .INSERT CFDEF.SLO       ; get machine definition
01m0001                                                 PROM = 0
01m0001                                                 LLOAD-ROT = 0
01m0001                                                 :17400
01m0001                                                 ];.REPEAT 1 - F4SW
01m0001                                                 .REPEAT F4SW [
01m0001                                                 .INSERT F41ON.SLO[1,114]
01m0001                                                 .INSERT F4DEF.SLO[1,114]
01m0001                                                 PROM = 0
01m0001                                                 ubloc = 17000
01m0001                                                 :16400
01m0001
01m0001                                                 ];.REPEAT F4SW
01m0001                                                 ];.REPEAT 1 + (DSKBT-BEG ) 0)
01m0001
01m0001                                                 .REPEAT F4SW [
01m0001                                                 modify-field alu-dst
01m0001                                                         O_AC = 2
01m0001                                                    $
01m0001                                                 modify-field CYLEN
01m0001                                                         FIXM = no-cdc cy0       4;12            ;350
01m0001                                                    $
01m0001                                                 ];[
01m0001                                                 modify-field alu-dst
01m0001         000007000000000000000000000000000000            O_AC = 2
01m0001                                                    $
01m0001                                                 modify-field CYLEN
01m0001         000000000000000000000000036016400               FIXM = no-cdc cy0       4;12            ;350
01m0001         000000000000000000000000010016000
01m0001                                                    $
01 0001                                                 ].REPEAT F4SW
01 0002
01 0003                                                 LIST
01 0004
01 0005         1 F4PROM = PROM * F4SW
01 0006
01 0007         10        DSKUDEV = 10              ;Micro-device for DSK
01 0008         2 APR-SW = 2                  ;Subselect for data switches
01 0009         4 APR-ASW = 4                 ;Subselect for address switches
01 0010
01 0011                                                 ;MAPF for disk
01 0012         4 DSK-RD-CMD = 4              ;Read command and status
01 0013         4 DSK-LD-CMD = 4              ;Load command
01 0014         5 DSK-LD-MA = 5               ;Load memory address
01 0015         6 DSK-LD-DA = 6               ;Load disk address
```

```
01 0016        7 DSK-LD-ECC = 7          ;Load special function (INIT, GO)
01 0017
01 0018                                                ;Commands for LD-ECC
01 0019        1 DSK-GO = 1              ;Command to start a command
01 0020        2 DSK-INIT = 2            ;Command to initialize disk
01 0021
01 0022        10        NSECTORS = 8            ;Number of sectors/surface
01 0023        3 LOG2NSEC = 3            ;Log 2 of NSECTORS
01 0024        23        NHEADS = 19.            ;Number of heads/cylinder
01 0025
01 0026        7 BOOT-DRIVE = 7          ;Unit number of drive to boot from
01 0027        1 cdc160kludge = 1        ;*** Hack disk address
01 0028
01 0029                                                .REPEAT F4PROM [
01 0030                                                BYTPTR = 441000000074   ;Use different buffer than TENEX does
01m0031                                                ];[
01m0031        41000,,74 BYTPTR = 441000000074   ;Use different buffer than TENEX does
01 0031                                                ].REPEAT F4PROM
01 0032
01 0033                                                .REPEAT 1 - F4PROM [
01 0034
01 0035                                                HDSKBT: JUMP[DSKUBT] NORM $
01 0036                                                        ;Boot microcode from WAITS.UCD[1,2]
01 0037                                                JUMP[DSKMBT] NORM $
01 0038                                                        ;Boot macrocode from WAITS.DMP[1,2]
01 0039                                                JUMP[DSKUBS] NORM $
01 0040                                                        ;Boot macrocode, file name in switches, UFD in a
ddress
01 0041                                                JUMP[DSKMBS] NORM $
01 0042                                                        ;Boot macrocode, file name in switches, UFD in a
ddress
01 0043                                                JUMP[DSKDBT] NORM $
01 0044                                                        ;Load DSKDMP
01 0045                                        .REPEAT 1 - F4SW [
01 0046                                                D[CONST (HDSKBT / 400)] ROT[8] SDISP $
01 0047                                                        ;Access to high memory.
01 0048                                        ];.REPEAT 1 - F4SW
01 0049                                        .REPEAT F4SW [
01 0050                                                NOP $
01 0051                                        ];.REPEAT F4SW
01 0052                                                JUMP[DSKUSW] NORM $
01 0053                                                        ;Access to switch micro-boot. Skipped for microb
oot
01 0054                                                JUMP[DSKMSW] NORM $
01 0055                                                        ;Access to switch macro-boot. Skipped for microb
oot
01 0056                                        .repeat 1 - f4sw [
01 0057                                                CONTA[SWAITS + 5] $
01 0058
01 0059                                        ;Boot microcode from WAITS.UCD[1,2]
01 0060                                        DSKUBT: D[CONST 1] DEST[Q] PUSHJ[DSKRCL] NORM $
01 0061                                                        ;Do recalibrate to be safe
01 0062                                                        ;Read MFD starting at track 1
01 0063                                                D[CONST 0] DEST[MA] PUSHJ[DSKBTO] NORM $
```

```
01 0064
01 0065
01 0066
01 0067
01 0068
01 0069
01 0070
01 0071
01 0072
01 0073
01 0074
01 0075
01 0076
memory
```

```
        ;Read starting at word 0
D[1,,1] ALU[D+1] DEST[Q] PUSHJ[SRCMFN] $
        ;Search [1,1] for [1,2]
ALU[Q] DEST[O] SPEC[A-MEM-APR&DEST-A-MEM] NORM $
        ;Set address of UFD
D[CONST O] DEST[MA] PUSHJ[DSKBTO] NORM $
        ;Read retrieval and UFD
PUSHJ[SWAITS] NORM $
        ;Set file name
PUSHJ[SRCUCD] NORM $
        ;Search UFD for WAITS.UCD
PUSHJ[DSKUB1] NORM $
        ;Now, read in the file and stuff into microcode
```

```
01 0077                                                                 D[10] SPEC[A-MEM-APR] DEST[Q] JUMP[DSKMB2] $
01 0078                                                                         ;Read .DMP file as well.
01 0079                                                         ;       ---
01 0080
01 0081                                                         ];.repeat 1 - f4sw
01 0082
01 0083                                                         ];.REPEAT 1 - F4PROM
01 0084
01 0085                                                         ;Boot macrocode from WAITS.DMP[1,2]
01 0086 17124 010730170000000014003622255551456000             DSKMBT: D[CONST 1] DEST[Q] PUSHJ[DSKRCL] NORM $
01 0087                                                                 ;Do recalibrate to be safe
01 0088                                                                 ;Read MFD starting at track 1
01 0089 17125 710731170000000014001622255551456000                     D[CONST 0] DEST[MA] PUSHJ[DSKBTO] NORM $
01 0090                                                                 ;Read starting at word 0
01 0091 17126 011700170037000014003622255551456000                     D[1,,1] ALU[D+1] DEST[Q] PUSHJ[SRCMFN] $
01 0092                                                                 ;Search [1,1] for [1,2]
01 0093                                                         DSKMB2:
01 0094                                                         ;This stuff can go away when WAITS does this chore.
01 0095                                                         .REPEAT 1 - F4SW [
01 0096                                                                 D[CONST 1] ROT[18.] DEST[HI-ABS-MA] PUSHJ[CORCLR] $
01 0097                                                                 ;Clear upper moby
01 0098                                                                 D[CONST 0] ROT[18.] DEST[HI-ABS-MA] PUSHJ[CORCLR] $
01 0099                                                                 ;Clear lower moby
01 0100                                                         ];.REPEAT 1 - F4SW
01 0101 17127 710731170000000014001622255551456000                     D[CONST 0] DEST[MA] PUSHJ[DSKBTO] NORM $
01 0102                                                                 ;Read retrieval and UFD
01 0103 17130 010731170000000014161622254114560000                     PUSHJ[SWAITS] NORM $
01 0104                                                                 ;Set file name
01 0105 17131 010731170000000014161622254114560000                     PUSHJ[SRCDMP] NORM $
01 0106                                                                 ;Search UFD for WAITS.DMP
01 0107 17132 010731170000000014161620254114560000                     JUMP[DSKMB1] NORM $
01 0108                                                                 ;Now, read in the file and untangle .DMP file
01 0109                                                         ;       ---
01 0110
01 0111                                                         .REPEAT 1 - F4SW [
01 0112                                                         ;Boot microcode from file name in switches
01 0113                                                         DSKUBS: D[CONST 1] DEST[Q] PUSHJ[DSKRCL] NORM $
01 0114                                                                 ;Do recalibrate to be safe
01 0115                                                                 ;Read MFD starting at track 1
01 0116                                                                 D[CONST 0] DEST[MA] PUSHJ[DSKBTO] NORM $
01 0117                                                                 ;Read starting at word 0
01 0118                                                                 D[1,,1] ALU[D+1] DEST[Q] PUSHJ[SRCMFN] $
01 0119                                                                 ;Search [1,1] for [1,2]
01 0120                                                                 ALU[Q] DEST[O] SPEC[DEST-A-MEM] NORM $
01 0121                                                                 ;Set address of UFD
01 0122                                                                 D[CONST 0] DEST[MA] PUSHJ[DSKBTO] NORM $
01 0123                                                                 ;Read retrieval and UFD
01 0124                                                                 D[CONST 0] DEST[DEV-ADR] SPEC[IOB-IN] NORM $
01 0125                                                                 ;Start reading switches to get disk address
01 0126                                                                 ;Setup to read console switches
01 0127                                                                 MAPF[APR-SW] CYLEN[IOB-IN] D[IOD] DEST[AR] PUSHJ[SRCUCD]
 $
```

```
01 0128                                                                 ;Read file name and search UFD
01 0129                                                         PUSHJ[DSKUB1] NORM $
01 0130                                                                 ;Now, read in the file and stuff into microcode
memory
01 0131                                                 ];.REPEAT 1 - F4SW
01 0132 17133 010731170000362674161620254114560000             jump[.] $
01 0133                                                                 ;Hang for now.
01 0134                                                 ;       ---
01 0135
01 0136                                                 .REPEAT 1 - F4PROM [
01 0137                                                 ;Boot macrocode from DSKDMP.DMP[1,2]
01 0138                                                 DSKDBT: D[CONST 1] DEST[Q] PUSHJ[DSKRCL] NORM $
01 0139                                                                 ;Do recalibrate to be safe
01 0140                                                                 ;Read MFD starting at track 1
01 0141                                                         D[CONST 0] DEST[MA] PUSHJ[DSKBTO] NORM $
01 0142                                                                 ;Read starting at word 0
01 0143                                                         D[1,,1] ALU[D+1] DEST[Q] PUSHJ[SRCMFN] $
01 0144                                                                 ;Search [1,1] for [1,2]
01 0145                                                         D[CONST 0] DEST[MA] PUSHJ[DSKBTO] NORM $
01 0146                                                                 ;Read retrieval and UFD
01 0147                                                         PUSHJ[SDSKDM] NORM $
01 0148                                                                 ;Set file name
01 0149                                                         PUSHJ[SRCDMP] NORM $
01 0150                                                                 ;Search UFD for WAITS.DMP
01 0151                                                         JUMP[DSKMB1] NORM $
01 0152                                                                 ;Now, read in the file and untangle .DMP file
01 0153                                                 ;       ---
01 0154
01 0155                                                 ;Boot macrocode from file name in switches
01 0156                                                 DSKMBS: D[CONST 1] DEST[Q] PUSHJ[DSKRCL] NORM $
01 0157                                                                 ;Do recalibrate to be safe
01 0158                                                                 ;Read MFD starting at track 1
01 0159                                                         D[CONST 0] DEST[MA] PUSHJ[DSKBTO] NORM $
01 0160                                                                 ;Read starting at word 0
01 0161                                                         D[1,,1] ALU[D+1] DEST[Q] PUSHJ[SRCMFN] $
01 0162                                                                 ;Search [1,1] for [1,2]
01 0163                                                         D[CONST 0] DEST[MA] PUSHJ[DSKBTO] NORM $
01 0164                                                                 ;Read retrieval and UFD
01 0165                                                         D[CONST 0] DEST[DEV-ADR] SPEC[IOB-IN] NORM $
01 0166                                                                 ;Start reading switches to get disk address
01 0167                                                                 ;Setup to read console switches
01 0168                                                         MAPF[APR-SW] CYLEN[IOB-IN] D[IOD] DEST[AR] PUSHJ[SRCDMP]
 $
01 0169                                                                 ;Read file name and search UFD for .DMP file
01 0170                                                         JUMP[DSKMB1] NORM $
01 0171                                                                 ;Now, read in the file and untangle .DMP file
01 0172                                                 ;       ---
01 0173
01 0174                                                 ;Boot microcode from address in switches
01 0175                                                 DSKUSW: D[CONST 0] DEST[DEV-ADR] JUMP[. + 2] $
01 0176                                                                 ;Setup to read console switches
01 0177                                                         SPEC[IOB-IN] NORM $
01 0178                                                                 ;Start reading switches to get disk address
01 0179                                                         MAPF[APR-SW] CYLEN[IOB-IN] D[IOD] DEST[Q] PUSHJ[DSKRCL]
```

```
$
01 0180
01 0181
01 0182
01 0183
01 0184
01 0185
01 0186
 are
01 0186
01 0187
01 0188
01 0189
] NORM $

01 0189
```

```
                ;Recalibrate, just to be sure
DSKUB1:  D[CONST 0] DEST[MA] PUSHJ[DSKBTO] NORM $
                ;Start reading into zero
                ;Read retrieval and file
.REPEAT 1 - F4SW [
        D[CONST (200 / 100)] ROT[6] DEST[MA] NORM jump[. + 1] $
                ;First place in memory to read. (First 200 words

 retrieval)
        D[CONST (0 / 200)] ROT[8] DEST[AR] CYLEN[FIXM] $
                ;First place to read boot into is 17000 for now
        D[CONST ((16000 - 0) / 200)] ROT[7] ALU[D-1] DEST[IR-ADR
```

```
                                                    ;Number of words to transfer
                                   DSKUB3: D[MEM] DEST[MUCODE-HI] LONG $
                                                    ;Copy for macrocode memory into microcode memory (high word)
                                           D[MA] ALU[D+1] DEST[MA] NORM $
                                                    ;Fetch next memory location
                                           CYLEN[FIXM] $
                                                    ;Wait for fetch
                                           D[MEM] DEST[MUCODE-LO] LONG $
                                                    ;Copy for macrocode memory into microcode memory (low word)
                                           D[MA] ALU[D+1] DEST[MA Q] NORM $
                                                    ;Fetch next memory location
                                           D[MA] MASK[9 + 3] COND[-OBUS=0] JUMP[DSKUB4] C550 $
                                                    ;Are we on a new track?
                                           D[CONST (200 / 100)] ROT[6] ALU[D+Q] DEST[MA] NORM $
                                                    ;Yes, Step over retrieval!
                                   DSKUB4: D[AR] ALU[D+1] DEST[AR] NORM $
                                                    ;Advance microcode address
                                           D[IR] MASK[22] ALU[D-1] DEST[IR-ADR] COND[-OBUS<0] JUMP[DSKUB3] C550 $
                                                    ;Count words to transfer
                                           D[CONST 0] DEST[DEV-ADR] SPEC[IOB-IN] NORM $
                                                    ;Begin-reading switches
                                           MAPF[APR-ASW] CYLEN[IOB-IN] D[IOD] DEST[AR] $
                                                    ;Finish reading switches
                                           D[AR] ROT[2] COND[-OBUS<0] POPJ $
                                                    ;If STOP is off, boot system as well
                                   ];.REPEAT 1 - F4SW
                                           jump[.] $
                                                    ;Hang for now.

                                   ;Boot macrocode from address in switches
                                   DSKMSW: D[CONST 0] DEST[DEV-ADR] JUMP[. + 1] $
                                                    ;Setup to read console switches
                                           SPEC[IOB-IN] NORM $
                                                    ;Make sure the map is turned off.
                                                    ;Start reading switches to get disk address
                                           MAPF[APR-SW] CYLEN[IOB-IN] D[IOD] DEST[Q] PUSHJ[DSKRCL] $
                                                    ;Recalibrate, just to be sure
                                   ];.REPEAT 1 - F4PROM
                                   ;        \ /
17134 71073117000000001400162225551456000   DSKMB1: D[CONST 0] DEST[MA] PUSHJ[DSKBT0] NORM $
                                                    ;Start reading into zero
                                                    ;Read retrieval and file
17135 01073017000006055416162365651416000           D[MA] DEST[Q] NORM $
                                                    ;Save number of words read in Q for easy testing
```

```
01 0234 17136 710731170000362761405620255514560000
01 0235
 are
01 0235
01 0236 17137 010733170000060554171623655514160000
01 0237
rts at
01 0237
01 0238 17140 010731000000000014031620256512560000
01 0239
01 0240
01 0241 17141 011621006000000014161620256514560000
01 0242
01 0243 17142 010732170000060554161503656514160000
01 0244
01 0245 17143 010503170000060541405623655514160000
01 0246
01 0247 17144 710732170000060554161623654515160000
01 0248
01 0249 17145 010731170000060554161623654112160000
01 0250
01 0251 17146 010731170000060554161503654515160000
01 0252
ffer
01 0253 17147 711702170000060554161643656514160000
01 0254
01 0255
 cycle
01 0256 17150 711702170000363014161620256514560000
01 0257
01 0258
cycle
01 0259
01 0260 17151 710731170000060540317623655514160000
01 0261
01 0262 17152 030731170000060554161503654513160000
01 0263 17153 710731170000060554075643655514160000
01 0264 17154 010731170000060554045503647114160000
01 0265
01 0266 17155 710731170000060554077643655514160000
01 0267 17156 710731170000060540625623655514160000
01 0268
01 0269 17157 010731170000060554161623654112160000
01 0270
01 0271 17160 710731170000060554045463654515160000
01 0272
01 0273
01 0274
01 0275
01 0276
01 0277
01 0278
01 0279
01 0280
```

```
        D[CONST (200 / 100)] ROT[6] DEST[MA] NORM jump[. + 1] $
                ;First place in memory to read. (First 200 words
 retrieval)
        D[CONST 74] DEST[AC] NORM $
                ;First place in memory to write. (.DMP files sta
 JOBSAV)
DSKMB3: DFRQ D[MA] MASK[9 + 3] COND[-OBUS=0] JUMP[DSKMB4] C550 $
                ;Are we on a new track?  If not, we're OK
                ;Wait for fetch to complete
        D[MA] ALU[D-Q] COND[-OBUS<0] JUMP[DSKMB5] C550 $
                ;Yes, check for end-of-transfer
        D[MA] DEST[O_AC HOLD] $
                ;Save AC, copy MA into AC
        D[CONST (200 / 100)] ROT[6] ALU[D+AC] DEST[AC] NORM $
                ;Step over retrieval!
        D[MEM] DEST[O_AC MA] NORM $
                ;Restore AC.  Set MA
        DFRQ CYLEN[FIXM] $
                ;Wait for memory
DSKMB4: D[MEM] DEST[HOLD] NORM $
                ;Copy from memory read buffer to memory write bu

        D[MA] ALU[D+1] DEST[O_AC MA STRT-WRT] NORM $
                ;Increment fetch address and save in AC
                ;Move old contents on AC into MA and start write

        D[MA] ALU[D+1] DEST[O_AC MA] JUMP[DSKMB3] NORM $
                ;Increment store address and save in AC
                ;Move old contents on AC into MA and start read

;       ---
DSKMB5: D[CONST (116 / 2)] ROT[1] DEST[MA] NORM $
                ;Copy JOBSYM for EDDT
        DF/WT D[MEM] DEST[HOLD] LONG $
        D[CONST 36] DEST[MA STRT-WRT] NORM $
        D[MASK 18.] DEST[HOLD] NORM $
                ;Set memory size for EDDT
        D[CONST 37] DEST[MA STRT-WRT] NORM $
        D[CONST (120 / 10)] ROT[3] DEST[MA] NORM $
                ;Read JOBSA
        DFRQ CYLEN[FIXM] $
                ;Wait for fetch
        D[MEM] MASK[18.] DEST[MA PC] NORM $
                ;Set starting address
.REPEAT 1 - F4SW [
        D[CONST 40] ROT[6] DEST[Q] NORM $
                ;Set low 12 bits of address
        ALU[0] C600 SDISP $
                ;Go to DWP's kludge at 10000!!!
];.REPEAT 1 - F4SW
.REPEAT F4SW [
        D[CONST 70] DEST[AR] JUMP[UBLOC + 1] $
```

```
                            ;Jump to starting address
];[
17161 0107313700003600341616202555145600 0
               D[CONST 70] DEST[AR] JUMP[UBLOC + 1] $
                            ;Jump to starting address
].REPEAT F4SW
;              ---

.REPEAT F4PROM [
;Interface to F4 Console Computer
UBOOTD:        D[CONST 1] DEST[Q] PUSHJ[DSKRCL] NORM $
                            ;Do recalibrate to be safe
                            ;Read MFD starting at track 1
               D[CONST 0] DEST[MA] PUSHJ[DSKBTO] NORM $
                            ;Read starting at word 0
               D[1,,1] ALU[D+1] DEST[Q] PUSHJ[SRCMFN] $
                            ;Search [1,1] for [1,2]
               D[CONST 0] DEST[MA] PUSHJ[DSKBTO] NORM $
                            ;Read retrieval and UFD
               PUSHJ[SWAITS] NORM $
```

```
01 0297                                                 ;Set file name
01 0298                                         PUSHJ[SRCXXX] NORM $
01 0299                                                 ;Search for appropriate extension
01 0300                                 DSKCB1: D[CONST 50] ROT[6] DEST[MA] PUSHJ[DSKBTO] NORM $
01 0301                                                 ;Start reading into zero
01 0302                                                 ;Read retrieval and file
01 0303                                         D[MA] DEST[Q] NORM $
01 0304                                                 ;Save number of words read in Q for easy testing
01 0305                                         D[CONST (5200 / 100)] ROT[6] DEST[MA] NORM jump[. + 1] $
01 0306                                                 ;First place in memory to read. (First 200 words
 are
01 0306                                                  retrieval)
01 0307                                         D[CONST 50] ROT[6] DEST[AC] NORM $
01 0308                                                 ;First place in memory to write. (.DMP files sta
rts at
01 0308                                                  JOBSAV)
01 0309                                 DSKCB3: DFRQ D[MA] MASK[9 + 3] COND[-OBUS=0] JUMP[DSKCB4] C550 $
01 0310                                                 ;Are we on a new track?  If not, we're OK
01 0311                                                 ;Wait for fetch to complete
01 0312                                         D[MA] ALU[D-Q] COND[-OBUS<0] JUMP[DSKCB5] C550 $
01 0313                                                 ;Yes, check for end-of-transfer
01 0314                                         D[MA] DEST[O_AC HOLD] $
01 0315                                                 ;Save AC, copy MA into AC
01 0316                                         D[CONST (5200 / 100)] ROT[6] ALU[D+AC] DEST[AC] NORM $
01 0317                                                 ;Step over retrieval!
01 0318                                         D[MEM] DEST[O_AC MA] NORM $
01 0319                                                 ;Restore AC.  Set MA
01 0320                                         DFRQ CYLEN[FIXM] $
01 0321                                                 ;Wait for memory
01 0322                                 DSKCB4: D[MEM] DEST[HOLD] NORM $
01 0323                                                 ;Copy from memory read buffer to memory write bu
ffer
01 0324                                         D[MA] ALU[D+1] DEST[O_AC MA STRT-WRT] NORM $
01 0325                                                 ;Increment fetch address and save in AC
01 0326                                                 ;Move old contents on AC into MA and start write
 cycle
01 0327                                         D[MA] ALU[D+1] DEST[O_AC MA] JUMP[DSKCB3] NORM $
01 0328                                                 ;Increment store address and save in AC
01 0329                                                 ;Move old contents on AC into MA and start read
cycle
01 0330                                 ;       ---
01 0331                                 DSKCB5: D[AMEM-ABS APR-DATASW] ROT[27.] MASK[9.] DEST[Q] LONG $
01 0332                                                 ;GET FILE # IN Q
01 0333                                         D[CONST 2] ALU[D#Q] COND[-ZERO] JUMP[DSKBTP] LONG $
01 0334                                                 ;MEMIMG.ETC
01 0335                                         D[CONST 72] DEST[AR] JUMP[UBLOC + 1] $
01 0336                                                 ;Jump to starting address
01 0337
01 0338                                 DSKBTP: D[NLIT BYTPTR] DEST[GBL-BPT] LONG $
01 0339                                                 ;Set byte pointer
01 0340                                         D[CONST 74] ALU[AC-D] DEST[AR] $
01 0341                                                 ;Determine word count
```

```
01 0342
01 0343
01 0344
01m0345
01m0345
01m0345 17162 010730170000000014003622255514560000
01m0345
01m0345
01m0345 17163 710731170000000014001622255514560000
01m0345
01m0345 17164 011700170037000014003622255514560000
01m0345
01m0345 17165 710731170000000014001622255514560000
01m0345
01m0345 17166 010731170000000014161622254114560000
01m0345
01m0345 17167 010731170000000014161622254114560000
01m0345
01m0345 17170 710731170000000001521622255514560000
01m0345
01m0345
01m0345 17171 010730170000060554161623656514160000
01m0345
01m0345 17172 710731170000363661525620255514560000
01m0345
 are
01m0345
01m0345 17173 010733170000060541521623655514160000
01m0345
rts at
01m0345
01m0345 17174 010731000000000140316202565125600 0
01m0345
01m0345
01m0345 17175 011621006000000014161620256514560000
01m0345
01m0345 17176 010732170000060554161503656514160000
01m0345
01m0345 17177 010503170000060541525623655514160000
01m0345
01m0345 17200 710732170000060554161623654515160000
01m0345
01m0345 17201 010731170000060554161623654112160000
01m0345
01m0345 17202 010731170000060554161503654515160000
01m0345
ffer
01m0345 17203 711702170000060554161643656514160000
01m0345
01m0345
 cycle
01m0345 17204 711702170000363714161620256514560000
01m0345
01m0345
```

```
        D[AR] ROT[2] DEST[GBL-BCT] JUMP[UMERGE] LONG $
                ;Set byte count
                ;Re-enter Foonly's bootstrap
];[
;Interface to F4 Console Computer
UBOOTD: D[CONST 1] DEST[Q] PUSHJ[DSKRCL] NORM $
                ;Do recalibrate to be safe
                ;Read MFD starting at track 1
        D[CONST 0] DEST[MA] PUSHJ[DSKBTO] NORM $
                ;Read starting at word 0
        D[1,,1] ALU[D+1] DEST[Q] PUSHJ[SRCMFN] $
                ;Search [1,1] for [1,2]
        D[CONST 0] DEST[MA] PUSHJ[DSKBTO] NORM $
                ;Read retrieval and UFD
        PUSHJ[SWAITS] NORM $
                ;Set file name
        PUSHJ[SRCXXX] NORM $
                ;Search for appropriate extension
DSKCB1: D[CONST 50] ROT[6] DEST[MA] PUSHJ[DSKBTO] NORM $
                ;Start reading into zero
                ;Read retrieval and file
        D[MA] DEST[Q] NORM $
                ;Save number of words read in Q for easy testing
        D[CONST (5200 / 100)] ROT[6] DEST[MA] NORM jump[. + 1] $
                ;First place in memory to read. (First 200 words

 retrieval)

        D[CONST 50] ROT[6] DEST[AC] NORM $
                ;First place in memory to write. (.DMP files sta

 JOBSAV)
DSKCB3: DFRQ D[MA] MASK[9 + 3] COND[-OBUS=0] JUMP[DSKCB4] C550 $
                ;Are we on a new track?  If not, we're OK
                ;Wait for fetch to complete
        D[MA] ALU[D-Q] COND[-OBUS<0] JUMP[DSKCB5] C550 $
                ;Yes, check for end-of-transfer
        D[MA] DEST[O_AC HOLD] $
                ;Save AC, copy MA into AC
        D[CONST (5200 / 100)] ROT[6] ALU[D+AC] DEST[AC] NORM $
                ;Step over retrieval!
        D[MEM] DEST[O_AC MA] NORM $
                ;Restore AC.  Set MA
        DFRQ CYLEN[FIXM] $
                ;Wait for memory
DSKCB4: D[MEM] DEST[HOLD] NORM $
                ;Copy from memory read buffer to memory write bu

        D[MA] ALU[D+1] DEST[O_AC MA STRT-WRT] NORM $
                ;Increment fetch address and save in AC
                ;Move old contents on AC into MA and start write

        D[MA] ALU[D+1] DEST[O_AC MA] JUMP[DSKCB3] NORM $
                ;Increment store address and save in AC
                ;Move old contents on AC into MA and start read
```

```
cycle
01m0345
01m0345 17205 01073017000000034662362367011416000
01m0345
01m0345 17206 01066100000000001400562025551456000
01m0345
01m0345 17207 01073137000036003416562025551456000
01m0345
01m0345
01m0345 17210 01073117110200000017134365511416000
01m0345
01m0345 17211 01151137000006055417162365551416000
01m0345
01m0345 17212 01073117000036040056132025411456000
01m0345
01m0345
01 0345
01 0346
01 0347
```

```
;       ---
DSKCB5: D[AMEM-ABS APR-DATASW] ROT[27.] MASK[9.] DEST[Q] LONG $
                ;GET FILE # IN Q
        D[CONST 2] ALU[D#Q] COND[-ZERO] JUMP[DSKBTP] LONG $
                ;MEMIMG.ETC
        D[CONST 72] DEST[AR] JUMP[UBLOC + 1] $
                ;Jump to starting address

DSKBTP: D[NLIT BYTPTR] DEST[GBL-BPT] LONG $
                ;Set byte pointer
        D[CONST 74] ALU[AC-D] DEST[AR] $
                ;Determine word count
        D[AR] ROT[2] DEST[GBL-BCT] JUMP[UMERGE] LONG $
                ;Set byte count
                ;Re-enter Foonly's bootstrap
].REPEAT F4PROM

.repeat 0 [
```

```
01 0348                                         ;Read N contiguous blocks
01 0349                                         DSJBT3: D[CONST 0] ROT[35. - 26.] DEST[AR] PUSHJ[DSK-OP] NORM $
01 0350                                                 ;Read a block
01 0351                                                 PUSHJ[DSK-WT] NORM $
01 0352                                                 ;Wait for completion
01 0353                                                 ALU[Q+1] DEST[AR] NORM $
01 0354                                                 ;Advance sector counter. Save result in AR for n
ow
01 0355                                                 D[MA] DEST[Q] NORM $
01 0356                                                 ;Setup to advance MA
01 0357                                                 D[CONST 1] ROT[9] ALU[D+Q] DEST[MA] NORM $
01 0358                                                 ;Advance by one page (=512 words)
01 0359                                                 D[AR] MASK[8] DEST[Q] $
01 0360                                                 ;Mask off sector part
01 0361                                                 D[CONST NSECTORS] ALU[Q-D] COND[OBUS(0] JUMP[DSJBT4] C55
0 $
01 0362                                                 ;Check for end-of-track
01 0363                                                 D[AR] ALU[D-Q] DEST[Q] NORM $
01 0364                                                 ;Set to sector zero
01 0365                                                 D[CONST 1] ROT[35. - 27.] ALU[D+Q] DEST[Q AR] NORM $
01 0366                                                 ;Advance head part.
01 0367                                                 ;*** Don't worry about head overflow for now
01 0368                                         ;       \ /
01 0369                                         DSJBT4: D[AR] DEST[Q] NORM $
01 0370                                                 ;Put disk address back into Q
01 0371                                                 D[IR] MASK[22] ALU[D-1] DEST[IR-ADR] COND[-OBUS(0] JUMP[
DSJBT3]
01 0371                                          C550 $
01 0372                                                 ;Check for end-of-transfer
01 0373                                                 POPJ NORM $
01 0374                                         ;       ---
01 0375                                         ];repeat 0
01 0376
01 0377                                         ;Read from WAITS disk file into memory.  No attempt is made to a
void
01 0377                                          reading
01 0378                                         ;retrieval part of track.  Q contains track address of first tra
ck of
01 0378                                          file.
01 0379                                         ;Also, turns off map.
01 0380 17213 010731170040060554002463655514170000 DSKBT0: D[CONST 1] DEST[DEV-ADR] NORM $
01 0381                                                 ;Select MAP
01 0382 17214 010241170031060554161523654114160000     ALU[0] DEST[IOD] SPEC[IOB-OUT] SHORT $
01 0383                                                 ;Turn off all random enablings
01 0384 17215 010731170240060554020463655514170000     MAPF[2] CYLEN[IOB-OUT] D[CONST DSKUDEV] DEST[DEV-ADR] $
01 0385                                                 ;Reselect disk.
01 0386 17216 010731170000000014161622254114560000 DSKBT1: PUSHJ[DSKCVA] NORM $
01 0387                                                 ;Convert to local disk address
01 0388 17217 010731170000000014161622254114560000 DSKBT2: PUSHJ[DSK-WT] NORM $
01 0389                                                 ;Make sure disk is ready (not needed for first b
lock)
01 0390 17220 010731370000000014001622255514560000     D[CONST 0] DEST[AR] PUSHJ[DSK-OP] NORM $
```

```
01 0391
01 0392 17221 01073100200000000764562225651456000
NORM $
01 0393
nto MA 0

01 0393
01 0394 17222 01120137000006055416162365411416000
ALU[Q+1] DEST[AR] NORM $
01 0395
ow
01 0396 17223 01073017000006055416162365651416000
01 0397
01 0398 17224 71060117000006054220362365551416000
01 0399
01 0400 17225 01073017000006055402162365411416000
01 0401
01 0402 17226 01161100600000001402162025551456000
50 $
01 0403
not.
01 0404 17227 01073017000036437416162025411456000
01 0405
01 0406 17230 01073137000006055416162365651416000
01 0407
01 0408
01 0409 17231 01073000200000000600362025651456000
01 0410
ock
01 0411 17232 01073017000006055404562365551416000
01 0412
01 0413 17233 01162117000006055404444365551417000
01 0414
01 0415 17234 01073017000006054561362365651416000
 $
01 0416
01 0417 17235 71060117000006055404162365551416000
01 0418
01 0419 17236 01073117000006055416162365411216000
01 0420
01 0421 17237 01073000200000001204562025451556000
] C550 $

01 0421
01 0422
        ;Get disk address
01 0423 17240 71073117000036435416162025411456000
01 0424
01 0425 17241 71073117000000001416162025411456000
01 0426
01 0427
01 0428
01 0429
01 0430
```

```
                ;Read block from disk
        D[MA] ROT[36. - 5] MASK[18.] COND[OBUS=0] PUSHJ[DSK-WT]

                ;Wait for completion here if reading retrieval i




                ;Advance sector counter. Save result in AR for n

        D[MA] DEST[Q] NORM $
                ;Setup to advance MA
        D[CONST 1] ROT[9] ALU[D+Q] DEST[MA] NORM $
                ;Advance by one page (=512 words)
        D[AR] MASK[8] DEST[Q] $
                ;Mask off sector part
        D[CONST NSECTORS] ALU[Q-D] COND[-OBUS<0] JUMP[DSKBT3] C5

                ;Check for end-of-track.  Read rest of track if

        D[AR] DEST[Q] JUMP[DSKBT2] NORM $
                ;Restore Q
DSKBT3: D[MA] DEST[AR] NORM $
                ;Same MA for now
        D[MA] ROT[36. - (9 + LOG2NSEC)] MASK[1] DEST[Q]
                        COND[OBUS=0] JUMP[DSKBT4] C550 $
                ;Decide which half to use for pointer to next bl

        D[CONST 18.] DEST[Q] NORM $
                ;Use right half
DSKBT4: D[CONST 18.] ALU[D-Q] DEST[ROTR] NORM $
                ;Compute proper half
        D[MA] ROT[36. - (9 + LOG2NSEC + 1)] MASK[5] DEST[Q] NORM

                ;Get address within retrieval
        D[CONST 20] ALU[D+Q] DEST[MA] NORM $
                ;Fetch pointer from retrieval
        DFRQ CYLEN[FIXM] $
                ;Wait for memory
        D[MEM] ROT[R] MASK[18.] DEST[Q] COND[OBUS=0] JUMP[DSKBT5




        D[AR] DEST[MA] JUMP[DSKBT1] NORM $
                ;Restore MA
DSKBT5: D[AR] DEST[MA] JUMP[DSK-WT] NORM $
                ;Restore MA and wait for disk to finish.

.REPEAT F4PROM [
SRCXXX: D[AMEM-ABS APR-DATASW] ROT[27.] MASK[2] DEST[Q] LONG $
                ;GET FILE # IN Q
```

```
01 0431
01 0432
01 0433
01 0434
01 0435
01 0436
01 0437
01m0438
01m0438 17242 010730170000000346605623670114160000
01m0438
01m0438 17243 010601170000000000001503655114160000
01m0438 17244 010731170000000000160007254515560000
01m0438 17245 010731170000000014161620254114560000
01m0438 17246 010731170000000014161620254114560000
01m0438 17247 010731170000000014161620254114560000
01m0438 17250 010731170000000014161620254114560000
01m0438
01 0438
01 0439
01 0440
01 0441
01 0442
01 0443
01 0444
01 0445
01 0446
01 0447
```

```
        D[LABEL SRCDSP] ALU[D+Q] DEST[HOLD] $
        D[MEM] ROT[MUA-ROT] ODISP LONG $
SRCDSP: JUMP[SRCUCD] $          ;Microcode
        JUMP[SRCUDG] $          ;Microdiag
        JUMP[SRCCCL] $          ;CCL
        JUMP[SRCDMP] $          ;System

];[
SRCXXX: D[AMEM-ABS APR-DATASW] ROT[27.] MASK[2] DEST[Q] LONG $
                ;GET FILE # IN Q
        D[LABEL SRCDSP] ALU[D+Q] DEST[HOLD] $
        D[MEM] ROT[MUA-ROT] ODISP LONG $
SRCDSP: JUMP[SRCUCD] $          ;Microcode
        JUMP[SRCUDG] $          ;Microdiag
        JUMP[SRCCCL] $          ;CCL
        JUMP[SRCDMP] $          ;System

].REPEAT F4PROM

.REPEAT F4SW [
SRCUDG: D[AR] DEST[Q] $                                 ;WAITSX.OBJ
        D[CONST 70] ALU[DORQ] DEST[AR] JUMP[SRCUCD] $
SRCUCD: D[LIT 574252] DEST[IR-ADR] $                    ;.OBJ
        JUMP[DSKSRC] $

SRCCCL: D[LIT 434354] DEST[IR-ADR] POPJ $               ;.CCL
        JUMP[DSKSRC] $
```

```
01m0448                                                    ];[
01m0448 17251 010730170000060554161623654114160000        SRCUDG: D[AR] DEST[Q] $                          ;WAITSX.OBJ
01m0448 17252 010631370000000014161620255514560000               D[CONST 70] ALU[DORQ] DEST[AR] JUMP[SRCUCD] $
01m0448 17253 610731170000001370525443655114160000        SRCUCD: D[LIT 574252] DEST[IR-ADR] $             ;.OBJ
01m0448 17254 010731170000000014161620254114560000               JUMP[DSKSRC] $
01m0448
01m0448 17255 610731170000001070731444255114160000        SRCCCL: D[LIT 434354] DEST[IR-ADR] POPJ $        ;.CCL
01m0448 17256 010731170000000014161620254114560000               JUMP[DSKSRC] $
01 0448                                                    ].REPEAT F4SW
01 0449
01 0450                                                    .REPEAT 1 - F4PROM [
01 0451
01 0452                                                    ;Set filename to DSKDMP
01 0453                                                    SDSKDM: D[CONST 44] ROT[30.] DEST[Q] NORM $              ;D
01 0454                                                            D[CONST 63] ROT[24.] ALU[DORQ] DEST[Q] NORM $    ;S
01 0455                                                            D[CONST 53] ROT[18.] ALU[DORQ] DEST[Q] NORM $    ;K
01 0456                                                            D[CONST 44] ROT[12.] ALU[DORQ] DEST[Q] NORM $    ;D
01 0457                                                            D[CONST 55] ROT[6.] ALU[DORQ] DEST[Q] NORM $     ;M
01 0458                                                            D[CONST 60] ALU[DORQ] DEST[AR] POPJ NORM $       ;P
01 0459
01 0460                                                    ];.REPEAT 1 - F4PROM
01 0461
01 0462                                                    ;Set filename to WAITS
01 0463 17257 010730170000060547557623655514160000        SWAITS: D[CONST 67] ROT[30.] DEST[Q] NORM $              ;W
01 0464 17260 010630170000060546103623655514160000                D[CONST 41] ROT[24.] ALU[DORQ] DEST[Q] NORM $    ;A
01 0465 17261 010630170000060544523623655514160000                D[CONST 51] ROT[18.] ALU[DORQ] DEST[Q] NORM $    ;I
01 0466 17262 010630170000060543151623655514160000                D[CONST 64] ROT[12.] ALU[DORQ] DEST[Q] NORM $    ;T
01 0467 17263 010630170000060541547623655514160000                D[CONST 63] ROT[6.] ALU[DORQ] DEST[Q] NORM $     ;S
01 0468                                                    ;;;     D[CONST 00] ALU[DORQ] DEST[AR] POPJ NORM $       ;
01 0469 17264 010631370000060554051624255514160000                D[CONST 24] ALU[DORQ] DEST[AR] POPJ NORM $       ;4
01 0470
01 0471                                                    .REPEAT 1 - F4SW [
01 0472                                                    ;Search for .UCD file
01 0473                                                    SRCUCD: D[CONST 65] ROT[12.] DEST[Q] NORM $              ;U
01 0474                                                            D[CONST 43] ROT[6.] ALU[DORQ] DEST[Q] NORM $     ;C
01 0475                                                            D[CONST 44] ALU[DORQ] DEST[IR-ADR]               ;D
01 0476                                                                        JUMP[DSKSRC] NORM $
01 0477                                                    ];.REPEAT 1 - F4SW
01 0478
01 0479                                                    ;Search for .DMP file
01 0480 17265 010730170000060543111623655514160000        SRCDMP: D[CONST 44] ROT[12.] DEST[Q] NORM $              ;D
01 0481 17266 010630170000060541533623655514160000                D[CONST 55] ROT[6.] ALU[DORQ] DEST[Q] NORM $     ;M
01 0482 17267 610631170000060554141443655514160000                D[CONST 60] ALU[DORQ] DEST[IR-ADR] NORM $        ;P
01 0483 17270 010731170000000014161620254114560000                            JUMP[DSKSRC] NORM $
01 0484
01 0485                                                    ;Search for UFD in AR
01 0486 17271 010631370037060541003623655514160000        SRCMFN: D[1,,1] ROT[4] ALU[DORQ] DEST[AR] NORM $         ;[n,n]
01 0487 17272 010730170000060543153623655514160000        SRCMFD: D[CONST 65] ROT[12.] DEST[Q] NORM $              ;U
01 0488 17273 010630170000060541515623655514160000                D[CONST 46] ROT[6.] ALU[DORQ] DEST[Q] NORM $     ;F
01 0489 17274 610631170000060554111443655514160000                D[CONST 44] ALU[DORQ] DEST[IR-ADR] $             ;D
01 0490                                                    ;        \ /
01 0491                                                    ;Search UFD for XXXXXX.YYY, where AR contains XXXXXX and IR-ADR
```

```
contains
01 0491                                                          YYY
01 0492                                                 ;Return pointer to first block in Q
01 0493 17275 710731170000060541405623655514160000     DSKSRC: D[CONST 2] ROT[6] DEST[MA] NORM $
01 0494                                                                 ;Skip retrieval
01 0495 17276 010731170000060554161623654112160000             DFRQ CYLEN[FIXM] $
01 0496                                                                 ;Wait for memory
01 0497                                                 ;               \ /
01 0498 17277 010730170000060554161623654515160000     DSKSR2: D[MEM] DEST[Q] NORM $
01 0499                                                                 ;Save file name
01 0500 17300 711701170000060554161623656514160000             D[MA] ALU[D+1] DEST[MA] NORM $
01 0501                                                                 ;Increment to get extension
01 0502 17301 011621000000000014161620254112560000             DFRQ D[AR] ALU[D-Q] COND[-OBUS=0] JUMP[DSKSR3] C550 $
01 0503                                                                 ;If not the same file, skip this entry
01 0504 17302 010730170000060544445623654515160000             D[MEM] ROT[18.] MASK[18.] DEST[Q] $
01 0505                                                                 ;Get extension from left half
01 0506 17303 011621000000000014045620257514560000             D[IR] MASK[18.] ALU[D-Q] COND[-OBUS=0] JUMP[DSKSR3] C550
 $
01 0507                                                                 ;Skip this entry if not same extension
01 0508 17304 711701170000060554161623656514160000             D[MA] ALU[D+1] DEST[MA] $
01 0509                                                                 ;Skip date
01 0510 17305 711701170000060554161623656514160000             D[MA] ALU[D+1] DEST[MA] $
01 0511                                                                 ;Read pointer to file
01 0512 17306 030730170000060554161624254513160000             DF/WT D[MEM] DEST[Q] POPJ LONG $
01 0513                                                                 ;Done.  Return pointer to file in Q.
01 0514                                                 ;               ---
01 0515 17307 010720170000060554041623655514160000     DSKSR3: D[CONST 20] ALU[D-1] DEST[Q] NORM $
01 0516                                                                 ;Advance to next entry in table
01 0517 17310 710601170000060554161623656514160000             D[MA] ALU[D+Q] DEST[MA] NORM $
01 0518 17311 010731000000365774031620256512560000             DFRQ D[MA] MASK[9 + LOG2NSEC] COND[-OBUS=0] JUMP[DSKSR2]
 C550 $
01 0519                                                                 ;Check for end of track.
01 0520 17312 010721170000366254001620255514560000             ALU[-1] JUMP[.] $
01 0521                                                                 ;File not found.
01 0522
01 0523                                                 ;Read block in whose disk address is in Q, memory address in MA,
 and
01 0524                                                 ;command in AR.  Preserves everything except DEV-ADR and IOD.
01 0525                                                 ;Caution:  This routine ties up IOB (stops DMA) for about 6 usec
.
01 0526 17313 010731170040060554020463655514170000     DSK-OP: D[CONST DSKUDEV] DEST[DEV-ADR] NORM $
01 0527                                                                 ;Set device address for disk
01 0528 17314 010630170000060547217623655514160000             D[CONST BOOT-DRIVE] ROT[35. - 6.] ALU[DORQ] DEST[Q] $
01 0529                                                                 ;Include unit number (*** kludge ***)
01 0530 17315 010651170031060547003523655514160000             D[CONST 1] ROT[35. - 7] ALU[-D&Q] DEST[IOD] SPEC[IOB-OUT
] NORM $

01 0530
01 0531
        ;Put out disk address with select off
01 0532                                                         MAPF[DSK-LD-DA] CYLEN[IOB-OUT]
01 0533                                                                         D[CONST 1] ROT[35. - 7] ALU[DORQ] DEST[I
OD]
01 0534 17316 010631170631060547003523655514160000                             SPEC[IOB-OUT] $
```

```
01 0535
01 0536
01 0537 17317 010731170631060554161523656514160000
01 0538
01 0539
01 0540 17320 010731170531060554161523654114160000
01 0541
01 0542
01 0543 17321 010731170431060554003523655514160000
$
01 0544
01 0545 17322 010731170700060554161624254114160000
01 0546
01 0547
01 0548
01 0549
01 0550 17323 010731170010060540024003655514170000
01 0551
01 0552 17324 010731070000366514161620254114560000
```

```
                ;Put out disk address with select on
        MAPF[DSK-LD-DA] CYLEN[IOB-OUT]
                        D[MA] DEST[IOD] SPEC[IOB-OUT] $
                ;Put out memory address
        MAPF[DSK-LD-MA] CYLEN[IOB-OUT]
                        D[AR] DEST[IOD] SPEC[IOB-OUT] $
                ;Load command
        MAPF[DSK-LD-CMD] CYLEN[IOB-OUT]
                        D[CONST DSK-GO] DEST[IOD] SPEC[IOB-OUT]

                ;Start disk controller
        MAPF[DSK-LD-ECC] CYLEN[IOB-OUT] POPJ $
                ;Done

;Wait for disk to complete.  Clobbers only AR.
;MUST WAIT FOR DRIVE!!!
DSK-WT: D[CONST 10.] ROT[LLOAD-ROT] LLOAD NORM $
                ;Load loop counter to give DMA a chance
        LOOP[.] C500 $
```

```
01 0553
01 0554 17325 010731170030060554161623654114160000                     ;Wait N+1 microseconds
                                                                SPEC[IOB-IN] SHORT $
01 0555                                                                 ;Start reading status
01 0556 17326 010731370400060554161623657114160000             MAPF[DSK-RD-CMD] CYLEN[IOB-IN] D[IOD] DEST[AR] $
01 0557                                                                 ;Finish reading status
01 0558 17327 010731004000366574161620254114560000             D[AR] COND[OBUS(0] JUMP[.] NORM $
01 0559                                                                 ;Select error. Display error status on OBUS
01 0560 17330 010731006000366462161620254114560000             D[AR] ROT[8.] COND[-OBUS(0] JUMP[DSK-WT] $
01 0561                                                                 ;Wait some more for disk controller
01 0562 17331 010731004000366460561620254114560000             D[AR] ROT[2.] COND[OBUS(0] JUMP[DSK-WT] $
01 0563                                                                 ;Wait some more for disk to go ready
01 0564 17332 010731006000060547161624254114160000             D[AR] ROT[28.] COND[-OBUS(0] POPJ C550 $
01 0565                                                                 ;Done if no error
01 0566 17333 010731170000366674161620254114560000             D[AR] JUMP[.] NORM $
01 0567                                                                 ;Display error status on OBUS
01 0568
01 0569                                                 ;Recalibrate disk (reset)
01 0570                                                 ;Assumes disk address in Q.  Clobbers AR, DEV-ADR
01 0571 17334 010731170040060554020463655514170000     DSKRCL: D[CONST DSKUDEV] DEST[DEV-ADR] NORM $
01 0572                                                                 ;Set device address for disk
01 0573 17335 010731170031060554005523655514160000             D[CONST DSK-INIT] DEST[IOD] SPEC[IOB-OUT] NORM $
01 0574                                                                 ;Initialize disk controller
01 0575 17336 010731170730060554161623654114160000     DSKRC2: MAPF[DSK-LD-ECC] CYLEN[IOB-OUT] SPEC[IOB-IN] $
01 0576                                                                 ;Finish send command
01 0577                                                                 ;Start reading status
01 0578 17337 010731370400060554161623657114160000             MAPF[DSK-RD-CMD] CYLEN[IOB-IN] D[IOD] DEST[AR] $
01 0579                                                                 ;Finish reading status
01 0580 17340 010731006000366742161620254114560000             D[AR] ROT[8.] COND[-OBUS(0] JUMP[DSKRC2] $
01 0581                                                                 ;If not ready, wait some more
01 0582 17341 010732370000060544207623655514160000             D[CONST 3] ROT[35. - 18.] DEST[O_AC AR] NORM $
01 0583                                                                 ;Setup for recalibrate (recalibrate + fault clea
r)
01 0584                                                                 ;Save AC
01 0585 17342 010533170000060542211623655514160000             D[CONST 4] ROT[35. - 26.] ALU[DORAC] DEST[AC] $
01 0586                                                                 ;Finish constructing recalibrate command
01 0587 17343 010732370000366274161622254114560000             D[AR] DEST[O_AC] DEST[AR] PUSHJ[DSK-OP] $
01 0588                                                                 ;Swap AR and AC, then do recalibrate
01 0589 17344 010650170000366467217620255514560000             D[CONST BOOT-DRIVE] ROT[35. - 6.] ALU[-D&Q] DEST[Q] JUMP
[DSK-WT]                                                        NORM $
01 0589                                                                 ;*** Undo drive number kludge
01 0590                                                                 ;Wait for completion
01 0591
01 0592                                                 ;Convert WAITS disk address into F2 disk address (unit BOOT-DRIV
01 0593                                                 E only)
01 0594                                                 ;Slow but simple divide by NHEADS.  Takes track address in Q
01 0595                                                 ;Return disk address in Q and MEM
01 0596 17345 010731170000060554001503655514160000     DSKCVA: D[CONST 0] ALU[D] DEST[HOLD] NORM $
01 0597                                                                 ;Result starts being zero
01 0598 17346 011610004000000014047620255514560000     DSKCV1: D[CONST NHEADS] ALU[Q-D] DEST[Q] COND[OBUS(0] JUMP[DSKCV
2] C550 $
01 0598
```

```
01 0599                                                                                                  ;Trial subtraction.
01 0600 17347 011701170000367154161500254515560000                                                      D[MEM] ALU[D+1] DEST[HOLD] JUMP[DSKCV1] NORM $
01 0601                                                                                                  ;Count cylinders
01 0602 17350 010600170000060554047623655514160000                                                  DSKCV2: D[CONST NHEADS] ALU[D+Q] DEST[Q] NORM $
01 0603                                                                                                  ;Restoring add
01 0604                                                                                              .repeat cdc160kludge [;***
01 0605                                                                                                      d[mem] rot[1] dest[hold] norm $
01 0606                                                                                                      d[const 10.] alu[q-d] cond[obus(0] jump[dskcv3] $
01 0607                                                                                                      d[const 10.] alu[q-d] dest[q] $
01 0608                                                                                                      d[mem] alu[d+1] dest[hold] $
01 0609                                                                                              dskcv3:
01m0610                                                                                              ];[;***
01m0610 17351 010731170000060540361503654515160000                                                          d[mem] rot[1] dest[hold] norm $
01m0610 17352 011611004000000014025620255514560000                                                          d[const 10.] alu[q-d] cond[obus(0] jump[dskcv3] $
01m0610 17353 011610170000060554025623655514160000                                                          d[const 10.] alu[q-d] dest[q] $
01m0610 17354 011701170000060554161503654515160000                                                          d[mem] alu[d+1] dest[hold] $
01m0610                                                                                              dskcv3:
01 0610                                                                                              ].repeat cdc160kludge
01 0611 17355 010601170000060542161503654515160000                                                          D[MEM] ROT[27. - 19.] ALU[D+Q] DEST[HOLD] NORM $
01 0612                                                                                                  ;Add head number to cylinder number
01 0613 17356 010730170000060542161504254515160000                                                          D[MEM] ROT[35. - 27.] DEST[Q HOLD] NORM POPJ $
01 0614                                                                                                  ;Align, save and return
01 0615
01 0616                                                                                              ;Clear 256K of memory.  Must not clobber Q (!)
01 0617 17357 710241170000060554161623654114160000                                                  CORCLR: ALU[0] DEST[MA] NORM $
01 0618 17360 010241170000060554161563654114160000                                                          ALU[0] DEST[MEMSTO] NORM $
01 0619 17361 711701170000060554161623656514160000                                                          D[MA] ALU[D+1] DEST[MA] NORM $
01 0620 17362 010731000000367414045620256514560000                                                          D[MA] MASK[18.] COND[-OBUS=0] JUMP[. - 2] C600 $
01 0621 17363 010731170000060554161624254114160000                                                          POPJ NORM $
01 0622
01 0623                                                                                              .REPEAT 1 - PROM [
01 0624                                                                                              .REPEAT 1 - F4SW [
01 0625
01 0626                                                                                              :7775
01 0627                                                                                              DSKSBT: D[CONST ((DSKMBS \ 10000) / 200)] ROT[7] DEST[Q] JUMP [.
 + 1] $
01 0628                                                                                                      D[CONST (XUCODE * 10) + 2] ROT[9.] ALU[D+1] C600 SDISP $
01 0629                                                                                                      ;This takes us to loc. 2001 and pre-sets the 100
00 bit.
01 0630                                                                                              DSKBT:  D[CONST ((HDSKBT \ 10000) / 200)] ROT[7] DEST[Q] JUMP[.
- 1] $
01 0631                                                                                                      ;Jump into high part of umemory, using 12-bit ad
dr. in Q

01 0631
01 0632                                                                                              :2001
;Just in case it doesn't exist
01 0633                                                                                                      D[CONST XUCODE] ROT[12.] ALU[DORQ] SDISP $
01 0634                                                                                                        ;This finally takes us to loc. 10000+[Q]
01 0635
01 0636                                                                                              ;Here's another entry point
01 0637                                                                                              :17777
01 0638                                                                                                      D[CONST (HDSKBT / 400)] ROT[8] SDISP $
01 0639
```

```
01 0640
01 0641
01 0642
01 0643
01 0644
```

```
:0
        JUMP[DSKBT] $           ;For loading from tape (sigh...)

];.REPEAT 1 - F4SW
];.REPEAT 1 - PROM
```

```
09 0491                                                 ].REPEAT WAITS
09 0492
09 0493                                                 .REPEAT 1 - WAITS [
09 0494                                                 ;SUCK IN NEXT DATA PAGE, INIT BYTE COUNT
09 0495
09 0496                                                 DSKIO:  D[DSK-AOB] COND[-NEGATIVE] JUMP[NOMORE] LONG $
09 0497                                                                 ;IF PAGE COUNTER AOBJN WENT POSITIVE, RAN OUT OF
 PAGES
09 0498                                                         D[DSK-AOB] MASK[18.] DEST[Q] LONG $
09 0499                                                                 ;GET PAGE # IN Q
09 0500                                                         D[CONST XBPAG] ROT[9.] ALU[D+Q] DEST[MA] LONG $
09 0501                                                                 ;LOAD MA WITH ADR OF XB WORD CONTAINING NEXT DIS
K ADR
09 0502                                                         DF/WT D[MEM] DEST[Q] COND[ZERO] JUMP[NOTSEQ] LONG $
09 0503                                                                 ;GET SOFT ADR IN DSK-SDA BUT JUMP IF IT IS ZERO
09 0504                                                         ALU[Q] DEST[DSK-SDA] PUSHJ[CVDSK] LONG $
09 0505                                                                 ;CONVERT TO HARD DA IN DSK-HDA
09 0506                                                         D[CONST DATPAG] ROT[9.] DEST[DSK-MEM] PUSHJ[RDSK] LONG $
09 0507                                                                 ;DSK-HDA SETUP, DSK-MEM GET MEM ADR, READ DATA P
G
09 0508                                                         D[DSK-AOB] DEST[Q] LONG $
09 0509                                                                 ;GET AOBJN PTR IN Q
09 0510                                                         D[1,,1] ALU[D+Q] DEST[DSK-AOB] LONG $
09 0511                                                                 ;UPDATE AOBJN PTR
09 0512                                                         D[LIT 4. * 512.] DEST[GBL-BCT] POPJ LONG $
09 0513                                                                 ;INIT # BYTES LEFT TO A FULL PAGE WORTH
09 0514                                                                 ;RETURN
09 0515
09 0516                                                 ;HERE IF EXPECTED A PAGE PTR FROM XB BUT FOUND A ZERO INSTEAD
09 0517                                                 ;MEANING FILE IS NOT SEQUENTIAL UP TO THE TERMINATING FLAG ADR
09 0518
09 0519                                                 NOTSEQ: JUMP[XUBOOT] LONG $
09 0520
09 0521                                                 ;HERE IF AOBJN PTR WHICH SCANS XB WENT NON-NEGATIVE
09 0522                                                 ;MEANING WE SAW ALL PAGES THERE ARE TO SEE
09 0523
09 0524                                                 NOMORE: JUMP[XUBOOT] LONG $
09 0525
09 0526                                                 ];.REPEAT 1 - WAITS
09 0527
```

```
10 0528
10 0529
10 0530
10 0531
10 0532
10 0533
10 0534
10 0535
10 0536
10 0537
10 0537
10 0538
ONG $
10 0539
10 0540
10 0541
10 0542
10 0543
10 0544
10 0544
10 0545
10 0546
10 0547
10 0548
10 0549
10 0550
10 0551
10 0552
10 0553
ONG $
10 0554
10 0555
10 0556
10 0557
10 0558
10 0559
10 0560
10 0561
10 0562
10 0563
10 0564
10 0565
10 0566
10 0567
10 0568
10 0569
 IN Q
10 0570
10 0571
UALT
10 0572
10 0573
```

```
.REPEAT 1 - WAITS [

;CONVERT SOFT DISK ADR IN DSK-SDA TO HDWR ADR IN DSK-HDA AND AR
;DESTROYS DSK-SDA AND MOST EVERYTHING ELSE

CVDSK:   D[CONST HOMPAG] ROT[9.] DEST[Q] LONG $
         D[CONST HOMHW3] ALU[D+Q] DEST[MA] LONG $
                  ;LOAD MA WITH LOC OF
 (NTKUN*NSURFS*NSECS),,(NSURFS*NSECS)
         DF/WT D[MEM] ROT[18.] MASK[18.] DEST[AR] PUSHJ[DSKDIV] L

                  ;READ LH, DO DSK-SDA / AR => AR,DSK-SDA
         D[AR] ROT[8. + 8. + 13.] DEST[DSK-HDA] LONG $
                  ;WRITE PACK TO DSK-HDA POSITIONED
         D[CONST HOMPAG] ROT[9.] DEST[Q] LONG $
         D[CONST HOMHW3] ALU[D+Q] DEST[MA] LONG $
                  ;LOAD MA WITH LOC OF
 (NTKUN*NSURFS*NSECS),,(NSURFS*NSECS)
         DF/WT D[MEM] MASK[18.] DEST[AR] PUSHJ[DSKDIV] LONG $
                  ;READ RH, DO DSK-SDA / AR => AR,DSK-SDA
         D[DSK-HDA] DEST[Q] LONG $
         D[AR] ROT[8. + 8.] ALU[DORQ] DEST[DSK-HDA] LONG $
                  ;IOR IN CYL FIELD TO HDWR DA
         D[CONST HOMPAG] ROT[9.] DEST[Q] LONG $
         D[CONST HOMHW2] ALU[D+Q] DEST[MA] LONG $
                  ;LOAD MA WITH LOC OF  (NSECS),,(NWSEC)
         DF/WT D[MEM] ROT[18.] MASK[18.] DEST[AR] PUSHJ[DSKDIV] L

                  ;READ LH, DO DSK-SDA / AR => AR,DSK-SDA
         D[DSK-HDA] DEST[Q] LONG $
         D[AR] ROT[8.] ALU[DORQ] DEST[Q] LONG $
                  ;IOR IN SURF FIELD TO HDWR DA
         D[DSK-SDA] ALU[DORQ] DEST[AR] LONG $
                  ;IOR IN SECTOR FIELD TO HDWR DA
         D[AR] DEST[DSK-HDA] LONG POPJ $
                  ;PUT IN DSK-HDA AND POPJ


;DO DSK-SDA / AR => AR,DSK-SDA  (AR/ QUOTIENT, DSK-SDA/ RMDR)

DSKDIV:  D[AR] COND[ZERO] JUMP[XUBOOT] LONG $
                  ;CATCH NO DIVIDE CASE
         D[DSK-SDA] MASK[20.] DEST[Q] LONG $
                  ;LO DIVIDEND NEVER BIGGER THAN DISK ADR MAX, GET

         ALU[0] DEST[IR-ALL] LONG $
                  ;USE AC 0 WHEN REFERRING TO AC VIA ACSEL[AC] DEF

         ALU[0] DEST[AC] JUMP[DSKDV0] LONG $
                  ;CLEAR HI DIVIDEND
```

```
10 0574
10 0575
10 0576
10 0577
10 0578
10 0579
10 0580
10 0581
10 0582
10 0583
+ 2] LONG
10 0583
10 0584
+ 1] LONG
10 0584
10 0585
10 0586
10 0587
10 0588
10 0589
10 0590
]
10 0590
10 0591
10 0592
10 0593
10 0594
10 0595
10 0596
10 0597
10 0598
10 0599
10 0600
10 0601
10 0602
```

```
                ;DO AC,Q / AR

        .EVEN

;COPIED SOMEWHAT FROM F4INST

DSKDV0:

.REPEAT 36. [
        D[AR] ALU[AC-D] DEST[D6] MASK[73] COND[SIGNON] LBJUMP[.

 $
        D[AR] ALU[AC+D] DEST[D6] MASK[73] COND[SIGNON] LBJUMP[.

 $
            ];REPEAT 36

        D[AR] ALU[AC-D] DEST[D6] MASK[73] JUMP[DSKDV1] LONG $
        D[AR] ALU[AC+D] DEST[D6] MASK[73] LONG $
DSKDV1: ALU[SH-AC] DEST[D5] MASK[0] LONG $
        D[CONST 1] ROT[35.] ALU[D#AC] DEST[DSK-SDA] COND[SIGNOFF

 JUMP[DSKDV2] LONG $
                ;ADJUST REM SIGN, CHECK IT, PUT IT IN DSK-SDA
        D[AR] ALU[D+AC] DEST[DSK-SDA] LONG $
                ;ADJUST REM., PUT IT IN DSK-SDA
DSKDV2: ALU[Q] DEST[AR] LONG POPJ $
                ;GET QUOTIENT IN AR
                ;AR/ QUOTIENT, DSK-SDA/ RMDR

];.REPEAT 1 - WAITS
```

```
11 0603
11 0604
11 0605                                                 .REPEAT 1 - WAITS [
11 0606
11 0607                                                 ;ACCEPTS 10 IN DEVADR
11 0608                                                 ;ACCEPTS HARD DA IN DSK-HDA
11 0609                                                 ;ACCEPTS MEM IN DSK-MEM
11 0610
11 0611                                                 RDSK:   ALU[0] DEST[DSK-RTY] LONG $
11 0612                                                                 ;RESET RETRY COUNTER, CLEAR RECAL'ING BIT 0, ERR
 RTN IS
11 0612                                                  RDSKE0
11 0613                                                 RDSK1:  ALU[0] DEST[IOD] SPEC[IOB-OUT] LONG $
11 0614                                                                 ;THIS IS TOP OF RETRY LOOP
11 0615                                                                 ;SET DSK CTRL COMMAND REGISTER TO 0 (DISABLES IN
TS).
11 0616                                                                 MAPF[4] D[CONST 2] DEST[IOD] SPEC[IOB-OUT] CYLEN[IOB-OUT
] $
11 0617                                                                 ;LE [2] CLR CTRLR
11 0618                                                                 MAPF[7] D[DSK-RTY] DEST[Q] CYLEN[IOB-OUT] $
11 0619                                                                 D[MASK 4.] ROT[30.] ALU[D&Q] DEST[DSK-RTY] LONG $
11 0620                                                                 ;CLEAR ALL BUT RETRY COUNTER
11 0621                                                                 START-OUT D[DSK-HDA] DEST[IOD Q] LONG $
11 0622                                                                 ;LA [DA] deselect (B7 off in legal disk adr)
11 0623                                                                 MAPF[6] LONG $
11 0624                                                                 START-OUT D[CONST 1] ROT[28.] ALU[DORQ] DEST[IOD] NORM $
11 0625                                                                 ;LA [1B7!DA] select and load DA
11 0626                                                 MAPF[6] SPEC[IOB-IN] CYLEN[IOB-OUT] $
11 0627
11 0628                                                 D[IOD] MAPF[0] DEST[Q AR] CYLEN[IOB-IN] $
11 0629                                                                 ;RC INTO Q AND AR
11 0630                                                 D[CONST 5] ROT[33.] ALU[D&Q] COND[-ZERO] JUMP[RDSKE0] LO
NG $
11 0631                                                                 ;HANDLE ERROR IF S.NRDY!S.SERR ON IN LH
11 0632                                                 D[CONST 4] ROT[15.] DEST[Q] LONG $
11 0633                                                 D[CONST 4] ROT[9.] ALU[DORQ] DEST[AR] PUSHJ[RDSKGO] LONG
 $
11 0634                                                                 ;SEND 404000 CLEAR FAULT COMMAND, GET STATUS BAC
K IN AR
11 0635
11 0636                                                 D[DSK-MEM] DEST[IOD] SPEC[IOB-OUT] LONG $
11 0637                                                                 ;LM [MEM ADR] LOAD MA
11 0638                                                 MAPF[5] SPEC[IOB-IN] CYLEN[IOB-OUT] $
11 0639                                                                 ;RC INTO AR
11 0640                                                 D[IOD] MAPF[0] DEST[AR] CYLEN[IOB-IN] $
11 0641                                                 D[AR] ROT[2.] COND[SIGNON] JUMP[RDSKE0] LONG $
11 0642                                                                 ;HANDLE ERROR IF S.NRDY ON
11 0643                                                 D[AR] ROT[28.] COND[SIGNON] JUMP[RDSKE0] LONG $
11 0644                                                                 ;HANDLE ERROR IF S.ANY ON
11 0645                                                 D[DSK-RTY] DEST[Q] LONG $
11 0646                                                 D[CONST 1] ROT[34.] ALU[DORQ] DEST[DSK-RTY] LONG $
11 0647                                                                 ;SAY WE WANT RDSKE1 ROUTINE ON ERROR IN RDSKGO
                                                        ALU[0] DEST[AR] PUSHJ[RDSKGO] LONG $
```

```
11 0648                                         ;LC [0] CMD[READ]
11 0649                                 D[AR] ROT[13.] COND[SIGNON] JUMP[RDSKEO] LONG $
11 0650                                         ;HANG IF S.IPE ON
11 0651                                 D[AR] ROT[28.] COND[SIGNON] JUMP[RDSKEO] LONG $
11 0652                                         ;HANG IF S.ANY ON
11 0653                                 POPJ LONG $
11 0654
11 0655                         ];.REPEAT 1 - WAITS
11 0656
```

```
12 0657
12 0658
12 0659                                         .REPEAT 1 - WAITS [
12 0660
12 0661                                         ;ACCEPTS CMD IN AR (STORES IT IN DSK-LCM TOO)
12 0662                                         ;RETURNS RC IN AR FOR SUCCESS
12 0663                                         ;RESYNCS STACK AND JUMPS TO RDSKE0 OR RDSKE1 ON TIMEOUT FAILURE
12 0664                                         ;B1 OF DSK-RTY DETERMINES WHICH ERROR ROUTINE JUMPED TO
12 0665
12 0666                                         RDSKG0: D[DSK-RTY] DEST[Q] LONG $
12 0667                                                 D[MASK 6.] ROT[30.] ALU[D&Q] DEST[DSK-RTY] LONG $
12 0668                                                         ;RESET TIMER PART OF DSK-RTY
12 0669                                                 D[CONST 2] DEST[IOD] SPEC[IOB-OUT] LONG $
12 0670                                                         ;LE [2] CLR CTRLR
12 0671                                                 MAPF[7] D[AR] DEST[IOD] SPEC[IOB-OUT] CYLEN[IOB-OUT] $
12 0672                                                         ;LC AR LOAD COMMAND
12 0673                                                 MAPF[4] D[AR] DEST[DSK-LCM] CYLEN[IOB-OUT] $
12 0674                                                         ;SAVE LAST CMD IN AMEM
12 0675                                                 D[CONST 1] DEST[IOD] SPEC[IOB-OUT] LONG $
12 0676                                                         ;LE [1] START CTRLR
12 0677                                         RDSKG1: MAPF[7] SPEC[IOB-IN] CYLEN[IOB-OUT] $
12 0678                                                 D[IOD] MAPF[0] ROT[31.] DEST[AR] COND[SIGNON] CYLEN[IOB-
IN] $
12 0679                                                         ;READ RC, JUMP IF S.NACT ON
12 0680                                                 D[DSK-RTY] DEST[Q] LONG JUMPLC[RDSKG2] $
12 0681                                                 ALU[Q+1] DEST[DSK-RTY] LONG $
12 0682                                                         ;BUMP TIMER
12 0683                                                 D[DSK-RTY] ROT[6.] COND[SIGNOFF] JUMP[RDSKG1] LONG $
12 0684                                                         ;LOOP TIL COUNT CARRIES INTO BIT 6 (LOTSA TIME)
12 0685                                                 D[AR] ROT[5.] DEST[AR] LONG$
12 0686                                                         ;ROTATE AR BACK TO ORIGINAL CONFIGURATION
12 0687                                                 D[DSK-RTY] ROT[1.] COND[SIGNOFF] MU-POP JUMP[RDSKE0] LON
G $
12 0688                                                         ;POP STACK, JUMP TO APPROPRIATE ERROR ROUTINE
12 0689                                                 MU-POP JUMP[RDSKE1] LONG $
12 0690
12 0691                                         RDSKG2: D[AR] ROT[5.] DEST[AR] POPJ LONG $
12 0692                                                 ;ROTATE AR ALL THE WAY AROUND TO RERUN ORIGINAL
STATUS
12 0692                                          BITS
12 0693
12 0694                                         ];.REPEAT 1 - WAITS
12 0695
```

```
13 0696
13 0697
13 0698                                                 .REPEAT 1 - WAITS [
13 0699                                                 .REPEAT 1 - RCVERR [
13 0700
13 0701                                                 ;GET HANGS WHEN NOT ASSEMBLING ERROR RECOVERY ROUTINES
13 0702
13 0703                                                 RDSKE0: JUMP[XUBOOT] LONG $
13 0704                                                 RDSKE1: JUMP[XUBOOT] LONG $
13 0705
13 0706                                                                 ];.REPEAT 1 - RCVERR
13 0707
13 0708
13 0709
13 0710                                                 ;OPTIONAL DISK ERROR RECOVERY ROUTINES
13 0711
13 0712                                                 .REPEAT RCVERR [
13 0713
13 0714                                                 ;GET TO RDSKE0 IF 0B1 IN DSK-RTY (INSISTS SEEK ERR WITHOUT SELEC
T ERR)
13 0715                                                 ;GET TO RDSKE1 IF 1B1 IN DSK-RTY
13 0716                                                 ;ACCEPTS STATUS IN AR
13 0717                                                 ;RECALS AND JUMPS TO RDSK1 TO RESTART IF ERROR COUNT DID NOT RUN
 OUT
13 0718
13 0719                                                 RDSKE0: D[AR] COND[SIGNON] JUMP[XUBOOT]  LONG $
13 0720                                                                 ;HANG ON S.SERR
13 0721                                                         D[AR] ROT[4.] COND[SIGNOFF] JUMP[XUBOOT] LONG $
13 0722                                                                 ;HANG IF NOT S.SKER
13 0723                                                 RDSKE1: D[DSK-RTY] DEST[Q] LONG $
13 0724                                                         D[CONST 1] ROT[34.] ALU[-D&Q] DEST[DSK-RTY] LONG $
13 0725                                                                 ;CLEAR FLAG TO GET RDSKE0 ERROR RTN NEXT
13 0726                                                         D[DSK-RTY] DEST[Q] COND[SIGNON] JUMP[XUBOOT] LONG $
13 0727                                                                 ;HANG IF WE ARE HERE WHILE RECAL'ING
13 0728                                                         D[CONST 1] ROT[35.] ALU[DORQ] DEST[DSK-RTY] LONG $
13 0729                                                                 ;TURN ON RECAL'ING BIT
13 0730                                                         D[CONST 14] ROT[15.] DEST[Q] LONG $
13 0731                                                         D[CONST 4] ROT[9.] ALU[DORQ] DEST[AR] PUSHJ[RDSKGO] LONG
 $
13 0732                                                                 ;SEND RECAL CMD 1404000, GET STATUS IN AR
13 0733                                                         D[AR] ROT[28.] COND[SIGNON] JUMP[XUBOOT] LONG $
13 0734                                                                 ;HANG IF S.ANY DURING RECAL
13 0735                                                         D[DSK-RTY] ROT[6.] ALU[D+1] DEST[AR] LONG $
13 0736                                                                 ;BUMP RETRY COUNT
13 0737                                                         D[AR] ROT[32.] COND[SIGNON] JUMP[XUBOOT] LONG $
13 0738                                                                 ;HANG IF CARRIED INTO BIT 2 (RETRY 8 TIMES)
13 0739                                                         D[AR] ROT[30.] DEST[DSK-RTY] JUMP[RDSK1] LONG $
13 0740                                                                 ;WRITE UPDATED COUNTER AND FRIENDS TO DSK-RTY AN
D RETRY
13 0741
13 0742                                                                 ];.REPEAT RCVERR
13 0743
```

```
13 0744
13 0745                                    ];.REPEAT 1 - WAITS
```

```
14 0746
14 0747
14 0748                                                 ;ENTER WITH # OF FILES TO SKIP IN TAP-FLN
14 0749                                                 ;READ A RECORD STARTING AT DATA PAGE AND SETUP # BYTES IN GBL-BC
T
14 0750
14 0751 17364 010730170000060554161623663514160000     TAPIO:  D[TAP-STA] DEST[Q] LONG $
14 0752 17365 010641000000000014021620255514560000             D[CONST 10] ALU[D&Q] COND[-ZERO] JUMP[TPNEWR] LONG $
14 0753                                                                 ;JUMP IF NOT 1ST TIME CALLING THIS
14 0754 17366 010631170000060554021163655514160000             D[CONST 10] ALU[DORQ] DEST[TAP-STA] LONG $
14 0755                                                                 ;INT FLAG SET SAY WE DID THIS 1ST TIME
14 0756 17367 010241170031060554161523654114160000             ALU[0] DEST[IOD] START-OUT LONG $
14 0757                                                                 ;Turn off "FORMATTER ENABLE"
14 0758 17370 010731170200000014161622254114560000             MAPF[TP.WF] PUSHJ[TAPERW] LONG $
14 0759                                                                 ;REWIND TAPE
14 0760 17371 010731170431060542203523655514160000     TAPIO1: MAPF[TP.WM] START-OUT D[CONST 1] ROT[35. - 26.] DEST[IOD
] C-OUT $
14 0760
14 0761                                                                 ;ENABLE THE FORMATTER.
14 0762 17372 010240170230060554161623654114160000             MAPF[TP.WF] START-IN ALU[0] DEST[Q] LONG $
14 0763                                                                 ;READ STATUS BITS.
14 0764 17373 010771370400060554161623657114160000             MAPF[TP.RS] D[IOD] ALU[NOTD] DEST[AR] LONG $
14 0765 17374 010731006000367623161620254114560000             D[AR] ROT[12.]  -OBUS<0 JUMP[TAPIO1] LONG $
14 0766                                                                 ;Wait until tape is at load point.
14 0767
14 0768                                                 ;TPNEWR:  START NEW RECORD
14 0769                                                 ;TPNEWF:  START NEW FILE
14 0770
14 0771                                                 TPNEWR:
14 0772 17375 010731004000361270761620263514560000     TPNEWF: D[TAP-STA] ROT[35.] COND[SIGNON] JUMP[XUBOOT] LONG $
14 0773                                                                 ;IF SAW EOF OF FILE OF INTEREST, LOSE
14 0774 17376 710731170000000012001623655114160000             D[LIT DATPAG * 1000] DEST[MA] LONG $
14 0775                                                                 ;INIT MA TO START OF DATA PAGE
14 0776 17377 610241170000060554161463654114160000             ALU[0] DEST[PC] LONG $
14 0777                                                                 ;COUNT # BYTES IN PC
14 0778 17400 010731170000000014161622254114560000             PUSHJ[KNYRGO] LONG $
14 0779                                                                 ;start read.
14 0780 17401 010731170000000014161622254114560000             PUSHJ[KNYWAIT] NORM $
14 0781                                                                 ;Wait for first byte.
14 0782 17402 010771170230060554161503657114160000             START-IN MAPF[TP.RC] D[IOD] ALU[NOTD] DEST[HOLD] C800 $
14 0783                                                                 ;Read 1st byte of new record.
14 0784                                                                 ;BUMP BYTE COUNTER
14 0785 17403 000731170105000014021500254515560000             PC-INC MAPF[1] D[MEM] MASK[8.] DEST[HOLD] JUMP[BYTLPA] C
550 $
14 0786                                                                 ;MAPF[1] clears byte ready
14 0787                                                                 ;Extract data byte from other status information
14 0788                                                                 ;ENTER BYTE READ LOOP
14 0789
14 0790                                                 ;All D[MEM]'s here reading from HOLD.
14 0791
14 0792 17404 711701170000000014161622256514560000     BYTLP:  D[MA] ALU[D+1] DEST[MA] PUSHJ[TAPAGN] NORM $
14 0793                                                                 ;BUMP MA, GET 1ST BYTE
```

```
14 0794 17405 0107301700000000071616222545155600
14 0795
14 0796 17406 0106301700000000051616222545155600
14 0797
14 0798 17407 0106301700000000031616222545155600
14 0799
14 0800 17410 0106311700003701011615602545155600
14 0801
14 0802
14 0803
14 0804
14 0805 17411 0107311700100605402720036555141700
14 0806
14 0807
14 0808 17412 0107311700300605541616236541141600
14 0809
14 0810 17413 0107310702000000141615002571145600
14 0811
14 0812
blems
14 0813
14 0814 17414 0107311700300000141616222541145600
14 0815
14 0816 17415 0107310060003612701616202545155600
14 0817
14 0818 17416 0107310040000000103616202545155600
14 0819
14 0820 17417 0107300020000000141616202631145600
14 0821
ROCESS.
14 0822
AIN
14 0822
14 0823 17420 0116111700033677340031402555145600
NG $
14 0824
14 0825
14 0826
14 0827
14 0828
14 0829 17421 0107310060303702665616202545155600
14 0830
 loop)
14 0831
have
14 0832
14 0833 17422 0107711702300605541615036571141600
14 0834
14 0835
14 0836 17423 0007311701050605540215042545151600
14 0837
14 0838
14 0839
14 0840
```

```
BYTLPA: D[MEM] ROT[34] DEST[Q] PUSHJ[TAPAGN] NORM $
                ;ENTER 1ST BYTE, GET 2ND BYTE
        D[MEM] ROT[24] ALU[DORQ] DEST[Q] PUSHJ[TAPAGN] NORM $
                ;ENTER 2ND BYTE, GET 3RD BYTE
        D[MEM] ROT[14] ALU[DORQ] DEST[Q] PUSHJ[TAPAGN] NORM $
                ;ENTER 3RD BYTE, GET 4TH BYTE
        D[MEM] ROT[4] ALU[DORQ] DEST[MEMSTO] JUMP[BYTLP] NORM $
                ;ENTER 4TH BYTE, IGNORE BITS 32:35 OF MEM WORD

;GET NEXT BYTE AS TAPE FLYS BY

TAPAGN: D[CONST 35] ROT[LLOAD-ROT + 1] DEST[LLOAD] NORM $
                ;Set loop counter to do timeout
                ;(TIMEOUT ABOUT 78 USEC)
        START-IN NORM $
                ;Ask tape for a byte and status thereof
TPWTB1: MAPF[TP.RC] D[IOD] DEST[HOLD] C800 LOOP[TPWTB2] $
                ;Read byte and status.  Byte comes complemented.
                ;Result is put in HOLD to avoid synchronizer pro

                ;Do timeout check and branch if still waiting
        START-IN PUSHJ[FMNBWT] LONG $
                ;Wait for FORMATTER IDLE SO CAN DETECT EOF TOO.
        D[MEM] ROT[32.] -OBUS<0 JUMP[XUBOOT] LONG $
                ;Check for error status. DIE IF ERROR
        D[MEM] ROT[33.] OBUS<0 JUMP[TAPEOR] LONG $
                ;If end-of-file not seen, ASSUME END-OF-RECORD.
        D[TAP-FLN] DEST[Q] COND[ZERO] JUMP[TAPEOF] LONG $
                ;IF HIT EOF AND THIS IS FILE OF INTEREST, THEN P

                ;ENDING CONDITION SHOULD BE TRMADR DETECTED IN M

 STUFF.
        MU-POP D[CONST 1] ALU[Q-D] DEST[TAP-FLN] JUMP[TPNEWF] LO

                ;RESYNC STACK
                ;DECREMENT # FILES TO SKIP, DO NEXT FILE

;PART OF BYTE READY LOOP

TPWTB2: START-IN D[MEM] ROT[26.] -OBUS<0 C550 JUMP[TPWTB1] $
                ;Check for byte ready (this is a two instruction

                ;Start getting byte and status again in case we

                ;  to loop
        START-IN MAPF[TP.RC] D[IOD] ALU[NOTD] DEST[HOLD] C800 $
                ;READ THE DATA AGAIN (NOW THAT IT'S STABLE !)
                ;BUMP BYTE COUNTER
        PC-INC MAPF[1] D[MEM] MASK[8.] DEST[HOLD] C550 POPJ $
                ;MAPF[1] clears byte ready
                ;Extract data byte from other status information

;HERE WHEN HIT EOR
```

```
14 0841                                                         ;SET BYTE COUNT AND RETURN, RECORD IN DATA PAGE
14 0842
14 0843 17424 0107311700030605541613236561141600 0     TAPEOR: MU-POP D[PC] DEST[GBL-BCT] LONG $
14 0844                                                                 ;RESYNC STACK
14 0845                                                                 ;STORE BYTE COUNT IN AMEM
14 0846 17425 0107310020000605541616242631141600 0             D[TAP-FLN] COND[ZERO] POPJ LONG$
14 0847                                                                 ;RET IF FILE OF INTEREST
14 0848 17426 0107311700003677341616202541145600 0             JUMP[TPNEWR] LONG$
14 0849                                                                 ;IGNORE REC IF NOT OF INTEREST
14 0850
14 0851                                                         ;HERE WHEN HIT EOF FOR FILE OF INTEREST.
14 0852                                                         ;SET FLAG REMEMBERING IT INCASE TURKEY TRIES TO READ PAST IT
14 0853
14 0854 17427 0107301700000605541616236635141600 0     TAPEOF: D[TAP-STA] DEST[Q] LONG $
14 0855                                                                 ;GET FLAGS
14 0856 17430 0106311700000605540031636555141600 0             D[CONST 1] ALU[DORQ] DEST[TAP-STA]  LONG $
14 0857                                                                 ;REMEMBER HIT EOF OF FILE OF INTERST.
14 0858                                                                 ;NOW TREAT AS NORMAL EOR
```

```
14 0859 17431 010731170003060554161323656114160000              MU-POP D[PC] DEST[GBL-BCT] LONG$
14 0860                                                                 ;STORE BTYE COUNT
14 0861 17432 010731170000060554161624254114160000              POPJ LONG$
14 0862                                                                 ;RET
14 0863
```

```
15 0864
15 0865
15 0866                                                 ;REWIND TAPE
15 0867
15 0868 17433 010731170000060554051503655551416000     TAPERW: D[CONST 24] DEST[HOLD] LONG $
                                                                ;RWD AND GO BITS -- KNYGOA WILL DELETE THE GO BI
15 0869
T !
                                                                START-OUT LONG $
15 0870 17434 010731170031060554161623654114160000             ;Give TP MR to clear mode, error status
15 0871                                                         MAPF[TP.MR] D[MEM] DEST[Q] C-OUT PUSHJ[KNYGOA] $
15 0872 17435 010730170500000014161622254515560000             ;Put command bits in Q and start formatter.
15 0873                                                         START-OUT D[CONST 1] ROT[35. - 26.] DEST[IOD] LONG $
15 0874 17436 010731170031060542203523655514160000             ;Clear all command bits except FORMATTER ENABLE.
15 0875                                                         MAPF[TP.WF] LONG POPJ $
15 0876 17437 010731170200060554161624254114160000
15 0877
15 0878
15 0879                                                 ;INITIALIZE DATA CHANNEL
15 0880
15 0881 17440 010731170031060554161623654114160000     DCINIT: START-OUT LONG $
15 0882 17441 010731170510060540020003655514170000             MAPF[TP.MR] D[CONST 8.] ROT[LLOAD-ROT] DEST[LLOAD] LONG
$
                                                                ;Give TP MR
15 0883                                                         START-OUT ALU[O] DEST[IOD] LONG $
15 0884 17442 010241170031060554161523654114160000             ;Load O into cntma (the COUNT and MA registers)
15 0885                                                         MAPF[TP.WMA] START-OUT D[MASK 12.] ROT[35. - 13.] DEST[I
15 0886 17443 010731170331060545431523647114160000     $
OD] LONG                                                                ;Load cntma again-- since COUNT is currently  O,
15 0887                                                 will
 this
15 0887                                                                 ; force BUF CNT to be O.  But, we put -1 into CO
15 0888
UNT this

15 0888
15 0889
        ; time, which, with BUF CNT = O, will make BUF CNT load
                                                                ; properly when STARTDC loads cntma.
15 0890
15 0891 17444 010731170331060554005523655514160000     MAPF[TP.WMA] START-OUT D[CONST 2] DEST[IOD] LONG $
15 0892                                                         ;Set MBUSY
15 0893 17445 010731070100371134161620254114560000     MAPF[TP.WC] C550 LOOP[.] $
15 0894                                                         ;Wait for a few usec.  This clears mem rq.
15 0895 17446 010241170031060554161523654114160000     START-OUT ALU[O] DEST[IOD] LONG $
15 0896                                                         ;Clr MBUSY
15 0897 17447 010241170100060554161044254114160000     MAPF[TP.WC] ALU[O] DEST[TAP-LWT] POPJ LONG $
15 0898                                                         ;Zero out copy of TP.WC contents.
15 0899
15 0900
```

```
16 0901
16 0902
16 0903                                                 ;Wait for first byte of read data (BOOTSTRAP mode only)
16 0904
16 0905 17450 010731170030060554161623654114160000     KNYWAIT:  START-IN LONG $
16 0906 17451 010731370130060543177623655514160000             MAPF[1] START-IN D[CONST 77] ROT[12.] DEST[AR] LONG $
16 0907                                                                 ;Clear read data ready flag
16 0908 17452 010731170200060554161503657114160000     KNYW1:    MAPF[TP.RC] D[IOD] DEST[HOLD] C600 $
16 0909 17453 010731004030060546561624254515160000             START-IN D[MEM] ROT[26.] OBUS<0 POPJ C550 $
16 0910                                                                 ;Return if READ DATA RDY is now on.
16 0911                                                                 ;START READ OF BYTE
16 0912 17454 010721200030371254161620254114560000             START-IN D[AR] ALU[D-1] DEST[AR] C550 -OBUS=0 JUMP[KNYW1
] $
16 0913                                                                 ;START READ OF STATUS, LOOP TILL TIME OUT
16 0914 17455 010731170000361274161620254114560000             JUMP[XUBOOT] LONG $
16 0915                                                                 ;TIMEOUT FOR 1ST BYTE, DIE
16 0916
16 0917
16 0918                                                 ; KNYRGO -- Called to start tape motion on reads.
16 0919
16 0920 17456 010731170000371014161622254114560000     KNYRGO:   PUSHJ[DCINIT] LONG $
16 0921                                                                 ;Init the data channel.
16 0922 17457 010731370000060540203623660114160000             D[TAP-DPM]  ROT[1] MASK[1] DEST[AR] LONG $
16 0923                                                                 ;Get the 32-bit mode flag.
16 0924 17460 010731170031060541561523654114160000             START-OUT D[AR] ROT[35. - 29.] DEST[IOD] LONG $
16 0925                                                                 ;Position it for the hardware.
16 0926 17461 010730170400000014001620255514560000             MAPF[TP.WM] D[CONST 0] DEST[Q]   C-OUT JUMP[KNYGOA] $
16 0927                                                                 ;Send command to formatter.
16 0928
16 0929
16 0930                                                 ;Send a tape-motion command to the formatter.  Call with command
 bits
16 0931                                                 ;  (except for FMTR ENABLE and GO) in Q.  Clobbers Q, HOLD, AR
16 0932
16 0933 17462 010731170031060542203523655514160000     KNYGOA:   START-OUT D[CONST 1] ROT[35. - 26.] DEST[IOD] LONG $
16 0934                                                                 ;Send the FORMATTER ENABLE bit.
16 0935 17463 010630170200000002203622255514560000             MAPF[TP.WF] D[CONST 1] ROT[35. - 26.] ALU[DORQ] DEST[Q]
 PUSHJ[FMNBWT] LONG $
16 0935                                                                 ;Add the FMTR ENBL bit to the command word.
16 0936                                                                 ;Wait for FORMATTER NOT BUSY.
16 0937                                                         START-OUT D[CONST 1] ROT[35. - 33.] ALU[D#Q] DEST[IOD] C
16 0938 17464 010661170031060540403523655514160000
-OUT $
16 0939                                                                 ;Set the GO bit to fmtr (except: on RWD, CLEAR t
he bit !)
16 0939
16 0940 17465 010651170231060540403523655514160000             MAPF[TP.WF] START-OUT D[CONST 1] ROT[35. - 33.] ALU[-D&Q
]
16 0940                                                  DEST[IOD] LONG $
16 0941                                                                 ;Send command word again, without GO bit.
16 0942 17466 010731170200060554161624254114160000             MAPF[TP.WF] LONG POPJ $
16 0943
16 0944
```

16 0945

```
17 0946
17 0947
17 0948                                                 ;Wait for formatter to be not busy.
17 0949                                                 ;Return tape status in MEM;  timeout in 164 msec.
17 0950                                                 ;Duration of loop should be 10 usec. (for TRCHECK).
17 0951
17 0952 17467 010731370030060543403623655514160000     FMNBWT: START-IN D[CONST 1] ROT[14.] DEST[AR] LONG $
17 0953 17470 010731170400060554161503657114160000     FMNBW1: MAPF[TP.RS] D[IOD] DEST[HOLD] C800 $
17 0954                                                                 ;GET STATUS BITS.
17 0955 17471 010721204000000014161620254114560000             D[AR] ALU[D-1] DEST[AR] C550 OBUS(0  JUMP[FMTHNG] $
17 0956 17472 010731170010060540034003655514170000             D[CONST 14.] ROT[LLOAD-ROT] DEST[LLOAD] C600 $
17 0957 17473 010731070000371674161620254114560000             C500 LOOP[.] $
17 0958                                                                 ;We execute this instr. 15. times, for a 7.5 use
c delay.

17 0958
17 0959 17474 010731006030371601761620254515560000
START-IN D[MEM] ROT[7] C550 -OBUS(0 JUMP[FMNBW1] $
17 0960                                                                 ;check for 'BUSY'
17 0961 17475 010731170000060554161024254114160000             D[AR] DEST[TAP-TIM] LONG POPJ $
17 0962                                                                 ;Save ending timeout count (for TRCHECK)
17 0963
17 0964
17 0965 17476 010241170031060554161523654114160000     FMTHNG: START-OUT ALU[0]  DEST[IOD] LONG $
17 0966                                                                 ;Turn off "FORMATTER ENABLE"
17 0967 17477 010731170200361274161620254114560000     MAPF[TP.WF] JUMP[XUBOOT] LONG $
17 0968                                                                 ;ABORT, FMTR HUNG
17 0969
17 0970
```

SLOE Aug 18, 1986 18:08:11 file DSK:F4UBOT.NEW -- of -- XBOOT

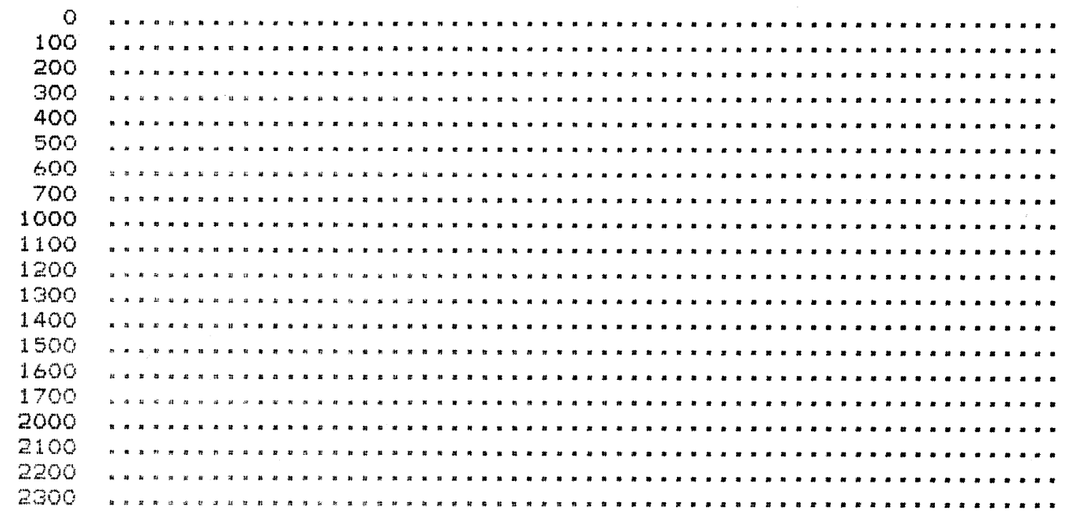
18 0971
18 0972
18 0973 list-labels

labels

0	38	0	.	14	792	17404	BYTLP	14	794	17405	BYTLPA				
1	617	17357	CORCLR	15	881	17440	DCINIT	1	526	17313	DSK-OP	1	550	17323	DSK-WT
1	1	17124	DSKBT-BEG	1	380	17213	DSKBTO	1	386	17216	DSKBT1	1	388	17217	DSKBT2
1	406	17230	DSKBT3	1	413	17233	DSKBT4	1	425	17241	DSKBT5	1	345	17210	DSKBTP
1	345	17170	DSKCB1	1	345	17174	DSKCB3	1	345	17202	DSKCB4	1	345	17205	DSKCB5
1	598	17346	DSKCV1	1	602	17350	DSKCV2	1	610	17355	dskcv3	1	596	17345	DSKCVA
1	229	17134	DSKMB1	1	93	17127	DSKMB2	1	238	17140	DSKMB3	1	251	17146	DSKMB4
1	260	17151	DSKMB5	1	86	17124	DSKMBT	1	575	17336	DSKRC2	1	571	17334	DSKRCL
1	498	17277	DSKSR2	1	515	17307	DSKSR3	1	493	17275	DSKSRC	17	953	17470	FMNBW1
17	952	17467	FMNBWT	17	965	17476	FMTHNG	7	378	17056	GET1BY	7	385	17060	GET2BY
7	400	17066	GET3BY	7	414	17074	GET4BY	5	170	17001	HALTED	8	455	17112	IBP
8	434	17102	ILDB	5	293	17016	IMGDUN	8	481	17122	IOBOTH	8	473	17121	IODSK
8	468	17117	IOTAP	16	933	17462	KNYGOA	16	920	17456	KNYRGO	16	908	17452	KNYW1
16	905	17450	KNYWAIT	0	38		0*NAF	8	464	17115	NEEDIO	6	354	17045	OLD-MMB-FIX
1	448	17255	SRCCCL	1	480	17265	SRCDMP	1	438	17245	SRCDSP	1	487	17272	SRCMFD
1	486	17271	SRCMFN	1	448	17253	SRCUCD	1	448	17251	SRCUDG	1	438	17242	SRCXXX
1	463	17257	SWAITS	14	805	17411	TAPAGN	14	854	17427	TAPEOF	14	843	17424	TAPEOR
15	868	17433	TAPERW	14	751	17364	TAPIO	14	760	17371	TAPIO1	14	772	17375	TPNEWF
14	771	17375	TPNEWR	14	810	17413	TPWTB1	14	829	17421	TPWTB2	5	169	17000	UBOOTA
1	345	17162	UBOOTD	5	183	17006	UBOOTT	5	171	17002	UBTA1	6	309	17020	ULOADL
6	304	17020	UMERGE	6	366	17053	XUBOOT								

18 0974 list-cormap

Storage map



2400 .....
2500 .....
2600 .....
2700 .....
3000 .....
3100 .....
3200 .....
3300 .....
3400 .....
3500 .....
3600 .....
3700 .....
4000 .....
4100 .....
4200 .....
4300 .....
4400 .....
4500 .....
4600 .....
4700 .....
5000 .....
5100 .....
5200 .....
5300 .....
5400 .....
5500 .....
5600 .....
5700 .....
6000 .....
6100 .....
6200 .....
6300 .....
6400 .....
6500 .....
6600 .....
6700 .....
7000 .....
7100 .....
7200 .....
7300 .....
7400 .....
7500 .....
7600 .....
7700 .....
10000 .....
10100 .....
10200 .....
10300 .....
10400 .....
10500 .....
10600 .....
10700 .....
11000 .....
11100 .....

```
11200 ..................................................................
11300 ..................................................................
11400 ..................................................................
11500 ..................................................................
11600 ..................................................................
11700 ..................................................................
12000 ..................................................................
12100 ..................................................................
12200 ..................................................................
12300 ..................................................................
12400 ..................................................................
12500 ..................................................................
12600 ..................................................................
12700 ..................................................................
13000 ..................................................................
13100 ..................................................................
13200 ..................................................................
13300 ..................................................................
13400 ..................................................................
13500 ..................................................................
13600 ..................................................................
13700 ..................................................................
14000 ..................................................................
14100 ..................................................................
14200 ..................................................................
14300 ..................................................................
14400 ..................................................................
14500 ..................................................................
14600 ..................................................................
14700 ..................................................................
15000 ..................................................................
15100 ..................................................................
15200 ..................................................................
15300 ..................................................................
15400 ..................................................................
15500 ..................................................................
15600 ..................................................................
15700 ..................................................................
16000 ..................................................................
16100 ..................................................................
16200 ..................................................................
16300 ..................................................................
16400 ..................................................................
16500 ..................................................................
16600 ..................................................................
16700 ..................................................................
17000 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
17100 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
17200 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
17300 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
17400 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
17500 ..................................................................
17600 ..................................................................
17700 ..................................................................
```

320 locations used, highest used = 17477

SLOE Aug 18, 1986 18:08:11 file DSK:XBOOT.SLO -- of -- XBOOT

01 0005