Apple II ProDOS 1.7 Source Code Listing

UNIVERSAL BOOT LOADER

Written by Apple Computer Inc in 1988

Source File Name: BOOT.SRC

```
0000001:  *<esc>e
0000002:                        sbtl                    'universal boot loader - stage 2'
0000003:                        page
0000004:                        rep                     60
0000005:  * prodos universal boot loader.  this is the second stage boot
0000006:  *   for all apple manufactured apple ii disk drives.
0000007:  * it is located at block zero (0) of a prodos or sos formatted
0000008:  *   disk(ette).  if booted in apple /// native mode, the regular
0000009:  *   sos boot will be attempted.
0000010:                        rep                     60
0000011:                        skp                     1
0000012:                        org                     $800
0000013:  dcmd                  equ                     $42             ;disk command (=1 for read)
0000014:  unit                  equ                     $43             ;(16*slot)+(128*(drive-1))
0000015:  buff                  equ                     $44             ;ram address
0000016:  blok                  equ                     $46             ;disk address
0000017:                        skp                     1
0000018:  dent                  equ                     $48             ;device call entry address.
0000019:  idxl                  equ                     $4a             ;pointer to low page of index block
0000020:  idxh                  equ                     $4c             ;pointer to high page of index block
0000021:  idxp                  equ                     $4e             ;index byte pointer.
0000022:  iobuff                equ                     $60
0000023:                        skp                     1
0000024:  * the following are for disk ii only:
0000025:                        skp                     1
0000026:  dbuf                  equ                     $26
0000027:  slotz                 equ                     $2b
0000028:  oddbits               equ                     $3c
0000029:  sector                equ                     $3d
0000030:  trktmp                equ                     $40
0000031:  track                 equ                     $41
0000032:  prior                 equ                     $50
0000033:  trkn                  equ                     $51
0000034:  rtrycnt               equ                     $52
0000035:  curtrk                equ                     $53
0000036:  trkcnt                equ                     $54
0000037:  *
0000038:  q6l                   equ                     $c08c
0000039:  motoron               equ                     $c089
0000040:  motoroff              equ                     $c088
0000041:  phaseoff              equ                     $c080
0000042:  nbuf1                 equ                     $300
```

```
0000043:  dnib                equ                 $2d6
0000044:                      skp                 1
0000045:  * directory dependent stuff...
0000046:  clrscrn             equ                 $fc58
0000047:  scrn                equ                 $5ae
0000048:  dostyp              equ                 $ff
0000049:  sosid               equ                 $c00
0000050:  entlen              equ                 sosid+$23
0000051:  kernel              equ                 $2000
0000052:                      page
0000053:  xboot               dfb                 $01           ;(prodos boot id)
0000054:  entry               sec                               ;(apple iii enters xboot 'ora $38')
0000055:                      bcs                 entry1        ;branch if not apple iii native mode.
0000056:                      jmp                 goapl3        ;go do apple iii boot!
0000057:                      skp                 1
0000058:  entry1              stx                 unit          ;save unit number.
0000059:                      cmp                 #$03          ;for disk ii.
0000060:                      php                               ;save result, it may be irrelevent.
0000061:                      txa                               ;find out if disk ii.
0000062:                      and                 #$70          ;strip drive # if any.
0000063:                      lsr                 a
0000064:                      lsr                 a             ;get slot address.
0000065:                      lsr                 a
0000066:                      lsr                 a
0000067:                      ora                 #$c0
0000068:                      sta                 dent+1
0000069:                      ldy                 #$ff          ;look at last byte.
0000070:                      sty                 dent
0000071:                      plp                               ;restore carry (if disk ii & sect 0&2 read
                                                                 carry set).
0000072:                      iny                               ;make y=0
0000073:                      lda                 (dent),y      ;get device entry addr.
0000074:                      bne                 ndsk2         ;branch if not disk ii (16 sector).
0000075:                      bcs                 isdsk2        ;branch if it is disk ii, but block 0 read.
0000076:                      lda                 #3            ;make rom read only sector 2
0000077:                      sta                 xboot         ;to complete block 0
0000078:                      inc                 sector        ;(was = 1)
0000079:                      lda                 dent+1        ;do rts to re-enter rom.
0000080:                      pha
0000081:                      lda                 #$5b
0000082:                      pha
0000083:                      rts                               ;go read sector 2 into $900.
```

```
0000084:                        skp             1
0000085:  isdsk2                sta             trktmp          ;make sure previous track =0
0000086:                        sta             dent            ;and dent points at beginning of slot
0000087:                        ldy             #$63            ;move code from card to ram
0000088:  mvboot                lda             (dent),y
0000089:                        sta             zzstart-$5e,y
0000090:                        iny
0000091:                        cpy             #$eb            ;have we moved enough?
0000092:                        bne             mvboot
0000093:                        ldx             #6              ;now modify code to handle errors.
0000094:  modboot               ldy             mods,x
0000095:                        lda             chgs,x
0000096:                        sta             zzstart,y
0000097:                        lda             endcode,x
0000098:                        sta             zzzend,x
0000099:                        dex
0000100:                        bpl             modboot
0000101:                        lda             #<d2io          ;reset device entry
0000102:                        sta             dent+1          ; to point at disk ii routines.
0000103:                        lda             #>d2io          ;get low addr (must be <$80)
0000104:  ndsk2                 ldy             #0              ;make sure y=0 again.
0000105:                        cmp             #$f9
0000106:                        bcs             bterr1          ;branch if not bootable device.
0000107:                        sta             dent            ;save low adr of device call entry.
0000108:                        sty             iobuff
0000109:                        sty             idxl
0000110:                        sty             idxh            ;(y=0)
0000111:                        sty             idxp
0000112:                        sty             blok+1
0000113:                        iny
0000114:                        sty             dcmd            ;set read command.
0000115:                        iny
0000116:                        sty             blok            ; to read directory blocks
0000117:                        lda             #$c             ; 2-5 at $c00
0000118:                        sta             iobuff+1
0000119:                        sta             idxl+1
0000120:                        page
0000121:  rddir                 jsr             goread          ;call read block routine.
0000122:                        bcs             bterr2          ;give up on error.
0000123:                        inc             iobuff+1
0000124:                        inc             iobuff+1
0000125:                        inc             blok
```

```
0000126:                        lda             blok            ;have all directory blocks been read?
0000127:                        cmp             #6
0000128:                        bcc             rddir           ;loop if not.
0000129:                        skp             1
0000130:                        lda             sosid           ;is it a prodos (sos) directory?
0000131:                        ora             sosid+1
0000132:  bterr1                bne             booterr         ;branch if not.
0000133:                        lda             #4              ;begin look-up with first entry past header.
0000134:                        bne             nxdent1         ;branch always
0000135:  nxdent                lda             idxl
0000136:  nxdent1               clc
0000137:                        adc             entlen          ;bump to next directory entry.
0000138:                        tay                             ;save in y for now.
0000139:                        bcc             nxdent2         ;branch if not a page cross.
0000140:                        inc             idxl+1
0000141:                        lda             idxl+1          ;check for new block.
0000142:                        lsr             a               ;if even then new block.
0000143:                        bcs             nxdent2
0000144:                        cmp             #$a             ;have all file names been compared?
0000145:                        beq             nopro           ;branch if no pro.kernel.
0000146:                        ldy             #4              ;else, begin at block beginning.
0000147:  nxdent2               sty             idxl            ;note: this method treats garbage at
0000148:                        lda             sysname         ; the end of the dir block as an entry.
0000149:                        and             #$f             ;get target name length.
0000150:                        tay
0000151:  lookpro               lda             (idxl),y        ;look for matching name.
0000152:                        cmp             sysname,y       ;last to first method.
0000153:                        bne             nxdent          ;branch if no match.
0000154:                        dey                             ;else check all characters.
0000155:                        bpl             lookpro         ;including length/storage type.
0000156:                        and             #$f0            ;make sure storage type is a tree!
0000157:                        cmp             #$20
0000158:                        bne             booterr         ;branch if not.
0000159:                        ldy             #$10            ;get file type & index block addr.
0000160:                        lda             (idxl),y
0000161:                        cmp             #dostyp         ;is it a system file?
0000162:                        bne             booterr
0000163:                        iny
0000164:                        lda             (idxl),y
0000165:                        sta             blok
0000166:                        iny
0000167:                        lda             (idxl),y
```

```
0000168:                        sta             blok+1
0000169:                        lda             #0              ;now set up to read kernel.
0000170:                        sta             idxl
0000171:                        ldy             #$1e            ;read index block at $1e00 and
0000172:                        sty             idxl+1          ; kernel at $2000
0000173:                        sty             iobuff+1
0000174:                        iny
0000175:                        sty             idxh+1
0000176:  rdkernl               jsr             goread          ;read index block.
0000177:  bterr2                bcs             booterr
0000178:                        inc             iobuff+1
0000179:                        inc             iobuff+1
0000180:                        ldy             idxp            ;get index pointer
0000181:                        inc             idxp            ;bump for next time.
0000182:                        lda             (idxl),y
0000183:                        sta             blok
0000184:                        lda             (idxh),y        ;high disk addr.
0000185:                        sta             blok+1
0000186:                        ora             (idxl),y        ;if both=0 then done.
0000187:                        bne             rdkernl         ;branch if more to read.
0000188:                        skp             1
0000189:                        jmp             kernel          ;go execute kernel code.
0000190:                        skp             1
0000191:  nopro                 equ             *
0000192:  booterr               equ             *
0000193:                        jmp             quitmes
0000194:                        skp             1
0000195:                        msb             off
0000196:  sysname               dfb             $26
0000197:                        asc             "prodos          "
0000198:                        skp             1
0000199:  goread                lda             iobuff
0000200:                        sta             buff
0000201:                        lda             iobuff+1
0000202:                        sta             buff+1
0000203:                        jmp             (dent)
0000204:  *
0000205:  mods                  dfb             mod1,mod2,mod3,mod4
0000206:                        dfb             mod5,mod6,mod7
0000207:  *
0000208:  chgs                  dfb             chg1,chg2,chg3,chg4
0000209:                        dfb             chg5,chg6,chg7
```

```
0000210:  *
0000211:  endcode             equ               *
0000212:                      ldx               slotz
0000213:                      clc
0000214:                      rts
0000215:                      jmp               seek
0000216:  *
0000217:  goapl3              equ               *+$9800
0000218:                      lda               #$9f          ;make apple iii boot using block 1.
0000219:                      pha                             ;(the return address is $a000)
0000220:                      lda               #$ff
0000221:                      pha
0000222:                      lda               #1            ;read block 1.
0000223:                      ldx               #0
0000224:                      jmp               $f479
0000225:                      skp               2
0000226:  quitmes             jsr               clrscrn       ;clear video.
0000227:                      ldy               #meslen       ;print message centered on screen.
0000228:  prmess              lda               errmess,y
0000229:                      sta               scrn,y
0000230:                      dey
0000231:                      bpl               prmess
0000232:  hang                jmp               hang
0000233:  *
0000234:                      msb               on
0000235:  meslen              equ               28
0000236:  errmess             asc               "*** unable to load prodos ***"
0000237:  *
0000238:                      page
0000239:  setphase            lda               curtrk        ;get current track
0000240:  clrphase            and               #3            ;mask for 1 of 4 phases
0000241:                      rol               a             ;double for phaseon/off index
0000242:                      ora               slotz
0000243:                      tax
0000244:                      lda               phaseoff,x    ;turn on/off one phase
0000245:                      lda               #$2c
0000246:  ************************
0000247:  *                      *
0000248:  *   mswait subroutine  *
0000249:  *                      *
0000250:  ************************
0000251:  mswait              ldx               #$11
```

```
0000252:  msw1                dex                             ;delay 86 usec.
0000253:                      bne                 msw1
0000254:                      sbc                 #$1         ;done 'n' intervals?
0000255:                      bne                 mswait      ;(a-reg counts)
0000256:                      ldx                 slotz       ;restore x-reg
0000257:                      rts
0000258:  *
0000259:  *
0000260:  d2io                lda                 blok        ;figure out track & sector.
0000261:                      and                 #7          ;strip track for now.
0000262:                      cmp                 #4
0000263:                      and                 #3
0000264:                      php
0000265:                      asl                 a
0000266:                      plp
0000267:                      rol                 a           ;now we have the first sector of block.
0000268:                      sta                 sector
0000269:                      lda                 blok+1      ;get high block #
0000270:                      lsr                 a           ;shift hi addr to carry.
0000271:                      lda                 blok        ;now figure track #
0000272:                      ror                 a
0000273:                      lsr                 a
0000274:                      lsr                 a
0000275:                      sta                 track
0000276:                      asl                 a
0000277:                      sta                 trkn
0000278:                      lda                 buff+1
0000279:                      sta                 dbuf+1
0000280:                      ldx                 slotz
0000281:                      lda                 motoron,x
0000282:                      jsr                 rdsector    ;go read sector.
0000283:                      inc                 dbuf+1      ;bump address
0000284:                      inc                 sector
0000285:                      inc                 sector      ;and sector #
0000286:                      bcs                 quitrd      ;branch if error.
0000287:                      jsr                 rdsector
0000288:  quitrd              ldy                 motoroff,x
0000289:  erretrn             rts                             ;return error status in carry.
0000290:  *
0000291:  rdsector            equ                 *           ;do seek then read sector.
0000292:  *
0000293:  seek                lda                 trktmp      ;get track we're on.
```

```
0000294:                        asl                     a
0000295:                        sta                     curtrk
0000296:                        lda                     #$0
0000297:                        sta                     trkcnt          ;halftrack count.
0000298:  seek2                 lda                     curtrk          ;save curtrk for
0000299:                        sta                     prior           ;delayed turnoff.
0000300:                        sec
0000301:                        sbc                     trkn            ;delta-tracks.
0000302:                        beq                     seekend         ;br if curtrk=destination
0000303:                        bcs                     out             ;(move out, not in)
0000304:                        inc                     curtrk          ;incr current track (in).
0000305:                        bcc                     skin            ;(always taken)
0000306:  out                   dec                     curtrk          ;decr current track (out).
0000307:  skin                  sec
0000308:  step2                 jsr                     setphase
0000309:                        lda                     prior
0000310:                        clc                                     ;for phaseoff
0000311:                        jsr                     clrphase        ;de-energize previous phase.
0000312:                        bne                     seek2           ;(always taken)
0000313:  seekend               equ                     *
0000314:  *
0000315:  rdsect1               ldy                     #$7f            ;allow 127 mistakes.
0000316:                        sty                     rtrycnt
0000317:                        php
0000318:  *
0000319:  tryread               plp                                     ;fix stack.
0000320:                        skp                     1
0000321:  rdhead                sec                                     ;anticipate error.
0000322:                        dec                     rtrycnt         ;if = 0 then give up!
0000323:                        beq                     erretrn         ;branch if can't fine/read sector.
0000324:                        clc                                     ;indicate reading header.
0000325:  rddata                php                                     ;carry set if reading sector.
0000326:  rd0                   dey                                     ;every time y=0 decrement find count.
0000327:                        beq                     tryread
0000328:                        skp                     1
0000329:  zzstart               equ                     *
0000330:  * from zzstart to zzend code is moved from
0000331:  * rom and modified to match this code...
0000332:                        skp                     1
0000333:  rd1                   lda                     q6l,x           ;read a byet from the state machine.
0000334:                        bpl                     rd1             ;loop until ready.
0000335:                        dsect
```

```
0000336:                        org             zzstart+5       ;equivalent to org *
0000337:  rd1a                  eor             #$d5            ;mark 1?
0000338:  mod1                  equ             *-zzstart+1
0000339:                        bne             rd0             ;branch if not.
0000340:  chg1                  equ             rd0-*
0000341:  rd2                   lda             q6l,x
0000342:                        bpl             rd2
0000343:                        cmp             #$aa            ;mark 2?
0000344:                        bne             rd1a
0000345:                        nop                             ;waste a little time.
0000346:  rd3                   lda             q6l,x
0000347:                        bpl             rd3
0000348:                        cmp             #$96            ;header mark 3?
0000349:                        beq             rdhd1           ;branch if it is.
0000350:                        plp                             ;were we looking for data mark 3?
0000351:  mod2                  equ             *-zzstart+1
0000352:                        bcc             rdhead          ;branch if not.
0000353:  chg2                  equ             rdhead-*
0000354:                        eor             #$ad            ;data mark 3?
0000355:                        beq             rddt1           ;go read data field if true...
0000356:  mod3                  equ             *-zzstart+1
0000357:  rdhd0                 bne             rdhead          ;otherwise, start over.
0000358:  chg3                  equ             rdhead-*
0000359:  rdhd1                 ldy             #3              ;read in trk,sect,&volume #.
0000360:  rdhd2                 sta             trktmp          ;save last result in temp
0000361:  rdhd3                 lda             q6l,x
0000362:                        bpl             rdhd3
0000363:                        rol             a
0000364:                        sta             oddbits         ;save odd bits (7,5,3,1)
0000365:  rdhd4                 lda             q6l,x
0000366:                        bpl             rdhd4
0000367:                        and             oddbits         ;combine even and odd to form value.
0000368:                        dey
0000369:                        bne             rdhd2           ;read in next pair.
0000370:                        plp
0000371:                        cmp             sector          ;last byte formed is sector#
0000372:  mod4                  equ             *-zzstart+1
0000373:                        bne             rdhead          ;branch if target sector not found.
0000374:  chg4                  equ             rdhead-*
0000375:                        skp             1
0000376:                        lda             trktmp          ;previous result is track #
0000377:                        cmp             track           ;is desired track found?
```

```
0000378:  mod5                equ                 *-zzstart+1
0000379:                      bne                 goseek          ;re-seek if mismatch.
0000380:  chg5a               equ                 *
0000381:  mod6                equ                 *-zzstart+1
0000382:                      bcs                 rddata          ;branch if proper track always.
0000383:  chg6                equ                 rddata-*
0000384:                      skp                 1
0000385:  rddt1               ldy                 #$56            ;read 2 bit groupings first.
0000386:  rddt1a              sty                 oddbits
0000387:  rddt2               ldy                 q6l,x
0000388:                      bpl                 rddt2
0000389:                      eor                 dnib,y          ;denibblize using table left from boot rom.
0000390:                      ldy                 oddbits         ;save in nbuf1
0000391:                      dey
0000392:                      sta                 nbuf1,y
0000393:                      bne                 rddt1a          ;loop until all 86 groups are read.
0000394:                      skp                 1
0000395:  rddt3               sty                 oddbits         ;now count up for 6-bit groups.
0000396:  rddt4               ldy                 q6l,x
0000397:                      bpl                 rddt4
0000398:                      eor                 dnib,y
0000399:                      ldy                 oddbits         ;save result to specified buffer.
0000400:                      sta                 (dbuf),y
0000401:                      iny
0000402:                      bne                 rddt3           ;loop for 256 bytes.
0000403:  rdchk               ldy                 q6l,x           ;now verify checksum...
0000404:                      bpl                 rdchk
0000405:                      eor                 dnib,y          ;must be equal...
0000406:  mod7                equ                 *-zzstart+1
0000407:                      bne                 rdhd0           ;branch if error.
0000408:  chg7                equ                 rdhd0-*
0000409:                      ldy                 #0              ;make y=0
0000410:  nxttwo              ldx                 #$56            ;now combine 2-bit group with 6 bit group
0000411:  twobit              dex                                 ;all done with this group?
0000412:                      bmi                 nxttwo          ;branch if so.
0000413:                      lda                 (dbuf),y
0000414:                      lsr                 nbuf1,x
0000415:                      rol                 a
0000416:                      lsr                 nbuf1,x
0000417:                      rol                 a
0000418:                      sta                 (dbuf),y
0000419:                      iny
```

```
0000420:                        bne             twobit
0000421:                        skp             1
0000422:  zzzend                equ             *
0000423:                        ldx             slotz
0000424:                        clc                             ;indicate good read.
0000425:                        rts
0000426:  chg5                  equ             *-chg5a
0000427:  goseek                jmp             seek
0000428:                        dend
0000429:                        ds              $a00-zzstart-5,0
```

```
0000430:  * the following is the apple /// sos boot loader.
0000431:                        msb                 off
0000432:                        sbtl                "sos system boot 1.1"
0000433:  ****************************************************************
0000434:  *
0000435:  * sos system boot
0000436:  *
0000437:  * the code resides on blocks 0 and 1 of every sos diskette.
0000438:  * its job is to locate the file named 'sos.kernel' on the
0000439:  * boot diskette (drive 1), load the entire file into memory
0000440:  * and then transfer control to the second stage boot,
0000441:  * (sos loader).
0000442:  *
0000443:  * this first stage boot is designed to have minimal knowledge
0000444:  * of both the rom code and the operating system including
0000445:  * its associated drivers.
0000446:  *
0000447:  * assumptions:
0000448:  *
0000449:  *   1.  screen is cleared and 40 column b&w mode is selected.
0000450:  *
0000451:  *   2.  blockio routine is in rom with the entry pt at $f479.
0000452:  *
0000453:  *   3.  hardware:  see equates
0000454:  *
0000455:  *   4.  sos directory format
0000456:  *
0000457:  *   5.  file 'sos.kernel' format
0000458:  *
0000459:  * potential problems:
0000460:  *
0000461:  *   1.  this code disregards the address/count information
0000462:  *       affixed to the front of the sos loader module.
0000463:  *
0000464:  *   2.  if code grows beyond current size, the padding at end of the code
0000465:  *       needs to be modified.  (the code currently resides in less than a
0000466:  *       single block; thus two padding statements necessary to have total
0000467:  *       be two blocks on diskette.)
0000468:  *
0000469:  *
0000470:  ****************************************************************
0000471:                        page
```

```
0000472:  *
0000473:  * hardware addresses
0000474:  *
0000475:  e.reg               equ                 $ffdf
0000476:  b.reg               equ                 $ffef
0000477:  kybdstrb            equ                 $c010
0000478:  *
0000479:  * monitor data and addresses
0000480:  *
0000481:  ibcmd               equ                 $87
0000482:  ibbufp              equ                 $85
0000483:  blockio             equ                 $f479
0000484:  *
0000485:  * zero page storage (z reg = $03)
0000486:  *
0000487:  zpage               equ                 $e0
0000488:  blknum              equ                 zpage+0       ; & 1
0000489:  *
0000490:  begin               equ                 zpage+2       ; & 3
0000491:  end                 equ                 zpage+4       ; & 5
0000492:  blk.ctr             equ                 zpage+6
0000493:  temp                equ                 zpage+7
0000494:  *
0000495:  sosldr              equ                 zpage+8       ; & 9
0000496:  *
0000497:  * equates
0000498:  *
0000499:  dirblk0             equ                 $a400
0000500:  entry0              equ                 dirblk0+4     ; loc of first file entry in directory
0000501:  entry.len           equ                 entry0+$1f    ; loc of entry length in directory
0000502:  storage             equ                 0             ; file's storage type
0000503:  sapling             equ                 $20           ; storage type = tree index file w/one index
                                                                  block
0000504:  rootdir             equ                 $f0           ; storage type = root directory
0000505:  nextdblk            equ                 2             ; loc of next directory block
0000506:  *
0000507:  k.xblk              equ                 $c00          ;start loc of sos.kernel's index block
0000508:  xblk                equ                 $11           ; loc of index block address in file entry
0000509:  k.file              equ                 $1e00         ; start loc of sos.kernel file
0000510:  k.label             equ                 k.file+0      ; loc of label in file "sos.kernel"
0000511:  k.hdr.cnt           equ                 k.file+8      ;   "    header       "
0000512:  ****************************************************************
```

```
0000513:  *
0000514:  * sos system boot - entry point
0000515:  *
0000516:  ********************************************************************
0000517:  *
0000518:  bootinfo             equ                 *
0000519:  asmbase              equ                 * ;assembly base address
0000520:  runbase              equ                 $a000          ;execution base address
0000521:                       jmp                 boot+runbase-asmbase
0000522:                       asc                 "sos boot  1.1 "    ; sos boot identification "stamp"
0000523:                       page
0000524:  ********************************************************************
0000525:  *
0000526:  * local data storage
0000527:  *
0000528:  ********************************************************************
0000529:  *
0000530:  namlen               dfb                 10
0000531:  name                 asc                 "sos.kernel     "
0000532:  name2                asc                 "sos krnl"
0000533:  name2.len            equ                 *-name2
0000534:  *
0000535:  * messages
0000536:  *
0000537:  msg                  equ                 * ; message table
0000538:  *
0000539:  msg0                 asc                 "i/o error"
0000540:  msg0l                equ                  *-msg0
0000541:  xmsg0                dw                   *-msg-1
0000542:  *
0000543:  msg1                 asc                 "file 'sos.kernel' not found"
0000544:  msg1l                equ                 *-msg1
0000545:  xmsg1                dw                  *-msg-1
0000546:  *
0000547:  msg2                 asc                 "invalid kernel file"
0000548:  msg2l                equ                 *-msg2
0000549:  xmsg2                dw                  *-msg-1
0000550:  *
0000551:  * -- these dw's get around tla's hatred of hibyte/lobyte stuff
0000552:  *
0000553:  xk.xblk              dw                  k.xblk
0000554:  xk.file              dw                  k.file
```

```
0000555:  xk.hdr.cnt          dw                  k.hdr.cnt+6     ;includes 6-byte offset
0000556:  xentry0             dw                  entry0
0000557:                      page
0000558:  ***************************************************************
0000559:  *
0000560:  * sos system boot - main code body
0000561:  *
0000562:  ***************************************************************
0000563:  *
0000564:  * turn off interrupts & decimal mode
0000565:  *
0000566:  boot                sei
0000567:                      cld
0000568:  *
0000569:  * set up environment register and init stack
0000570:  *
0000571:                      lda                 #$77            ;1mhz dsbl
0000572:  *                                 i/o enbl
0000573:  *                                 primary stack enbl
0000574:  *                                 reset/nmi enbl
0000575:  *                                 write prot. dsbl
0000576:  *                                 primary stack enbl
0000577:  *                                 rom1 enbl
0000578:  *                                 rom enbl
0000579:                      sta                 e.reg
0000580:                      ldx                 #$fb
0000581:                      txs
0000582:                      bit                 kybdstrb        ; turns off kybd
0000583:                      lda                 #$40            ; "rti" instruction
0000584:                      sta                 $ffca           ; prevents reboot w/keyboard nmi
0000585:  *
0000586:  * find highest memory bank in system and set bank reg to it
0000587:  * - max memsize = 256k.
0000588:  *
0000589:                      lda                 #7
0000590:                      sta                 b.reg
0000591:                      ldx                 #0
0000592:  boot005             dec                 b.reg
0000593:                      stx                 $2000
0000594:                      lda                 $2000
0000595:                      bne                 boot005
0000596:  *
```

```
0000597:  * read in blocks 1 thru n (rest of boot and all of root dir.)
0000598:  *
0000599:                        lda                 #1
0000600:                        sta                 blknum
0000601:                        lda                 #0
0000602:                        sta                 blknum+1
0000603:  *
0000604:                        lda                 #0
0000605:                        sta                 ibbufp
0000606:                        lda                 #$a2
0000607:                        sta                 ibbufp+1
0000608:  *
0000609:                        jsr                 read.blk+runbase-asmbase ; rest of boot (block 1)
0000610:  *
0000611:                        inc                 blknum         ; first root directory block (block 2)
0000612:                        lda                 #0
0000613:                        sta                 blk.ctr
0000614:  rd.dir                inc                 ibbufp+1
0000615:                        inc                 ibbufp+1
0000616:                        inc                 blk.ctr
0000617:  *
0000618:                        jsr                 read.blk+runbase-asmbase; root directory
0000619:  *
0000620:                        ldy                 #nextdblk      ; if nextdir field = 0 then done
0000621:                        lda                 (ibbufp),y
0000622:                        sta                 blknum
0000623:                        iny
0000624:                        lda                 (ibbufp),y
0000625:                        sta                 blknum+1
0000626:                        bne                 rd.dir
0000627:                        lda                 blknum
0000628:                        bne                 rd.dir
0000629:                        page
0000630:  *
0000631:  * search directory for file 'sos.kernel'
0000632:  *
0000633:                        lda                 xentry0+runbase-asmbase ;get lo byte of address
0000634:                        sta                 begin
0000635:                        lda                 xentry0+1+runbase-asmbase
0000636:                        sta                 begin+1
0000637:  *
0000638:  search               clc                                ; end:=begin+512-entry.len
```

```
0000639:                        lda             begin+1
0000640:                        adc             #2
0000641:                        sta             end+1
0000642:                        sec
0000643:                        lda             begin
0000644:                        sbc             entry.len
0000645:                        sta             end
0000646:                        lda             end+1
0000647:                        sbc             #0
0000648:                        sta             end+1
0000649:  *
0000650:  srch020               ldy             #0              ; does count match?
0000651:                        lda             (begin),y
0000652:                        and             #$f
0000653:                        cmp             namlen+runbase-asmbase
0000654:                        bne             srch040         ; no match
0000655:  *
0000656:                        tay
0000657:  srch030               lda             (begin),y       ; do chars match?
0000658:                        cmp             name-1+runbase-asmbase,y
0000659:                        bne             srch040         ; no match
0000660:                        dey
0000661:                        bne             srch030
0000662:  *
0000663:                        ldy             #storage        ;test storage type
0000664:                        lda             (begin),y       ;must be sapling
0000665:                        and             #$f0
0000666:                        cmp             #sapling
0000667:                        beq             match
0000668:                        cmp             #rootdir        ;skip if stg type=rootdir
0000669:                        beq             srch040
0000670:  *
0000671:                        ldx             xmsg2+runbase-asmbase  ;err,invalid kernel file
0000672:                        ldy             #msg2l
0000673:                        jmp             prnt.msg+runbase-asmbase
0000674:  *
0000675:  srch040               clc
0000676:                        lda             begin
0000677:                        adc             entry.len
0000678:                        sta             begin
0000679:                        lda             begin+1
0000680:                        adc             #0
```

```
0000681:                        sta                     begin+1
0000682:                        lda                     end
0000683:                        cmp                     begin           ;is begin <=end?
0000684:                        lda                     end+1
0000685:                        sbc                     begin+1
0000686:                        bcs                     srch020         ;yes,search next field in current block
0000687:  *
0000688:                        clc                                     ;begin :=end+entry.len
0000689:                        lda                     end
0000690:                        adc                     entry.len
0000691:                        sta                     begin
0000692:                        lda                     end+1
0000693:                        adc                     #0
0000694:                        sta                     begin+1
0000695:  *
0000696:                        dec                     blk.ctr
0000697:                        bne                     search          ;search the next dir block
0000698:  *
0000699:                        ldx                     xmsg1+runbase-asmbase   ;err, can't find 'sos.kernel'
0000700:                        ldy                     #msg1l
0000701:                        jmp                     prnt.msg+runbase-asmbase
0000702:                        page
0000703:  *
0000704:  * file entry 'sos.kernel' found
0000705:  * read in its index block ($c00) and first data block ($1e00)
0000706:  *
0000707:  match                 ldy                     #xblk
0000708:                        lda                     (begin),y
0000709:                        sta                     blknum
0000710:                        iny
0000711:                        lda                     (begin),y
0000712:                        sta                     blknum+1
0000713:                        lda                     xk.xblk+runbase-asmbase   ;get lo byte of address
0000714:                        sta                     ibbufp
0000715:                        lda                     xk.xblk+1+runbase-asmbase ;get hi byte of address
0000716:                        sta                     ibbufp+1
0000717:                        jsr                     read.blk+runbase-asmbase; index block
0000718:  *
0000719:                        lda                     xk.file+runbase-asmbase   ;get lo byte of address
0000720:                        sta                     ibbufp
0000721:                        lda                     xk.file+1+runbase-asmbase
0000722:                        sta                     ibbufp+1
```

```
0000723:                        lda                 k.xblk
0000724:                        sta                 blknum
0000725:                        lda                 k.xblk+$100
0000726:                        sta                 blknum+1
0000727:                        jsr                 read.blk+runbase-asmbase; first data block
0000728:  *
0000729:  * check the label, should be 'sos krnl'
0000730:  *
0000731:                        ldx                 #name2.len-1
0000732:  chk010                lda                 k.label,x
0000733:                        cmp                 name2+runbase-asmbase,x
0000734:                        beq                 chk020
0000735:                        ldx                 xmsg2+runbase-asmbase   ; err, invalid kernel file
0000736:                        ldy                 #msg2l
0000737:                        jmp                 prnt.msg+runbase-asmbase
0000738:  chk020                dex
0000739:                        bpl                 chk010
0000740:  *
0000741:  * read in the rest of the data blocks in file "sos.kernel"
0000742:  *
0000743:                        lda                 #0
0000744:                        sta                 temp
0000745:  *
0000746:  data010                inc                 temp
0000747:                        inc                 ibbufp+1
0000748:                        inc                 ibbufp+1
0000749:  *
0000750:                        ldx                 temp           ; get block address of next data block
0000751:                        lda                 k.xblk,x
0000752:                        sta                 blknum
0000753:                        lda                 k.xblk+$100,x
0000754:                        sta                 blknum+1
0000755:  *
0000756:                        lda                 blknum         ; is next block address = 0 ?
0000757:                        bne                 data020
0000758:                        lda                 blknum+1
0000759:                        beq                 entry.a3       ; yes, stop reading
0000760:  *
0000761:  data020               jsr                 read.blk+runbase-asmbase; read data block
0000762:                        jmp                 data010+runbase-asmbase ;  and repeat
0000763:  *
0000764:  * build sos loader entry point address
```

```
0000765:  *
0000766:  entry.a3            clc                              ; sosldr:=k.hdr.cnt+(k.hdr.cnt)
0000767:                      lda                 xk.hdr.cnt+runbase-asmbase
0000768:                      adc                 k.hdr.cnt
0000769:                      sta                 sosldr
0000770:                      lda                 xk.hdr.cnt+1+runbase-asmbase
0000771:                      adc                 k.hdr.cnt+1
0000772:                      sta                 sosldr+1
0000773:  *
0000774:  * now jump to sos loader (secondary bootstrap)
0000775:  *
0000776:                      jmp                 (sosldr)
0000777:  *
0000778:  ********************************************************************
0000779:  *
0000780:  * finished !!
0000781:  *
0000782:  * state of registers:
0000783:  *
0000784:  * b reg = highest 32k bank
0000785:  * e reg = $77
0000786:  * z reg = $03
0000787:  *
0000788:  * file "sos.kernel":
0000789:  *
0000790:  * index block is at $c00...$fff
0000791:  * data block 0 is at $2200..$23ff
0000792:  * data block 1 is at $2400..$25ff
0000793:  *  " "
0000794:  * data block n "
0000795:  *
0000796:  ********************************************************************
0000797:                        page
0000798:  ********************************************************************
0000799:  *
0000800:  * read block routine
0000801:  *
0000802:  * input: blknum & ibbufp
0000803:  *
0000804:  ********************************************************************
0000805:  *
0000806:  read.blk             equ                 *
```

```
0000807:                        lda                     #1
0000808:                        sta                     ibcmd
0000809:  *
0000810:                        lda                     blknum
0000811:                        ldx                     blknum+1
0000812:                        jsr                     blockio
0000813:                        bcs                     rd.err
0000814:                        rts                                     ; normal exit
0000815:  *
0000816:  rd.err                 ldx                     xmsg0+runbase-asmbase   ;err, i/o error
0000817:                        ldy                     #msg0l
0000818:                        jmp                     prnt.msg+runbase-asmbase
0000819:                        page
0000820:  ********************************************************************
0000821:  *
0000822:  * print message
0000823:  *
0000824:  * input: msg index  (x)
0000825:  *  msg length (y)
0000826:  ********************************************************************
0000827:  msgline               equ                     $5a8            ; prnt.msg routine
0000828:  *
0000829:  prnt.msg              equ                     *
0000830:                        sty                     temp            ; center msg (y:=40-len/2+len)
0000831:                        sec
0000832:                        lda                     #40
0000833:                        sbc                     temp
0000834:                        lsr                     a
0000835:                        clc
0000836:                        adc                     temp
0000837:                        tay
0000838:  *
0000839:  prnt010               lda                     msg+runbase-asmbase,x
0000840:                        sta                     msgline-1,y
0000841:                        dex
0000842:                        dey
0000843:                        dec                     temp
0000844:                        bne                     prnt010
0000845:  *
0000846:                        lda                     $c040             ; sound bell
0000847:                        jmp                     *+runbase-asmbase ; hang until reboot (ctrl/reset)
0000848:  *
```

```
0000849:  ****************************************************************
0000850:  *
0000851:  * padding to end of two blocks. modify if code length increases
0000852:  * beyond one block.
0000853:  *
0000854:  ****************************************************************
0000855:  *
0000856:  pad                 equ                 *-asmbase
0000857:                      ds                  512-pad,0      ;pad to end of block
0000858:  zzend                equ                 *
0000859:  *<esc>e

END OF LISTING
```