COMPUSER Volume 1 Nr 2 March 1988

A description of the DOS65 Disc format, Andrew Gregory

COMPUSER Volume 2 Nr 1 January 1989

Junior DOS65 Erik vd Broek

Directory sort routine DOS65, Andrew Gregory

A DESCRIPTION OF THE DOS65 DISC FORMAT.

INTRODUCTION.

The DOS65 operating has an efficient file manager which has a lot of nice features. In particular, it allows filenames of up to fourteen letters in length, it draws distinction between ASCII files and binary files and between command and data files, it allows directories of almost unlimited size and it re-uses the space occupied by deleted files so that discs never require compacting. These features require a disc format more complicated than is usual on an eight bit machine. This is the subject of this article.

DOS65 uses sectors of 256 bytes on single density (also called FM, Frequency modulated) or double density (MFM, Modified FM) discs. Single density discs have ten sectors per track on each side. Double density discs have sixteen or eighteen, the latter being known as `extra density' on DOS65 machines. The sectors are labelled from one upwards. The numbering covers both sides, so for example sector 17 on a double sided double density disc is actually sector 1 on the second side. Tracks are labelled from 0 to 39 or 79, depending on whether the disc has 40 or 80 tracks. An individual sector is referred to by it's track-sector address or `tsa'. The DOS65 sector read and write routines use what is termed `physical addressing' because the actual sector specified is read or written to. But the file read and write routines use what is termed `logical addressing'. The physical sector is obtained from the logical sector in a look-up table which has been constructed so that the number of physical sectors between adjacent logical sectors is a constant for that disc known as the skew. This speeds up the disc access time by giving DOS65 time to prepare itself to read a sector after reading the previous one without the disc having to make a complete revolution.

There are several types of sector on the disc. Each will be considered individually.

THE SYSTEM SECTOR.

The system sector resides at track 0 sector 1 side 0 on every disc. Amongst other things it contains the look-up table which relates logical and physical addressing and it tells DOS65 which sectors on the disc have been used. The names and functions of the various locations are listed below. You will find their addresses in your dvar.MAC file.

s.stab - (32 bytes). The look-up which relates logical and physical addressing. Each entry is greater then the previous one by the `skew' with which the disc is formatted. (Usually 2 or 3).

s.mode - (1 byte). This gives information about the contents of the look-up table. Bit 7 is `1' if there is no table, in which case logical and physical addressing are the same. This would probably be so for a silicon disc. Bit 6 is `1' if there is a side offset, meaning that the table only covers one side. In this event when finding a physical sector on the second side the number of sectors per side must be subtracted before the look-up and then added on afterwards. The remaining bits are unused.

s.mtrk - (1 byte). The number of tracks formatted. (40 or 80).

s.mcil - (1 byte). The number of sectors per track. (Total of both sides).

s.msec - (1 byte). The number of sectors per track per side. If this is half of the value contained in s.mcil the disc is double sided, if it is equal the disc is single sided.

s.bpat - (4 bytes). A bitmap pattern. Always $ff,$ff,$ff,$ff.

s.acnt - (1 byte). The allocation count, the number of sectors represented by each bit in the bit map table, s.bmap.

s.sht - (1 byte). The shift factor map per physical track. It contains the number of times a track number has to be multiplied by two to find the corresponding address in the bit map table, s.bmap.

s.tbas - (1 byte). Lowest track number on which tsl sectors can be written.

s.tbam - (1 byte). The tsl bit map of the first bitmap byte.

s.dbas - (1 byte). Lowest track number on which data sectors can be written.

s.boot - (2 bytes). Logical tsl sector of the boot file. (Found by I/O65 during booting).

s.dir - (14 bytes). Track sector addresses of sub-directories. They are zero for sub-directories which have not been created.

s.name - (24 bytes). Disc name.

s.cdat - (4 bytes). Creation date of the disc.

s.mdat - (4 bytes). Date of last modification of the disc.

s.bmap - (Rest of sector) . A `bitmap' of the disc usage. Each bit represents s.acnt sectors. If a bit is a `1' then the sectors which it represents are unallocated.

THE DIRECTORY SECTORS.

The first root (@) directory sector is found at track 0 sector 3. The track sector addresses of the sub-directories A to G are given at the location s.dir in the system sector. If they are zero the sub-directory does not exist. Each directory sector contains 15 entries. Starting at the first byte of the sector each has a 14 letter filename (with unused letters filled with zeros) followed by the logical track sector address (tsa) of the first track sector list sector of that file. Bytes 240 and 241 of the sector contain the logical track sector address of the next directory sector if there is one, otherwise both bytes are zero.
If bit 7 of the first byte of a filename is a '1' then the file has been deleted.

THE TRACK SECTOR LIST SECTORS.

Every file has at least one track sector list (tsl) sector associated with it. The tsl sectors list the logical track sector addresses of the data sectors in the correct order. In addition, the first tsl sector gives information about the file. The structure of the first sector is as follows:

ftsl - (2 bytes). logical tsa of next tsl sector. (zero if none).

btsl - (2 bytes). Backward link to previous logical tsl sector. Both zero for first tsl.

flmo - (1 byte). File mode. As shown by the `CAT' command.

```
Bit 7 - `1' if file can be read.
Bit 6 - `1' if file can be overwritten.
Bit 5 - `1' if file can be deleted.
Bit 4 - `1' if mode x (not in use yet).
Bit 3 - `1' if a command file.
Bit 2 - `1' if file executable.
Bit 1 and 0. File type: %00 Default.  ` '
                        %01 Ascii    `a'
                        %10 Binary   `b'
                        %11 Tokens   `c'
```

dblk - (1 byte). Directory sector number.

dpos - (1 byte). Position in sector. (0 to 15).

stad - (2 bytes). Load address of a file of default type. (BOOT for example). Note the load addresses of binary files are given in the data sectors.

rnad - (2 bytes). Run address of a file of default type. Note the run address of binary files is given in the data sectors.

flln - (2 bytes). Length of file.

vers - (1 byte). Version number of file. (Incremented each time file is updated).

credat - (3 bytes). Creation date of file.

moddat - (3 bytes). Date of last modification of file.

The remaining bytes ($20 onwards) are logical tsa's of data sectors. Subsequent tsl sectors contain:

ftsl - logical tsa of next tsl sector. Zero if none.

btsl - logical tsa of previous tsl sector.

All the following bytes contain the tsa's of data sectors.

DATA SECTORS.

Data sectors are entirely filled by data, they contain no information about the structure of the disc. When a file of type default, a or c is loaded into memory each of the sectors is loaded one after another in the order given by the TSL sectors.

A binary file can be loaded in two different ways. If it is loaded with the command 'load filename xxxx' where the load address xxxx is given it will load sector by sector in exactly the same way as the other file types. But data loaded in this way is not executable because mixed with the 6502 code are what I shall call 'information blocks' of five or eight bytes which give the load and execute addresses. There is always one at the beginning of a file preceding the data. A hexadecimal listing is:

02 rL rH 01 lL lH nL nH xx xx xx ...

Where xx xx xx ... are bytes of actual binary data. The 02 is a code which indicates that the following bytes are the low and high run address. It begins one of the information blocks in a 6502 code file. The 01 code occurs in every complete information block. Following it are four bytes. The first two are the load address of the next part of the file and the second two are the number of bytes to be loaded before the next information block is encountered. A file may be load at many different places in memory as a consequence of this scheme. The information block does not define the length of a file, so it is not uncommon for a file to end:

02 rL rH

When binary files are loaded with the load command without an address being specified or when they are loaded as commands the information blocks will cause the different parts of the file to be loaded at the correct places in memory.

JUNIOR - DOS65

By : Erik van den Broek, Holland.

It is realy rather easy to implement the DOS65 (as far as I know todays best keyboard-driven-65XX(X)-operating system), once you have build a JUNIOR equiped with a VDU-card. You wouldn't say so if you glance at the drawing below, but if you look better you will see that it is all rather straightforward in fact.

Features:
- switch-selection between: 1) bit-by-bit, pin-by-pin compatible (VDU)Junior
  2) simultanious Junior/DOS65 operation
  3) bit-by-bit, pin-by-pin compatible DOS65
- no change in PCB's
- all JUNIOR software remains the same (even cassette)
- easy debugging

Requirements for the Junior-with-Interface-and-VDU-card-owner:

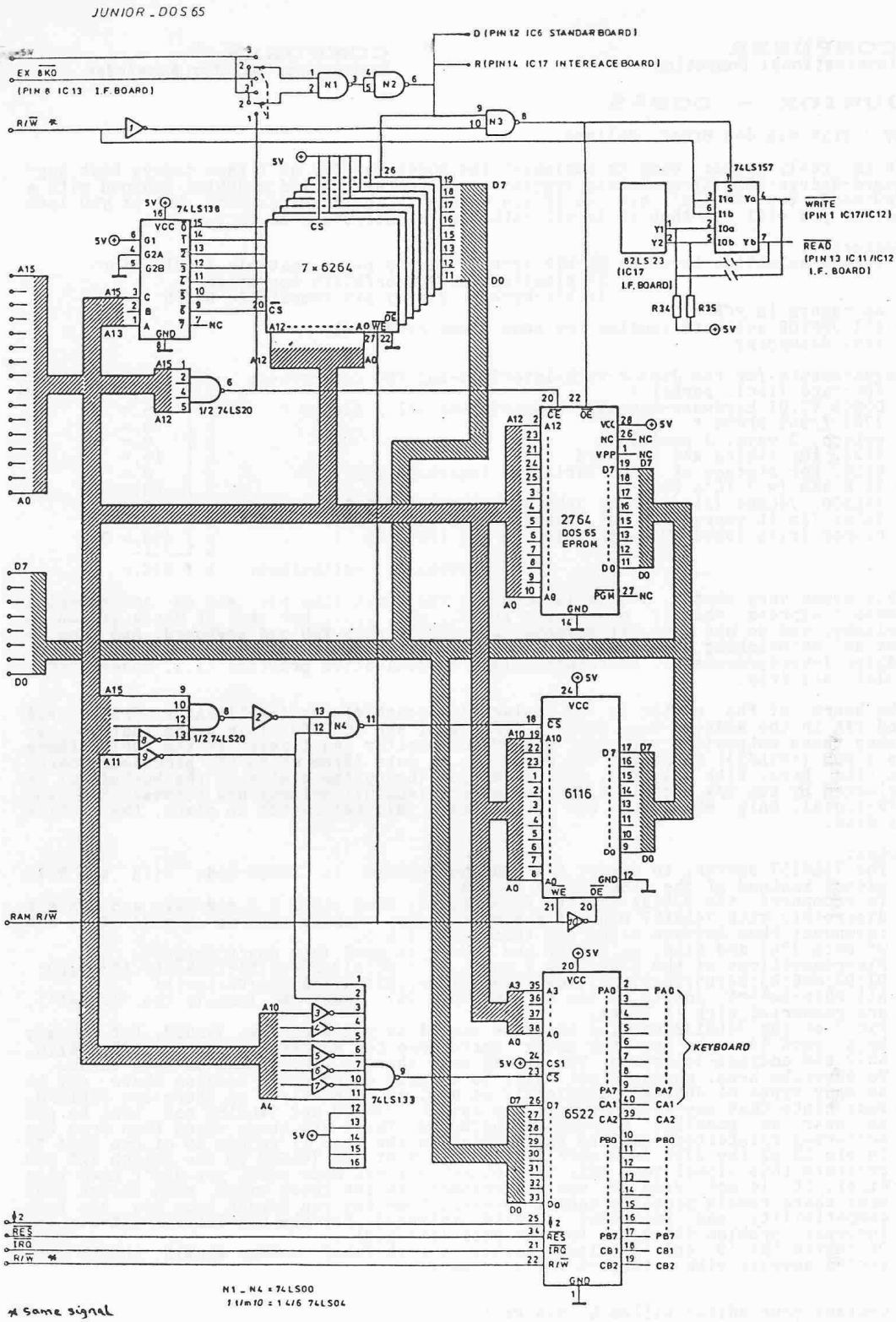
- FDC-card (incl. parts) *	± ƒ 120,=
- DOS65 V2.01 hardware-manual / operating-manual / floppy *	± ƒ 65,=
- 2764 I/O65 EPROM *	± ƒ 25,=
- switch, 2 ways, 3 positions	± ƒ 8,=
- 6522, for timing and keyboard	± ƒ 15,=
- 6116, for storage of DOS65 variables (=garbage-ram)	± ƒ 8,=
- 56 K RAM (= 7 IC's 6264)	± ƒ 60,=
- 74LS00; 74LS04 (2); 74LS20; 74LS133; 74LS138; 74LS157	± ƒ 8,=
- Extra (do it yourself)print, connector, sockets, etc.	± ƒ 75,=
- Floppy-drive (SHUGART-compatible (is <u>not</u> IBM-comp.))	± ƒ 450,= ?
pessimistic estimation:	± ƒ 834,=

This seems very costly, but it is probably the first time you see an amount which doesn't express what it only costs if, if, if ....., but what it costs if you're unlucky, and do not have yet anything but JUNIOR with VDU and keyboard. And you do get an astonishing good computer, because of all the software like full-screen-editor (=wordprocessor), macro-assembler, communication programs (f.i. modem-protocols), all free.

The heart of the matter is to replace (switchable) the (small)RAM's, EPROMs, VIA and PIA in the address-area $0000 - $1FFF by 1 RAM IC 6264. This means that you can debug these extensions with your old JUNIOR-monitor and tracer. In the JUNIOR there is a ROM (=82LS23) to select the direction of data (from or to the processor-board) in this area. With only RAM, this direction (being the status of the busbuffer) is selected by the R/W signal. A digital switch (=LS157) selects now between ROM- and R/W-signal. Only while you are putting this (digital)switch in place, the machine is dead.

Notes:
- The 74LS157 serves, to direct the databusbuffers in DOS65-mode with the R/W-signal instead of the promsignals Y1 and Y2.
- To reconnect the 82S23: remove from socket, bend pin 1 & 2 sideways and connect discretely with 74LS157 pin 2 & 5 respectively. Remove pull-up R's 34 & 35 and reconnect them between mains and those pins 2 & 5.
- Of both 2764 and 6116, only half the memory is used (you don't notice).
- Pin-connections of the 6264 (pin 1 until 28): NC;A12;A7;A6;A5;A4;A3;A2;A1;A0;D0; D1;D2;GND;D3;D4;D5;D6;D7;CS1(low);A10;OE(low);A11;A9;A8;CS2;WE(low);+
- All chip-select inputs of the 6264's (=pin 26), EXEPT ONE (namely the 'lowest'), are connected with +5 Volts.
- Pin 7 of the 74LS138 carries the same signal as pin 6 of the 74LS20, but it may be a good idea to use two extra cards: one for memory and one for EPROM 6116, 6522 and address-selection. The signal must then be represented on both cards.
- To describe here, where to put what, is a waste of effort, because there may be as many types of JUNIORS (physically as well as 'mentally') as there are JUNIORs. Some hints that may help however: the switch, 74LS00 and 74LS157 can best be put as near as possible to the motherboard. There are three wires then from the motherboard/interface-unit to the card(s) on the bus: 1) to pin 26 of one 6264 2) to pin 22 of the 2764 (=OE not) 3) from pin 6 of the 74LS20 to the switch (if you generate this signal very near the motherboard yet once more, you don't need this wire). It is not wise to use bus-connections for these wires, even though some pins there remain probably unused forever. Some day you forget your bus has lost compatibility and implement a card intended for the new ELEKTOR-65K-bus (no incompat. problem there but for two pins (16A/16C)).
- Of course the 'D' and 'R' signal-inputs on both (old) cards, should not be connected anymore with either +5 Volts or mass.

* Contact your editor Willem L. van Pelt.



Herman Zondag

```
        ttl "tp DOS65 Directory sort utility V0.10          %s              page %d"
;
        pag     66
;
;file           dirsort.MAC
;
;last modified  8th April 1988
;
;program        DIRSORT
;
;function       put a directory in alpabetical order
;
;usage          DIRSORT dir:
;
;by             Andrew Gregory
;               35 Stafford Road
;               Sidcup
;               Kent
;               DA14 6PU
;               England
;
;
;DOS65 routines and variables
        lib     dvar.mac
        opt     lis
;
;main workspace
;
secmem  equ     $20
;
;zero page workspace
;
        org     $80
ndirs   res     1               number of directories loaded
subd    res     1               sub-dir
ndrive  res     1
tmp     res     1               workspace
a0      res     2               workspace
a1      res     2               address pointers to filenames
a2      res     2
;
s2      res     2
s3      res     2
s4      res     2
s5      res     2
cbp     res     2
order   res     1
curdir  res     1
;
        org     $1000
        jmp     begin
;
;absolute workspace
fdirsec res     1               first sector
fdirtk  res     1
;
;give help reply
        fcc     $C8,$C5,$CC,$DO
help    jsr     print
        fcc     'Directory sort utility V0.10\r'
        fcc     'Syntax  : DIRSORT dir:\r'
        fcc     'Options : -Y Do not ask for permission\r'
        fcc     'Example : DIRSORT 0:A/\r'
        fcc     'Sorts subdirectory A on drive 0',0
;
        fcc     'By A.P.Gregory',0
;
4       jsr     print
        fcc     'Illegal -options\r',0
        jmp     13.f
;
begin   php                     save flags
        jsr     sopt
        fcc     'Y',0
```

```
            bcs     4.b             if error
            sty     cbp
            sta     cbp+1           command string address
            jsr     sync            printer spooler would
                                    -interfere by changing rwpoin
;
            lda     udrive          default is U:
            jsr     separ           set fdrive and subd
;
            ldy     #-1             skip spaces
3           iny
            lda     [cbp],y         get char from string
            beq     begin2          default if none
            cmp     #13             CR?
            beq     begin2
            cmp     #32             Space
            beq     3.b
;
;First character is a '0', '1', '2', 'U', 'S', or 'W'
            cmp     #'3
            bcc     3.f             number
;
;'S', 'U' or 'W'
            jsr     loupch          upper case
            ldx     #2              search table for it
1           cmp     drtab,x
            beq     2.f
            dex
            bpl     1.b
            bmi     12.f            illegal
2           lda     sdrive,x
            jsr     separ           separate drive and dir
            jsr     ckcolon
            bne     12.f
            jsr     cklast
            bne     12.f            if error
begin2      jmp     begin1          proceed
;
drtab       fcc     'SUW'
;
separ       pha
            and     #%11
            sta     fdrive
            sta     ndrive
            pla
            lsra                    get sub-dir
            lsra
            sta     subd
            rts
;
ckcolon     iny
            lda     [cbp],y
            cmp     #':             nothing else will do
            rts
;
cklast      iny
            lda     [cbp],y
            beq     1.f
            cmp     #13
            beq     1.f
            cmp     #32
1           rts
;
;illegal directory
12          jsr     print
            fcc     'Illegal drive/ directory specification\r',0
13          jsr     seter
exit        plp
            rts                 return to DOS65
;
;
;'0', '1' or '2'
3           sec
            sbc     #'0
            bmi     12.b            error
```

```
                sta     ndrive          drive number
                sta     fdrive          needed by dirinit
                jsr     ckcolon         check next is colon
                bne     12.b
                iny                     next char @, A..G
                ldx     #0
                stx     subd            @ is default sub-dir
                lda     [cbp],y
                beq     begin1          default is @
                cmp     #13             CR?
                beq     begin1
                cmp     #32             Space
                beq     begin1
                cmp     #'@
                beq     begin1
                jsr     loupch          Upper case Y saved
                cmp     #'H
                bcs     121.f           error
                sbc     #'A-2
                bmi     121.f           error
                sta     subd
                iny
                lda     [cbp],y
                cmp     #'/
                beq     1.f
                dey                     / not compulsory
        1       jsr     cklast          check no more
                beq     begin1
        121     jmp     12.b
        ;
        begin1  lda     opt             -Y option?
                bmi     1.f
                jsr     print
                fcc     'Sort directory ',0
                lda     fdrive          print dr:dir/
                clc
                adc     #'0
                jsr     out
                lda     #':
                jsr     out
                lda     subd
                bne     2.f
                lda     #'@
                bne     3.f
        2       clc
                adc     #'A-1
        3       jsr     out
                jsr     print
                fcc     '/   (Y/N*)? ',0
                jsr     bufin
                jsr     loupch
                cmp     #'Y
                bne     exit1
        ;
        1       jsr     dirinit         open directory
                bcs     doserr
                lda     #0
                sta     ndirs
                lda     #secmem
                sta     curdir          next dir sec
                lda     subd            @ directory?
                beq     20.f            if yes
        ;
                asla                    multiply by 2
                clc
                adc     #s.dir-2        get directory pos
                tay
                lda     [rwpoin],y
                tax                     track
                iny
                lda     [rwpoin],y      sector
                beq     exit1           if non-existent
                tay
                jmp     21.f
```

```
;
exit1   jmp     exit
;
doserr  jsr     ermes           error
        jmp     exit
;
20      ldx     #dir0tk
        ldy     #dir0sc
;
21      stx     fdirtk          first directory trk
        sty     fdirsec         first directory sector
;
;directory loading loop
lloop   lda     curdir
        cmp     #$A1            max 32 sectors
        bcs     3.f
        jsr     ldsec           loads sec, sets rwpoin
        bcs     doserr
        inc     ndirs
        jsr     ndts            get X and Y of next sector
        beq     2.f             last if sector 0
        inc     curdir
        jmp     lloop
;
3       jsr     print
        fcc     '*** Directory too big\r',0
        jmp     exit
;
2       lda     ndirs
        cmp     #3
        bcc     1.f
        jsr     print
        fcc     'Please wait...\r',0
1       jsr     zero            Obliterate all invalid names
        jsr     sort
        lda     #$40
        jsr     clstcon         disable ^C
;
;save sorted  directory
;Note - surplus directory sectors are not freed.
;DOS65 does not provide a mechanism for doing this.
sloop   lda     #secmem
        sta     curdir
        ldx     fdirtk
        ldy     fdirsec
sloop1  lda     curdir
        jsr     svsec           save sector
        bcs     doserr1
        inc     curdir          next memory
        jsr     ndts            set track-sector
        bne     sloop1          unless end
        jmp     exit
;
doserr1 jmp     doserr
;
;simple shell sort routine.
;Compare each pair and interchange if wrong way round.
;repeat until in order.
sort    lda     ndirs
        jsr     mul15           s2 is number of filenames
        lda     s2
        ora     s2
        bne     sort1
        rts                     return if empty dir
;
sort1   lda     #0
        sta     order           in order flag
        jsr     sort5
        lda     order           if further sorting needed
        bne     sort1
        rts
;
sort5   ldx     s2              set up s4 to last number+1
        stx     s4
```

```
            ldx     s2+1
            stx     s4+1
            jmp     sort2
;
sort3       lda     s4+1            address of s4 into a1
            ldx     s4
            ldy     #0
            jsr     entad1
            lda     s5+1            address of s5 into a2
            ldx     s5
            ldy     #2
            jsr     entad1
;
            jsr     coma2a1         compare them
            bcc     sort2           branch if in order
            lda     #$ff            set flag
            sta     order
            jsr     swpa2a1         swap them
;
sort2       dec     s4              higher in list
            bne     2.f
            lda     s4+1
            beq     1.f             if reached top
            dec     s4+1
2           lda     s4
            sec
            sbc     #1              s5 one above s4
            sta     s5
            lda     s4+1
            sbc     #0
            sta     s5+1
            jmp     sort3
1           rts
;
;routine to compare names at [a2] and
;[a1] and set carry flag if interchange
;is needed.
coma2a1     ldy     #0
            lda     [a1],y          lower entry zero?
            beq     3.f             do not swap if so
            lda     [a2],y          one above [a1]
            beq     4.f             swap if zero
5           pha
            lda     [a1],y          ignore bit 7
            and     #$7f
            sta     tmp
            pla
            cmp     tmp
            bcc     2.f             if lower entry higher
            bne     4.f             if not same
            iny
            lda     [a2],y
            and     #$7f
            bne     5.b             no swap if zero
3           clc
2           rts
;
4           sec
            rts
;
;routine to interchange names [a2] and [a1]
swpa2a1     ldy     #15
1           lda     [a2],y
            pha
            lda     [a1],y
            sta     [a2],y
            pla
            sta     [a1],y
            dey
            bpl     1.b
            rts
;
;fill all invalid filenames with zeros
zero        jsr     s2s3set
```

```
            jmp     zloop1
;
zloop       jsr     entaddr
            ldy     #0
            lda     [a1],y
            bmi     1.f             if deleted
            bne     zloop2          if valid
1           tya
            ldy     #15             fill with 0's
2           sta     [a1],y
            dey
            bpl     2.b
zloop2      inc     s3
            bne     zloop1
            inc     s3+1
zloop1      dec     s2
            bne     zloop
            dec     s2+1
            bne     zloop
            rts
;
;prepare for scan through directory memory.
s2s3set lda     #0
            sta     s3              zero s3
            sta     s3+1
s2set       lda     ndirs
            jsr     mul15           max entries in s2
            inc     s2
            inc     s2+1
            rts
;
;routine to multiply A by 15 (A<50)
;answer in s2
mul15       sta     s2
            ldx     #0
            stx     s2+1
            asla                    by 2
            asla                    by 4
            clc
            adc     s2              by 5
            sta     s2
            asl     s2              by 10
            rol     s2+1
            clc
            adc     s2              by 15
            sta     s2
            lda     s2+1
            adc     #0
            sta     s2+1
            rts
;
;routine to convert entry number s3 to
;an address and store in a1
;divide by 15. Answer is page, remainder*16 pos.
entaddr lda     s3+1
            ldx     s3
            ldy     #0
;
;entry point to put answer in a1+y (y even)
entad1      stx     a0              store low part
            sta     a0+1            and high part
            tya
            pha
            ldx     #16             16 bits
            lda     #0
            tay
1           asl     a0
            rol     a0+1
            rola
            cmp     #15
            bcc     2.f
            sbc     #15
2           pha
            tya
```

```
            rola
            tay
            pla
            dex
            bne     1.b
            asla
            asla
            asla
            asla
            sta     a0              offset in page
            tya                     page number
            clc
            adc     #secmem         add start address
            sta     a0+1
            pla
            tax
            lda     a0
            sta     a1,x
            lda     a0+1
            sta     a1+1,x
            rts
;
;Make X=track and Y=sector of next directory.
;set Z if none
;Note rwpoin must be set up.
ndts        ldy     #240
            lda     [rwpoin],y      track of next dir
            tax
            iny
            lda     [rwpoin],y
            tay                     sector of next dir
            rts
;
;subroutines to load and save logical sector Y track X at page A
;system sector must be loaded.
;save
ldsec       jsr     rwset
            jsr     readsect        reads logical sector
            rts                     C=1 if error
;save
svsec       jsr     rwset
            jsr     writsect
            rts
;;
rwset       sta     rwpoin+1        set page address and
            lda     #0              prepare for read/writesect
            sta     rwpoin
            lda     ndrive          drive number
            rts
```