



```

0000 1: ;-----
0000 2: ;
0000 3: ;      Copyright 2010 Intergraded Logic Systems
0000 4: ;      Source Code is Copyright Stephen J. Car
0000 5: ;
0000 6: ;
0000 7: ;      Please do not share this source!
0000 8: ;-----
0000 9: ;      (General hard disk formatter for real.dos SpartaDO
0000 10: ;
0000 11: ;-----
0000 12: ;      Real.dos
0000 13: ;      This program is to replace hdfmdir.COM wit
0000 14: ;      fmtmdir.com
0000 15: ;
0000 16: ;      Code History.
0000 17: ;
0000 18: ;Init design,      10/28/2005      sjc
0000 19: ;  updated      10/30/2005      sjc
0000 20: ;  updated      12/30/2008      sjc
0000 21: ;      Holding down the "select key will force 655
0000 22: ;      changed name from hdfmdir to fmtmdir
0000 23: ;
0000 24: ;Last updated      08/29/2012      sjc
0000 25: ;      changed to call from $e459 to 'LSIO'
0000 26: ;
0000 27: ;
0000 28: ;
0000 29: ;      ~~~~~
0000 30: ;-----
0000 31: ; Notes: o This source code MAY NOT be placed for download
0000 32: ;      o "mea" is a macro that loads the address of the
0000 33: ;      into the pointer specified by the second field.
0000 34: ;-----
0000 35: ; Assembler: MADMAC (tm) ST Cross Assembler (Atari Corp)
0000 36: ;      XASM  st and IBM VERSIONS
0000 37: ;-----
0000 38: ;
0000 39: com_file_name:      .macro
0000 40: ;      dc.b  13,"filename.ext",$9b
0000 41: ;      dc.b  12,"Fmtmdir.com ",$9b
0000 42: ;      .endm
0000 43: ;
0000 44: ;
0000 45: ;~~~~~
0000 46: ; File revision history. Version Number in hex
0000 47: ;
0000 48: file_ver:      equ  $18
0000 49: ;~~~~~
0000 50: ; the Month in dec for the first time this revision
0000 51: ;was compiled
0000 52: ;
0000 53: c_month:      equ  10

```

```

0000 54: ;~~~~~
0000 55: ; the day in dec for the first time this revision
0000 56: ;was compiled
0000 57: c_day:          equ 12
0000 58: ;~~~~~
0000 59: ; the year in dec for the first time this revision
0000 60: ;was compiled
0000 61: c_year:          equ 2012
0000 62: ;~~~~~
0000 63: ; Type of compiler used in program
0000 64: ;
0000 65: ; 0 = unknown Compiler
0000 66: ; 1 = xasm
0000 67: ; 2 = mac_65
0000 68: ; 3 = basic
0000 69: ; 4 = compiled basic xl
0000 70: ; 5 = C65
0000 71: ; 6 = Action
0000 72: ;
0000 73: ; We can add more as time goes on!
0000 74: ;
0000 75: ;
0000 76: ;
0000 77: xasm:            equ 1
0000 78: mac_65:          equ 2
0000 79: basic:           equ 3
0000 80: basicxl:         equ 4
0000 81: c65:             equ 5
0000 82: action:          equ 6
0000 83:
0000 84: file_compiler:    equ xasm
0000 85: ;~~~~~
0000 86: ; is this relocatable code ?
0000 87: ; anyother value other than 1 or 2 would be unknown
0000 88: r..yes:          equ 1
0000 89: r..no:           equ 2
0000 90: ;
0000 91: ;~~~~~
0000 92: ; Gotta define if it can be relocated
0000 93: l_relocatable:    equ r..no
0000 94: ;
0000 95: ;
0000 96: ;~~~~~
0000 97: ; Gotta define our Header control Byte
0000 98: ;$ff=bypass intro
0000 99: ;$15 = Bypass Dos check
0000 100: r..crl:          equ $7d
0000 101: r..crlf:         equ $9b
0000 102: r..space:        equ $20
0000 103: r..bypassDOS:    equ $15
0000 104: r..bypass:       equ $ff
0000 105:
0000 106: l_frstscreenbyte: equ r..space

```

```

0000    107: ;~~~~~
0000    108: ; The language the output file is in.
0000    109: ;
0000    110: ;
0000    111: ; 0 =   Undefined
0000    112: ; 1 =   English
0000    113: ; 2 =   German
0000    114: ;
0000    115: r..language:      equ   1
0000    116: ;~~~~~
0000    117: ;
0000    118: ;
0000    119:
0000    120:          .include   equates
0000    121:          .include   globals
0000    122:          .include   macros
0000    123: ;
4000    124:          .org   $4000
4000    125: ;
4000    126:
4000    127: win_start:
4000    128: header_info:
4000    129:          .include   header
42a7    130:
42a7 a03f 131:          ldy   #LBUF
42a9 b10a 132: mux_1a:          lda   (comtab),y
42ab c920 133:          cmp   #' '
42ad f00b 134:          beq   mux_1b
42af c99b 135:          cmp   #$9b
42b1 f004 136:          beq   .a
42b3 c8   137:          iny
42b4 4ca942 138:          jmp   mux_1a
42b7 4c0080 139: .a:          jmp   start_it
42ba      140:
42ba      141: mux_1b:
42ba a200 142:          ldx   #$00
42bc c8   143:          iny
42bd      144:
42bd b10a 145: .a:          lda   (comtab),y
42bf 20fd42 146:          jsr   upcase
42c2 c99b 147:          cmp   #$9b
42c4 f00d 148:          beq   .noparm
42c6 c944 149:          cmp   #'D'
42c8 f00c 150:          beq   .got_d
42ca e8   151:          inx
42cb 8ed242 152:          stx   .holdx
42ce c8   153:          iny
42cf 4cbd42 154:          jmp   .a
42d2      155:
42d2 00   156: .holdx:          dc.b  0
42d3      157:
42d3 4c0080 158: .noparm:          jmp   START_IT
42d6      159:

```

```

42d6 c8 160: .got_d:      iny
42d7 b10a 161:      lda (comtab),y
42d9 c931 162:      cmp #'1'
42db 90f6 163:      bcc .noparm
42dd c93a 164:      cmp #'9'+1
42df b0f2 165:      bcs .noparm
42e1 8dfc42 166:      sta .myhold
42e4 e92f 167:      sbc #47
42e6 8d0103 168:      STA DUNIT
42e9 c8 169:      iny
42ea b10a 170:      lda (comtab),y
42ec 20fd42 171:      jsr upcase
42ef c951 172:      cmp #'Q'
42f1 d0e0 173:      bne .noparm
42f3 8d1d80 174:      sta bypass_menu
42f6 adfc42 175:      lda .myhold
42f9 4c0080 176:      jmp start_it
42fc      177:
42fc      178:
42fc 00 179: .myhold:      dc.b 0
42fd      180: ;
42fd      181: ;-----
42fd      182: ;CHANGES THE CHARTER TO UPCASE
42fd      183: ;-----
42fd      184: upcase:
42fd c961 185:      cmp #61
42ff 9006 186:      bcc .a
4301 c97a 187:      cmp #7a
4303 b002 188:      bcs .a
4305 e91f 189:      sbc #1f
4307 60 190: .a:      rts
4308      191:
4308      192:
4308      193:
4308 4c0080 194:      jmp START_IT
430b      195:
430b      196:
430b      197: ;      .org $4560
430b      198: Split:
430b      199: BOOTI:
430b      200: ;#####
430b      201: ;
430b      202: ; $3000
430b      203: ; =====
430b      204: ; Stage 1 boot program
430b      205: ; =====
430b      206:
430b      207: initz1:
430b      208: ;
430b      209: BOOTS:
430b 00 210:      dc.b $00
430c 03 211:      dc.b $03
430d 00 212:      dc.b $00

```

```

430e 30 213:      dc.b  $30
430f e007 214:      cpx  #$07
4311 4c8b43 215: w_f01:  JMP  LOADER
4314      216: ;
4314      217: ;sector map main directory
4314      218: ;+9:
4314 0000 219:      dc.w  0
4316      220: ; Total number of sectors on disk
4316      221: ;+11:
4316 0000 222:      dc.w  0
4318      223: ; Number of free sectors on disk
4318      224: ;+13:
4318 0000 225:      dc.w  0
431a      226: ; Number of bit map sectors
431a      227: ;+15:
431a 00 228:      dc.b  0
431b      229: ;
431b 0400 230:      dc.w  4
431d      231: ;+18:
431d 0000 232:      dc.w  0
431f      233: ;+20:
431f 0000 234:      dc.w  0
4321      235: ;
4321      236: ;+22
4321      237: Disk_Volume_name:
4321 202020 238:      dc.b  "  "
4329      239: ;+30:
4329 00 240:      dc.b  0
432a      241: ;+31:
432a      242: DENLOA:
432a 00 243:      dc.b  0
432b 20 244:      dc.b  $20
432c 06 245:      dc.b  6
432d 01 246:      dc.b  1
432e ff 247:      dc.b  -1
432f ff 248:      dc.b  -1
4330 00 249:      dc.b  0
4331 00 250:      dc.b  0
4332      251:
4332      252: ;+39:
4332 00 253:      dc.b  0
4333      254: ;+40:
4333      255: FIRMAP:
4333 0000 256:      dc.w  0
4335 00 257:      dc.b  0
4336      258:
4336      259: ;+43:
4336 00 260:      dc.b  0
4337      261: ;+44:
4337 00 262:      dc.b  0
4338      263: ;+45:
4338 0000 264:      dc.w  0
433a      265: ;+47:

```

```

433a 00 266:          dc.b 0
433b      267: ;+48:
433b 457272 268: NODMSG:          dc.b "Error: No DOS",$9b
4349      269: ;
4349      270: ;
4349      271: ;   get next sector map.. initz new map #
4349      272: ;   -----
4349      273: ;
4349      274: ;
4349      275: LOANM:
4349 ad3343 276: w_f02:      LDA  FIRMAP
434c 8d0a03 277:          STA  SECTOR
434f ad3443 278: w_f03:      LDA  FIRMAP+1
4352 8d0b03 279:          STA  SECTOR+1
4355 a900 280:          LDA  #low secmap
4357 a22f 281:          LDX  #high secmap
4359 20fc43 282: w_f04:      JSR  READS
435c ad002f 283:          LDA  secmap
435f 8d3343 284: w_f05:      STA  FIRMAP
4362 ad012f 285:          LDA  secmap+1
4365 8d3443 286: w_f06:      STA  FIRMAP+1
4368 a004 287:          LDY  #4
436a 8491 288:          STY  SECPTR
436c      289: ;
436c      290: ;   get next sector number..
436c      291: ;   -----
436c      292: ;
436c      293: ;
436c a491 294: GNXSEC:      LDY  SECPTR
436e cc2a43 295: w_f07:      CPY  DENLOA
4371 f0d6 296:          BEQ  LOANM
4373 b9002f 297:          LDA  secmap,Y
4376 8d0a03 298:          STA  SECTOR
4379 b9012f 299:          LDA  secmap+1,Y
437c 8d0b03 300:          STA  SECTOR+1
437f c8 301:          INY
4380 c8 302:          INY
4381 8491 303:          STY  SECPTR
4383 60 304: LRTS:      RTS
4384      305: ;
4384 6ce202 306: DOINITZ:  JMP ($2e2)
4387      307: ;
4387      308: l407c:
4387 00 309:          dc.b 0
4388 00 310:          dc.b 0
4389 00 311:          dc.b 0
438a 00 312:          dc.b 0
438b      313: ;
438b a200 314: LOADER:      LDX  #0
438d ad2a43 315: w_f08:      LDA  DENLOA          ; s
4390 8591 316:          STA  SECPTR          ;
4392 8590 317:          STA  CHPTR          ;
4394 8d0803 318:          STA  DBYTLO          ; s

```

```

4397 d001 319:      BNE  ISX
4399 e8 320:      INX
439a 8e0903 321: ISX:      STX  DBYTHI          ; s
439d 201544 322: w_f09:    JSR  GETBYTE          ; g
43a0 8596 323:      STA  temp
43a2 201544 324: w_f10:    JSR  GETBYTE
43a5 2596 325:      AND  temp
43a7 c9ff 326:      CMP  #$FF
43a9 d037 327:      BNE  TYPER          ; N
43ab      328: BEGIN:
43ab a983 329: w_lb_1:      LDA  # low LRTS
43ad 8de202 330:      STA  INITAD
43b0 a943 331: w_hb_1:      LDA  # high LRTS
43b2 8de302 332:      STA  INITAD+1
43b5 201544 333: w_f11:    JSR  GETBYTE
43b8 8592 334:      STA  buf          ; g
43ba 201544 335: w_f12:    JSR  GETBYTE
43bd 8593 336:      STA  buf+1
43bf 0592 337:      ORA  buf
43c1 f01c 338:      BEQ  STAD          ; a
43c3 201544 339: w_f13:    JSR  GETBYTE
43c6 38 340:      SEC
43c7 e592 341:      SBC  buf          ; c
43c9 48 342:      PHA
43ca 08 343:      PHP
43cb 201544 344: w_f14:    JSR  GETBYTE
43ce 28 345:      PLP
43cf e593 346:      SBC  buf+1
43d1 8595 347:      STA  LENG+1        ; s
43d3 68 348:      PLA
43d4 8594 349:      STA  LENG
43d6      350:
43d6 207644 351: w_f15:    JSR  LOABUF          ; and load
43d9 208443 352: w_f16:    JSR  DOINITZ          ; r
43dc 4cab43 353: w_f17:    JMP  BEGIN          ; do start.
43df      354: ;
43df 6ce002 355: STAD:      JMP  ($2e0)
43e2      356: ;
43e2      357: ;
43e2      358: ;
43e2      359: ; error.. no DOS on diskette
43e2      360: ; -----
43e2      361: ;
43e2      362: TYPER:
43e2 a93b 363: w_lb_2:      LDA  #low NODMSG
43e4 a243 364: w_hb_2:      LDX  #high NODMSG
43e6 8d4403 365:      STA  ICBAL
43e9 8e4503 366:      STX  ICBAH
43ec 8e4803 367:      STX  ICBL
43ef a909 368:      LDA  #9
43f1 8d4203 369:      STA  ICCOM
43f4 a200 370:      LDX  #0
43f6 2056e4 371:      JSR  CIO

```



```

43f9      372:
43f9      373: FOREV:
43f9 4cf943 374: w_f18:      JMP   FOREV           ; endless l
43fc      375: ;
43fc      376: ;
43fc      377: ;
43fc      378: ;      read sector at address
43fc      379: ;      -----
43fc      380: ;
43fc      381: READS:
43fc a040 382:      LDY   #$40           ; r
43fe 8c0303 383:      STY   DSTATS
4401 8d0403 384:      STA   DBUFLO
4404 8e0503 385:      STX   DBUFHI           ; n
4407 ad0a03 386:      LDA   SECTOR
440a 0d0b03 387:      ORA   SECTOR+1
440d f0d3 388:      BEQ   TYPER           ; i
440f 201580 389:      JSR   XSIO           ; u
4412 30ce 390:      BMI   TYPER           ; i
4414 60 391:      RTS
4415      392: ;
4415      393: ;
4415      394: ;      get byte/burst routine
4415      395: ;      -----
4415      396: ;
4415      397: ;
4415      398: GETBYTE:
4415 a900 399:      LDA   #0
4417 8595 400:      STA   LENG+1
4419 8594 401:      STA   LENG
441b      402:
441b a690 403: GETBYT:      LDX   CHPTR
441d ec2a43 404: w_f19:      CPX   DENLOA           ; c
4420 f006 405:      BEQ   LOANS           ;
4422 bd002e 406:      LDA   datbuf,X
4425 e690 407:      INC   CHPTR
4427 60 408:      RTS
4428      409: ;
4428      410: ;
4428      411: ;
4428      412: ;      Load next sector in buffer/ or burst if possible
4428      413: ;      -----
4428      414: ;
4428      415: LOANS:
4428 206c43 416: w_f20:      JSR   GNXSEC           ; g
442b a595 417:      LDA   LENG+1           ; c
442d d017 418:      BNE   OKBUR
442f ad2a43 419: w_f21:      LDA   DENLOA
4432 f004 420:      BEQ   NOBUR
4434 a594 421:      LDA   LENG
4436 300e 422:      BMI   OKBUR           ; i
4438      423:
4438 a900 424: NOBUR:      LDA   #low datbuf

```

```

443a a22e 425:      LDX  #high datbuf
443c 20fc43 426: w_f22: JSR  READS          ; r
443f 38 427:      SEC
4440 2690 428:      ROL  CHPTR          ; p
4442 ad002e 429:      LDA  datbuf          ; g
4445 60 430:      RTS
4446 431: ;
4446 432: ;
4446 433: ; Do burst read sector
4446 434: ; -----
4446 435: ;
4446 a592 436: OKBUR: LDA  buf          ; g
4448 a693 437:      LDX  buf+1
444a 20fc43 438: w_f23: JSR  READS          ; r
444d a592 439:      LDA  buf          ; a
444f 18 440:      CLC
4450 6d0803 441:      ADC  DBYTLO
4453 8592 442:      STA  buf
4455 a593 443:      LDA  buf+1
4457 6d0903 444:      ADC  DBYTHI
445a 8593 445:      STA  buf+1
445c 38 446:      SEC          ; subtract
445d a594 447:      LDA  LENG
445f ed0803 448:      SBC  DBYTLO
4462 8594 449:      STA  LENG
4464 a595 450:      LDA  LENG+1
4466 ed0903 451:      SBC  DBYTHI
4469 8595 452:      STA  LENG+1
446b 4c2844 453: w_f24: JMP  LOANS          ; g
446e 454: ;
446e 455: ;
446e 456: ;
446e 457: ; Load entire buffer from file
446e 458: ; -----
446e 459: ;
446e 460: ;
446e a594 461: LOABUF: LDA  LENG          ; d
4470 d002 462:      BNE  NOB
4472 c695 463:      DEC  LENG+1
4474 c694 464: NOB:   DEC  LENG
4476 465:
4476 466:
4476 467: LOABUF:
4476 201b44 468: w_f25: JSR  GETBYT          ; g
4479 a000 469:      LDY  #0
447b 9192 470:      STA  (buf),Y
447d e692 471:      INC  buf
447f d002 472:      BNE  SK11
4481 e693 473:      iNC  buf+1
4483 a594 474: SK11: LDA  LENG
4485 0595 475:      ORA  LENG+1
4487 d0e5 476:      BNE  LOABUF
4489 60 477:      RTS

```

```

448a 00 478:      brk
448b      479:
448b      480: zend1:  ds.b  0
448b      481:
448b      482:
448b      483: ;
448b      484: ;
448b      485: ;
448b      486: ;
8000      487:      .org  $8000
8000      488: ;=====
8000      489: ;   Beginning and initialization
8000      490: ;=====
8000      491: ;
8000      492: ;
8000      493: START_IT:
8000 20f08d 494:      jsr  rlocate
8003      495:
8003 a50a 496: BEGINI:      LDA  COMTAB
8005 38 497:      SEC
8006 e90a 498:      SBC  #low lsio
8008 8d1680 499:      STA  XSIO+1
800b a50b 500:      LDA  COMTAB+1
800d e900 501:      SBC  #high lsio
800f 8d1780 502:      STA  XSIO+2
8012 4c1e80 503:      JMP  FORMAT
8015 6c0000 504: XSIO:      jmp  (0)
8018      505: ;
8018 ae2b88 506: RETURN:      LDX  stackp
801b 9a 507:      TXS
801c 60 508:      RTS
801d      509: ;
801d      510: ;
801d      511: ;=====
801d      512: ;   Main parameter input
801d      513: ;=====
801d 00 514: bypass_menu: dc.b  0
801e      515:
801e ba 516: format:      TSX
801f 8e2b88 517:      STX  stackp
8022      518:
8022      519:
8022      520: ;      JSR  printsi
8022      521: ;      dc.b  $7D,$9B,$9b,$9b
8022      522: ;      dc.b  "  RealDos Directory Format Utility
8022      523: ;      dc.b  "  Written By Stephen J. Carden",
8022      524: ;      dc.b  "  (C)2010 Integrated Logic Systems
8022      525: ;      dc.b  $9b,$9b,$9b,-1
8022      526:
8022      527:
8022 ad1d80 528:      lda  bypass_menu
8025 c900 529:      cmp  #$00
8027 f003 530:      beq  .fmt

```

```

8029 4ce480 531:      jmp    f.c
802c      532: .fmt:
802c 20a285 533:      JSR    printsi
802f 9b9b 534:      dc.b   $9b,$9b
8031 506c65 535:      dc.b   "Please enter the number of the",$9
8050 447269 536:      dc.b   "Drive (Dn:) you wish to format: ",
8071      537:
8071      538: f_menu:
8071 207e85 539:      jsr    inch
8074 c931 540:      CMP    #'1'
8076 90f9 541:      BCC    f_menu
8078 c93a 542:      CMP    #'9'+1
807a b0f5 543:      BCS    f_menu
807c 208985 544:      JSR    ECHO
807f 20a285 545:      JSR    printsi
8082 9bff 546:      dc.b   $9B,-1
8084      547: ;      AND    #7
8084 e92f 548:      sbc    #47          ; t
8086 8d0103 549:      STA    DUNIT
8089 20a285 550:      JSR    printsi
808c 9b9b 551:      dc.b   $9B,$9B
808e 566f6c 552:      dc.b   "Volume Name ? ",-1
809d      553:
809d a008 554:      ldy    #8
809f a921 555:      LDA    #low Disk_Volume_name
80a1 a243 556:      LDX    #high Disk_Volume_name
80a3 202686 557:      JSR    GETLINE
80a6 20a285 558:      JSR    printsi
80a9 9b9b 559:      dc.b   $9B,$9B
80ab 496e69 560:      dc.b   "Initialize... Are You Sure? ",-1
80c8      561:
80c8 207e85 562: .a:      jsr    inch
80cb c959 563:      CMP    #'Y'
80cd f015 564:      BEQ    .c
80cf c979 565:      CMP    #'y'
80d1 f011 566:      BEQ    .c
80d3 c94e 567:      CMP    #'N'
80d5 f004 568:      BEQ    .b
80d7 c96e 569:      CMP    #'n'
80d9 d0ed 570:      BNE    .a
80db      571: .b:
80db 20a285 572:      JSR    printsi
80de 6e9bff 573:      dc.b   "n",$9b,-1
80e1 4c1880 574:      JMP    RETURN
80e4      575: ;
80e4      576: .c:
80e4      577: f.c:
80e4 20a285 578:      jsr    printsi
80e7 599bff 579:      dc.b   "Y",$9b,-1
80ea      580:
80ea      581:
80ea      582:
80ea      583:

```

```

80ea a939 584:      LDA  #low NAMES
80ec 8d0403 585:      STA  DBUFLO
80ef a988 586:      LDA  #high NAMES
80f1 8d0503 587:      STA  DBUFHI
80f4 a931 588:      LDA  #$31
80f6 8d0003 589:      STA  DDEVIC
80f9 a94e 590:      LDA  #'N'
80fb 8d0203 591:      STA  DCOMND
80fe a940 592:      LDA  #$40
8100 8d0303 593:      STA  DSTATS
8103 a90c 594:      LDA  #low 12
8105 8d0803 595:      STA  DBYTLO
8108 a900 596:      LDA  #high 12
810a 8d0903 597:      STA  DBYTHI
810d 201580 598:      JSR  xsio
8110 1022 599:      BPL  .d
8112 20a285 600:      JSR  printsi
8115 3f3f44 601:      dc.b  "??Drive not Configurable??",$9b,-1
8131 4c1880 602:      JMP  RETURN
8134      603: ;
8134      604:
8134      605: .d:
8134 207584 606:      jsr  calc_sector_count
8137      607:
8137 ad1fd0 608:      lda  consol
813a c907 609:      cmp  #consol_any
813c f00b 610:      beq  .tam
813e ad1fd0 611:      lda  consol
8141 c905 612:      cmp  #consol_select
8143 f012 613:      beq  .damn
8145 a908 614:      lda  #08
8147 d016 615:      bne  .usp
8149 ad2243 616: .tam:      Lda  Disk_Volume_name+1
814c c900 617:      cmp  #$00
814e d03d 618:      bne  .e
8150 ad2143 619:      lda  Disk_Volume_name
8153 c900 620:      cmp  #$00
8155 d036 621:      bne  .e
8157      622: .damn:
8157 a9ff 623:      lda  #$ff          ; loading a
8159 8d2243 624:      sta  Disk_Volume_name+1
815c 8d2143 625:      sta  Disk_Volume_name
815f      626:
815f      627: .usp:
815f 20a285 628:      jsr  printsi
8162 9b9b20 629:      dc.b  $9b,$9b," Setting Disk to ",-1
8176      630:
8176 ad4385 631:      lda  end_sector
8179 ae4485 632:      ldx  end_sector+1
817c 206185 633:      jsr  pr_card
817f      634:
817f      635:
817f 20a285 636:      jsr  printsi

```

```

8182 205365 637:      dc.b  " Sectors",$9b,$9b,-1
818d      638: ;
818d      639: .e
818d a901 640:      LDA  #1
818f 8d2588 641:      STA  DENS
8192 c901 642:      CMP  #1
8194 f021 643:      BEQ  .f
8196 20a285 644:      JSR  printsi
8199 3f3f53 645:      dc.b  "??Shows Multiple Tracks??",$9b,-1
81b4 4c1880 646:      JMP  RETURN
81b7      647: ;
81b7      648: .f
81b7 ad4185 649:      lda  real_sector_count
81ba      650: ;      LDA Disk_Volume_name+1
81ba 8d2388 651:      STA  MAXSEC
81bd      652:
81bd ad4285 653:      lda  real_sector_count+1
81c0      654: ;      LDA  Disk_Volume_name
81c0 8d2488 655:      STA  MAXSEC+1
81c3      656:
81c3 ad4088 657:      LDA  NAMES+7
81c6 8d2188 658:      STA  DENSITY
81c9 a9ff 659:      LDA  #$FF
81cb 8d2288 660:      STA  SECTORS
81ce 209b87 661:      JSR  RECALC
81d1 207186 662:      JSR  WRITE
81d4 20a285 663:      JSR  printsi
81d7 9b9b 664:      dc.b  $9b,$9b
81d9 447269 665:      dc.b  "Drive Initialized...",$9b,-1
81ef 4c1880 666:      JMP  RETURN
81f2      667: ;
81f2      668: ;
81f2      669: ;End of the directory formatting
81f2      670: ;
81f2      671: ;
81f2      672: get_sector_count:
81f2 a931 673:      lda  #$31
81f4 8d0003 674:      sta  $300
81f7 ad0103 675:      lda  dunit
81fa 8d0103 676:      sta  $301
81fd a94e 677:      lda  #'N'          ;$4e
81ff 8d0203 678:      sta  $0302
8202 a940 679:      lda  #$40
8204 8d0303 680:      sta  $0303
8207 a913 681:      lda  # low huh_data
8209 8d0403 682:      sta  $0304
820c a984 683:      lda  # high huh_data
820e 8d0503 684:      sta  $0305
8211 a90c 685:      lda  #$0c
8213 8d0803 686:      sta  $0308
8216 a900 687:      lda  #$00
8218 8d0903 688:      sta  $0309
821b 8d0a03 689:      sta  $030a

```

```

821e 8d0b03 690:      sta  $030b
8221 201580 691:      jsr  xsio          ; Changed f
8224 ad0303 692:      lda  $0303
8227 c901  693:      cmp  #$01
8229 f001  694:      beq  .huh_1
822b 60    695:      rts
822c      696: .huh_1:
822c ad1fd0 697:      lda  consol
822f c905  698:      cmp  #consol_select
8231 f001  699:      beq  tam
8233 60    700:      rts
8234      701: tam:
8234 20a285 702:      jsr  printsi
8237 7d9b9b 703:      dc.b  $7d,$9b,$9b,$9b
823b 202020 704:      dc.b  "      Drive configuration table
8266 ad1384 705:      lda  huh_data
8269 20fc83 706:      jsr  sub_error
826c 20a285 707:      jsr  printsi
826f 6e756d 708:      dc.b  "number of tracks",$9B,$FF
8281 ad1484 709:      lda  huh_data+1
8284 20fc83 710:      jsr  sub_error
8287 20a285 711:      jsr  printsi
828a 537465 712:      dc.b  "Step rate! normaly 1",$9B,$FF
82a0 ad1584 713:      lda  huh_data+2
82a3 20fc83 714:      jsr  sub_error
82a6 20a285 715:      jsr  printsi
82a9 536563 716:      dc.b  "Sector/Track high byte",$9B,$FF
82c1 ad1684 717:      lda  huh_data+3
82c4 20fc83 718:      jsr  sub_error
82c7 20a285 719:      jsr  printsi
82ca 536563 720:      dc.b  "Sector/Track LOW byte",$9B,$FF
82e1 ad1784 721:      lda  huh_data+4
82e4 20fc83 722:      jsr  sub_error
82e7 20a285 723:      jsr  printsi
82ea 4d6178 724:      dc.b  "Max head number",$9B,$FF
82fb ad1884 725:      lda  huh_data+5
82fe 20fc83 726:      jsr  sub_error
8301 20a285 727:      jsr  printsi
8304 44656e 728:      dc.b  "Density -0=s, 4 double, 8 High",$9
8324 ad1984 729:      lda  huh_data+6
8327 20fc83 730:      jsr  sub_error
832a 20a285 731:      jsr  printsi
832d 427974 732:      dc.b  "Byte/sector h byte 1=256 0=128",$9
834d ad1a84 733:      lda  huh_data+7
8350 20fc83 734:      jsr  sub_error
8353 20a285 735:      jsr  printsi
8356 427974 736:      dc.b  "Byte/sector l byte 0=256 1=128",$9
8376 ad1b84 737:      lda  huh_data+8
8379 20fc83 738:      jsr  sub_error
837c 20a285 739:      jsr  printsi
837f 447269 740:      dc.b  "Drive present flag -returns $ff",$
83a0 ad1c84 741:      lda  huh_data+9
83a3 20fc83 742:      jsr  sub_error

```

```

83a6 20a285 743:      jsr  printsi
83a9 6e6f74 744:      dc.b  "not used",$9B,$FF
83b3 ad1d84 745:      lda  huh_data+10
83b6 20fc83 746:      jsr  sub_error
83b9 20a285 747:      jsr  printsi
83bc 6e6f74 748:      dc.b  "not used",$9B,$FF
83c6 ad1e84 749:      lda  huh_data+11
83c9 20fc83 750:      jsr  sub_error
83cc 20a285 751:      jsr  printsi
83cf 6e6f74 752:      dc.b  "not used",$9B
83d8 9b5072 753:      dc.b  $9b,"Press any key to continue",$FF
83f3 200b86 754:      jsr  get_key
83f6 a97d 755:      lda  #$7d
83f8 208985 756:      jsr  echo
83fb 60 757:      rts
83fc 758: sub_error:
83fc 8d1284 759:      sta  .dan_k
83ff 20a285 760:      jsr  printsi
8402 2024ff 761:      dc.b  " ",$FF
8405 ad1284 762:      lda  .dan_k
8408 204685 763:      jsr  drive_error
840b 20a285 764:      jsr  printsi
840e 2020ff 765:      dc.b  " ",$FF
8411 60 766:      rts
8412 767: ;
8412 00 768: .dan_k:      dc.b  0
8413 000000 769: huh_data:  dc.b  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
8426 000000 770: huh_data_d: dc.b  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
8439 771:
8439 772: check_for_emulator:
8439 ad1384 773:      lda  huh_data
843c c928 774:      cmp  #$28
843e f001 775:      beq  .mite_be
8440 60 776:      rts
8441 777: .mite_be
8441 ad1584 778:      lda  huh_data+2
8444 c900 779:      cmp  #$00
8446 f001 780:      beq  .flop
8448 60 781:      rts
8449 782: .flop:
8449 ad1684 783:      lda  huh_data+3
844c c913 784:      cmp  #18+1
844e 9001 785:      bcc  .go_on
8450 60 786:      rts
8451 787: .go_on:
8451 ad1b84 788:      lda  huh_data+8
8454 c901 789:      cmp  #$01
8456 d01c 790:      bne  .end
8458 ad1c84 791:      lda  huh_data+9
845b c9c0 792:      cmp  #$c0
845d d015 793:      bne  .end
845f ad1d84 794:      lda  huh_data+10
8462 c900 795:      cmp  #$00

```



```

8464 d00e 796:      bne .end
8466 ad1e84 797:      lda  huh_data+11
8469      798: ;      cmp  #$40          ;???????
8469      799: ;      bne  .end
8469      800: ;
8469      801: ; This is for a 720 drive
8469      802: ; ok we have an emulator drive
8469      803: ;      lda  huh_data+2
8469      804: ;      cmp  #high 18
8469      805: ;      bne  .end
8469      806: ;      lda  huh_data+3
8469      807: ;      cmp  #low 18
8469      808: ;      bne  .a
8469      809:
8469 a91e 810:      lda  #30
846b 8de687 811:      sta  walter+1
846e a904 812:      lda  #$04
8470 8d1884 813:      sta  huh_data+5
8473      814: .a:
8473 60 815:      rts
8474      816: .end:
8474 60 817:      rts
8475      818:
8475      819:
8475      820:
8475      821: calc_sector_count:
8475 20f281 822:      jsr  get_sector_count
8478 203984 823:      jsr  check_for_emulator
847b      824: ;      jsr  tam
847b a9ff 825:      lda  #$ff
847d 8d4585 826:      sta  sector_density
8480      827:
8480 ad1884 828:      lda  huh_data+5
8483 c904 829:      cmp  #$04
8485 f00c 830:      beq  .aa
8487      831:
8487 c908 832:      cmp  #$08
8489 d010 833:      bne  .a
848b a902 834:      lda  #$02
848d 8d3f85 835:      sta  percom_density
8490 4cac84 836:      jmp  .b
8493      837:
8493 a901 838: .aa:      lda  #$01
8495 8d3f85 839:      sta  percom_density
8498 4cac84 840:      jmp  .b
849b      841:
849b a901 842: .a:      lda  #$01
849d 8d3f85 843:      sta  percom_density
84a0 ad1884 844:      lda  huh_data+5
84a3 c900 845:      cmp  #$00
84a5 d005 846:      bne  .b
84a7 a980 847:      lda  #$80
84a9 8d4585 848:      sta  sector_density

```

```
84ac      849:
84ac      850: .b:
84ac ad1584 851:      lda  huh_data+2
84af c900  852:      cmp  #$00
84b1 d00f  853:      bne  .c
84b3 ad1684 854:      lda  huh_data+3
84b6 c900  855:      cmp  #$00
84b8 d008  856:      bne  .c
84ba a9ff  857:      lda  #$ff
84bc 8d1584 858:      sta  huh_data+2
84bf 8d1684 859:      sta  huh_data+3
84c2      860:
84c2 ad1584 861: .c:      lda  huh_data+2
84c5 8d3e85 862:      sta  sector_per_trac+1
84c8 ad1684 863:      lda  huh_data+3
84cb 8d3d85 864:      sta  sector_per_trac
84ce      865:
84ce ad1384 866:      lda  huh_data
84d1 c903  867:      cmp  #$03
84d3 b005  868:      bcs  .d
84d5 a901  869:      lda  # low 1
84d7 8d1384 870:      sta  huh_data
84da      871:
84da ad1384 872: .d:      lda  huh_data
84dd 8d3b85 873:      sta  number_tracks
84e0 a900  874:      lda  #$00
84e2 8d3c85 875:      sta  number_tracks+1
84e5      876:
84e5      877:
84e5 2044da 878:      jsr  zfr0
84e8 ad3b85 879:      lda  number_tracks
84eb 85d4  880:      sta  fr0
84ed ad3c85 881:      lda  number_tracks+1
84f0 85d5  882:      sta  fr0+1
84f2 20aad9 883:      jsr  ifp
84f5 a900  884:      lda  #$00
84f7 85f2  885:      sta  cix
84f9 20b6dd 886:      jsr  fmove
84fc      887:
84fc      888:
84fc ad3d85 889:      lda  sector_per_trac
84ff 85d4  890:      sta  fr0
8501 ad3e85 891:      lda  sector_per_trac+1
8504 85d5  892:      sta  fr0+1
8506 a900  893:      lda  #$00
8508 85f2  894:      sta  cix
850a 20aad9 895:      jsr  ifp
850d 20dbda 896:      jsr  fmult
8510 20b6dd 897:      jsr  fmove
8513      898:
8513      899:
8513 ad3f85 900:      lda  percom_density
8516 85d4  901:      sta  fr0
```

```

8518 ad4085 902:      lda  percom_density+1
851b 85d5  903:      sta  fr0+1
851d a900  904:      lda  #$00
851f 85f2  905:      sta  cix
8521 20aad9 906:      jsr  ifp
8524 20dbda 907:      jsr  fmult
8527      908:
8527 20d2d9 909:      jsr  fpi
852a a5d4  910:      lda  fr0
852c a6d5  911:      ldx  fr0+1
852e 8d4185 912:      sta  real_sector_count
8531 8e4285 913:      stx  real_sector_count+1
8534 8d4385 914:      sta  end_sector
8537 8e4485 915:      stx  end_sector+1
853a 60    916:      rts
853b      917:
853b 0000  918: number_tracks:  dc.w  0
853d 0000  919: sector_per_trac:  dc.w  0
853f 0000  920: percom_density:   dc.w  0
8541 0000  921: real_sector_count: dc.w  0
8543 0000  922: end_sector:      dc.w  0
8545 ff    923: sector_density:   dc.b  -1
8546      924:
8546      925:
8546      926: ;
8546      927: drive_error:
8546 48    928:      pha
8547 4a    929:      lsr
8548 4a    930:      lsr
8549 4a    931:      lsr
854a 4a    932:      lsr
854b 205185 933:      jsr  .a
854e 68    934:      pla
854f 290f  935:      and  #$0f
8551 c90a  936: .a:      cmp  #$0a
8553 b004  937:      bcs  .b
8555 0930  938:      ora  #$30
8557 d002  939:      bne  .c
8559 6936  940: .b:      adc  #$36
855b 208985 941: .c:      jsr  ECHO
855e 60    942:      rts
855f      943: ;
855f      944:
855f      945: pr_byte:
855f a200  946:      ldx  #$00
8561      947: pr_card:
8561 85d4  948:      sta  fr0
8563 86d5  949:      stx  fr0+1
8565 a900  950:      lda  #$00
8567 85f2  951:      sta  cix
8569 20aad9 952:      jsr  ifp
856c 20e6d8 953:      jsr  fasc
856f a000  954:      ldy  #$00

```

```

8571      955: .aprbloop:
8571 b1f3 956:      lda  (inbuff),y
8573 08 957:      php
8574 297f 958:      and  #$7f
8576 c8 959:      iny
8577 208985 960:      jsr  echo
857a 28 961:      plp
857b 10f4 962:      bpl  .aprbloop
857d 60 963:      rts
857e      964:
857e      965: ;
857e      966: ;   Input character with escape processing
857e      967: ;   -----
857e      968: ;
857e 200b86 969: INCH:      JSR  GET_KEY
8581 c91b 970:      CMP  #27
8583 f001 971:      BEQ  EXI
8585 60 972:      RTS
8586 4c1880 973: EXI:      JMP  RETURN
8589      974: ;
8589      975: ;
8589      976: ;
8589      977: ;=====
8589      978: ;   print/input routines
8589      979: ;=====
8589      980: ;
8589      981: ;   Print character
8589      982: ;   -----
8589      983: ; in:
8589      984: ;   A   = character to print
8589      985: ; out:
8589      986: ;   all registers preserved
8589      987: ;
8589 20d485 988: ECHO:      JSR  SAVER
858c 209285 989:      JSR  ZOUT
858f 4c0486 990:      JMP  RESALL
8592      991: ;
8592 8d9e85 992: ZOUT:      STA  ZTEMP+1
8595 ad4703 993:      LDA  $0347
8598 48 994:      PHA
8599 ad4603 995:      LDA  $0346
859c 48 996:      PHA
859d a900 997: ZTEMP:     LDA  #0
859f a200 998:      LDX  #0      ;1   ; force NO
85a1 60 999:      RTS
85a2      1000: ;
85a2      1001: ;
85a2      1002: ;   Print text
85a2      1003: ;   -----
85a2      1004: ; out:
85a2      1005: ;   all registers preserved
85a2      1006: ; notes:
85a2      1007: ;   This print routine will print all characters

```

```

85a2    1008: ;    following the JSR PRINT until a delimiter of
85a2    1009: ;    -1 ($FF) is reached. PRINT will return to the
85a2    1010: ;    point one byte beyond the delimiter.
85a2    1011: ;
85a2 20d485 1012: prints:    JSR    SAVER
85a5 ba    1013:            TSX
85a6 bd0501 1014:            LDA    $0105,X
85a9 8db585 1015:            STA    ZOCH+1
85ac bd0601 1016:            LDA    $0106,X
85af 8db685 1017:            STA    ZOCH+2
85b2    1018: ;
85b2 a001 1019:            LDY    #1
85b4 b9ffff 1020: ZOCH:    LDA    $FFFF,Y
85b7 c9ff 1021:            CMP    #$FF
85b9 f006 1022:            BEQ    ZECHO
85bb 208985 1023:            JSR    ECHO
85be c8    1024:            INY
85bf d0f3 1025:            BNE    ZOCH
85c1 98    1026: ZECHO:    TYA
85c2 18    1027:            CLC
85c3 7d0501 1028:            ADC    $0105,X
85c6 9d0501 1029:            STA    $0105,X
85c9 bd0601 1030:            LDA    $0106,X
85cc 6900 1031:            ADC    #0
85ce 9d0601 1032:            STA    $0106,X
85d1 4c0486 1033:            JMP    RESALL
85d4    1034: ;
85d4    1035: ;
85d4    1036: ;    SAVE & RESTORE registers
85d4    1037: ;    -----
85d4    1038: ;
85d4 08    1039: SAVER:    PHP
85d5 48    1040:            PHA
85d6 48    1041:            PHA
85d7 48    1042:            PHA
85d8 08    1043:            PHP
85d9 48    1044:            PHA
85da 8a    1045:            TXA
85db 48    1046:            PHA
85dc ba    1047:            TSX
85dd bd0901 1048:            LDA    $0109,X
85e0 9d0501 1049:            STA    $0105,X
85e3 bd0701 1050:            LDA    $0107,X
85e6 9d0901 1051:            STA    $0109,X
85e9 bd0101 1052:            LDA    $0101,X
85ec 9d0701 1053:            STA    $0107,X
85ef bd0801 1054:            LDA    $0108,X
85f2 9d0401 1055:            STA    $0104,X
85f5 bd0601 1056:            LDA    $0106,X
85f8 9d0801 1057:            STA    $0108,X
85fb 98    1058:            TYA
85fc 9d0601 1059:            STA    $0106,X
85ff 68    1060:            PLA

```

```

8600 aa 1061:      TAX
8601 68 1062:      PLA
8602 28 1063:      PLP
8603 60 1064:      RTS
8604      1065: ;
8604 68 1066: RESALL:      PLA
8605 a8 1067:      TAY
8606 68 1068:      PLA
8607 aa 1069:      TAX
8608 68 1070:      PLA
8609 28 1071:      PLP
860a 60 1072:      RTS
860b      1073: ;
860b      1074: ;
860b      1075: ;
860b      1076: ;   Get character
860b      1077: ;   -----
860b      1078: ; out:
860b      1079: ;   A   = character (all others preserved)
860b      1080: ;
860b 201186 1081: GET_KEY: JSR  ZGETCH
860e a900 1082: ZCH:      LDA  #0
8610 60 1083:      RTS
8611      1084: ;
8611 20d485 1085: ZGETCH: JSR  SAVER
8614 201d86 1086:      JSR  ZPHG
8617 8d0f86 1087:      STA  ZCH+1
861a 4c0486 1088:      JMP  RESALL
861d      1089: ;
861d ad25e4 1090: ZPHG:   LDA  $E425
8620 48 1091:      PHA
8621 ad24e4 1092:      LDA  $E424
8624 48 1093:      PHA
8625 60 1094:      RTS
8626      1095: ;
8626      1096: ;
8626      1097: ;-----
8626      1098: ;
8626      1099: ;
8626      1100: ;   Get line
8626      1101: ;   -----
8626      1102: ; in:
8626      1103: ;   XA   = buffer address
8626      1104: ;   Y    = maximum number of characters
8626      1105: ; out:
8626      1106: ;   @XA  = data, return ends input, a backspace wor
8626      1107: ;   all registers preserved
8626      1108: ; notes:
8626      1109: ;   This routine will first clear the input window
8626      1110: ;
8626 20d485 1111: GETLINE: JSR  SAVER
8629 8c7086 1112:      STY  ZCNT
862c 8d6d86 1113:      STA  ZSTOR+1

```

```

862f 8e6e86 1114:      STX  ZSTOR+2
8632      1115: ;
8632 a000 1116:      LDY  #0
8634 a920 1117:      LDA  #$20
8636 206c86 1118: ZCLX:   JSR  ZSTOR
8639 c8 1119:      INY
863a cc7086 1120:      CPY  ZCNT
863d d0f7 1121:      BNE  ZCLX
863f      1122: ;
863f a000 1123:      LDY  #0
8641 200b86 1124: ZINLP:  JSR  GET_KEY
8644 c97e 1125:      CMP  #$7E
8646 f012 1126:      BEQ  ZBS
8648 c99b 1127:      CMP  #$9B
864a f01d 1128:      BEQ  ZENDZ
864c cc7086 1129:      CPY  ZCNT
864f f0f0 1130:      BEQ  ZINLP
8651 208985 1131:      JSR  ECHO
8654 206c86 1132:      JSR  ZSTOR
8657 c8 1133:      INY
8658 d0e7 1134:      BNE  ZINLP
865a      1135: ;
865a c000 1136: ZBS:    CPY  #0
865c f0e3 1137:      BEQ  ZINLP
865e 208985 1138:      JSR  ECHO
8661 88 1139:      DEY
8662 a920 1140:      LDA  #$20
8664 206c86 1141:      JSR  ZSTOR
8667 d0d8 1142:      BNE  ZINLP
8669 4c0486 1143: ZENDZ:  JMP  RESALL
866c      1144: ;
866c 99ffff 1145: ZSTOR:  STA  $FFFF,Y
866f 60 1146:      RTS
8670      1147:
8670      1148:
8670 00 1149: ZCNT:   dc.b  0
8671      1150: ;
8671      1151: ;
8671      1152: ;
8671      1153: ;=====
8671      1154: ;   write/calculate diskette data
8671      1155: ;=====
8671      1156: ;   Write all data to disk
8671      1157: ;   -----
8671 a201 1158: WRITE:  LDX  #1           ; W
8673 8e0a03 1159:      STX  SECTOR
8676 ca 1160:      DEX
8677 8e0b03 1161:      STX  SECTOR+1       ; b
867a      1162: ;
867a a243 1163:      LDX  #high BOOTI
867c a00b 1164:      LDY  #low BOOTI
867e a903 1165:      LDA  #3
8680 201487 1166:      JSR  WRBLOCK       ; w

```

```

8683      1167: ;
8683 ac1a43 1168:      LDY  BOOTI+BMD+SDQBM      ;#
8686 8ce788 1169:      STY  TEMPQX
8689      1170: ;
8689 ade688 1171:      LDA  DATASEC      ;Z
868c 4a    1172:      LSR
868d 4a    1173:      LSR
868e 4a    1174:      LSR
868f aa    1175:      TAX
8690 8ee888 1176:      STX  TEMPQX+1
8693 ade688 1177:      LDA  DATASEC
8696 2907  1178:      AND  #7
8698 a8    1179:      TAY
8699 b98787 1180:      LDA  MASK,Y
869c 49ff  1181:      EOR  #$FF
869e      1182: ;
869e 9dee89 1183: CLFIR:  STA  BITMAP,X
86a1 a900  1184:      LDA  #0
86a3 ca    1185:      DEX
86a4 10f8  1186:      BPL  CLFIR
86a6      1187: ;
86a6 aee888 1188:      LDX  TEMPQX+1
86a9 e8    1189:      INX
86aa 2c    1190:      dc.b $2C
86ab      1191: ;
86ab a200  1192: NXB:    LDX  #0      ;c
86ad a9ff  1193:      LDA  #$FF
86af 9dee89 1194: SEA:    STA  BITMAP,X
86b2 e8    1195:      INX
86b3 d0fa  1196:      BNE  SEA
86b5      1197: ;
86b5 ace788 1198:      LDY  TEMPQX      ;i
86b8 88    1199:      DEY
86b9 d011  1200:      BNE  WRBITX
86bb      1201: ;
86bb aeea88 1202:      LDX  NIBOFF      ;o
86be bd8787 1203:      LDA  MASK,X
86c1 aee988 1204:      LDX  BYTOFF
86c4 9dee89 1205: CLREST:  STA  BITMAP,X
86c7 a900  1206:      LDA  #0
86c9 e8    1207:      INX
86ca d0f8  1208:      BNE  CLREST
86cc      1209: ;
86cc a901  1210: WRBITX:  LDA  #1
86ce a289  1211:      LDX  #high BITMAP
86d0 a0ee  1212:      LDY  #low BITMAP
86d2 201487 1213:      JSR  WRBLOCK
86d5 cee788 1214:      DEC  TEMPQX
86d8 d0d1  1215:      BNE  NXB
86da      1216: ;
86da a900  1217:      LDA  #0
86dc aa    1218:      TAX
86dd 9dee8a 1219: CLMAPS:  STA  SCMAP,X

```



```

86e0 9dee8b 1220:      STA  DATSEC,X
86e3 e8      1221:      INX
86e4 d0f7 1222:      BNE  CLMAPS
86e6      1223: ;
86e6 ae1443 1224:      LDX  BOOTI+MDD          ; S
86e9 e8      1225:      INX
86ea 8ef28a 1226:      STX  SCMAP+4          ; P
86ed a901 1227:      LDA  #1
86ef a28a 1228:      LDX  #high SCMAP      ; W
86f1 a0ee 1229:      LDY  #low SCMAP
86f3 201487 1230:      JSR  WRBLOCK
86f6      1231: ;
86f6 a20a 1232:      LDX  #11-1          ; P
86f8 bd9087 1233: MVNAM:  LDA  NAMTAB,X
86fb 9df48b 1234:      STA  DATSEC+DEFNAM,X
86fe ca      1235:      DEX
86ff 10f7 1236:      BPL  MVNAM
8701 a928 1237:      LDA  #DESSUB+DESUSE
8703 8dee8b 1238:      STA  DATSEC+DESTA
8706 a917 1239:      LDA  #DDEX
8708 8df18b 1240:      STA  DATSEC+DENBYT
870b a901 1241:      LDA  #1
870d a28b 1242:      LDX  #high DATSEC
870f a0ee 1243:      LDY  #low DATSEC
8711 4c1487 1244:      JMP  WRBLOCK          ; W
8714      1245: ;
8714      1246: ;
8714      1247: ;   Write a block of sectors
8714      1248: ;   -----
8714      1249: ;
8714 8e0503 1250: WRBLOCK: STX  DBUFHI          ; a
8717 8deb88 1251:      STA  NUMSECS          ; n
871a 8c0403 1252:      STY  DBUFLO
871d      1253: ;
871d ad2188 1254:      LDA  DENSITY          ; g
8720 ae0b03 1255:      LDX  SECTOR+1
8723 d009 1256:      BNE  ISTHD
8725 ae0a03 1257:      LDX  SECTOR
8728 e004 1258:      CPX  #4
872a b002 1259:      BCS  ISTHD
872c a980 1260:      LDA  #$80
872e 8d0803 1261: ISTHD:  STA  DBYTLO
8731 a200 1262:      LDX  #0
8733 a8      1263:      TAY
8734 d002 1264:      BNE  ISSNG
8736 a201 1265:      LDX  #1
8738 8e0903 1266: ISSNG:  STX  DBYTHI
873b      1267: ;
873b a950 1268: REPOUT: LDA  #'P'
873d 8d0203 1269:      STA  DCOMND
8740 a980 1270:      LDA  #$80
8742 8d0303 1271:      STA  DSTATS
8745 201580 1272:      JSR  XSIO

```

```

8748 101c 1273:      BPL  OKWRTX
874a 20a285 1274:     JSR  printsi
874d 457272 1275:     dc.b  "Error writing sector", $9b, -1
8763 4c1880 1276:     JMP  RETURN
8766      1277: ;
8766 ee0a03 1278: OKWRTX:      INC  SECTOR
8769 d003 1279:      BNE  SK1
876b ee0b03 1280:      INC  SECTOR+1
876e      1281: ;
876e 18 1282: SK1:      CLC
876f ad0403 1283:      LDA  DBUFLO
8772 6d0803 1284:      ADC  DBYTLO
8775 8d0403 1285:      STA  DBUFLO
8778 ad0503 1286:      LDA  DBUFHI
877b 6d0903 1287:      ADC  DBYTHI
877e 8d0503 1288:      STA  DBUFHI
8781      1289: ;
8781 ceeb88 1290:      DEC  NUMSECS
8784 d0b5 1291:      BNE  REPOUT
8786 60 1292:      RTS
8787      1293: ;
8787      1294: ;
8787 0080c0 1295: MASK:      dc.b  $00,$80,$C0,$E0
878b f0f8fc 1296:      dc.b  $F0,$F8,$FC,$FE,$FF
8790      1297: ;
8790 4d4149 1298: NAMTAB:      dc.b  "MAIN  "
879b      1299: ;
879b      1300: ;
879b      1301: ;      Calculate all sector positions for given density
879b      1302: ;      -----
879b      1303: ;
879b ad2388 1304: RECALC:      LDA  MAXSEC
879e 8de788 1305:      STA  TEMPQX
87a1 ad2488 1306:      LDA  MAXSEC+1
87a4 8de888 1307:      STA  TEMPQX+1
87a7      1308: ;
87a7 a900 1309:      LDA  #0
87a9 a203 1310:      LDX  #3                      ; d
87ab      1311: REPROL:
87ab 4ee888 1312:      LSR  tempqx+1
87ae 6ee788 1313:      ROR  TEMPQX
87b1 6a 1314:      ROR
87b2 ca 1315:      DEX
87b3 d0f6 1316:      BNE  REPROL
87b5 4a 1317:      LSR
87b6 4a 1318:      LSR
87b7 4a 1319:      LSR
87b8 4a 1320:      LSR
87b9 4a 1321:      LSR
87ba 8dea88 1322:      STA  NIBOFF                      ; O
87bd      1323: ;
87bd ade788 1324:      LDA  TEMPQX
87c0 aee888 1325:      LDX  TEMPQX+1                      ; G

```

87c3		1326:	;			
87c3	2c2188	1327:		BIT	DENSITY	
87c6	1007	1328:		BPL	G2	
87c8	0a	1329:		ASL		
87c9	48	1330:		PHA		
87ca	8a	1331:		TXA		
87cb	2a	1332:		ROL		
87cc	aa	1333:		TAX		
87cd	68	1334:		PLA		
87ce	4a	1335:		LSR		
87cf		1336:	;			
87cf	8de988	1337:	G2:	STA	BYTOFF	
87d2	e8	1338:		INX		
87d3	8e1a43	1339:		STX	BOOTI+BMD+SDQBM	; #
87d6	8a	1340:		TXA		
87d7	18	1341:		CLC		
87d8	6904	1342:		ADC	#4	; A
87da	8d1443	1343:		STA	BOOTI+MDD	;
87dd	6902	1344:		ADC	#2	; G
87df	8de688	1345:		STA	DATASEC	
87e2	8d1f43	1346:		STA	BOOTI+BMD+SDDIR	; F
87e5		1347:				
87e5		1348:	;			
87e5		1349:	;			
87e5		1350:	>>>>>>>>>>>>>>>>>>>>>>Test code			
87e5		1351:	walter:			
87e5		1352:	;	ADC	#30	; l
87e5	6946	1353:		ADC	#70	; l
87e7		1354:				
87e7	8d1d43	1355:		STA	BOOTI+BMD+SDALLOC	; F
87ea	2c2c88	1356:		BIT	CURP	; c
87ed	1002	1357:		BPL	ISSOMD	
87ef	a900	1358:		LDA	#0	; z
87f1	29f8	1359:	ISSOMD:	AND	#\$F8	
87f3		1360:	;			
87f3	8d3343	1361:		STA	BOOTI+EXTR+DDMAP	; S
87f6		1362:	;			
87f6	38	1363:		SEC		
87f7	ad2388	1364:		LDA	MAXSEC	; #
87fa	8d1643	1365:		STA	BOOTI+BMD+SDTOT	
87fd	ede688	1366:		SBC	DATASEC	
8800	8d1843	1367:		STA	BOOTI+BMD+SDFRE	
8803	ad2488	1368:		LDA	MAXSEC+1	
8806	8d1743	1369:		STA	BOOTI+BMD+SDTOT+1	
8809	e900	1370:		SBC	#0	
880b	8d1943	1371:		STA	BOOTI+BMD+SDFRE+1	
880e	ad2188	1372:		LDA	DENSITY	; s
8811	8d2a43	1373:		STA	BOOTI+DPD+SPDEN	
8814	ad2788	1374:		LDA	TRACKS	
8817	8d2943	1375:		STA	BOOTI+DPD+SPTRK	
881a		1376:	;			
881a	ad0ad2	1377:		LDA	\$D20A	; g
881d	8d3243	1378:		STA	BOOTI+EXTR+DRAND	

```

8820 60 1379:      RTS
8821    1380: ;
8821    1381: ;
8821    1382:
8821    1383: DENSITY:      ds.b 1      ; bytes per
8822    1384: SECTORS:      ds.b 1      ; number of
8823    1385: MAXSEC:       ds.b 2      ; total num
8825    1386: DENS:        ds.b 1      ; drive typ
8826    1387: ;
8826    1388: INCH8:       ds.b 1      ; eight inc
8827    1389: TRACKS:      ds.b 1      ; number of
8828    1390: TRKINX:       ds.b 1      ; index int
8829    1391: DENINX:       ds.b 1      ; density i
882a    1392: ;
882a    1393: COUNT:       ds.b 1      ; general c
882b    1394: STACKP:      ds.b 1      ; return st
882c    1395: CURP:        ds.b 1      ; current D
882d    1396: DOSP:        ds.b 1      ; current #
882e    1397: DOSPT:       ds.b 10     ; space for
8838    1398: DIRP:        ds.b 1      ; current p
8839    1399: NAMES:       ds.b 2      ; ds.b 130
883b    1400: total.num.sec: ds.b 2
883d    1401:
883d    1402:            ds.b 126
88bb    1403:
88bb    1404: ENTRY:       ds.b 25     ; area for
88d4    1405: DOSI:        ds.b 3+8+5  ; room for
88e4    1406: US:         ds.b 1
88e5    1407: HIGHSP:      ds.b 1
88e6    1408: ;
88e6    1409: DATASEC:    ds.b 1      ; Beginning
88e7    1410: TEMPQX:      ds.b 2
88e9    1411: BYTOFF:      ds.b 1
88ea    1412: NIBOFF:      ds.b 1
88eb    1413: NUMSECS:     ds.b 1      ; Number se
88ec    1414: TABLEN:      ds.b 2      ; length of
88ee    1415: ;
88ee    1416: FMTBUF:      ds.b 256    ; buffer fo
89ee    1417: BITMAP:      ds.b 256    ; bitmap bu
8aee    1418: SCMAP:       ds.b 256    ; sector ma
8bee    1419: DATSEC:      ds.b 256    ; first dir
8cee    1420: TABLE:     ds.b 256    ; area to l
8dee    1421: ;
8dee    1422: ;
8dee    1423: ;
8dee    1424: ;~~~~~
8dee    1425: ; Ok the relocation marker is to tell the DOS we are now u
8dee    1426: ;more data at memlow..
8dee    1427: ;
8dee    1428: ;
8dee 00 1429: initz:      dc.b 0      ; start address
8def    1430:
8def    1431:

```

```

8def 00 1432: zend:          dc.b  0    ;end address
8df0 1433:
8df0 1434: ;
8df0 1435: ;~~~~~
8df0 1436: ;
8df0 1437: ;          *** END OF HANDLER ***
8df0 1438: ;~~~~~
8df0 1439: ;
8df0 1440: ; The rest of this relocater Crap Took me a week to f
8df0 1441: ;
8df0 1442: ; Relocater: main entry
8df0 1443: ; -----
8df0 1444: ; in:
8df0 1445: ; segtab = table of segment descriptors
8df0 1446: ; +00 = relocater table address
8df0 1447: ; +02 = originate address of block
8df0 1448: ; +04 = destination originate of block
8df0 1449: ; +06 = address of block
8df0 1450: ; +08 = number of bytes in block
8df0 1451: ; +10 = destination address of block
8df0 1452: ; +12 = Length of segment descriptor
8df0 1453: ; +14 = this is the word location adjust mem
8df0 1454:
8df0 1455: ;
8df0 1456: ;
8df0 1457: ;
8df0 1458: ;
8df0 1459: ; reltab = relocater table
8df0 1460: ; list of address of words to adjust 2
8df0 1461: ; list of address low bytes to adjust 2
8df0 1462: ; list of address high bytes to adjust 3
8df0 1463: ; followed by their low bytes
8df0 1464: ;~~~~~
8df0 1465: rlocate:
8df0 1466:
8df0 ad238f 1467: lda segtab+segaddress ; s
8df3 8d198f 1468: sta segtab+dorgadr ; P
8df6 8d1f8f 1469: sta segtab+blkdes ; P
8df9 ad248f 1470: lda segtab+segaddress+1 ; s
8dfc 8d1a8f 1471: sta segtab+dorgadr+1 ; P
8dff 8d208f 1472: sta segtab+blkdes+1 ; P
8e02 1473: ; jsr rlocate ; o
8e02 1474:
8e02 1475:
8e02 1476:
8e02 a900 1477: lda #0 ; s
8e04 8dcf8e 1478: sta segment
8e07 1479:
8e07 aecf8e 1480: segloop: ldx segment
8e0a bd158f 1481: lda segtab+rettad,x ; g
8e0d 8db58e 1482: sta relget+1
8e10 bd168f 1483: lda segtab+rettad+1,x
8e13 8db68e 1484: sta relget+2

```

```

8e16 0db58e 1485:      ora  relget+1          ; i
8e19 d001  1486:      bne  havseg
8e1b 60    1487:      rts
8e1c      1488:
8e1c 38    1489: havseg:      sec
8e1d bd198f 1490:      lda  segtab+dorgadr,x
8e20 fd178f 1491:      sbc  segtab+orgadr,x
8e23 8dd08e 1492:      sta  zoffset
8e26 bd1a8f 1493:      lda  segtab+dorgadr+1,x
8e29 fd188f 1494:      sbc  segtab+orgadr+1,x
8e2c 8dd18e 1495:      sta  zoffset+1
8e2f      1496:
8e2f 20c08e 1497: zwordlp:  jsr  getzwp
8e32 f013  1498:      beq  zlbytlp          ; i
8e34      1499:
8e34 b1d7  1500:      lda  (zwptr),y        ; a
8e36 18    1501:      clc
8e37 6dd08e 1502:      adc  zoffset
8e3a 91d7  1503:      sta  (zwptr),y
8e3c c8    1504:      iny
8e3d b1d7  1505:      lda  (zwptr),y
8e3f 6dd18e 1506:      adc  zoffset+1
8e42 91d7  1507:      sta  (zwptr),y
8e44 4c2f8e 1508:      jmp  zwordlp
8e47      1509:
8e47 20c08e 1510: zlbytlp:  jsr  getzwp          ; g
8e4a f00b  1511:      beq  zhbytlp          ; i
8e4c      1512:
8e4c b1d7  1513:      lda  (zwptr),y
8e4e 18    1514:      clc
8e4f 6dd08e 1515:      adc  zoffset
8e52 91d7  1516:      sta  (zwptr),y
8e54 4c478e 1517:      jmp  zlbytlp
8e57      1518:
8e57 20c08e 1519: zhbytlp:  jsr  getzwp          ; g
8e5a f012  1520:      beq  zmovlp          ; j
8e5c      1521:
8e5c 20b48e 1522:      jsr  relget          ; g
8e5f 18    1523:      clc
8e60 6dd08e 1524:      adc  zoffset
8e63 b1d7  1525:      lda  (zwptr),y
8e65 6dd18e 1526:      adc  zoffset+1
8e68 91d7  1527:      sta  (zwptr),y
8e6a 4c578e 1528:      jmp  zhbytlp
8e6d 60    1529:      rts
8e6e      1530:
8e6e      1531: zmovlp:
8e6e      1532:
8e6e adde8e 1533:      lda  segment+segmove    ; g
8e71 c9ff  1534:      cmp  #seg_off          ; $
8e73 d001  1535:      bne  .zmovlp          ; m
8e75 60    1536:      rts                  ; o
8e76 aecf8e 1537: .zmovlp:  ldx  segment          ; g

```

```

8e79 bd1b8f 1538:      lda  segtab+blkadr,x
8e7c 8d978e 1539:      sta  zmovfr+1
8e7f bd1c8f 1540:      lda  segtab+blkadr+1,x
8e82 8d988e 1541:      sta  zmovfr+2
8e85      1542:
8e85 bd1f8f 1543:      lda  segtab+blkdes,x
8e88 8d9a8e 1544:      sta  zmovto+1
8e8b bd208f 1545:      lda  segtab+blkdes+1,x
8e8e 8d9b8e 1546:      sta  zmovto+2
8e91      1547:
8e91      1548:
8e91 bc1e8f 1549:      ldy  segtab+blkbyt+1,x
8e94      1550:
8e94 a200 1551:      ldx  #0
8e96 bdfbff 1552: zmovfr:      lda  $ffff,x
8e99 9dffff 1553: zmovto:      sta  $ffff,x
8e9c e8 1554:      inx
8e9d d0f7 1555:      bne  zmovfr
8e9f ee988e 1556:      inc  zmovfr+2
8ea2 ee9b8e 1557:      inc  zmovto+2
8ea5 88 1558:      dey
8ea6 10ee 1559:      bpl  zmovfr          ; m
8ea8      1560:          ;
8ea8 adcf8e 1561:      lda  segment
8eab 18 1562:      clc
8eac 690c 1563:      adc  #seglen          ; g
8eae 8dcf8e 1564:      sta  segment
8eb1 4c078e 1565:      jmp  segloop
8eb4      1566:
8eb4      1567:
8eb4 adffff 1568: relget:      lda  $ffff
8eb7 eeb58e 1569:      inc  relget+1
8eba d003 1570:      bne  nc1
8ebc eeb68e 1571:      inc  relget+2
8ebf 60 1572: nc1:      rts
8ec0      1573:
8ec0 20b48e 1574: getzwp:      jsr  relget
8ec3 85d7 1575:      sta  zwptr          ; s
8ec5 20b48e 1576:      jsr  relget
8ec8 a000 1577:      ldy  #0
8eca 85d8 1578:      sta  zwptr+1
8ecc 05d7 1579:      ora  zwptr          ; c
8ece 60 1580:      rts
8ecf      1581:
8ecf 00 1582: segment:      dc.b  0          ; c
8ed0 0000 1583: zoffset:      dc.w  0          ; o
8ed2      1584: ;
8ed2      1585: ;~~~~~
8ed2      1586: ;
8ed2      1587: ;      Relocation data table
8ed2      1588: ;      -----
8ed2      1589: ;      We resolve Word locations First!
8ed2      1590: ;

```

```

8ed2 1591: ; Things like Lda $5000
8ed2 1592: ; Sta $5000
8ed2 1593: ; lda $5000,y
8ed2 1594: ; jsr $5000
8ed2 1595: ; jmp $5000
8ed2 1596: ;
8ed2 1597: ; Well you get the Idea..
8ed2 1598: ;
8ed2 1599: ;~~~~~
8ed2 1600: ;
8ed2 1601: ; $3000 tabel
8ed2 1602: ;
8ed2 1603: rtable1:
8ed2 1243 1604: dc.w w_f01+1
8ed4 4a43 1605: dc.w w_f02+1
8ed6 5043 1606: dc.w w_f03+1
8ed8 5a43 1607: dc.w w_f04+1
8eda 6043 1608: dc.w w_f05+1
8edc 6643 1609: dc.w w_f06+1
8ede 6f43 1610: dc.w w_f07+1
8ee0 8e43 1611: dc.w w_f08+1
8ee2 9e43 1612: dc.w w_f09+1
8ee4 a343 1613: dc.w w_f10+1
8ee6 1614:
8ee6 b643 1615: dc.w w_f11+1
8ee8 bb43 1616: dc.w w_f12+1
8eea c443 1617: dc.w w_f13+1
8eec cc43 1618: dc.w w_f14+1
8eee d743 1619: dc.w w_f15+1
8ef0 da43 1620: dc.w w_f16+1
8ef2 dd43 1621: dc.w w_f17+1
8ef4 fa43 1622: dc.w w_f18+1
8ef6 1e44 1623: dc.w w_f19+1
8ef8 2944 1624: dc.w w_f20+1
8efa 1625:
8efa 3044 1626: dc.w w_f21+1
8efc 3d44 1627: dc.w w_f22+1
8efe 4b44 1628: dc.w w_f23+1
8f00 6c44 1629: dc.w w_f24+1
8f02 7744 1630: dc.w w_f25+1
8f04 1631:
8f04 1632:
8f04 0000 1633: dc.w 0 ; ok done w
8f06 1634: ;
8f06 1635: ;
8f06 1636: ;Ok these are low Byte Locations.. only one byte changed!
8f06 1637: ; OK Resolving Low Byte's First
8f06 1638: ;
8f06 ac43 1639: dc.w w_lb_1+1
8f08 e343 1640: dc.w w_lb_2+1
8f0a 0000 1641: dc.w 0 ; ok done w
8f0c 1642: ;~~~~~
8f0c 1643: ;Now to resolve High Bytes..... Only one byte to change!

```



```

8f0c      1644: ; Here is what is needed... location of high byte in .wor
8f0c      1645: ; Second thing needed is the low byte of the address you a
8f0c      1646: ;to adjust!
8f0c      1647: ;
8f0c b143 1648:      dc.w  w_hb_1+1
8f0e 83   1649:      dc.b  LRTS
8f0f e543 1650:      dc.w  w_hb_2+1
8f11 3b   1651:      dc.b  NODMSG
8f12      1652:
8f12 0000 1653:      dc.w  0          ; ok we are
8f14 00   1654:      dc.b  0          ; ok we are
8f15      1655:
8f15      1656:
8f15      1657:
8f15      1658: ;~~~~~
8f15      1659: ;End of relocation Table!!!
8f15      1660: ;master table:
8f15      1661: ;
8f15      1662: segtab:
8f15 d28e 1663:      dc.w  rtable1    ; address of reloc
8f17 0b43 1664:      dc.w  initz1     ; assembly start a
8f19 0000 1665:      dc.w  0          ; relocate to addre
8f1b 0b43 1666:      dc.w  initz1     ; absolute start a
8f1d 8001 1667:      dc.w  zend1-initz1 ; size of block
8f1f 0000 1668:      dc.w  0          ; destination of bl
8f21 0000 1669:      dc.w  0          ; END of segments
8f23 0030 1670:      dc.w  $3000     ;segaddress memory
8f25 ff   1671:      dc.b  seg_off    ; -1 meand do not m
8f26      1672:
8f26      1673:
8f26      1674: win_end:  ds.b  0
8f26      1675: ;
8f26      1676: ;
8f26      1677:
8f26      1678:
8f26      1679: ;

```

End assembly: no errors