


```
0000 1: ;-----
0000 2: ;      Copyright 2012 Integrated Logic Systems
0000 3: ;      Source Code is Copyright Stephen J. Car
0000 4: ;
0000 5: ;
0000 6: ;      Please do not share this source!
0000 7: ;-----
0000 8: ;      SIO2USB.com
0000 9: ;
0000 10: ;This is a program that Reads the Special Time/Date out of
0000 11: ;      init  9/09/2012   sjc new design
0000 12: ;
0000 13: ;
0000 14: ;
0000 15: ;-----
0000 16: ; Notes: o This source code MAY NOT be placed for download
0000 17: ;      o "mea" is a macro that loads the address of the
0000 18: ;      into the pointer specified by the second field.
0000 19: ;-----
0000 20: ; Assembler: MADMAC (tm) ST Cross Assembler (Atari Corp)
0000 21: ;      XASM  st and IBM VERSIONS
0000 22: ;-----
0000 23:
0000 24: com_file_name:      .macro
0000 25: ;      dc.b  13,"filename.ext",$9b
0000 26: ;      dc.b  12,"SIO2USB.COM ", $9b
0000 27: ;      .endm
0000 28:
0000 29: ;
0000 30: ;-----
0000 31: ; File revision history. Version Number in hex
0000 32: ;
0000 33: file_ver:      equ  $10
0000 34: ;-----
0000 35: ; the Month in dec for the first time this revision
0000 36: ;was compiled
0000 37: ;
0000 38: c_month:      equ  9
0000 39: ;-----
0000 40: ; the day in dec for the first time this revision
0000 41: ;was compiled
0000 42: c_day:      equ  14
0000 43: ;-----
0000 44: ; the year in dec for the first time this revision
0000 45: ;was compiled
0000 46: c_year:      equ  2012
0000 47: ;-----
0000 48: ; Type of compiler used in program
0000 49: ;
0000 50: ; 0 = unknown Compiler
0000 51: ; 1 = xasm
0000 52: ; 2 = mac_65
0000 53: ; 3 = basic
```

```
0000      54: ; 4 = compiled basic xl
0000      55: ; 5 = C65
0000      56: ; 6 = Action
0000      57: ;
0000      58: ; We can add more as time goes on!
0000      59: ;
0000      60: ;
0000      61: ;
0000      62: xasm:          equ   1
0000      63: mac_65:        equ   2
0000      64: basic:         equ   3
0000      65: basicxl:       equ   4
0000      66: c65:          equ   5
0000      67: action:        equ   6
0000      68: ;
0000      69: file_compiler:     equ   xasm
0000      70: ;-----
0000      71: ; is this relocatable code ?
0000      72: ; anyother value other than 1 or 2 would be unknown
0000      73: r..yes:             equ   1
0000      74: r..no:           equ   2
0000      75: ;
0000      76: ;-----
0000      77: ; Gotta define if it can be relocated
0000      78: l_relocatable:       equ   r..no
0000      79: ;
0000      80: ;
0000      81: ;-----
0000      82: ; Gotta define if it can be relocated
0000      83: r..crl:              equ   $7d
0000      84: r..crlf:            equ   $9b
0000      85: r..space:          equ   $20
0000      86: r..bypassdos:       equ   $15 ; bypass dos check
0000      87: r..bpasintro:       equ   $ff
0000      88:
0000      89:
0000      90: l_frstscreenbyte:    equ   r..bypassdos
0000      91: ;-----
0000      92: ; The language the output file is in.
0000      93: ;
0000      94: ;
0000      95: ; 0 =   Undefined
0000      96: ; 1 =   English
0000      97: ; 2 =   German
0000      98: ;
0000      99: r..language:         equ   1
0000     100: ;-----
0000     101: ;Programming equates for the sio2usb!
0000     102: ;
0000     103: ;
0000     104: ;
0000     105: usbF_Read_Only:      equ   $01 ; Read Only
0000     106: usbF_Archive:        equ   $32 ; Archive Bit 0000x
```

```

0000 107: ;-----
0000 108: ;
0000 109: usb_Get_stat:    equ  $00  ; 4.4.1 Get U Statu
0000 110: usb_Set_RTC:     equ  $01  ; 4.4.2 Set Real Ti
0000 111: usb_Read_RTC:    equ  $02  ; 4.4.3 Read Real T
0000 112: usb_CD:          equ  $03  ; 4.4.4 Cha
0000 113: usb_RFATDIR:     equ  $04  ; 4.4.5 Read FAT Di
0000 114: usb_SET_ATARI_DRV_CFG: equ  $05  ; 4.4.6 Set ATARI d
0000 115: usb_Read_ATARI_DRV_CFG: equ  $06  ; 4.4.7 Rea
0000 116: usb_Create_ATR:   equ  $07  ; 4.4.8 Cre
0000 117: usb_Read_Free:    equ  $08  ; 4.4.9 Read amount
0000 118: usb_Delete_fd:    equ  $09  ; 4.4.10 Delete FAT
0000 119: usb_Read_VFAT_dir: equ  $0A  ; 4.4.11 Read VFAT
0000 120: usb_Rename_FAT:   equ  $0B  ; 4.4.12 Re
0000 121: ;usb_?:          equ  $0C  ; unknow
0000 122: usb_Copy_FAT_file: equ  $0D  ; 4.4.13 Copy FAT f
0000 123: usb_Change_Attrib: equ  $0F  ; 4.4.14 Change FAT
0000 124:
0000 125:
0000 126: ;-----
0000 127: ;
0000 128: pgm_sio2usb:      equ  $71      ; Program s
0000 129: siov_read:        equ  $40      ; Read
0000 130: siov_write:       equ  $80      ; write
0000 131: ;-----
0000 132: ;
0000 133: usb_disk1:        equ  1        ; Emulated
0000 134: usb_disk2:        equ  2        ; Emulated
0000 135: usb_disk3:        equ  3        ; Emulated
0000 136: usb_disk4:        equ  4        ; Emulated
0000 137: usb_disk0:        equ  0        ; Emulated
0000 138: ;
0000 139: usb_timeout:      equ  7        ; sio timeo
0000 140: xfersize:        equ  128      ; Buffer si
0000 141: ;
0000 142: usb_inuse:        equ  usb_disk0 ; USB devic
0000 143: siov_do:          equ  siov_write
0000 144: usb_disk:         equ  usb_disk0 ;
0000 145: ;-----
0000 146: ZPF0:            equ  $F0
0000 147: ZPF1:            equ  ZPF0+1
0000 148: SDBASE:          equ  $700
0000 149: SDDEV:           equ  $761
0000 150: SDDATE:          equ  $77B
0000 151: SDTIME:          equ  SDDATE+3
0000 152: EDITRV:          equ  $E400
0000 153:
0000 154:
0000 155:
0000 156: .include equates
0000 157: .include globals
0000 158: .include macros
0000 159:

```

```

4000      160:                .org $4000
4000      161:
4000      162: win_start:
4000      163: header_info:
4000      164:                .include      header
42a7      165: ;
42a7      166: Start:
42a7      167: Command_line:
42a7 a960  168:                lda    #$60
42a9 8da742 169:                sta    start
42ac a50a  170:                LDA    COMTAB
42ae 38    171:                SEC
42af e90a  172:                SBC    # low lsio
42b1 8dc442 173:                STA    XSIO+1
42b4 a50b  174:                LDA    COMTAB+1
42b6 e900  175:                SBC    # high lsio
42b8 8dc542 176:                STA    XSIO+2
42bb a96c  177:                lda    #$6c
42bd 8dc342 178:                sta    xsio
42c0 4c4d43 179:                jmp    read_sio2usb
42c3      180:
42c3      181: ;=====
42c3      182: ;RealDos Lsiov vector
42c3      183: ;
42c3 4cffff 184: XSIO:                jMP    $ffff
42c6      185:
42c6      186:
42c6      187: ;=====
42c6      188: ;it not in the Doc's but this is a length byte String
42c6      189: ;and you have to transfer 128 bytes not 6.
42c6      190: ;=====
42c6      191: ;Time Date Working buffer
42c6      192: ;
42c6 06    193: usb_Time_date:      dc.b  6
42c7 00    194: usb_second:         dc.b  0
42c8 34    195: usb_Minutes:        dc.b  52
42c9 0b    196: usb_Hour:           dc.b  11
42ca 0d    197: usb_day:            dc.b  13
42cb 09    198: usb_month:          dc.b  9
42cc 0c    199: usb_year:           dc.b  12
42cd      200: Data_buffer:        ds.b  128
434d      201: ;
434d      202: ;
434d      203: ;=====
434d      204: ;This sets up the $300 <==> $30b to Read 128 bytes
434d      205: ;First 7 Bytes Hold the Time And Date
434d      206: ;=====
434d      207: read_sio2usb:
434d a971  208:                lda    #pgm_sio2usb
434f 8d0003 209:                sta    $300        ; D
4352 a902  210:                lda    #usb_Read_RTC    ;
4354 8d0203 211:                sta    $0302        ; s
4357 a940  212:                lda    #siov_read

```

```

4359 8d0303 213:      sta  $0303
435c a900  214:      lda  #usb_disk
435e 8d0103 215:      sta  $301          ; D
4361      216:
4361 a9c6  217:      lda  # low usb_Time_date
4363 8d0403 218:      sta  $0304          ; D
4366 a942  219:      lda  # high usb_Time_date
4368 8d0503 220:      sta  $0305          ; D
436b      221:
436b a907  222:      lda  #usb_timeout
436d 8d0603 223:      sta  $0306          ; T
4370 a900  224:      lda  #$00
4372 8d0703 225:      sta  $0307          ; D
4375      226:
4375 a980  227:      lda  #low xfersize
4377 8d0803 228:      sta  $0308          ; N
437a a900  229:      lda  #high xfersize
437c 8d0903 230:      sta  $0309          ; N
437f a900  231:      lda  #$00
4381 8d0a03 232:      sta  $030a          ; D
4384 a900  233:      lda  #$00
4386 8d0b03 234:      sta  $030b          ; D
4389 20c342 235:      jsr  xsio
438c ad0303 236:      lda  $303
438f c901  237:      cmp  #$01
4391 f01a  238:      beq  got_sio2usb
4393      239:
4393 20c643 240: sio2usb_error:  jsr  printsi
4396 9b9b  241:      dc.b $9b,$9b
4398 457272 242:      dc.b "Error $"
439f ff    243:      dc.b -1
43a0 ad0303 244:      lda  $303
43a3 205f45 245:      jsr  drive_error
43a6 20c643 246:      jsr  printsi
43a9 9b9bff 247:      dc.b $9b,$9b,-1
43ac 60    248:      rts
43ad      249:
43ad 20c643 250: got_sio2usb:  jsr  printsi
43b0 9b9b  251:      dc.b $9b,$9b
43b2 53696f 252:      dc.b "Sio2USB "
43ba c4e5f4 253:      dc.b <+128>,"Detected!"
43c3 9b    254:      dc.b $9b
43c4 ff    255:      dc.b -1
43c5 60    256:      rts
43c6      257:
43c6      258:
43c6      259: ;-----
43c6      260: ;printsi Routine macro
43c6      261: ;this is a slow print to the screen
43c6      262: ;usage jsr  printsi
43c6      263: ; .byte $9b," string to be printed",$ff
43c6      264: ;-----
43c6      265: ;

```

```

43c6      266: printsi:
43c6 68    267:          pla
43c7 8dd743 268:          sta .pstr+1
43ca 68    269:          pla
43cb 8dd843 270:          sta .pstr+2
43ce eed743 271: .prsl:    inc .pstr+1
43d1 d003   272:          bne .pstr
43d3 eed843 273:          inc .pstr+2
43d6 adffff 274: .pstr:    lda $ffff
43d9 c9ff   275:          cmp #$ff
43db f006   276:          beq .estri
43dd 201c44 277:          jsr fast_output
43e0 4cce43 278:          jmp .prsl
43e3 add843 279: .estri:   lda .pstr+2
43e6 48     280:          pha
43e7 add743 281:          lda .pstr+1
43ea 48     282:          pha
43eb 60     283:          rts
43ec      284: ;
43ec      285: ;
43ec      286: ;
43ec      287: ;
43ec      288: ;-----
43ec      289: ;actual screen handler
43ec      290: ;screen offset definitions
43ec      291: ;-----
43ec      292: scr_offset:
43ec 0000   293:          dc.w 0
43ee 2800   294:          dc.w 40
43f0 5000   295:          dc.w 80
43f2 7800   296:          dc.w 120
43f4 a000   297:          dc.w 160
43f6 c800   298:          dc.w 200
43f8 f000   299:          dc.w 240
43fa 1801   300:          dc.w 280
43fc 4001   301:          dc.w 320
43fe 6801   302:          dc.w 360
4400 9001   303:          dc.w 400
4402 b801   304:          dc.w 440
4404 e001   305:          dc.w 480
4406 0802   306:          dc.w 520
4408 3002   307:          dc.w 560
440a 5802   308:          dc.w 600
440c 8002   309:          dc.w 640
440e a802   310:          dc.w 680
4410 d002   311:          dc.w 720
4412 f802   312:          dc.w 760
4414 2003   313:          dc.w 800
4416 4803   314:          dc.w 840
4418 7003   315:          dc.w 880
441a 9803   316:          dc.w 920
441c      317: ;
441c      318: echo:

```

```

441c      319: fast_output:
441c 855a 320:      sta $5a
441e ad0407 321:      lda sc.redirect
4421 c902 322:      cmp #print_p
4423 f01f 323:      beq .go_cio
4425      324: .leo:
4425 adff02 325:      lda $02ff      ; CONTROL 1
4428 c900 326:      cmp #$00
442a d0f9 327:      bne .leo
442c      328:
442c a55a 329:      lda $5a
442e c920 330:      cmp #32
4430 9004 331:      bcc .do_look
4432 c97d 332:      cmp #125
4434 9013 333:      bcc .do_here
4436 a000 334: .do_look:      ldy #0
4438 b93f45 335: .lookup:      lda .char_table,y
443b f00c 336:      beq .do_here
443d c55a 337:      cmp $5a
443f f003 338:      beq .go_cio
4441 c8 339:      iny
4442 d0f4 340:      bne .lookup
4444      341:
4444      342:
4444 a55a 343: .go_cio:      lda $5a
4446 4c4f45 344:      jmp putlocal
4449      345:
4449      346:
4449 a000 347: .do_here:      ldy #0
444b a55d 348:      lda $5d
444d 915e 349:      sta ($5e),y
444f a55a 350:      lda $5a
4451 c99b 351:      cmp #$9b
4453 f044 352:      beq .docr
4455 201e45 353: .notcr:      jsr .get_adr
4458 a55a 354:      lda $5a
445a 297f 355:      and #$7f
445c c920 356:      cmp #32
445e 9007 357:      bcc .add64
4460 c960 358:      cmp #96
4462 9009 359:      bcc .sub32
4464 4c7044 360:      jmp .asis
4467 18 361: .add64:      clc
4468 6940 362:      adc #64
446a 4c7044 363:      jmp .asis
446d 38 364: .sub32:      sec
446e e920 365:      sbc #32
4470 245a 366: .asis:      bit $5a
4472 1002 367:      bpl .xxlate
4474 0980 368:      ora #$80
4476 a000 369: .xxlate:      ldy #0
4478 915e 370:      sta ($5e),y
447a e655 371:      inc $55

```



```

447c e663 372:      inc  $63
447e a555 373:      lda  $55
4480 c928 374:      cmp  #40
4482 b019 375:      bcs  .next_row
4484 c8 376:      iny
4485 b15e 377:      lda  ($5e),y
4487 855d 378:      sta  $5d
4489 0980 379:      ora  #$80
448b aef002 380:     ldx  752
448e d002 381:      bne  .nocurs1
4490 915e 382:      sta  ($5e),y
4492 e65e 383: .nocurs1:    inc  $5e
4494 d002 384:      bne  .nooav
4496 e65f 385:     inc  $5e+1
4498 60 386: .nooav:    rts
4499 a900 387: .docr:      lda  #0
449b 8563 388:      sta  $63
449d a552 389: .next_row:    lda  $52
449f 8555 390:      sta  $55
44a1 e654 391:      inc  $54
44a3 a454 392:      ldy  $54
44a5 c018 393:      cpy  #24
44a7 b013 394:      bcs  .scroll
44a9 201e45 395:     jsr  .get_adr
44ac a000 396:      ldy  #0
44ae b15e 397:      lda  ($5e),y
44b0 855d 398:      sta  $5d
44b2 aef002 399:     ldx  752
44b5 d004 400:      bne  .nocurs2
44b7 0980 401:      ora  #$80
44b9 915e 402:      sta  ($5e),y
44bb 60 403: .nocurs2:    rts
44bc 404:
44bc c654 405: .scroll:    dec  $54
44be a558 406:      lda  $58
44c0 8568 407:      sta  $68
44c2 18 408:      clc
44c3 6928 409:      adc  #40
44c5 855e 410:      sta  $5e
44c7 a559 411:      lda  $58+1
44c9 8569 412:      sta  $68+1
44cb 6900 413:      adc  #0
44cd 855f 414:      sta  $5e+1
44cf a000 415:      ldy  #0
44d1 a203 416:      ldx  #3
44d3 b15e 417: .scloop:    lda  ($5e),y
44d5 9168 418:      sta  ($68),y
44d7 c8 419:      iny
44d8 d0f9 420:      bne  .scloop
44da e65f 421:      inc  $5e+1
44dc e669 422:      inc  $68+1
44de ca 423:      dex
44df d0f2 424:      bne  .scloop

```

```

44e1 b15e 425: .sloop2:      lda  ($5e),y
44e3 9168 426:          sta  ($68),y
44e5 c8    427:          iny
44e6 c098 428:          cpy  #152
44e8 90f7 429:          bcc  .sloop2
44ea a558 430:          lda  $58
44ec 18    431:          clc
44ed 6998 432:          adc  #920&$ff
44ef 855e 433:          sta  $5e
44f1 a559 434:          lda  $58+1
44f3 6903 435:          adc  #920/256
44f5 855f 436:          sta  $5e+1
44f7 a000 437:          ldy  #0
44f9 98    438:          tya
44fa 915e 439: .clear:      sta  ($5e),y
44fc c8    440:          iny
44fd c028 441:          cpy  #40
44ff 90f9 442:          bcc  .clear
4501 aef002 443:          ldx  752
4504 d006 444:          bne  .nocurs3
4506 a980 445:          lda  #$80
4508 a455 446:          ldy  $55
450a 915e 447:          sta  ($5e),y
450c a900 448: .nocurs3:    lda  #0
450e 855d 449:          sta  $5d
4510 a55e 450:          lda  $5e
4512 18    451:          clc
4513 6555 452:          adc  $55
4515 855e 453:          sta  $5e
4517 a55f 454:          lda  $5e+1
4519 6900 455:          adc  #0
451b 855f 456:          sta  $5e+1
451d 60    457:          rts
451e a554 458: .get_adr:    lda  $54
4520 0a    459:          asl
4521 a8    460:          tay
4522 a558 461:          lda  $58
4524 18    462:          clc
4525 79ec43 463:          adc  scr_offset,y
4528 855e 464:          sta  $5e
452a a559 465:          lda  $58+1
452c 79ed43 466:          adc  scr_offset+1,y
452f 855f 467:          sta  $5e+1
4531 a55e 468:          lda  $5e
4533 18    469:          clc
4534 6555 470:          adc  $55
4536 855e 471:          sta  $5e
4538 a55f 472:          lda  $5e+1
453a 6900 473:          adc  #0
453c 855f 474:          sta  $5e+1
453e 60    475:          rts
453f      476: .char_table:
453f 1b1c1d 477:          dc.b 27,28,29,30,31,125,126,127

```

```

4547 9c9d9e 478:          dc.b  156,157,158,159,253,254,255
454f      479:
454f      480:
454f      481: putlocal:
454f a20b  482:          ldx  #11
4551 8e4203 483:          stx  $0342
4554 a200  484:          ldx  #0
4556 8e4803 485:          stx  $0348
4559 8e4903 486:          stx  $0349
455c 4c56e4 487:          jmp  $e456
455f      488: ;
455f      489:
455f      490:
455f      491:
455f      492: ;-----
455f      493: ;This prints byte in the
455f      494: ;accumulator in hex
455f      495: ;-----
455f      496: ;
455f 48    497: drive_error:      pha
4560 4a    498:                  lsr
4561 4a    499:                  lsr
4562 4a    500:                  lsr
4563 4a    501:                  lsr
4564 206a45 502:                  jsr  .a
4567 68    503:                  pla
4568 290f  504:                  and  #$0f
456a c90a  505: .a:              cmp  #$0a
456c b004  506:                  bcs  .b
456e 0930  507:                  ora  #$30
4570 d002  508:                  bne  .c
4572 6936  509: .b:              adc  #$36
4574 201c44 510: .c:              jsr  echo
4577 60    511:                  rts
4578      512: ;
4578      513:
4578      514:
4578      515: win_end:          ds.b  0
4578      516: ;
4578      517: ;

```

End assembly: no errors