

Re: BIOS startup ??

Source: <http://linux.derkeiler.com/Mailing-Lists/Fedora/2007-08/msg03298.html>

- *From:* Les <hlhowell@xxxxxxxxxxxx>
 - *Date:* Sun, 19 Aug 2007 16:10:15 -0700
-

BIG SNIP!!

To describe what happens depends on what kind of hardware is being discussed. So first lets talk about how the processor itself starts up.

The power supply will receive AC, which is passed through a transformer and rectified (in switching mode supplies, sometimes the AC is rectified and chopped before going to the transformer), but eventually DC is created and controlled by circuitry in the powersupply. The voltage appears on the power supply lines with the return via ground in the boards. At the processor control function is a "reset flipflop", and a time delay. The time delay components are designed to operate either based on voltage (the supply exceeds a threshold), or just time by a resistor in series with a capacitor. As long as the voltage is below the reset threshold of the processor, the processor is stopped. At the point where the threshold is crossed the processor receives its first clock and begins processing from the default address (typically 0x0000000 on Intel x86 and related devices). At that location in ROM is a program called the bootstrap and from there the processor executes the bootstrap code. At the same time, disk drives have their heads retracted (they were moved there during the powerdown cycle) and the disk spins up. A tachometer monitors disk speed and holds off disk access until the disk is spinning correctly to prevent a "head crash". The air held to the disk by friction creates a small cushion so the head really "flies" a small distance from the disk. The head then moves to track 0, which is the inside track of the disk. When the boot strap starts up, it reasserts the move to track 0 and looks for the first sector. The head decodes the information on the disk, looking for the tunnel markers that center the head in the track, and also decoding the data looking for the sector marks. When the sector matching that sent by the program, the disk begins reading and reads the sector under the head. This is the MBR of the disk. This code is only slightly smarter than the hardware bootstrap, and so it needs disk and sector location to find the program to actually boot. Once the boot-strap has loaded the boot sector, that code is then executed and tells the disk to seek to the proper location to find the loader to use. The correct disk is selected, the correct sector for the loader is found and the loader is loaded and then program controll jumps to it. Now the loader will either load a program, or will read some file(s) to discover what

Re: BIOS startup ??

alternatives are available. If more than one, the code will typically print out (using bios calls) the alternatives to select from. Most loaders will time out to the first selection if no choice is forthcoming from the user.

The loader then loads the kernel. The kernel loads the drivers for all applicable hardware, and begins the process of resetting and setting the access to each of the various pieces of hardware needed by the computer (that is why you hear the disks, see the access lights on hubs, and the various external drives all flickering. They are being accessed and tested for proper operation (generally "Hello, are you there" kind of code, nothing spectacular in terms of test. Disks are then mounted, the monitor reset and the correct refresh rates setup, and things begin to look like an OS is working.

At another level, during power up, each card receives a copy of the reset line being brought high. These cards all do whatever their designers intended for them as the base starting point in hardware. I need to mention here that all buss specifications generally have three forms of reset. First is the one being discussed here, that is HARD reset or power on reset. This is a brutal form of reset and is not interruptable. All other actions are immediately stopped and the hardware goes to its initial state. This can be hazardous for some kinds of hardware, like disk drives. That is why hot swappable drives contain a way to invoke another form of reset prior to the hard reset. This is the software reset. In this case the current operation is completed, and then the interrupt is permitted to continue. In most cases back to the level of hard reset, but some instruments require something less. And the final form of reset is a "data reset". Sometimes you just want the current operation interrupted and the instrument to go to a nice form of standby. This is a data reset, and is implemented on things like audio systems where you want to stop the song that is playing, but want it to really shut down the output and turn off the data stream, but not go to a full reset.

So the memory will do a reset to ensure its clocking and refresh circuitry are in sync with the buss access, the Direct Memory Access controller and CPU interface for multiple CPU's will go to reset to properly synchronize the buss access, and the video processor will go to reset to begin executing its own bootstrap start up. Some systems with smart buss controllers for firewire, USB, and ports will also execute their own bootstrap. Each processor sets a flat to tell the main processor it is not ready, and when they have finished their own bootstrap their flags will reset and the system proceeds.

At the same time, it is reasonable to remember that there are multiple clocks in the system, for buss access, video control, port clocking etc. etc. And typically all the clocks (except the real time clock) are reset at power on reset. This establishes a known condition, and helps the designer control glitch energy which could cause problems.

Re: BIOS startup ??

As clock speeds have increased from kilohertz (the Altair ran at 650Khz originally) to hundreds of gigahertz, some of these processes are being modified. It is impossible to accurately control the relative phase of two separate 400Ghz clocks. Instead, they are created by Phase locked loops such that each is locked to a master clock. This master clock and the edges provided to each other high clock are controlled to prevent too much concentrated switch energy. So, memory chips are staggered, the processor busses are staggered and so forth by controlling the relative phase of these clocks using a voltage bias applied to the PLL to create a fixed offset of each relative clock. At least this is one of about 50 methods used. In any event, clocking is undergoing some radical design requirements changes, so some of this may be archaic as you read it.

However the need for a known logical starting state for each subsystem is more or less universal, and on all systems, from PC's to exotic hardware I have worked on to test integrated circuits this is how busses are implemented. It is also necessary to realize that signals have a relative time factor as well. Address must precede data, which must precede read or write strobes. The time relevance must remain across the buss, so as speeds increase the need to accurately control buss line length has become more imperative. As we move toward photo computation, the era of wire signaling is coming to an end. Wire is not fast enough, nor can edges be accurately sustained. This has already caused the creation of compound architectures where data is sent on one line and received on another or complex switching such as DDRR processes for RAM.

If you are interested in the physics changes being discussed, I suggest you join the ACM (Association for Computing Machinery) or the IEEE (Institute of Electrical and Electronic Engineers). If you are young and working in the field you need to remain in touch with the professional associations to know which way the future winds are blowing. Computing and Electronics are careers where learning never stops. I still know AM and vacuum tubes, but never use it and that was only 40 years ago. I haven't switched in a bootstrap on my Altair in nearly 30 years, but that knowledge gives me an edge in some of these discussions.

I hope this helps, and I am glad you are interested.

Regards,
Les H

fedora-list mailing list
fedora-list@xxxxxxxxxx
To unsubscribe: <https://www.redhat.com/mailman/listinfo/fedora-list>