

Comments from George Fulk of IBM

June 23, 2004

1. Section 8. The 4th sentence starts "The extended functions...". That should read "The legacy functions..." <OK>
2. Section 8. Minor typo in the 3rd sentence "DS;SI" should be "DS:SI". Use a colon instead of semi-colon. <OK>
3. Section 8. The following (or if you have better words) needs to be added. "All registers and data structures not explicitly identified in a particular API must be preserved." For instance, I recently had a new machine that would not boot some versions of DOS, OS/2, or older Windows. All 3 of those OSs used a particular int 13h call during the early boot sequence. This particular machine (with it's bad BIOS) corrupted a register it had no business using. This request is a seriously needed addition to the spec. Without it, a BIOS maker could claim no restrictions on register corruption. The register corruption is not only the 16-bit registers either. I've been BIOSs that corrupt the upper 16-bits of the 32-bit registers as well. <OK>
4. Table 5 Boot Volume Descriptor. Minor typo. I think the last field should be "4Bh-7FFh" not "4Ah-7FFh". <OK>
5. Table 7 Initial/Default Entry. Minor typo. The field at offset "06h-07h" should be type "Word" not "Byte". <OK>
6. Table 9 Section Entry. Minor typo. The field at offset "06h-07h" should be type "Word" not "Byte". <OK>
7. Table 9 Section Entry. Minor typo. The field at offset "0Dh-0Fh" should be "reserved" not "Selection Criteria". <OK>
8. Section 8.2. On exit, the returning AL register is missing. AL returns with the status from the prior int 13h operation. AH is always the status of the current operation. Need to add this text under carry clear: "AL - status of last int 13h operation". <OK>
9. Section 8.3. On exit, the returning AL register is missing. Also, I like to be very explicit in my definitions, so I'd like to see a statement about the buffer being filled on return. Add this text under carry clear success: "AL - returns the actual number of sectors read" and "buffer (pointed to by ES:BX) is filled with read data". [*Editors Note: Does the return value have meaning when data can be returned out of order*]
10. Section 8.3. What about partial success/failure? How is that going to be handled? For instance, a request to read 4 sectors from a device, but only 1 sector is available. Should BIOS fail the request outright without any read to the ES:BX data buffer? Or, should BIOS read what is available returning AL=1 and placing the 1 successfully read sector into the buffer pointed to by ES:BX. If the partial read is performed, does BIOS return CY AH=0 success, because a partial success occurred. In my opinion, based on legacy behavior, a partial read should fill the

buffer with the successfully read sectors. AL should have the number of sectors successfully read. AH should contain an error code and the CY flag on indicating an error. If this is acceptable, then this text should be added to carry set exit: "AL - returns the number of actual sector read (partial success possible)" and "buffer (pointed to by ES:BX) is filled with partial successfully read data". Here's a simple test to check for yourself the current behavior on a machine I just started up DOS and the DEBUG utility and stepped through this code.

```
mov ah,8
mov dl,0
int 13
mov dl,0
mov bx,200
push cs
pop es
mov ax,204
int 13
int 20
```

This sample code will first query the geometry of the A: diskette. CX and DH are filled with the max HST (aka CHS) values, so that they are pointing to the very last sector on the diskette. The read request is for 4 sectors, but only 1 sector is available. The behavior I've seen on most legacy machines is to read the 1 available sector into ES:BX, set CY on to indicate an error, set an error code in AH, and set AL to 1 to indicate 1 sector successfully read. ***[Editors Note: Does the return value have meaning when data can be returned out of order]***

11. Section 8.4. Function 3 write sectors has the exact same comments from #9 and #10 above that were mentioned for function 2 read sectors. ***[Editors Note: Does the return value have meaning when data can be returned out of order]***
12. Section 8.5. Function 4 verify sectors has the same comments regarding the AL register from #9 and #10 above that were mentioned for function 2 read sectors. Since verify does not have an ES:BX pointed buffer, only the AL comments from #9 and #10 apply here. Note, this also includes the same comments regarding partial reads from #10. That a partial verify will return CY AH=error AL=actual verified number of sectors. ***[Editors Note: Does the return value have meaning when data can be returned out of order]***
13. Missing int 13h ah=5 format sectors. This function has been around since the first days of the PC, and is still used today to format diskettes and hard disks. Note, the syntax for formatting diskettes and hard disks is different. Since this is an old, well known API it needs be mentioned. Either define the API in this spec, or define it as "BIOS implementation dependant". IMHO I prefer the first choice of adding the definition to the spec. ***[Editors Note: Does this have meaning today? Need statement from Phoenix & AMI]***
14. Section 3.2.6. CHS definition. Above I mentioned another old abbreviation HST. It stands for Head/Sector/Track. It's the exact same term as CHS. HST was used in the PC-1 days (1981). Somewhere around the PC-AT days (1984) the term CHS was introduced as a replacement for HST. <Rejected>

15. Section 8.9. i13.0C seek. There should be no ES:BX input parameter. <OK>
16. Section 8.14. i13.18 set media. The AH on entry should be 18h, not 15h. <OK>
17. Section 8.16 i13.25 identify. Entry AH should be 25h, not 19h. <OK>
18. Section 8.21 Lock. I believe what you have here is correct. This is just an interesting thought. AL returning the lock count instead of just 0 or 1 would be a lot more useful. Afterall, you make a call and get back a status of AL=1=Locked. You must issue some number of Unlocks until the media is free. Since the number of locks is (conviently) defined as unsigned char, the lock count could be returned in AL instead of a just the flag 1. <Rejected>
19. Section 8.24 i13.48 LBA get parms. Table 12 offset 0. The description about lengths is old. Specifically the comment "if the buffer length is 30 or greater on entry, it shall be 30 on exit." <Written as intended>
20. Section 8.24 i13.48 LBA get parms. Table 12 offset 32. The length should be 2Ch or 44 decimal, not 44h. <OK>
21. Section 8.29 i13.4D get catalog. The paragraph of text following table 18 is accidently copied from above. Needs to be deleted here. <OK>
22. Section 8.27 i13.4B01 get ElTorito status. Nothing wrong with what you have here. Just a general comment (my own personal experience). This is one of the worst APIs in existance. Nobody correctly implements this API in BIOS. Just about everyone has screwed up this one. I my latest PC-DOS we have ElTorito CD-Rom drive support. I use this API to try and find available drives, and the code looks terrible because of all the BIOS mistakes. Flame off. <good job!>
23. Section 8.6 i13.8 get CHS parms. AL returning 0 appears to be a BIOS change introduced in the 90's. I don't think it does any harm today. Since older BIOSes didn't change the value of AL on return you might want to label AL's return value as "reserved" instead of 0. <Will remove the AL return>
24. same section. You stated that this version of the document addresses hard disks (or non-removeable) media only. Floppies also use this API and return values in BL and ES:DI. Wouldn't it be good to label BL and ES:DI as reserved, so there's no confussion that those registers aren't a guarantee to be preserved. <OK>
25. same section. DL returns "total number of hard drives...". As mentioned on the phone would it be better to use the phrase "non-removeable" media than hard disk? Since various devices might be added to 0x80 chain that aren't really hard disks. <Will change to say total number of devices with an INNT 13 device number greater than 7Fh>
26. General comment about i13.0A,0B,0E,0F... if I'm correct these are all diagnostic APIs. No application would have a need to use them. <Still used in a manufacturing environment>
27. Returning to i13.02,03,04 return value AL. I know that BIOS of old always returned AL with the number of sectors actually transferred (same as i13.42/43/44 has defined). I've seen a couple of BIOS recently were under ElTorito boot they

return a corrupted value in AL. I'm not 100% certain if failing to correctly set AL on return will cause OS problems or not. I'll look through my source code and see if it is critical. Although very OLD programs could have a problem since IBM Tech Ref manuals of the 80s document the AL return value. <Refer to #9>