puZIP - ZIP File compression#

```
; DASM V2.12.04 source

; ZIP, 32k window, dynamic Huffman
; 174848  Defl:N   96995  45%  09-13-00  21:08  c4d2f886   t

; PUZIP, 32k window, fixed Huffman
; 174848  Defl:N  107741  39%  00-00-80  00:00  c4d2f886   T

; C64, 8k window, fixed Huffman
; 174848  Defl:N  113643  35%  00-00-80  00:00  c4d2f886   T

; C64, 4k window, fixed Huffman
; 174848  Defl:N  114699  34%  00-00-80  00:00  c4d2f886   T

; C64, 2k window, fixed Huffman
; 174848  Defl:N  119375  32%  00-00-80  00:00  c4d2f886   T


; PKZIP
; =====
; Length  Method   Size  Ratio   Date    Time   CRC-32     Name
; ------  ------   ----  -----   ----    ----   ------     ----
; 174848  Defl:N   60230  66%  12-02-99  18:03  d625f0bd   PARTS1.D64
; 174848  Defl:N   24441  86%  12-02-99  18:05  76e851cd   PARTS2.D64
; ------          ------  ---                              -------
; 349696           84671  76%                              2 files

; C128
; ====
; Length  Method   Size  Ratio   Date    Time   CRC-32     Name
; ------  ------   ----  -----   ----    ----   ------     ----
; 174848  Defl:N   75580  57%  00-00-80  00:00  d625f0bd   PARTS1.D64
; 174848  Defl:N   31867  82%  00-00-80  00:00  76e851cd   PARTS2.D64
; ------          ------  ---                              -------
; 349696          107447  69%                              2 files


; ZIP
; ===
; Length  Method   Size  Ratio   Date    Time   CRC-32     Name
; ------  ------   ----  -----   ----    ----   ------     ----
; 174848  Defl:N   59986  66%  02-09-89  00:45  d625f0bd   parts1.d64
; 174848  Defl:N   24418  86%  02-09-89  00:45  76e851cd   parts2.d64
; ------          ------  ---                              -------
; 349696           84404  76%                              2 files

; C64               39min (1581 files->1581)
; ===
; Length  Method   Size  Ratio   Date    Time   CRC-32     Name
; ------  ------   ----  -----   ----    ----   ------     ----
; 174848  Defl:N   71129  59%  06-07-01  02:39  d625f0bd   PARTS1.D64
; 174848  Defl:N   30614  83%  06-07-01  02:39  76e851cd   PARTS2.D64
; ------          ------  ---                              -------
; 349696          101743  71%                              2 files

; C64 with lazy matching    47min (1581 files->1581)
; ======================
```

```
; Length  Method   Size  Ratio   Date    Time   CRC-32     Name
; ------  ------   ----  -----   ----    ----   ------     ----
; 174848  Defl:N   70548  60%  06-07-01  02:39  d625f0bd   PARTS1.D64
; 174848  Defl:N   30491  83%  06-07-01  02:39  76e851cd   PARTS2.D64
; ------          ------  ---                             -------
; 349696          101039  71%                             2 files

;           Length   Size  Ratio   Date    Time   CRC-32     Name
; ZIP           174848   96995  45%  02-09-89  00:45  c4d2f886   t
; PuZip - 32k+lazy  174848  106660  39%  09-01-02  02:39  c4d2f886   t
; 0(8k+lazy)    19:50   174848  112701  36%  08-09-02  02:39  c4d2f886   T
; 1(4k+greedy)  15:12   174848  114635  34%  08-09-02  02:39  c4d2f886   T
; 2(2k+greedy)  13:51   174848  118802  32%  08-09-02  02:39  c4d2f886   T
; 3(1k+greedy)  12:57   174848  121699  30%  08-09-02  02:39  c4d2f886   T



    processor 6502

ACPTR  = $ffa5
CHKIN  = $ffc6
CHKOUT = $ffc9
CHRIN  = $ffcf
CHROUT = $ffd2
CIOUT  = $ffa8
CLOSE  = $ffc3
GETIN  = $ffe4
CLALL  = $ffe7
CLRCHN = $ffcc
LISTEN = $ffb1
OPEN   = $ffc0
SECOND = $ff93
SETLFS = $ffba
SETNAM = $ffbd
TALK   = $ffb4
TKSA   = $ff96
UNLSN  = $ffae
UNTLK  = $ffab
;READST = $ffb7

ST  = $90

pos = $0200 ; 4 bytes
arg = $0204 ; 2 bytes

#if MACH == 64 || MACH == 128
IDE64BlockWrite = $DEF1
IDE64BlockRead = $DEF4
IDE64BR = $6e ;$6e $6f
be  = $69 ;dc.b 0
D2PRA  = $DD00      ; c64 SERIAL PORT LOCATION
D1ICR  = $DC0D      ; 6526 cia INTERRUPT CONTROL REGISTER
D1SDR  = $DC0C      ; 6526 cia SERIAL DATA REGISTER
#endif

        ; name  -- input file name 16+2
CA_CRC   = 18   ; crc32 -- cyclic redundancy check
CA_CSize = 22   ; csize -- compressed size
CA_USize = 26   ; usize -- uncompressed size
```

```
CA_LHdr  = 30   ; lhdroff -- local header offset
CA_FLen  = 34   ; filename length
CA_TS    = 35   ; starting track & sector, if any
CA_Mode  = 37   ; compress mode -- "S" or "Y"
CASIZE   = 38


; ============================== VIC20 ===================================
#if MACH == 20
; 16k expansion required    - $1200-$5fff

MAX_MATCH  = 255    ;$0ff
BLOCK_SIZE = 4*256  ;$400   ; must be rounded to 256 bytes and power of 2
; Maximum for BLOCK_SIZE is 32768

LSHIFT = 4      ; 4 and 5 supported, (3 too slow, 6 memory-hungry)
LSIZE  = (1<<LSHIFT)

lPairSize = 4*LSIZE*LSIZE
bSkipSize = 2*BLOCK_SIZE
bufferSize = BLOCK_SIZE + MAX_MATCH+1

; lPair must be before bSkip in memory, both must be aligned to page
lPair  = $4c00  ; $4c00..$4fff  unsigned long  lPair[16*16]
bSkip  = lPair+lPairSize
        ; $5000..$57ff  unsigned short bSkip[BLOCK_SIZE]
buffer = $5800  ; $5800..$5cff  unsigned char  buffer[BLOCK_SIZE + MAX_MATCH]

;outbuf = $2b00
outend = $4a00  ; $2b00..$4a00  ; 181 files leaves 1k output buffer
MAXFILES = 181


byte   = $2
CRC    = $6a    ; $6a..6d
bTmp   = $14    ; $14,$15

rPtr   = $f7    ; $f7,$f8
wPtr   = $f9    ; $f9,$fa

zp0    = $fb
zp1    = $fc

tmp    = $fd
CAPtr  = $fe    ; $fe,$ff

crc32_tab0 = $5d00
crc32_tab1 = $5e00
crc32_tab2 = $5f00
crc32_tab3 = $4b00
inBuf = $4a00


    ORG $1201
    DC.B $0b,$12,20,0   ; '20 SYS4621'
    DC.B $9e,$34,$36,$32
    DC.B $31,0,0,0
#endif
```

```
; ============================== C16 ==================================
#if MACH == 16
; C16 $1000-$3fff   +/4 $1000-$fd00

MAX_MATCH  = 255    ;$0ff
BLOCK_SIZE = 4*256  ;$400   ; must be rounded to 256 bytes and power of 2
; Maximum for BLOCK_SIZE is 32768

LSHIFT = 4      ; 4 and 5 supported, (3 too slow, 6 memory-hungry)
LSIZE  = (1<<LSHIFT)

lPairSize = 4*LSIZE*LSIZE   ; $400
bSkipSize = 2*BLOCK_SIZE    ; $800
bufferSize = BLOCK_SIZE + MAX_MATCH+1   ; $500

; lPair must be before bSkip in memory, both must be aligned to page
lPair  = $2a00  ; $2a00..$2dff  unsigned long  lPair[16*16]
bSkip  = lPair+lPairSize
        ; $2e00..$35ff  unsigned short bSkip[BLOCK_SIZE]
buffer = $3600  ; $3600..$3aff  unsigned char  buffer[BLOCK_SIZE + MAX_MATCH]

outend = lPair
MAXFILES = 26   ; 26 leaves $100 for output buffer


;   $00D8-00E8  216-232 Area for use by application software


byte   = $dd
CRC    = $e0    ; $e0..$e3
bTmp   = $de    ; $de,$df

rPtr   = $e4    ; $e4,$e5
wPtr   = $e6    ; $e6,$e7

zp0    = $db
zp1    = $dc

tmp    = $da
CAPtr  = $d8    ; $d8,$d9

crc32_tab0 = $3b00
crc32_tab1 = $3c00
crc32_tab2 = $3d00
crc32_tab3 = $3e00
inBuf = $3f00


    ORG $1001
    DC.B $0b,$10,20,0   ; '20 SYS4109'
    DC.B $9e,$34,$31,$30
    DC.B $39,0,0,0
#endif


; ============================== Plus/4 ===============================
#if MACH == 4
; C16 $1000-$3fff   +/4 $1000-$fd00
```

```
MAX_MATCH  = 255    ;$0ff
BLOCK_SIZE = 32*256 ;$2000  ; must be rounded to 256 bytes and power of 2
; Maximum for BLOCK_SIZE is 32768

LSHIFT = 5      ; 4 and 5 supported, (3 too slow, 6 memory-hungry)
LSIZE  = (1<<LSHIFT)

lPairSize = 4*LSIZE*LSIZE   ; $1000
bSkipSize = 2*BLOCK_SIZE    ; $4000
bufferSize = BLOCK_SIZE + MAX_MATCH+1   ; $2100

; lPair must be before bSkip in memory, both must be aligned to page
lPair  = $8000  ; $8000..$8fff  unsigned long  lPair[32*32]
bSkip  = lPair+lPairSize
        ; $9000..$cfff  unsigned short bSkip[BLOCK_SIZE]
buffer = $d400  ; $d400..$f4ff  unsigned char  buffer[BLOCK_SIZE + MAX_MATCH]

outend = $6000
MAXFILES = 254  ; 254 leaves $344c for output buffer

crc32_tab0 = $d300
crc32_tab1 = $d200
crc32_tab2 = $d100
crc32_tab3 = $d000
inBuf = $7f00

;   $00D8-00E8  216-232 Area for use by application software


byte   = $dd
CRC    = $e0    ; $e0..$e3
bTmp   = $de    ; $de,$df

rPtr   = $e4    ; $e4,$e5
wPtr   = $e6    ; $e6,$e7

zp0    = $db
zp1    = $dc

tmp    = $da
CAPtr  = $d8    ; $d8,$d9



    ORG $1001
    DC.B $0b,$10,20,0   ; '20 SYS4109'
    DC.B $9e,$34,$31,$30
    DC.B $39,0,0,0
#endif


#if MACH == 64
; ================================ C64 ===================================
; $14..15, 2, $9e..9f
; $f7..ff


MAX_MATCH  = 255    ;$0ff
```

```
BLOCK_SIZE = 32*256 ;$2000  ; must be rounded to 256 bytes and power of 2

LSHIFT = 5      ; 4 and 5 supported, (3 too slow, 6 memory-hungry)
LSIZE  = (1<<LSHIFT)

lPairSize = 4*LSIZE*LSIZE   ; $1000
bSkipSize = 2*BLOCK_SIZE    ; $4000
bufferSize = BLOCK_SIZE + MAX_MATCH+1   ; $2100

; lPair must be before bSkip in memory, both must be aligned to page
lPair  = $7800  ; $7800..$87ff  unsigned long  lPair[32*32]
bSkip  = lPair+lPairSize
        ; $8800..$c7ff  unsigned short bSkip[BLOCK_SIZE]
buffer = $d000  ; $d000..$f0ff  unsigned char  buffer[BLOCK_SIZE + MAX_MATCH]

outend = $7800
MAXFILES = 254  ; 255 means disk image

crc32_tab0 = $c900
crc32_tab1 = $ca00
crc32_tab2 = $cb00
crc32_tab3 = $cc00
inBuf = $cd00

; CE00 and CF00 reserved for REU/RamDOS


byte   = $2
CRC    = $6a    ; $6a..6d
bTmp   = $14    ; $14,$15

rPtr   = $f7    ; $f7,$f8
wPtr   = $f9    ; $f9,$fa

zp0    = $fb
zp1    = $fc

tmp    = $fd
CAPtr  = $fe    ; $fe,$ff


FASTSER = $02a3

    ORG $0801
    DC.B $0b,8,64,0     ; '64 SYS2061'
    DC.B $9e,$32,$30,$36
    DC.B $31,0,0,0

    lda #$36
    sta 1
#endif

; ============================== C128 ====================================
#if MACH == 128

MAX_MATCH  = 255    ;$0ff
BLOCK_SIZE = 32*256 ;$2000  ; must be rounded to 256 bytes and power of 2

LSHIFT = 5      ; 4 and 5 supported, (3 too slow, 6 memory-hungry)
```

```
LSIZE  = (1<<LSHIFT)

lPairSize = 4*LSIZE*LSIZE   ;$1000
bSkipSize = 2*BLOCK_SIZE    ;$4000
bufferSize = BLOCK_SIZE + MAX_MATCH+1   ; $2100

buffer = $9f00  ; $9f00..$bfff  unsigned char  buffer[BLOCK_SIZE + MAX_MATCH]

; lPair must be before bSkip in memory, both must be aligned to page
; lPair and bSkip are now in bank1
lPair  = $1000  ; $1000..$1fff  unsigned long  lPair[32*32]
bSkip  = lPair+lPairSize
        ; $2000..$5fff  unsigned short bSkip[BLOCK_SIZE]

outend = $9f00  ; $3d00..$9eff  ; 254 files leaves 15k outpuf buffer
MAXFILES = 254  ; 255 means disk image


byte   = $86
tmp    = $87
CRC    = $6a    ; $6a..6d
bTmp   = $88    ; $88,$89

rPtr   = $8a    ; $8a,$8b

wPtr   = $fa    ; $fa,$fb

zp0    = $fc
zp1    = $fd

CAPtr  = $fe    ; $fe,$ff

prns = $ff7d

crc32_tab0 = $1300
crc32_tab1 = $1400
crc32_tab2 = $1500
crc32_tab3 = $1600
inBuf = $1700

FASTSER = $0a1c

SETBANK = $ff68     ; sta c6 stx c7
SPIN_OUT = $FF47    ; sET UP FAST SERIAL FOR INPUT OR OUTPUT.
            ;  sec FOR OUTPUT, clc FOR INPUT.
READBANK = $ff74    ; lda (ZP),y  bank in X, ZP in A
WRITEBANK = $ff77   ; sta (ZP),y  bank in X, value in A, ZP in $02b9
VERIFYBANK = $ff7a  ; cmp (ZP),y  bank in X, value in A, ZP in $02c8

JSRFAR = $02CD  ; JSR XXXX to Any Bank & Return
; 02de  return memory config
;0002       Bank Number, Jump to SYS Address
;0003-0004  SYS address, MLM register PC
;0005-0009  SYS and MLM register save (SR, A, X, Y, SP)


    ORG $1c01
    DC.B $0b,$1c,128,0  ; '128 SYS7181'
    DC.B $9e,$37,$31,$38
```

```
    DC.B $31,0,0,0

    lda $02de
    pha
    lda $ff00
    pha
    lda #$0e    ; RAM0, System ROM, RAM, BASIC
    sta $ff00
    lda #0
    sta $f7     ; enable SHIFT-C=
    sta $f8
    sta $f9
    tax
    jsr SETBANK ; I/O source/destination RAM0 64k
    jsr bank1install
#endif

    cli ; just in case..!?!?
    jsr prns
    dc.b 147
#if MACH == 4
    dc.b $b0,$ae,$20,$20,$b0,$ae,$20,$b0,$ae,$20,$20,$ae,$ae, 13
    dc.b $ab,$bd,$ae,$ae,$ac,$be,$7d,$ab,$bd,$20,$7b,$ad,$b3
#endif
#if MACH == 16
    dc.b $b0,$ae,$20,$20,$b0,$ae,$20,$b0,$ae,$20,$b0,$ae,$ae,$b0,$ae, 13
    dc.b $ab,$bd,$ae,$ae,$ac,$be,$7d,$ab,$bd,$20,$7d,$20,$7d,$ab,$ae
#endif
#if MACH == 64
    dc.b $b0,$ae,$20,$20,$b0,$ae,$20,$b0,$ae,$20,$b0,$ae,$b0,$ae,$ae,$ae, 13
    dc.b $ab,$bd,$ae,$ae,$ac,$be,$7d,$ab,$bd,$20,$7d,$20,$ab,$ae,$ad,$b3
#endif
#if MACH == 128
    dc.b $b0,$ae,$20,$20,$b0,$ae,$20,$b0,$ae,$20,$ae,$b0,$ae,$b0,$ae, 13
    dc.b $ab,$bd,$ae,$ae,$ac,$be,$7d,$ab,$bd,$20,$7d,$b0,$bd,$ab,$b3
#endif
    dc.b " V1.12     19.11.2002",13
#if MACH == 4
    dc.b $bd,$20,$ad,$bd,$ad,$bd,$bd,$bd,$20,$be,$20,$20,$bd
#endif
#if MACH == 16
    dc.b $bd,$20,$ad,$bd,$ad,$bd,$bd,$bd,$20,$be,$ad,$bd,$b1,$ad,$bd
#endif
#if MACH == 64
    dc.b $bd,$20,$ad,$bd,$ad,$bd,$bd,$bd,$20,$be,$ad,$bd,$ad,$bd,$20,$bd
#endif
#if MACH == 128
    dc.b $bd,$20,$ad,$bd,$ad,$bd,$bd,$bd,$20,$be,$ad,$ad,$bd,$ad,$bd
#endif
    dc.b " BY PASI'ALBERT'OJALA", 13
#if MACH != 20
    dc.b "        HTTP://WWW.CS.TUT.FI/%7EALBERT/", 13, 13, 0
#else
    dc.b " HTTP://WWW.CS.TUT.FI/%7EALBERT/", 13, 13, 0
#endif

    lda #8
    sta bits
```

```
#if MACH == 16 || MACH == 4
    sei
    sta $ff3f   ; RAM
#endif
#if MACH == 64
    sei
    ldy #$34
    sty 1
#endif

    ldx #0          ; create the CRC32 table (makecrc.c)
crc$    lda #0
    sta CRC+3
    sta CRC+2
    sta CRC+1
    stx CRC+0
    ldy #8
1$  lsr CRC+3
    ror CRC+2
    ror CRC+1
    ror CRC+0
    bcc 0$
    lda #$ed
    eor CRC+3
    sta CRC+3
    lda #$b8
    eor CRC+2
    sta CRC+2
    lda #$83
    eor CRC+1
    sta CRC+1
    lda #$20
    eor CRC+0
    sta CRC+0
0$  dey
    bne 1$
    lda CRC+0
    sta crc32_tab0,x
    lda CRC+1
    sta crc32_tab1,x
    lda CRC+2
    sta crc32_tab2,x
    lda CRC+3
    sta crc32_tab3,x
    inx
    bne crc$

#if MACH == 16 || MACH == 4
    sta $ff3e   ; ROM
    cli
#endif
#if MACH == 64
    ldy #$36
    sty 1
    cli
#endif

restart
#if MACH == 64 || MACH == 128
```

```
    jsr on      ; screen on / 2MHz mode if possible
#endif

idrive$ jsr prns
    dc.b "INPUT", 0

    lda #14     ; channel #
    jsr askdrive
    beq idrive$
    sta inDrive

#if MACH == 64 || MACH == 128
    jsr CheckBurst
    jsr CheckIDE64
#endif
    lda #0
    sta files
    lda #<CA
    sta CAPtr+0
    lda #>CA
    sta CAPtr+1

    jsr ReadDir

    ; Make the output buffer as big as possible,
    ; i.e. it starts right after the last file entry.
    lda CAPtr+1 ; HI
    sta obufhi  ; start of output buffer
    inc obufhi  ; right after filenames

    lda files
    cmp #255
    beq odrive$
    cmp #0
    bne files$
iagain$ lda #14     ; no files -> try again
    jsr CLOSE
    jmp idrive$

files$  ldx #<files
    ldy #>files
    lda #1
    jsr putdec
    jsr prns
    dc.b " FILES",13, 0

odrive$ jsr prns
    dc.b "OUTPUT", 0

    lda #15
    jsr askdrive
    bcs iagain$ ; RUN/STOP pressed - back to input drive
    beq odrive$ ; device not present - ask again
    sta zipDrive

    lda files
    cmp #255
    bne ofile$
    cmp inDrive
```

```
    bne ofile$
    ; image compression and one drives
    lda #15
    jsr CLOSE
    jsr prns
    dc.b "DISK IMAGE COMPRESSION NEEDS TWO DRIVES",13,0
    jmp odrive$

ofile$  jsr prns
    dc.b "ZIP FILE?", 13, 34, 0

    lda #0
    sta zipFile
    lda #<zipFile
    ldx #>zipFile
    ldy #28     ; max filename length
    jsr getstr
    sty zipLen
    lda #34
    jsr OK_KEY

    lda zipLen
    bne onok$

    lda #15     ; an empty filename.. ask the drive again
    jsr CLOSE
    jmp odrive$

onok$   lda zipFile+0
    cmp #"@"
    bne oopen$

    ; a disk command..
    lda zipLen
    cmp #1
    beq orerr$
    ldx #15
    jsr CHKOUT
    ldy #1
ocmd$   lda zipFile,y
    jsr CHROUT
    iny
    cpy zipLen
    bne ocmd$
    jsr CLRCHN

orerr$  ldx #15
    jsr geterror
    ldx #<errorStr
    lda #>errorStr
    ldy errorLen
    jsr putstr
    jmp ofile$


oopen$  ldx #<zipFile
    ldy #>zipFile
    lda zipLen
    jsr SETNAM
```

```
    lda #3
    ldx zipDrive
    ldy #1      ; open 3,8,1,"fileName"
    jsr SETLFS
    jsr OPEN

    ldx #15
    jsr geterror
    cmp #0
    beq 2$

    lda #3
    jsr CLOSE
    jsr CLRCHN
    ldx #<errorStr
    lda #>errorStr
    ldy errorLen
    jsr putstr
    jmp ofile$

2$  jsr CLRCHN
    lda files
    sta filecnt
    lda #<CA
    sta CAPtr+0
    lda #>CA
    sta CAPtr+1
    lda #0
    sta outSize+0
    sta outSize+1
    sta outSize+2
    sta outSize+3
    jsr InitBuffer32k

    jsr prns
    dc.b "ZIP COMMENT?", 13, 34, 0

    lda #0
    sta zipComment
    lda #<zipComment
    ldx #>zipComment
    ldy #36
    jsr getstr
    sty zipCommentLen
    lda #34
    jsr OK_KEY

; ---------------------------------------------

fileloop:
#if MACH == 64 || MACH == 128
    lda #0
    sta inMode
#endif
    lda files
    cmp #255
    beq disk$
    jmp fi$
```

```
    ; D64/D71/D81 read -- open direct access buffer if needed
disk$   ldy #CA_Mode
    lda #"Y"  ; always compress disk images
    sta (CAPtr),y
    ldy #0
dn$ lda diskName,y
    bpl nlow$
    eor #$a0
nlow$   sta (CAPtr),y
    beq nend$   ; NUL -> end of file name
    iny
    cpy #16
    bne dn$
nend$
    ; append .d81 .d71 or .d64
    cpy #15
    bcc append$ ; if 0..14 chars, just append
    ldy #14     ; if 15.. overwrite the end
append$ lda #"."
    sta (CAPtr),y
    iny
    lda #"D"
    sta (CAPtr),y
    iny
    lda #"8"
    sta (CAPtr),y
    iny
    lda #"1"
    sta (CAPtr),y

    lda fdos
    cmp #"D"
    beq dok$    ; ".D81"

    dey
    lda #"7"
    sta (CAPtr),y
    iny
    bit fflag
    bmi dok$    ; ".D71"

    dey
    lda #"6"
    sta (CAPtr),y
    iny
    lda #"4"
    sta (CAPtr),y   ; ".D64"
dok$
    iny

    tya
    ldy #CA_FLen
    sta (CAPtr),y

    ldy #1
    sty inTrack
    dey     ; ldy #0
    sty inSector
```

```
#if MACH == 64 || MACH == 128
    lda inBurst
    beq direct$ ; can't use burst read -> use sequential read
    inc inMode  ; burst mode, direct-access file not opened
    jmp 3$
direct$
#endif
    ldx #<d64chan$
    ldy #>d64chan$
    lda #1
    jsr SETNAM
    lda #2
    ldx inDrive
    tay ;ldy #2     ; open 2,8,2,"#"
    jsr SETLFS
    jsr OPEN

    ldx #14
    jsr geterror
    cmp #0
    beq noerr1$

    jsr CLRCHN
    ldx #<errorStr
    lda #>errorStr
    ldy errorLen
    jsr putstr

noerr1$ lda #"2"
    jsr initbp
    jmp 3$

d64chan$
    dc.b "#"

fi$
#if MACH == 64 || MACH == 128
    lda inBurst
    beq open$   ; can't use burst read -> use sequential read

    ldy #CA_TS
    lda (CAPtr),y
    beq open$   ; no T&S -> use sequential read
    sta inTrack
    iny
    lda (CAPtr),y
    sta inSector
    inc inMode
    jmp 3$
#endif

open$   ldy #CA_FLen
    lda (CAPtr),y
    ldx CAPtr+0
    ldy CAPtr+1
    jsr SETNAM

    lda #2
    ldx inDrive
```

```
    ldy #0      ; open 2,8,0,"fileName"
    jsr SETLFS
    jsr OPEN

    ldx #15
    jsr geterror
    cmp #0
    beq 3$
    lda #2
    jsr CLOSE
    ldx #<errorStr
    lda #>errorStr
    ldy errorLen
    jsr putstr
    jsr CR
    jmp fileloop    ; TODO: update fields etc.


    ; File opened and/or inTrack/inSector initialized
3$  lda #0
    sta be
    sta ST

    ldx #3
    ldy #CA_LHdr+3
5$  lda outSize+0,x
    sta (CAPtr),y       ; CA[files].lhdroff = curbyte;
    dey
    dex
    bpl 5$

    ldy #0
lhdr$   lda lhdr,y      ; Put local file header
    jsr PutBuffer
    iny
    cpy #26
    bne lhdr$

    ldy #CA_FLen
    lda (CAPtr),y
    sta curlen      ; put current file name length
    jsr PutBuffer
    lda #0
    jsr PutBuffer
    lda #0
    jsr PutBuffer
    lda #0
    jsr PutBuffer

    jsr InitBits

    ldy #0
floop$  lda (CAPtr),y       ; put filename
    bpl low$
    eor #$a0
low$    jsr PutBuffer
    lda (CAPtr),y
    jsr CHROUT
    iny
```

```
    cpy curlen
    bne floop$

    lda #" "
    jsr CHROUT

    ldx #3
6$  lda #255
    sta CRC+0,x
    lda #0
    sta usize+0,x
    lda outSize+0,x
    sta cstart+0,x
    dex
    bpl 6$

    lda #0
    sta zipMode ; not stored

file    ldy #CA_Mode
    lda (CAPtr),y
    cmp #"S"
    beq stored$
    jsr Huffman ; pass speed parameter in A
    jmp tail$

stored$ jsr Stored
tail$   ; Put data descriptor header
    lda #$50
    jsr PutBuffer
    lda #$4b
    jsr PutBuffer
    lda #$07
    jsr PutBuffer
    lda #$08
    jsr PutBuffer

    ldx #3
    ldy #CA_CRC+3
crc$    lda CRC+0,x
    eor #255
    sta (CAPtr),y
    dey
    dex
    bpl crc$

    ldx #3
    ldy #CA_CSize
    sec     ; csize =  curbyte - cstart;
subl$   lda outSize-CA_CSize,y
    sbc cstart-CA_CSize,y
    sta csize-CA_CSize,y
    sta (CAPtr),y
    iny
    dex
    bpl subl$

    ldx #3
    ldy #CA_USize
```

```
ul$ lda usize-CA_USize,y
    sta (CAPtr),y
    iny
    dex
    bpl ul$

    ; put CRC, CSize and USize
    ldx #12
    ldy #CA_CRC
cs$ lda (CAPtr),y
    jsr PutBuffer
    iny
    dex
    bne cs$

#if MACH == 64 || MACH == 128
    jsr on
    lda inMode
    bne noclose$    ; used burst read -> no close
#endif
    ldx #14
    jsr geterror
    cmp #0
    beq noerr$

    jsr CLRCHN
    ldx #<errorStr
    lda #>errorStr
    ldy errorLen
    jsr putstr
noerr$
    lda #2
    jsr CLOSE
noclose$

    jsr CLRCHN
    ldx #<usize
    ldy #>usize
    lda #4
    jsr putdec
    lda #"="
    jsr CHROUT
    lda #">"
    jsr CHROUT
    ldx #<csize
    ldy #>csize
    lda #4
    jsr putdec
    lda #" "
    jsr CHROUT
    jsr putratio

    lda #"%"
    jsr CHROUT
    jsr CR

    lda files
    cmp #255
    beq 4$
```

```
    lda CAPtr+0
    clc
    adc #CASIZE
    sta CAPtr+0
    bcc noc$
    inc CAPtr+1
noc$    dec filecnt
    beq 4$
    jmp fileloop

; ---------------------------------------------

4$  ldx #3
5$  lda #0
    sta csize,x
    lda outSize+0,x
    sta cstart+0,x
    dex
    bpl 5$

    lda files
    sta filecnt
    lda #<CA
    sta CAPtr+0
    lda #>CA
    sta CAPtr+1
headerloop
    ldy #0
ghdr$   lda ghdr,y      ; Put global header
    jsr PutBuffer
    iny
    cpy #16
    bne ghdr$

    ldy #CA_CRC
1$  lda (CAPtr),y
    jsr PutBuffer
    iny
    cpy #CA_LHdr
    bne 1$

    ldy #CA_FLen
    lda (CAPtr),y
    sta curlen
    jsr PutBuffer
    ldy #10     ;12
3$  lda #0
    jsr PutBuffer
    dey
    bpl 3$

    lda #$b4
    jsr PutBuffer
    lda #$80
    jsr PutBuffer

    ldy #CA_LHdr
    lda (CAPtr),y
```

```
    jsr PutBuffer
    iny
    lda (CAPtr),y
    jsr PutBuffer
    iny
    lda (CAPtr),y
    jsr PutBuffer
    iny
    lda (CAPtr),y
    jsr PutBuffer

    ldy #0
4$  lda (CAPtr),y   ; put filename
    bpl low$
    eor #$a0
low$    jsr PutBuffer
    iny
    cpy curlen
    bne 4$

    lda curlen
    clc
    adc #46
    bcc 7$
    inc csize+1
    clc
7$  adc csize+0
    sta csize+0
    bcc 6$
    inc csize+1
    bne 6$
    inc csize+2
    bne 6$
    inc csize+3
6$
    lda files
    cmp #255
    beq 0$

    lda CAPtr+0
    clc
    adc #CASIZE
    sta CAPtr+0
    bcc noc$
    inc CAPtr+1
noc$    dec filecnt
    beq 0$
    jmp headerloop

0$  ldy #0
ehdr$   lda ehdr,y      ; put end of header
    jsr PutBuffer
    iny
    cpy #8
    bne ehdr$

    lda files
    cmp #255
    bne 21$
```

```
    lda #1
21$ jsr PutBuffer
    lda #0
    jsr PutBuffer
    lda files
    cmp #255
    bne 22$
    lda #1
22$ jsr PutBuffer
    lda #0
    jsr PutBuffer

    ldy #0
11$ lda csize+0,y
    jsr PutBuffer
    iny
    cpy #4
    bne 11$

    ldy #0
2$  lda cstart+0,y
    jsr PutBuffer
    iny
    cpy #4
    bne 2$

    lda zipCommentLen   ; Zip comment length
    jsr PutBuffer
    lda #0
    jsr PutBuffer

    lda zipCommentLen
    beq flush$
    ldy #0
33$ lda zipComment,y
    bpl 34$
    eor #$a0
34$ jsr PutBuffer
    iny
    cpy zipCommentLen
    bne 33$

flush$  jsr FlushBuffer
#if MACH == 64 || MACH == 128
    jsr on
#endif
    lda #3
    jsr CLOSE

    ;lda #14
    ;jsr CLOSE
    ;lda #15
    ;jsr CLOSE

    jsr CLALL
    jsr CLRCHN
#if MACH == 64
    lda #$37
    sta 1
```

```
#endif
#if MACH == 128
    pla
    sta $ff00   ; restore memory config
    pla
    sta $02de   ; restore return memory config
#endif
    cli ; just in case
    rts



Huffman:
    ldy #0
    cmp #"Y"
    beq speed$
    cmp #"0"
    beq speed$
    iny
    cmp #"1"
    beq speed$
    iny
    cmp #"2"
    beq speed$
    iny
speed$  sty speed   ; speed = 0..3
;============================================================
    ; lPair must be before bSkip in memory
    ; memset(lPair, 0, sizeof(lPair));
    ; memset(bSkip, 0, sizeof(bSkip));

#if MACH == 16 || MACH == 4
    sei
    sta $ff3f   ; RAM
#endif
#if MACH == 64
    sei
    ldy #$34
    sty 1
#endif

#if MACH == 128
    jsr ClearSearch
#else
    lda #>lPair
    sta clr$+2
    ldx #>(lPairSize+bSkipSize)

    ldy #0
    tya     ;lda #0
clr$    sta $aa00,y
    iny
    bne clr$
    inc clr$+2
    dex
    bne clr$
#endif

    ;lda #0
```

```
    sta oMaxlen ; oMaxlen = 0; /* 0 = not started */
    sta oChar   ; oChar = 0;

    sta eof     ; eof = 0;
    sta skip    ; skip = 0;
    sta pos+0   ; p = 0;
    sta pos+1
    sta pos+2
    sta pos+3
    sta buc     ; bu = 0; bytes in the lookahead 'buffer'
    lda #<buffer ; wPtr = rPtr = &buffer[0];
    sta wPtr+0
    sta rPtr+0
    lda #>buffer
    sta wPtr+1
    sta rPtr+1

#if MACH == 16 || MACH == 4
    sta $ff3e   ; ROM
    cli
#endif
#if MACH == 64
    ldy #$36
    sty 1
    cli
#endif

    lda #1  ; last block, yes
    ldx #1
    jsr PutBits
    lda #1  ; fixed huffman
    ldx #2
    jsr PutBits

dataloop$
    ; Fill the lookahead buffer if no EOF seen yet..
    lda eof
    bne nofill$
    ; bu is 0 for first entry, bu is decremented each dataloop$,
    ; thus bu is never MAX_MATCH when while is entered
    ;ldx #2
    ;jsr CHKIN  ;while (!eof && bu < MAX_MATCH) {
fill$   ;jsr CHRIN  ;  int c = fgetc(fp);
#if MACH == 16 || MACH == 4
    sta $ff3e   ; ROM
    cli
#endif
#if MACH == 64
    ldy #$36
    sty 1
    cli
#endif
    jsr GetByte ;    if (c == EOF) {
#if MACH == 16 || MACH == 4
    sei
    sta $ff3f   ; RAM
#endif
#if MACH == 64
    sei
```

```
    ldy #$34
    sty 1
#endif
    ldy #0      ;   eof = 1;
    sta (wPtr),y    ;   break;
    sta delta$+1    ;    } else {
    jsr updcrc  ;   crc32 = updcrc(c, crc32);
    lda wPtr+1
    cmp #>buffer ;   *wPtr = c;
    bne nodelta$    ;   if (wPtr < &buffer[MAX_MATCH]) {
    tax
    clc
    adc #>BLOCK_SIZE
    sta wPtr+1
delta$  lda #0
    sta (wPtr),y    ;       *(wPtr + BLOCK_SIZE) = c;
    stx wPtr+1
nodelta$        ;   }
    inc wPtr+0  ;   wPtr++;
    bne wpe$
    inc wPtr+1  ;   if (wPtr == &buffer[BLOCK_SIZE]) {
    lda wPtr+1  ;       wPtr = &buffer[0];
    cmp #>(buffer+BLOCK_SIZE)
    bne wpe$
    lda #>buffer
    sta wPtr+1  ;   }
wpe$    inc buc     ;   bu++;
    bit ST
    bvc noeof$
    lda #0
    sta ST
    inc eof
    bne nofill$
noeof$          ;    }
    lda buc     ; bytes in the lookahead 'buffer'
    cmp #MAX_MATCH
    bne fill$   ;}

    ;jsr CLRCHN
nofill$
    ; Create index to our search structure and the
    ; corresponding pointer from the search string's
    ; two first characters.
#if MACH == 16 || MACH == 4
    sei
    sta $ff3f   ; RAM
#endif
#if MACH == 64
    sei
    ldy #$34
    sty 1
#endif

    ldy #1
    lda (rPtr),y
    tax
    dey
    lda (rPtr),y
#if MACH == 128
```

```
    sta arg+0
    stx arg+1
    jsr UpdSearch
#else
    ; no check.. it doesn't matter if we read 1 byte off the end
updsearch$      ;if (bu > 1) {
            ;  long off;
            ;  unsigned short index =
    asl     ;   ((*rPtr & (LSIZE-1)) << LSHIFT) |
    asl     ;   (*(rPtr+1) & (LSIZE-1));
    asl     ;  long *lPairPtr = &lPair[index];

#if LSHIFT == 4
    asl     ; shifting only leaves 4 bits
    sta zp0
    txa     ; HI
    and #15     ; take only 4 bits
    lsr     ; shift 2 bits to LO part, because
    ror zp0     ; array element is 4 bytes
    lsr
    ror zp0     ; leaves 2 bits in HI part
#endif

#if LSHIFT == 5
    sta zp0     ; shifting only leaves 5 bits
    txa
    and #31     ; take only 5 bits
    lsr     ; shift 1 bit to LO part
    ror zp0     ; leaves 3 bits in HI part
#endif
    ;clc        ; C already clear, because zp0 is *8 (or *16)
    adc #>lPair
    sta zp1
    lda pos+0   ;  unsigned short *tmp = &bSkip[p & (BLOCK_SIZE-1)];
    sta bTmp+0
    lda pos+1
    and #>(BLOCK_SIZE-1)
    asl bTmp+0
    rol
    ;clc        ; C already clear, because BLOCK_SIZE <= 32768
    adc #>bSkip
    sta bTmp+1

    ;       off = p - *lPairPtr;
    ;       if (off < (BLOCK_SIZE-MAX_MATCH)) {
    ;       *tmp = off;
    ;       } else {
    ;       *tmp = 0;
    ;       }
    ldy #0
    lda pos+0
    sec
    sbc (zp0),y
    sta off0$+1
    iny
    lda pos+1
    sbc (zp0),y
    sta off1$+1
    iny
```

```
    lda pos+2
    sbc (zp0),y
    sta off2$+1
    iny
    lda pos+3
    sbc (zp0),y
off2$   ora #0
    bne ptr0$   ;16 top bits != 0

    lda off1$+1
    cmp #>(BLOCK_SIZE-MAX_MATCH)
    bcc ptr$    ; HI byte <  --> ok!
    ; HI byte >=
    ;bne ptr0$  ; HI byte >  --> not ok
    ;todo: compare low byte -- but it doesn't matter much..

ptr0$   lda #0      ; too far or not found
    sta off0$+1
    sta off1$+1

    ; Update search structure pointers
ptr$    ldy #0
off0$   lda #0
    sta (bTmp),y    ;  *tmp=..
    iny
off1$   lda #0
    sta (bTmp),y
    dey     ;ldy #0
    lda pos+0
    sta (zp0),y ;  *lPairPtr = p;
    iny
    lda pos+1
    sta (zp0),y
    iny
    lda pos+2
    sta (zp0),y
    iny
    lda pos+3
    sta (zp0),y
#endif
    ; Skip or perform search

    lda skip    ;if (skip) {
    beq noskip$
    dec skip    ;  skip--;
    jmp eskip$
            ;} else {
noskip$ lda #1      ;short maxlen = 1, maxpos = 0, j;
    sta maxlen
    lda #<(BLOCK_SIZE-MAX_MATCH)
    sta left+0  ;int left = BLOCK_SIZE - MAX_MATCH;
    lda #>(BLOCK_SIZE-MAX_MATCH)
    sta left+1

#if 1
    ldy speed
    beq leftok$
ll$ lsr left+1
    ror left+0
```

```
    dey
    bne ll$
leftok$
#endif

    lda pos+0   ;unsigned short im = p & (BLOCK_SIZE-1);
    sta im+0
    lda pos+1
    and #>(BLOCK_SIZE-1)
    sta im+1

    ldy #0
    lda (rPtr),y
    sta nChar

while$  lda im+0    ;while ((j = bSkip[im])) {
    asl     ; array element size 2
    sta zp0
    lda im+1
    rol
    ;clc        ; C already clear, because im < 32768
    adc #>bSkip
    sta zp1
#if MACH == 128
    sei
    ldy #1
    ldx #1      ; RAM1
    lda #zp0
    jsr READBANK    ; lda (ZP),y  bank in X, ZP in A
    sta j1$+1
    dey
    ldx #1      ; RAM1
    lda #zp0
    jsr READBANK
    sta j0$+1
    cli
#else
    ldy #1
    lda (zp0),y
    sta j1$+1
    dey
    lda (zp0),y
    sta j0$+1
#endif
j1$ ora #0
    beq break$

    lda left+0
    sec
j0$ sbc #0
    sta left+0  ;left -= j;
    lda left+1
    sbc j1$+1   ;if (left < 0) {
    sta left+1  ;    break;
    bmi break$  ;}
            ;im = (im - j) & (BLOCK_SIZE-1);
    lda im+0
    sec
    sbc j0$+1
```

```
    sta im+0
    sta ap$+1
    lda im+1
    sbc j1$+1
    and #>(BLOCK_SIZE-1)
    sta im+1    ;ap = &buffer[im];
    clc
    adc #>buffer
    sta ap$+2

    ldy #0      ;j = 0;
ap$ lda $aaaa,y ;while (*ap++ == rPtr[j] && ++j < MAX_MATCH)
    cmp (rPtr),y    ;    ;
    bne out$
    iny
    bne ap$

    ; All 256 bytes were equal,
    ; but we currently handle only 8-bit MAX_MATCH i.e. 0..255
    ldy #MAX_MATCH

out$    cpy maxlen  ;if (j > maxlen) {
    bcc while$  ;    maxlen = j;
    beq while$  ;    maxpos = (int)(&rPtr[j] - ap) & (BLOCK_SIZE-1);
new$    sty maxlen  ;    if (maxlen == MAX_MATCH ||
    lda rPtr+0  ;   *rPtr == *(rPtr+1)) {
    sec     ;   break;
    sbc ap$+1   ;    }
    sta maxpos+0
    lda rPtr+1
    sbc ap$+2
    and #>(BLOCK_SIZE-1)
    sta maxpos+1

    cpy #MAX_MATCH
    beq break$

    lda nChar
    ldy #1
    cmp (rPtr),y
    beq break$  ; if seems like rle, got long enough..
            ;}
    jmp while$
break$

#if MACH == 16 || MACH == 4
    sta $ff3e   ; ROM
    cli
#endif
#if MACH == 64
    ldy #$36
    sty 1
    cli
#endif
    ; Check if we accidentally compared off the end..
    lda buc
    cmp maxlen  ;if (maxlen > bu) {
    bcs 0$
    sta maxlen  ;    maxlen = bu;
```

```
0$          ;}


    lda maxlen  ;if (maxlen >= 3) {
    sec
    sbc #3
    bcs enough$
    jmp lit$

enough$
#if 1
    lda speed   ; if (speed!=0) {
    beq lazylz$
    jsr PutLzNow    ; PutLz(fpout, maxlen, maxpos);
    lda maxlen  ; skip = maxlen-1;
    sec
    jmp cont0$  ; oMaxlen = 0;
            ; } else
#endif
lazylz$ lda oMaxlen
    sec     ;if (oMaxlen >= maxlen) {
    sbc maxlen      ;if (oMaxlen - maxlen >= 0) {
    bcc eelse$
    jsr PutLz   ;    PutLz(fpout, oMaxlen, oMaxpos);
    lda oMaxlen
    clc
cont0$  sbc #1
    sta skip    ;    skip = oMaxlen-2;
    lda #0
    sta oMaxlen ;    oMaxlen = 0; /* nothing in fifo */
    beq eend$
eelse$  lda oMaxlen ;} else {
    beq e2end$  ;    if (oMaxlen)
    clc
    lda oChar
    jsr PutFixed    ;   PutFixed(fpout, oChar);
e2end$  lda maxlen
    sta oMaxlen ;    oMaxlen = maxlen;
    lda maxpos+0
    sta oMaxpos+0
    lda maxpos+1
    sta oMaxpos+1   ;    oMaxpos = maxpos;
eend$   jmp lend$   ;}


lit$
#if 1
    lda speed   ;} else {
    beq lazylit$    ;if (speed!=0) {
    clc
    lda nChar   ; PutFixed(fpout, nChar);
    jsr PutFixed
    jmp cont1$  ; oMaxlen = 0;
            ;} else
#endif
lazylit$
    lda oMaxlen
    sec
    sbc #3      ;if (oMaxlen >= 3) {
```

```
    bcc lelse$
    jsr PutLz   ;    PutLz(fpout, oMaxlen, oMaxpos);
    lda oMaxlen
    sec
    sbc #2
    sta skip    ;    skip = oMaxlen-2;
cont1$  lda #0
    sta oMaxlen ;    oMaxlen = 0; /* nothing in fifo */
    beq lend$
lelse$  lda oMaxlen ;} else {
    beq lnolit$ ;    if (oMaxlen)
    clc
    lda oChar
    jsr PutFixed    ;   PutFixed(fpout, oChar);
lnolit$ lda #1
    sta oMaxlen ;    oMaxlen = 1;
lend$           ;}
    lda nChar
    sta oChar   ;    oChar = *rPtr;

eskip$          ;}

#if MACH == 16 || MACH == 4
    sta $ff3e   ; ROM
    cli
#endif
#if MACH == 64
    ldy #$36
    sty 1
    cli
#endif


    inc pos+0   ;p++;
    bne pe$
    inc pos+1
    bne pe$
    inc pos+2
    bne pe$
    inc pos+3
pe$

#if 1       ; progress indicator
    lda pos+0
    bne noprgupd$
#if MACH == 64 || MACH == 128
    lda pos+1
    and #31
    cmp #4
    bcs 123$
    jsr on      ; block 0,1,2, 32,33,34, 64,65,66.. turn screen on
    jmp 124$
123$    ;sec
    jsr off     ; block 3, 35, 67, 99.. screen off / 2MHz
124$
#endif
    lda pos+2
    lsr
    lsr
```

```
    lsr
    lsr
    tay
    beq pi0$
    lda hex,y
    and #63
#if MACH == 64 || MACH == 128
    sta $0422
#endif
#if MACH == 20
    sta $1010
#endif
#if MACH == 16 || MACH == 4
    sta $0c22
#endif
pi0$    lda pos+2
    and #$0f
    tay
    lda hex,y
    and #63
#if MACH == 64 || MACH == 128
    sta $0423
#endif
#if MACH == 20
    sta $1011
#endif
#if MACH == 16 || MACH == 4
    sta $0c23
#endif
    lda pos+1
    lsr
    lsr
    lsr
    lsr
    tay
    lda hex,y
    and #63
#if MACH == 64 || MACH == 128
    sta $0424
#endif
#if MACH == 20
    sta $1012
#endif
#if MACH == 16 || MACH == 4
    sta $0c24
#endif
    lda pos+1
    and #$0f
    tay
    lda hex,y
    and #63
#if MACH == 64 || MACH == 128
    sta $0425
#endif
#if MACH == 20
    sta $1013
#endif
#if MACH == 16 || MACH == 4
    sta $0c25
```

```
#endif
noprgupd$
#endif



    inc rPtr+0  ;rPtr++;
    bne re$
    ldx rPtr+1  ;if (rPtr == &buffer[BLOCK_SIZE]) {
    inx
    cpx #>(buffer+BLOCK_SIZE)
    bne reh$
    ldx #>buffer
reh$    stx rPtr+1  ;  rPtr = &buffer[0];
re$         ;}


    dec buc     ;bu--;
    bne bue$    ;if (!bu && eof) {
    lda eof     ;    break;
    bne break2$ ;}
bue$    jmp dataloop$

break2$
#if 1
    lda speed   ;if (speed!=0) {
    bne bend$   ;} else
lazyflush$
#endif
    lda oMaxlen
    sec
    sbc #3      ;if (oMaxlen >= 3) {
    bcc belse$
    jsr PutLz   ;    PutLz(fpout, oMaxlen, oMaxpos);
    jmp bend$
belse$  lda oMaxlen ;} else if (oMaxlen)
    beq bend$
    clc
    lda oChar
    jsr PutFixed    ;   PutFixed(fpout, oChar);
bend$           ;}

    sec
    lda #0
    jsr PutFixed    ;PutFixed(fpout, 256); /* EOF */

    ; save uncompressed size
    lda pos+0
    sta usize+0
    lda pos+1
    sta usize+1
    lda pos+2
    sta usize+2
    lda pos+3
    sta usize+3
    jmp FlushBits   ;FlushBits(fpout);

#if 1
PutLzNow
```

```
    ; PutLz(maxlen, maxpos)
    lda maxlen
    sta oMaxlen
    lda maxpos+0
    sta oMaxpos+0
    lda maxpos+1
    sta oMaxpos+1
#endif
    ; PutLz(oMaxlen, oMaxpos)
PutLz   lda oMaxlen
    sec
    sbc #3
    sta mx3$+1
    ldy #0      ;  j = 0;
cpl$    cmp cplens_3,y  ;  while (maxlen-3 >= cplens[j]-3)
    bcc cple$   ;    j++;
    iny
    bne cpl$
cple$   tya
    dey     ;  j--;
    sec
    jsr PutFixed    ;  PutFixed(fpout, 256+j+1);
    ldx cplext,y    ;  if (cplext[j]) {
    beq cps$
mx3$    lda #0      ; 0 means 256, 1 means 257, 2 means 258
    sec
    sbc cplens_3,y  ;    PutBits(fpout, cplext[j], (maxlen-3)-(cplens[j]-3));
    jsr PutBits ;  }
cps$    ldy #0      ;  j = 0;
cpd$    lda oMaxpos+1   ;  while (maxpos >= cpdist[j])
    cmp cpdistHi,y
    bcc cpde$
    bne cpdn$
    lda oMaxpos+0
    cmp cpdistLo,y
    bcc cpde$
cpdn$   iny     ;    j++;
    bne cpd$
cpde$   dey     ;  j--;
    tya
    ldx #5
    jsr PutBitsR    ;  PutBitsR(fpout, 5, j);
    lda oMaxpos+0
    sec
    sbc cpdistLo,y
    sta zp0
    lda oMaxpos+1
    sbc cpdistHi,y
    sta zp1

    ldx cpdext,y    ;  if (cpdext[j]) {
    beq onee$
    lda zp0 ; load in advance
    cpx #9      ;    if (cpdext[j] > 8) {
    bcc zerto8$
    ldx #8      ;      PutBits(fpout, 8, maxpos-cpdist[j]);
    ;lda zp0
    jsr PutBits
    lda cpdext,y
```

```
    sec
    sbc #8
    tax
    lda zp1
    ;jmp PutBits    ;      PutBits(fpout, cpdext[j]-8, (maxpos-cpdist[j])>>8);
    ; jsr + rts

zerto8$ ;lda zp0    ;    } else {
    jmp PutBits ;      PutBits(fpout, cpdext[j], maxpos-cpdist[j]);
    ; jsr + rts ;    }
onee$   rts     ;  }

;===============================================================


Stored:
    jsr FlushBuffer ; save the header
    ;-> next bytes will be at the start of the buffer

    lda #$80
    sta zipMode ; stored

real$   ; put fake values -- FlushBuffer will fix them
    lda #0
    sta flushed
    jsr PutBuffer   ; not last block, stored
    jsr PutBuffer
    jsr PutBuffer
    jsr PutBuffer
    jsr PutBuffer

again$  jsr GetByte
    ldy ST
    sty last
    pha
#if MACH == 16 || MACH == 4
    sei
    sta $ff3f   ; RAM
#endif
#if MACH == 64
    sei
    ldy #$34
    sty 1
#endif
    jsr updcrc
#if MACH == 16 || MACH == 4
    sta $ff3e   ; ROM
    cli
#endif
#if MACH == 64
    ldy #$36
    sty 1
    cli
#endif
    pla
    jsr PutBuffer

    ; Update uncompressed length
    inc usize+0
```

```
    bne 2$
    inc usize+1
    bne 2$
    inc usize+2
    bne 2$
    inc usize+3
2$  bit last
    bvs leave$  ; last block saved, time to leave
    lda flushed
    bne real$   ; PutBuffer flushed -> must insert another header
    beq again$  ; Not flushed, continue adding to output buffer..

leave$  jsr FlushBuffer
    lda #0
    sta zipMode
    rts


;--------------- Bit/byte input routines ---------------------

initbp  pha
    ldx #14
    jsr CHKOUT
    lda #"B"
    jsr CHROUT
    lda #"-"
    jsr CHROUT
    lda #"P"
    jsr CHROUT
    lda #" "
    jsr CHROUT
    pla
    jsr CHROUT
    lda #" "
    jsr CHROUT
    lda #"0"
    jsr CHROUT
    jmp CLRCHN


lastSector
    dc.b 0


sectorsPerTrack:
    dc.b 21,21,21,21,21,21,21,21    ;1..8
    dc.b 21,21,21,21,21,21,21,21    ;9..16
    dc.b 21,19,19,19,19,19,19,19    ;17..24
    dc.b 18,18,18,18,18,18,17,17    ;25..32
    dc.b 17,17,17           ;33..35

GetByteD64
    ldx be
    beq fill$
    jmp read$

fill$   lda #0
    sta lastSector
```

```
#if MACH == 64 || MACH == 128
    lda inBurst
    beq std$

    lda #4
    sta retry$

again$  jsr BREAD
    lda status
    and #15
    cmp #2
    bcs chke$
    jmp bok$
chke$   cmp #11
    bne berr$
    ; Disk changed -- the drive is confused
    ldx #14
    jsr BINQ
    dec retry$
    beq berr$
    jmp again$

retry$  dc.b 0
#endif

std$
#if MACH == 64 || MACH == 128
    clc     ; screen off, but 1MHz
    jsr off
#endif
    ldx #14
    jsr CHKOUT
    lda #"U"
    jsr CHROUT
    lda #"1"
    jsr CHROUT
    lda #" "
    jsr CHROUT
    lda #"2"  ; channel
    jsr CHROUT
    lda #" "
    jsr CHROUT
    lda #"0"  ; unit
    jsr CHROUT

    lda inTrack ; track
    jsr putval
    lda inSector    ; sector
    jsr putval
    jsr CLRCHN

    ldx #14
    jsr geterror    ; waits until the read is finished..
    cmp #0
    beq noerr$
#if MACH == 64 || MACH == 128
    jsr on
#endif
    ldx #<errorStr
```

```
    lda #>errorStr
    ldy errorLen
    jsr putstr
noerr$
    ldx #2
    jsr CHKIN
    ldy #0
b2loop$ jsr CHRIN
    sta inBuf,y
#if MACH == 64 || MACH == 128
    sta $d020
#endif
#if MACH == 20
    and #7
    eor $900f
    sta $900f
#endif
#if MACH == 16 || MACH == 4
    sta $ff19
#endif
    iny
    bne b2loop$
    jsr CLRCHN
    lda #0
    sta ST
#if MACH == 64 || MACH == 128
    sec     ; screen off, 2MHz
    jsr off
#endif
    jmp bok$

#if MACH == 64 || MACH == 128
berr$   jsr on
    ldx inTrack
    lda inSector
    jsr BERR
#endif

bok$    inc inSector    ; advance to the next sector
    lda fdos
    cmp #"D"
    bne 1541$
    lda #80+1
    sta ltrack$+1   ; 1581 has 80 tracks
    lda inSector
    cmp #40
    beq ntrack$ ; next track for 1581
    bne chk$

1541$   lda #35+1   ; one-sided has 35 tracks
    bit fflag
    bpl onesd$
    lda #2*35+1 ; double-sided has 70 tracks
onesd$  sta ltrack$+1
    lda inTrack
    cmp #36
    bcc cmp$
    sbc #35
cmp$    tax
```

```
    lda inSector
    cmp sectorsPerTrack-1,x
    bcc sectok$
ntrack$ lda #0
    sta inSector
    inc inTrack
chk$    lda inTrack
ltrack$ cmp #35+1
    bne sectok$
    inc lastSector  ; flag for EOF
sectok$
    ldx #0
    stx be

read$   inc be
    bne ok$
    lda lastSector
    beq ok$
    lda ST
    ora #$40
    sta ST
ok$ lda inBuf,x
    rts


GetByte lda files
    cmp #255
    beq GetByteD64k$
    ldx be
    beq fill0$

read$   inc be
    lda inBuf+0
    bne 1$
    cpx inBuf+1
    bne 1$
    lda #0
    sta be
    lda ST
    ora #$40
    sta ST
1$  lda inBuf,x
    rts

GetByteD64k$
    jmp GetByteD64

fill0$
#if MACH == 64 || MACH == 128
    lda inMode
    beq std$    ; can't use burst read -> faster to do it sequentially
    lda inTrack
    beq eof$
    ; can use burst and t&s known -> use burst read
    jsr BREAD
    lda status
    and #15
    cmp #2
    bcc bok$
```

```
    jsr on
    ldx inTrack
    lda inSector
    jsr BERR
    lda #0
    sta inBuf+0
    lda #255
    sta inBuf+1

bok$    lda inBuf+0 ; copy links for next read
    sta inTrack
    lda inBuf+1
    sta inSector
eof$    ldx #2
    stx be
    bne read$
#endif

std$    ldx #2
    stx be
    stx inBuf+0 ; Fake T&S
    stx inBuf+1

    lda #0
    sta ST

#if MACH == 64 || MACH == 128
    clc     ; screen off, but 1MHz
    jsr off
#endif

    ldx #2
    jsr CHKIN

#if MACH == 64 || MACH == 128
    lda IDE64present
    bpl fill$   ; IDE64 not found
    tya
    pha
    lda #IDE64BR
    ldx #254
    ldy #0
    jsr IDE64BlockRead
    pla
    tay
    bcs fill$   ; Tried to read from a floppy...
    inx
    inx
    stx be
    bit ST
    bvc break$
    dex
    bvs IDE64notall$
#endif

fill$   jsr CHRIN
    ldx be
    sta inBuf,x
```

```
#if MACH == 64 || MACH == 128
    sta $d020
#endif
#if MACH == 20
    and #7
    eor $900f
    sta $900f
#endif
#if MACH == 16 || MACH == 4
    sta $ff19
#endif
    bit ST
    bvc 0$
    inc be

IDE64notall$
    lda #0
    sta ST
    sta inBuf+0 ; EOF T
    stx inBuf+1 ; index of last valid byte
    jmp break$
0$  inc be
    bne fill$

break$  jsr CLRCHN
#if MACH == 64 || MACH == 128
    sec     ; screen off, 2MHz
    jsr off
#endif
    ldx #2
    stx be

    jmp read$


;--------------- Output/buffer routines ---------------------

updcrc:
    ; updcrc()
    ; (crc32_tab[((int)crc ^ cp) & 255] ^ (crc >> 8))
    sty y$+1
    eor CRC+0       ; 3
    tay         ; 2
    lda crc32_tab0,y    ; 4
    eor CRC+1       ; 3
    sta CRC+0       ; 3
    lda crc32_tab1,y    ; 4
    eor CRC+2       ; 3
    sta CRC+1       ; 3
    lda crc32_tab2,y    ; 4
    eor CRC+3       ; 3
    sta CRC+2       ; 3
    lda crc32_tab3,y    ; 4
    sta CRC+3       ; 3 =42 cycles/byte = ~7.5 secs for 170.75kB
y$  ldy #0
    rts


; $80 0  $40 1  $20 2  $10 3  $08 4  $04 5  $02 6  $01 7
```

```
InitBits
    lda #$80
    sta bits
    rts

bits    dc.b $80

FlushBits
    ldx bits
    bpl 1$
    rts
1$  lsr byte
    lsr bits
    bne 1$
    ;sec
    ror bits    ; -> $80
    lda byte
    jmp PutBuffer

PutBits cpx #0
    bne 1$
    rts
1$  lsr
    ror byte
    lsr bits
    bne 3$
    ;sec
    ror bits    ; -> $80
    pha
    lda byte
    jsr PutBuffer
    pla
3$  dex
    bne 1$
    rts


maskTab dc.b 1,2,4,8,16,32,64,128
PutBitsR
    sta tmp
    dex
    bpl 0$
    rts
0$  lda maskTab,x
    bit tmp
    clc
    beq 1$
    sec
1$  ror byte
    lsr bits
    bne 3$
    ;sec
    ror bits    ; -> $80
    lda byte
    jsr PutBuffer
3$  dex
    bpl 0$
    rts
```

```
PutFixed    ; C*256 + A
    bcc lt256$
    cmp #24
    bcs ge24$
    ;256+ 0..23
    ldx #7
    jmp PutBitsR
ge24$   adc #167    ;256+ 24..31
    ldx #8
    jmp PutBitsR
lt256$  cmp #144
    bcs ge144$
    adc #48
    ldx #8
    jmp PutBitsR
ge144$  pha
    ldx #1
    lda #1
    jsr PutBits
    pla
    ldx #8
    jmp PutBitsR


    ; 70+12 cycles/byte (w/ CRC32)
PutBuffer
    sty puty+1  ;4
    stx putx+1
    inc outSize+0   ; (256*9+(256*8+(256*8+5)/256)/256)/256
    bne 0$      ;  = 9.0314 cycles/byte
    inc outSize+1
    bne 0$
    inc outSize+2
    bne 0$
    inc outSize+3
0$

outCmd  sta $aaaa

    inc outCmd+1    ;6
    bne 2$      ;2/3
    inc outCmd+2
    lda outCmd+2
    cmp #>outend
    bne 2$
    ; Save buffer data buffer32k .. (outCmd)
    jsr FlushBuffer
2$
puty    ldy #0  ; restore Y value
putx    ldx #0
    rts



FlushBuffer ; Can trash A & Y, X
#if MACH == 64 || MACH == 128
    clc     ; screen off, but 1MHz
    jsr off
```

```
#endif

    lda zipMode
    beq std$
    ; if Stored, fix the block size, its complement etc.

    lda obufhi  ; fix the HI parts
    sta bu0$+2
    sta bu1$+2
    sta bu2$+2
    sta bu3$+2
    sta bu4$+2
    bit last
    bvc 5$
bu0$    inc $2000+0 ; 0 -> 1 == last block, stored
5$  lda outCmd+1
    sec
    sbc #5
bu1$    sta $2000+1 ; size LO
    eor #255
bu2$    sta $2000+3
    lda outCmd+2
    sbc obufhi
bu3$    sta $2000+2 ; size HI
    eor #255
bu4$    sta $2000+4

std$    ldx #3
    jsr CHKOUT  ; (sends LISTEN+SECOND)

    lda #0
    sta bTmp+0
    lda obufhi
    sta bTmp+1

1$
#if MACH == 64 || MACH == 128
    lda IDE64present
    bpl noIDE64here$    ; IDE64 not found
    lda outCmd+1
    sec
    sbc bTmp+0
    tax
    lda outCmd+2
    sbc bTmp+1
    tay
    lda #bTmp
    jsr IDE64BlockWrite
    bcc IDE64writeback$ ;Tried to write to a floppy...
noIDE64here$
#endif
    lda bTmp+0
    cmp outCmd+1
    bne 0$
    lda bTmp+1
    cmp outCmd+2
    bne 0$

IDE64writeback$
```

```
    jsr CLRCHN
    jsr InitBuffer32k
    inc flushed
    rts

0$  ldy #0
    lda (bTmp),y
#if MACH == 64 || MACH == 128
    sta $d020
    jsr CHROUT
#endif
#if MACH == 20
    pha
    jsr CHROUT
    pla
    and #7
    eor $900f
    sta $900f
#endif
#if MACH == 16 || MACH == 4
    sta $ff19
    jsr CHROUT
#endif
    inc bTmp+0
    bne 1$
    inc bTmp+1
    jmp 1$

InitBuffer32k
    ; Init buffer pointer
    lda #0
    sta outCmd+1
    lda obufhi
    sta outCmd+2
    rts

flushed dc.b 0


;--------------- Misc utility routines ---------------------
    ;      0   1   2   3   4   5   6   7   8   9     10  11  12   13  14   15  16  17
dec0    dc.b $00,$00,$80,$40,$a0,$10,$e8,$64,$0a,$01,   $00,$00,$00, $99,$5c, $89,$8d,
dec1    dc.b $ca,$e1,$96,$42,$86,$27,$03,$00,$00,$00,   $00,$00,$00, $99,$8f, $41,$06,
dec2    dc.b $9a,$f5,$98,$0f,$01,$00,$00,$00,$00,$00,   $00,$00,$00, $19,$02, $00,$00,
dec3    dc.b $3b,$05,$00,$00,$00,$00,$00,$00,$00,$00,   $64,$0a,$01, $00,$00, $00,$00,
decv    dc.b 0,0,0,0
decs    dc.b 0
decout  dc.b 0

putpct  sta decbyt+1
    lda #10
    sta decstrt+1
    lda #18
    jmp putdec2

putdec  sta decbyt+1
    lda #0
    sta decstrt+1
    lda #10
```

```
putdec2 sta decend+1
    stx bTmp+0
    sty bTmp+1
    lda #0
    sta decs
    sta decout
    sta decv+1
    sta decv+2
    sta decv+3

decbyt  ldy #4
    dey
0$  lda (bTmp),y
    sta decv,y
    dey
    bpl 0$

decstrt ldy #0
decend
for$    cpy #10
    bne 2$
    lda decout
    rts
2$  ldx #0
while$  lda decv+3
    cmp dec3,y
    bcc breakw$ ; a < dec
    bne ge$     ; a > dec
    lda decv+2
    cmp dec2,y
    bcc breakw$ ; a < dec
    bne ge$     ; a > dec
    lda decv+1
    cmp dec1,y
    bcc breakw$ ; a < dec
    bne ge$     ; a > dec
    lda decv+0
    cmp dec0,y
    bcc breakw$ ; a < dec
ge$ inx     ; a >= dec
    sec
    lda decv+0
    sbc dec0,y
    sta decv+0
    lda decv+1
    sbc dec1,y
    sta decv+1
    lda decv+2
    sbc dec2,y
    sta decv+2
    lda decv+3
    sbc dec3,y
    sta decv+3
    jmp while$
breakw$
    cpy #9
    beq put$
    txa
```

```
    ora decs
    beq noput$
put$    txa
    ora #"0"
    jsr CHROUT
    inc decout
    inc decs
noput$  cpy #14     ; 12 for decimal, 14 for %
    bne 1$
    lda #"."
    jsr CHROUT
    inc decout
    inc decs
1$  iny
    jmp for$

ratioa  dc.b 0,0,0,0
ratiob  dc.b 0,0,0,0
ratiov  dc.b 0,0,0,0
ratiosh dc.b 0

putratio
    ldy #3
0$  lda csize,y
    sta ratioa,y
    lda usize,y
    sta ratiob,y
    lda #0
    sta ratiov,y
    dey
    bpl 0$
    ;lda #0
    sta ratiosh
while0$         ; scale divider
    ldy #3
1$  lda ratiob,y
    cmp ratioa,y
    bcc lt$
    bne breakw$
    dey
    bpl 1$
    jmp breakw$ ; b==a
lt$ ; b < a
    asl ratiob+0
    rol ratiob+1
    rol ratiob+2
    rol ratiob+3
    inc ratiosh
    jmp while0$
breakw$
    ldx #0
for$    ldy #3
2$  lda ratioa,y
    cmp ratiob,y
    bcc break2$
    bne ge$
    dey
    bpl 2$
ge$ ; a >= b
```

```
    sec
    lda ratioa+0
    sbc ratiob+0
    sta ratioa+0
    lda ratioa+1
    sbc ratiob+1
    sta ratioa+1
    lda ratioa+2
    sbc ratiob+2
    sta ratioa+2
    lda ratioa+3
    sbc ratiob+3
    sta ratioa+3
    sec
break2$
    rol ratiov+0
    rol ratiov+1
    rol ratiov+2
    rol ratiov+3

    asl ratioa+0
    rol ratioa+1
    rol ratioa+2
    rol ratioa+3
    inx
    cpx #25     ; 25-bit result
    bne for$

5$  dec ratiosh
    bmi 4$
    asl ratiov+0
    rol ratiov+1
    rol ratiov+2
    rol ratiov+3
    jmp 5$
4$
    ldx #<ratiov
    ldy #>ratiov
    lda #4
    jmp putpct

puthex  pha
    lsr
    lsr
    lsr
    lsr     ; 0..9 -> $30..$39, A..F -> $41..$46
    tay
    lda hex,y
    jsr CHROUT
    pla
    and #15
    tay
    lda hex,y
    jmp CHROUT
hex dc.b "0123456789ABCDEF"


#if MACH == 64 || MACH == 128
on
```

```
#if MACH == 64
    lda $d011
    ora #$10
    sta $d011
    lda #0      ; 1 MHz
    sta $d030
#else
    lda $d7     ; 40/80 columns
    bne off     ; 80 columns -> turn it off
    sta $d030   ; 1 MHz
    lda $d011
    ora #$10    ; blank off
    sta $d011
#endif
    rts

#if MACH == 64
    ; NOTE: standard slow serial does not work with 2MHz in C64 mode
onemeg  lda #0      ; 1MHz
    sta $d030
#endif

off lda $d011
    and #$ef
    sta $d011
#if MACH == 64
    ; NOTE: standard slow serial does not work with 2MHz in C64 mode
    lda #0
    rol     ; clc = 1MHz, sec = 2 MHz
    sta $d030
#else
    ; NOTE: standard slow serial does work with 2MHz in C128 mode
    lda #1
    sta $d030
#endif
    rts
#endif

putstr  stx 0$+1
    sta 0$+2
    sty 1$+1
    jsr CLRCHN
    ldy #0
0$  lda $aaaa,y
    jsr CHROUT
    iny
1$  cpy #0
    bne 0$
    rts



getstr  sta 0$+1    ; <buffer
    sta 2$+1
    stx 0$+2    ; >buffer
    stx 2$+2
    sty 1$+1    ; max len+1
    jsr CLRCHN
    ldy #0
```

```
0$  lda $aaaa,y
    beq edit$
    jsr CHROUT
    iny
    bne 0$

edit$   sty 3$+1
    jsr getkey
3$  ldy #0
    bcs edit$
    cmp #13     ; return
    beq quit$
    cmp #157    ; cursor left
    beq bs$
    cmp #20     ; backspace
    bne no8$
bs$ cpy #0
    beq edit$
    jsr CHROUT  ; <- destructive bs?
    dey
    jmp edit$
no8$    cmp #" "
    bcc edit$   ; 0..31
1$  cpy #0
    beq edit$   ; no more room
2$  sta $aaaa,y
    jsr CHROUT
    iny
    jmp edit$
quit$   rts


getkey  jsr CLRCHN
    jsr GETIN
    clc
    bne ok$
    sec
ok$ rts


geterror:   ; read the disk error channel - channel in X
#if MACH == 16 || MACH == 4
    ldy #30
11$ dey
    bpl 11$
#endif
    jsr CHKIN
#if MACH == 16 || MACH == 4
    ldy #30
12$ dey
    bpl 12$
#endif
    jsr CHRIN   ; read the first digit
    sta errorStr+0
    and #15
    tax     ; and store it
    jsr CHRIN   ; read the second digit
    sta errorStr+1
    and #15
```

```
    clc
    adc tentimes,x
    sta errorCode
    pha
    ;beq skip$  ; no error -> do not read the rest
    ldy #2
2$  jsr CHRIN
    sta errorStr,y
    iny
    cmp #13     ; IDE64 does not set ST at the end of the error string
    beq 3$
    lda ST      ; READST
    beq 2$
3$  sty errorLen
skip$   jsr CLRCHN  ; clear the channel
    pla
    rts     ; return (error code in accumulator)


oPages  dc.b 0

putval  pha     ; 0..99 decimal
    lda #" "
    jsr CHROUT
    pla

    ldx #0
1$  cmp #10
    bcc 0$
    inx
    sec
    sbc #10
    jmp 1$
0$  cpx #0
    beq 2$      ; turn off leading zero
    pha
    txa
    clc
    adc #"0"
    jsr CHROUT
    pla
2$  clc
    adc #"0"
    jmp CHROUT


#if MACH != 128
prns    jsr CLRCHN
    pla
    sta addr$+1
    pla
    sta addr$+2
loop$   inc addr$+1
    bne addr$
    inc addr$+2
addr$   lda $aaaa
    beq out$
    jsr CHROUT
    jmp loop$
```

```
out$    lda addr$+2
    pha
    lda addr$+1
    pha
    rts
#endif



;--------------- Directory scan/list routine --------------------

dir dc.b "$0:*"
fname   ds.b 17
fnameln dc.b 0
ftype   dc.b 0
ftrack  dc.b 0
fsector dc.b 0
fblocks dc.b 0,0
ftypstr dc.b "SEQ PRG USR REL CBM ?6? ?7? "
fcount  dc.b 0
fdos    dc.b 0
fflag   dc.b 0  ; Double-sided flag
fdtype  dc.b 0
diskName
    ds.b 23



ReadDir lda #0
    sta fcount
    sta fdtype
#if MACH == 64 || MACH == 128
    jsr prns
    dc.b 13
    dc.b 18,"D",146," DISK IMAGE COMPRESSION (D64/D71/D81)",13
    dc.b 18,"N",146," DO NOT ARCHIVE THIS FILE",13
    dc.b 18,"Y",146," ARCHIVE WITH MAXIMUM COMPRESSION",13
    dc.b 18,"1",146,"-",18,"3",146," ARCHIVE WITH FASTER COMPRESSION",13
    dc.b 18,"S",146," ARCHIVE STORED (NO COMPRESSION)",13
    dc.b 18,"Q",146," QUIT MENU AND BEGIN CREATING ARCHIVE",13
    dc.b 13, 0
#endif
    ; Open directory
    ldx #<dir
    ldy #>dir
    lda #4
    jsr SETNAM

    ; We can't use "I0" because of IDE64
    ; We have this hack instead.
    lda #2      ; first a dummy open
    ldx inDrive
    ldy #0      ; open 2,8,0,"$0:*"
    jsr SETLFS
    jsr OPEN
    lda #2
    jsr CLOSE
```

```
    lda #2      ; first try raw directory
    ldx inDrive
    ldy #2      ; open 2,8,2,"$0:*"
    jsr SETLFS
    jsr OPEN

    ldx #14
    jsr geterror
    cmp #0
    beq 2$

    ldx #<errorStr
    lda #>errorStr
    ldy errorLen
    jsr putstr

    lda #2
    jsr CLOSE

    lda #2      ; could not open raw, try normal directory
    ldx inDrive
    ldy #0      ; open 2,8,0,"$0:*"
    jsr SETLFS
    jsr OPEN

    inc fdtype

    lda ST
    beq 2$

quit0$  lda #2
    jsr CLOSE
    jsr CLRCHN

    ldx #14
    jsr geterror
    ldx #<errorStr
    lda #>errorStr
    ldy errorLen
    jmp putstr

2$  ldx #2      ; disk name: 142..164 165..168 unused
    jsr CHKIN
    jsr CHRIN   ; get disk format "A" for 1541/51/71/4040 or "D" for 1581
    sta fdos    ; "C" for 8050
    lda ST
    bne quit0$
    jsr CHRIN   ; get double-sided flag for "A"
    sta fflag
    cmp #$04
    bne notide$
    lda fdos
    cmp #$01
    bne notide$
    ; probably IDE64 or some other device that does not support
    ; raw directories
    inc fdtype

    ldy #6
```

```
is0$    jsr CHRIN
    cmp #34
    beq is01$
    dey
    bne is0$    ; skip line link, line number, reverse + "
is01$

    ldy #0
is1$    jsr CHRIN   ; get disk name, disk ID and dos/disk version
    sta diskName,y
    cmp #0
    beq gotdisk$
    iny
    cpy #23
    bne is1$
is2$    jsr CHRIN   ; get 0
    cmp #0
    bne is2$
    jmp gotdisk$


notide$ lda fdos
    cmp #"D"
    beq 1581$   ; no skip for 1581 & CMD Ramlink & CMD HD
    cmp #"H"
    beq 1581$
    ldy #2      ; skip 2 bytes for 8050
    cmp #"C"
    beq skip0$
    ldy #140    ; skip 140 bytes for 1541

skip0$  jsr CHRIN
    dey
    bne skip0$  ; skip disk info

1581$   ldy #0
dname$  jsr CHRIN   ; get disk name, disk ID and dos/disk version
    sta diskName,y
    iny
    cpy #23
    bne dname$

    lda fdos
    cmp #"C"
    bne nskp2$
    ldy #253    ; skip 735-229=506=253*2 bytes for 8050
skip2$  jsr CHRIN
    jsr CHRIN
    dey
    bne skip2$
    ; TODO: skip extra 2*254 bytes for double-sided 8050 disks

nskp2$  ldy #229    ; skip 229 bytes for 1581 and 8050, CMD Ramlink & HD
    lda fdos
    cmp #"A"
    bne skip1$
    ldy #89     ; skip 89 bytes for 1541
skip1$  jsr CHRIN
    dey
```

```
    bne skip1$  ; skip disk info

gotdisk$        ; display the disk name
    jsr CLRCHN

    lda #18
    jsr CHROUT
    lda #34
    jsr CHROUT
    ldy #0
dname2$ lda diskName,y
    cpy #16
    bne yes$
    lda #34
yes$    jsr CHROUT
    iny
    cpy #23
    bne dname2$
    jsr prns
    dc.b 146,13,0


dirloop$
    ldx #2
    jsr CHKIN

    lda fdtype
    bne ide2$
    jmp noide2$

ide2$   jsr CHRIN   ; skip line link
    jsr CHRIN

    jsr CHRIN
    sta fblocks+0
    lda ST
    bne ret$
    jsr CHRIN
    sta fblocks+1

is3$    jsr CHRIN
    tax
    beq ret$    ; xx BLOCKS FREE.   -- zero before quote
    lda ST
    bne ret$
    cpx #34
    bne is3$

    ldy #0
is4$    jsr CHRIN   ; name
    cmp #34
    beq is5$
    sta fname,y
    iny
    cpy #17
    bne is4$
    dey
is5$    lda #0
    sta fname,y
```

```
    sty fnameln

is6$    jsr CHRIN
    iny
    cpy #18
    bne is6$

    ldx #$81
    cmp #"S"  ; "SEQ"
    beq is7$
    inx
    cmp #"P"  ; "PRG"
    beq is7$
    inx
    cmp #"U"  ; "USR"
    beq is7$
    inx
    cmp #"R"  ; "REL"
    beq is7$
    inx
    cmp #"C"  ; "CBM"
    beq is7$
    cmp #"D"  ; "DIR" (IDE64)
    beq is7$
    ldx #$80    ; "DEL"
is7$
    stx ftype

is8$    jsr CHRIN
    ldx ST
    bne ret$
    cmp #0
    bne is8$
    sta ftrack
    sta fsector
    jmp noskip$

ret$    lda #2
    jsr CLOSE
    jmp CLRCHN  ; jsr + rts


noide2$ jsr CHRIN   ; file type 0
    sta ftype
    lda ST
    and #$c0
    bne ret$

    jsr CHRIN   ; track     1
    sta ftrack
    jsr CHRIN   ; sector    2
    sta fsector
    ldy #0      ; 16
nameloop$
    jsr CHRIN   ; name      3-18 (0-15)
    cmp #$a0
    beq nameend$
    sta fname,y
    iny
```

```
    cpy #16
    bne nameloop$
    lda #0
    sta fname,y
    sty fnameln
    jmp other$
nameend$
    lda #0
    sta fname,y
    sty fnameln
    iny
other$  jsr CHRIN   ; skip the rest of the filename & other fields
    iny
    cpy #25     ; 19-27 (16-24)
    bne other$
    jsr CHRIN   ; blocks lo 28
    sta fblocks+0
    jsr CHRIN   ; blocks hi 29
    sta fblocks+1

    inc fcount
    lda fcount
    and #7
    beq noskip$
    jsr CHRIN   ; 'link'    30
    jsr CHRIN   ; 'link'    31

noskip$
    jsr CLRCHN
    lda ftype
    and #7
    bne notdel$
    jmp dirloop$    ; skip DELeted files

notdel$
    ldx #<fblocks
    ldy #>fblocks
    lda #2
    jsr putdec
    tay
0$  cpy #4
    bcs 1$
    lda #" "
    jsr CHROUT
    iny
    jmp 0$

1$  ldx #" "
    lda ftype
    bmi closed$
    ldx #"*"
closed$
    txa
    jsr CHROUT

    lda ftype
    and #7
    asl
    asl ; *4
```

```
    clc
    adc #<(ftypstr-4)
    tax
    lda #>(ftypstr-4)
    adc #0
    ldy #3
    jsr putstr

    lda #"<"
    bit ftype
    bvs locked$
    lda #" "
locked$
    jsr CHROUT

    lda #34
    jsr CHROUT
    ldx #<fname
    lda #>fname
    ldy fnameln
    jsr putstr
    lda #34
    jsr CHROUT
    ldy fnameln
tasaa$  lda #" "
    jsr CHROUT
    iny
    cpy #17
    bne tasaa$

    jsr prns
    dc.b 18,"Y",146," ",18,"S",146," ",18,"D",146," ",18,"N",146," ",18,"Q",146,"?", 0

ack$    jsr getkey
    bcs ack$
    cmp #"Y"
    beq this$
    cmp #"0"
    beq this$
    cmp #"1"
    beq this$
    cmp #"2"
    beq this$
    cmp #"3"
    beq this$
    cmp #"S"
    beq this$
    cmp #3
    beq quit$
    cmp #"Q"
    beq quit$
    cmp #"D"
    beq d64$
    cmp #"N"
    bne ack$

#if MACH != 128
    ldy #37
lnoff$  lda #20     ; backspace
```

```
    jsr CHROUT
    dey
    bpl lnoff$
#else
    ldy #37
lnoff$  lda #157    ; <--
    jsr CHROUT
    lda #32     ; space
    jsr CHROUT
    lda #157    ; <--
    jsr CHROUT
    dey
    bpl lnoff$
#endif
    jmp dirloop$

d64$    lda fdos
    cmp #"A"  ; 1541/1571 disk
    beq dok$
    cmp #"D"  ; 1851 disk
    beq dok$
    jmp ack$    ; not a disk format we support!

dok$    lda #255
    sta files
quit$   lda #2
    jsr CLOSE
    jsr CLRCHN
    jmp CR

this$   sta tmp
    jsr OK_KEY
    lda #0
    ldy #CASIZE-1
clr$    sta (CAPtr),y   ; clear
    dey
    bpl clr$

    ldy #0
cp$ lda fname,y
    sta (CAPtr),y
    iny
    cpy fnameln
    bne cp$
    lda ftype
    and #7
    cmp #2
    beq typok$  ; If not PRG, append ",S" ",U" ",R"
    lda #","
    sta (CAPtr),y
    iny
    lda ftype
    and #7
    asl
    asl
    tax
    lda ftypstr-4,x
    sta (CAPtr),y
    iny
```

```
typok$  tya
    ldy #CA_FLen
    sta (CAPtr),y   ; name lenght
    ldy #CA_TS
    lda ftrack
    sta (CAPtr),y   ; start track
    iny
    lda fsector
    sta (CAPtr),y   ; start sector
    lda tmp
    ldy #CA_Mode
    sta (CAPtr),y   ; compress mode

    inc files
    lda CAPtr+0
    clc
    adc #CASIZE
    sta CAPtr+0
    bcc noc$
    inc CAPtr+1
noc$    lda files
    cmp #MAXFILES
    beq toomany$
    jmp dirloop$
toomany$
    lda #2
    jsr CLOSE
    jsr CLRCHN
    jsr prns
    dc.b "MAXIMUM NUMBER OF FILES REACHED",13,0
    rts


askdrive:
    sta chan$+1
    jsr prns
    dc.b " DRIVE? ",18,"8",146,"-",18,"9",146,",1",18,"0",146,"-1",18,"7",146, 0

ask$    jsr getkey
    bcs ask$
    cmp #3
    bne notc$
    jsr CR
    sec
    bcs brk$
notc$   cmp #"0"
    bcc ask$    ; smaller than "0"
    cmp #":"
    bcs ask$    ; ":" or greater
    sec
    sbc #"0"
    cmp #8
    bcs ook$
    ;clc
    adc #10 ; 0..7 becomes 10..17, 8..9 stays 8..9
ook$    sta drive$+1

    jsr putval
```

```
    jsr CR

    lda #0
    jsr SETNAM
chan$   lda #15
drive$  ldx #8
    ldy #15     ; open chan,drive,15
    jsr SETLFS
    jsr OPEN

    lda chan$+1
    jsr CLOSE
    lda ST      ; READST
    bne nok$
    jsr OPEN    ; reopen
    clc
    lda drive$+1
    rts
nok$    clc
brk$    lda #0
    rts


OK_KEY  jsr CHROUT
CR  lda #13
    jmp CHROUT


d64mode dc.b 0      ; 0 = file, 4 = D64, 7 = D71, 8 = D81, $84 = one drive
d64chan dc.b "#"

inDrive dc.b 8
#if !(MACH == 64 || MACH == 128)
be  dc.b 0  ; buffer index
#endif



cplens_3    ; Copy lengths for literal codes 257..285 - 3
    dc.b   3-3,   4-3,   5-3,   6-3,   7-3,   8-3,   9-3,  10-3
    dc.b  11-3,  13-3,  15-3,  17-3,  19-3,  23-3,  27-3,  31-3
    dc.b  35-3,  43-3,  51-3,  59-3,  67-3,  83-3,  99-3, 115-3
    dc.b 131-3, 163-3, 195-3, 227-3, 258-3  ;,   0,   0

cpdistLo    ; Copy offsets for distance codes 0..29
    dc.b $01, $02, $03, $04, $05, $07, $09, $0d
    dc.b $11, $19, $21, $31, $41, $61, $81, $c1
    dc.b $01, $81, $01, $01, $01, $01, $01, $01
    dc.b $01, $01, $01, $01, $01, $01, $ff

cpdistHi    ; Copy offsets for distance codes 0..29
    dc.b $00, $00, $00, $00, $00, $00, $00, $00
    dc.b $00, $00, $00, $00, $00, $00, $00, $00
    dc.b $01, $01, $02, $03, $04, $06, $08, $0c
    dc.b $10, $18, $20, $30, $40, $60, $ff

cplext  ; Extra bits for literal codes 257..285
    dc.b 0, 0, 0, 0, 0, 0, 0, 0
    dc.b 1, 1, 1, 1, 2, 2, 2, 2
```

```
    dc.b 3, 3, 3, 3, 4, 4, 4, 4
    dc.b 5, 5, 5, 5, 0  ;, 99, 99

cpdext  ; Extra bits for distance codes
    dc.b  0,  0,  0,  0,  1,  1,  2,  2
    dc.b  3,  3,  4,  4,  5,  5,  6,  6
    dc.b  7,  7,  8,  8,  9,  9, 10, 10
    dc.b 11, 11, 12, 12, 13, 13


;=============================================================================
; Start of work space - variables

tentimes:
    dc.b 0,10,20,30,40,50,60,70,80,90



#if MACH == 64 || MACH == 128
BREAD   sty y$+1
    ldx #$c0    ; 1581 BURST: LEXS000N  L=logical t&s
    lda fdos
    cmp #"D"
    beq 2$
    ; Other than 1581-format disks
    ldx #$40    ; 1571 BURST: TEBS000N  T=no transfer
2$  stx cmdline+2
    ;stx $0400

    LDA inTrack
    STA cmdline+3
    LDA inSector
    STA cmdline+4
    LDA #1
    STA cmdline+5

    LDA #$06
    STA cmdlen
    ldx #14
    JSR sendcmd

    SEI
    BIT D1ICR
    LDA D2PRA
    EOR #$10
    STA D2PRA

1$  LDA #8
wait0$  BIT D1ICR
    BEQ wait0$
    LDA D1SDR
    STA status

    and #15
    cmp #11     ; disk changed -> no data
    beq 4$

    LDA D2PRA
    EOR #$10
```

```
    STA D2PRA

    ldy #0
3$  LDA #8
wait1$  BIT D1ICR
    BEQ wait1$

    LDA D2PRA
    EOR #$10
    STA D2PRA

    LDA D1SDR
    sta inBuf,y
#if MACH == 64 || MACH == 128
    sta $d020
#endif
#if MACH == 20
    and #7
    eor $900f
    sta $900f
#endif
    INY
    BNE 3$

4$  CLI
y$  ldy #0
    RTS


    ; the channel number must be in X
BINQ    LDA #$04
    STA cmdline+2
    LDA #3
    STA cmdlen
    JSR sendcmd

    SEI
    BIT D1ICR

    LDA D2PRA
    EOR #$10
    STA D2PRA

    LDA #8
wait1$  BIT D1ICR
    BEQ wait1$

    LDA D1SDR
    STA status
        CLI
    RTS


cmdline
    dc.b "U0", $80, 1, 1, 1, 1, 1, 1
cmdlen
    dc.b 0
oldclk
    dc.b 0
```

```
status
    dc.b 0

sendcmd
    JSR CHKOUT
    LDX #0
    LDY cmdlen
1$  LDA cmdline,x
    JSR CHROUT
    INX
    DEY
    BNE 1$
    jmp CLRCHN


BERR    pha
    txa
    pha
    lda #"E"
    jsr CHROUT
    lda status
    jsr puthex
    lda #","
    jsr CHROUT

    lda status
    and #15
    tay
    lda ePtrLo,y
    sta str$+1
    lda ePtrHi,y
    sta str$+2
    ldy #0
str$    lda $aaaa,y
    beq end$
    jsr CHROUT
    iny
    bne str$
end$
    lda #","
    jsr CHROUT
#if 1
    pla
    sta tmp
    ldx #<tmp
    ldy #>tmp
    lda #1
    jsr putdec
#else
    lda #"T"
    jsr CHROUT
    pla
    jsr puthex
#endif
    lda #","
    jsr CHROUT
#if 1
    pla
    sta tmp
```

```
    ldx #<tmp
    ldy #>tmp
    lda #1
    jsr putdec
#else
    lda #"S"
    jsr CHROUT
    pla
    jsr puthex
#endif
    lda #13
    jmp CHROUT

e0  dc.b "OK",0
e1  dc.b "SELECTED PARTITION",0
e2  dc.b "NO SECTOR",0
e3  dc.b "NO SYNC",0
e4  dc.b "NO DATA BLOCK",0
e5  dc.b "DATA CRC ERROR",0
e6  dc.b "FORMAT ERROR",0
e7  dc.b "VERIFY ERROR",0
e8  dc.b "WRITE PROTECT",0
e9  dc.b "HEADER CRC ERROR",0
e10 dc.b "DATA EXTENDS INTO NEXT BLOCK",0
e11 dc.b "DISK CHANGED",0
e12 dc.b "DISK FORMAT NOT LOGICAL",0
e13 dc.b "DISK CONTROLLER IC ERROR",0
e14 dc.b "SYNTAX ERROR",0
e15 dc.b "NO DISK",0

ePtrLo  dc.b <e0,<e1,<e2,<e3,<e4,<e5,<e6,<e7,<e8,<e9,<e10,<e11,<e12,<e13,<e14,<e15
ePtrHi  dc.b >e0,>e1,>e2,>e3,>e4,>e5,>e6,>e7,>e8,>e9,>e10,>e11,>e12,>e13,>e14,>e15
#endif



lhdr    dc.b $50,$4b,$03,$04, $14,$00,$08,$00, $08,$00
    dc.b $9f,$05,$00,$00, $00,$00,$00,$00
    dc.b $00,$00,$00,$00, $00,$00,$00,$00
ghdr    dc.b $50,$4b,$01,$02, $14,$00,$14,$00
    dc.b $00,$00,$08,$00, $e0,$14,$69,$2d       ; 9.11.2002 02:39
ehdr    dc.b $50,$4b,$05,$06, $00,$00,$00,$00

inTrack     dc.b 0
inSector    dc.b 0
inMode      dc.b 0
#if MACH != 20
inBurst     dc.b 0
#endif
#if MACH == 64 || MACH == 128
IDE64present:
    dc.b 0
#endif

zipMode     dc.b 0  ; 0 == normal, $80 == stored
zipDrive    dc.b 8
zipFile     ds.b 30
zipLen      dc.b 0
zipComment  ds.b 36
```

```
zipCommentLen   dc.b 0

errorCode   dc.b 0
errorLen    dc.b 0
errorStr    ds.b 64

usize       dc.b 0,0,0,0
csize       dc.b 0,0,0,0
cstart      dc.b 0,0,0,0
outSize     dc.b 0,0,0,0
curlen      dc.b 0
last        dc.b 0

im      dc.b 0,0
left        dc.b 0,0
buc     dc.b 0
loops       dc.b 0
eof     dc.b 0
skip        dc.b 0
maxlen      dc.b 0
maxpos      dc.b 0,0

oMaxlen     dc.b 0
oMaxpos     dc.b 0,0
oChar       dc.b 0
nChar       dc.b 0
speed       dc.b 0

#if MACH == 64 || MACH == 128
CheckIDE64:
    ldy #2
    lda $DF09   ; check for Action Replay!
    cmp #$78    ; SEI
    beq 1$      ; could be AR, must not read $DExx!
0$  lda $DE60,y
    cmp IDEID,y
    bne 1$
    dey
    bpl 0$
    lda #<(inBuf+2)
    sta IDE64BR
    lda #>(inBuf+2)
    sta IDE64BR+1
1$  sty IDE64present    ; negative means IDE64 present
    rts

IDEID   dc.b "IDE"

CheckBurst:
    lda #0
    sta inBurst
    sta FASTSER

    lda inDrive
#if MACH == 64
    jsr myLISTEN
    lda #$6f
    jsr mySECOND
    lda #"U"
```

```
    jsr myCIOUT
    lda #"I"
    jsr myCIOUT
    lda #"+"
    jsr myCIOUT
    jsr myUNLSN
#else
    jsr LISTEN
    lda #$6f
    jsr SECOND
    lda #"U"
    jsr CIOUT
    lda #"I"
    jsr CIOUT
    lda #"+"
    jsr CIOUT
    jsr UNLSN
#endif
    bit FASTSER
    bvc nob$
    inc inBurst
    ;ldx #14        ; 1570/1571 seems to need this w/ C64+burst
    ;jsr BINQ
    ;lda status
    ;jsr puthex
    ;jsr CR
nob$    rts


#if MACH == 64
myLISTEN
    ora #$20
    pha
    bit $94         ; C3PO - buffered char flag
    bpl 0$
    sec
    ror $a3         ; set no-ATN flag
    jsr myPB        ; send buffered char
    lsr $94         ; clear buffered char flag
    lsr $a3         ; clear no-ATN flag
0$  pla
    sta $95         ; set buffered character
    sei
    jsr ee97        ; DOUT lo, release DATA
#if 0
    cmp #$3f
    bne 1$
    jsr ee85        ; CLKOUT lo, release CLK
#else
    lda D2PRA
    and #$08
    bne 1$
    jsr SPOUT       ; SPOUT
    lda #$ff
    sta D1SDR
    jsr WAITSPIN        ; waitSP+SPIN
    txa
    ldx #20
2$  dex
```

```
    bne 2$
    tax
#endif
1$  lda D2PRA
    ora #$08
    sta D2PRA       ; ATNOUT hi, ATN low

myDEL   sei
    lda D2PRA       ; CLKOUT hi, CLK low
    ora #$10
    sta D2PRA
    jsr ee97        ; DOUT lo, release DATA
    txa         ; DELAY937 delay for 937 cycles
    ldx #$b8
0$  dex
    bne 0$
    tax

myPB    sei
    jsr ee97        ; DOUT lo, release DATA
    jsr eea9        ; wait settle, DATA->C
    bcs edad        ; device not present
    bit D1ICR
    jsr ee85        ; CLKOUT lo, release CLK
    bit $a3         ; under ATN?
    bpl 3$          ;yes->wait for DATA up
    ; not under ATN, wait for DATA up+down+up
5$  jsr eea9        ; wait settle, DATA->C
    bcc 5$          ;wait for DATA released
4$  jsr eea9        ; wait settle, DATA->C
    bcs 4$          ;wait for DATA low
#if 0
3$  jsr eea9        ; wait settle, DATA->C
    bcc 3$          ;wait for DATA released
#else
3$  lda D2PRA
    cmp D2PRA
    bne 3$
    pha
    lda D1ICR
    and #$08
    beq 6$
    lda #$c0
    sta FASTSER
6$  pla
    bpl 3$
#endif

#if 0
    jmp $ee8e
#else
    lda D2PRA   ; CLKOUT hi, CLK low
    ora #$10
    sta D2PRA

    lda #$08
    sta $a5     ; 8 bits to send
ed66    lda D2PRA
    cmp D2PRA
```

```
    bne ed66
    asl
    bcc edaf+1  ;timeout read+write
    ror $95     ; next bit to send
    bcs ed7a
    jsr eea0    ; DOUT hi, DATA low
    bne ed7d
ed7a    jsr ee97    ; DOUT lo, release DATA
ed7d
    jsr ee85    ; CLKOUT lo, release CLK
    nop
    nop
    nop
    nop
    lda D2PRA
    and #$df
    ora #$10
    sta D2PRA   ; release DATA, CLK low
    dec $a5
    bne ed66    ;until 8 bits sent

    lda #$04
    sta $dc07   ; timer B low byte
    lda #$19
    sta $dc0f   ; force load, one-shot, run timer B
    lda D1ICR   ; clear old interrupt bits
ed9f    lda D1ICR
    and #$02    ; timer B expired?
    bne edaf+1  ;timeout read+write
    jsr eea9    ; wait settle, DATA->C
    bcs ed9f    ;wait for ACK
    cli
    rts
#endif


edad    lda #$80    ; device not present
edaf    bit $03a9   ; timeout read+write
    ora ST
    sta ST
    cli
    clc
    bcc ee03


mySECOND
    sta $95
    jsr myDEL   ; CLKLODELAY
    lda D2PRA
    and #$f7
    sta D2PRA   ; ATNOUT lo, ATN hi
    rts

myCIOUT
    bit $94
    bmi 0$
    sec
    ror $94
    bne 1$
```

```
0$  pha
    jsr myPB    ; put buffered char
    pla
1$  sta $95
    clc
    rts

myUNLSN
    lda #$3f
    ;pha
    ;lda FASTSER
    ;and #$7f
    ;sta FASTSER
    ;pla
    jsr myLISTEN+2
    jmp ee03

SPOUT
    ;lda #$7f
    ;sta D1ICR
    lda #$00
    sta $dc05
    lda #$04
    sta $dc04
    lda $dc0e
    and #$80
    ora #$55
    sta $dc0e
    bit D1ICR
    rts
WAITSPIN
0$  lda D1ICR
    and #$08
    beq 0$
SPIN
    lda $dc0e
    and #$80    ; preserve TOD mode, SP to input, timer A continuous
    ora #$01
    sta $dc0e

    lda #$25        ; Set the normal (PAL) frequence to TimerA
    sta $dc04       ; Change if you want to preserve NTSC-rate
    lda #$40
    sta $dc05

    rts


ee97    lda D2PRA   ; DOUT lo, release DATA
    and #$df
    sta D2PRA
    rts

eea0    lda D2PRA   ; DOUT hi, DATA low
    ora #$20
    sta D2PRA
    rts

eea9    lda D2PRA   ; wait settle, DATA->C
```

```
    cmp D2PRA
    bne eea9
    asl
    rts

ee03    lda D2PRA
    and #$f7
    sta D2PRA   ; ATNOUT lo, ATN hi
    txa
    ldx #$0a
0$  dex
    bne 0$
    tax
    jsr ee85    ; CLKOUT lo, release CLK
    jmp ee97    ; DOUT lo, release DATA

ee85    lda D2PRA   ; CLKOUT lo, release CLK
    and #$ef
    sta D2PRA
    rts

#endif
#endif

obufhi  dc.b 0
files   dc.b 0      ; number of files
filecnt dc.b 0

#if MACH == 128
UpdSearch:
    ldx #<FarUpdSearch
    lda #>FarUpdSearch
    jmp farcall

ClearSearch:
    ldx #<FarClearSearch
    lda #>FarClearSearch
    ;jmp farcall

farcall stx 4   ; low byte in X, high byte in A
    sta 3
    lda $02de
    pha
    lda $ff00
    sta $02de   ; =return memory config
    jsr JSRFAR
    pla
    sta $02de
    cli
    rts

bank1install:
    ; Install code to bank 1
    lda #bTmp
    sta $02b9   ; ZP for writebank
    lda #<bank1code
    sta bTmp+0
    lda #>bank1code
    sta bTmp+1
```

```
    ldy #0
0$  lda bank1org,y
    ldx #1      ; bank 1 (RAM1)
    jsr WRITEBANK
    iny
    cpy #bank1end-bank1org
    bne 0$

    lda #1      ; bank 1 (RAM1)
    sta 2
    sei
    php
    pla
    sta 5   ; SR
    cli
    rts

bank1org:
#rorg $0400
bank1code:

FarUpdSearch
    ; no check.. it doesn't matter if we read 1 byte off the end
updsearch$      ;if (bu > 1) {
    lda arg+0   ;  long off;
            ;  unsigned short index =
    asl     ;   ((*rPtr & (LSIZE-1)) << LSHIFT) |
    asl     ;   (*(rPtr+1) & (LSIZE-1));
    asl     ;  long *lPairPtr = &lPair[index];

#if LSHIFT == 4
    asl     ; shifting only leaves 4 bits
    sta zp0
    lda arg+1   ; HI
    and #15     ; take only 4 bits
    lsr     ; shift 2 bits to LO part, because
    ror zp0     ; array element is 4 bytes
    lsr
    ror zp0     ; leaves 2 bits in HI part
#endif

#if LSHIFT == 5
    sta zp0     ; shifting only leaves 5 bits
    lda arg+1
    and #31     ; take only 5 bits
    lsr     ; shift 1 bit to LO part
    ror zp0     ; leaves 3 bits in HI part
#endif
    ;clc        ; C already clear, because zp0 is *8 (or *16)
    adc #>lPair
    sta zp1
    lda pos+0   ;  unsigned short *tmp = &bSkip[p & (BLOCK_SIZE-1)];
    sta bTmp+0
    lda pos+1
    and #>(BLOCK_SIZE-1)
    asl bTmp+0
    rol
    ;clc        ; C already clear, because BLOCK_SIZE <= 32768
```

```
    adc #>bSkip
    sta bTmp+1

    ;       off = p - *lPairPtr;
    ;       if (off < (BLOCK_SIZE-MAX_MATCH)) {
    ;       *tmp = off;
    ;       } else {
    ;       *tmp = 0;
    ;       }
    ldy #0
    lda pos+0
    sec
    sbc (zp0),y
    sta off0$+1
    iny
    lda pos+1
    sbc (zp0),y
    sta off1$+1
    iny
    lda pos+2
    sbc (zp0),y
    sta off2$+1
    iny
    lda pos+3
    sbc (zp0),y
off2$   ora #0
    bne ptr0$   ;16 top bits != 0

    lda off1$+1
    cmp #>(BLOCK_SIZE-MAX_MATCH)
    bcc ptr$    ; HI byte <  --> ok!
    ; HI byte >=
    ;bne ptr0$  ; HI byte >  --> not ok
    ;todo: compare low byte -- but it doesn't matter much..

ptr0$   lda #0      ; too far or not found
    sta off0$+1
    sta off1$+1

    ; Update search structure pointers
ptr$    ldy #0
off0$   lda #0
    sta (bTmp),y    ;  *tmp=..
    iny
off1$   lda #0
    sta (bTmp),y
    dey     ;ldy #0
    lda pos+0
    sta (zp0),y ;  *lPairPtr = p;
    iny
    lda pos+1
    sta (zp0),y
    iny
    lda pos+2
    sta (zp0),y
    iny
    lda pos+3
    sta (zp0),y
    rts
```

```
FarClearSearch:
    lda #>lPair
    sta clr$+2
    ldx #>(lPairSize+bSkipSize)

    ldy #0
    tya     ;lda #0
clr$    sta $aa00,y
    iny
    bne clr$
    inc clr$+2
    dex
    bne clr$
    rts
#rend
bank1end
#endif  ; MACH==128

CA
;   ds.b MAXFILES* CASIZE
```