S100computers.com IDE Interface Board
Disk Layout Considerations for CP/M 3.0 System Build

This document is aimed at new S-100 computer users to give a foundational understanding of the CP/M file system & boot process and to explain some program code changes to the 'wrlba' section of the disk access routines in the CP/M system files and all subsequent programs that access the IDE CF drive.

One common problem for the new S100 enthusiast is that it often necessary to have access to a working CP/M system in order to build a CP/M operating system for a new hardware configuration (chicken & egg situation). In order to break into this cycle one solution that I originally investigated and implemented for myself was disk sector map editing, this method has recently been superceded by making a minor change to the IDE CF disk layout to make it compatible with the 'CPMTOOLS' toolset available for the PC/Linux platforms.

please refer to the subsequent work done by Dave (yoda) on the N8VEM wiki
http://n8vem-sbc.pbworks.com/w/browse/#view=ViewFolder&param=CPM3-Disk-Images

Whilst investigating the process of making a CP/M 3.0 disk image it became apparent that the current S100computers.com IDE disk layout implementation leaves holes (areas of unaccessed sectors) in the disk sector map on the destination media. This is due to the IDE disk being switched into LBA mode but raw CP/M track and sector values are passed to the drive without translation to a true LBA value. Whilst this is not really an issue from the point of wasted space as CP/M drives are very small (8MB) It does create a huge problem when trying to build a CP/M disk image by disk sector map editing (as it is difficult to calculate where to locate data in the disk sector map) or by using CPMTOOLS as the layout is not a valid format..

The solution was to make some changes to the 'wrlba' code sequences in the CP/M bios files to calculate the true LBA address (without holes) from the track and sector requests, then we can predictably locate data into the correct disk sector locations and ensure compatibility with the CPMTOOLS toolset.

To understand how this problem occurs it is helpful to have some background information on how IDE drives are addressed in LBA mode, for those who like to know the detail I would suggest a search on the internet as due to various historical changes in IDE drive technology and access modes & methods it is beyond the scope of this document to fully cover this topic. But in simplistic terms once our destination IDE drive is switched into LBA mode by sending the correct bit patterrn to b5 – b7 of the IDE disk head register, all sectors on the disk (upto 137GB) can be accessed via one long sequential 28 bit word. In reality we are only interested in the lower 16 bits, the upper 12 bits are all zeros in our application.

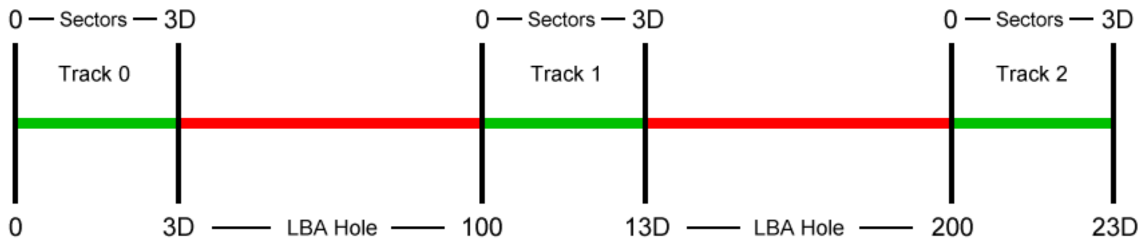
Careful examination of the original 'wrlba' routine code in the CPMLDR file HLDRBIOS.ASM shows that three bytes are sent to the drive to address the required disk sector

'High Trk'	(REGcylinderMSB)	= Always Zero
'Low Trk'	(REGcylinderLSB)	= CP/M requested track
'Sector'	(REGsector)	= CP/M requested sector

We also note that in the disk parameter block CP/M has been configured for 61 sectors per track

```
; IDE HARD DISK PARAMETER BLOCK:
HD$DPB:         DPB     512,61,256,2048,1024,1,8000H
```

So as CP/M accesses the disk medium and reaches sector 61 (3DH) the next available sector from CP/M's perspective is sector 0 on the next track. This means for our lowest 8 bits of the drive LBA sector address we have skipped over values 62 – 255 (3FH – FFH) creating a area of unaccessed sectors on the physical disk medium. This is represented in the diagram below.



The solution to fixing this problem is to convert the CP/M track and sector addresses into a true LBA address, this will close up the holes in the disk sector map and leave us with a disk format that can be manually edited in a predictable manner as well as being recognised by the cpmtools toolset.

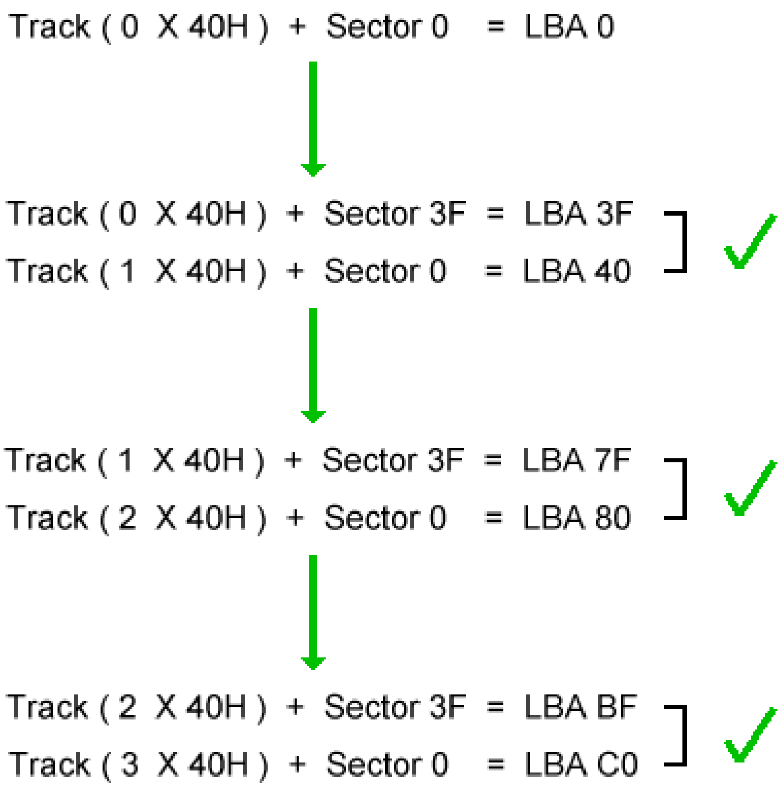
One decision that was made upfront was to fix the number of sectors per track at 64, being a binary power of two it keeps the program code tight and efficient, see below for code details

```
wrlba:
        LHLD    @TRK        ;Get CPM requested Track Hi&Lo
        MVI     H,00H       ;zero high track byte
        MOV     A,L         ;load low track byte to accumulator
        CPI     00H         ;check for 0 track and skip track loop
        JZ      lbasec
        MVI     B,06H       ;load counter to shift low track value 6 places to left i.e X 64
lbatrk:
        DAD     H           ;Add HL to itself 6 times to multiply by 64
        DJNZ    lbatrk      ;loop around 6 times i.e x 64

lbasec:
        LDA     @SECT       ;Get CPM requested sector
        ADD     L           ;Add value in L to sector info in A
        JNC     lbatot      ;If no carry jump to lba 'total' calculation
        INR     H           ;carry one over to H
lbatot:
        MOV     L,A         ;copy accumulator to L
                            ;HL should now contain correct LBA value
```

Wrlba Code Description

The CP/M requested track (0 – 255) is loaded into the HL register pair, then the H register is set to zero as this will be our overflow register during the calculation. The track value in the L register is then tested to see if it is zero, if it is then the multiplication routine is skipped (jump to lbasec), if not then the B register is set to the value 6 indicating a loop of 6 iterations. Knowing that our HL register pair contains a value of at least 1 the HL register pair is now added to itself six times, each time doubling the contents, in effect moving its contents six bits to the left (possibly moving some bits into H).

Next the CP/M requested sector value (between 0 and 63) is loaded into the accumulator and the value that we now have in the L register is added to the accumulator, we then look at the carry flag to see if the addition has caused the accumulator to overflow. If the carry has not been set (no overflow) then we know that we do not have to compensate the H register by incrementing it. Finally we copy the value of the accumulator back to the L register. HL now contains the true LBA sector address value for the given CP/M requested track and sector request. The diagram below shows what happens during track transitions.


Track (0 X 40H) + Sector 0 = LBA 0
Track (0 X 40H) + Sector 3F = LBA 3F
Track (1 X 40H) + Sector 0 = LBA 40
Track (1 X 40H) + Sector 3F = LBA 7F
Track (2 X 40H) + Sector 0 = LBA 80
Track (2 X 40H) + Sector 3F = LBA BF
Track (3 X 40H) + Sector 0 = LBA C0

Finally our LBA sector address is passed to the IDE CF drive by the following code sequence which essentially hasn't changed, just the order of values for reasons of code clarity.

```
MVI     D,0                     ;Send 0 for upper cyl value
MVI     E,REGcylinderMSB
CALL    IDEwr8D                 ;Send info to drive

MOV     D,H                     ;load lba high byte to D from H
MVI     E,REGcylinderLSB
CALL    IDEwr8D                 ;Send info to drive

MOV     D,L                     ;load lba low byte to D from L
MVI     E,REGsector
CALL    IDEwr8D                 ;Send info to drive

MVI     D,1                     ;For now, one sector at a time
MVI     E,REGseccnt
CALL    IDEwr8D

RET
```

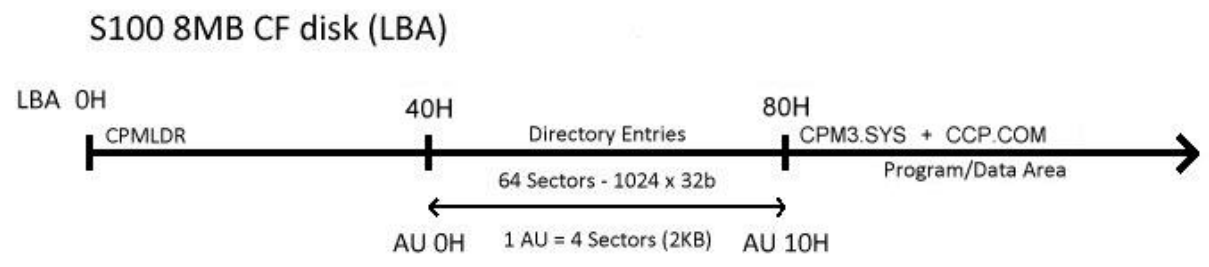
CP/M System Files and Disk Layout

In order for our CP/M system to boot up and sign on with a command prompt there are three files that need to be loaded into memory in sequence each building upon the other. I am not going to go into any detail on the content of these files but instead will point you to the excellent write up done by John Monahan on the S100computers.com website. What I aim to convey is a brief summary of the layout of these files on the CF disk media in order that our CP/M system will successfully boot to a command prompt.

http://www.s100computers.com/Software%20Folder/CPM3%20BIOS%20Installation/CPM3%20HDISK%20BIOS%20Software.htm

The first of the three files we need to load into memory is CPMLDR.COM, this file is hardware configuration specific and performs the sole function of locating the system file CPM3.SYS (also hardware configuration specific) on the disk medium and loading it into memory before transferring control to it. The CPMLDR.COM program itself needs to be located on the system track of the CF disk (track 0) beginning at sector 1, this location is reserved specifically for this file and this is where the ROM/Monitor bootstrap code will look for it as the file directory on track 1 is not read at this point in the boot process.

Once the first 12 sectors (1 – 12) are loaded into memory, control is passed to CPMLDR.COM and it reads the file directory on track 1 of the CF disk to locate CPM3.SYS in the program/data area of the CF disk (track 2 onwards). Once loaded into memory CPM3.SYS will then read the directory on track 1 to load in the final program module CCP.COM to give us a command prompt on the console. This is represented in the diagram below.



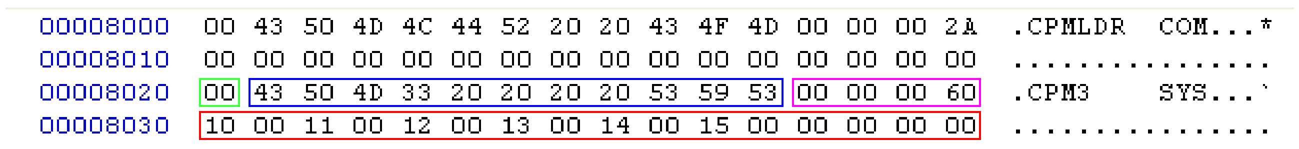
So looking at the diagram we see three distinct areas that influence the boot process.

Track 0

Consists of 64 sectors numbered 0 to 63 (0H – 3fH) each containing 512 bytes formatted that each byte is E5H. Onto this track starting at sector 1 the CPMLDR.COM file is loaded, there may be a directory entry associated with this file but it is not read during the boot process and only serves to allow the CPMLDR file to show up on a DIR directory listing.

Track 1

Consists of 64 sectors numbered 64 to 127 (40H – 7FH) each containing 512 bytes giving us a total directory track size of 32Kb permitting 1024 directory entries of 32 bytes each in length. A typical directory entry is shown below.



```
00008000   00 43 50 4D 4C 44 52 20 20 43 4F 4D 00 00 00 2A   .CPMLDR  COM...*
00008010   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00008020   00 43 50 4D 33 20 20 20 20 53 59 53 00 00 00 60   .CPM3    SYS...`
00008030   10 00 11 00 12 00 13 00 14 00 15 00 00 00 00 00   ................
```

We can see from the above diagram that the 16 byte directory entry is divided into 4 main areas. The first area outlined in green is the CP/M 'user' number, this would normally be 0 but could be another value between 1 – 15 if the file is assigned to a user number other than 0. A value of E5 in this field indicates the file has been deleted.

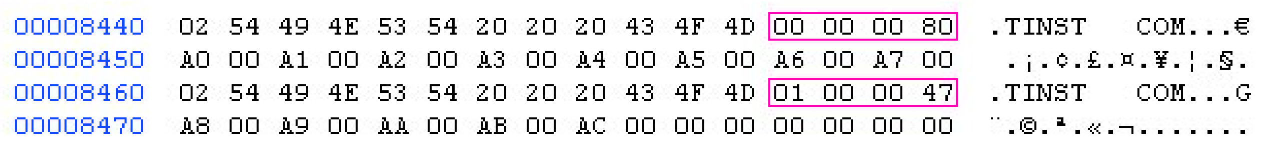
In the second area outlined in blue, 11 bytes are assigned to the filename in 8 + 3 format.

In the third area outlined in mauve are four one byte values, two called extents which essentially tie together multiple directory entries, and the other two values relate to the number of records (128 bytes blocks) in this extent and the last record byte counter which stores how many bytes of program data are stored in the last record.

Finally, the forth area outlined in red, is a block of 16 bytes assigned to recording the disk 'allocation units' (groups of 4 sectors) where the file data is stored on the disk medium. The 16 bytes are grouped as 8 x 16 bit words where the left hand byte of the pair represents the LSB and the right hand byte the MSB.

Track 2 onwards

Consists of sectors numbered 128 onwards (16384 on 8Mb disk) where CP/M stores program data. Where a file is larger than a 2Kb Allocation Unit, multiple AU's will be allocated to the file and multiple AU index entries will be recorded in the directory record. Where the file is sufficiently large to fill all the directory record space with AU index entries a second directory entry will be created for the file to allow further AU index entries to be recorded, these two directory will be tied together using 'extents' to make one logical directory entry.



```
00008440  02 54 49 4E 53 54 20 20 20 43 4F 4D 00 00 00 80  .TINST   COM...€
00008450  A0 00 A1 00 A2 00 A3 00 A4 00 A5 00 A6 00 A7 00   .¡.¢.£.¤.¥.¦.§.
00008460  02 54 49 4E 53 54 20 20 20 43 4F 4D 01 00 00 47  .TINST   COM...G
00008470  A8 00 A9 00 AA 00 AB 00 AC 00 00 00 00 00 00 00  ¨.©.ª.«.¬.......
```

In the diagram above I have highlighted two areas in mauve, the 1st and 2nd digits in each field are the extent numbers LSB followed by MSB these signify the order of the directory entries. The 3rd digit is reserved and set to 0, finally the 4th (right most) digit stores the number of 128 byte records in this extent where values can range from 01H to 80H.

Conclusion

What is presented here is a simplified overview of the disk layout and filing system used by CP/M 3.0 on the S100computers.com IDE CF disk card. I hope this document has been helpfull in providing a foundational understanding of what is going on at the CF disk media level as we boot and access files on our S100 CP/M systems, for a more detailed description of these processes please refer to the Digital Research CP/M 3 Documentation.

==

Important Notice

==

What follows below is a document detailing my early work in investigating how CP/M accesses files on a disk and how these findings shaped the re-write of the wrlba code routine. Please note that all sector values shown in the document below are one sector lower that the finalised code (above) due to some early assumptions on the LBA conversion formula. The document text colour has been changed to grey to indicate that this part of the document is for information only and non authoritative.

For all new CP/M disk image builds please refer to the subsequent work done by Dave (yoda) on the N8VEM wiki

http://n8vem-sbc.pbworks.com/w/browse/#view=ViewFolder&param=CPM3-Disk-Images

==

S100 Computers Z80 Master CPU / IDE CF
CP/M 3.0 System Build

This document is aimed at helping the newbie S100 computer enthusiast get his/her system from a pile of assembled PCB's to a simple NON-BANKED CP/M 3.0 command prompt.

One common problem for the new S100 enthusiast is that it often necessary to have access to a working CP/M system in order to build CP/M for a new system (chicken & egg situation), the method described here provides one way around this situation. Another difficulty is that the current S100Computer implementation of LBA leaves holes in the disk sector map on the destination media, whilst this is not an issue from the point of wasted space as CP/M drives are very small by modern standards It does create a huge problem when trying to build a CP/M disk by direct sector editing (the method used here) as it is difficult to calculate where to locate data in the disk sector map. The solution was to make some changes to the 'wrlba' code sequences in the CP/M bios files to calculate a linear LBA address (no holes) from the track and sector requests then we can predictably locate data into the correct sector locations.

From here on in it is assumed that you have already made the necessary code changes to the various bios files provided on the S100Computers.com website as detailed in my document Z80_LBA_Addressing.doc , alternatively you can use the archive file CPM3_LBA.ZIP where the necessary changes to the source files have been made for you.

Please note that no warranty is expressed or implied in the use of this method as you may see different responses to the actions taken in this process. You will need to work around these issues as/when/if they crop up.

You will need:-

A copy of HxD Hex Editor - http://mh-nexus.de/en/hxd/

A modified copy of S100 Computers MYIDE software running in ROM (mapped to E000H) to perform boot.
(See LBA_Addressing.doc for details as i have not yet modified my Z80 Monitor for linear LBA)

A copy of the archive file CPM3_LBA.ZIP

A Compact Flash card reader attached to a PC running Microsoft Windows

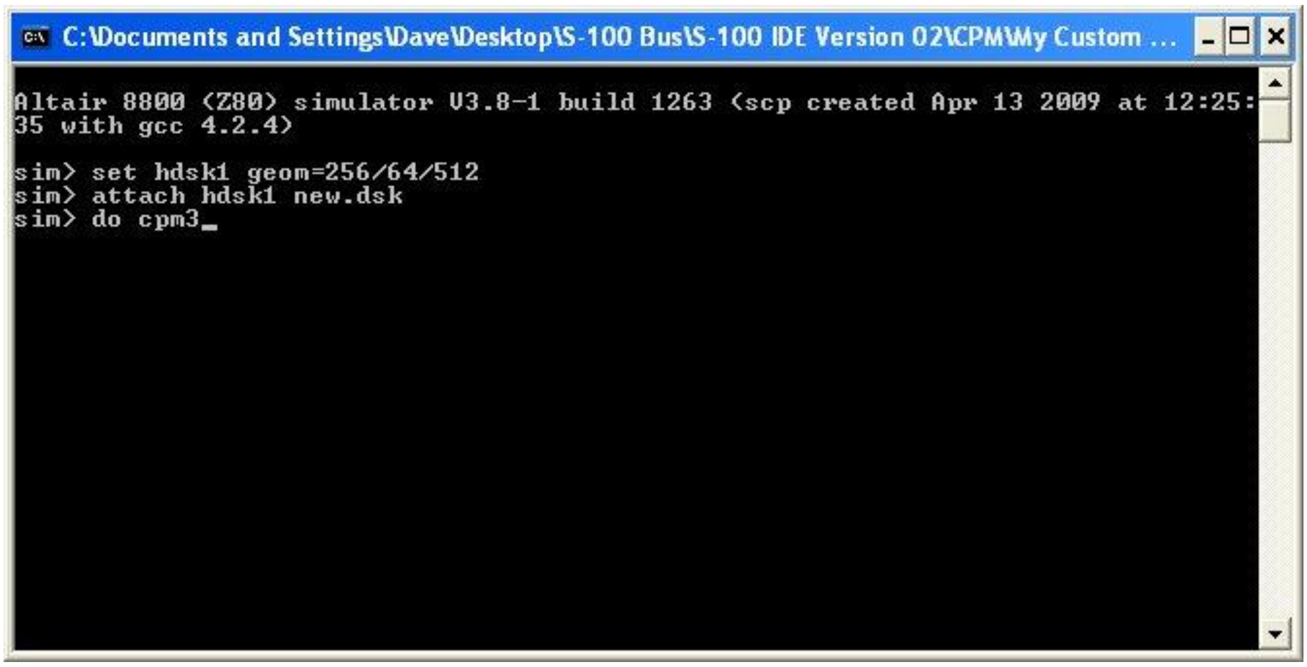
A S100 Computers Z80 based system with, IDE / CF interface board & Console I/O board

Before you begin:-

Read the S100 Computers website page 'BRINGING UP CPM3 FOR THE FIRST TIME ON A IDE HARD DISK BASED SYSTEM' read it several times, stew on it for a few days then read it again.

Now you can begin....

Assemble and write CPMLDR.COM to CF card


C:\Documents and Settings\Dave\Desktop\S-100 Bus\S-100 IDE Version 02\CPM\My Custom ...

```
Altair 8800 (Z80) simulator V3.8-1 build 1263 (scp created Apr 13 2009 at 12:25:
35 with gcc 4.2.4)

sim> set hdsk1 geom=256/64/512
sim> attach hdsk1 new.dsk
sim> do cpm3_
```

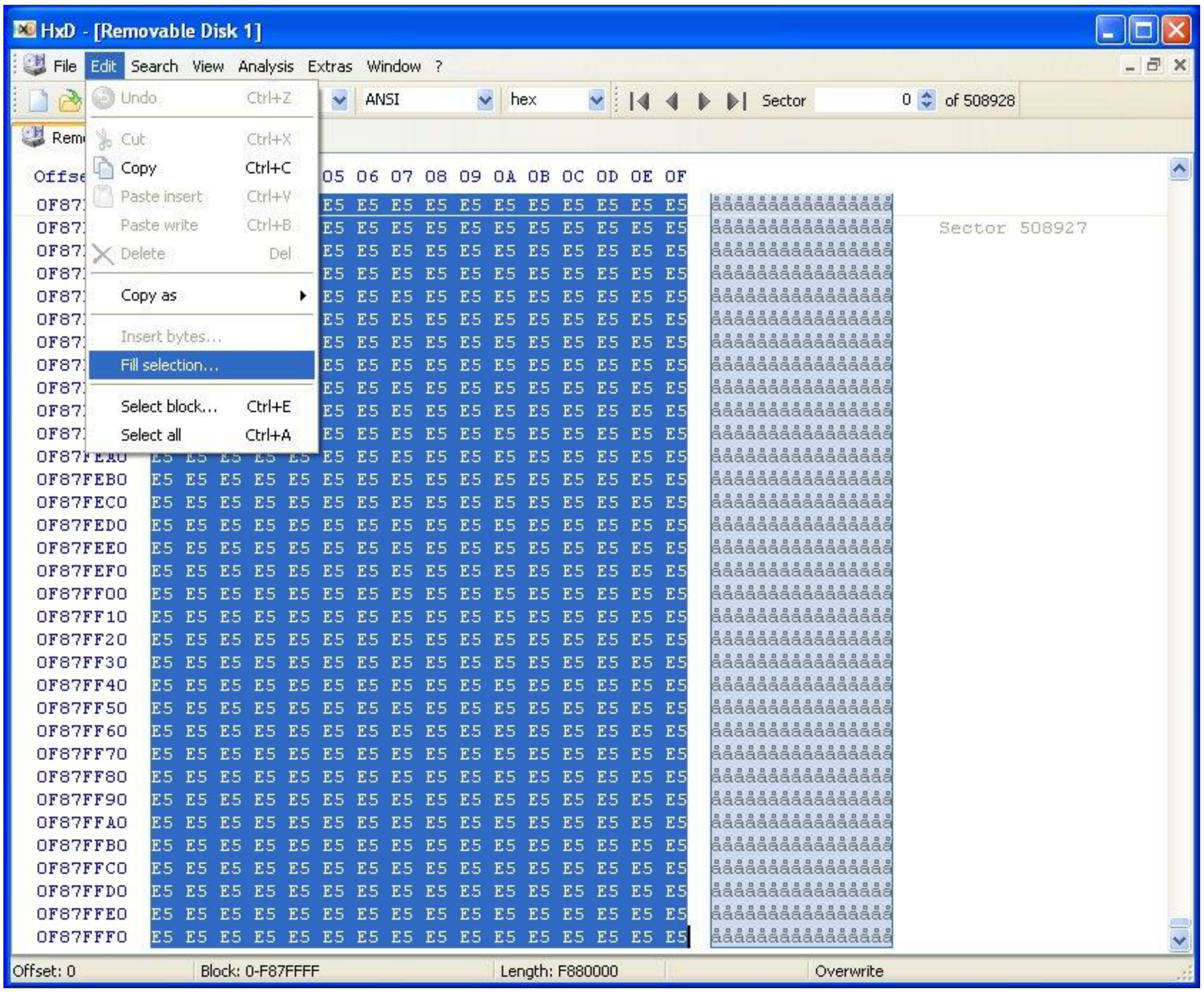
Extract the archive CPM3_LBA.ZIP to a folder on your PC hard drive
Open the folder CPMLDR, launch altairz80.exe and enter the following commands in the order shown.

```
set hdsk1 geom=256/64/512
attach hdsk1 new.dsk
do cpm3
```

At the CP/M a: prompt change to the 'i' drive and type the following commands allowing a little time for the submit file to run.

```
submit hsysgen.sub
pip j:cpmldr.com=i:cpmldr.com
```

The 'J' drive should now hold a copy of CPMLDR.COM



```
Offse                 05 06 07 08 09 0A 0B 0C 0D 0E 0F
0F87                  E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5  åååååååååååååååå
0F87                  E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5  åååååååååååååååå   Sector 508927
0F87                  E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5  åååååååååååååååå
0F87                  E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5  åååååååååååååååå
0F87                  E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5  åååååååååååååååå
0F87                  E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5  åååååååååååååååå
0F87                  E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5  åååååååååååååååå
0F87                  E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5  åååååååååååååååå
0F87                  E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5  åååååååååååååååå
0F87                  E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5  åååååååååååååååå
0F87                  E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5  åååååååååååååååå
0F87FEA0 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5  åååååååååååååååå
0F87FEB0 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5  åååååååååååååååå
0F87FEC0 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5  åååååååååååååååå
0F87FED0 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5  åååååååååååååååå
0F87FEE0 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5  åååååååååååååååå
0F87FEF0 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5  åååååååååååååååå
0F87FF00 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5  åååååååååååååååå
0F87FF10 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5  åååååååååååååååå
0F87FF20 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5  åååååååååååååååå
0F87FF30 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5  åååååååååååååååå
0F87FF40 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5  åååååååååååååååå
0F87FF50 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5  åååååååååååååååå
0F87FF60 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5  åååååååååååååååå
0F87FF70 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5  åååååååååååååååå
0F87FF80 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5  åååååååååååååååå
0F87FF90 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5  åååååååååååååååå
0F87FFA0 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5  åååååååååååååååå
0F87FFB0 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5  åååååååååååååååå
0F87FFC0 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5  åååååååååååååååå
0F87FFD0 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5  åååååååååååååååå
0F87FFE0 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5  åååååååååååååååå
0F87FFF0 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5 E5  åååååååååååååååå

Offset: 0        Block: 0-F87FFFF        Length: F880000        Overwrite
```

Launch HxD Hex Editor and open a compact flash card ensuring you disable read only mode

Format CF card

With the compact flash tab selected hit ctrl-A on your keyboard to highlight all bytes then select 'Fill Selection' from the edit menu, select the fill byte as E5 and click ok, this should change all the bytes selected to E5. Now save the changes by clicking the floppy disk icon on the toolbar.

Open our disk image

From the Extras menu select 'open disk image' , browse to our disk image 'new.dsk' in the CPMLDR folder and select open, accept the 512 byte sectors message.
This will open the disk image file in a second tab in the display, scroll through the sectors and notice how they appear to be labelled from 0 onwards at 512 byte intervals, notice also that the directory table is located at sector LBA 48 and the file cpmldr.com is located at sector LBA 112. These are both in the incorrect location for our purposes but we will correct that as we progress.

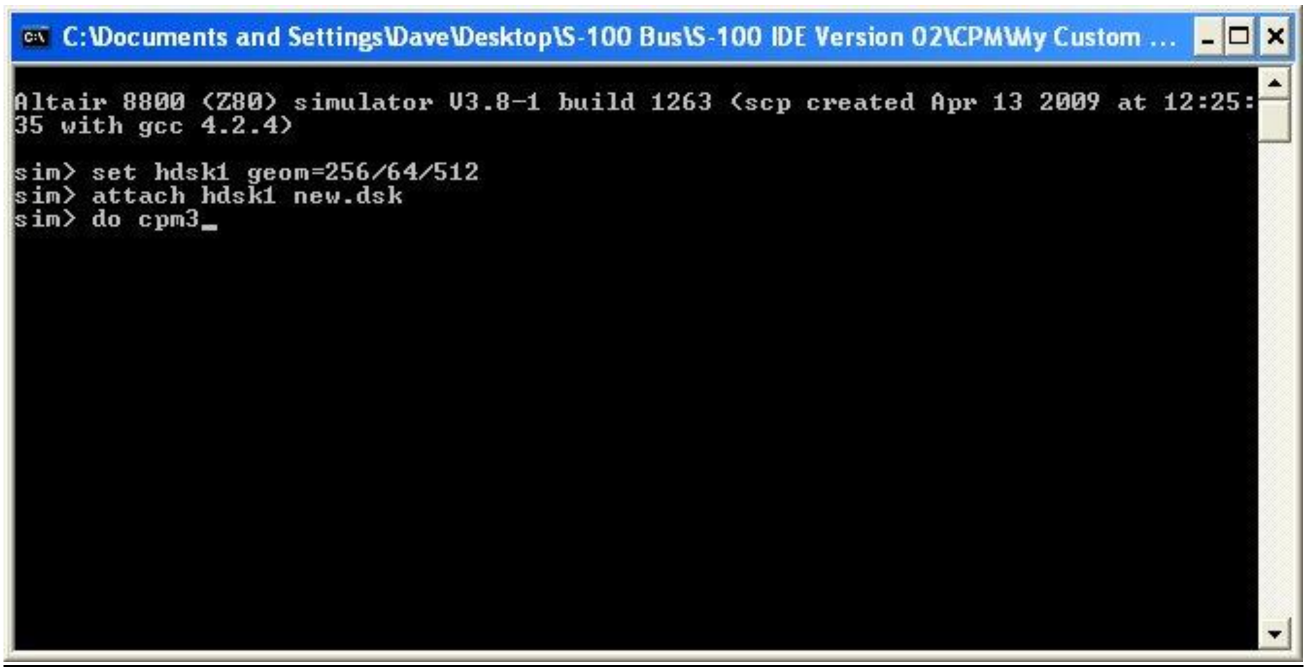
Writing CPMLDR.COM to system track

In a CHS disk setup the cpmldr.com file needs to be written to sector 1 on track 0 (0/0/1), this translates to LBA sector 0 on track 0 from what information I have gleaned over the internet and certainly this is the way I have coded the CHS to LBA translation.

Highlight the 12 sectors 112 to 123 in our disk image 'new.dsk' and select copy from the edit menu.
Select the tab 'Removable disk 1' and centre the cursor on the first byte of sector 0, go to the edit menu and select 'paste write', the data should now be visible starting at sector 0, hit the floppy disk icon on the toolbar and save the changes to the CF card.

Clean up the disk image 'new.dsk'
It's easier to progress at each stage if you have a clean disk image so highlight the first 150 sectors or so of 'new.dsk' and fill them with E5's remembering to save the changes to the disk image. Now close the tab 'new.dsk' by right click – close.
Alternatively you can delete the current 'new.dsk' file and create a new copy of blank.dsk and rename it to 'new.dsk' so you are working with a blank disk image at each stage.

Assemble and write CPM3.SYS to CF card


C:\Documents and Settings\Dave\Desktop\S-100 Bus\S-100 IDE Version 02\CPM\My Custom ...

```
Altair 8800 (Z80) simulator V3.8-1 build 1263 (scp created Apr 13 2009 at 12:25:
35 with gcc 4.2.4)

sim> set hdsk1 geom=256/64/512
sim> attach hdsk1 new.dsk
sim> do cpm3_
```

Open the folder CPM3NB, launch altairz80.exe and enter the following commands in the order shown.

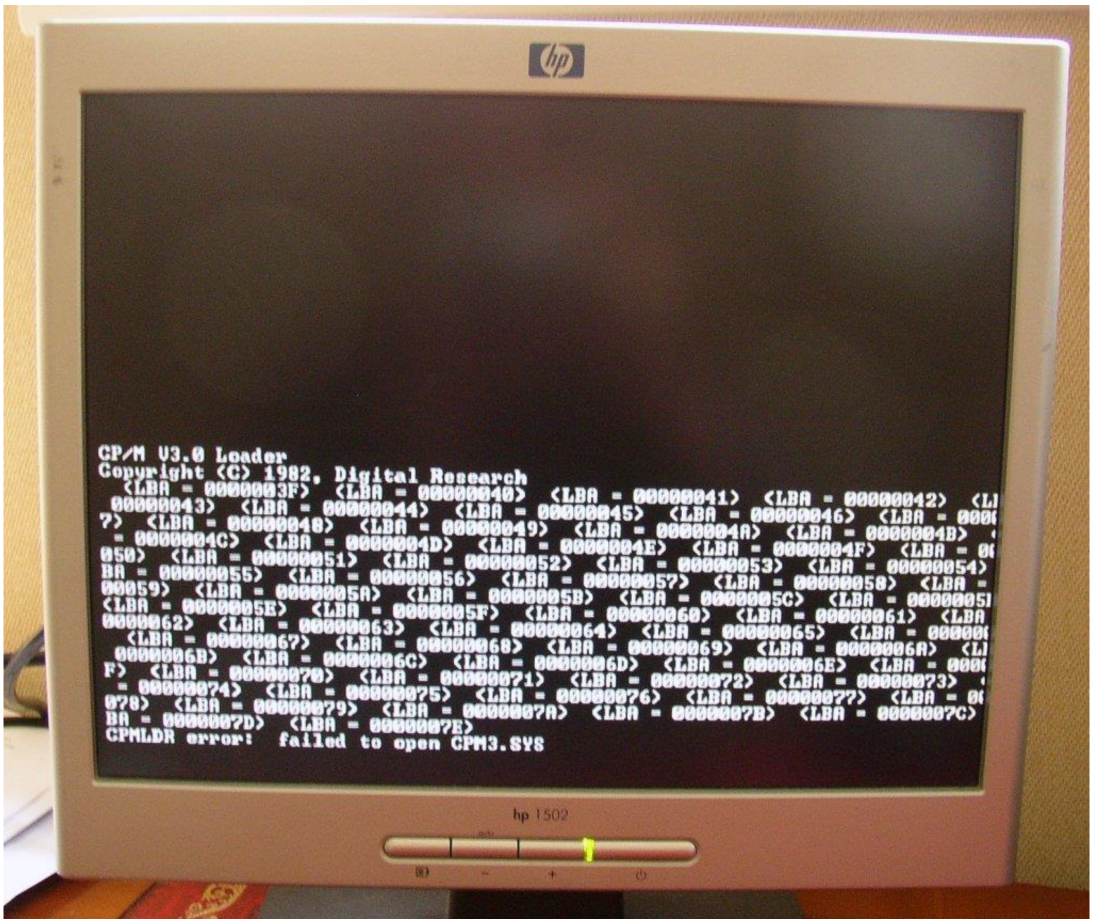
```
set hdsk1 geom=256/64/512
attach hdsk1 new.dsk
do cpm3

submit hmakecpm.sub

pip j:cpm3.sys=i:cpm3.sys
```

<u>Open our disk image</u>
From the Extras menu select ‘open disk image’ browse to our disk image ‘new.dsk’ then select open and accept the 512 byte sectors message. Scroll through the disk image and notice again that the directory table is located at sector LBA 48 (30H) and the file cpm3.sys is located at sector LBA 112 (7FH) onwards.
 Copy LBA sectors 112 - 135 from the disk image ‘new.dsk’ to the same sector numbers on the CF card anyway so that you can observe the problem.

*** Notice also that on the second line of the directory table entry we see 3 x 16 bit entries for the allocation units (AU’s) where the file cpm3.sys is stored***

```
00 43 50 4D 33 20 20 20 20 53 59 53 00 00 00 60          .CPM3    SYS...`
08 00 09 00 0A 00 00 00 00 00 00 00 00 00 00 00          ................
```

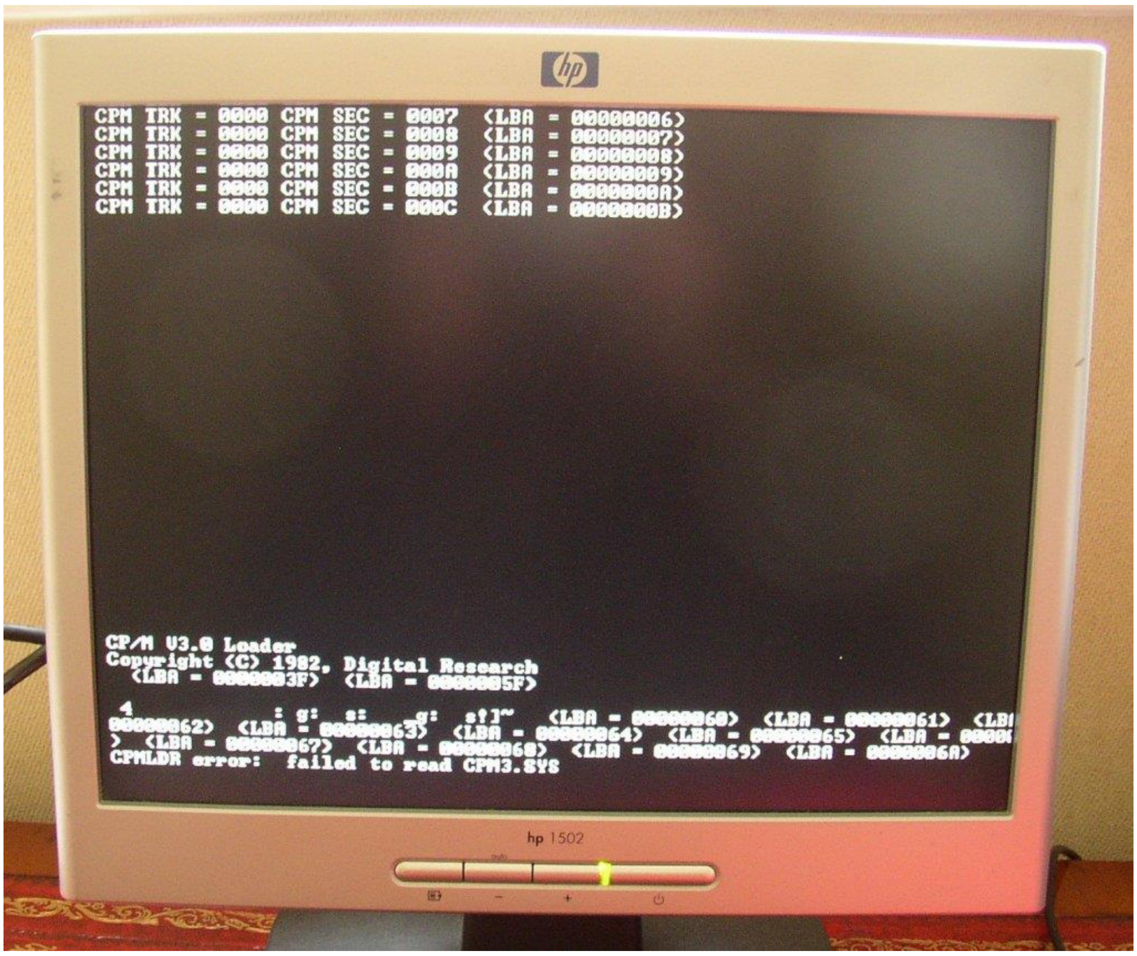
Remove the CF card from your card reader and try to boot your S100 Z80 system with it to see the effect.



```
CP/M V3.0 Loader
Copyright (C) 1982, Digital Research
  <LBA = 0000003F>  <LBA = 00000040>  <LBA = 00000041>  <LBA = 00000042>  <LI
 00000043>  <LBA = 00000044>  <LBA = 00000045>  <LBA = 00000046>  <LBA = 0000
7>  <LBA = 00000048>  <LBA = 00000049>  <LBA = 0000004A>  <LBA = 0000004B>  <
 = 0000004C>  <LBA = 0000004D>  <LBA = 0000004E>  <LBA = 0000004F>  <LBA = 00
050>  <LBA = 00000051>  <LBA = 00000052>  <LBA = 00000053>  <LBA = 00000054>
BA = 00000055>  <LBA = 00000056>  <LBA = 00000057>  <LBA = 00000058>  <LBA =
00059>  <LBA = 0000005A>  <LBA = 0000005B>  <LBA = 0000005C>  <LBA = 0000005I
<LBA = 0000005E>  <LBA = 0000005F>  <LBA = 00000060>  <LBA = 00000061>  <LBA
0000062>  <LBA = 00000063>  <LBA = 00000064>  <LBA = 00000065>  <LBA = 00000
  <LBA = 00000067>  <LBA = 00000068>  <LBA = 00000069>  <LBA = 0000006A>  <LI
 0000006B>  <LBA = 0000006C>  <LBA = 0000006D>  <LBA = 0000006E>  <LBA = 0000
F>  <LBA = 00000070>  <LBA = 00000071>  <LBA = 00000072>  <LBA = 00000073>  <
 = 00000074>  <LBA = 00000075>  <LBA = 00000076>  <LBA = 00000077>  <LBA = 00
078>  <LBA = 00000079>  <LBA = 0000007A>  <LBA = 0000007B>  <LBA = 0000007C>
BA = 0000007D>  <LBA = 0000007E>
CPMLDR error:  failed to open CPM3.SYS
```

When you try to boot CP/M you will see that cpmldr.com tries to locate the directory table on what could be thought of as ‘track 1’ starting at LBA 3FH and finishing at 7EH (64 sectors) but from our copy operation above the directory table was placed at LBA 48 (LBA 30H) so it needs to be moved.

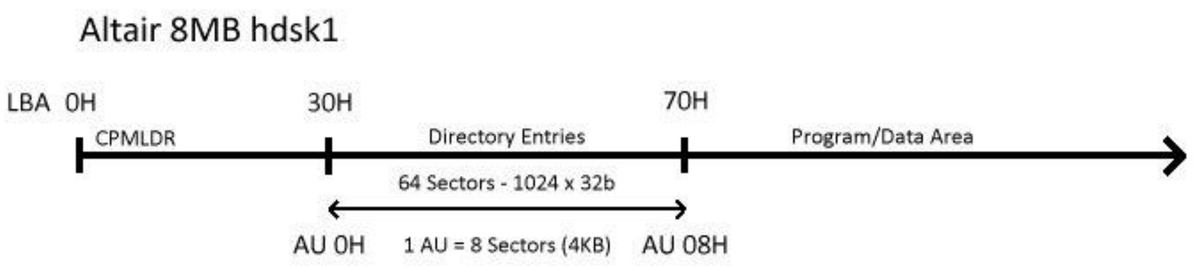
Load the CF card back into HxD Hex Editor and copy LBA sector 48 to LBA sector 63, then overwrite LBA sector 48 with E5's and save changes. Now try to boot the CF card again to observe the next problem.

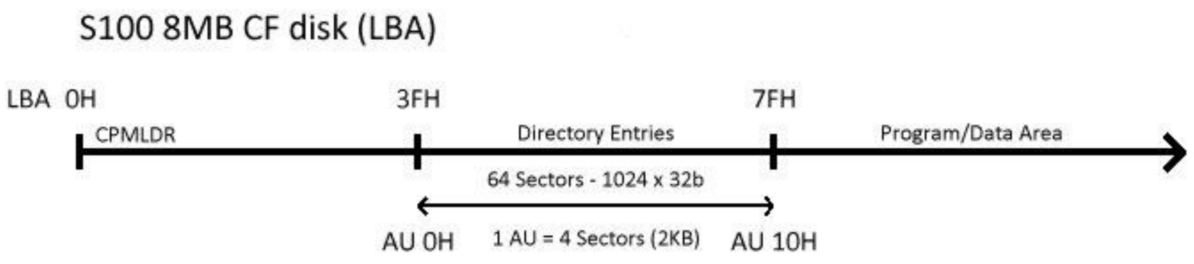


```
CPM TRK = 0000 CPM SEC = 0007  <LBA = 00000006>
CPM TRK = 0000 CPM SEC = 0008  <LBA = 00000007>
CPM TRK = 0000 CPM SEC = 0009  <LBA = 00000008>
CPM TRK = 0000 CPM SEC = 000A  <LBA = 00000009>
CPM TRK = 0000 CPM SEC = 000B  <LBA = 0000000A>
CPM TRK = 0000 CPM SEC = 000C  <LBA = 0000000B>




CP/M V3.0 Loader
Copyright (C) 1982, Digital Research
  <LBA = 0000003F>  <LBA = 0000005F>

 4              : g:  s:     g:  s!]~   <LBA = 00000060>  <LBA = 00000061>  <LB
00000062>  <LBA = 00000063>  <LBA = 00000064>  <LBA = 00000065>  <LBA = 0000
>  <LBA = 00000067>  <LBA = 00000068>  <LBA = 00000069>  <LBA = 0000006A>
CPMLDR error:  failed to read CPM3.SYS
```

We can now see that cpmldr.com has located the directory table at LBA sector 3FH and has attempted to read the first sector of cpm3.sys at LBA sector 5FH it then turns to jibberish, so the next question is where should the file cpm3.sys be located on the CF card ?

Going back to the beginning of this section we can see that in our disk image ‘new.dsk’ the directory table was located at LBA 30H and the data started at LBA 70H, a difference of 64 sectors or 32Kbytes. When the disk image was created on the SIMH Altair we specified the disk geometry but the program chose the number of directory entries and the Allocation Unit size for us. We still need 64 sectors on our CF card for the 1024 directory entries so with the directory table now starting at LBA 3FH it should finish at LBA 7EH inclusive making the start of the program/data area LBA 7FH. With this in mind we also need to update the directory table entry for cpm3.sys with the correct AU entries, there will also be twice as many entries as our Allocation Unit size was set up as 2kB clusters (AU’s) in the CP/M bios source files and not 4kB that the Altair sim has created.





So, load the CF card back into HxD Hex Editor and copy the data LBA sectors 112 – 135 (cpm3.sys) on the CF card to LBA sectors 127 – 150 then fill LBA 112 – 126 with E5’s to tidy up, save the changes to the CF card.

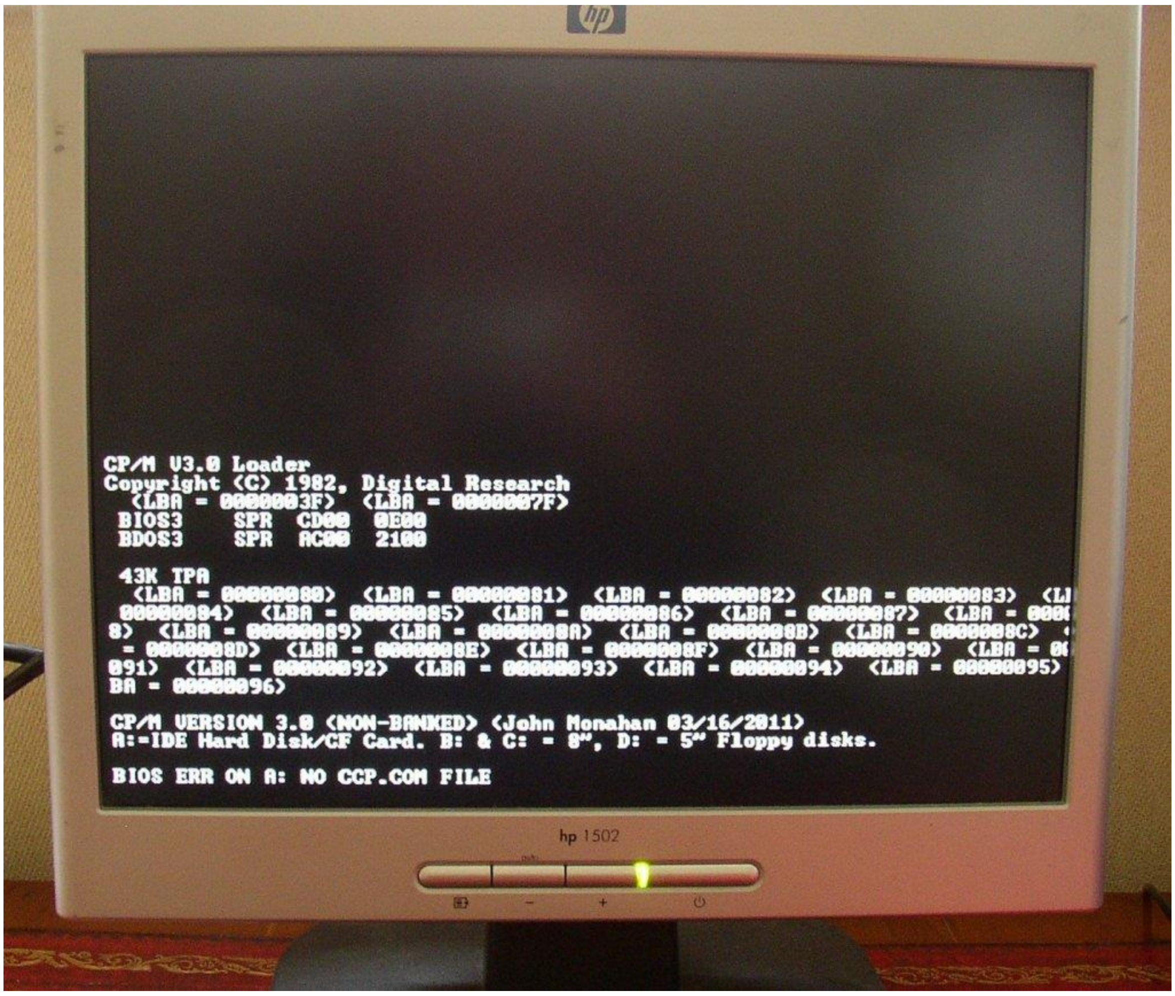
We now need to update the directory table entry to reflect the new location of the file.

```
00 43 50 4D 33 20 20 20 20 53 59 53 00 00 00 60        .CPM3    SYS...`
10 00 11 00 12 00 13 00 14 00 15 00 00 00 00 00        ................
```

Referring to the diagram above we can see that the first valid allocation unit in the program/data area is 10H So amend the entries in the directory table with sequential values starting at 10H remembering that we have twice as many entries due to the smaller cluster (AU) size.

Save the changes by clicking the floppy disk icon on the toolbar.

Now load the CF card back into the S100 system and attempt another CP/M boot.



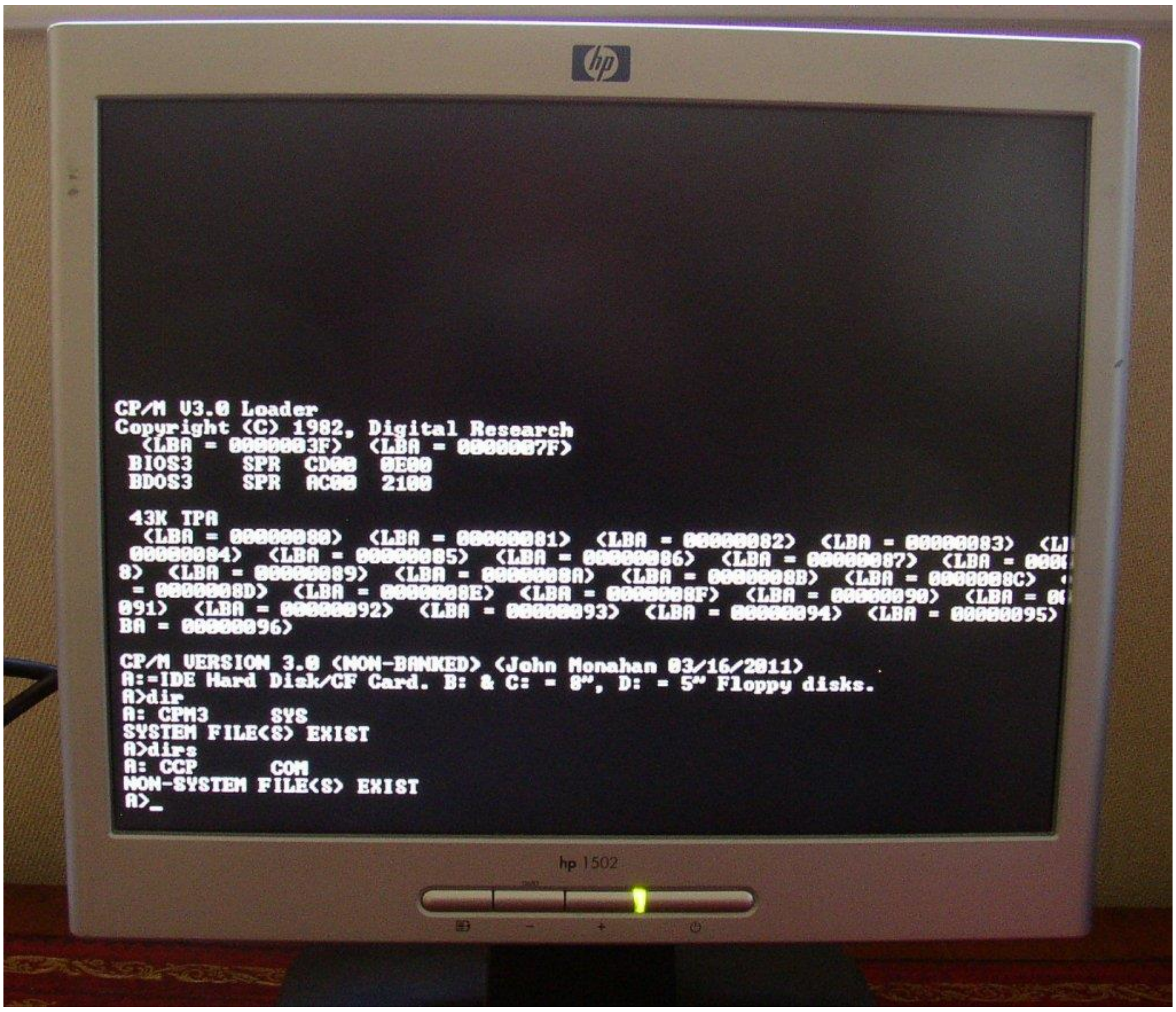
```
CP/M V3.0 Loader
Copyright (C) 1982, Digital Research
  <LBA = 0000003F>  <LBA = 0000007F>
 BIOS3    SPR  CD00  0E00
 BDOS3    SPR  AC00  2100

 43K TPA
  <LBA = 00000080>  <LBA = 00000081>  <LBA = 00000082>  <LBA = 00000083>  <LI
00000084>  <LBA = 00000085>  <LBA = 00000086>  <LBA = 00000087>  <LBA = 000
8>  <LBA = 00000089>  <LBA = 0000008A>  <LBA = 0000008B>  <LBA = 0000008C>  <
 = 0000008D>  <LBA = 0000008E>  <LBA = 0000008F>  <LBA = 00000090>  <LBA = 00
091>  <LBA = 00000092>  <LBA = 00000093>  <LBA = 00000094>  <LBA = 00000095>
BA = 00000096>

CP/M VERSION 3.0 (NON-BANKED) (John Monahan 03/16/2011)
A:=IDE Hard Disk/CF Card. B: & C: = 8", D: = 5" Floppy disks.

BIOS ERR ON A: NO CCP.COM FILE
```

We can now see that CPMLDR.COM has loaded up and handed control over to CPM3.SYS (no more LBA sector numbers reported) which is now reporting that CCP.COM could not be found.

You now need to repeat the above process to copy CCP.COM from the SIMH Altair 'i' drive to a blank drive 'j'
Using the command
pip j:CCP.COM=i:CCP.COM[R]

Place the CCP.COM file data in LBA sectors 151 – 157 on the CF card and then copy and paste the directory table entry and place it beneath the directory table entry for CPM3.SYS
The correct AU values for the CCP.SYS entry are shown below.

```
00 43 43 50 20 20 20 20 20 C3 CF 4D 00 00 00 19     .CCP     ÃÏM....
16 00 17 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
```



```
CP/M V3.0 Loader
Copyright (C) 1982, Digital Research
  <LBA = 0000003F>  <LBA = 0000007F>
 BIOS3    SPR  CD00  0E00
 BDOS3    SPR  AC00  2100

 43K TPA
  <LBA = 00000080>  <LBA = 00000081>  <LBA = 00000082>  <LBA = 00000083>  <LI
 00000084>  <LBA = 00000085>  <LBA = 00000086>  <LBA = 00000087>  <LBA = 000
8>  <LBA = 00000089>  <LBA = 0000008A>  <LBA = 0000008B>  <LBA = 0000008C>  <
 = 0000008D>  <LBA = 0000008E>  <LBA = 0000008F>  <LBA = 00000090>  <LBA = 00
091>  <LBA = 00000092>  <LBA = 00000093>  <LBA = 00000094>  <LBA = 00000095>
BA = 00000096>

CP/M VERSION 3.0 (NON-BANKED) (John Monahan 03/16/2011)
A:=IDE Hard Disk/CF Card. B: & C: = 8", D: = 5" Floppy disks.
A>dir
A: CPM3     SYS
SYSTEM FILE(S) EXIST
A>dirs
A: CCP      COM
NON-SYSTEM FILE(S) EXIST
A>_
```

Finally the system boots to a command prompt and we can perform a DIR command to view our files. As a final step you may want to go back to recompiling CPMLDR from the version without the LBA sector reporting to clean up the screen.

Enjoy........