



```

0000 1: ;-----
0000 2: ;
0000 3: ; Copyright 2012 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: ; title 'Black box dir builder'
0000 10: ; This program looks at your partition table. Loads the pe
0000 11: ; you select and then builds the directory with the name i
0000 12: ; table.
0000 13: ;
0000 14: ;-----
0000 15: ; dirbuild.s
0000 16: ;
0000 17: ; -----
0000 18: ;-----
0000 19: ; Notes: o This source code MAY NOT be placed for download
0000 20: ; o "mea" is a macro that loads the address of the
0000 21: ; into the pointer specified by the second field.
0000 22: ;-----
0000 23: ; Assembler: MADMAC (tm) ST Cross Assembler (Atari Corp)
0000 24: ; XASM st and IBM VERSIONS
0000 25: ;-----
0000 26: ; macros
0000 27: mea: .macro field,ptr
0000 28: lda #low %1
0000 29: sta %2
0000 30: lda #high %1
0000 31: sta %2 + 1
0000 32: .endm
0000 33:
0000 34: my_companyid: .macro
0000 35: dc.b 24,"Integrated Logic System
0000 36: .endm
0000 37:
0000 38: ; this can be the name of a company or user
0000 39: Author_name: .macro ; always 31 bytes
0000 40: dc.b 17
0000 41: dc.b "Stephen J. Carden"
0000 42: ds.b 13
0000 43: .endm
0000 44:
0000 45: register_user: .macro ; always 31 bytes
0000 46: ds.b 31
0000 47: .endm
0000 48: ;=====
0000 49: ; Relocator Vars
0000 50: ;
0000 51: ;
0000 52: rettad: equ 0 ; r
0000 53: orgadr: equ 2 ; o

```

```

0000 54: dorgadr:          equ  4      ; d
0000 55: blkadr:             equ  6      ; a
0000 56: blkbyt:             equ  8      ; b
0000 57: blkdes:             equ 10      ; d
0000 58: seglen:             equ 12      ; L
0000 59:
0000 60: segaddress:         equ 14      ; t
0000 61: segmove:           equ 15      ; t
0000 62:
0000 63: seg_on:             equ  0      ; d
0000 64: seg_off:            equ $ff     ; d
0000 65:
0000 66: t_off1:              equ Split-$3000
0000 67:
0000 68: ;-----
0000 69: ;LSIO                [COMTAB-10]
0000 70: ;This location contains the address of the SIO routine Spa
0000 71: ;This is actually a vector, so U may replace this address
0000 72: ;The ramdisk patches in here to trap access to the drive i
0000 73: ;Many commands use this vector to run the dos's high speed
0000 74: LSIO:                equ 10
0000 75: ;-----
0000 76: ;ECHOFLG             [COMTAB-8]
0000 77: ;This location contains the index into HATABS <table of ha
0000 78: ;addresses> of the file SpartaDos is echoing output to. A
0000 79: ;indicates that echoing is inactive. This Location Is Vali
0000 80: ;Running Under SpartaDos 2.x
0000 81: ECHOFLG:             equ  8
0000 82: ;-----
0000 83: ;BATFLG              [COMTAB-6]
0000 84: ;This location contains the index into HATABS <table to ha
0000 85: ;addresses> of the file SpartaDos is receiving input from.
0000 86: ;indicates that no batch file is active. This Location Is
0000 87: ;While Running Under SpartaDos 2.x
0000 88: BATFLG:              equ  6
0000 89: ;-----
0000 90: ;WRTCMD              [COMTAB-2]
0000 91: ;This location contains the SIO write command. A W is the
0000 92: ;verify command, & P is the write with no verify command.
0000 93: ;Is Valid Only While Running Under SpartaDos 2.x.
0000 94: WRTCMD:              equ  2
0000 95:
0000 96: ;-----
0000 97: ;WARMST              [COMTAB-1]
0000 98: ;This flag, if set, indicates that the command processor i
0000 99: ;cold start. It is cleared <to 0> whenever the command pro
0000 100: ;entered. It is used to trap errors when trying to open ST
0000 101: ;or AUTORUN.SYS.
0000 102: WARMST:              equ  1
0000 103: ;-----
0000 104: ;COMTAB              [COMTAB]
0000 105: ;This location contains a 6502 jump instruction 2 the comm
0000 106: ;BASIC enters here on a dos command.

```

```

0000 107: comtab:                equ   $0a
0000 108: ;-----
0000 109: ;ZCRNAME                [COMTAB+3]
0000 110: ;This location contains a 6502 jump instruction 2 the file
0000 111: ;routine <CRNAME>. This is used by most external dos comma
0000 112: ;the next filename on the command line. The command line i
0000 113: ;the crunched filename ends up at COMFNAM. This routine su
0000 114: ;default drive number if necessary. The zero flag on retur
0000 115: ;if n filename is on the command line. Each call returns t
0000 116: ;filename on the command line.
0000 117: ZCRNAME:                equ   3
0000 118: ;-----
0000 119: ;ZDIVIO                 [COMTAB+6]
0000 120: ;This location contains the address of the divert input/ou
0000 121: ; of I/O> routine. From an assembly language program, U ma
0000 122: ;through an indirect jump to ZDIVIO with the filename at C
0000 123: ;Y register equal to 0 if output <PRINT>, or 1 if input <-
0000 124: ZDIVIO:                equ   6
0000 125: ;-----
0000 126: ;ZXDIVIO                [COMTAB+8]
0000 127: ;This location contains the address of the stop divert inp
0000 128: ;routine. From an assembly language program, U may call th
0000 129: ;through an indirect jump to ZXDIVIO with the Y register e
0000 130: ;stopping output <PRINT>, or 1 if stopping input <force en
0000 131: ZXDIVIO:                equ   8
0000 132: ;-----
0000 133: ;BUFOFF                 [COMTAB+10]
0000 134: ;This location contains the current offset into the comman
0000 135: ;CRNAME uses this pointer to fetch the next parameter on t
0000 136: ;command line <at LBUF> & move it to COMFNAM.
0000 137: BUFOFF:                equ   10
0000 138:
0000 139: ;-----
0000 140: ;ZORIG                  [COMTAB+11]
0000 141: ;This location contains the start address of SpartaDos. $6
0000 142: ;start address of SPEED.DOS, STANDARD.DOS and $700 is the
0000 143: ;of all other vectors.
0000 144: ZORIG:                equ   11
0000 145: ;-----
0000 146: ;DATER                  [COMTAB+13]
0000 147: ;This location contains the current date in DD/MM/YY forma
0000 148: ;This is the date that sparta inserts in the directory whe
0000 149: ;or directory is created. To override this, see TDOVER.
0000 150: dater:                equ   13
0000 151: ;-----
0000 152: ;TIMER                  [COMTAB+16]
0000 153: ;This location contains the current time in HH/MM/SS forma
0000 154: ;in BCD format, therefore it can be read with no conversio
0000 155: ;This is the time that sparta inserts in the directory whe
0000 156: ;or directory is created. To override this, see TDOVER.
0000 157: timer:                equ   16
0000 158: ;-----
0000 159: ;ODATER                 [COMTAB+19]

```

```

0000 160: ;This location contains the alternate date in DD/MM/YY for
0000 161: ;Sparta uses this date instead of DATER if the TDOVER flag
0000 162: odater:                equ 19
0000 163: ;-----
0000 164: ;OTIMER                [COMTAB+22]
0000 165: ;This location contains the alternate time in HH/MM/SS for
0000 166: ;Sparta uses this time instead of TIMER if the TDOVER flag
0000 167: otimer:                 equ 22
0000 168: ;-----
0000 169: ;TDOVER                [COMTAB+25]
0000 170: ;This location contains the time/date override flag. It is
0000 171: ;to use DATER and TIMER when it creates new files, and set
0000 172: ;use ODATER and OTIMER. This is used by file copy programs
0000 173: ;SPCOPY/MENU> to insure that the time and date of each fil
0000 174: tdover:                 equ 22
0000 175: ;-----
0000 176: ;TRUN                  [COMTAB+26]
0000 177: ;This location contains the RUN address of a load file. Th
0000 178: ;is updated during the internal load operation, so BASIC o
0000 179: ;program may check what the load address was. RUNLOC is up
0000 180: ;location by the command processor only.
0000 181: TRUN:                   equ 26
0000 182: ;-----
0000 183: ;SBUFF                [COMTAB+28]
0000 184: ; This is the start address of Sparta DOS sector buffers.
0000 185: sbuff:                  equ 28
0000 186: ;-----
0000 187: ;DDENT                [COMTAB+29]
0000 188: ;This location contains the density <sector size> of each
0000 189: ;all__sparta 1.x only supports 4 drives.> A value of 0 ind
0000 190: ;sectors, and a 128 indicates 128 byte sectors. This Table
0000 191: ;For Version 1.x sparta. The DDENT table is inaccessible i
0000 192: ;sparta 2.x.
0000 193: DDENT:                  equ 29
0000 194: ;-----
0000 195: ;SMEMLO                [COMTAB+30]
0000 196: ;This is the top of SpartaDOS low memory. Handlers added s
0000 197: ;up the memory between SMEMLO and MEMLO.
0000 198: smemlo:                 equ 30
0000 199: ;-----
0000 200: ;INCOMND              [COMTAB+32]
0000 201: ;A -1 here indicates that we are in the command processor
0000 202: ;commands, etc.). A 0 indicates that we are in BASIC or so
0000 203: ;program. This is used by the initialization routine to de
0000 204: ;to enter the command processor or not.
0000 205: incomnd:                equ 32
0000 206: ;-----
0000 207: ;-----
0000 208: ;COMFNAM              [COMTAB+33]
0000 209: ;This is the buffer for the output of the ZCRNAME routine.
0000 210: ;It is a 28 byte long buffer and ALWAYS begins in the form
0000 211: ;are only looking for parameters, u may start looking at C
0000 212: COMFNAM:                equ 33

```

```

0000 213: ;-----
0000 214: ;RUNLOC          [COMTAB+61]
0000 215: ;This location contains the run address of a .COM file whe
0000 216: ;through the command processor <as a command>. If no addre
0000 217: ;after the RUN command, this is the address too be run.
0000 218: RUNLOC:          equ    61
0000 219: ;-----
0000 220: ;LBUF            [COMTAB+63]
0000 221: ;This location contains the input buffer. This is where th
0000 222: ;stored. LBUF is 64 bytes in length.
0000 223: LBUF:            equ    63    ; lbuf offs
0000 224: ;-----
0000 225: ;
0000 226: ;   READ THIS BEFORE USING VECTORS
0000 227: ;
0000 228: ;SpartaDOS 3.3 and Real.dos contains many vectors pertaini
0000 229: ;setting, reading, displaying of the time and date, execut
0000 230: ;lines, initializing the system and many more. These vecto
0000 231: ;are : contained in the RAM under the operating system sta
0000 232: ; $FFCO. They may be accessed by the following Instruction
0000 233: ;
0000 234: ;           lda $0301    ; PIA
0000 235: ;           pha          ; save old value of
0000 236: ;           and #$FE     ; set bit 0 to OFF
0000 237: ;           sta $0301    ; enable RAM under
0000 238: ;           jsr VGETTO   ; call routing at $
0000 239: ;           pla
0000 240: ;           sta $0301    ; restore the port
0000 241: ;
0000 242: ;These functions each contain a jump (JMP) instruction to
0000 243: ;function. If a function is not initially supported (as in
0000 244: ;will contain a SEC and RTS instruction rather I. than a J
0000 245: ;vectors are currently supported:
0000 246: ;
0000 247: ;
0000 248: ;-----
0000 249: ;euro_time
0000 250: ;this is a one byte location that hold a '50' for pal mach
0000 251: ;this is a one byte location that hold a '60' for ntsc mac
0000 252: euro_time:      equ    $ff91
0000 253: ;-----
0000 254: ;vgettd ($FFCO)
0000 255: ;This function returns the current time/date at TIMER and
0000 256: ;(locations at COMTAB-see page 110 of the Sparta DOS manua
0000 257: ;On return, the carry is SET if the function failed. When
0000 258: ;opened for write, Sparta DOS makes a call here to update
0000 259: ; so it can move this data into the directory entry. Also
0000 260: ; DATE internal commands make calls here to get the curren
0000 261: ; and ZHAND also use this vector.
0000 262: vgettd:        equ    $ffc0
0000 263: ;
0000 264: ;-----
0000 265: ;VSETTD ($FFC3)

```

```

0000 266: ;This function sets the time/date. On entry, the new time
0000 267: ;TIMER and DATER (locations at COMTAB-see page 110 of the
0000 268: ;On return, the carry is SET if the function failed. This
0000 269: ;the commands TIME, DATE, and the :a; ZHAND set functions.
0000 270: vsettd          equ  $ffc3
0000 271:
0000 272: ;-----
0000 273: ;VTDON ($FFC6)
0000 274: ;This function turns the time/date display line on or off.
0000 275: ;the Y register contains 0 if to turn the line off, or 1 i
0000 276: ;line on. On return, the carry is SET if the function fail
0000 277: ;is not supported internally by SpartaDOS. The TDLIN hand
0000 278: ;into this vector for use by the TD command along with the
0000 279: ;38 and 39. (See the TDLIN command.)
0000 280: vtdon:          equ  $ffc6
0000 281:
0000 282: ;-----
0000 283: ;VFMTTD ($FFC9)
0000 284: ;This function returns the formatted time/date line into '
0000 285: ;buffer. On entry, the X and Y registers 1 contain the hig
0000 286: ;the buffer address respectively. On exit, the carry is SE
0000 287: ;failed. This function is not supported internally by Spar
0000 288: ;The TDLIN handler patches into this vector for use by TD
0000 289: ;ZHAND routines. (See the TDLIN command.)
0000 290: VFMTTD:         equ  $FFC9
0000 291:
0000 292: ;-----
0000 293: ;VINITZ ($FFCC)
0000 294: ;This vector is called after SpartaDOS has finished initia
0000 295: ;after a RESET occurs. The command AUTOBAT patches into th
0000 296: ;a batch file right after RESET. (By initialization, we me
0000 297: ;making a call through DOSINI-e.g. JMP (DOSINI). The OS mo
0000 298: ;calls this vector before it enters a cartridge or DOS.)
0000 299: VINITZ:         equ  $FFCC
0000 300:
0000 301: ;-----
0000 302: ;VINITZ2 ($FFCF)
0000 303: ;This vector is called after SpartaDOS has finished initia
0000 304: ;after a NON-RESET occurs. Several SpartaDOS commands (suc
0000 305: ;UNERASE) will initialize DOS before and after they perfor
0000 306: ;They do this by jumping through the DOSINI vector as does
0000 307: ;routine after a RESET.
0000 308: VINITZ2:        equ  $FFCF
0000 309:
0000 310: ;-----
0000 311: ;VXCOMLI ($FFD2)
0000 312: ;This vector calls the command processor to execute a comm
0000 313: ;at LBUF. BUFOFF should be 0 on entry. Any errors that may
0000 314: ;of executing the command line shall be printed as usual.
0000 315: ;printed before or after command execution. This is the me
0000 316: ;DUP.SYS for Sparta DOS could use to perform its functions
0000 317: vxcomli:        equ  $ffd2
0000 318:

```

```

0000 319: ;-----
0000 320: ;VCOMND ($FFD5)
0000 321: ;This vector calls the main command processor program entr
0000 322: ;patch into this vector if you wish to supply your own com
0000 323: ;(The 'I current one simply prints the prompt, inputs a li
0000 324: ;calls VXCOMLI, and jumps back to the beginning.) This is
0000 325: ;uses to gain entry from a DOS command in BASIC.
0000 326: VCOMND:                equ   $FFD5
0000 327: ;-----
0000 328: ;VPRINT ($FFD8)
0000 329: ;This vector points to the Sparta DOS general print routin
0000 330: ;method is:
0000 331: ;
0000 332: ;I use a similar routin  Jsr printsi
0000 333: ;
0000 334: ;                JSR   VPRINT
0000 335: ;                dc.b  "This is a message"
0000 336: ;
0000 337: ;
0000 338: VPRINT:                equ   $FFD8
0000 339: ;-----
0000 340: ;VKEYON ($FFDB)
0000 341: ;This function turns the keyboard buffer on or off. On ent
0000 342: ;contains 0 if to turn the buffer off, or 1 if to turn the
0000 343: ;function is supported internally by SpartaDOS.
0000 344: ;
0000 345: VKEYON:                equ   $FFDB
0000 346: ;
0000 347: ;
0000 348: ;-----
0000 349: ;tsr ($fff5)
0000 350: ;this location registers its presents of relocatable file(
0000 351: ;should keep the relocatable file from running twice, or s
0000 352: ;presents of another file. example wedge uses 4 banks of r
0000 353: ;not want the ramdisk driver allocating thoes banks for th
0000 354: ;
0000 355: Ramdisk:    equ   1      ;done! ramdisk inst
0000 356: tdline:     equ   2      ;done! tdli
0000 357: wedge:      equ   3      ;done! wedg
0000 358: Ram_ex:     equ   4      ;done! Ramd
0000 359: iomon:      equ   5      ;done!
0000 360: APE_HND:    equ   6      ;done!
0000 361: PRC_hnd     equ   7      ;done!
0000 362: RS232:      equ   8      ;done!
0000 363: Rverter:    equ   9      ;done! this is alre
0000 364: MUXTIME:    equ  10      ;done!
0000 365: Turboss:    equ  11      ;done!
0000 366: ;          equ  12
0000 367: ;          equ  13
0000 368: ;          equ  14
0000 369: ;          equ  15
0000 370: ;          equ  16
0000 371: ;          equ  17

```



```

0000 372: ; equ 18
0000 373: ; equ 19
0000 374: ; equ 20
0000 375: ; equ 21
0000 376: ; equ 22
0000 377: dis_bat: equ 23 ;This marks the sys
0000 378: ver.Real: equ 24 ;Done! Sparta.com s
0000 379: prokey: equ 25 ;done!
0000 380: gdevice: equ 26 ;done!
0000 381: dosmenu: equ 27 ;done!
0000 382: hyp_r: equ 28 ;done!
0000 383: shutdown: equ 29 ;done!
0000 384: Ape_present: equ 30 ;done! Ape R Device
0000 385: r_handlr: equ 31 ;
0000 386: p_handlr: equ 32
0000 387: ;
0000 388: tsr: equ $fff5
0000 389:
0000 390: ;-----
0000 391: ;
0000 392: ; Floating point math
0000 393: ifp: equ $d9aa
0000 394: cix: equ $f2
0000 395: inbuff: equ $f3
0000 396: fr0: equ $d4
0000 397: fr1: equ $e0
0000 398: afp: equ $d800
0000 399: fasc: equ $d8e6
0000 400: fpi: equ $d9d2
0000 401: fmove: equ $ddb6
0000 402:
0000 403: fsub: equ $da60
0000 404: fadd: equ $da66
0000 405: fmul: equ $dadb
0000 406: fdiv: equ $db28
0000 407: zfr0: equ $da44
0000 408:
0000 409:
0000 410: Portb: equ $d301
0000 411:
0000 412: ;
0000 413: ;=====
0000 414: ; system declarations
0000 415: ;=====
0000 416: ;
0000 417: ; *****
0000 418: ; * SpartaDOS extrinsic data variables *
0000 419: ; *****
0000 420: ;
0000 421: ;SPTR: equ $43 ; Pointer t
0000 422: ADDR: equ $45 ; Pointer t
0000 423: RSLW: equ $48
0000 424: CENTRY: equ $49

```

```

0000 425: ;
0000 426: ; *****
0000 427: ; * Misc. ATARI system variables *
0000 428: ; *****
0000 429: ;
0000 430: COLDST:          equ  $0244 ; cold star
0000 431: RUNAD:             equ  $02E0 ; binray lo
0000 432: INITAD:           equ  $02E2 ; binary la
0000 433: MEMHI:             equ  $02E5 ; Top of me
0000 434: MEMLO:            equ  $02E7 ; Low memor
0000 435:
0000 436: ;           ; DOS initialization
0000 437: ;
0000 438: ;   DOS init and run vectors
0000 439: ;   -----
0000 440: ;
0000 441: i_WARMST:           equ  $08  ; system in
0000 442: BOOTQ:              equ  $09  ; Successfu
0000 443: DOSINI:             equ  $0C  ; V
0000 444: zwptr:              equ  $D7
0000 445: ;
0000 446: ;
0000 447: ;
0000 448: ; *****
0000 449: ; * ATARI SIO system interface *
0000 450: ; *****
0000 451: ;
0000 452:
0000 453: DDEVIC:             equ  $0300 ; U
0000 454: DUNIT:              equ  $0301 ; Unit numb
0000 455: DCOMND:             equ  $0302 ; B
0000 456: DSTATS:             equ  $0303 ; C
0000 457: DBUFLO:             equ  $0304 ; D
0000 458: DBUFHI:             equ  $0305
0000 459: DTIMLO:             equ  $0306 ; D
0000 460: DBYTLO:             equ  $0308 ; N
0000 461: DBYTHI:             equ  $0309
0000 462: SECTOR:             equ  $030A ; S
0000 463: SIO:                equ  $E459 ; S
0000 464: ;
0000 465: ;
0000 466: ; *****
0000 467: ; * ATARI CIO system interface *
0000 468: ; *****
0000 469: ;
0000 470: ;   Zero page IOCB declaration
0000 471: ;   -----
0000 472: ;
0000 473: ICHIDZ:             equ  $20  ; H
0000 474: ICDNOZ:             equ  $21  ; D
0000 475: ICCOMZ:             equ  $22  ; C
0000 476: ICSTAZ:             equ  $23  ; S
0000 477: ICBALZ:             equ  $24  ; B

```

```

0000 478: ICBAHZ:          equ  $25
0000 479: ICPTLZ:          equ  $26  ; P
0000 480: ICPTHZ:          equ  $27
0000 481: ICBL LZ:          equ  $28  ; B
0000 482: ICBLHZ:          equ  $29
0000 483: ICAX1Z:          equ  $2A  ; A
0000 484: ICAX2Z:          equ  $2B
0000 485: ICAX3Z:          equ  $2C
0000 486: ICIDNO:          equ  $2E  ; i
0000 487: ICCHAR:          equ  $2F  ; C
0000 488: s_ptr:           equ  $a5      ; wo
0000 489: d_ptr:           equ  $a6      ; p
0000 490: HATABS:          equ  $031A  ; D
0000 491: ;
0000 492: ;   I/O control blocks - main
0000 493: ;   -----
0000 494: ;
0000 495: ICHID:              equ  $0340  ; Handler i
0000 496: ICDNO:              equ  $0341  ; Device nu
0000 497: ICCOM:              equ  $0342  ; Command c
0000 498: ICSTA:              equ  $0343  ; Status of
0000 499: ICBAL:              equ  $0344  ; Buffer ad
0000 500: ICBAH:              equ  $0345
0000 501: ICPTL:              equ  $0346  ; Put byte
0000 502: ICPH:              equ  $0347
0000 503: ICBL LZ:          equ  $0348  ; Buffer le
0000 504: ICBLH:              equ  $0349
0000 505: ICAX1:              equ  $034A  ; Auxiliary
0000 506: ICAX2:              equ  $034B
0000 507: ICAX3:              equ  $034C  ; Position
0000 508: CIO:              equ  $E456
0000 509: ;
0000 510: ;   Command codes for IOCB
0000 511: ;   -----
0000 512: ;
0000 513: CCOPEN:              equ  3      ; O
0000 514: CCGTRC:              equ  5      ; G
0000 515: CCGTCH:              equ  7      ; G
0000 516: CCPURC:              equ  9      ; P
0000 517: CCPUCH:              equ  11     ; P
0000 518: CCCLOS:              equ  12     ; C
0000 519: CCSTAT:              equ  13     ; I
0000 520: CCSPEC:              equ  14     ; I
0000 521: ;
0000 522: ;   Special entry commands for disk
0000 523: ;   -----
0000 524: ;
0000 525: SCRENA:              equ  32     ; R
0000 526: SCDELE:              equ  33     ; D
0000 527: SCLOCK:              equ  34     ; L
0000 528: SCPROT:              equ  35     ; P
0000 529: SCUNPR:              equ  36     ; U
0000 530: SCPONT:              equ  37     ; P

```

```

0000 531: SCNOTE:          equ 38 ; N
0000 532: SCLen:          equ 39 ; Get file
0000 533: SCLOAD:          equ 40 ; L
0000 534: SCSAVE:          equ 41 ; S
0000 535: SCCRED:          equ 42 ; C
0000 536: SCDELD:          equ 43 ; D
0000 537: SCCWD:           equ 44 ; C
0000 538: SCBOOT:          equ 45 ; S
0000 539: SCUNLO:          equ 46 ; U
0000 540: SCCHKD:          equ 47 ; C
0000 541: SCQDIR:          equ 48 ; G
0000 542: SCFORM:          equ 254 ; F
0000 543: SCMAXX:          equ 48 ; l
0000 544: ;
0000 545: ;
0000 546: ;   Format of data returned from 'scchkd'
0000 547: ;   -----
0000 548: ; notes:
0000 549: ;   This data returned in a buffer pointed to by
0000 550: ;   icbl (data length normally). The icba (buffer addr
0000 551: ;   points to the 'Dn:' like normal. The following is
0000 552: ;   format of the data returned (start address is icbl
0000 553: ;
0000 554: CDVER:              equ 0  ; version o
0000 555: CDDEN:              equ 1  ; density
0000 556: CDTOT:              equ 2  ; number of
0000 557: CDFRE:              equ 4  ; number of
0000 558: CDVOL:              equ 6  ; volume na
0000 559: CDSEQ:              equ 14 ; sequence
0000 560: CDRAN:              equ 15 ; random #
0000 561: CDWRT:              equ 16 ; write loc
0000 562: ;
0000 563: ;
0000 564: ;   CIO open type flags (aux1)
0000 565: ;   -----
0000 566: ;
0000 567: OMAPP:              equ $01 ; %000000
0000 568: OMFMT:              equ $02 ; %000000
0000 569: OMREA:              equ $04 ; %000001
0000 570: OMWRT:              equ $08 ; %000010
0000 571: OMDIR:              equ $10 ; %000100
0000 572: OMSUB:              equ $20 ; %001000
0000 573: ;
0000 574: ;   *****
0000 575: ;   * SpartaDOS error codes *
0000 576: ;   *****
0000 577: ;
0000 578: BREAK:              equ $80 ; B
0000 579: IOCBN:              equ $81 ; IOCB alre
0000 580: NODEVI:              equ $82 ; N
0000 581: NOREAD:              equ $83 ; F
0000 582: BADCOME:          equ $84 ; Bad IOCB
0000 583: NOPER:              equ $85 ; Channel/I

```

```

0000 584: BADIOCB:          equ  $86  ; Bad IOCB/
0000 585: NOWRITE:           equ  $87  ; File/IOCB
0000 586: EOFR:             equ  $88  ; Atari EOF
0000 587: TRUNCR:          equ  $89  ; T
0000 588: NODRIVE:         equ  $8A  ; drive gon
0000 589: NACK:            equ  $8B  ; drive did
0000 590: SFRMER:         equ  $8C  ; s
0000 591: CSRRNG:         equ  $8D  ; c
0000 592: OVRRUN:         equ  $8E  ; S
0000 593: CHKSER:         equ  $8F  ; C
0000 594: BADSEC:         equ  $90  ; B
0000 595: ILLSCR:         equ  $91  ; I
0000 596: IOBCR:          equ  $92  ; I
0000 597: NORAMS:         equ  $93  ; N
0000 598: NSPDOS:         equ  $94  ; N
0000 599: ;rev20: = $95    ; Diskette not version 2.0 or better
0000 600: NODIR:           equ  $96  ; Directory
0000 601: FILEX:          equ  $97  ; File exis
0000 602: NBINF:          equ  $98  ; Not binar
0000 603: LOGON:          equ  $99  ; Logon mes
0000 604: ;      = $9A
0000 605: ;      = $9B
0000 606: ;      = $9C
0000 607: ;      = $9D
0000 608: ;      = $9E
0000 609: ;      = $9F
0000 610: DRIVER:         equ  $A0  ; Drive num
0000 611: ;      = $A1    ; Too many channels open
0000 612: FULLER:         equ  $A2  ; D
0000 613: ILWCAR:         equ  $A3  ; I
0000 614: ERAPRO:         equ  $A4  ; F
0000 615: FNAMEERR:       equ  $A5  ; File name
0000 616: RNGER:          equ  $A6  ; Position
0000 617: CANTDEL:        equ  $A7  ; Cannot de
0000 618: INVALIDC:       equ  $A8  ; I
0000 619: WRTLOK:         equ  $A9  ; D
0000 620: NOFOU:          equ  $AA  ; File not
0000 621: ;
0000 622: ;
0000 623: ;
0000 624: ;
0000 625: ; *****
0000 626: ; * SpartaDOS database offset declarations *
0000 627: ; *****
0000 628: ;
0000 629: ; Sector map descriptor
0000 630: ; -----
0000 631: ;
0000 632: SMNL:             equ  0      ; Sector nu
0000 633: SMLL:             equ  2      ; Sector nu
0000 634: ;                  ; zero imp
0000 635: SMSN:             equ  4      ; Start of
0000 636: ;                  ; means an

```

```

0000 637: ;
0000 638: ;   Directory entry descriptor
0000 639: ;   -----
0000 640: MDD:           equ 9   ; Offset in
0000 641: MDOFF:         equ 0   ; This des
0000 642: ;               ; when in
0000 643: ;
0000 644: DESTA:         equ 0   ; Statut of
0000 645: ;               ; follows:
0000 646: DESSUB:        equ OMSUB ;
0000 647: DESUSE:        equ $08  ;
0000 648: DESDEL:        equ $10  ;
0000 649: DESPRO:        equ $01  ;
0000 650: DESARC:        equ $02  ;
0000 651: DESHID:        equ $04  ;
0000 652: ;
0000 653: DESMAP:        equ 1   ; S
0000 654: DENBYT:        equ 3   ; N
0000 655: DEFNAM:        equ 6   ; F
0000 656: DEEXT:         equ 14  ; File name
0000 657: DEDAT:         equ 17  ; Date of c
0000 658: DETIM:         equ 20  ; Time of c
0000 659: ;
0000 660: ;
0000 661: ; lable below was dex but now is ddex to keep from
0000 662: ; confusing the compiler
0000 663: DDEX:         equ 23   ; Length of
0000 664: ;
0000 665: ;
0000 666: ;   Drive/diskette description block
0000 667: ;   -----
0000 668: ;
0000 669: BMD:          equ MDD+2 ; Offset in
0000 670: SDOFF:        equ 2   ; Offset to
0000 671: SDTOT:        equ 0   ; Total num
0000 672: SDFRE:        equ 2   ; Number of
0000 673: SDQBM:        equ 4   ; Number of
0000 674: SDBMS:        equ 5   ; First bit
0000 675: SDALLOC:      equ 7   ; Sector nu
0000 676: SDDIR:        equ 9   ; Sector nu
0000 677: SDVOL:        equ 11  ; Volume na
0000 678: SDX:          equ 19  ; Length of
0000 679: ;
0000 680: ;   Drive setup parameters
0000 681: ;   -----
0000 682: ;
0000 683: DPD:          equ BMD+SDX ; Offset in
0000 684: ;
0000 685: SPTRK:        equ 0   ; Number of
0000 686: SPDEN:        equ 1   ; Diskette
0000 687: CPREV:        equ 2   ; Revision
0000 688: CPBUF:        equ 3   ; Number of
0000 689: CPDRV:        equ 4   ; Default d

```

```

0000 690: CPSYS:          equ 5    ; Do AUTORU
0000 691: CPEXC:          equ 6    ; Do AUTORU
0000 692: CCPX:          equ 7    ; Length of
0000 693: ;
0000 694: CDOFF:            equ SDOFF+SDX ; Offset
0000 695: ;
0000 696: ;   Extra parameters (NOT ON DOS 1.1)
0000 697: ;   -----
0000 698: ;
0000 699: EXTR:             equ DPD+CCPX
0000 700: BTSEC:           equ 0    ; number of
0000 701: DSEQU:           equ 1    ; access se
0000 702: DRAND:           equ 2    ; random nu
0000 703: DDMAP:           equ 3    ; First sec
0000 704: DDLOK:           equ 5    ; Write loc
0000 705: ;
0000 706: ;
0000 707: ;
0000 708: DRIVE:           equ DUNIT
0000 709: srcdrv:           equ dunit
0000 710: ;
0000 711: ;
0000 712: ;
0000 713: ;-----
0000 714: ;   boot program for DOS II
0000 715: ;=====
0000 716: ;
0000 717: DATBUF:           equ $2E00 ; l
0000 718: SECMAP:           equ $2F00 ; l
0000 719: CHPTR:           equ $90    ; character
0000 720: SECPTR:           equ $91    ; p
0000 721: BUF:             equ $92    ; data buff
0000 722: LENG:            equ $94    ; length of
0000 723: TEMP:            equ $96    ; a temp
0000 724: ;
0000 725: ;;   System Equates
0000 726: ;-----
0000 727: ;
0000 728: DOSVEC:           equ $0A
0000 729: ICAX4Z:           equ $2D
0000 730: ICAX5Z:           equ $2E
0000 731: RECVDN:           equ $39
0000 732: FEOF:           equ $3F
0000 733: ERRNO:           equ $49
0000 734: FKDEF:           equ $60
0000 735: PALNTS:           equ $62
0000 736: LOGCOL:           equ $63
0000 737: ADRESS:           equ $64
0000 738: SAVADR:           equ $68
0000 739: BITMSK:           equ $6E
0000 740: SHFAMT:           equ $6F
0000 741: COLAC:           equ $72
0000 742: DAUX1:           equ $030A

```

```
0000 743: DAUX2:          equ  $030B
0000 744: RANDOM:          equ  $D20A
0000 745: consol:           equ  $d01f
0000 746: KEYBDV:          equ  $E420
0000 747: CIOV:           equ  $E456
0000 748: SIOV:           equ  $E459
0000 749:
0000 750: ;-----
0000 751: ;
0000 752: ; External reference Equates
0000 753: ;-----
0000 754:
0000 755:
0000 756:
0000 757: ;
0000 758: pr_ptr:           equ  $84
0000 759:
0000 760: ;-----
0000 761: ; pbi common Routines
0000 762: ;
0000 763: ;
0000 764: Ld1e0:           equ  $d1e0
0000 765: Ld1e2:           equ  $d1e2
0000 766: pbi_ram:         equ  $d600
0000 767: pbi_rom:         equ  $D800
0000 768: PBI_SIO:         equ  $D805
0000 769:
0000 770: ;-----
0000 771: ;I am defining
0000 772: ; CSS Black Box Vars Here
0000 773: ;
0000 774: ;-----
0000 775: ;
0000 776: BBHD1:           equ  $D600
0000 777: BBHD2:           equ  $D612
0000 778: BBHD3:           equ  $D624
0000 779: BBHD4:           equ  $D636
0000 780: BBHD5:           equ  $D648
0000 781: BBHD6:           equ  $D65A
0000 782: BBHD7:           equ  $D66C
0000 783: BBHD8:           equ  $D67E
0000 784: BBHD9:           equ  $D690
0000 785: ;
0000 786: RAMPAGE:         equ  $D1BC
0000 787: CFG_PAGE:        equ  $F5
0000 788: MISC_PAGE:       equ  $FE
0000 789: DUMMYPAGE:       equ  $FF
0000 790: BB_Sensel:       equ  $D1C0
0000 791: PBIBANK:         equ  $D1FF
0000 792: PBISHADOW:       equ  $0248
0000 793: HDCODE_BANK:     equ  $04
0000 794: ;
0000 795:
```



```
0000 796: ; Disk Drive Numbers
0000 797: ;
0000 798: DISK0:          equ  $00
0000 799: DISK1:          equ  $10
0000 800: DISK2:          equ  $20
0000 801: DISK3:          equ  $30
0000 802: DISK4:          equ  $40
0000 803: DISK5:          equ  $50
0000 804: DISK6:          equ  $60
0000 805: DISK7:          equ  $70
0000 806: ;
0000 807:
0000 808:
0000 809: ;-----
0000 810: ; Current Month in dec for each time it is compiled
0000 811: ;even if there is no code change
0000 812: l_month:          equ  10
0000 813: ;-----
0000 814: ; Current day in dec for each time it is compiled
0000 815: ;even if there is no code change
0000 816: l_day:             equ  15
0000 817: ;-----
0000 818: ; Current year in dec for each time it is compiled
0000 819: ;even if there is no code change
0000 820: l_year:             equ  2012
0000 821: ;-----
0000 822: ; Current year in dec for each time it is compiled
0000 823: ;even if there is no code change this is a short year form
0000 824: c_year_short:       equ  12
0000 825: ;-----
0000 826: ; Current build so we know witch set of com files
0000 827: ;goes together!
0000 828: l_dos_build:         equ  31
0000 829: ;-----
0000 830: ; is this file registered?
0000 831: ; yes = 1  no = 0  $81 site license
0000 832: ;
0000 833: file..yes:          equ  1
0000 834: file..no:           equ  0
0000 835: file..site:         equ  $81
0000 836: ;-----
0000 837: ;Who Compiled this file:
0000 838: ;UNknown:           equ  0
0000 839: ;Steve Carden:      equ  1
0000 840: ;Ken Ames:          equ  2
0000 841: ;Michael:           equ  3
0000 842: ;
0000 843: if.who_compiled:     equ  1
0000 844: ;-----
0000 845: ; Gotta define if it is registered.
0000 846: file_registered:     equ  file..no
0000 847: ;
0000 848: ;
```

```
0000      849: ; File revision history. Variables
0000      850:
0000      851: com_file_name:      .macro
0000      852: ;                  dc.b  12,"filename.ext",$9b
0000      853:                  dc.b  12,"Dirbuild.com",$9b
0000      854:                  .endm
0000      855:
0000      856:
0000      857: ;-----
0000      858: ; File revision history. Version Number in hex
0000      859: ;
0000      860: file_ver:          equ  $11
0000      861: ;-----
0000      862: ; the Month in dec for the first time this revision
0000      863: ;was compiled
0000      864: ;
0000      865: c_month:          equ  10
0000      866: ;-----
0000      867: ; the day in dec for the first time this revision
0000      868: ;was compiled
0000      869: c_day:           equ  10
0000      870: ;-----
0000      871: ; the year in dec for the first time this revision
0000      872: ;was compiled
0000      873: c_year:          equ  2012
0000      874: ;-----
0000      875: ; Type of compiler used in program
0000      876: ;
0000      877: ; 0 = unknown Compiler
0000      878: ; 1 = xasm
0000      879: ; 2 = mac_65
0000      880: ; 3 = basic
0000      881: ; 4 = compiled basic xl
0000      882: ; 5 = C65
0000      883: ; 6 = Action
0000      884: ;
0000      885: ; We can add more as time goes on!
0000      886: ;
0000      887: ;
0000      888: xasm:            equ   1
0000      889: mac_65:          equ   2
0000      890: basic:           equ   3
0000      891: basicxl:         equ   4
0000      892: c65:            equ   5
0000      893: action:          equ   6
0000      894:
0000      895: file_compiler:   equ   xasm
0000      896: ;-----
0000      897: ; is this relocatable code ?
0000      898: ; anyother value other than 1 or 2 would be unknown
0000      899: r.yes:          equ    1
0000      900: r.no:           equ    2
0000      901: ;
```

```

0000 902: ;-----
0000 903: ; Gotta define if it so I can print what I need
0000 904: r.crl:      equ  $7d
0000 905: r.crlf:      equ  $9b
0000 906: r.space:     equ  $20
0000 907: r.bypassDOS:  equ  $15
0000 908: r.bypass:     equ  $ff
0000 909:
0000 910: l_frstscreenbyte: equ  r.crl
0000 911: ;-----
0000 912: ; Gotta define if it can be relocated
0000 913: l_relocatable: equ  r..yes
0000 914: ;
0000 915: ;-----
0000 916: ; The language the output file is in.
0000 917: ;
0000 918: ;
0000 919: ; 0 =  Undefined
0000 920: ; 1 =  English
0000 921: ; 2 =  German
0000 922: ;
0000 923: r.language:   equ  1
0000 924: ;-----
0000 925: ;
0000 926: ;   Start of actual code
0000 927:
0000 928:
4000 929:      .org  $4000
4000 930: win_start:
4000 931: header_info:
4000 932:      .include  header
42a7 933: Head_back:
42a7 934:
42a7 935:
42a7 20b45b 936:      jsr  pbi_test
42aa adb35b 937:      lda  pbi_interface
42ad c900 938:      cmp  #$00
42af f038 939:      beq  .a
42b1 c901 940:      cmp  #$01
42b3 f009 941:      beq  .mio
42b5 c902 942:      cmp  #$02
42b7 f008 943:      beq  .bb
42b9 c903 944:      cmp  #$03
42bb f007 945:      beq  .kpi
42bd 60 946:      rts
42be 947:
42be 4c0d43 948: .mio:      jmp  .show_mio
42c1 949:
42c1 4c3452 950: .bb:      jmp  start
42c4 951:
42c4 204b57 952: .kpi:      jsr  printsi
42c7 fd 953:      dc.b 253
42c8 9b9b4e 954:      dc.b $9b,$9b,"No KPI support at

```

```

42e8 60 955:          rts
42e9 956:
42e9 204b57 957: .a:      jsr  printsi
42ec fd 958:          dc.b 253
42ed 9b9b4e 959:          dc.b $9b,$9b,"No PBI device dete
4309 209b56 960:          jsr  bell
430c 60 961:          rts
430d 962:
430d 204b57 963: .show_mio:  jsr  printsi
4310 fd 964:          dc.b 253
4311 9b9b4e 965:          dc.b $9b,$9b,"No MIO support at
4331 209b56 966:          jsr  bell
4334 60 967:          rts
4335 968:
4335 969:
4335 970:
5000 971:          .org $5000
5000 972:
5000 4c3452 973:          jmp  start
5003 974:
5003 975: buffer:          ds.b 256
5103 976:
5103 977: buffer_1:        ds.b 256
5203 978:
5203 8000 979: sector_size:     dc.w 128
5205 08 980: my_srcdrv:       dc.b 8
5206 08 981: dstdrv:          dc.b 8
5207 00 982: bb_count:        dc.b 0          ; c
5208 983:
5208 984:
5208 0000 985: start_sector:    dc.w 0
520a ffff 986: end_sector:      dc.w $ffff
520c 987:
520c 988:
520c 989: partition_info:   ds.b 9
5215 990:          ds.b 1          ; b
5216 991: par_density:      ds.b 1          ; B
5217 992:          ds.b 4          ; a
521b 993: par_sect_size:    ds.w 1          ; n
521d 994:          ds.b 1
521e 995:          ds.b 1
521f 996: my_drive_lo:
521f 00 997:          dc.b BBHD1
5220 12 998:          dc.b BBHD2
5221 24 999:          dc.b BBHD3
5222 36 1000:         dc.b BBHD4
5223 48 1001:         dc.b BBHD5
5224 5a 1002:         dc.b BBHD6
5225 6c 1003:         dc.b BBHD7
5226 7e 1004:         dc.b BBHD8
5227 90 1005:         dc.b BBHD9
5228 1006:
5228 1007: my_drive_high:

```

```

5228 d6 1008:          dc.b BBHD1 / 256
5229 d6 1009:          dc.b BBHD2 / 256
522a d6 1010:          dc.b BBHD3 / 256
522b d6 1011:          dc.b BBHD4 / 256
522c d6 1012:          dc.b BBHD5 / 256
522d d6 1013:          dc.b BBHD6 / 256
522e d6 1014:          dc.b BBHD7 / 256
522f d6 1015:          dc.b BBHD8 / 256
5230 d6 1016:          dc.b BBHD9 / 256
5231      1017:
5231      1018:
5231      1019:
5231      1020:
5231 4cffff 1021: xiov:      jmp  $ffff
5234      1022:
5234      1023:
5234      1024: start:
5234 a960 1025:          LDA  #$60
5236 8d3452 1026:          STA  start
5239 208081 1027:          jsr  START_IT    ; i
523c      1028:
523c      1029: ;----- ldy #$00
523c      1030:
523c ad1fd0 1031:          lda  consol    ; c
523f c907 1032:          cmp  #7        ; l
5241 f003 1033:          beq  .zzz      ; n
5243 4c7c5c 1034:          jmp  help_menu  ; S
5246      1035: .zzz:
5246      1036:
5246 ad0007 1037:          lda  $700
5249 c952 1038:          cmp  #'R'
524b f002 1039:          beq  .real_dos
524d d013 1040:          bne  .no_dos
524f      1041:
524f      1042: .real_dos:
524f a03f 1043:          LDY  #LBUF
5251      1044:
5251 b10a 1045: .comp:          lda  (comtab),y
5253 c99b 1046:          cmp  #$9b
5255 f008 1047:          beq  .exit
5257 c920 1048:          cmp  #' '
5259 f00a 1049:          beq  .cont
525b c8 1050:          iny
525c 4c5152 1051:          jmp  .comp
525f      1052:
525f      1053:
525f 4ce652 1054: .exit:          jmp  error
5262      1055:
5262 4c4153 1056: .no_dos:          jmp  nodos
5265      1057:
5265      1058:
5265      1059: .cont:
5265 c8 1060:          iny

```

```

5266 c8 1061:          iny
5267 b10a 1062:        LDA  (comtab),Y
5269 8d0862 1063:      sta  par_drv_letter+1
526c c931 1064:        CMP  #'1'
526e 9007 1065:        bcc  .ea
5270 c93a 1066:        cmp  #'9'+1
5272 b003 1067:        bcs  .ea
5274 4c7a52 1068:      jmp  .c
5277      1069:
5277 4ce652 1070: .ea:      jmp  error
527a      1071:
527a      1072:
527a a200 1073: .c:      ldx  #$00
527c      1074:
527c e8 1075: .d:      inx
527d c8 1076: .e:      iny
527e b10a 1077:      lda  (comtab),y
5280 c922 1078:      cmp  #""
5282 f0f9 1079:      beq  .e
5284 c99b 1080:      cmp  #$9b
5286 f00f 1081:      beq  .done_with_cli
5288 c93e 1082:      cmp  #'>'
528a d002 1083:      bne  .tam
528c a95c 1084:      lda  #"
528e      1085: .tam
528e 9d6562 1086:      sta  par_Name,x
5291 ee6562 1087:      inc  par_Name
5294 4c7c52 1088:      jmp  .d
5297      1089:
5297      1090: .done_with_cli:
5297 ad0862 1091:      lda  par_drv_letter+1
529a e930 1092:      sbc  #48
529c 8d0552 1093:      sta  my_srcdrv
529f      1094:
529f      1095:
529f ad0552 1096:      lda  my_srcdrv
52a2 e901 1097:      sbc  #1
52a4 a8 1098:      tay
52a5 b91f52 1099:      lda  my_drive_lo,y
52a8 8d4356 1100:      sta  bb_set_drv+1
52ab b92852 1101:      lda  my_drive_high,y
52ae 8d4456 1102:      sta  bb_set_drv+2
52b1      1103:
52b1 4c6f53 1104:      jmp  start_it_now
52b4      1105:
52b4      1106: exit_program:
52b4 a99b 1107:      LDA  #$9B
52b6 20ca56 1108:      JSR  echo
52b9      1109: ;      LDX  stack_save
52b9      1110: ;      TXS
52b9 60 1111:      RTS
52ba      1112:
52ba      1113: good_api:

```

```

52ba 204b57 1114:      jsr  printsi
52bd 20446f 1115:      dc.b  " Done!",$9b,-1
52c5 4cb452 1116:      jmp  exit_program
52c8      1117:
52c8      1118: Bad_api:
52c8 8ce552 1119:      sty  rc_data
52cb 204b57 1120:      jsr  printsi
52ce 206572 1121:      dc.b  " error code #",-1
52dc ade552 1122:      lda  rc_data
52df 20945b 1123:      jsr  pr_byte
52e2 4cb452 1124:      jmp  exit_program
52e5      1125:
52e5      1126:
52e5      1127:
52e5 00      1128: rc_data:      dc.b  0
52e6      1129:
52e6      1130:
52e6      1131:
52e6      1132: error:
52e6 204b57 1133:      JSR  printsi
52e9 9b4469 1134:      dc.b  $9b,"Dirbuild Dn:"
52f6 9b      1135:      dc.b  $9b
52f7 d2e5f1 1136:      dc.b  <+128>,"Requires a
530b 9b9b   1137:      dc.b  $9b,$9b
530d 446e3a 1138:      dc.b  "Dn: n= drive id
5328 53796e 1139:      dc.b  "Syntax =",$9b
5331 646972 1140:      dc.b  "dirbuild D8:"
533d 9b9bff 1141:      dc.b  $9b,$9b,-1
5340      1142:
5340      1143:
5340 60      1144:      RTS
5341      1145: ;
5341      1146:
5341      1147: nodos:
5341 204b57 1148:      JSR  printsi
5344 9b      1149:      dc.b  $9b
5345 9b4469 1150:      dc.b  $9b,"Dirbuild "
534f f2e5f1 1151:      dc.b  <+128>,"requires"
5357 205265 1152:      dc.b  " RealDos & Black B
536b 9b9bff 1153:      dc.b  $9b,$9b,-1
536e 60      1154:      RTS
536f      1155:
536f      1156:
536f      1157:
536f      1158: start_it_now:
536f 204b57 1159:      JSR  printsi
5372 207dff 1160:      dc.b  $20,$7d,-1
5375      1161:
5375 20a442 1162:      jsr  Head_back-3 ;re
5378 a900   1163:      lda  #low 0
537a 8d0852 1164:      sta  start_sector
537d a900   1165:      lda  #high 0
537f 8d0852 1166:      sta  start_sector

```

```

5382 a000 1167:          ldy  #$00
5384 a900 1168:          lda  #$00
5386 990351 1169: .zero:      sta  buffer_1,y
5389 996662 1170:          sta  cmdtab2,y
538c c8 1171:          iny
538d c000 1172:          cpy  #$00
538f d0f5 1173:          bne  .zero
5391 a000 1174:          ldy  #$00
5393 a96c 1175:          lda  #$6c
5395 990350 1176: .b6c:      sta  buffer,y
5398 c8 1177:          iny
5399 c000 1178:          cpy  #$00
539b d0f8 1179:          bne  .b6c
539d a202 1180:          ldx  #2
539f a00c 1181:          ldy  #12
53a1 205856 1182:          jsr  gotoxy
53a4 204b57 1183:          jsr  printsi
53a7 577269 1184:          dc.b  "Writting a Partiti
53bd 1185:
53bd 1186:
53bd 1187: bb_loop_check:
53bd a205 1188:          ldx  #5
53bf a014 1189:          ldy  #20
53c1 205856 1190:          jsr  gotoxy
53c4 ad0752 1191:          lda  bb_count
53c7 20945b 1192:          jsr  pr_byte
53ca ad0752 1193:          lda  bb_count
53cd 1194:
53cd c960 1195:          cmp  #96
53cf f01c 1196:          beq  .all_done
53d1 a900 1197:          lda  #$00
53d3 8d0852 1198:          sta  start_sector
53d6 8d0952 1199:          sta  start_sector+1
53d9 ee0752 1200:          inc  bb_count
53dc ad0752 1201:          lda  bb_count
53df 200e56 1202:          jsr  SET_PART
53e2 b0d9 1203:          bcs  bb_loop_check
53e4 1204:
53e4 20fe53 1205:          jsr  .show_name
53e7 1206:
53e7 1207: ;          jmp  .all_done
53e7 1208:
53e7 1209: ;          jsr  run_loop
53e7 1210:
53e7 1211:
53e7 1212:
53e7 209781 1213:          jsr  format
53ea 1214:
53ea 4cbd53 1215:          jmp  bb_loop_check
53ed 1216:
53ed 1217: .all_done:
53ed 204b57 1218:          jsr  printsi
53f0 9b9b41 1219:          dc.b  $9b,$9b,"All Done!"

```



```

53fd 60 1220:          rts
53fe      1221:
53fe      1222: .show_name:
53fe a97d 1223:          lda  #$7d
5400 20ca56 1224:          jsr  echo
5403 20a442 1225:          jsr  Head_back-3
5406      1226:
5406      1227: ;          ldx  #34
5406      1228: ;          ldy  #5
5406      1229: ;          jsr  gotoxy
5406      1230: ;          jsr  printsi
5406      1231: ;          dc.b  " ",-1
5406      1232:
5406 a218 1233:          ldx  #24
5408 a00b 1234:          ldy  #11
540a 205856 1235:          jsr  gotoxy
540d 204b57 1236:          jsr  printsi
5410 202020 1237:          dc.b  " ",-1
541e a218 1238:          ldx  #24
5420 a00b 1239:          ldy  #11
5422 205856 1240:          jsr  gotoxy
5425      1241:
5425      1242:
5425 a922 1243:          lda  #""
5427 20ca56 1244:          jsr  echo
542a ad0c52 1245:          lda  partition_info
542d c900 1246:          cmp  #$00
542f f006 1247:          beq  .n2
5431 8d1680 1248:          sta  14016
5434 20ca56 1249:          jsr  echo
5437      1250: .n2:
5437 ad0d52 1251:          lda  partition_info+1
543a c900 1252:          cmp  #$00
543c f006 1253:          beq  .n3
543e 8d1780 1254:          sta  14016+1
5441 20ca56 1255:          jsr  echo
5444      1256: .n3:
5444 ad0e52 1257:          lda  partition_info+2
5447 c900 1258:          cmp  #$00
5449 f006 1259:          beq  .n4
544b 8d1880 1260:          sta  14016+2
544e 20ca56 1261:          jsr  echo
5451      1262: .n4:
5451 ad0f52 1263:          lda  partition_info+3
5454 c900 1264:          cmp  #$00
5456 f006 1265:          beq  .n5
5458 8d1980 1266:          sta  14016+3
545b 20ca56 1267:          jsr  echo
545e      1268:
545e      1269: .n5:
545e ad1052 1270:          lda  partition_info+4
5461 c900 1271:          cmp  #$00
5463 f006 1272:          beq  .n6

```

```

5465 8d1a80 1273:          sta 14016+4
5468 20ca56 1274:          jsr echo
546b      1275:
546b      1276: .n6:
546b ad1152 1277:          lda partition_info+5
546e c900 1278:          cmp #$00
5470 f006 1279:          beq .n7
5472 8d1b80 1280:          sta 14016+5
5475 20ca56 1281:          jsr echo
5478      1282:
5478      1283: .n7:
5478 ad1252 1284:          lda partition_info+6
547b c900 1285:          cmp #$00
547d f006 1286:          beq .n8
547f 8d1c80 1287:          sta 14016+6
5482 20ca56 1288:          jsr echo
5485      1289:
5485      1290: .n8:
5485 ad1352 1291:          lda partition_info+7
5488 c900 1292:          cmp #$00
548a f006 1293:          beq .n9
548c 8d1d80 1294:          sta 14016+7
548f 20ca56 1295:          jsr echo
5492      1296:
5492      1297: .n9:
5492 ad1452 1298:          lda partition_info+8
5495 c900 1299:          cmp #$00
5497 f006 1300:          beq .na
5499 8d1e80 1301:          sta 14016+8
549c 20ca56 1302:          jsr echo
549f      1303: .na:
549f a922 1304:          lda #""
54a1 20ca56 1305:          jsr echo
54a4      1306:
54a4      1307:
54a4      1308:
54a4      1309:
54a4 a205 1310:          ldx #5
54a6 a00e 1311:          ldy #14
54a8 205856 1312:          jsr gotoxy
54ab 204b57 1313:          jsr printsi
54ae 202020 1314:          dc.b "
54ce a205 1315:          ldx #5
54d0 a00e 1316:          ldy #14
54d2 205856 1317:          jsr gotoxy
54d5      1318:
54d5 204b57 1319:          jsr printsi
54d8 546f74 1320:          dc.b "Total Partition Se
54f1      1321:
54f1 ad1b52 1322:          LDA par_sect_size
54f4 38 1323:          SEC
54f5 e901 1324:          SBC # low 1
54f7 8d0a52 1325:          STA end_sector

```

```

54fa ad1c52 1326:      LDA   par_sect_size+1
54fd e900 1327:      SBC   # high 1
54ff 8d0b52 1328:      STA   end_sector+1
5502      1329:
5502 ad0a52 1330:      lda   end_sector
5505 ae0b52 1331:      ldx   end_sector+1
5508 20965b 1332:      jsr   pr_card
550b 60 1333:      rts
550c      1334:
550c      1335:
550c      1336:
550c      1337:
550c      1338:
550c      1339:
550c      1340:
550c      1341: run_loop:
550c ad0852 1342:      LDA   start_sector
550f 18 1343:      CLC
5510 6901 1344:      ADC   # low 1
5512 8d0852 1345:      STA   start_sector
5515 ad0952 1346:      LDA   start_sector+1
5518 6900 1347:      ADC   # high 1
551a 8d0952 1348:      STA   start_sector+1
551d a99b 1349:      lda   #$9b
551f 20ca56 1350:      jsr   echo
5522      1351:
5522 a222 1352:      ldx   #34
5524 a005 1353:      ldy   #5
5526 205856 1354:      jsr   gotoxy
5529      1355:
5529 ad0852 1356:      lda   start_sector
552c ae0952 1357:      ldx   start_sector+1
552f      1358:
552f 20965b 1359:      jsr   pr_card
5532 ad0952 1360:      lda   start_sector+1
5535 c900 1361:      cmp   #$00
5537 d014 1362:      bne   .do_256
5539 ad0852 1363:      lda   start_sector
553c c903 1364:      cmp   #$03
553e b00d 1365:      bcs   .do_256
5540      1366:
5540 a980 1367:      lda   #low 128
5542 8d0352 1368:      sta   sector_size
5545 a900 1369:      lda   #high 128
5547 8d0452 1370:      sta   sector_size+1
554a 4c5755 1371:      jmp   .do_rw
554d      1372:
554d      1373:
554d a900 1374: .do_256:      lda   #low 256
554f 8d0352 1375:      sta   sector_size
5552 a901 1376:      lda   #high 256
5554 8d0452 1377:      sta   sector_size+1
5557      1378:

```

```

5557      1379: .do_rw:
5557 206955 1380:          jsr  read_write
555a      1381:
555a ad0952 1382:          lda  start_sector+1
555d c9ff  1383:          cmp  #$ff
555f d0ab  1384:          bne  run_loop
5561 ad0852 1385:          lda  start_sector
5564 c9ff  1386:          cmp  #$ff
5566 d0a4  1387:          bne  run_loop
5568      1388:
5568 60    1389:          rts
5569      1390:
5569      1391:
5569      1392:
5569      1393:
5569      1394:
5569      1395:
5569      1396: read_write:
5569      1397: ;read
5569 a931  1398:          lda  #$31
556b 8d0003 1399:          sta  $300
556e ad0552 1400:          lda  my_srcdrv
5571 8d0103 1401:          sta  $301
5574 a952  1402:          lda  #'R'      ; r
5576 8d0203 1403:          sta  $302      ;dc
5579 a940  1404:          lda  #$40      ;re
557b 8d0303 1405:          sta  $303      ;I
557e a903  1406:          lda  #low buffer
5580 8d0403 1407:          sta  dbuflo
5583 a950  1408:          lda  #high buffer
5585 8d0503 1409:          sta  dbufhi
5588 ad0352 1410:          lda  sector_size
558b 8d0803 1411:          sta  $308
558e      1412:
558e ad0452 1413:          lda  sector_size+1
5591 8d0903 1414:          sta  $309
5594 ad0852 1415:          lda  start_sector
5597 8d0a03 1416:          sta  $30a      ;lo
559a ad0952 1417:          lda  start_sector+1
559d 8d0b03 1418:          sta  $30b      ;hi
55a0 20f255 1419:          jsr  do_r_siov    ; r
55a3 a000  1420:          ldy  #$00
55a5 a96c  1421:          lda  #$6c
55a7      1422:
55a7 b90350 1423: .loop:          lda  buffer,y
55aa c96c  1424:          cmp  #$6c
55ac d008  1425:          bne  .nowrt
55ae c8    1426:          iny
55af c000  1427:          cpy  #$00
55b1 d0f4  1428:          bne  .loop
55b3 4cb755 1429:          jmp  .write_it
55b6      1430:
55b6      1431:

```

```

55b6      1432:
55b6 60    1433: .nowrt:                rts
55b7      1434:
55b7      1435:
55b7      1436:
55b7      1437: .write_it
55b7 a931  1438:                lda  #$31
55b9 8d0003 1439:                sta  $300
55bc ad0552 1440:                lda  my_srcdrv
55bf 8d0103 1441:                sta  $301
55c2 a957  1442:                lda  #'W'      ; W
55c4 8d0203 1443:                sta  $302      ;dc
55c7 a980  1444:                lda  #$80      ;re
55c9 8d0303 1445:                sta  $303      ;I
55cc a903  1446:                lda  #low buffer_1
55ce 8d0403 1447:                sta  dbuflo
55d1 a951  1448:                lda  #high buffer_1
55d3 8d0503 1449:                sta  dbufhi
55d6 ad0352 1450:                lda  sector_size
55d9 8d0803 1451:                sta  $308
55dc      1452:
55dc ad0452 1453:                lda  sector_size+1
55df 8d0903 1454:                sta  $309
55e2 ad0852 1455:                lda  start_sector
55e5 8d0a03 1456:                sta  $30a      ;lo
55e8 ad0952 1457:                lda  start_sector+1
55eb 8d0b03 1458:                sta  $30b      ;hi
55ee 200056 1459:                jsr  do_w_siov  ; w
55f1      1460:
55f1 60    1461:                rts
55f2      1462:
55f2      1463:
55f2      1464: do_r_siov:
55f2 a952  1465:                lda  #'R'      ; r
55f4 8d0203 1466:                sta  $302      ;dc
55f7 a940  1467:                lda  #$40      ;re
55f9 8d0303 1468:                sta  $303      ;I
55fc 2059e4 1469:                jsr  $e459
55ff      1470: ;                lda  $303      ;I
55ff      1471: ;                cmp  #$01
55ff      1472: ;                bne  do_r_siov
55ff 60    1473:                rts
5600      1474:
5600      1475: do_w_siov:
5600 a957  1476:                lda  #'W'      ; W
5602 8d0203 1477:                sta  $302      ;dc
5605 a980  1478:                lda  #$80      ;re
5607 8d0303 1479:                sta  $303      ;I
560a 2059e4 1480:                jsr  $e459
560d      1481: ;                lda  $303      ;I
560d      1482: ;                cmp  #$01
560d      1483: ;                bne  do_w_siov
560d 60    1484:                rts

```

```

560e 1485:
560e 1486: ;
560e 1487: ;
560e 1488: ;
560e 1489: ; LDA #20 ;map in partition 20
560e 1490: ;
560e 1491: SET_PART:
560e 18 1492: CLC
560f 690f 1493: ADC #$0F ;th
5611 a2fe 1494: LDX #MISC_PAGE ;se
5613 8ebcd1 1495: STX RAMPAGE
5616 8d53d6 1496: STA $D653 ;we
5619 1497: ; ;st
5619 a904 1498: LDA #HDCODE_BANK ;ba
561b 8d4802 1499: STA PBISHADOW ;s
561e 8dffd1 1500: STA PBIBANK ;pu
5621 202cd8 1501: JSR $D82C ;**
5624 a900 1502: LDA #0
5626 8d4802 1503: STA PBISHADOW
5629 8dffd1 1504: STA PBIBANK
562c b022 1505: BCS SET_ERROR
562e ad56d6 1506: LDA $D656 ;if
5631 301d 1507: BMI SET_ERROR ;pa
5633 a000 1508: LDY #0 ;pr
5635 a2fe 1509: SET_1: LDX #MISC_PAGE ;ge
5637 8ebcd1 1510: STX RAMPAGE ;pa
563a b956d6 1511: LDA $D656,Y ;mi
563d a2f5 1512: LDX #CFG_PAGE ;no
563f 8ebcd1 1513: STX RAMPAGE ;co
5642 1514: bb_set_drv:
5642 99ffff 1515: STA $ffff,Y ;$d
5645 990c52 1516: sta partition_info,y
5648 c8 1517: INY
5649 c012 1518: CPY #18 ;c
564b 90e8 1519: BCC SET_1
564d 18 1520: CLC ;We
564e 9001 1521: BCC SET_2 ;sk
5650 38 1522: SET_ERROR: SEC ;se
5651 a9ff 1523: SET_2: LDA #DUMMYPAGE ;re
5653 8dbcd1 1524: STA RAMPAGE
5656 60 1525: RTS ;a
5657 1526: ;
5657 1527: ;
5657 1528: ;
5657 1529: ;
5657 00 1530: override: dc.b $00 ;a
5658 1531: ;
5658 1532: ;
5658 8655 1533: gotoxy: stx 85
565a 8454 1534: sty 84
565c 60 1535: rts
565d 1536: ;
565d 1537: ;

```

```

565d a555 1538: savexy:          lda 85
565f 8d7356 1539:          sta save.x
5662 a554 1540:          lda 84
5664 8d7356 1541:          sta save.y
5667 60 1542:          rts
5668 1543:
5668 ad7356 1544: loadxy:          lda save.x
566b 8555 1545:          sta 85
566d ad7356 1546:          lda save.y
5670 8554 1547:          sta 84
5672 60 1548:          rts
5673 1549:
5673 1550: save.x:
5673 1551: save.y:
5673 1552:
5673 1553: ;-----;
5673 1554: ;prints Routeen macro      ;
5673 1555: ;this is a slow print to the screen;
5673 1556: ;-----;
5673 1557: ;
5673 68 1558: sprints:          pla
5674 85b0 1559:          sta $b0
5676 68 1560:          pla
5677 85b1 1561:          sta $b1
5679 a001 1562:          ldy #$01
567b b1b0 1563: .a:          lda ($b0).y
567d c9ff 1564:          cmp #$ff
567f f009 1565:          beq .b
5681 20ca56 1566:          jsr sprint
5684 209456 1567:          jsr .c
5687 4c7b56 1568:          jmp .a
568a 209456 1569: .b:          jsr .c
568d a5b1 1570:          lda $b1
568f 48 1571:          pha
5690 a5b0 1572:          lda $b0
5692 48 1573:          pha
5693 60 1574:          rts
5694 e6b0 1575: .c:          inc $b0
5696 d002 1576:          bne .exit
5698 e6b1 1577:          inc $b1
569a 60 1578: .exit:        rts
569b 1579: ;
569b 1580: ;
569b 1581: ;-----
569b 1582: ;this one make's a nice tone
569b 1583: ;-----
569b 1584:
569b 1585: bell:
569b 8dc756 1586:          sta bell_m3
569e 8cc856 1587:          sty bell_m3+1
56a1 8ec956 1588:          stx bell_m3+2
56a4 a0af 1589:          ldy #$af
56a6 a914 1590:          lda #$14

```

```

56a8 8d00d2 1591:          sta  $d200
56ab 8c01d2 1592: bell_m1:      sty  $d201
56ae a900  1593:          lda  #$00
56b0 8514  1594:          sta  $14
56b2 a514  1595: bell_m2:      lda  $14
56b4 c902  1596:          cmp  #$02
56b6 90fa  1597:          bcc  bell_m2
56b8 88    1598:          dey
56b9 c0a0  1599:          cpy  #$a0
56bb b0ee  1600:          bcs  bell_m1
56bd adc756 1601:          lda  bell_m3
56c0 acc856 1602:          ldy  bell_m3+1
56c3 aec956 1603:          ldx  bell_m3+2
56c6 60    1604:          rts
56c7 000000 1605: bell_m3:      dc.b  $00,$00,$00
56ca      1606: ;
56ca      1607: ;
56ca      1608: ;
56ca      1609: ;
56ca      1610: ;-----;
56ca      1611: ; PRINT BYTE ;
56ca      1612: ;-----;
56ca      1613: echo:
56ca 85b8  1614: sprint:      sta  $b8
56cc a904  1615:          lda  #$04
56ce 8d4a03 1616:          sta  $034a
56d1 86b6  1617:          stx  $b6
56d3 84b7  1618:          sty  $b7
56d5 a206  1619:          ldx  #$06
56d7 bd01e4 1620:          lda  $e400+1,x
56da 85b3  1621:          sta  $b3
56dc bd00e4 1622:          lda  $e400,x
56df 85b2  1623:          sta  $b2
56e1 a5b8  1624:          lda  $b8
56e3 20f256 1625:          jsr  .a
56e6 a6b6  1626:          ldx  $b6
56e8 a4b7  1627:          ldy  $b7
56ea a90c  1628:          lda  #$0c
56ec 8d4a03 1629:          sta  $034a
56ef a5b8  1630:          lda  $b8
56f1 60    1631:          rts
56f2      1632: ;
56f2 e6b2  1633: .a:          inc  $b2
56f4 6cb200 1634:          jmp  ($b2)
56f7 60    1635:          rts
56f8      1636: ;-----;
56f8      1637: ; GET KEY ROUTEEN ;
56f8      1638: ;this routeen get's one key and;
56f8      1639: ;return it to the accumulator ;
56f8      1640: ;-----;
56f8      1641: ;
56f8 86b6  1642: getkey:      stx  $b6
56fa 84b7  1643:          sty  $b7

```



```

56fc 200457 1644:          jsr  .a
56ff a6b6 1645:          ldx  $b6
5701 a4b7 1646:          ldy  $b7
5703 60 1647:          rts
5704 ad25e4 1648: .a:          lda  $e420+5
5707 48 1649:          pha
5708 ad24e4 1650:          lda  $e420+4
570b 48 1651:          pha
570c 60 1652:          rts
570d 1653: ;
570d 1654: ;-----;
570d 1655: ;CHANGES THE CHARTER TO UPCASE;
570d 1656: ;-----;
570d 1657: upcase:
570d c961 1658:          cmp  #$61
570f 9006 1659:          bcc  .a
5711 c97a 1660:          cmp  #$7a
5713 b002 1661:          bcs  .a
5715 e91f 1662:          sbc  #$1f
5717 60 1663: .a:          rts
5718 1664: ;
5718 1665: ;-----;
5718 1666: ;CHANGES THE CHARTER TO LOCASE;
5718 1667: ;-----;
5718 1668: locase:
5718 c941 1669:          cmp  #$41
571a 9006 1670:          bcc  .a
571c c95a 1671:          cmp  #$5a
571e b002 1672:          bcs  .a
5720 691f 1673:          adc  #$1f
5722 60 1674: .a:          rts
5723 1675: ;
5723 1676: ;-----;
5723 1677: ; ERROR NUMBER          ;
5723 1678: ;this take's the value of the a reg;
5723 1679: ;and put a hex number to the screen;
5723 1680: ;-----;
5723 1681: ;
5723 00 1682: h_cardd:          dc.b  $00
5724 1683: ;
5724 1684: ; print a hex word value
5724 1685: ; low byte in a reg x is high byte
5724 1686: ;
5724 8d2357 1687: h_card:          sta  h_cardd
5727 8a 1688:          txa
5728 203257 1689:          jsr  drive_error
572b ad2357 1690:          lda  h_cardd
572e 203257 1691:          jsr  drive_error
5731 60 1692:          rts
5732 1693: ;
5732 1694: ;
5732 48 1695: drive_error:      pha
5733 4a 1696:          lsr

```

```

5734 4a 1697:          lsr
5735 4a 1698:          lsr
5736 4a 1699:          lsr
5737 203d57 1700:      jsr  .a
573a 68 1701:          pla
573b 290f 1702:          and  #$0f
573d c90a 1703: .a:      cmp  #$0a
573f b004 1704:          bcs  .b
5741 0930 1705:          ora  #$30
5743 d002 1706:          bne  .c
5745 6936 1707: .b:      adc  #$36
5747 20ca56 1708: .c:      jsr  sprint
574a 60 1709:          rts
574b 1710: ;
574b 1711: ;
574b 1712: ;-----;
574b 1713: ; jsr printsi ;
574b 1714: ;this prints an inline string;
574b 1715: ; terminated by $ff ;
574b 1716: ;-----;
574b 1717: ; jsr prints
574b 1718: ; dc.b "print this", $9B,$FF
574b 1719: echosi:
574b 68 1720: printsi:      pla
574c 8d5c57 1721:          sta  pstr+1
574f 68 1722:          pla
5750 8d5d57 1723:          sta  pstr+2
5753 ee5c57 1724: prsl:      inc  pstr+1
5756 d003 1725:          bne  pstr
5758 ee5d57 1726:          inc  pstr+2
575b adffff 1727: pstr:      lda  $ffff
575e c9ff 1728:          cmp  #$ff
5760 f006 1729:          beq  estri
5762 20a157 1730:          jsr  fast_output
5765 4c5357 1731:          jmp  prsl
5768 ad5d57 1732: estri:      lda  pstr+2
576b 48 1733:          pha
576c ad5c57 1734:          lda  pstr+1
576f 48 1735:          pha
5770 60 1736:          rts
5771 1737: ;
5771 1738: ;-----;
5771 1739: ;actual screen handler ;
5771 1740: ;screen offset definitions;
5771 1741: ;-----;
5771 1742: scr_offset:
5771 0000 1743:          dc.w  0
5773 2800 1744:          dc.w  40
5775 5000 1745:          dc.w  80
5777 7800 1746:          dc.w 120
5779 a000 1747:          dc.w 160
577b c800 1748:          dc.w 200
577d f000 1749:          dc.w 240

```

```

577f 1801 1750:          dc.w 280
5781 4001 1751:          dc.w 320
5783 6801 1752:          dc.w 360
5785 9001 1753:          dc.w 400
5787 b801 1754:          dc.w 440
5789 e001 1755:          dc.w 480
578b 0802 1756:          dc.w 520
578d 3002 1757:          dc.w 560
578f 5802 1758:          dc.w 600
5791 8002 1759:          dc.w 640
5793 a802 1760:          dc.w 680
5795 d002 1761:          dc.w 720
5797 f802 1762:          dc.w 760
5799 2003 1763:          dc.w 800
579b 4803 1764:          dc.w 840
579d 7003 1765:          dc.w 880
579f 9803 1766:          dc.w 920
57a1      1767: ;
57a1 855a 1768: fast_output:      sta $5a
57a3 c920 1769:          cmp #32
57a5 9004 1770:          bcc do_look
57a7 c97d 1771:          cmp #125
57a9 9013 1772:          bcc do_here
57ab a000 1773: do_look:          ldy #0
57ad b9b458 1774: lookup:          lda char_table,
57b0 f00c 1775:          beq do_here
57b2 c55a 1776:          cmp $5a
57b4 f003 1777:          beq go_cio
57b6 c8 1778:          iny
57b7 d0f4 1779:          bne lookup
57b9 a55a 1780: go_cio:          lda $5a
57bb 4cc458 1781:          jmp putlocal
57be a000 1782: do_here:          ldy #0
57c0 a55d 1783:          lda $5d
57c2 915e 1784:          sta ($5e),y
57c4 a55a 1785:          lda $5a
57c6 c99b 1786:          cmp #$9b
57c8 f044 1787:          beq docr
57ca 209358 1788: notcr:          jsr get_adr
57cd a55a 1789:          lda $5a
57cf 297f 1790:          and #$7f
57d1 c920 1791:          cmp #32
57d3 9007 1792:          bcc add64
57d5 c960 1793:          cmp #96
57d7 9009 1794:          bcc sub32
57d9 4ce557 1795:          jmp asis
57dc 18 1796: add64:          clc
57dd 6940 1797:          adc #64
57df 4ce557 1798:          jmp asis
57e2 38 1799: sub32:          sec
57e3 e920 1800:          sbc #32
57e5 245a 1801: asis:          bit $5a
57e7 1002 1802:          bpl xxlate

```

```

57e9 0980 1803:          ora  #$80
57eb a000 1804: xxlate:      ldy  #0
57ed 915e 1805:          sta  ($5e),y
57ef e655 1806:          inc  $55
57f1 e663 1807:          inc  $63
57f3 a555 1808:          lda  $55
57f5 c928 1809:          cmp  #40
57f7 b019 1810:          bcs  next_row
57f9 c8 1811:          iny
57fa b15e 1812:          lda  ($5e),y
57fc 855d 1813:          sta  $5d
57fe 0980 1814:          ora  #$80
5800 aef002 1815:          ldx  752
5803 d002 1816:          bne  nocurs1
5805 915e 1817:          sta  ($5e),y
5807 e65e 1818: nocurs1:      inc  $5e
5809 d002 1819:          bne  nooav
580b e65f 1820:          inc  $5e+1
580d 60 1821: nooav:      rts
580e a900 1822: docr:      lda  #0
5810 8563 1823:          sta  $63
5812 a552 1824: next_row:    lda  $52
5814 8555 1825:          sta  $55
5816 e654 1826:          inc  $54
5818 a454 1827:          ldy  $54
581a c018 1828:          cpy  #24
581c b013 1829:          bcs  scroll
581e 209358 1830:          jsr  get_adr
5821 a000 1831:          ldy  #0
5823 b15e 1832:          lda  ($5e),y
5825 855d 1833:          sta  $5d
5827 aef002 1834:          ldx  752
582a d004 1835:          bne  nocurs2
582c 0980 1836:          ora  #$80
582e 915e 1837:          sta  ($5e),y
5830 60 1838: nocurs2:    rts
5831 c654 1839: scroll:      dec  $54
5833 a558 1840:          lda  $58
5835 8568 1841:          sta  $68
5837 18 1842:          clc
5838 6928 1843:          adc  #40
583a 855e 1844:          sta  $5e
583c a559 1845:          lda  $58+1
583e 8569 1846:          sta  $68+1
5840 6900 1847:          adc  #0
5842 855f 1848:          sta  $5e+1
5844 a000 1849:          ldy  #0
5846 a203 1850:          ldx  #3
5848 b15e 1851: scloop:      lda  ($5e),y
584a 9168 1852:          sta  ($68),y
584c c8 1853:          iny
584d d0f9 1854:          bne  scloop
584f e65f 1855:          inc  $5e+1

```

```

5851 e669 1856:      inc    $68+1
5853 ca 1857:      dex
5854 d0f2 1858:      bne    scloop
5856 b15e 1859: scloop2:  lda    ($5e),y
5858 9168 1860:      sta    ($68),y
585a c8 1861:      iny
585b c098 1862:      cpy    #152
585d 90f7 1863:      bcc    scloop2
585f a558 1864:      lda    $58
5861 18 1865:      clc
5862 6998 1866:      adc    #920&$ff
5864 855e 1867:      sta    $5e
5866 a559 1868:      lda    $58+1
5868 6903 1869:      adc    #920/256
586a 855f 1870:      sta    $5e+1
586c a000 1871:      ldy    #0
586e 98 1872:      tya
586f 915e 1873: clear:  sta    ($5e),y
5871 c8 1874:      iny
5872 c028 1875:      cpy    #40
5874 90f9 1876:      bcc    clear
5876 aef002 1877:      ldx    752
5879 d006 1878:      bne    nocurs3
587b a980 1879:      lda    #$80
587d a455 1880:      ldy    $55
587f 915e 1881:      sta    ($5e),y
5881 a900 1882: nocurs3:  lda    #0
5883 855d 1883:      sta    $5d
5885 a55e 1884:      lda    $5e
5887 18 1885:      clc
5888 6555 1886:      adc    $55
588a 855e 1887:      sta    $5e
588c a55f 1888:      lda    $5e+1
588e 6900 1889:      adc    #0
5890 855f 1890:      sta    $5e+1
5892 60 1891:      rts
5893 a554 1892: get_adr:  lda    $54
5895 0a 1893:      asl
5896 a8 1894:      tay
5897 a558 1895:      lda    $58
5899 18 1896:      clc
589a 797157 1897:      adc    scr_offset,y
589d 855e 1898:      sta    $5e
589f a559 1899:      lda    $58+1
58a1 797257 1900:      adc    scr_offset+1,y
58a4 855f 1901:      sta    $5e+1
58a6 a55e 1902:      lda    $5e
58a8 18 1903:      clc
58a9 6555 1904:      adc    $55
58ab 855e 1905:      sta    $5e
58ad a55f 1906:      lda    $5e+1
58af 6900 1907:      adc    #0
58b1 855f 1908:      sta    $5e+1

```

```

58b3 60 1909: rts
58b4 1910: char_table:
58b4 1b1c1d 1911: dc.b 27,28,29,30,31,125,
58bc 9c9d9e 1912: dc.b 156,157,158,159,253
58c4 a20b 1913: putlocal: ldx #11
58c6 8e4203 1914: stx $0342
58c9 a200 1915: ldx #0
58cb 8e4803 1916: stx $0348
58ce 8e4903 1917: stx $0349
58d1 4c56e4 1918: jmp $e456
58d4 1919: ;
58d4 1920: ;
58d4 1921: ;
58d4 1922: ;
58d4 1923: ;-----;
58d4 1924: ; JSR ECHOS ;
58d4 1925: ;LOAD THE LOW BYTE INTO PRPTR ;
58d4 1926: ;LOAD THE HIGH BYTE INTO PRPTR+1;
58d4 1927: ;JSR ECHOS dc.b $00,"NAME" ;
58d4 1928: ;-----;
58d4 1929: ;
58d4 1930: prints:
58d4 1931: echos:
58d4 48 1932: pha
58d5 48 1933: pha
58d6 18 1934: clc
58d7 ad0659 1935: lda prptr
58da 8df658 1936: sta .b+1
58dd 8dff58 1937: sta .c+1
58e0 ad0759 1938: lda prptr+1
58e3 8df758 1939: sta .b+2
58e6 8d0059 1940: sta .c+2
58e9 a900 1941: lda #$00
58eb 8d0859 1942: sta prptr+2
58ee ac0859 1943: .a: ldy prptr+2
58f1 c8 1944: iny
58f2 8c0859 1945: sty prptr+2
58f5 b9ffff 1946: .b: lda $ffff,y
58f8 20ca56 1947: jsr sprint
58fb ac0859 1948: ldy prptr+2
58fe cfffff 1949: .c: cpy $ffff
5901 90eb 1950: bcc .a
5903 68 1951: pla
5904 68 1952: pla
5905 60 1953: rts
5906 1954: ;PR_PTR
5906 000000 1955: prptr: dc.b 0,0,0
5909 1956: ;
5909 1957: ;
5909 1958: ;-----;
5909 1959: ; JSR GETSTRING ;
5909 1960: ;LOAD THE LOW BYTE INTO sptr ;
5909 1961: ;THE HIGH BYTE INTO sptr+1 ;

```

```

5909      1962: ;LOAD THE MAX LENGHT INTO Y REG.;
5909      1963: ;-----;
5909      1964: ;
5909      1965: ;
5909      1966: getstring:
5909 a99b 1967:          lda  #$9b
590b 8d5459 1968:          sta  x1_get+1
590e 8c6359 1969:          sty  x_get+1
5911 e0ff 1970:          cpx  #$ff
5913 d003 1971:          bne  wq_get
5915 8e5459 1972:          stx  x1_get+1
5918 ad8d59 1973: wq_get:          lda  sptr
591b 8d5159 1974:          sta  g_get+1
591e 8d7759 1975:          sta  b_get+1
5921 8d5c59 1976:          sta  s_get+1
5924 8d7a59 1977:          sta  w_get+1
5927 8d6c59 1978:          sta  t_get+1
592a 8d7059 1979:          sta  back_s+1
592d ad8e59 1980:          lda  sptr+1
5930 8d5259 1981:          sta  g_get+2
5933 8d7859 1982:          sta  b_get+2
5936 8d5d59 1983:          sta  s_get+2
5939 8d7b59 1984:          sta  w_get+2
593c 8d6d59 1985:          sta  t_get+2
593f 8d7159 1986:          sta  back_s+2
5942 a001 1987:          ldy  #$01
5944 18 1988: get_loop:      clc
5945 20f856 1989:          jsr  getkey
5948 c91c 1990:          cmp  #$1c      ; C
594a 9004 1991:          bcc  g_get
594c c920 1992:          cmp  #$20
594e 90f4 1993:          bcc  get_loop
5950 99ffff 1994: g_get:          sta  $ffff,y
5953 c99b 1995: x1_get:          cmp  #$9b
5955 f011 1996:          beq  g_exit
5957 c97e 1997:          cmp  #$7e
5959 f014 1998:          beq  back_s
595b eeffff 1999: s_get:          inc  $ffff
595e 20ca56 2000:          jsr  sprint
5961 c8 2001:          iny
5962 c0ff 2002: x_get:          cpy  #$ff
5964 f002 2003:          beq  g_exit
5966 d0dc 2004:          bne  get_loop
5968 20ca56 2005: g_exit:          jsr  sprint
596b eeffff 2006: t_get:          inc  $ffff
596e 60 2007:          rts
596f adffff 2008: back_s:          lda  $ffff
5972 c900 2009:          cmp  #$00
5974 f0ce 2010:          beq  get_loop
5976 ceffff 2011: b_get:          dec  $ffff
5979 adffff 2012: w_get:          lda  $ffff
597c c900 2013:          cmp  #$00
597e 90c4 2014:          bcc  get_loop

```

```

5980 98  2015:          tya
5981 88  2016:          dey
5982 a97e 2017:          lda  #$7e
5984 20ca56 2018:        jsr  sprint
5987 c900 2019:          cmp  #$00
5989 d0b9 2020:          bne  get_loop
598b f0b7 2021:          beq  get_loop
598d     2022:
598d     2023:
598d 0000 2024: sptr:          dc.w  0
598f     2025:
598f     2026:
598f     2027: my_d_ptr:
598f 0000 2028: dptr:          dc.w  0
5991 000000 2029: test_byte:      dc.b  0,0,0,0
5995     2030: ;
5995     2031: ;
5995     2032: ;-----;
5995     2033: ;COMPARE STRING'S sptr TO DPTR;
5995     2034: ;-----;
5995     2035: ;
5995 18  2036: cmpstr:          clc
5996 a901 2037:          lda  #$01
5998 8d9459 2038:          sta  test_byte+3
599b ad8d59 2039:          lda  sptr
599e 8dde59 2040:          sta  c_sptr+1
59a1 8dce59 2041:          sta  cmp_word+1
59a4 8dc659 2042:          sta  cmp_check1+1
59a7 ad8e59 2043:          lda  sptr+1
59aa 8ddf59 2044:          sta  c_sptr+2
59ad 8dcf59 2045:          sta  cmp_word+2
59b0 8dc759 2046:          sta  cmp_check1+2
59b3 ad8f59 2047:          lda  dptr
59b6 8de759 2048:          sta  c_dptr+1
59b9 8dc959 2049:          sta  cmp_check2+1
59bc ad9059 2050:          lda  dptr+1
59bf 8de859 2051:          sta  c_dptr+2
59c2 8dca59 2052:          sta  cmp_check2+2
59c5 adffff 2053: cmp_check1:      lda  $ffff
59c8 cdffff 2054: cmp_check2:      cmp  $ffff
59cb d026 2055:          bne  w_cmpstr
59cd adffff 2056: cmp_word:          lda  $ffff
59d0 8d9359 2057:          sta  test_byte+2
59d3 a000 2058:          ldy  #$00
59d5 c8  2059: cmp_loop:          iny
59d6 18  2060:          clc
59d7 cc9359 2061:          cpy  test_byte+2
59da f01c 2062:          beq  cmp_exit
59dc 18  2063:          clc
59dd b9ffff 2064: c_sptr:          lda  $ffff,y
59e0 201857 2065:          jsr  locase
59e3 8d9159 2066:          sta  test_byte
59e6 b9ffff 2067: c_dptr:          lda  $ffff,y

```



```

59e9 201857 2068:          jsr  locase
59ec cd9159 2069:          cmp  test_byte
59ef d002  2070:          bne  w_cmpstr
59f1 f0e2  2071:          beq  cmp_loop
59f3      2072: w_cmpstr:
59f3 a900  2073:          lda  #$00
59f5 8d9459 2074:          sta  test_byte+3
59f8 ad9459 2075: cmp_exit:  lda  test_byte+3
59fb c901  2076:          cmp  #$01
59fd 60   2077:          rts
59fe      2078: ;
59fe      2079: ;
59fe      2080: ;-----;
59fe      2081: ; LENGHT TO 0 BYTE STRING      ;
59fe      2082: ; SRC STRING sptr DES STRING DPTR;
59fe      2083: ;-----;
59fe      2084: ;
59fe      2085: toascii:
59fe ad8d59 2086:          lda  sptr
5a01 8d2f5a 2087:          sta  src_asc+1
5a04 8d275a 2088:          sta  src_hold+1
5a07 ad8e59 2089:          lda  sptr+1
5a0a 8d305a 2090:          sta  src_asc+2
5a0d 8d285a 2091:          sta  src_hold+2
5a10 ad8f59 2092:          lda  dptr
5a13 8d2a5a 2093:          sta  des_hold+1
5a16 8d385a 2094:          sta  src_e+1
5a19 ad9059 2095:          lda  dptr+1
5a1c 8d2b5a 2096:          sta  des_hold+2
5a1f 8d395a 2097:          sta  src_e+2
5a22 a001  2098:          ldy  #$01
5a24 a200  2099:          ldx  #$00
5a26 b9ffff 2100: src_hold:  lda  $ffff,y
5a29 9dffff 2101: des_hold:  sta  $ffff,x
5a2c c8     2102:          iny
5a2d e8     2103:          inx
5a2e ccffff 2104: src_asc:   cpy  $ffff
5a31 f002  2105:          beq  src_exit
5a33 d0f1  2106:          bne  src_hold
5a35 a900  2107: src_exit:  lda  #$00
5a37 9dffff 2108: src_e:     sta  $ffff,x
5a3a 60     2109:          rts
5a3b      2110: ;
5a3b      2111: ;-----;
5a3b      2112: ; 0 TO LENGHT STRING;
5a3b      2113: ;-----;
5a3b      2114: tostr:
5a3b ad8d59 2115:          lda  sptr
5a3e 8d675a 2116:          sta  tostr_sr+1
5a41 ad8e59 2117:          lda  sptr+1
5a44 8d685a 2118:          sta  tostr_sr+2
5a47 ad8f59 2119:          lda  dptr
5a4a 8d6e5a 2120:          sta  tostr_dr+1

```

```

5a4d 8d735a 2121:      sta  tostr_hold+1
5a50 8d625a 2122:      sta  tostr_ho+1
5a53 ad9059 2123:      lda  dptr+1
5a56 8d6f5a 2124:      sta  tostr_dr+2
5a59 8d745a 2125:      sta  tostr_hold+2
5a5c 8d635a 2126:      sta  tostr_ho+2
5a5f a000 2127:      ldy  #$00
5a61 8cffff 2128: tostr_ho:      sty  $ffff
5a64 a201 2129:      ldx  #$01
5a66 b9ffff 2130: tostr_sr:      lda  $ffff,y
5a69 c900 2131:      cmp  #$00
5a6b f00c 2132:      beq  tostr_exit
5a6d 9dffff 2133: tostr_dr:      sta  $ffff,x
5a70 c8 2134:      iny
5a71 e8 2135:      inx
5a72 eeffff 2136: tostr_hold:      inc  $ffff
5a75 f0ef 2137:      beq  tostr_sr
5a77 d0ed 2138:      bne  tostr_sr
5a79 2139: tostr_exit:
5a79 60 2140:      rts
5a7a 2141: ;
5a7a 2142: ;
5a7a 2143: ;-----;
5a7a 2144: ; JSR CONCAT APPENDS;
5a7a 2145: ; SPRT TO DPTR  ;
5a7a 2146: ;-----;
5a7a 2147: ;
5a7a 2148: concat:
5a7a ad8d59 2149:      lda  sptr
5a7d 8dab5a 2150:      sta  con_s+1
5a80 8db65a 2151:      sta  con_hold+1
5a83 ad8e59 2152:      lda  sptr+1
5a86 8dac5a 2153:      sta  con_s+2
5a89 8db75a 2154:      sta  con_hold+2
5a8c ad8f59 2155:      lda  dptr
5a8f 8dae5a 2156:      sta  con_d+1
5a92 8da55a 2157:      sta  con_de+1
5a95 8db15a 2158:      sta  con_dd+1
5a98 ad9059 2159:      lda  dptr+1
5a9b 8daf5a 2160:      sta  con_d+2
5a9e 8da65a 2161:      sta  con_de+2
5aa1 8db25a 2162:      sta  con_dd+2
5aa4 acffff 2163: con_de:      ldy  $ffff
5aa7 c8 2164:      iny
5aa8 a201 2165:      ldx  #$01
5aaa bdffff 2166: con_s:      lda  $ffff,x
5aad 99ffff 2167: con_d:      sta  $ffff,y
5ab0 eeffff 2168: con_dd:      inc  $ffff
5ab3 c8 2169:      iny
5ab4 e8 2170:      inx
5ab5 ecffff 2171: con_hold:      cpx  $ffff
5ab8 30f0 2172:      bmi  con_s
5aba 60 2173:      rts

```

```

5abb 2174: ;
5abb 2175: ;
5abb 2176: ;-----;
5abb 2177: ;JSR MOVESTR;
5abb 2178: ;-----;
5abb 2179: movestr:
5abb ad8d59 2180:      lda  sptr
5abe 8ddd5a 2181:      sta  mov_s+1
5ac1 8dda5a 2182:      sta  mov_h+1
5ac4 ad8e59 2183:      lda  sptr+1
5ac7 8dde5a 2184:      sta  mov_s+2
5aca 8ddb5a 2185:      sta  mov_h+2
5acd ad8f59 2186:      lda  dptr
5ad0 8de05a 2187:      sta  mov_d+1
5ad3 ad9059 2188:      lda  dptr+1
5ad6 8de15a 2189:      sta  mov_d+2
5ad9 acffff 2190: mov_h:      ldy  $ffff
5adc b9ffff 2191: mov_s:      lda  $ffff,y
5adf 99ffff 2192: mov_d:      sta  $ffff,y
5ae2 c000 2193:      cpy  #$00
5ae4 f005 2194:      beq  mov_ew
5ae6 88 2195:      dey
5ae7 d0f3 2196:      bne  mov_s
5ae9 f0f1 2197:      beq  mov_s
5aeb 60 2198: mov_ew:      rts
5aec 2199: ;-----
5aec 2200: ;JSR GETCARD
5aec 2201: ;
5aec 2202: ;-----
5aec 002020 2203: gtbyte:      dc.b  $00,"
5af4 202020 2204: qstrin:      dc.b  "  "
5afb 0306 2205: gtwr:      dc.b  $03,$06
5afd 2206: getbyte:
5afd adfb5a 2207:      lda  gtwr
5b00 8d195b 2208:      sta  gt1+1
5b03 a000 2209:      ldy  #$00
5b05 8cec5a 2210:      sty  gtbyte
5b08 cdfb5a 2211:      cmp  gtwr
5b0b f00b 2212:      beq  gt1
5b0d 2213: getcard:
5b0d adfc5a 2214:      lda  gtwr+1
5b10 8d195b 2215:      sta  gt1+1
5b13 a000 2216:      ldy  #$00
5b15 8cec5a 2217:      sty  gtbyte
5b18 c0ff 2218: gt1:      cpy  #$ff
5b1a f03e 2219:      beq  gt3
5b1c 20f856 2220:      jsr  getkey
5b1f 200d57 2221:      jsr  upcase
5b22 c99b 2222:      cmp  #$9b
5b24 f034 2223:      beq  gt3
5b26 c97e 2224:      cmp  #$7e      ; B
5b28 f01b 2225:      beq  gt2
5b2a c930 2226:      cmp  #'0'

```

```

5b2c 90ea 2227:      bcc  gt1
5b2e c93a 2228:      cmp  #'.'
5b30 10e6 2229:      bpl  gt1
5b32 acec5a 2230:     ldy  gtbyte
5b35 c8 2231:      iny
5b36 8cec5a 2232:     sty  gtbyte
5b39 99ec5a 2233:     sta  gtbyte,y
5b3c 20ca56 2234:     jsr  echo
5b3f c9ff 2235:      cmp  #$ff
5b41 d0d5 2236:      bne  gt1
5b43 f0d3 2237:      beq  gt1
5b45 acec5a 2238: gt2:  ldy  gtbyte ; BACKSPACE
5b48 c000 2239:      cpy  #$00
5b4a f0cc 2240:      beq  gt1
5b4c a97e 2241:      lda  #$7e
5b4e 20ca56 2242:     jsr  echo
5b51 ceec5a 2243:     dec  gtbyte
5b54 c9ff 2244:      cmp  #$ff
5b56 d0c0 2245:      bne  gt1
5b58 f0be 2246:      beq  gt1
5b5a 2247: ;
5b5a 2248: ; -MATH FOR CARD CONVERSION-
5b5a 2249: gt3:
5b5a a99b 2250:      lda  #$9b
5b5c acec5a 2251:     ldy  gtbyte
5b5f c8 2252:      iny
5b60 99ec5a 2253:     sta  gtbyte,y
5b63 8cec5a 2254:     sty  gtbyte
5b66 a9ec 2255:      lda  # low gtbyte
5b68 8d8d59 2256:     sta  sptr
5b6b a95a 2257:      lda  # high gtbyte
5b6d 8d8e59 2258:     sta  sptr+1
5b70 a9f4 2259:      lda  # low qstrin
5b72 8d8f59 2260:     sta  dptr
5b75 a95a 2261:      lda  # high qstrin
5b77 8d9059 2262:     sta  dptr+1
5b7a 20fe59 2263:     jsr  toascii
5b7d a9f4 2264:      lda  # low qstrin
5b7f 85f3 2265:     sta  inbuff
5b81 a95a 2266:      lda  # high qstrin
5b83 85f4 2267:     sta  inbuff+1
5b85 a900 2268:      lda  #$00
5b87 85f2 2269:     sta  cix
5b89 2000d8 2270:     jsr  afp
5b8c 20d2d9 2271:     jsr  fpi
5b8f a5d4 2272:      lda  fr0
5b91 a6d5 2273:      ldx  fr0+1
5b93 60 2274:      rts
5b94 2275:
5b94 2276:
5b94 2277: pr_byte:
5b94 a200 2278:      ldx  #$00
5b96 2279: pr_card:

```

```

5b96 85d4 2280:          sta  fr0
5b98 86d5 2281:          stx  fr0+1
5b9a a900 2282:          lda  #$00
5b9c 85f2 2283:          sta  cix
5b9e 20aad9 2284:         jsr  ifp
5ba1 20e6d8 2285:         jsr  fasc
5ba4 a000 2286:         ldy  #$00
5ba6      2287: .a:
5ba6 b1f3 2288:          lda  (inbuff),y
5ba8 08 2289:          php
5ba9 297f 2290:          and  #$7f
5bab c8 2291:          iny
5bac 20ca56 2292:         jsr  echo
5baf 28 2293:          plp
5bb0 10f4 2294:          bpl  .a
5bb2 60 2295:          rts
5bb3      2296: ;
5bb3      2297:
5bb3 00 2298: pbi_interface:      dc.b  0
5bb4      2299:
5bb4      2300:
5bb4      2301: pbi_test:
5bb4 20165c 2302:         jsr  mio_test
5bb7 20f65b 2303:         jsr  kpi_test
5bba 20be5b 2304:         jsr  black_box_test
5bbd 60 2305:          rts
5bbe      2306:
5bbe      2307:
5bbe      2308:
5bbe      2309:
5bbe      2310:
5bbe      2311: ;-----
5bbe      2312: ;
5bbe      2313: ; Checking on the black box if it is present!
5bbe      2314: ;
5bbe      2315: BLACK_BOX_test:
5bbe 78 2316:          SEI
5bbf a902 2317:          LDA  #2
5bc1 8dc0d1 2318:          STA  BB_Sensel
5bc4 a010 2319:          LDY  #$10
5bc6      2320: .a:
5bc6 a202 2321:          LDX  #2
5bc8 b900d8 2322:          LDA  pbi_rom,Y
5bcb ddf25b 2323:          CMP  .data,X
5bce f007 2324:          BEQ  .b
5bd0 c8 2325:          INY
5bd1 d0f3 2326:          BNE  .a
5bd3 4cea5b 2327:          JMP  .no_interface
5bd6 60 2328:          RTS
5bd7      2329: ;
5bd7      2330: .b:
5bd7 c8 2331:          INY
5bd8 ca 2332:          DEX

```

```

5bd9 300a 2333:      BMI  .yes_interface
5bdb b900d8 2334:      LDA  pbi_rom,Y
5bde ddf25b 2335:      CMP  .data,X
5be1 d0e3 2336:      BNE  .a
5be3 f0f2 2337:      BEQ  .b
5be5      2338:
5be5      2339: .yes_interface:
5be5 a902 2340:      lda  #$02          ; y
5be7 8db35b 2341:      sta  pbi_interface
5bea      2342: .no_interface:
5bea d8 2343:      CLD
5beb a900 2344:      LDA  #0
5bed 8dc0d1 2345:      STA  BB_Sensel
5bf0 58 2346:      CLI
5bf1 60 2347:      RTS
5bf2      2348: ;
5bf2 d00829 2349: .data:      dc.b  $D0,$08,$29,$00
5bf6      2350: ;;-----
5bf6      2351: ;
5bf6      2352: ;routine to check for the kpi
5bf6      2353: ;
5bf6      2354: kpi_test:
5bf6 78 2355:      sei
5bf7 a928 2356:      lda  #$28
5bf9 8dfffd1 2357:      sta  PBIBANK      ;tu
5bfc ad01d6 2358:      lda  pbi_ram+1
5bff c943 2359:      cmp  #'C'
5c01 d011 2360:      bne  no_kpi
5c03 ad07d6 2361:      lda  pbi_ram+7
5c06 c94b 2362:      cmp  #'K'          ;as
5c08 d00a 2363:      bne  no_kpi      ;if
5c0a a903 2364:      lda  #$03          ; y
5c0c 8db35b 2365:      sta  pbi_interface
5c0f a900 2366:      lda  #$00
5c11 8dfffd1 2367:      sta  $d1ff          ; a
5c14      2368: no_kpi:
5c14 58 2369:      cli
5c15 60 2370:      rts              ;or
5c16      2371: ;
5c16      2372:
5c16      2373: ;;-----
5c16      2374: ; This Routine Tests for a MIO in system
5c16      2375: ;
5c16      2376: ;
5c16      2377: MIO_TEST:
5c16 8d775c 2378:      sta  mio_acc
5c19 8c785c 2379:      sty  mio_y
5c1c 8e795c 2380:      stx  mio_x
5c1f ba 2381:      tsx
5c20 8e7a5c 2382:      stx  mio_stack
5c23 78 2383:      sei
5c24 a514 2384:      lda  $14
5c26 c514 2385: tammy:      cmp  $14

```

```

5c28 f0fc 2386:      beq  tammy
5c2a 38 2387:      SEC                      ; D
5c2b ade0d1 2388:      LDA  Ld1e0
5c2e 8d755c 2389:      STA  M_HOLD
5c31 ade2d1 2390:      LDA  Ld1e2
5c34 8d765c 2391:      STA  M_HOLD+1
5c37 a900 2392:      LDA  #$00
5c39 8de0d1 2393:      STA  Ld1e0
5c3c a920 2394:      LDA  #$20
5c3e 8de2d1 2395:      STA  Ld1e2
5c41 a943 2396:      LDA  #'C'
5c43 cd01d6 2397:      CMP  pbi_ram+1
5c46 d00a 2398:      BNE  NO_MIO
5c48 cd0ad6 2399:      CMP  pbi_ram+10
5c4b d005 2400:      BNE  NO_MIO
5c4d a901 2401:      lda  #$01
5c4f 8d7b5c 2402:      sta  mio_there
5c52      2403:
5c52      2404: NO_MIO:
5c52 a900 2405:      lda  #$00
5c54 8de0d1 2406:      sta  $d1e0
5c57 8de2d1 2407:      sta  $d1e2
5c5a 58 2408:      CLI
5c5b ae7a5c 2409:      ldx  mio_stack
5c5e 9a 2410:      txs
5c5f ad775c 2411:      lda  mio_acc
5c62 ac785c 2412:      ldy  mio_y
5c65 ae795c 2413:      ldx  mio_x
5c68 ad7b5c 2414:      lda  mio_there
5c6b c900 2415:      cmp  #$00
5c6d f005 2416:      beq  .exit
5c6f a901 2417:      lda  #$01                      ; y
5c71 8db35b 2418:      sta  pbi_interface
5c74      2419:
5c74      2420: .exit:
5c74 60 2421:      rts
5c75      2422:
5c75 0000 2423: M_HOLD:      dc.b  0,0
5c77      2424:
5c77 00 2425: mio_acc:      dc.b  0
5c78 00 2426: mio_y:      dc.b  0
5c79 00 2427: mio_x:      dc.b  0
5c7a 00 2428: mio_stack:   dc.b  0
5c7b 00 2429: mio_there:   dc.b  0
5c7c      2430:
5c7c      2431:
5c7c      2432: ;
5c7c      2433: ;
5c7c      2434: ;~~~~~
5c7c      2435: ; help menu section
5c7c      2436: ;~~~~~
5c7c      2437: help_menu:
5c7c a901 2438:      lda  #$01

```

```

5c7e 8df002 2439:      sta 752
5c81 20a442 2440:      jsr Head_back-3
5c84 204b57 2441:      jsr printsi
5c87 9b 2442:      dc.b $9b
5c88 646972 2443:      dc.b "dirbuild.s.COM ver
5ca3 697320 2444:      dc.b "is Copyright 2012
5cba 577269 2445:      dc.b "Written by Stephen
5cd8 497427 2446:      dc.b "It's intent is to
5cfd 6f6e20 2447:      dc.b "on your Black Box
5d23 666f72 2448:      dc.b "format a new hard
5d45 796f75 2449:      dc.b "you build all your
5d68 617420 2450:      dc.b "at your partitions
5d8f 736563 2451:      dc.b "sectors that are f
5db1 496620 2452:      dc.b "If you do run prog
5dd1 9bff 2453:      dc.b $9b,$ff
5dd3 20e361 2454:      jsr pause
5dd6 204b57 2455:      jsr printsi
5dd9 9b 2456:      dc.b $9b
5dda 507572 2457:      dc.b "Purge_6c will read
5e01 696620 2458:      dc.b "if you a sector is
5e2a 697320 2459:      dc.b "is Purge_6c will r
5e53 243030 2460:      dc.b "$00.", $9B
5e58 9b 2461:      dc.b $9B
5e59 9b 2462:      dc.b $9b
5e5a 486176 2463:      dc.b "Have fun and drop
5e79 546865 2464:      dc.b "The Atari 8-bit is
5e9a 6d616e 2465:      dc.b "many Hours of fun.
5ebc 546869 2466:      dc.b "This DIEHARD machi
5ee1 20666f 2467:      dc.b " form.", $9B
5ee8 9bff 2468:      dc.b $9b,-1
5eea 20e361 2469:      jsr pause
5eed 204b57 2470:      jsr printsi
5ef0 9b 2471:      dc.b $9b
5ef1 546865 2472:      dc.b "The source code to
5f0e 697320 2473:      dc.b "is not to be distr
5f27 576974 2474:      dc.b "Without Written co
5f46 617574 2475:      dc.b "author, Stephen J.
5f61 646972 2476:      dc.b "dirbuild.s.com can
5f87 617320 2477:      dc.b "as part of any pro
5fab 696e63 2478:      dc.b "including sharewar
5fc6 596f75 2479:      dc.b "You are permitted
5fe8 616e79 2480:      dc.b "anyone you WISH!!"
5ffb 20e361 2481:      jsr pause
5ffe 204b57 2482:      jsr printsi
6001 9b2053 2483:      dc.b $9b," Should a comp
6025 492077 2484:      dc.b "I will provide a s
6049 636f73 2485:      dc.b "cost $100.00 US Do
606b 657863 2486:      dc.b "exception of Video
608b 52696e 2487:      dc.b "Ringquist. Atari u
60af 697420 2488:      dc.b "it FREE of charge
60cb 424253 2489:      dc.b "BBS Systems", $9b,-
60d8 20e361 2490:      jsr pause
60db 204b57 2491:      jsr printsi

```



```

60de 9b9b 2492:          dc.b  $9b,$9b
60e0 496620 2493:         dc.b  "If you wish to hav
60ff 636f64 2494:         dc.b  "code then contact:
6113 202020 2495:         dc.b  "   Stephen J.
612c 202020 2496:         dc.b  "   3321 Jonath
6146 202020 2497:         dc.b  "   Augusta, Ga
6160 202020 2498:         dc.b  "   sjcarden@be
617f      2499: ;
617f 4f7220 2500:         dc.b  "Or send a donation
61a1 746865 2501:         dc.b  "the above address.
61b5 204b57 2502:         jsr  printsi
61b8 9b9b 2503:         dc.b  $9b,$9b
61ba 205072 2504:         dc.b  " Press "
61c1 dbd2e5 2505:         dc.b  <+128>,"[Return]"
61c9 20616e 2506:         dc.b  " any other key ",$
61da 20f856 2507:         jsr  getkey
61dd a900 2508:         lda  #$00
61df 8df002 2509:         sta  752
61e2 60 2510:          rts
61e3      2511:
61e3      2512:
61e3      2513:
61e3      2514: ;
61e3      2515: ;          a more command
61e3      2516: pause:
61e3 204b57 2517:         jsr  printsi
61e6 dba0ad 2518:         dc.b  <+128>,"[ -More- ]"
61f0 ff 2519:          dc.b  -1
61f1 20f856 2520:         jsr  getkey
61f4 204b57 2521:         jsr  printsi
61f7 7e7e7e 2522:         dc.b  "~~~~~",$FF
6205 60 2523:          rts
6206      2524: ;
6206      2525:
6206 00 2526: stack_save:      dc.b  0
6207      2527:
6207      2528:
6207      2529:
6207      2530: par_drv_letter:
6207 016e 2531:          dc.b  1,"n"
6209      2532: par_sector_size:
6209 00 2533:          dc.b  0
620a      2534: par_sector_size_128:
620a 043132 2535:          dc.b  4,"128"
620e      2536:          ds.b  25
6227      2537: par_sector_size_256:
6227 043235 2538:          dc.b  4,"256"
622b      2539:          ds.b  25
6244 017c 2540: par_sep:          dc.b  1,"|"
6246      2541: par_sector_number:
6246 003635 2542:          dc.b  0,"65535"
624c      2543:          ds.b  25
6265      2544: ;

```

```

6265      2545: ;
6265      2546: ; ibm drive letterpath*.atr|65535|256|8
6265      2547: ;
6265 00    2548: par_Name:          dc.b  0
6266      2549: cmdtab2:          ds.b  256
6366      2550: ;
6366      2551: ;
6366      2552: ;-----
6366      2553: ;This is the actual directory formatter.
6366      2554: ;-----
6366      2555: ;
6366      2556: ;
6366      2557: ;
6366      2558: ;This is the actual directory formatter.
6366      2559: ;
6366      2560: ;
6366      2561: ;
6366      2562: ;
7ffd      2563:          .org  $7ffd          ; w
7ffd      2564: ;
7ffd 4c8081 2565:          jmp  START_IT          ;di
8000      2566:
8000      2567: Split:
8000      2568: ;#####
8000      2569: ;
8000      2570: ;
8000      2571: ;  =====
8000      2572: ;  Stage 1 boot program
8000      2573: ;  =====
8000      2574: ;
8000      2575:
8000      2576: initz1:
8000      2577: ;
8000      2578: BOOTS:
8000 00    2579:          dc.b  $00
8001 03    2580:          dc.b  $03
8002 00    2581:          dc.b  $00
8003 30    2582:          dc.b  $30
8004 e007 2583:          cpx  #$07
8006 4c8080 2584: w_f01:    JMP  LOADER          ;$3080
8009      2585: ;
8009      2586: ;sector map main directory
8009      2587: l4009:
8009 0000 2588:          dc.w  0
800b      2589: ; Total number of sectors on disk
800b 0000 2590: l400b:    dc.w  0
800d      2591: ; Number of free sectors on disk
800d 0000 2592: l400d:    dc.w  0
800f      2593: ; Number of bit map sectors
800f 00    2594: l400f:    dc.b  0
8010      2595: ;
8010 0400 2596:          dc.w  4
8012 0000 2597: l4012:    dc.w  0

```

```

8014 0000 2598: L4014:          dc.w  0
8016 202020 2599: l4016:          dc.b  "  "
801e 00 2600: l401e:          dc.b  0
801f 2601: l401f:
801f 00 2602: DENLOA:          dc.b  0
8020 20 2603:          dc.b  $20
8021 06 2604:          dc.b  6
8022 01 2605:          dc.b  1
8023 ff 2606:          dc.b -1
8024 ff 2607:          dc.b -1
8025 00 2608:          dc.b  0
8026 00 2609:          dc.b  0
8027 00 2610: l4027:          dc.b  0
8028 2611: L4028:
8028 0000 2612: FIRMAP:          dc.w  0
802a 00 2613:          dc.b  0
802b 00 2614: l402b:          dc.b  0
802c 00 2615: l402c:          dc.b  0
802d 0000 2616: l402d:          dc.w  0
802f 00 2617: l402f:          dc.b  0
8030 2618:
8030 2619:
8030 457272 2620: NODMSG:          dc.b  "Error: No DOS",$9b
803e 2621: ;
803e 2622: ;
803e 2623: ;  get next sector map.. initz new map #
803e 2624: ;  -----
803e 2625: ;
803e 2626:
803e 2627: LOANM:
803e ad2880 2628: w_f02:          lda  FIRMAP
8041 8d0a03 2629:          sta  SECTOR
8044 ad2980 2630: w_f03:          lda  FIRMAP+1
8047 8d0b03 2631:          sta  SECTOR+1
804a a900 2632:          lda  # low secmap
804c a22f 2633:          ldx  # high secmap
804e 20f180 2634: w_f04:          jsr  READS
8051 ad002f 2635:          lda  secmap
8054 8d2880 2636: w_f05:          sta  FIRMAP
8057 ad012f 2637:          lda  secmap+1
805a 8d2980 2638: w_f06:          sta  FIRMAP+1
805d a004 2639:          ldy  #4
805f 8491 2640:          sty  SECPTR
8061 2641: ;
8061 2642: ;  get next sector number..
8061 2643: ;  -----
8061 2644: ;
8061 2645:
8061 a491 2646: GNXSEC:          ldy  SECPTR
8063 cc1f80 2647: w_f07:          CPY  DENLOA
8066 f0d6 2648:          beq  LOANM
8068 b9002f 2649:          lda  secmap,Y
806b 8d0a03 2650:          sta  SECTOR

```

```

806e b9012f 2651:      lda  secmap+1,Y
8071 8d0b03 2652:      sta  SECTOR+1
8074 c8      2653:      INY
8075 c8      2654:      INY
8076 8491 2655:      sty  SECPTR
8078 60      2656: LRTS:      rts
8079      2657: ;
8079 6ce202 2658: DOINITZ:      JMP  ($2e2)
807c      2659: ;
807c      2660: 1407c:
807c 00      2661:      dc.b  0
807d 00      2662:      dc.b  0
807e 00      2663:      dc.b  0
807f 00      2664:      dc.b  0
8080      2665:
8080 a200 2666: LOADER:      ldx  #0
8082 ad1f80 2667: w_f08:      lda  DENLOA
8085 8591 2668:      sta  SECPTR
8087 8590 2669:      sta  CHPTR
8089 8d0803 2670:      sta  DBYTLO
808c d001 2671:      BNE  ISX
808e e8      2672:      INX
808f 8e0903 2673: ISX:      STX  DBYTHI
8092 200a81 2674: w_f09:      jsr  f_getbyte
8095 8596 2675:      sta  temp
8097 200a81 2676: w_f10:      jsr  f_getbyte
809a 2596 2677:      AND  temp
809c c9ff 2678:      CMP  #$FF
809e d037 2679:      BNE  TYPER
80a0      2680: BEGIN:
80a0 a978 2681: w_lb_1:      lda  # low LRTS
80a2 8de202 2682:      sta  INITAD
80a5 a980 2683: w_hb_1:      lda  # high LRTS
80a7 8de302 2684:      sta  INITAD+1
80aa 200a81 2685: w_f11:      jsr  f_getbyte
80ad 8592 2686:      sta  buf
80af 200a81 2687: w_f12:      jsr  f_getbyte
80b2 8593 2688:      sta  buf+1
80b4 0592 2689:      ORA  buf
80b6 f01c 2690:      beq  STAD
80b8 200a81 2691: w_f13:      jsr  f_getbyte
80bb 38      2692:      SEC
80bc e592 2693:      SBC  buf
80be 48      2694:      PHA
80bf 08      2695:      PHP
80c0 200a81 2696: w_f14:      jsr  f_getbyte
80c3 28      2697:      PLP
80c4 e593 2698:      SBC  buf+1
80c6 8595 2699:      sta  LENG+1
80c8 68      2700:      PLA
80c9 8594 2701:      sta  LENG
80cb      2702:
80cb 206b81 2703: w_f15:      jsr  LOABUF

```

```

80ce 207980 2704: w_f16:      jsr  DOINITZ
80d1 4ca080 2705: w_f17:      jmp   BEGIN
80d4      2706: ;
80d4 6ce002 2707: STAD:      jmp   ($2e0)
80d7      2708: ;
80d7      2709: ;
80d7      2710: ;
80d7      2711: ;   error.. no DOS on diskette
80d7      2712: ;   -----
80d7      2713: ;
80d7      2714: TYPER:
80d7 a930  2715: w_lb_2:      lda   # low NODMSG
80d9 a280  2716: w_hb_2:      ldx   # high NODMSG
80db 8d4403 2717:      sta  ICBAL
80de 8e4503 2718:      STx  ICBAH
80e1 8e4803 2719:      STx  ICBL
80e4 a909  2720:      lda  #9
80e6 8d4203 2721:      sta  ICCOM
80e9 a200  2722:      ldx  #0
80eb 2056e4 2723:      jsr  CIO
80ee      2724:
80ee      2725: FOREV:
80ee 4cee80 2726: w_f18:      jmp   FOREV
80f1      2727: ;
80f1      2728: ;
80f1      2729: ;
80f1      2730: ;   read sector at address
80f1      2731: ;   -----
80f1      2732: ;
80f1      2733: READS:
80f1 a040  2734:      ldy  #$40
80f3 8c0303 2735:      sty  DSTATS
80f6 8d0403 2736:      sta  DBUFLO
80f9 8e0503 2737:      STx  DBUFHI
80fc ad0a03 2738:      lda  SECTOR
80ff 0d0b03 2739:      ORA  SECTOR+1
8102 f0d3  2740:      beq  TYPER
8104 2059e4 2741:      jsr  SIO
8107 30ce  2742:      bmi  TYPER
8109 60    2743:      rts
810a      2744: ;
810a      2745: ;
810a      2746: ;   get byte/burst routine
810a      2747: ;   -----
810a      2748: ;
810a      2749: ;
810a      2750: f_getbyte:
810a a900  2751:      lda  #0
810c 8595  2752:      sta  LENG+1
810e 8594  2753:      sta  LENG
8110      2754:
8110 a690  2755: GETBYT:      ldx  CHPTR
8112 ec1f80 2756: w_f19:      CPx  DENLOA

```

```

8115 f006 2757:          beq  LOANS
8117 bd002e 2758:          lda  datbuf,X
811a e690 2759:          INC  CHPTR
811c 60 2760:          rts
811d 2761: ;
811d 2762: ;
811d 2763: ;
811d 2764: ;    Load next sector in buffer/ or burst if possible
811d 2765: ;    -----
811d 2766: ;
811d 2767: LOANS:
811d 206180 2768: w_f20:      jsr  GNXSEC
8120 a595 2769:          lda  LENG+1
8122 d017 2770:          bne  OKBUR
8124 ad1f80 2771: w_f21:      lda  DENLOA
8127 f004 2772:          beq  NOBUR
8129 a594 2773:          lda  LENG
812b 300e 2774:          bmi  OKBUR
812d 2775:
812d a900 2776: NOBUR:      lda  # low datbuf
812f a22e 2777:          idx  # high datbuf
8131 20f180 2778: w_f22:      jsr  READS
8134 38 2779:          SEC
8135 2690 2780:          ROI  CHPTR
8137 ad002e 2781:          lda  datbuf
813a 60 2782:          rts
813b 2783: ;
813b 2784: ;
813b 2785: ;    Do burst read sector
813b 2786: ;    -----
813b 2787: ;
813b a592 2788: OKBUR:      lda  buf
813d a693 2789:          idx  buf+1
813f 20f180 2790: w_f23:      jsr  READS
8142 a592 2791:          lda  buf
8144 18 2792:          clc
8145 6d0803 2793:          adc  DBYTLO
8148 8592 2794:          sta  buf
814a a593 2795:          lda  buf+1
814c 6d0903 2796:          adc  DBYTHI
814f 8593 2797:          sta  buf+1
8151 38 2798:          SEC
8152 a594 2799:          lda  LENG
8154 ed0803 2800:          SBC  DBYTLO
8157 8594 2801:          sta  LENG
8159 a595 2802:          lda  LENG+1
815b ed0903 2803:          SBC  DBYTHI
815e 8595 2804:          sta  LENG+1
8160 4c1d81 2805: w_f24:      jmp  LOANS
8163 2806: ;
8163 2807: ;
8163 2808: ;
8163 2809: ;    Load entire buffer from file

```

```

8163      2810: ; -----
8163      2811: ;
8163      2812:
8163 a594 2813: LOABUFX:      lda  LENG
8165 d002 2814:              bne  NOB
8167 c695 2815:              DEC  LENG+1
8169 c694 2816: NOB:          DEC  LENG
816b      2817:
816b      2818:
816b      2819: LOABUF:
816b 201081 2820: w_f25:      jsr  GETBYT ; get byte..
816e a000 2821:              ldy  #0
8170 9192 2822:              sta  (buf),Y
8172 e692 2823:              INC  buf
8174 d002 2824:              bne  SK11
8176 e693 2825:              iNC  buf+1
8178 a594 2826: SK11:          lda  LENG
817a 0595 2827:              ORA  LENG+1
817c d0e5 2828:              bne  LOABUFX
817e 60   2829:              rts
817f 00   2830:              brk
8180      2831:
8180      2832: zend1:        ds.b  0
8180      2833:
8180      2834:
8180      2835: BOOTI:  =  SPLIT  ; $4000  ; location of boot sectors
8180      2836:
8180      2837: ;
8180      2838: ;
8180      2839: ;
8180      2840: ;=====
8180      2841: ;   Beginning and initialization
8180      2842: ;=====
8180      2843: ;
8180      2844: ;
8180      2845: START_IT:
8180 20828f 2846:              jsr  rlocate
8183      2847:
8183 a50a 2848: BEGINI:          lda  COMTAB ; calc addr
8185 38   2849:              SEC
8186 e90a 2850:              SBC  # low lsio
8188 8d9481 2851:              sta  XSIO+1
818b a50b 2852:              lda  COMTAB+1
818d e900 2853:              SBC  # high lsio
818f 8d9581 2854:              sta  XSIO+2
8192      2855:
8192 60   2856:              rts
8193      2857:
8193      2858: ;              jmp  FORMAT
8193      2859:
8193      2860:
8193 6c0000 2861: XSIO:           jmp  (0)
8196      2862: ;

```

```

8196      2863: RETURN:
8196      2864: ;          ldx  stackp
8196      2865: ;          TXS
8196 60    2866:          rts
8197      2867: ;
8197      2868: ;
8197      2869: ;=====
8197      2870: ;    Main parameter input
8197      2871: ;=====
8197      2872: ;
8197      2873: format:
8197      2874: ;          tsx
8197      2875: ;          stx  stackp
8197 205d56 2876:          jsr  savexy
819a      2877: ;
819a a202  2878:          ldx  #2
819c a005  2879:          ldy  #5
819e 205856 2880:          jsr  gotoxy
81a1      2881: ;
81a1 203487 2882:          jsr  f_printsi
81a4 9b    2883:          dc.b  $9b
81a5 202020 2884:          dc.b  "  "
81ab cdf5ec 2885:          dc.b  <+128>,"Multi Directory For
81c4 9b    2886:          dc.b  $9B
81c5 202020 2887:          dc.b  "  Written By Stephen J.
81e6 202020 2888:          dc.b  "  For the Black Box all v
8208 9b    2889:          dc.b  $9b
8209 20426c 2890:          dc.b  "  Black Box Partition being
822e 202020 2891:          dc.b  "  "
8233 fcc3f5 2892:          dc.b  <+128>,"|Current Drive==>"
8244 9bff  2893:          dc.b  $9b,-1
8246      2894: ;
8246 206856 2895:          jsr  loadxy
8249 203487 2896:          jsr  f_printsi
824c 9b9b  2897:          dc.b  $9b,$9b
824e 9b9b9b 2898:          dc.b  $9b,$9b,$9b,$9b,$9b,-1
8254      2899: ;
8254 ad0552 2900:          lda  my_srcdrv
8257 8d0103 2901:          sta  DUNIT
825a ad1fd0 2902:          lda  consol
825d c907  2903:          cmp  #7
825f f005  2904:          beq  .hoho
8261 a900  2905:          lda  #$00
8263 8d5756 2906:          sta  override
8266      2907: ;
8266      2908: .hoho:
8266 ad5756 2909:          lda  override
8269 c951  2910:          cmp  #'Q'
826b f04f  2911:          beq  .c
826d      2912: ;
826d 203487 2913:          jsr  f_printsi
8270 9b9b  2914:          dc.b  $9B,$9B
8272 496e69 2915:          dc.b  "Initialize... Are You Sure

```



57

```

830f      2969:
830f ad1fd0 2970:      lda  consol
8312 c907  2971:      cmp  #7      ;
8314 f00b  2972:      beq  .tam
8316 ad1fd0 2973:      lda  consol
8319 c905  2974:      cmp  #$05  ; select key
831b f012  2975:      beq  .damn
831d a908  2976:      lda  #08
831f d016  2977:      bne  .usp
8321 adce89 2978: .tam:      lda  names+3
8324 c900  2979:      cmp  #$00
8326 d03d  2980:      bne  .e
8328 adcd89 2981:      lda  names+2
832b c900  2982:      cmp  #$00
832d d036  2983:      bne  .e
832f      2984: .damn:
832f a9ff  2985:      lda  #$ff  ; loading an ff in
8331 8dce89 2986:      sta  names+3
8334 8dcd89 2987:      sta  names+2
8337      2988:
8337      2989: .usp:
8337 203487 2990:      jsr  f_printsi
833a 9b9b20 2991:      dc.b  "$9b,$9b," Setting Disk to "
834e      2992:
834e add586 2993:      lda  f_end_sector
8351 aed686 2994:      ldx  f_end_sector+1
8354 20f386 2995:      jsr  f_pr_card
8357      2996:
8357      2997:
8357 203487 2998:      jsr  f_printsi
835a 205365 2999:      dc.b  " Sectors",$9b,$9b,-1
8365      3000: ;
8365      3001: .e
8365 a901  3002:      lda  #1
8367 8db789 3003:      sta  DENS
836a c901  3004:      cmp  #1
836c f021  3005:      beq  .f
836e 203487 3006:      jsr  f_printsi
8371 3f3f53 3007:      dc.b  "??Shows Multiple Tracks??"
838c 4c9681 3008:      jmp  RETURN
838f      3009: ;
838f      3010: .f:
838f add386 3011:      lda  real_sector_count
8392      3012: ;      lda  NAMES+3
8392 8db589 3013:      sta  MAXSEC
8395      3014:
8395 add486 3015:      lda  real_sector_count+1
8398      3016: ;      lda  NAMES+2
8398 8db689 3017:      sta  MAXSEC+1
839b      3018:
839b add289 3019:      lda  NAMES+7
839e 8db389 3020:      sta  DENSITY
83a1 a9ff  3021:      lda  #$FF

```

```

83a3 8db489 3022:      sta  SECTORS
83a6 202d89 3023:      jsr  RECALC
83a9 200388 3024:      jsr  WRITE
83ac 203487 3025:      jsr  f_printsi
83af 9b9b 3026:      dc.b  $9b,$9b
83b1 447269 3027:      dc.b  "Drive Initialized...",$9b,
83c7 4c9681 3028:      jmp  RETURN
83ca      3029: ;
83ca      3030: ;
83ca      3031: ;End of the directory formatting
83ca      3032: ;
83ca      3033: ;
83ca      3034:
83ca      3035: get_sector_count:
83ca a931 3036:      lda  #$31
83cc 8d0003 3037:      sta  $300
83cf ad0103 3038:      lda  srcdrv
83d2 8d0103 3039:      sta  $301
83d5 a94e 3040:      lda  #'N'
83d7 8d0203 3041:      sta  $0302
83da a940 3042:      lda  #$40
83dc 8d0303 3043:      sta  $0303
83df a9f0 3044:      lda  # low huh_data
83e1 8d0403 3045:      sta  $0304
83e4 a985 3046:      lda  # high huh_data
83e6 8d0503 3047:      sta  $0305
83e9 a90c 3048:      lda  #$0c
83eb 8d0803 3049:      sta  $0308
83ee a900 3050:      lda  #$00
83f0 8d0903 3051:      sta  $0309
83f3 8d0a03 3052:      sta  $030a
83f6 8d0b03 3053:      sta  $030b
83f9 2059e4 3054:      jsr  $e459
83fc ad0303 3055:      lda  $0303
83ff c901 3056:      cmp  #$01
8401 f001 3057:      beq  .huh_1
8403 60 3058:      rts
8404      3059: .huh_1:
8404 ad1fd0 3060:      lda  consol
8407 c905 3061:      cmp  #5
8409 f001 3062:      beq  .tam
840b 60 3063:      rts
840c      3064: .tam:
840c a97d 3065:      lda  #$7d
840e 20ca56 3066:      jsr  echo
8411 20a442 3067:      jsr  Head_back-3
8414      3068:
8414 203487 3069:      jsr  f_printsi
8417 9b 3070:      dc.b  $9b
8418 202020 3071:      dc.b  " Drive configuration
8443 adf085 3072:      lda  huh_data
8446 20d985 3073:      jsr  sub_error
8449 203487 3074:      jsr  f_printsi

```

```

844c 6e756d 3075:      dc.b  "number of tracks", $9B, $FF
845e adf185 3076:      lda  huh_data+1
8461 20d985 3077:      jsr  sub_error
8464 203487 3078:      jsr  f_printsi
8467 537465 3079:      dc.b  "Step rate! normaly 1", $9B,
847d adf285 3080:      lda  huh_data+2
8480 20d985 3081:      jsr  sub_error
8483 203487 3082:      jsr  f_printsi
8486 536563 3083:      dc.b  "Sector/Track high byte", $9
849e adf385 3084:      lda  huh_data+3
84a1 20d985 3085:      jsr  sub_error
84a4 203487 3086:      jsr  f_printsi
84a7 536563 3087:      dc.b  "Sector/Track LOW byte", $9B
84be adf485 3088:      lda  huh_data+4
84c1 20d985 3089:      jsr  sub_error
84c4 203487 3090:      jsr  f_printsi
84c7 4d6178 3091:      dc.b  "Max head number", $9B, $FF
84d8 adf585 3092:      lda  huh_data+5
84db 20d985 3093:      jsr  sub_error
84de 203487 3094:      jsr  f_printsi
84e1 44656e 3095:      dc.b  "Density -0=s, 4 double, 8
8501 adf685 3096:      lda  huh_data+6
8504 20d985 3097:      jsr  sub_error
8507 203487 3098:      jsr  f_printsi
850a 427974 3099:      dc.b  "Byte/sector h byte 1=256 0
852a adf785 3100:      lda  huh_data+7
852d 20d985 3101:      jsr  sub_error
8530 203487 3102:      jsr  f_printsi
8533 427974 3103:      dc.b  "Byte/sector l byte 0=256 1
8553 adf885 3104:      lda  huh_data+8
8556 20d985 3105:      jsr  sub_error
8559 203487 3106:      jsr  f_printsi
855c 447269 3107:      dc.b  "Drive present flag -return
857d adf985 3108:      lda  huh_data+9
8580 20d985 3109:      jsr  sub_error
8583 203487 3110:      jsr  f_printsi
8586 6e6f74 3111:      dc.b  "not used", $9B, $FF
8590 adfa85 3112:      lda  huh_data+10
8593 20d985 3113:      jsr  sub_error
8596 203487 3114:      jsr  f_printsi
8599 6e6f74 3115:      dc.b  "not used", $9B, $FF
85a3 adfb85 3116:      lda  huh_data+11
85a6 20d985 3117:      jsr  sub_error
85a9 203487 3118:      jsr  f_printsi
85ac 6e6f74 3119:      dc.b  "not used", $9B
85b5 9b5072 3120:      dc.b  $9b, "Press any key to conti
85d0 209d87 3121:      jsr  get_key
85d3 a97d 3122:      lda  #$7d
85d5 201b87 3123:      jsr  f_ECHO
85d8 60 3124:      rts
85d9 3125: sub_error:
85d9 8def85 3126:      sta  .dan_k
85dc 203487 3127:      jsr  f_printsi

```

```

85df 2024ff 3128:          dc.b  "$", $FF
85e2 adef85 3129:          lda  .dan_k
85e5 20d886 3130:          jsr  f_drive_error
85e8 203487 3131:          jsr  f_printsi
85eb 2020ff 3132:          dc.b  " ", $FF
85ee 60     3133:          rts
85ef      3134: ;
85ef 00     3135: .dan_k:          dc.b  0
85f0 000000 3136: huh_data:        dc.b  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
8603 000000 3137: huh_data_d:      dc.b  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
8616      3138:
8616      3139:
8616      3140:
8616      3141:
8616      3142:
8616      3143:
8616      3144: calc_sector_count:
8616 20ca83 3145:          jsr  get_sector_count
8619 a9ff   3146:          lda  #$ff
861b 8dd786 3147:          sta  sector_density
861e adf585 3148:          lda  huh_data+5
8621 c908   3149:          cmp  #$08
8623 d008   3150:          bne  .a
8625 a902   3151:          lda  #$02
8627 8dd186 3152:          sta  percom_density
862a 4c3e86 3153:          jmp  .b
862d a901   3154: .a:          lda  #$01
862f 8dd186 3155:          sta  percom_density
8632 adf585 3156:          lda  huh_data+5
8635 c900   3157:          cmp  #$00
8637 d005   3158:          bne  .b
8639 a980   3159:          lda  #$80
863b 8dd786 3160:          sta  sector_density
863e      3161:
863e      3162: .b:
863e adf285 3163:          lda  huh_data+2
8641 c900   3164:          cmp  #$00
8643 d00f   3165:          bne  .c
8645 adf385 3166:          lda  huh_data+3
8648 c900   3167:          cmp  #$00
864a d008   3168:          bne  .c
864c a9ff   3169:          lda  #$ff
864e 8df285 3170:          sta  huh_data+2
8651 8df385 3171:          sta  huh_data+3
8654      3172:
8654 adf285 3173: .c:          lda  huh_data+2
8657 8dd086 3174:          sta  sector_per_trac+1
865a adf385 3175:          lda  huh_data+3
865d 8dcf86 3176:          sta  sector_per_trac
8660      3177:
8660 adf085 3178:          lda  huh_data
8663 c903   3179:          cmp  #$03
8665 b005   3180:          bcs  .d

```

```

8667 a901 3181:      lda  # low 1
8669 8df085 3182:      sta  huh_data
866c adf085 3183: .d:      lda  huh_data
866f 8dcd86 3184:      sta  heads
8672 a900 3185:      lda  #$00
8674 8dce86 3186:      sta  heads+1
8677      3187:
8677      3188:
8677 2044da 3189:      jsr  zfr0
867a adcd86 3190:      lda  heads
867d 85d4 3191:      sta  fr0
867f adce86 3192:      lda  heads+1
8682 85d5 3193:      sta  fr0+1
8684 20aad9 3194:      jsr  ifp
8687 a900 3195:      lda  #$00
8689 85f2 3196:      sta  cix
868b 20b6dd 3197:      jsr  fmove
868e adcf86 3198:      lda  sector_per_trac
8691 85d4 3199:      sta  fr0
8693 add086 3200:      lda  sector_per_trac+1
8696 85d5 3201:      sta  fr0+1
8698 a900 3202:      lda  #$00
869a 85f2 3203:      sta  cix
869c 20aad9 3204:      jsr  ifp
869f 20dbda 3205:      jsr  fmul
86a2 20b6dd 3206:      jsr  fmove
86a5 add186 3207:      lda  percom_density
86a8 85d4 3208:      sta  fr0
86aa add286 3209:      lda  percom_density+1
86ad 85d5 3210:      sta  fr0+1
86af a900 3211:      lda  #$00
86b1 85f2 3212:      sta  cix
86b3 20aad9 3213:      jsr  ifp
86b6 20dbda 3214:      jsr  fmul
86b9 20d2d9 3215:      jsr  fpi
86bc a5d4 3216:      lda  fr0
86be a6d5 3217:      ldx  fr0+1
86c0 8dd386 3218:      sta  real_sector_count
86c3 8ed486 3219:      stx  real_sector_count+1
86c6 8dd586 3220:      sta  f_end_sector
86c9 8ed686 3221:      stx  f_end_sector+1
86cc 60 3222:      rts
86cd      3223:
86cd 0000 3224: heads:      dc.w  0
86cf 0000 3225: sector_per_trac:  dc.w  0
86d1 0000 3226: percom_density:      dc.w  0
86d3 0000 3227: real_sector_count:  dc.w  0
86d5 0000 3228: f_end_sector:      dc.w  0
86d7 ff 3229: sector_density:      dc.b  -1
86d8      3230:
86d8      3231:
86d8      3232: ;
86d8      3233: f_drive_error:

```

```

86d8 48 3234:      pha
86d9 4a 3235:      lsr
86da 4a 3236:      lsr
86db 4a 3237:      lsr
86dc 4a 3238:      lsr
86dd 20e386 3239:      jsr  .a
86e0 68 3240:      pla
86e1 290f 3241:      and  #$0f
86e3 c90a 3242: .a:      cmp  #$0a
86e5 b004 3243:      bcs  .b
86e7 0930 3244:      ora  #$30
86e9 d002 3245:      bne  .c
86eb 6936 3246: .b:      adc  #$36
86ed 201b87 3247: .c:      jsr  f_ECHO
86f0 60 3248:      rts
86f1      3249: ;
86f1      3250:
86f1      3251: f_pr_byte:
86f1 a200 3252:      ldx  #$00
86f3      3253: f_pr_card:
86f3 85d4 3254:      sta  fr0
86f5 86d5 3255:      stx  fr0+1
86f7 a900 3256:      lda  #$00
86f9 85f2 3257:      sta  cix
86fb 20aad9 3258:      jsr  ifp
86fe 20e6d8 3259:      jsr  fasc
8701 a000 3260:      ldy  #$00
8703      3261: .a:
8703 b1f3 3262:      lda  (inbuff),y
8705 08 3263:      php
8706 297f 3264:      and  #$7f
8708 c8 3265:      iny
8709 201b87 3266:      jsr  f_ECHO
870c 28 3267:      plp
870d 10f4 3268:      bpl  .a
870f 60 3269:      rts
8710      3270:
8710      3271: ;
8710      3272: ;      Input character with escape processing
8710      3273: ;      -----
8710      3274: ;
8710 209d87 3275: INCH:      jsr  GET_KEY
8713 c91b 3276:      CMP  #27
8715 f001 3277:      BEQ  EXI
8717 60 3278:      rts
8718 4c9681 3279: EXI:      JMP  RETURN
871b      3280: ;
871b      3281: ;
871b      3282: ;
871b      3283: ;=====
871b      3284: ;      print/input routines
871b      3285: ;=====
871b      3286: ;

```

```

871b 3287: ; Print character
871b 3288: ; -----
871b 3289: ; in:
871b 3290: ; A = character to print
871b 3291: ; out:
871b 3292: ; all registers preserved
871b 3293: ;
871b 206687 3294: f_ECHO: jsr SAVER
871e 202487 3295: jsr ZOUT
8721 4c9687 3296: jmp RESALL
8724 3297: ;
8724 8d3087 3298: ZOUT: sta ZTEMP+1
8727 ad4703 3299: lda $0347
872a 48 3300: PHA
872b ad4603 3301: lda $0346
872e 48 3302: PHA
872f a900 3303: ZTEMP: lda #0
8731 a201 3304: ldx #1 ; force NO output t
8733 60 3305: rts
8734 3306: ;
8734 3307: ;
8734 3308: ; Print text
8734 3309: ; -----
8734 3310: ; out:
8734 3311: ; all registers preserved
8734 3312: ; notes:
8734 3313: ; This print routine will print all characters
8734 3314: ; following the JSR PRINT until a delimiter of
8734 3315: ; -1 ($FF) is reached. PRINT will return to the
8734 3316: ; point one byte beyond the delimiter.
8734 3317: ;
8734 206687 3318: f_printsi: jsr SAVER
8737 ba 3319: TSX
8738 bd0501 3320: lda $0105,X
873b 8d4787 3321: sta ZOCH+1
873e bd0601 3322: lda $0106,X
8741 8d4887 3323: sta ZOCH+2
8744 3324: ;
8744 a001 3325: ldy #1
8746 b9ffff 3326: ZOCH: lda $FFFF,Y
8749 c9ff 3327: cmp #$FF
874b f006 3328: beq ZECHO
874d 201b87 3329: jsr f_ECHO
8750 c8 3330: INY
8751 d0f3 3331: bne ZOCH
8753 98 3332: ZECHO: TYA
8754 18 3333: clc
8755 7d0501 3334: adc $0105,X
8758 9d0501 3335: sta $0105,X
875b bd0601 3336: lda $0106,X
875e 6900 3337: adc #0
8760 9d0601 3338: sta $0106,X
8763 4c9687 3339: jmp RESALL

```



```

8766      3340: ;
8766      3341: ;
8766      3342: ;   SAVE & RESTORE registers
8766      3343: ;   -----
8766      3344: ;
8766 08    3345: SAVER:          PHP
8767 48    3346:          PHA
8768 48    3347:          PHA
8769 48    3348:          PHA
876a 08    3349:          PHP
876b 48    3350:          PHA
876c 8a    3351:          TXA
876d 48    3352:          PHA
876e ba    3353:          TSX
876f bd0901 3354:          lda  $0109,X
8772 9d0501 3355:          sta  $0105,X
8775 bd0701 3356:          lda  $0107,X
8778 9d0901 3357:          sta  $0109,X
877b bd0101 3358:          lda  $0101,X
877e 9d0701 3359:          sta  $0107,X
8781 bd0801 3360:          lda  $0108,X
8784 9d0401 3361:          sta  $0104,X
8787 bd0601 3362:          lda  $0106,X
878a 9d0801 3363:          sta  $0108,X
878d 98     3364:          TYA
878e 9d0601 3365:          sta  $0106,X
8791 68     3366:          PLA
8792 aa     3367:          TAX
8793 68     3368:          PLA
8794 28     3369:          PLP
8795 60     3370:          rts
8796      3371: ;
8796 68     3372: RESALL:          PLA
8797 a8     3373:          TAY
8798 68     3374:          PLA
8799 aa     3375:          TAX
879a 68     3376:          PLA
879b 28     3377:          PLP
879c 60     3378:          rts
879d      3379: ;
879d      3380: ;
879d      3381: ;
879d      3382: ;   Get character
879d      3383: ;   -----
879d      3384: ; out:
879d      3385: ;   A   = character (all others preserved)
879d      3386: ;
879d 20a387 3387: GET_KEY:          jsr  ZGETCH
87a0 a900   3388: ZCH:          lda  #0
87a2 60     3389:          rts
87a3      3390: ;
87a3 206687 3391: ZGETCH:          jsr  SAVER
87a6 20af87 3392:          jsr  ZPHG

```

```

87a9 8da187 3393:          sta  ZCH+1
87ac 4c9687 3394:          jmp  RESALL
87af      3395: ;
87af ad25e4 3396: ZPHG:          lda  $E425
87b2 48      3397:          PHA
87b3 ad24e4 3398:          lda  $E424
87b6 48      3399:          PHA
87b7 60      3400:          rts
87b8      3401: ;
87b8      3402: ;
87b8      3403: ;-----
87b8      3404: ;
87b8      3405: ;
87b8      3406: ;   Get line
87b8      3407: ;   -----
87b8      3408: ; in:
87b8      3409: ;   XA   = buffer address
87b8      3410: ;   Y    = maximum number of characters
87b8      3411: ; out:
87b8      3412: ;   @XA  = data, return ends input, a backspace wor
87b8      3413: ;   all registers preserved
87b8      3414: ; notes:
87b8      3415: ;   This routine will first clear the input window
87b8      3416: ;
87b8 206687 3417: GETLINE:      jsr  SAVER
87bb 8c0288 3418:          STY  ZCNT
87be 8dff87 3419:          sta  ZSTOR+1
87c1 8e0088 3420:          STx  ZSTOR+2
87c4      3421: ;
87c4 a000 3422:          ldy  #0
87c6 a920 3423:          lda  #$20
87c8 20fe87 3424: ZCLX:          jsr  ZSTOR
87cb c8      3425:          INY
87cc cc0288 3426:          CPY  ZCNT
87cf d0f7 3427:          bne  ZCLX
87d1      3428: ;
87d1 a000 3429:          ldy  #0
87d3 209d87 3430: ZINLP:          jsr  GET_KEY
87d6 c97e 3431:          cmp  #$7E
87d8 f012 3432:          beq  ZBS
87da c99b 3433:          cmp  #$9B
87dc f01d 3434:          beq  ZENDZ
87de cc0288 3435:          cpy  ZCNT
87e1 f0f0 3436:          beq  ZINLP
87e3 201b87 3437:          jsr  f_ECHO
87e6 20fe87 3438:          jsr  ZSTOR
87e9 c8      3439:          INY
87ea d0e7 3440:          bne  ZINLP
87ec      3441: ;
87ec c000 3442: ZBS:          cpy  #0
87ee f0e3 3443:          beq  ZINLP
87f0 201b87 3444:          jsr  f_ECHO
87f3 88      3445:          DEY

```

```

87f4 a920 3446:      lda  #$20
87f6 20fe87 3447:      jsr  ZSTOR
87f9 d0d8 3448:      bne  ZINLP
87fb 4c9687 3449: ZENDZ:      jmp  RESALL
87fe      3450: ;
87fe 99ffff 3451: ZSTOR:      sta  $FFFF,Y
8801 60 3452:      rts
8802 00 3453: ZCNT:      dc.b 0
8803      3454: ;
8803      3455: ;
8803      3456: ;
8803      3457: ;=====
8803      3458: ;   write/calculate diskette data
8803      3459: ;=====
8803      3460: ;   Write all data to disk
8803      3461: ;   -----
8803 a201 3462: WRITE:      ldx  #1 ; Write out all sec
8805 8e0a03 3463:      STx  SECTOR
8808 ca 3464:      DEX
8809 8e0b03 3465:      STx  SECTOR+1 ; begin at sector
880c      3466: ;
880c a280 3467:      ldx  # high BOOTI
880e a000 3468:      ldy  # low BOOTI
8810 a903 3469:      lda  #3
8812 20a688 3470:      jsr  WRBLOCK ; write boot progra
8815      3471: ;
8815 ac0f80 3472:      ldy  BOOTI+BMD+SDQBM ; # bitmap
8818 8c798a 3473:      sty  TEMPQX
881b      3474: ;
881b ad788a 3475:      lda  DATASEC ; Zero beginning of
881e 4a 3476:      LSR
881f 4a 3477:      LSR
8820 4a 3478:      LSR
8821 aa 3479:      TAX
8822 8e7a8a 3480:      STx  TEMPQX+1
8825 ad788a 3481:      lda  DATASEC
8828 2907 3482:      AND #7
882a a8 3483:      TAY
882b b91989 3484:      lda  MASK,Y
882e 49ff 3485:      EOr  #$FF
8830      3486: ;
8830 9d808b 3487: CLFIR:      sta  BITMAP,X
8833 a900 3488:      lda  #0
8835 ca 3489:      DEX
8836 10f8 3490:      BPl  CLFIR
8838      3491: ;
8838 ae7a8a 3492:      ldx  TEMPQX+1
883b e8 3493:      INX
883c 2c 3494:      dc.b $2C
883d      3495: ;
883d a200 3496: NXB:      ldx  #0 ; clear bit map
883f a9ff 3497:      lda  #$FF
8841 9d808b 3498: SEA:      sta  BITMAP,X

```

```

8844 e8 3499:      INX
8845 d0fa 3500:      bne  SEA
8847      3501: ;
8847 ac798a 3502:      ldy  TEMPQX ; if not last, then
884a 88 3503:      DEY
884b d011 3504:      bne  WRBITX
884d      3505: ;
884d ae7c8a 3506:      ldx  NIBOFF ; offset into nibbl
8850 bd1989 3507:      lda  MASK,X
8853 ae7b8a 3508:      ldx  BYTOFF
8856 9d808b 3509: CLREST:      sta  BITMAP,X ; clear ou
8859 a900 3510:      lda  #0
885b e8 3511:      INX
885c d0f8 3512:      bne  CLREST
885e      3513: ;
885e a901 3514: WRBITX:      lda  #1 ; write one
8860 a28b 3515:      ldx  # high BITMAP
8862 a080 3516:      ldy  # low BITMAP
8864 20a688 3517:      jsr  WRBLOCK
8867 ce798a 3518:      DEc  TEMPQX
886a d0d1 3519:      bne  NXB
886c      3520: ;
886c a900 3521:      lda  #0
886e aa 3522:      TAX
886f 9d808c 3523: CLMAPS:      sta  SCMAP,X ; Clear sec
8872 9d808d 3524:      sta  DATSEC,X
8875 e8 3525:      INX
8876 d0f7 3526:      bne  CLMAPS
8878      3527: ;
8878 ae0980 3528:      ldx  BOOTI+MDD ; Sector map # of
887b e8 3529:      INX
887c 8e848c 3530:      STx  SCMAP+4 ; Put data sector #
887f a901 3531:      lda  #1
8881 a28c 3532:      ldx  # high SCMAP ; Write sector
8883 a080 3533:      ldy  # low SCMAP
8885 20a688 3534:      jsr  WRBLOCK
8888      3535: ;
8888 a20a 3536:      ldx  #11-1 ; Put directory nam
888a bd2289 3537: MVNAM:      lda  NAMTAB,X
888d 9d868d 3538:      sta  DATSEC+DEFNAM,X
8890 ca 3539:      DEX
8891 10f7 3540:      BPl  MVNAM
8893 a928 3541:      lda  #DESSUB+DESUSE
8895 8d808d 3542:      sta  DATSEC+DESTA
8898 a917 3543:      lda  #DDEX
889a 8d838d 3544:      sta  DATSEC+DENBYT
889d a901 3545:      lda  #1
889f a28d 3546:      ldx  # high DATSEC
88a1 a080 3547:      ldy  # low DATSEC
88a3 4ca688 3548:      jmp  WRBLOCK ; Write directory s
88a6      3549: ;
88a6      3550: ;
88a6      3551: ;      Write a block of sectors

```

```

88a6      3552: ; -----
88a6      3553: ;
88a6 8e0503 3554: WRBLOCK:      STx  DBUFHI ; address
88a9 8d7d8a 3555:      sta NUMSECS ; number sectors to
88ac 8c0403 3556:      sty  DBUFLO
88af      3557: ;
88af adb389 3558:      lda  DENSITY ; get density
88b2 ae0b03 3559:      ldx  SECTOR+1
88b5 d009 3560:      bne  ISTHD
88b7 ae0a03 3561:      ldx  SECTOR
88ba e004 3562:      CPx  #4
88bc b002 3563:      bcs  ISTHD
88be a980 3564:      lda  #$80
88c0 8d0803 3565: ISTHD:      sta  DBYTLO
88c3 a200 3566:      ldx  #0
88c5 a8 3567:      TAY
88c6 d002 3568:      bne  ISSNG
88c8 a201 3569:      ldx  #1
88ca 8e0903 3570: ISSNG:      STx  DBYTHI
88cd      3571: ;
88cd a950 3572: REPOUT:      lda  #'P' ; A
88cf 8d0203 3573:      sta  DCOMND
88d2 a980 3574:      lda  #$80
88d4 8d0303 3575:      sta  DSTATS
88d7 209381 3576:      jsr  XSIO
88da 101c 3577:      BPl  OKWRTX
88dc 203487 3578:      jsr  f_printsi
88df 457272 3579:      dc.b "Error writing sector",$9b,
88f5 4c9681 3580:      jmp  RETURN
88f8      3581: ;
88f8 ee0a03 3582: OKWRTX:      INc  SECTOR
88fb d003 3583:      bne  SK1
88fd ee0b03 3584:      INc  SECTOR+1
8900      3585: ;
8900 18 3586: SK1:      clc
8901 ad0403 3587:      lda  DBUFLO
8904 6d0803 3588:      adc  DBYTLO
8907 8d0403 3589:      sta  DBUFLO
890a ad0503 3590:      lda  DBUFHI
890d 6d0903 3591:      adc  DBYTHI
8910 8d0503 3592:      sta  DBUFHI
8913      3593: ;
8913 ce7d8a 3594:      DEc  NUMSECS
8916 d0b5 3595:      bne  REPOUT
8918 60 3596:      rts
8919      3597: ;
8919      3598: ;
8919 0080c0 3599: MASK:      dc.b $00,$80,$C0,$E0
891d f0f8fc 3600:      dc.b $F0,$F8,$FC,$FE,$FF
8922      3601: ;
8922 4d4149 3602: NAMTAB:      dc.b "MAIN  "
892d      3603: ;
892d      3604: ;

```

```

892d    3605: ;    Calculate all sector positions for given density
892d    3606: ;    -----
892d    3607: ;
892d adb589 3608: RECALC:          lda    MAXSEC ; Get last
8930 8d798a 3609:          sta    TEMPQX
8933 adb689 3610:          lda    MAXSEC+1
8936 8d7a8a 3611:          sta    TEMPQX+1
8939    3612: ;
8939 a900 3613:          lda    #0
893b a203 3614:          ldx    #3 ; divide by 8 for #
893d 4e7a8a 3615: REPROL:          LSR    tempqx+1
8940 6e798a 3616:          ROr    TEMPQX
8943 6a 3617:          ROR
8944 ca 3618:          DEX
8945 d0f6 3619:          bne    REPROL
8947 4a 3620:          LSR
8948 4a 3621:          LSR
8949 4a 3622:          LSR
894a 4a 3623:          LSR
894b 4a 3624:          LSR
894c 8d7c8a 3625:          sta    NIBOFF ; Offset into byte
894f    3626: ;
894f ad798a 3627:          lda    TEMPQX
8952 ae7a8a 3628:          ldx    TEMPQX+1 ; Get # of bit map
8955    3629: ;
8955 2cb389 3630:          BIT    DENSITY
8958 1007 3631:          BPl    G2
895a 0a 3632:          ASL
895b 48 3633:          PHA
895c 8a 3634:          TXA
895d 2a 3635:          ROL
895e aa 3636:          TAX
895f 68 3637:          PLA
8960 4a 3638:          LSR
8961    3639: ;
8961 8d7b8a 3640: G2:          sta    BYTOFF
8964 e8 3641:          INX
8965 8e0f80 3642:          STx    BOOTI+BMD+SDQBM ; # bit map
8968 8a 3643:          TXA
8969 18 3644:          clc
896a 6904 3645:          adc    #4
896c 8d0980 3646:          sta    BOOTI+MDD
896f 6902 3647:          adc    #2
8971 8d788a 3648:          sta    DATASEC
8974 8d1480 3649:          sta    BOOTI+BMD+SDDIR
8977    3650: ;          adc    #30
8977 6946 3651:          adc    #70
8979 8d1280 3652:          sta    BOOTI+BMD+SDALLOC
897c 2cbe89 3653:          BIT    CURP
897f 1002 3654:          BPl    ISSOMD
8981 a900 3655:          lda    #0
8983 29f8 3656: ISSOMD:          AND    #$F8
8985    3657: ;

```

```

8985 8d2880 3658:      sta  BOOTI+EXTR+DDMAP
8988      3659: ;
8988 38 3660:      SEC
8989 adb589 3661:      lda  MAXSEC
898c 8d0b80 3662:      sta  BOOTI+BMD+SDTOT
898f ed788a 3663:      SBc  DATASEC
8992 8d0d80 3664:      sta  BOOTI+BMD+SDFRE
8995 adb689 3665:      lda  MAXSEC+1
8998 8d0c80 3666:      sta  BOOTI+BMD+SDTOT+1
899b e900 3667:      SBc  #0
899d 8d0e80 3668:      sta  BOOTI+BMD+SDFRE+1
89a0 adb389 3669:      lda  DENSITY ; set density & tra
89a3 8d1f80 3670:      sta  BOOTI+DPD+SPDEN
89a6 adb989 3671:      lda  TRACKS
89a9 8d1e80 3672:      sta  BOOTI+DPD+SPTRK
89ac      3673: ;
89ac ad0ad2 3674:      lda  $D20A ; get random number
89af 8d2780 3675:      sta  BOOTI+EXTR+DRAND
89b2 60 3676:      rts
89b3      3677: ;
89b3      3678: ;
89b3      3679: ;
89b3 3680: DENSITY:      ds.b 1
89b4 3681: SECTORS:      ds.b 1
89b5 3682: MAXSEC:      ds.b 2
89b7 3683: DENS:      ds.b 1
89b8 3684: ;
89b8 3685: INCH8:      ds.b 1
89b9 3686: TRACKS:      ds.b 1
89ba 3687: TRKINX:      ds.b 1
89bb 3688: DENINX:      ds.b 1
89bc 3689: ;
89bc 3690: COUNT:      ds.b 1
89bd 3691: STACKP:      ds.b 1
89be 3692: CURP:      ds.b 1
89bf 3693: DOSP:      ds.b 1
89c0 3694: DOSPT:      ds.b 10
89ca 3695: DIRP:      ds.b 1
89cb 3696: NAMES:      ds.b 130
8a4d 3697: ENTRY:      ds.b 25
8a66 3698: DOSI:      ds.b 3+8+5
8a76 3699: US:      ds.b 1
8a77 3700: HIGHSP:      ds.b 1
8a78 3701: ;
8a78 3702: DATASEC:      ds.b 1
8a79 3703: TEMPQX:      ds.b 2
8a7b 3704: BYTOFF:      ds.b 1
8a7c 3705: NIBOFF:      ds.b 1
8a7d 3706: NUMSECS:      ds.b 1
8a7e 3707: TABLEN:      ds.b 2
8a80 3708: ;
8a80 3709: FMTBUF:      ds.b 256
8b80 3710: BITMAP:      ds.b 256

```

```

8c80    3711: SCMAP:          ds.b  256
8d80    3712: DATSEC:          ds.b  256
8e80    3713: TABLE:         ds.b  256
8f80    3714: ;
8f80    3715: ;
8f80    3716: ;
8f80    3717: ;-----
8f80    3718: ; Ok the relocation marker is to tell the DOS we are now u
8f80    3719: ;more data at memlow..
8f80    3720: ;
8f80    3721: ;
8f80 00  3722: initz:          dc.b  0    ; start address
8f81    3723:
8f81    3724:
8f81 00  3725: zend:          dc.b  0    ;end address
8f82    3726:
8f82    3727: ;
8f82    3728: ;-----
8f82    3729: ;
8f82    3730: ;          *** END OF HANDLER ***
8f82    3731: ;-----
8f82    3732: ;
8f82    3733: ;   The rest of this relocater Crap Took me a week to f
8f82    3734: ;
8f82    3735: ;   Relocater: main entry
8f82    3736: ;   -----
8f82    3737: ; in:
8f82    3738: ;   segtab = table of segment descriptors
8f82    3739: ;       +00 = relocater table address
8f82    3740: ;       +02 = originate address of block
8f82    3741: ;       +04 = destination originate of block
8f82    3742: ;       +06 = address of block
8f82    3743: ;       +08 = number of bytes in block
8f82    3744: ;       +10 = destination address of block
8f82    3745: ;       +12 = Length of segment descriptor
8f82    3746: ;       +14 = this is the word location adjust mem
8f82    3747: ;
8f82    3748: ;
8f82    3749: ;
8f82    3750: ;
8f82    3751: ;
8f82    3752: ;   reltab = relocater table
8f82    3753: ;       list of address of words to adjust  2
8f82    3754: ;       list of address low bytes to adjust  2
8f82    3755: ;       list of address high bytes to adjust  3
8f82    3756: ;       followed by their low bytes
8f82    3757: ;-----
8f82    3758: rlocate:
8f82    3759:
8f82 adb590 3760:          lda    segtab+segaddress
8f85 8dab90 3761:          sta    segtab+dorgadr
8f88 8db190 3762:          sta    segtab+blkdes
8f8b adb690 3763:          lda    segtab+segaddress+1

```



```

8f8e 8dac90 3764:          sta  segtab+dorgadr+1
8f91 8db290 3765:          sta  segtab+blkdes+1
8f94      3766: ;          jsr  rlocate
8f94      3767:
8f94      3768:
8f94      3769:
8f94 a900 3770:          lda  #0
8f96 8d6190 3771:          sta  segment
8f99      3772:
8f99 ae6190 3773: segloop:      lda  segment
8f9c bda790 3774:          lda  segtab+rettad,x
8f9f 8d4790 3775:          sta  relget+1
8fa2 bda890 3776:          lda  segtab+rettad+1,x
8fa5 8d4890 3777:          sta  relget+2
8fa8 0d4790 3778:          ora  relget+1
8fab d001 3779:          bne  havseg
8fad 60 3780:          rts
8fae      3781:
8fae 38 3782: havseg:          sec
8faf bdab90 3783:          lda  segtab+dorgadr,x
8fb2 fda990 3784:          sbc  segtab+orgadr,x
8fb5 8d6290 3785:          sta  zoffset
8fb8 bdac90 3786:          lda  segtab+dorgadr+1,x
8fbb fdaa90 3787:          sbc  segtab+orgadr+1,x
8fbe 8d6390 3788:          sta  zoffset+1
8fc1      3789:
8fc1 205290 3790: zwordlp:      jsr  getzwp
8fc4 f013 3791:          beq  zlbytlp
8fc6      3792:
8fc6 b1d7 3793:          lda  (zwptr),y
8fc8 18 3794:          clc
8fc9 6d6290 3795:          adc  zoffset
8fcc 91d7 3796:          sta  (zwptr),y
8fce c8 3797:          iny
8fcf b1d7 3798:          lda  (zwptr),y
8fd1 6d6390 3799:          adc  zoffset+1
8fd4 91d7 3800:          sta  (zwptr),y
8fd6 4cc18f 3801:          jmp  zwordlp
8fd9      3802:
8fd9 205290 3803: zlbytlp:      jsr  getzwp
8fdc f00b 3804:          beq  zhbytlp
8fde      3805:
8fde b1d7 3806:          lda  (zwptr),y
8fe0 18 3807:          clc
8fe1 6d6290 3808:          adc  zoffset
8fe4 91d7 3809:          sta  (zwptr),y
8fe6 4cd98f 3810:          jmp  zlbytlp
8fe9      3811:
8fe9 205290 3812: zhbytlp:      jsr  getzwp
8fec f012 3813:          beq  zmovlp
8fee      3814:
8fee 204690 3815:          jsr  relget
8ff1 18 3816:          clc

```

```

8ff2 6d6290 3817:      adc    zoffset
8ff5 b1d7  3818:      lda    (zwptr),y
8ff7 6d6390 3819:      adc    zoffset+1
8ffa 91d7  3820:      sta    (zwptr),y
8ffc 4ce98f 3821:      jmp    zhbytlp
8fff 60    3822:      rts
9000      3823:
9000      3824: zmovlp:
9000      3825:
9000 ad7090 3826:      lda    segment+segmove
9003 c9ff  3827:      cmp    #seg_off
9005 d001  3828:      bne    .zmovlp
9007 60    3829:      rts
9008 ae6190 3830: .zmovlp:      ldx    segment
900b bdad90 3831:      lda    segtab+blkadr,x
900e 8d2990 3832:      sta    zmovfr+1
9011 bdae90 3833:      lda    segtab+blkadr+1,x
9014 8d2a90 3834:      sta    zmovfr+2
9017      3835:
9017 bdb190 3836:      lda    segtab+blkdes,x
901a 8d2c90 3837:      sta    zmovto+1
901d bdb290 3838:      lda    segtab+blkdes+1,x
9020 8d2d90 3839:      sta    zmovto+2
9023      3840:
9023      3841:
9023 bcb090 3842:      ldy    segtab+blkbyt+1,x
9026      3843:
9026 a200  3844:      ldx    #0
9028 bdffff 3845: zmovfr:      lda    $ffff,x
902b 9dffff 3846: zmovto:      sta    $ffff,x
902e e8    3847:      inx
902f d0f7  3848:      bne    zmovfr
9031 ee2a90 3849:      inc    zmovfr+2
9034 ee2d90 3850:      inc    zmovto+2
9037 88    3851:      dey
9038 10ee  3852:      bpl    zmovfr
903a      3853:
903a ad6190 3854:      lda    segment
903d 18    3855:      clc
903e 690c  3856:      adc    #seglen
9040 8d6190 3857:      sta    segment
9043 4c998f 3858:      jmp    segloop
9046      3859:
9046      3860:
9046 adffff 3861: relget:      lda    $ffff
9049 ee4790 3862:      inc    relget+1
904c d003  3863:      bne    nc1
904e ee4890 3864:      inc    relget+2
9051 60    3865: nc1:      rts
9052      3866:
9052 204690 3867: getzwp:      jsr    relget
9055 85d7  3868:      sta    zwptr
9057 204690 3869:      jsr    relget

```

```

905a a000 3870:          ldy  #0
905c 85d8 3871:          sta  zwptr+1
905e 05d7 3872:          ora  zwptr
9060 60   3873:          rts
9061     3874:
9061 00   3875: segment:      dc.b  0
9062 0000 3876: zoffset:      dc.w  0
9064     3877: ;
9064     3878: ;-----
9064     3879: ;
9064     3880: ;          Relocation data table
9064     3881: ;          -----
9064     3882: ;          We resolve Word locations First!
9064     3883: ;
9064     3884: ;  Things like  Lda $5000
9064     3885: ;              Sta $5000
9064     3886: ;              lda $5000,y
9064     3887: ;              jsr $5000
9064     3888: ;              jmp $5000
9064     3889: ;
9064     3890: ;  Well you get the Idea..
9064     3891: ;
9064     3892: ;-----
9064     3893: ;
9064     3894: ; $3000 tabel
9064     3895: ;
9064     3896: rtable1:
9064 0780 3897:          dc.w  w_f01+1
9066 3f80 3898:          dc.w  w_f02+1
9068 4580 3899:          dc.w  w_f03+1
906a 4f80 3900:          dc.w  w_f04+1
906c 5580 3901:          dc.w  w_f05+1
906e 5b80 3902:          dc.w  w_f06+1
9070 6480 3903:          dc.w  w_f07+1
9072 8380 3904:          dc.w  w_f08+1
9074 9380 3905:          dc.w  w_f09+1
9076 9880 3906:          dc.w  w_f10+1
9078     3907:
9078 ab80 3908:          dc.w  w_f11+1
907a b080 3909:          dc.w  w_f12+1
907c b980 3910:          dc.w  w_f13+1
907e c180 3911:          dc.w  w_f14+1
9080 cc80 3912:          dc.w  w_f15+1
9082 cf80 3913:          dc.w  w_f16+1
9084 d280 3914:          dc.w  w_f17+1
9086 ef80 3915:          dc.w  w_f18+1
9088 1381 3916:          dc.w  w_f19+1
908a 1e81 3917:          dc.w  w_f20+1
908c     3918:
908c 2581 3919:          dc.w  w_f21+1
908e 3281 3920:          dc.w  w_f22+1
9090 4081 3921:          dc.w  w_f23+1
9092 6181 3922:          dc.w  w_f24+1

```

```

9094 6c81 3923:          dc.w  w_f25+1
9096      3924:
9096      3925:
9096 0000 3926:          dc.w  0              ; o
9098      3927: ;
9098      3928: ;
9098      3929: ;Ok these are low Byte Locations.. only one byte changed!
9098      3930: ; OK Resolving Low Byte's First
9098      3931: ;
9098 a180 3932:          dc.w  w_lb_1+1
909a d880 3933:          dc.w  w_lb_2+1
909c 0000 3934:          dc.w  0              ; o
909e      3935: ;-----
909e      3936: ;Now to resolve High Bytes..... Only one byte to change!
909e      3937: ; Here is what is needed... location of high byte in .wor
909e      3938: ; Second thing needed is the low byte of the address you a
909e      3939: ;to adjust!
909e      3940: ;
909e a680 3941:          dc.w  w_hb_1+1
90a0 78   3942:          dc.b  LRTS
90a1 da80 3943:          dc.w  w_hb_2+1
90a3 30   3944:          dc.b  NODMSG
90a4      3945:
90a4 0000 3946:          dc.w  0              ; o
90a6 00   3947:          dc.b  0              ; o
90a7      3948:
90a7      3949:
90a7      3950:
90a7      3951: ;-----
90a7      3952: ;End of relocation Table!!!
90a7      3953: ;master table:
90a7      3954: ;
90a7      3955: segtab:
90a7 6490 3956:          dc.w  rtable1        ; a
90a9 0080 3957:          dc.w  initz1         ; a
90ab 0000 3958:          dc.w  0              ; r
90ad 0080 3959:          dc.w  initz1         ; a
90af 8001 3960:          dc.w  zend1-initz1    ; s
90b1 0000 3961:          dc.w  0              ; d
90b3 0000 3962:          dc.w  0              ; E
90b5 0030 3963:          dc.w  $3000          ;se
90b7 ff   3964:          dc.b  seg_off        ; -
90b8      3965:
90b8      3966:
90b8      3967: win_end:
90b8      3968:
90b8      3969:
90b8      3970: ;

```

End assembly: no errors