

Re: Throughput question with CF/DiskOnChip

Source: <http://linux.derkeiler.com/Newsgroups/comp.os.linux.embedded/2005-01/0008.html>

From: Michael Schnell (mschnell_at_bschnell_dot_de_at_aol.com)

Date: 01/02/05

Date: Sun, 02 Jan 2005 15:19:00 +0100

> *Hmmm...I have not read about this. Of course I can understand the possibility of losing a portion of a file that is in the process of being flushed to disk if the CF is removed with the write in progress (and this is an acceptable constraint for this particular application). However, I don't understand how the CF card could be damaged and made unusable if it is removed at the wrong time. My PDA has a CF card, and it has no knowledge of when I would remove the CF card...but I can remove the card whenever I want. How does it solve this problem?*

>

This has been discussed several times in this forum. Please see the back-log. To handle wear out effects, a CF internally monitors write and erase times and replaces dying blocks with fresh (spare) ones. To do this it uses some blocks as reference table. If such a table is changed (which can happen within any a normal write request), part of it is erased and then rewritten. When power goes down in that moment the table is damaged and the CF is unusable as it can't find the internal memory block associated to an external address range. It can't even be formatted with normal means. There will be special propriety IDE commands that allow to revive the CF by rewriting the address allocation table to a standard default allocation (loosing all data).

>

>

> *The ext2 files system on top of the DiskOnChip appears to be a fairly common activity:*

>

> http://www.rtd.com/NEW_appnote/SWM-640000017%20rev%20A.pdf

> <http://www.gctglobal.com/Download/DiskOnChip/diskonchip.html>

> <http://www.denx.de/twiki/bin/view/DULG/DiskOnChip>

>

> *Why exactly do you discourage this activity?*

>

>

I was speaking about a DOC with no hardware intelligence (like that of a CF) but that is a quite naked flash chip.

EXP2 and all "normal" file systems rewrite some blocks of the "disk" (e.g. the FAT with a DOS file system, more complex structures with e.g. EXP2). Writing a single byte in a Flash causes a complete block (size depending on hardware 128 Byte ... 128 K) to be erased and rewritten. A flash block can only be rewritten a certain number of times (depending on hardware 10.000 ... 1.000.000 times). When using the chip for some kind of log file, this number can be reached quite fast. A dedicated flash file system knows about the flash blocks and (re)uses them cyclicly to manage this "wear out" effect. Moreover it will automatically remove blocks that get unwritable in spite of the rotation scheme.

>

> *The DOC has a considerable on-board intelligence.*

If so, you are in bad luck, as now the same applies as with the CF. With a "dumb" DOC you can use a Flash file system, while with a CF-Card type of device this does not help, as only the manufacturer knows about the block size and handling.

>

> *IIRC, JFFS2 was in the works for supporting the DOC, but at the time > the system was being developed it was not ready yet.*

JFFS2 is ready since ages, but of course you need a media access driver for the device in question, too. If that is available now and specified for JFFS usage, this seems the way to go.

> *we haven't seen the*

> *device fail (unless, of course, the "blip" we are seeing is due to the > number of bad blocks increasing, and thus the DOC driver/hardware > taking longer and longer to find valid flash blocks to write to).*

>

I don't think so. (There should be a way to read the count of bad blocks from the media access driver.) I suppose the "blip" is just a cache write-back, that EXT2 does. This is not a good thing, either, as a large cache will make you loose a lot of data when power fails and the file system is not dismounted before. Here a journaling file system could help a lot, but same creates even more write accesses and increases the wear out problem.

-Michael