


```

0000 1: ;-----
0000 2: ;
0000 3: ;      Copyright 2006 Intergraded Logic Systems
0000 4: ;      Source Code is Copyright Stephen J. Car
0000 5: ;
0000 6: ;
0000 7: ;      Please do not share this source!
0000 8: ;-----
0000 9: ;      (General hard disk formatter for real.dos SpartaDO
0000 10: ;
0000 11: ;-----
0000 12: ;      Real.dos
0000 13: ;      This program is to replace
0000 14: ;      percom.com
0000 15: ;
0000 16: ;      Code History.  Init design,  may 19, 2010
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:
0000 27: com_file_name:      .macro
0000 28: ;      dc.b  13,"filename.ext",$9b
0000 29: ;      dc.b  12,"Percom.com  ", $9b
0000 30: ;      .endm
0000 31:
0000 32: ;
0000 33: ;~~~~~
0000 34: ; File revision history. Version Number in hex
0000 35: ;
0000 36: file_ver:      equ  $12
0000 37: ;~~~~~
0000 38: ; the Month in dec for the first time this revision
0000 39: ;was compiled
0000 40: ;
0000 41: c_month:      equ  9
0000 42: ;~~~~~
0000 43: ; the day in dec for the first time this revision
0000 44: ;was compiled
0000 45: c_day:      equ  7
0000 46: ;~~~~~
0000 47: ; the year in dec for the first time this revision
0000 48: ;was compiled
0000 49: c_year:      equ  2012
0000 50: ;~~~~~
0000 51: ; Type of compiler used in program
0000 52: ;
0000 53: ; 0 = unknown Compiler

```

```

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

```

```

0000      107:
0000      108:          .include    equates
0000      109:          .include    globals
0000      110:          .include    macros
0000      111: ;
4000      112:          .org    $4000
4000      113: ;
4000      114:
4000      115: win_start:
4000      116: header_info:
4000      117:          .include    header
42a7      118:
42a7      119:
42a7      120:
42a7 a03f 121:          LDY    #LBUF
42a9      122:
42a9 b10a 123: .comp:          lda    (comtab),y
42ab c99b 124:          cmp    #$9b
42ad f008 125:          beq    .exit
42af c920 126:          cmp    #' '
42b1 f007 127:          beq    .cont
42b3 c8   128:          iny
42b4 4ca942 129:          jmp    .comp
42b7      130:
42b7      131:
42b7 4c2346 132: .exit:          jmp    error
42ba      133:
42ba      134:
42ba      135:
42ba      136:
42ba      137: .cont:
42ba c8   138:          iny
42bb c8   139:          iny
42bc b10a 140:          LDA    (comtab),Y
42be e930 141:          sbc    #48
42c0 8d0103 142:          sta    dunit
42c3 c901 143:          CMP    #$01          ;$2
42c5 9008 144:          bcc    .ea
42c7 c90a 145:          cmp    #$09+1
42c9 b004 146:          bcs    .ea
42cb f005 147:          beq    .c
42cd d003 148:          bne    .c
42cf      149:
42cf 4c2346 150: .ea:          jmp    error
42d2      151:
42d2      152: .c:
42d2      153: ;
42d2      154: START_IT:
42d2 a50a 155:          LDA    COMTAB ; calc address of S
42d4 38   156:          SEC
42d5 e90a 157:          SBC    # low lsio
42d7 8de945 158:          STA    XSIO+1
42da a50b 159:          LDA    COMTAB+1

```

```

42dc e900 160:          SBC  # high lsio
42de 8dea45 161:        STA  XSIO+2
42e1 a96c 162:          lda  #$6c
42e3 8de845 163:        sta  xsio
42e6      164:
42e6      165:
42e6      166: get_sector_count:
42e6 a931 167:          lda  #$31
42e8 8d0003 168:        sta  $300
42eb ad0103 169:        lda  dunit
42ee 8d0103 170:        sta  $301
42f1 a94e 171:          lda  #'N'
42f3 8d0203 172:        sta  $0302
42f6 a940 173:          lda  #$40
42f8 8d0303 174:        sta  $0303
42fb a9fa 175:          lda  # low huh_data
42fd 8d0403 176:        sta  $0304
4300 a944 177:          lda  # high huh_data
4302 8d0503 178:        sta  $0305
4305 a90c 179:          lda  #$0c
4307 8d0803 180:        sta  $0308
430a a900 181:          lda  #$00
430c 8d0903 182:        sta  $0309
430f 8d0a03 183:        sta  $030a
4312 8d0b03 184:        sta  $030b
4315 20e845 185:        jsr  XSIO
4318 ad0303 186:        lda  $0303
431b c901 187:        cmp  #$01
431d f001 188:        beq  .huh_1
431f      189:
431f 60 190:          rts
4320      191: .huh_1:
4320      192: .tam:
4320 205646 193:        jsr  printsi
4323 202020 194:        dc.b  "      Drive configuration
434e adfa44 195:        lda  huh_data
4351 20e344 196:        jsr  sub_error
4354 205646 197:        jsr  printsi
4357 6e756d 198:        dc.b  "number of tracks",$9B,$FF
4369 adfb44 199:        lda  huh_data+1
436c 20e344 200:        jsr  sub_error
436f 205646 201:        jsr  printsi
4372 537465 202:        dc.b  "Step rate! normaly 1",$9B,
4388 adfc44 203:        lda  huh_data+2
438b 20e344 204:        jsr  sub_error
438e 205646 205:        jsr  printsi
4391 536563 206:        dc.b  "Sector/Track high byte",$9
43a9 adfd44 207:        lda  huh_data+3
43ac 20e344 208:        jsr  sub_error
43af 205646 209:        jsr  printsi
43b2 536563 210:        dc.b  "Sector/Track LOW byte",$9B
43c9 adfe44 211:        lda  huh_data+4
43cc 20e344 212:        jsr  sub_error

```

```

43cf 205646 213:      jsr  printsi
43d2 4d6178 214:      dc.b  "Max head number",$9B,$FF
43e3 adff44 215:      lda  huh_data+5
43e6 20e344 216:      jsr  sub_error
43e9 205646 217:      jsr  printsi
43ec 44656e 218:      dc.b  "Density -0=s, 4 double, 8
440c ad0045 219:      lda  huh_data+6
440f 20e344 220:      jsr  sub_error
4412 205646 221:      jsr  printsi
4415 427974 222:      dc.b  "Byte/sector h byte 1=256 0
4435 ad0145 223:      lda  huh_data+7
4438 20e344 224:      jsr  sub_error
443b 205646 225:      jsr  printsi
443e 427974 226:      dc.b  "Byte/sector l byte 0=256 1
445e ad0245 227:      lda  huh_data+8
4461 20e344 228:      jsr  sub_error
4464 205646 229:      jsr  printsi
4467 447269 230:      dc.b  "Drive present flag -return
4488 ad0345 231:      lda  huh_data+9
448b 20e344 232:      jsr  sub_error
448e 202045 233:      jsr  quote
4491      234:
4491 ad0345 235:      lda  huh_data+9
4494 203d46 236:      jsr  echo
4497      237:
4497 205646 238:      jsr  printsi
449a 27206e 239:      dc.b  "" not used",$9B,$FF
44a6 ad0445 240:      lda  huh_data+10
44a9 20e344 241:      jsr  sub_error
44ac      242:
44ac 202045 243:      jsr  quote
44af      244:
44af ad0445 245:      lda  huh_data+10
44b2 203d46 246:      jsr  echo
44b5      247:
44b5 205646 248:      jsr  printsi
44b8 27206e 249:      dc.b  "" not used",$9B,$FF
44c4 ad0545 250:      lda  huh_data+11
44c7 20e344 251:      jsr  sub_error
44ca 202045 252:      jsr  quote
44cd      253:
44cd ad0545 254:      lda  huh_data+11
44d0 203d46 255:      jsr  echo
44d3      256:
44d3      257:
44d3 205646 258:      jsr  printsi
44d6 27206e 259:      dc.b  "" not used",$9B
44e1 ff      260:      dc.b  $FF
44e2 60      261:      rts
44e3      262: sub_error:
44e3 8df944 263:      sta  .dan_k
44e6 205646 264:      jsr  printsi
44e9 2024ff 265:      dc.b  "$",$FF

```

```

44ec adf944 266:      lda  .dan_k
44ef 20eb45 267:      jsr  drive_error
44f2 205646 268:      jsr  printsi
44f5 2020ff 269:      dc.b  " ",$FF
44f8 60     270:      rts
44f9      271: ;
44f9 00     272: .dan_k:      dc.b  0
44fa 000000 273: huh_data:    dc.b  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
450d 000000 274: huh_data_d:  dc.b  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
4520      275:
4520      276:
4520      277:
4520 a927   278: quote:      lda  #"
4522 203d46 279:      jsr  echo
4525 60     280:      rts
4526      281:
4526      282:
4526      283: calc_sector_count:
4526 20e642 284:      jsr  get_sector_count
4529 a9ff   285:      lda  #$ff
452b 8de745 286:      sta  sector_density
452e adff44 287:      lda  huh_data+5
4531 c908   288:      cmp  #$08
4533 d008   289:      bne  .a
4535 a902   290:      lda  #$02
4537 8de145 291:      sta  percom_density
453a 4c4e45 292:      jmp  .b
453d a901   293: .a:      lda  #$01
453f 8de145 294:      sta  percom_density
4542 adff44 295:      lda  huh_data+5
4545 c900   296:      cmp  #$00
4547 d005   297:      bne  .b
4549 a980   298:      lda  #$80
454b 8de745 299:      sta  sector_density
454e      300:
454e      301: .b:
454e adfc44 302:      lda  huh_data+2
4551 c900   303:      cmp  #$00
4553 d00f   304:      bne  .c
4555 adfd44 305:      lda  huh_data+3
4558 c900   306:      cmp  #$00
455a d008   307:      bne  .c
455c a9ff   308:      lda  #$ff
455e 8dfc44 309:      sta  huh_data+2
4561 8dfd44 310:      sta  huh_data+3
4564      311:
4564 adfc44 312: .c:      lda  huh_data+2
4567 8de045 313:      sta  sector_per_trac+1
456a adfd44 314:      lda  huh_data+3
456d 8ddf45 315:      sta  sector_per_trac
4570      316:
4570 adfa44 317:      lda  huh_data
4573 c903   318:      cmp  #$03

```

```

4575 b005 319:      bcs    .d
4577 a901 320:      lda    # low 1
4579 8dfa44 321:     sta    huh_data
457c adfa44 322: .d:      lda    huh_data
457f 8ddd45 323:     sta    heads
4582 a900 324:      lda    #$00
4584 8dde45 325:     sta    heads+1
4587      326:
4587      327:
4587 2044da 328:     jsr    zfr0
458a addd45 329:     lda    heads
458d 85d4 330:      sta    fr0
458f adde45 331:     lda    heads+1
4592 85d5 332:      sta    fr0+1
4594 20aad9 333:     jsr    ifp
4597 a900 334:      lda    #$00
4599 85f2 335:      sta    cix
459b 20b6dd 336:     jsr    fmove
459e addf45 337:     lda    sector_per_trac
45a1 85d4 338:      sta    fr0
45a3 ade045 339:     lda    sector_per_trac+1
45a6 85d5 340:      sta    fr0+1
45a8 a900 341:      lda    #$00
45aa 85f2 342:      sta    cix
45ac 20aad9 343:     jsr    ifp
45af 20dbda 344:     jsr    fmult
45b2 20b6dd 345:     jsr    fmove
45b5 ade145 346:     lda    percom_density
45b8 85d4 347:      sta    fr0
45ba ade245 348:     lda    percom_density+1
45bd 85d5 349:      sta    fr0+1
45bf a900 350:      lda    #$00
45c1 85f2 351:      sta    cix
45c3 20aad9 352:     jsr    ifp
45c6 20dbda 353:     jsr    fmult
45c9 20d2d9 354:     jsr    fpi
45cc a5d4 355:      lda    fr0
45ce a6d5 356:      ldx    fr0+1
45d0 8de345 357:     sta    real_sector_count
45d3 8ee445 358:     stx    real_sector_count+1
45d6 8de545 359:     sta    end_sector
45d9 8ee645 360:     stx    end_sector+1
45dc 60 361:      rts
45dd      362:
45dd 0000 363: heads:      dc.w 0
45df 0000 364: sector_per_trac: dc.w 0
45e1 0000 365: percom_density: dc.w 0
45e3 0000 366: real_sector_count: dc.w 0
45e5 0000 367: end_sector:      dc.w 0
45e7 ff 368: sector_density: dc.b -1
45e8      369:
45e8 4c59e4 370: XSIO:      jMP    siov
45eb      371:

```



```

45eb      372: ;
45eb      373: drive_error:
45eb 48    374:          pha
45ec 4a    375:          lsr
45ed 4a    376:          lsr
45ee 4a    377:          lsr
45ef 4a    378:          lsr
45f0 20f645 379:          jsr  .a
45f3 68    380:          pla
45f4 290f   381:          and  #$0f
45f6 c90a   382: .a:      cmp  #$0a
45f8 b004   383:          bcs  .b
45fa 0930   384:          ora  #$30
45fc d002   385:          bne  .c
45fe 6936   386: .b:      adc  #$36
4600 203d46 387: .c:      jsr  ECHO
4603 60     388:          rts
4604      389: ;
4604      390:
4604      391: pr_byte:
4604 a200   392:          ldx  #$00
4606      393: pr_card:
4606 85d4   394:          sta  fr0
4608 86d5   395:          stx  fr0+1
460a a900   396:          lda  #$00
460c 85f2   397:          sta  cix
460e 20aad9 398:          jsr  ifp
4611 20e6d8 399:          jsr  fasc
4614 a000   400:          ldy  #$00
4616      401: .aprbloop:
4616 b1f3   402:          lda  (inbuff),y
4618 08     403:          php
4619 297f   404:          and  #$7f
461b c8     405:          iny
461c 203d46 406:          jsr  echo
461f 28     407:          plp
4620 10f4   408:          bpl  .aprbloop
4622 60     409:          rts
4623      410:
4623      411:
4623      412:
4623      413: error:
4623 205646 414:          jsr  printsi
4626 9b9b   415:          dc.b $9b,$9b
4628 557361 416:          dc.b "Usage: Percom Dn:"
4639 9b9bff 417:          dc.b $9b,$9b,-1
463c 60     418:          rts
463d      419:
463d      420:
463d      421:
463d      422:
463d      423:
463d      424:

```

```

463d 425: ;
463d 426: ;
463d 427: ;=====
463d 428: ; print/input routines
463d 429: ;=====
463d 430: ;
463d 431: ; Print character
463d 432: ; -----
463d 433: ; in:
463d 434: ; A = character to print
463d 435: ; out:
463d 436: ; all registers preserved
463d 437: ;
463d 208846 438: ECHO: JSR SAVER
4640 204646 439: JSR ZOUT
4643 4cb846 440: JMP RESALL
4646 441: ;
4646 8d5246 442: ZOUT: STA ZTEMP+1
4649 ad4703 443: LDA $0347
464c 48 444: PHA
464d ad4603 445: LDA $0346
4650 48 446: PHA
4651 a900 447: ZTEMP: LDA #0
4653 a200 448: LDX #0 ;1 ; force NO
4655 60 449: RTS
4656 450: ;
4656 451: ;
4656 452: ; Print text
4656 453: ; -----
4656 454: ; out:
4656 455: ; all registers preserved
4656 456: ; notes:
4656 457: ; This print routine will print all characters
4656 458: ; following the JSR PRINT until a delimiter of
4656 459: ; -1 ($FF) is reached. PRINT will return to the
4656 460: ; point one byte beyond the delimiter.
4656 461: ;
4656 208846 462: printsi: JSR SAVER
4659 ba 463: TSX
465a bd0501 464: LDA $0105,X
465d 8d6946 465: STA ZOCH+1
4660 bd0601 466: LDA $0106,X
4663 8d6a46 467: STA ZOCH+2
4666 468: ;
4666 a001 469: LDY #1
4668 b9ffff 470: ZOCH: LDA $FFFF,Y
466b c9ff 471: CMP #$FF
466d f006 472: BEQ ZECHO
466f 203d46 473: JSR ECHO
4672 c8 474: INY
4673 d0f3 475: BNE ZOCH
4675 98 476: ZECHO: TYA
4676 18 477: CLC

```

```

4677 7d0501 478:          ADC  $0105,X
467a 9d0501 479:          STA  $0105,X
467d bd0601 480:          LDA  $0106,X
4680 6900  481:          ADC  #0
4682 9d0601 482:          STA  $0106,X
4685 4cb846 483:          JMP  RESALL
4688      484: ;
4688      485: ;
4688      486: ;   SAVE & RESTORE registers
4688      487: ;   -----
4688      488: ;
4688 08    489: SAVER:          PHP
4689 48    490:          PHA
468a 48    491:          PHA
468b 48    492:          PHA
468c 08    493:          PHP
468d 48    494:          PHA
468e 8a    495:          TXA
468f 48    496:          PHA
4690 ba    497:          TSX
4691 bd0901 498:          LDA  $0109,X
4694 9d0501 499:          STA  $0105,X
4697 bd0701 500:          LDA  $0107,X
469a 9d0901 501:          STA  $0109,X
469d bd0101 502:          LDA  $0101,X
46a0 9d0701 503:          STA  $0107,X
46a3 bd0801 504:          LDA  $0108,X
46a6 9d0401 505:          STA  $0104,X
46a9 bd0601 506:          LDA  $0106,X
46ac 9d0801 507:          STA  $0108,X
46af 98    508:          TYA
46b0 9d0601 509:          STA  $0106,X
46b3 68    510:          PLA
46b4 aa    511:          TAX
46b5 68    512:          PLA
46b6 28    513:          PLP
46b7 60    514:          RTS
46b8      515: ;
46b8 68    516: RESALL:          PLA
46b9 a8    517:          TAY
46ba 68    518:          PLA
46bb aa    519:          TAX
46bc 68    520:          PLA
46bd 28    521:          PLP
46be 60    522:          RTS
46bf      523: ;
46bf      524: ;
46bf      525: ;
46bf      526: ;   Get character
46bf      527: ;   -----
46bf      528: ; out:
46bf      529: ;   A    = character (all others preserved)
46bf      530: ;

```

```

46bf 20c546 531: GET_KEY:      JSR  ZGETCH
46c2 a900 532: ZCH:          LDA  #0
46c4 60 533:                RTS
46c5 534: ;
46c5 208846 535: ZGETCH:          JSR  SAVER
46c8 20d146 536:                JSR  ZPHG
46cb 8dc346 537:                STA  ZCH+1
46ce 4cb846 538:                JMP  RESALL
46d1 539: ;
46d1 ad25e4 540: ZPHG:          LDA  $E425
46d4 48 541:                PHA
46d5 ad24e4 542:                LDA  $E424
46d8 48 543:                PHA
46d9 60 544:                RTS
46da 545: ;
46da 546: ;
46da 547: ;-----
46da 548: ;
46da 549: ;
46da 550: ;   Get line
46da 551: ;   -----
46da 552: ; in:
46da 553: ;   XA   = buffer address
46da 554: ;   Y    = maximum number of characters
46da 555: ; out:
46da 556: ;   @XA  = data, return ends input, a backspace wor
46da 557: ;   all registers preserved
46da 558: ; notes:
46da 559: ;   This routine will first clear the input window
46da 560: ;
46da 208846 561: GETLINE:      JSR  SAVER
46dd 8c2447 562:                STY  ZCNT
46e0 8d2147 563:                STA  ZSTOR+1
46e3 8e2247 564:                STX  ZSTOR+2
46e6 565: ;
46e6 a000 566:                LDY  #0
46e8 a920 567:                LDA  #$20
46ea 202047 568: ZCLX:        JSR  ZSTOR
46ed c8 569:                INY
46ee cc2447 570:                CPY  ZCNT
46f1 d0f7 571:                BNE  ZCLX
46f3 572: ;
46f3 a000 573:                LDY  #0
46f5 20bf46 574: ZINLP:      JSR  GET_KEY
46f8 c97e 575:                CMP  #$7E
46fa f012 576:                BEQ  ZBS
46fc c99b 577:                CMP  #$9B
46fe f01d 578:                BEQ  ZENDZ
4700 cc2447 579:                CPY  ZCNT
4703 f0f0 580:                BEQ  ZINLP
4705 203d46 581:                JSR  ECHO
4708 202047 582:                JSR  ZSTOR
470b c8 583:                INY

```

```
470c d0e7 584:          BNE  ZINLP
470e      585: ;
470e c000 586: ZBS:      CPY  #0
4710 f0e3 587:          BEQ  ZINLP
4712 203d46 588:          JSR  ECHO
4715 88 589:          DEY
4716 a920 590:          LDA  #$20
4718 202047 591:          JSR  ZSTOR
471b d0d8 592:          BNE  ZINLP
471d 4cb846 593: ZENDZ:      JMP  RESALL
4720      594: ;
4720 99ffff 595: ZSTOR:      STA  $FFFF,Y
4723 60 596:          RTS
4724 00 597: ZCNT:      dc.b  0
4725      598: ;
4725      599: ;
4725      600:
4725      601:
4725      602:
4725      603: win_end:    ds.b  0
4725      604: ;
4725      605: ;
4725      606:
4725      607:
4725      608: ;
```

End assembly: no errors