Diagram

Description automatically generated



to se...@googlegroups.com, Joseph Honold

I had those questions too, and through a weird stroke of luck Teo Swee Ann (designer of ESP32, and CEO//Founder of Espressif) contacted me to compliment our project on the work we've done so far, so I took the opportunity to have a long discussion with him on a bunch of internals, and he stated, quite plainly:

\* The ESP32's GPIO pins are long term +5v tolerant (they can read +5v just fine, and it does not impact the MTBF of the device).

\* The consequence of this can be some forward biased readings from the nearby DAC

\* The ESP32 does indeed assert GPIO at +3.3v.

As for the 74LS07, we needed an open collector driver to properly drive the Atari SIO bus when there are multiple peripherals connected, and it took Joe a few revisions to come up with a circuit that worked well. It isn't needed if the Fujinet is the only device on the bus. For the Apple2 Smartport version, it winds up getting in the way, for example, so we reverted to the 1.0 design that directly connects the SmartPort to the ESP32.

As for your last question regarding the MOSFETs, I believe so, but maybe Joe can elaborate more on this. (hop on the list?) hehe.

-Thom





You received this message because you are subscribed to a topic in the Google Groups "SEBHC" group.  
To unsubscribe from this topic, visit <https://groups.google.com/d/topic/sebhc/JUzWjF1xhIE/unsubscribe>.  
To unsubscribe from this group and all its topics, send an email to [sebhc+un...@googlegroups.com](https://groups.google.com/).  
To view this discussion on the web visit [https://groups.google.com/d/msgid/sebhc/721854df-0b0d-4851-8826-b7c88bd7ba28n%40googlegroups.com](https://groups.google.com/d/msgid/sebhc/721854df-0b0d-4851-8826-b7c88bd7ba28n%40googlegroups.com?utm_medium=email&utm_source=footer).

thom.che...@gmail.com's profile photo

thom.che...@gmail.com

Jan 23, 2022, 11:28:55 AM

to SEBHC

Because it didn't make it to the list:  
  
**Joseph Honold**

10:12 AM (15 minutes ago)

Those FETs turn off the buffer when you turn off FujiNet which cuts off all the lines from the esp32 to SIO. We had interference noise when FujiNet was off and Atari was on. The esp32 was being back-powered through the gpio pins.

# LBA

Douglas Miller

Apr 6, 2022, 10:32:27 AM (9 days ago)

to se...@googlegroups.com

In my opinion, emulating the H17 is a non-starter. It is the most software-heavy of all the devices, and emulating it would also be heavy.

MTR-90 might have space in it, I know the MMS-84B ROM (sort of based on MTR-90) does not have space. If you have the new Z80 v4.0 CPU board, we have plenty of space in that ROM.

It looks like you a loosely emulating SCSI protocol/commands? with extensions for the "juke box" feature for selecting disk images? So, that would be a new boot routine even if it exactly matched SCSI (I don't think the existing H67/SASI would work anyway).

So, the H89 (console) boot semantics, I think starting with MTR-89, allow for a "string" to be appending to the boot command. I use this in the Z80 v4.0 ROM to allow for booting VDIP1 files or selecting CP/NET boot image files. If this is working in the ROM you choose to modify, that could be the mechanism to select the ESP32 image you want to boot from. Something like:

H: Boot ZZ-0:somefile

where 'Z' is whatever letter you choose to represent the ESP32 (a lot are already assigned). Of course, ROM space is still an issue.

One comment on the commands: is it necessary to use head/track/sector rather than "logical block address"? The whole head/track/sector stuff sorts of restricts how the storage space is handled, and creates overhead in the ESP32. It also may limit the size of storage images that can be used - I'm thinking that "floppy only" may be too restrictive.

I think HDOS internally uses LBA, so that one is easy. For CP/M, the H67 (SASI) driver would be a good example. For the GIDE and SDC interfaces, we also use LBA.

With the MTR-90 boot syntax, I was envisioning that:

**H: Boot ZZ-0:somefile**

**would translate to your "Select disk image" command (cmd 21) using the "0" as the LUN (disk number) and "somefile" as the string (filename to "mount") - followed by the typical boot (read 10 sectors and jump to 0x2280).**

Another thought for future expansion is providing a way to "pass-through" commands (blobs) to other subsystems inside the ESP32 (i.e. not only SCSI-like storage commands). It's sort of like your "juke box" commands. My first example for this would be CP/NET, but I'm pretty sure there are others. SCSI provides a mechanism for doing that, not that you need to follow it, for example to implement SED-OPAL drives: where a completely separate "entity" exists within the drive firmware and one can encapsulate OPAL commands in some special SCSI command block, and the designated entity then receives, interprets, and executes the "blob". That will also need to include some mechanism for "receiving" response "blobs". We can iterate on exactly how to do that.

I'm not sure of the official source for Heath CP/M BIOS, but the MMS CP/M 2.2 driver for the H67 (MMS 77320) is at https://github.com/durgadas311/MmsCpm3/blob/master/cpm2/sys/src/m320.asm

The code in SET$SEC translates CP/M track and sector values into LBA. Note that this code handles 256 or 512 byte sector sizes (maybe also 128 and 1024) - via BLCODE -, so is a bit more complex. The HSTSEC value has already been converted to physical sector number. Also, the CP/M track and sector values are "arbitrary" and determined by the DPB. The DPBs are read off the disk, so are not present in the code. However, I believe the sectors-per-track was always 64 (enforced). Note that this code is doing 24-bit arithmetic, at least for the final LBA computation which adds in the partition offset. SECPTR is a pointer to the partition table entry for the selected drive, 3 bytes of offset in 128-byte (logical) sector units.

Basically, in C parlance, this code should be doing:

lba = (((hsttrk << 6) + partition[drive]) >> blcode) + hstsec;

This LBA value is then stored in the SASI command buffer, CMBFR. Note that the SASI (as well as SCSI) commands store the LBA big-endian. (the partition table is also maintained big-endian)

Let me know if you have any further questions.

CHS conversion[[edit](https://en.wikipedia.org/w/index.php?title=Logical_block_addressing&action=edit&section=5)]

|  |  |
| --- | --- |
| **LBA and CHS equivalence with 16 heads per cylinder** | |
| **LBA value** | **CHS tuple** |
| 0 | 0, 0, 1 |
| 1 | 0, 0, 2 |
| 2 | 0, 0, 3 |
| 62 | 0, 0, 63 |
| 63 | 0, 1, 1 |
| 945 | 0, 15, 1 |
| 1007 | 0, 15, 63 |
| 1008 | 1, 0, 1 |
| 1070 | 1, 0, 63 |
| 1071 | 1, 1, 1 |
| 1133 | 1, 1, 63 |
| 1134 | 1, 2, 1 |
| 2015 | 1, 15, 63 |
| 2016 | 2, 0, 1 |
| 16,127 | 15, 15, 63 |
| 16,128 | 16, 0, 1 |
| 32,255 | 31, 15, 63 |
| 32,256 | 32, 0, 1 |
| 16,450,559 | 16319, 15, 63 |
| 16,514,063 | 16382, 15, 63 |

In the LBA addressing scheme, sectors are numbered as integer indexes; when mapped to CHS ([cylinder-head-sector](https://en.wikipedia.org/wiki/Cylinder-head-sector)) [tuples](https://en.wikipedia.org/wiki/Tuple), LBA numbering starts with the first cylinder, first head, and track's first sector. Once the track is exhausted, numbering continues to the second head, while staying inside the first cylinder. Once all heads inside the first cylinder are exhausted, numbering continues from the second cylinder, etc. Thus, the lower the LBA value is, the closer the physical sector is to the hard drive's first (that is, outermost[[5]](https://en.wikipedia.org/wiki/Logical_block_addressing#cite_note-5)) cylinder.

CHS tuples can be mapped to LBA address with the following formula:[[6]](https://en.wikipedia.org/wiki/Logical_block_addressing#cite_note-6)[[7]](https://en.wikipedia.org/wiki/Logical_block_addressing#cite_note-7)

*LBA* = (*C* × *HPC* + *H*) × *SPT* + (S − 1)

where

* *C*, *H* and *S* are the cylinder number, the head number, and the sector number
* *LBA* is the logical block address
* *HPC* is the maximum number of heads per cylinder (reported by disk drive, typically 16 for 28-bit LBA)
* *SPT* is the maximum number of sectors per track (reported by disk drive, typically 63 for 28-bit LBA)

LBA addresses can be mapped to CHS tuples with the following formula ("mod" is the [modulo operation](https://en.wikipedia.org/wiki/Modulo_operation), i.e. the [remainder](https://en.wikipedia.org/wiki/Remainder), and "÷" is [integer division](https://en.wikipedia.org/wiki/Integer_division), i.e. the [quotient](https://en.wikipedia.org/wiki/Quotient) of the division where any fractional part is discarded):

*C* = *LBA* ÷ (*HPC* × *SPT*)

*H* = (*LBA* ÷ *SPT*) mod *HPC*

*S* = (*LBA* mod *SPT*) + 1

According to the ATA specifications, "If the content of words (61:60) is greater than or equal to 16,514,064, then the content of word 1 [the number of logical cylinders] shall be equal to 16,383."[[1]](https://en.wikipedia.org/wiki/Logical_block_addressing#cite_note-Working_Draft_of_ATA/ATAPI-5-1): 20 Therefore, for LBA 16450559, an ATA drive may actually respond with the CHS *tuple* (16319, 15, 63), and the number of cylinders in this scheme must be much larger than 1024 allowed by INT 13h.[[a]](https://en.wikipedia.org/wiki/Logical_block_addressing#cite_note-8)