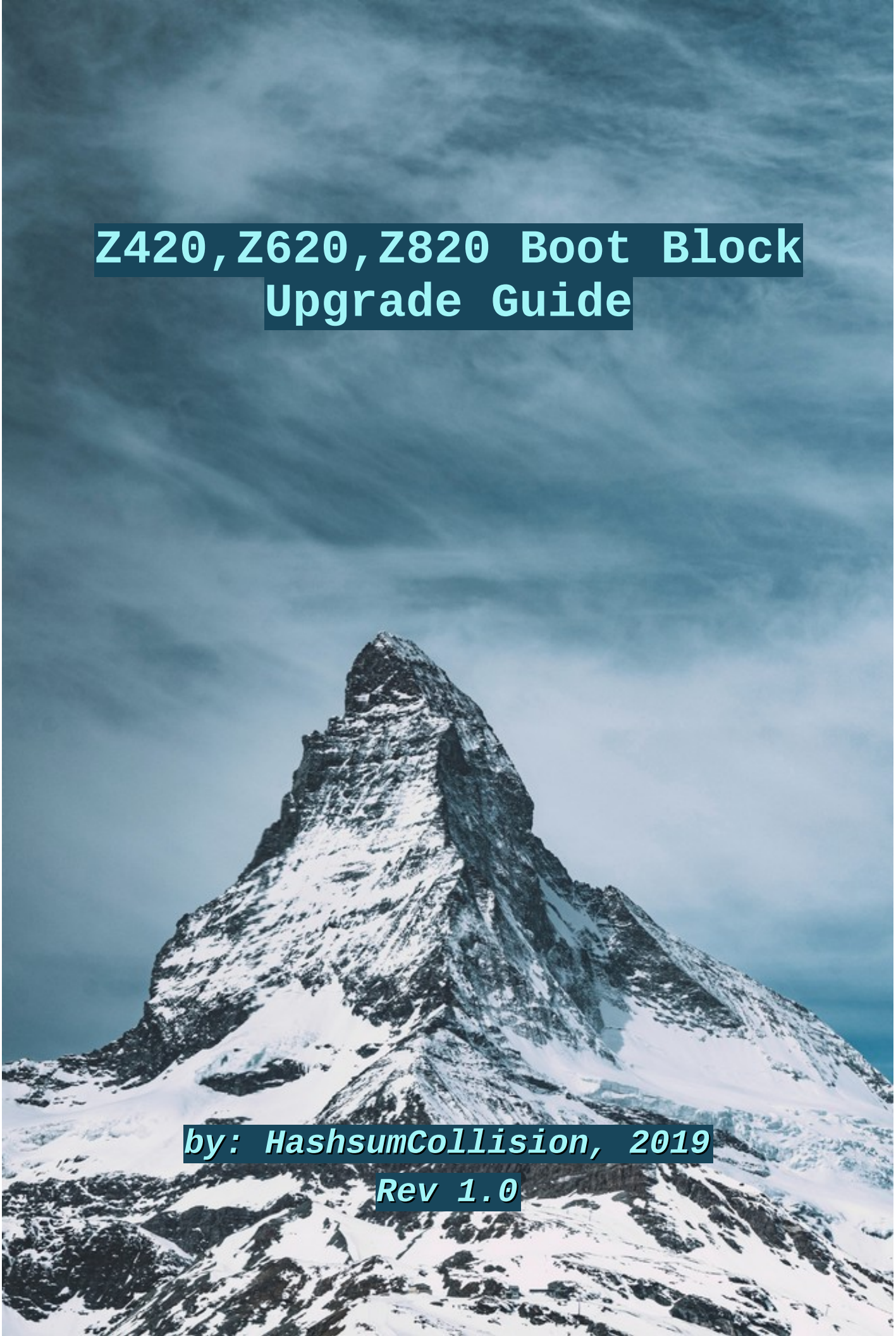
Z420,Z620,Z820 Boot Block Upgrade Guide



by: HashsumCollision, 2019

Rev 1.0

Motivation

Hewlett-Packard and the Two Boot Blocks

In 2011 HP released the Zx20 range of workstations – Z220, Z420, Z620, Z820. The Z420, Z620, and Z820 all used the new Intel Sandy Bridge-EP Xeon E5 processors with the C602 chipset. These processors have as many as 8 cores / 16 threads. The high end 8-core Xeon E5 2690 has a base clock of 2.9GHz and turbo as high as 3.8GHz.

In 2013 Intel released the Ivy Bridge successor chips to Sandy Bridge – compatible with the exact same C602 chipset. The only change needed is to update various parts of the BIOS code to support the new CPUs. The Ivy Bridge-EP chips could have as many as 10 or 12 cores. Some of the 8 core variants boost clock as high as 4GHz, and the highest end 12-core, the Xeon E5 2697 v2, clocks about the same the E5 2690 but with 12 cores: +50% multicore performance!. HP released ‘version 2’ Zx20 workstations with these new CPUs.

Sadly, HP did not design their BIOS updates for the Zx20 workstations in such a way to allow updates of the critical boot block section to add the code needed to support the new CPUs for the ‘version 1’ systems that have the original Sandy Bridge only boot block from 2011. The only difference between a ‘version 1’ (Sandy Bridge only) and ‘version 2’ (can use Sandy or Ivy Bridge CPUs) is that contents of a 16 megabyte flash chip on the motherboard. Even the second CPU riser for the Z620 is absolutely no different between the two versions.

So, if you can find a way to get the 2013 boot block into the BIOS flash chip, your version 1 Zx20 is now a version 2 with a lot more CPU selection.

The Easy Way

I will state outright that it is much easier to go online and buy a version 2 motherboard. It may cost 2-3x as much as a version 1 (*), but can be swapped into a system in 30 minutes.

The process of updating the boot block and other parts of the BIOS will probably take somewhere between 5 and 20 hours of your life, depending on existing experience. You will also likely need to wait a month or two for the supplies to arrive from China.

(*) this varies a lot. When I bought my v1 Z620 in Fall 2019 it was 1/3 the price of the cheapest v2 Z620. Perhaps it will get cheaper to get v2 systems and motherboards now that any given Ivy Bridge workstation is pushing 5+ years of use.

The Full Advantages of a 2013 Boot Block

In general the Ivy Bridge CPUs clock higher for the same number of cores and the same for those with extra cores. There is also a modest IPC improvement. Altogether, an upgrade could mean +25% in single and multi threaded performance or +50% in multi-core with slightly higher single-core.

There is also potentially 16.6% more RAM bandwidth – Ivy Bridge Xeons can use up to 1866MHz DDR3 in comparison to the 1600MHz of Sandy Bridge Xeons. This can make a noticeable difference by itself in certain applications.

Finally, Ivy Bridge was manufactured on Intel's 22nm process vs the 32nm process of Sandy Bridge, bringing lower power consumption.. Although Ivy Bridge CPUs sometimes run hotter due to changes to the CPU die heatsink (IHS), Ivy Bridge CPUs should use less power for a given workload.

Audience – who should be flashing their boot block?

- Technically experienced and curious people on a budget that want to have a powerful workstation, server, or even gaming PC
  - Already own a Zx20 V1 workstation or want to save 100 to 200 dollars by getting a V1 workstation
  - Prior experience in Linux (including command line) and basics of breadboarding electronics.
- Interested in learning more about how computers work at a low level. This is overwhelming as a beginner project but a great intermediate project.
- Willing to go through a little hassle - learn new skills, buy some (cheap) new equipment.
- Accepting of the potential risks – it's unlikely to have a permanently broken motherboard, but slightly possible to flash it wrong and need to restore the BIOS chip to its previous state.

Theory

SPI Flash Chips

SPI (Serial Peripheral Interface) is a very simple protocol that lets computer-ish things talk to each other in a standard way. SPI has a variable clockspeed; the faster you can clock your SPI flash chip, the higher the read and write speed you can achieve. In our case we will use 10MHz for speed or 512KHz for debugging.

SPI flash chips are a way for manufacturers to put a cheap, simple programmable chip on a motherboard where they can store the BIOS.

On the Zx20 workstations, this SPI flash chip is located on the bottom right corner of the motherboard, has 16 pins (although only 8 are used), and has a small circular dot on the chip's surface in a corner that indicates orientation.

The Boot Block Region

The boot block is the basic code of the BIOS. The "crisis recovery" code that allows you to recover from a bad BIOS update is contained here. It is located at the end of the SPI flash chip, in the final 65536 bytes.

The 2013 version boot block is required to be able to use v2 Xeons. It is not clear what code exists in the boot block that enables use of v2 Xeons, but in some way it must be there. The first BIOS version with a 2013 boot block and BIOS code compatible with v2 Xeons is v3.50.

Intel Management Engine BIOS Firmware

The Intel Management Engine (ME) is a small, separate computer that runs with absolute power over your actual computer. For corporations it enables remote management of PCs through the AMT feature.

It is located in the PCH (Platform Controller Hub), the motherboard chipset.

The BIOS contains firmware that allows it to communicate with the ME. With Sandy Bridge v7 firmware was shipped out, and with Ivy Bridge, v8 firmware. Version 8 firmware can still work with SB CPUs, but version 7 firmware is not designed to work with IB CPUs.

There is no actual benefit for most computer users provided by the ME, but it is always running and cannot be disabled (although, it can be neutered).

To fully complete the transformation of a version 1 Zx20 to version 2, ME version 8 firmware will be flashed to the SPI flash chip.

Zx20 BIOS Layout and Addresses

Modern Intel BIOSes are split into a standard set of regions and the Zx20 BIOSes are no different. A region is simply a range of bytes in the flash chip's storage. Specifically, these are the important byte addresses for Zx20 BIOS regions:

- Descriptor region: 0x000000 to 0x000FFF
- GbE: 0x001000 to 0x002FFF
- PDR: 0x003000 to 0x004FFF
- ME: 0x005000 to 0x50FFFF (~5 megabytes, the 'corporate' ME version)
- BIOS: 0x510000 to 0xFFFFFF (there's a lot more to a 'BIOS' than just the BIOS!)
- Boot block region: 0xFF0000 to 0xFFFFFF (within the BIOS region)

You can see that at the address one region ends, the next byte another region begins.

Things like MAC addresses and serial numbers are also stored in the BIOS flash in specific places:

- MAC addresses: "0x001000,0x002000,0x03470C, 0x03B84C AND 0x03B985"
- Serial number: "0x3B93A, 0x3BA6A"

This is why you need to be careful about flashing an entire BIOS update straight to the chip – you lose important system-specific information. Your computer will probably still work but there will be some oddities.

Reading and Writing to SPI flash chips with Flashrom (Hardware Flashing)

Flashrom is open source software that can use a wide variety of SPI interface devices (eg internal SPI, CH341a, BusPirate) to interact with SPI flash storage devices. You can dump the contents of SPI flash devices and write to SPI flash devices.

We can use the SPI interface of a Raspberry Pi to flash Zx20 SPI chips as the Raspberry Pi exposes this SPI interface on the GPIO pins. We can dump and flash the SPI chip of the Zx20 as much as we want (up to ~100,000 times or so according to the datasheet) so even if your initial BIOS editing is flawed, you just hook the Raspberry Pi back up to the Zx20 and try again.

Software Flashing the Boot Block with Intel FPT

Running operating systems on the Zx20 are actually able to ‘see’ the SPI flash chip the BIOS is stored on, however these days key regions of the BIOS are usually blocked from being written to and even read from.

However, on Zx20 systems, a couple of motherboard headers (E14 BB and E1 ME/AMT FDO) allow you to disable these protections. Then using Intel Flash Programming Tool (FPT) you can write to wherever you want from Windows or Linux or FreeDOS.

Warning: You get one shot with this approach – if you mess up your editing of the dumped BIOS, you have probably broken your boot block and more. Your system will no longer boot, not even to BIOS. Your only recourse is to restore your SPI chip contents with programming hardware, the process that most of this guide is dedicated to.

I have included instructions for how to do this in the “Additional Methods” section of the guide.

If you are prepared to buy the programming hardware anyway, then trying the software method could be considered a shortcut to a 2013 boot block for those who can accept the risks.

If you don’t want to buy or learn how to use the programming hardware, I highly discourage trying the software method unless you like bricked motherboards.

Zx20 PCB FAB REV and Board Revision

I’m not sure of the exact distinction, but each motherboard has two revisions: PCB FAB and just ‘Rev’. PCB FAB is printed on the motherboard, ‘Rev’ is printed in the bottom right corner of a sticker on the motherboard.

PCB FAB REV locations:

- Z820: You can see the board revision to the right of the PCIe slots and left/above the motherboard heatsink, under the silkscreened HP logo
- Z620: Printed between the bottom two PCIe connectors, directly left of the CMOS battery
- Z420: Same as Z620

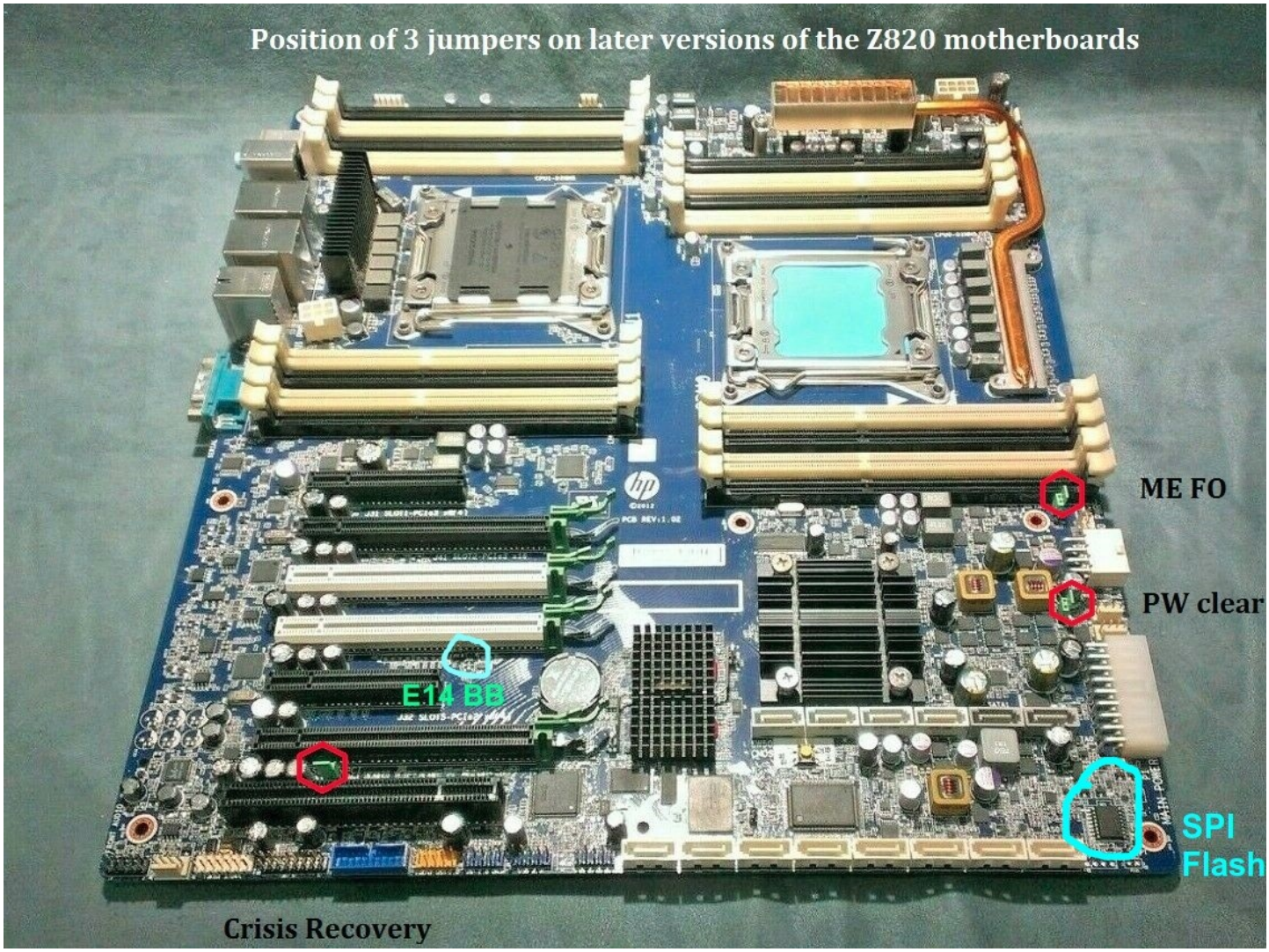
Motherboard revision locations:

- Z820: The sticker is underneath the PCB REV, right of the 2nd PCIe slot
- Z620: The sticker is right of the PCB FAB REV, left of the CMOS battery
- Z420: Same as Z620

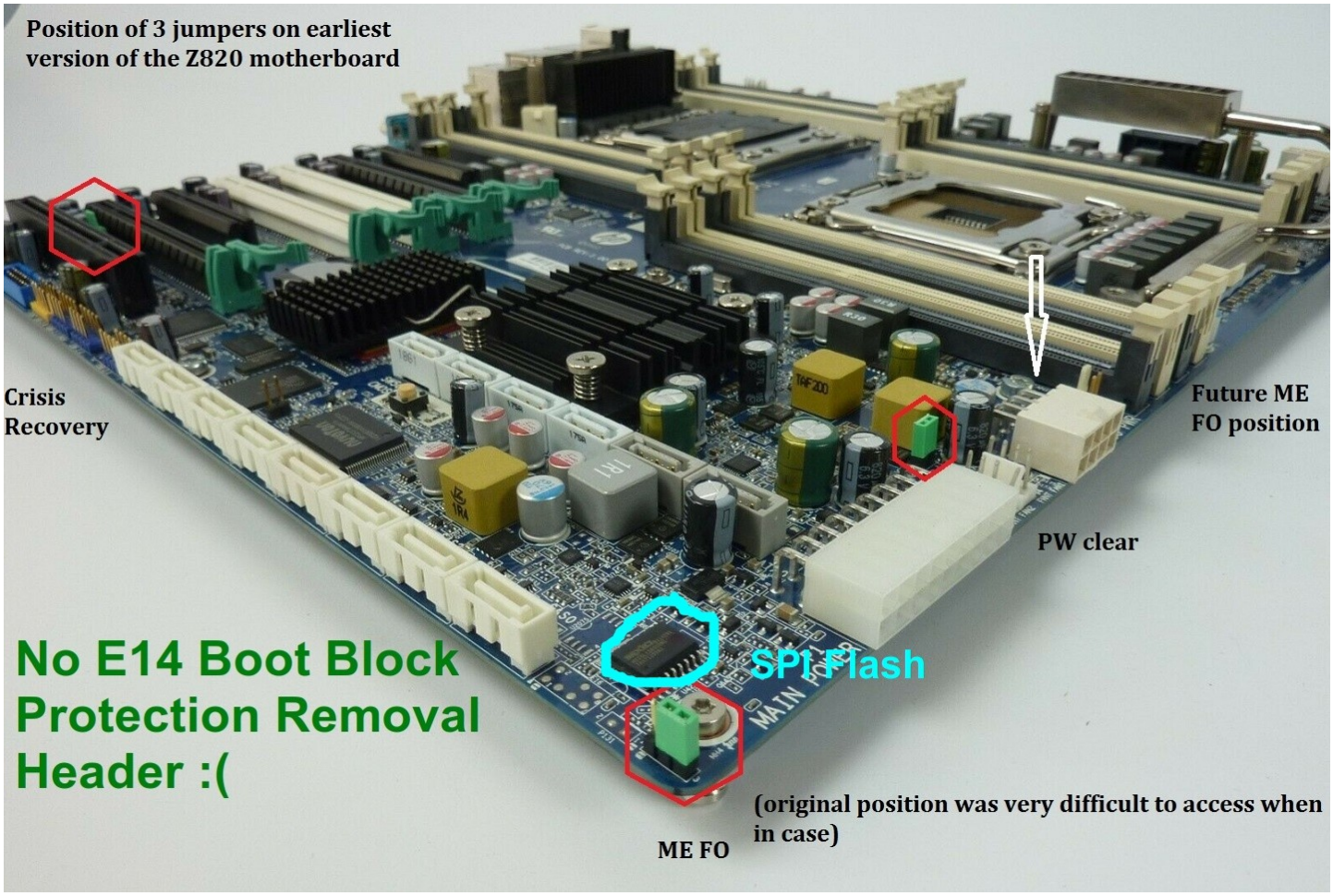
There are some reports on the forums that Z820 ‘Rev 1.00’ boards will not work (not defined exactly – I assume it means v2 CPUs can’t be used, even with boot block upgrade).

Zx20 Motherboard Jumper and SPI chip Locations

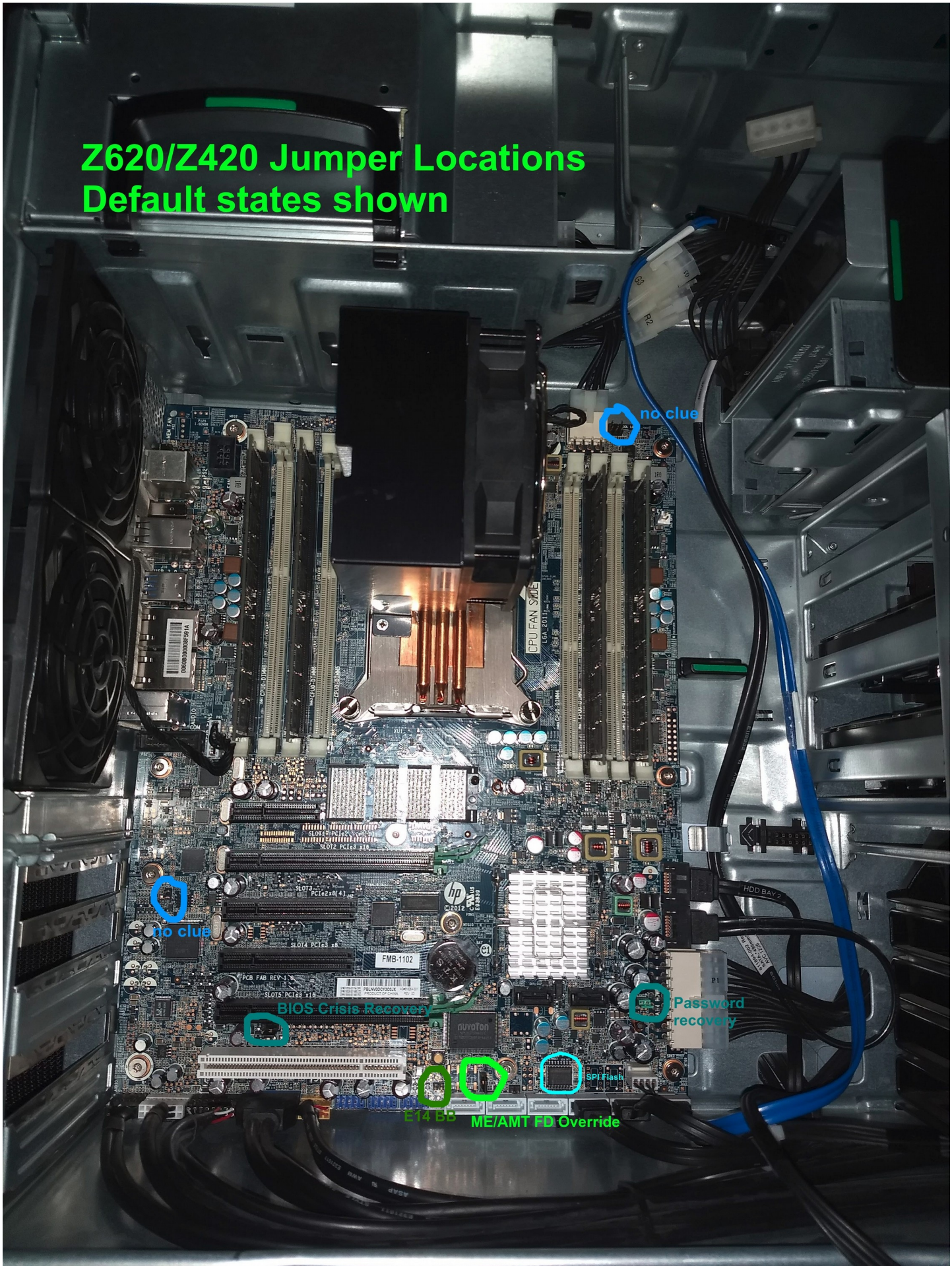
Z820 (edited from source)



Z820 (early revisions) (edited from source)



Z620 / Z420


Z620/Z420 Jumper Locations
Default states shown
no clue
no clue
BIOS Crisis Recovery
Password recovery
SPI Flash
E14 BB
ME/AMT FD Override

Requirements

Mandatory

- A Z420/Z620/Z820, of course. Version 1 with 2011 boot block date.
  Later board revisions seem more likely to work but no conclusive evidence exists.
- Disable Intel AMT in the BIOS – we want the Intel ME out of the way as much as possible.
- Find your BIOS flash chip on your board and take down its exact model number. Find the data sheet of the flash chip to double check the pinout and voltage levels.
- Record system-specific information that could be lost to a bad flash
  - your MAC addresses for normal NIC and AMT NIC (BIOS->File->System Info)
  - Serial number, asset tag (BIOS-File->System Info)
  - System UUID (BIOS->Security->System IDs)
- SOP16/SOIC16 breakout cable with clamp (eg this, there is also a more expensive but higher quality one made by Pomona)
- Raspberry Pi 3 (eg and an 8gb+ SD card), or other way of flashing the SPI chip (eg CH341a, BusPirate).
  - The Raspberry Pi A+, B+, 2, 3, and 4 should all work. Zero and Zero W I am not as sure but should also be fine. Original Pi A/B will probably need an external 3.3v supply if the chip is still soldered to the board. Use a good PSU with the RPi that is known to provide sufficient current.
  - For the CH341a, try to avoid the black PCB one as it puts out 5V on the logic pins. The SPI chip (3.3v logic) can deal with it but it's better to avoid it. The green one seems to be fine, see this picture at the end of this guide.
  - If you have a BusPirate, update it to the version 7 firmware.
- Multimeter for continuity testing of the clamp cable (or some way of figuring out what pin the corner of the clamp is so you know how the chip pinout is presented on the breakout board).
  - Also useful if you need to use a CH341a or external 3.3v source to make sure they actually put out 3.3v, as Chinese QC occasionally lets devices through that put out 7V on the 3.3v VCC.
  - If you don't own one and just want a cheap option to use, this one seems alright
- Various jumper cables – male-male, male-female, female-female such as from this pack. You can find these everywhere – amazon, electronics hobby stores, ebay, aliexpress, etc.
- A breadboard, eg this one.
- A USB flash drive for FreeDOS, at least 8GB. Choose a reliable one because you're going to be running important code from it. For example this one but many others are fine too.
- Flashrom installed in your Raspberry Pi

- UEFITool interactive BIOS info viewer – binaries available for Windows, must be compiled on Linux.
- Hex editor, you can use HxD (Windows) or Bless (Linux)
- v3.85 BIOS update files
  - Z420/Z620: ftp://ftp.hp.com/pub/softpaq/sp70001-70500/sp70077.tgz
  - Z820: https://ftp.hp.com/pub/softpaq/sp70001-70500/sp70078.tgz
  - If you need a way of opening .tgz, install 7zip
  - Tip: When looking for files like complete BIOS or ME firmware .bin, switch the OS on the HP drivers website to Linux – Linux. The downloads are archives with the files rather than .exe.
- Intel ME tools from winraid forums (v7 and v8)
  - "ME System Tools"
  - Has Intel Flash Programming Tool and several other useful tools
- Buccaneering and curious spirit
- Basic Windows and Linux experience, including using Linux command line
- The acceptance that while flashing BIOSes is a common and usually very safe process, things can go wrong and there's a small chance your Zx20 will not turn on again and you may need a new motherboard. BIOS chip, or to ask a friend very nicely to fix your problem.

Total Cost for Suggested Approach

Assuming you have none of the special equipment to start off with.

Item	**Cost (USD), pre-tax**
SOIC16 clip	6
Raspberry Pi 3 kit	50
32GB micro SD card for RPi	7
Multimeter with auto-range ohmeter and voltmeter	20
Jumper cables	3.35
Breadboard	1.2
Total:	**87.55**

The biggest part of this is the Raspberry Pi – but perhaps you can plan to use it for another project like PiHole, VPN into your home network, more electronics stuff, etc.

Other options for the programmer such as CH341a ($3) or BusPirate have worked for others but will likely require an external 3.3v supply, lifting the Vcc pin, or desoldering the whole SPI chip.

You also have a time cost of ~5-20h depending on experience to read this guide, order any necessary tools, and then to the disassembly, dump, edit, flash, test, reassembly, etc.

The value of this boot block upgrade project is about half being able to use the IB CPUs and 1866MHz RAM, and half what you learn along the way.

Suggested Environment

- Windows or Linux PC – can be laptop or desktop
  - Runs UEFITool
  - Runs the hex editor
  - Stores the dumps retrieved from the Raspberry Pi
  - Has the 2013 boot block update .bin
- Raspberry Pi
  - Runs flashrom to dump and write the BIOS
  - Some way of sending the dump and receiving the edited BIOS from the PC
  - If you don’t have a spare screen and keyboard/mouse for the Pi then you can SSH or VNC into the Pi – not covered in this guide.
  - SPI pins connected to the breakout board of the programmer, use the breadboard to split the 3.3v output to the multiple SPI pins 3.3v goes to.

Technically the whole thing could be done from the RPi, but we want to minimize power use so that the Pi’s 3.3v supply has as much current available as possible to counteract the passive drain of the motherboard.

Overall Plan

1. Update BIOS to v3.85 if it is below version 3.50.
2. Prepare Z620 for flashing – disconnect external IO and power, disconnect motherboard power connectors, remove PCIe cards.
3. Figure out pinout of clamp cable, connect Raspberry Pi SPI pins to clamp breakout board, clamp onto chip
4. Verify connection to SPI chip is good by seeing if flashrom can detect the SPI flash
5. Dump the SPI contents with flashrom several times and ensure identical hashes. Verify further by opening file in UEFITool.
6. Edit a copy of the dumped file with a hex editor to insert the 2013 boot block
7. Verify changes to edited BIOS file
8. Write edited BIOS to SPI chip
9. Re-assemble your Zx20, plug in monitor, keyboard, and power. Boot to BIOS and verify the boot block has been updated.
10. Update ME v7->v8 firmware if required

Update the BIOS

Minimum BIOS Version

The minimum BIOS version to use Ivy Bridge Xeons is 3.50. I recommend using version 3.85, as it is the version immediately preceding the hostile changes added in v3.88. Note that version 3.85 is before the spectre and meltdown mitigations, as well as various other security vulnerability fixes. For personal use this is a calculated risk but probably fine.

Note on BIOS versions >3.88

Version 3.88 “Adds a mechanism to detect unsupported combinations of processor and Management Engine firmware. In such cases, system will display an error message and fail to boot.”

This means that if you try to use a v2 CPU, but still have Management Engine v7 and not v8 firmware, the system will refuse to boot.

If you are already version 3.88 or later, and have ME firmware v7, then your options are

1. Ensure you update your ME firmware
2. Downgrade to 3.85

Both options are discussed later, having ME firmware v7 with the 2013 boot block does not pose an issue until you actually swap the v2 CPU in.

Updating the BIOS with FreeDOS

- Windows users can run DOSFlash in cmd or powershell directly, but it is not as reliable as flashing from FreeDOS or the tool within the BIOS
- Make a FreeDOS USB drive, for example with rufus.
- Then, make a folder on the USB drive called ‘Data’.
- Copy the ‘DOS Flash’ folder from the update to the ‘Data’ folder.
- Reboot and select the USB drive as a boot device
- Choose the 2nd option ‘FreeDOS Safe Mode’
- Navigate to the ‘DOS Flash’ directory:

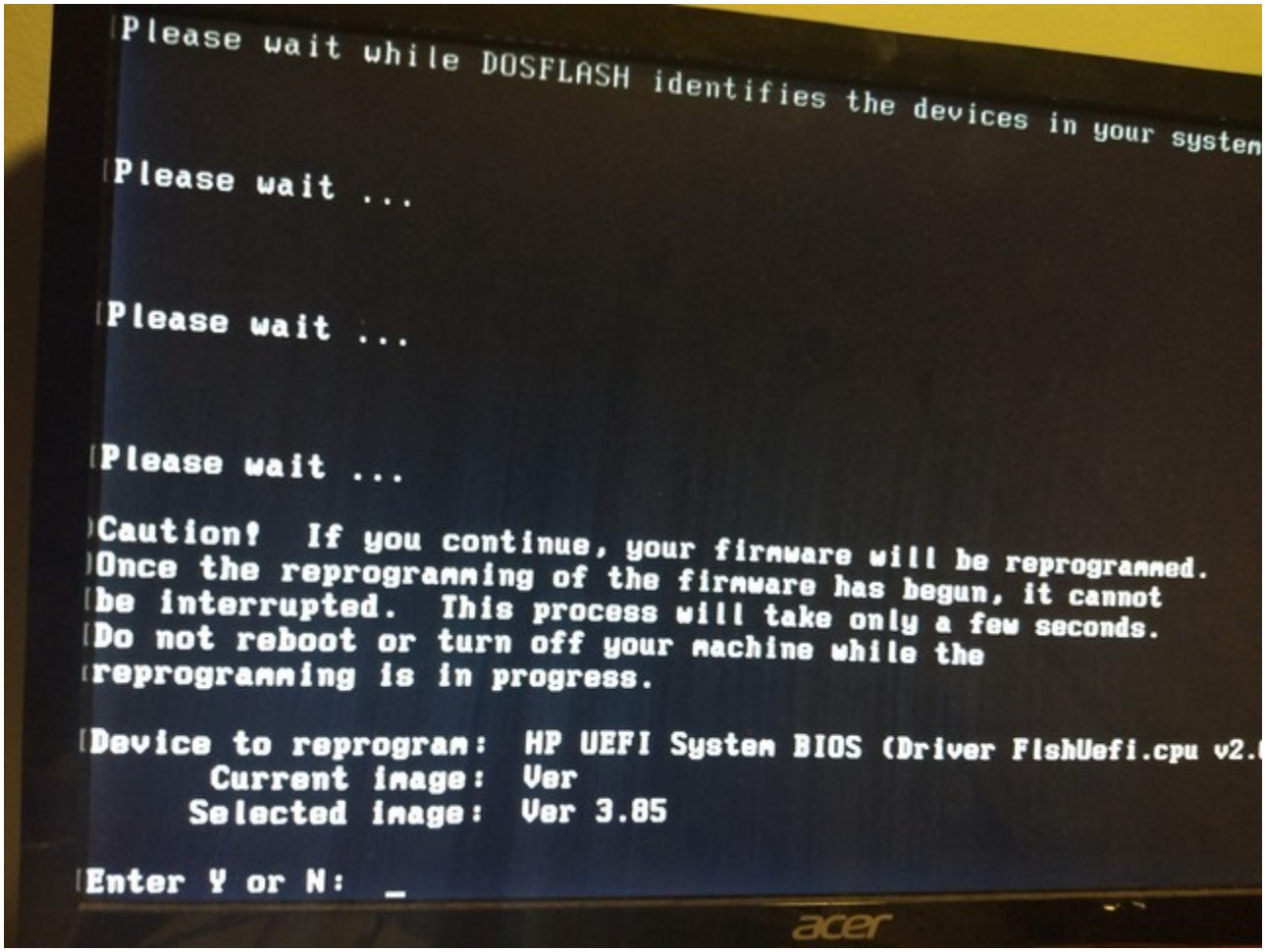
```
C:
cd Data
cd DOSFLA~1
```

- Run the update utility:

```
DOSFlash.exe
```

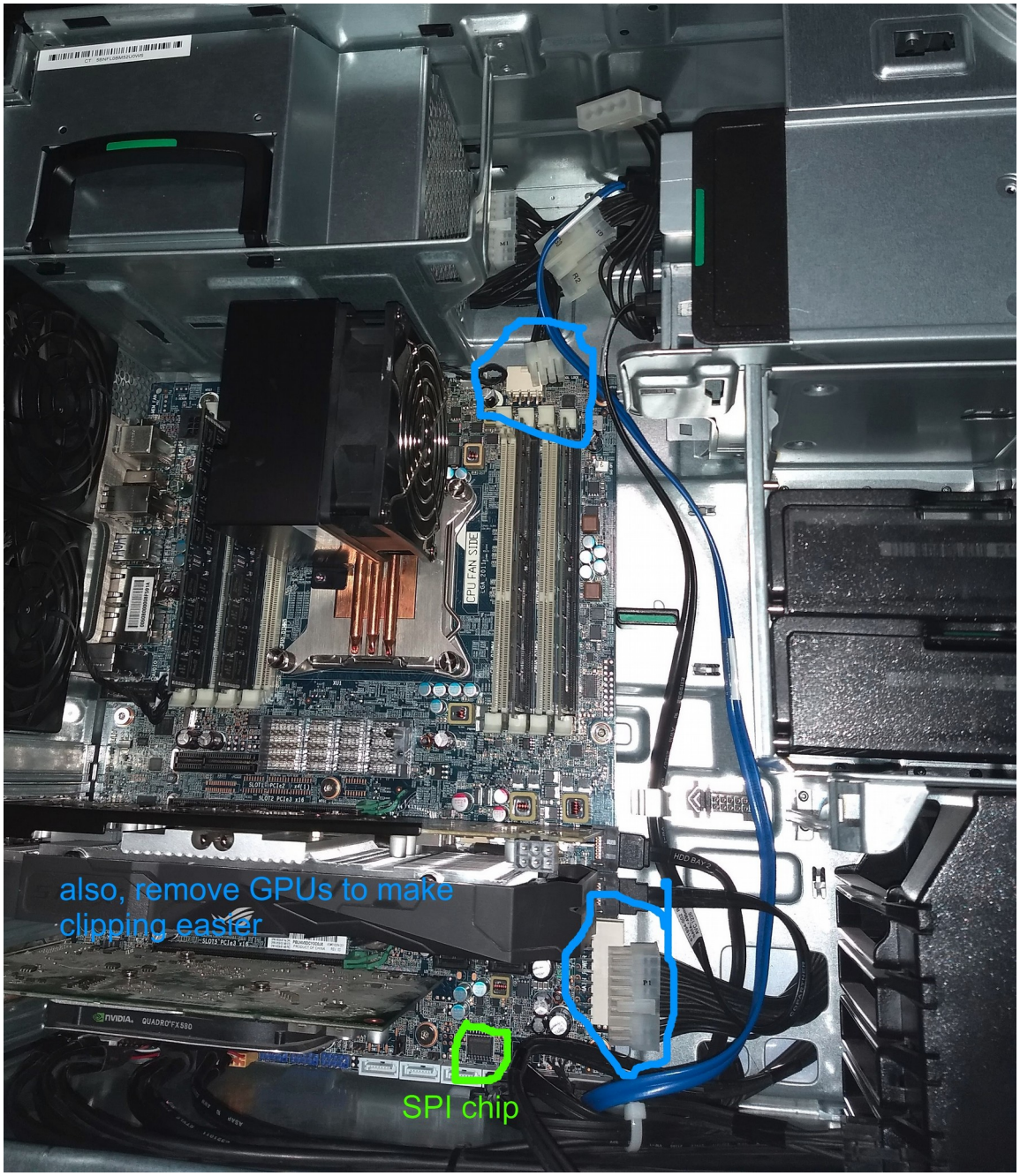
- Follow the prompts to install the v3.85 BIOS

- The process should look like this:



Preparing Z620 for BIOS Read and Flash

- Turn off system, **disconnect power and peripherals**, disconnect motherboard connectors, remove GPUs and other PCI(e) cards. Bring your Zx20 to somewhere where's it's convenient to have the programmer beside it, and to use your PC.

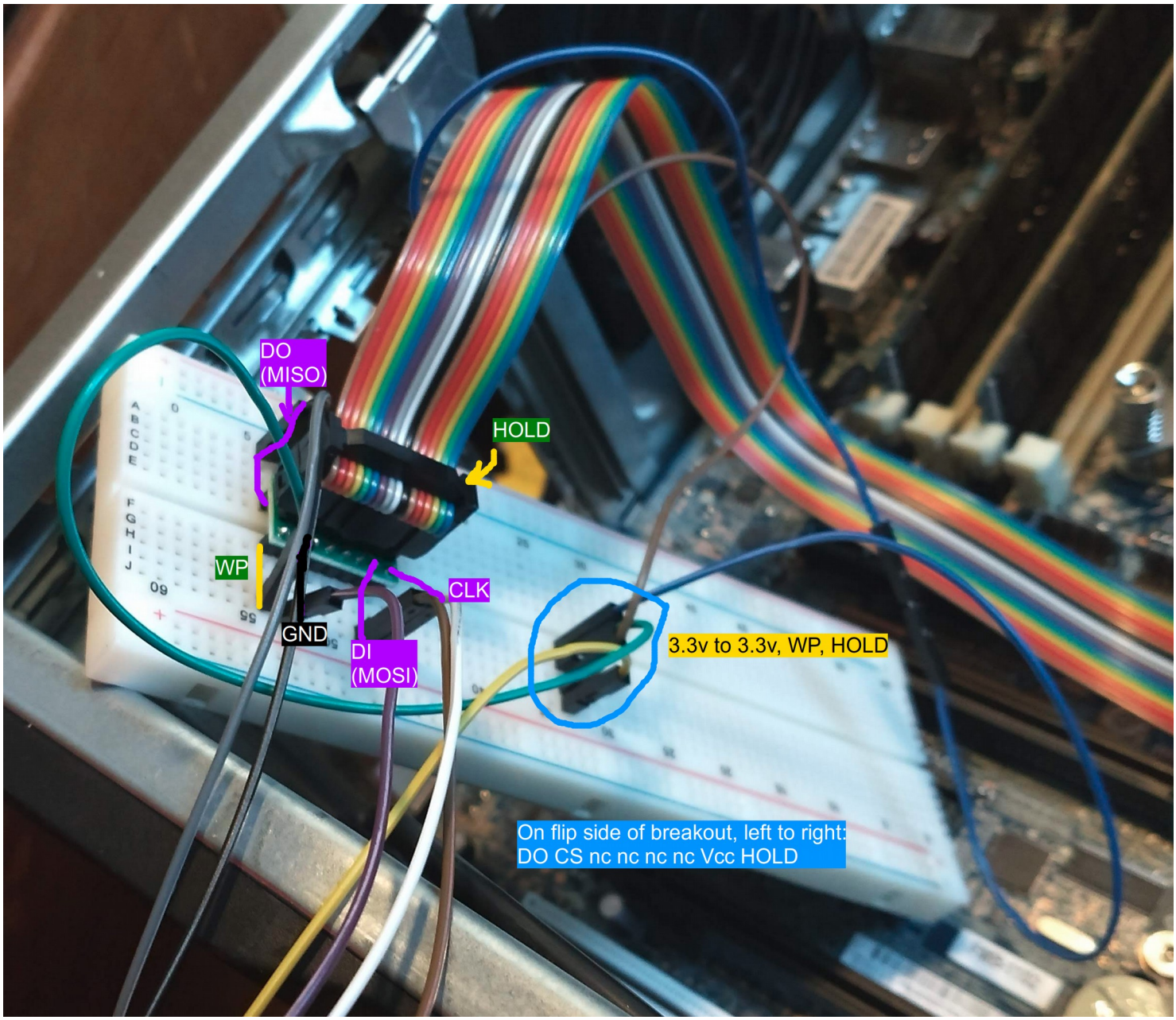


Connecting Programmer to SPI Chip

1. Figure out pinout of clamp cable

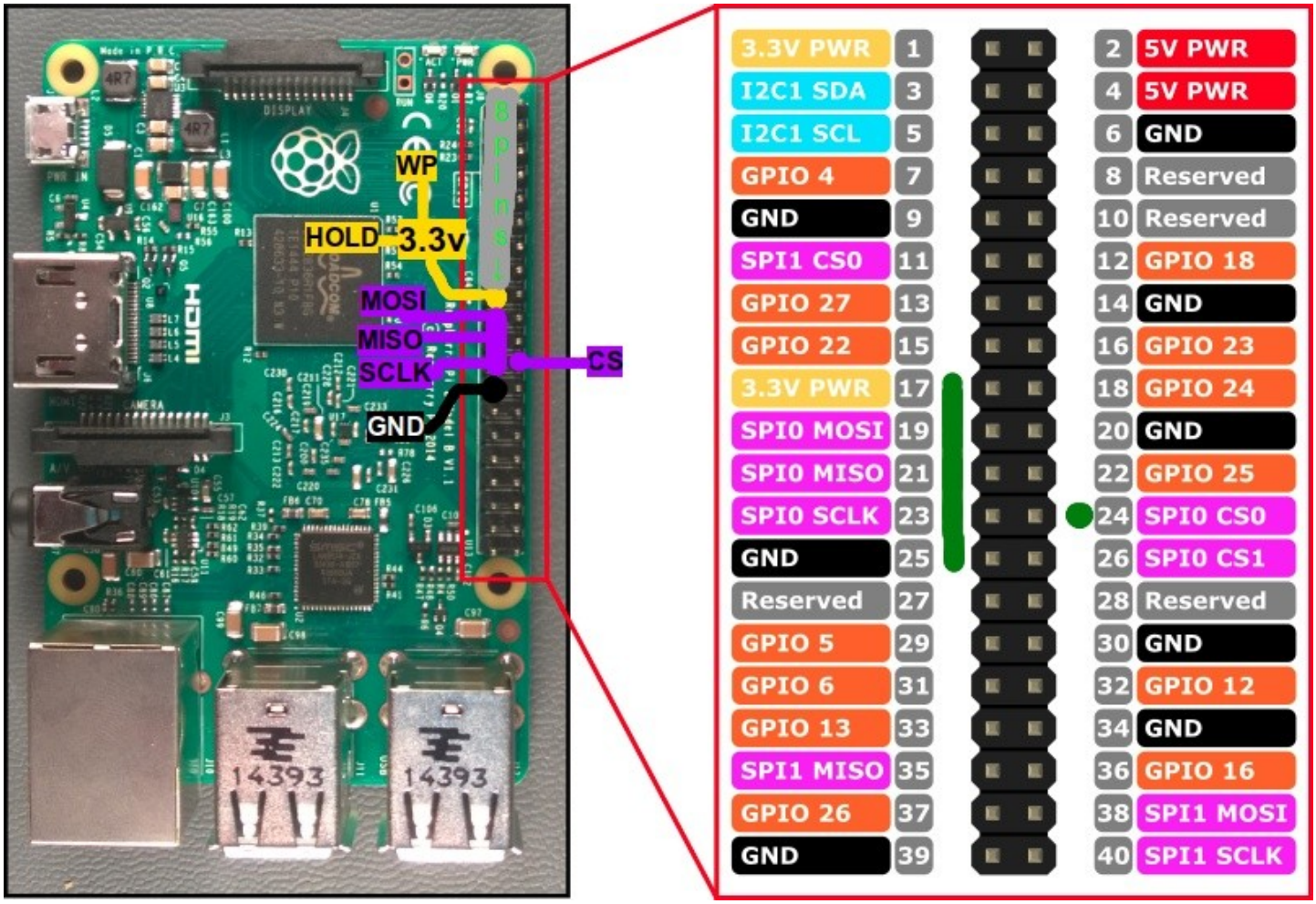
The typical SOIC16/SOP16 clamp cables are a wonderful rainbow with no way to tell where each clamped pin will be present on the breakout board.

Use a multimeter to measure the corner pins on the breakout board against a given corner of the clamp. Takes notes or make markings in some way to remind you which end of the clamp to put on the dot in the corner of the SPI chip, and what the pinout of the breakout board is.



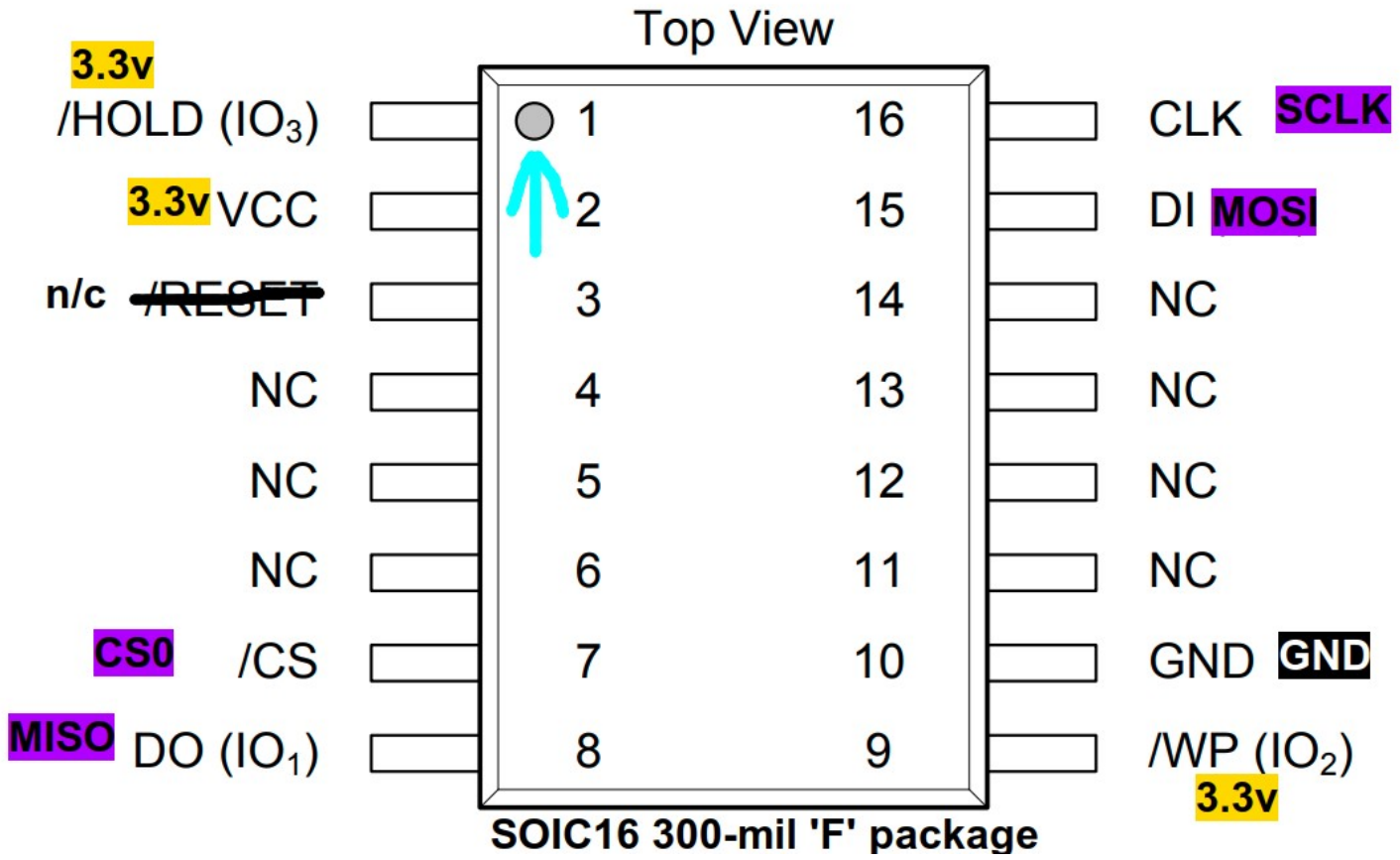
2. Connect Raspberry Pi SPI pins to appropriate pins on breakout board

Here's the what the Raspberry Pi pins are each for, and what SPI flash pins to connect them to:



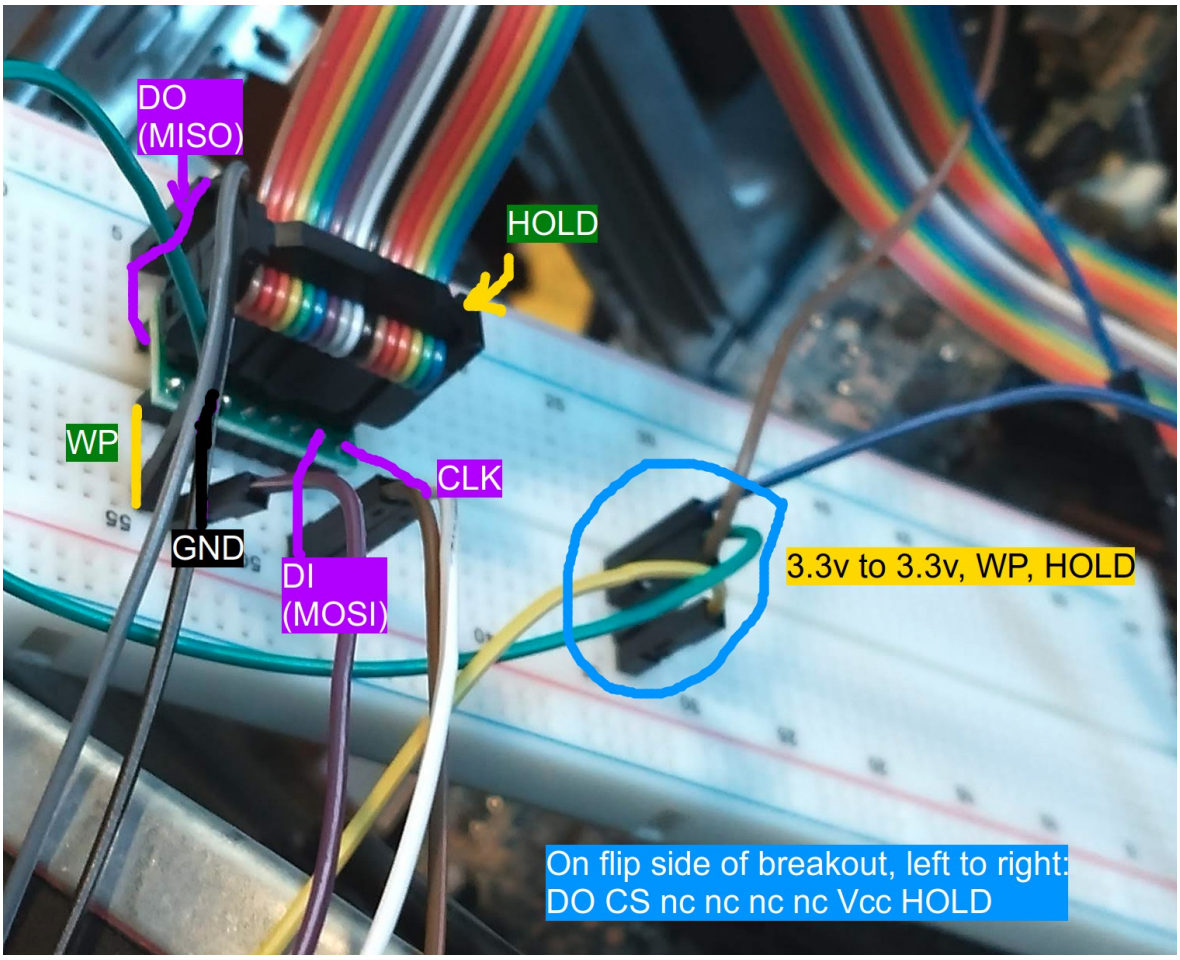
(edited from source)

The Raspberry Pi pins need to be connected to the corresponding pins on the SPI flash chip, as exposed through the breakout board of the clamp. Here is the pinout for the Winbond W25Q128FV, the chip I had on my board:



Note: You must look at the SPI chip on your motherboard and consult its datasheet if it is not a Winbond W25Q128BVFG!

Here's what connecting the clamp breakout to my Raspberry Pi looked like:



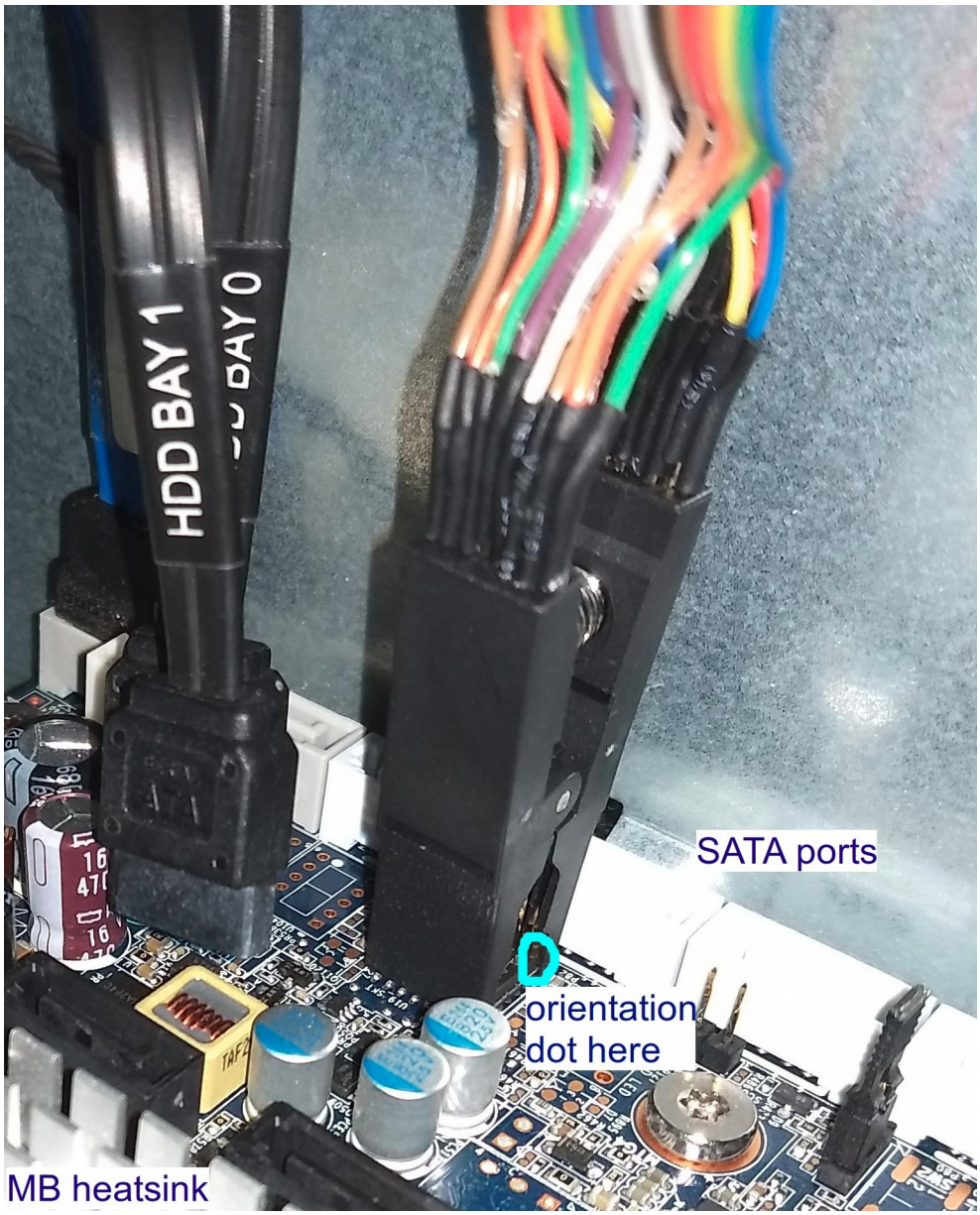
3. Clamp onto chip

This can be extremely frustrating to do and the only approach is to give it many good tries. Open the clamp up wide but don't force the spring. Align the corner of the clamp you chose with the circle corner indicator on the SPI flash chip. Push the clamp down onto the pins and then release your pinch on the clamp, it should clamp on firmly and upright to the pins.

Move on to the next section, but return here if you have trouble detecting the SPI flash chip.

Sometimes a pin in the clamp would seem to get stuck in a pushed up position and I would need to jiggle it a bit with a thin piece of wire (such as the male end of a jumper cable) to bring it back down.

Once you are sure of your wiring, any issues with detecting the chip and reading/writing to it are probably due to the clamp. It must have been my 20th clamp overall (including dealing with incorrect wiring) when everything finally went well.



Verify Connection to SPI chip

Run flashrom with only the argument specifying what programmer you want to use. Using the Raspberry Pi the programmer is the Pi's "internal" SPI (which is exposed on the GPIO pins you connected).

```
flashrom -p linux_spi:dev=/dev/spidev0.0,spispeed=10000
```

If your clamp and wiring are good, then you will see something like the following output:

```
flashrom  on Linux 4.19.75-v7+ (armv7l)
flashrom is free software, get the source code at https://flashrom.org

Using clock_gettime for delay loops (clk_id: 1, resolution: 1ns).
Found Winbond flash chip "W25Q128.V" (16384 kB, SPI) on linux_spi.
```

The flash chip is detected, matching the model number you found earlier and the 16MB BIOS size.

Dump SPI Chip Contents (your current BIOS)

Dump Flash Contents

If you can detect your flash chip, then dump its contents:

```
flashrom -p linux_spi:dev=/dev/spidev0.0,spispeed=10000 -r z620_bios_1.bin
```

Take the SHA256 hashsum:

```
sha256sum z620_bios_1.bin
```

Repeat this 3 more times and ensure the hashes are the same each time.

If any of the hashes differ, then your connection to the chip is somewhat flaky and you do not want to be writing a new BIOS with it. Check your wiring and reclamp onto the chip.

Make a backup of the dumped file once you have it hashing consistently. It's only 16MB so you may as well stick it somewhere else on your hard drive and in the cloud.

Verify Dumped File Contents

The first time I had a good clamp and could detect the SPI chip, I had completely random SHA sums that changed with every read of the SPI chip – it turned out I was reading mostly '1's with some random noise and definitely not getting a good dump of the BIOS. It's important to verify consistency across several dumps of the chip, and once consistent that the dumps are a fully valid BIOS.

The size of the dumped file should be exactly 16777216, you can check by running:

```
du -b your_bios_file.bin
```

In your linux environment.

Copy the dumped file to your windows environment.
Open it in UEFITool, it should display the sections of the BIOS. You will see one error about "invalid checksum 5Ah, should be AAh", this is normal.

Editing the 2013 boot block into your dumped BIOS

HxD Instructions (Windows)

- Make a copy of your dumped BIOS .bin, call it something like ‘z620_bios_edited.bin’
- Open the copy of your dumped .bin and the v3.85 update .bin (‘J61_0385.BIN’) in HxD in separate tabs
- Select the update .bin tab
- Do Edit->Select block and select 00FF0000 to 00FFFFFF
- Do Edit->Copy
- Switch to the tab for the bios_edited .bin
- Do Edit->Select block and select 00FF0000 to 00FFFFFF
- Do Edit->Paste
- Save the changes to the bios_edited file

Bless Hex Editor Instructions (Linux)

- Make a copy of your dumped BIOS .bin, call it something like ‘z620_bios_edited.bin’
- Open the copy of your dumped .bin and the v3.85 update .bin (‘J61_0385.BIN’) in Bless in separate tabs
- Select the the update .bin tab
- Do Edit->Select Range and select 0xFF0000 to 0xFFFFFF
- Do Edit->Copy
- Switch to the tab for the bios_edited .bin
- Do Edit->Select Range select 0xFF0000 to 0xFFFFFF
- Do Edit->Paste
- Save the changes to the bios_edited file

Congratulations, you just modified a BIOS file.

In theory we can also update the ME firmware at this stage, but for simplicity we will do it in software later.

Verifying your edited BIOS file

After making the changes:

- **verify the size is still 16777216 bytes**
- **and that the file opens in UEFITool.**
- **DO NOT FLASH if either check fails**

As a further check, check the sections of edited file against the originals.
Easiest to do in a Linux environment, Windows users could use the WSL or Cygwin.

Check the pre-bootblock section of the edited file against the original dump:

Dump pre-bootblock section of original dumped BIOS:

```
dd if=z620_dump_1_rpi3.bin of=original_bin_prebootblock.bin skip=0
count=16711680 iflag=skip_bytes,count_bytes
```

Dump pre-bootblock section of edited BIOS:

```
dd if=z620_dump_1_rpi3_edited.bin of=edited_bin_prebootblock.bin skip=0
count=16711680 iflag=skip_bytes,count_bytes
```

Compare SHA sums of pre-bootblock section:

```
sha256sum original_bin_prebootblock.bin;
sha256sum edited_bin_prebootblock.bin;
```

The SHA sums must match exactly, or you have made a mistake in the hex editor

Check that the boot block of the edited file matches the 2013 boot block from the update:

Dump bootblock section of BIOS update file:

```
dd if=J61_0385.BIN of=update_bin_bootblock.bin skip=16711680 count=65535
iflag=skip_bytes,count_bytes
```

Dump bootblock section of edited BIOS:

```
dd if=z620_dump_1_rpi3_edited.bin of=edited_bin_bootblock.bin
skip=16711680 count=65535 iflag=skip_bytes,count_bytes
```

Compare SHA sums of pre-bootblock section:

```
sha256sum update_bin_bootblock.bin;
sha256sum edited_bin_bootblock.bin;
```

The SHA sums must match exactly, or you have made a mistake in the hex editor

Write new BIOS to SPI chip

Before flashing, dump the SPI contents and check the sha256sum again to make sure your clamp connection is still good.

Write your edited BIOS to the SPI flash:

```
flashrom -p linux_spi:dev=/dev/spidev0.0,spispeed=10000 -w
z620_dump_1_rpi3_edited.bin -V
```

Because of the -V option (verbose), a lot of output will scroll by. When the command has finished executing, you should see:

```
Erase/write done.
Verifying flash... VERIFIED.
```

Do one more dump and sha256sum to ensure the edited bin file wrote correctly.

Verify boot block has been updated

Reassemble your workstation, remembering to reconnect the 2 motherboard power connectors. Turn the computer on – if all has gone well, it will turn on just as before - and go into the BIOS (F10).

Go to File->System Information, and you should now see the boot block date of ‘03/06/2013’

Upgrading ME firmware v7 -> v8 in OS

I am currently waiting for my test v2 CPU to arrive before upgrading my ME firmware, to see if BIOS v3.85 + ME v7 works and then if BIOS >= v3.88 + ME v7 works. If you are not adventurous, then wait a couple of months for me to try this and update the guide.
I will be using the below resources when I do upgrade:

Using intel FPT to flash ME, general instructions: https://h30434.www3.hp.com/t5/Business-PCs-Workstations-and-Point-of-Sale-Systems/Intel-Network-Card-and-Management-Engine-fail-to-initialize/m-p/6978635/highlight/true#M26184

Dan_WGBU on Z820 ME flash: https://h30434.www3.hp.com/t5/Desktops-Archive-Read-Only/Z820-ME-Firmware-amp-Management-is-Disabled-in-BIOS/m-p/5572326/highlight/true#M530575

someone that did a ME v7 -> v8 upgrade for a H61 chipset desktop:
https://www.win-raid.com/t4565f39-Little-help-with-update-H-chipset-to-ME-version-for-Ivy-bridge-support.html

The most direct approach appears to be using the ME firmware upgrade files from the HP provided ME firmware upgrade softpaqs, but using FWUpdLclMe8 directly, as described by SDH. Second most direct is to extract the ME region from an update .bin (BIOSes as low as v2.07 have ME v8, 3.85 has 8.1.57.1557), edit it into the dumped BIOS, and then flash it in FreeDOS or by SPI programmer.

Some possible options:

- Use HP update SP66702 (superseded by 82644 for windows, 82684 for linux/DOS) as described by SDH in this post. Move the AMT/ME header first.
  - Use FWUpdLclMe8.exe and the 8.1.72.3002.bin file, SDH reported that he was able to upgrade a ME v7 system to v8
  - I recommend doing this in FreeDOS and not windows (especially not windows 10) to minimize the chance of something going wrong
- use UEFITool to replace the ME region of dumped .bin with v8. Boot into FreeDOS and use Intel FPT to flash the edited BIOS.
  - May want to then use fpt -greset which initiates a system hard reset, needed for the ME to initialize on the new version properly (?)
  - Can also use programmer instead of fpt, although the risk is lower here
  - This may require some changes to the descriptor region too as described in the winraid H61 ME upgrade post
- Use ME/TXE injector as described here. This may not be capable v7->v8 upgrades.
  It results in the new ME region being flashed using command:

```
fptw64.exe -ME -F new_me.bin
```

- Use fwupdate.exe with generic intel ME update image? Fwupdate is a tool that upgrades the ME firmware area using special update images. I am not sure if Zx20 are compatible with this method.
  One would get the v8 firmware update bin from section B1 here and then use fwupdate to flash it, then do a fpt -greset after
- Perhaps we could me_cleaner the BIOS and avoid having to mess around with the ME altogether?

Special Case Considerations

1. My BIOS is at version 3.88 or greater. How should I upgrade the boot block?
   - Dump as usual and then grab the 2013 boot block from the bin file of your update files of your BIOS version.
2. How can I update the ME firmware (from 7 to 8) if I run into issues? (most relevant for BIOS v3.88 and up)
   - Options are to use intel fptw64 and the E1 AMT/ME FDO header to flash ME v8, or edit the dumped BIOS and replace the ME7 firmware region with ME8
3. My SPI chip is not winbond 25Q128BV
   - Look up the datasheet for your chip and find the pinout – the pins should be the same even if their position in the SOP16 package is different. Also, read the datasheet to find out if WP and HOLD should be held at 3.3v or GND.
4. My ME is in manufacturing mode now and the boot message bothers me
   - See Agrawitt's post on flashing the ME firmware
5. My workstation won't boot anymore!
   - Reduce yourself to 1 RAM stick, 1 CPU, 1 graphics card and no HDDs
   - Ensure you connected everything correctly internally – motherboard power, fan connectors, GPU power
   - If you think you messed up the BIOS editing, flash your original dump back
   - Clearly record what you did and ask for help

Alternative Methods for Various Stages

Dumping/Flashing entirety of SPI chip using software

Note: Tested on Z620, but only to dump the SPI contents and write them back

On later board revisions, both the E1 (ME/AMT flash override) and E14 (boot block protection) headers are available on the motherboard. By changing the position of the jumper on E1 and placing a jumper on the 2-pins of E14, the boot block and ME regions are no longer protected. It should be possible to use the Intel flash programming tool to dump the original BIOS and write the modified BIOS.

The ‘earliest’ Z820 board revisions do not have the E14 header. Not sure for Z420/Z620 – my PCB1.0 / Rev 0D Z620 has an E14 header left of the AMT/ME FDO header.

Reading BIOS with Intel Flash Programming Tool:

- Download “Intel ME System Tools v8” from win-raid intel tools post. Put the folders in the “Data” folder of your FreeDOS USB drive.
- Move E1 (ME/AMT FDO override) header to occupy the bottom 2 pins and bridge E14 (boot block protection removal) header.
- Reboot into FreeDOS, select safe mode, C:, navigate to “FLASHP~1” folder.
- Run the dump command:

```
FPT.EXE -d BACKUP.BIN
```

- For me the command took 1m15s to run to completion, at the end displaying “FPT Operation Passed”.

Writing BIOS with Intel Flash Programming Tool:

- Follow first 2 steps of ‘Reading BIOS with Intel Flash Programming Tool’.
- Put the BIOS file you wish to write onto the USB drive, most convenient is to put it within the same folder as fpt.exe.
- Reboot into FreeDOS, select safe mode, C:, navigate to “FLASHP~1” folder.
- Run the write command:

```
FPT.EXE -F hp_bios_edited.bin
```

- You will probably get a prompt about “PDR region exists in flash”. This is the Platform Descriptor Register which contains some system specific information, if the BIOS you are flashing is based on a dump from your system then you shouldn’t lose any information in the PDR.
- For me the command took 32s to run to completion, at the end displaying “FPT Operation Passed”.

Partial or full desoldering of SPI chip

If you are good at SMD soldering, it will probably be easier to desolder the chip, do the dump, edits, and flash, and then solder it back on. I believe the right way to do this is with a temperature controlled hot air gun and flux. You could even order an identical SPI flash chip so that you don't need to flash to the original.

Another option, to deal with the passive drain of the board on the 3.3v supply of the programmer, is to desolder and lift the Vcc pin of the flash chip up from the board. Then, put an insulator between the lifted pin and the pad, and clamp on normally. One user on the HP forums said he needed to do this for 12 of the 14 Z820 that he flashed, and that it was simpler to desolder all the pins on the same side of the chip as the Vcc pin. See the "z820 e5-2600 v2 ivy bridge upgrade" thread for more details.

External 3.3v supply to Vcc to account for passive drain

Another way to deal with the 3.3v drain is to use a strong enough 3.3v power supply that the drain is not important. These are:

- The 3.3v supply of ATX PSUs, these have been used going back years.
- There are also 3.3v supplies with decent amperage from China such as the AMS1117. Check the voltage with a multimeter because some units will have flaws and do something like put out 7v on the 3.3v pin.
- If you are lucky then you have a controllable voltage and current benchtop power supply :)

Modifying the update bin rather than the dumped BIOS and flashing the modified update bin

HP forums user and Z workstation expert mtothaj has suggested that it's easier to record your MAC address / serial / asset tag and program that into the update .bin, than copy the boot block and potentially ME firmware to the dumped BIOS.

To me it seems to be about equal effort, and it's a very good idea to get a backup of the original contents of the SPI chip anyway.

Editing ME v8 firmware into the dumped BIOS file

Note: Not tested yet

< ... todo ... extract from BIOS update, edit into dumped BIOS. Same thing as using UEFITool >

Using IC hook grabbers to connect to the SPI flash chip pins

Note: Not tested yet

IC pigtail hook grabbers let you hook around the soldered pin. For example these ones. They are a bit more expensive than the clamp and I don't know what difference they make – supposedly they are easier to use.

Buying some spare SPI flash chips and a SOIC16->DIP16 adapter to be able to practice flashing

Note: Not tested yet

If you want to practice setting up the wiring, reading and flashing chips, and possibly even clipping on; you can get the exact model of flash chip (eg winbond 25q128bvfg 5pcs ~$7.5) and a SOIC16->DIP16 adapter.

You could also get a SOIC16->DIP16 PCB adapter (and some headers) and solder one of your spare SPI chips if you want to practice clamping to a soldered chip, and you have the skill for SMD soldering, but don't want to take an iron to your board.

Compiling UEFITool on Linux

< ... todo ... mostly involves knowing what dependencies to install >

Future Work

1. Find out to what degree using ME v7 with 2013 boot block and Xeon E5 v2 CPU is actually an issue.
   Some forum posts warn vaguely of “big incompatibilities”, and it seems many people report that Sandy Bridge boards upgraded to Ivy Bridge firmware still won’t work without the ME v8 firmware (I mainly read this about laptops). However, in the forum posts about the Z820 boot block flashing, people reported success without mentioning upgrading the ME.
2. Figure out what all the jumpers on the Z620 / Z420 / Z820 motherboard jumpers do and document it publicly. (see the ones I labelled ‘no clue’ in the Z620 diagram)
3. Find out if a post on a Chinese developer website by user “baobaomao” has any extra information (https://download.csdn.net/download/baobaomao/11099679). I couldn’t get their login system to accept any third party logins (even from other Chinese services like weibo or wechat) or let me make an account. I’m guessing it’s just going to be using a programmer to write the boot block region, but maybe there’s some unique insights.
4. Find out if the recommendation should be followed that resistors are put in front of the data lines and a capacitor put between Vcc and GND, between the Raspberry Pi pins and the SPI chip. As described in this BIOS flashing guide (https://www.win-raid.com/t58f16-Guide-Recover-from-failed-BIOS-flash-using-Raspberry-PI.html) and the winbond reference schematic (https://www.winbond.com/resource-files/an0000026%20reference%20schematic%20for%20winbond%20spi%20nor%20flash%20v1.1%2003112019.pdf). As far as I can tell, these are good best practices, and can mitigate certain issues (see winbond pdf), but not at all required.
5. Automate the editing of the dumped BIOS files / update BIN
   - Extract important info like MAC addresses
   - Do some sanity checks on the dumped file
   - Create the file that needs to be flashed back automatically (patch in boot block, ME firmware, addresses)
   - Python or shell script, simple javascript website?
6. Look into whether UEFITool can do the boot block replacement for us, given the two files (no hex editing). The ‘extract’ and ‘replace’ features seem to be able to do the ME region, but the boot block is within the BIOS region.
7. Explore the microcode in the boot block to find out exactly which CPUs are supported (eg will those OEM-only or ES/QS CPUs work).

Various Notes

ME firmware upgrades

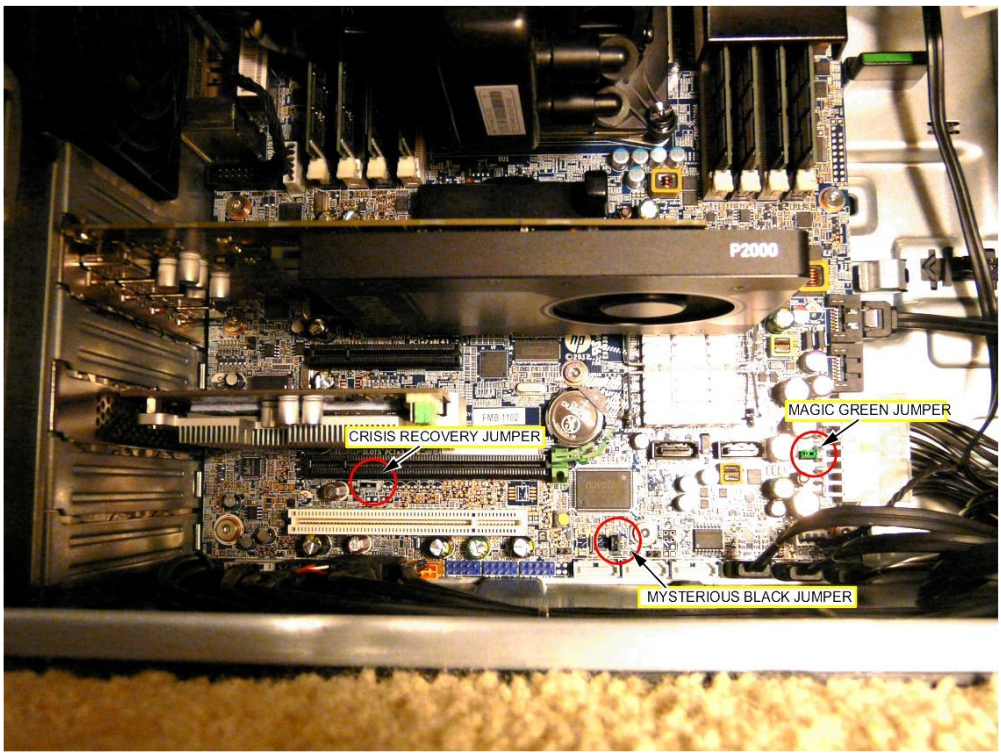
One post in the “Z820 ivy bridge upgrade” thread on HP forums mentions “**Switching the E14 jumper position (earlier post by Pabla) AND installing the E1 jumper does write enable the BIOS flash but only that portion from 0x000000 to 0x50FFFF**. That section is where the Intel ME/ATM code (and other non-BIOS data) resides. The BIOS flash addresses above that range remain write protected. That is where we hit a brick wall since the bootblock resides starting at 0xFF0000.” I’m not sure where the E14/E1 jumpers for Zx20 are as I couldn’t find a board picture that was high res enough, when I have it apart again I will hunt for them on my Z620. However based on the discussion in the thread I would guess E1 is the ME/AMT flash override jumper. E1 seems to also be called FDO (Flash Descriptor Override) per this discussion about the Z200. So perhaps it unlocks changes to the flash descriptor and also to the ME region.

Dan_WBGU mentioned in the ‘z820 ivy bridge upgrade’ thread that E14 is not present on some earlier Z820 (and Z420/620?) boards (-001) as there was no room in the crowded PCB at the time, test pad J144 was removed to make room in later board revisions. I assume this applies to Z420/Z620 too.

Dan_WBGU himself wrote a short guide on how to flash ME firmware on Z820.

Motherboard Jumpers

Forum user BambiBoomZ made the following image of the **Z620** motherboard after removing the lime green jumper on the advice of an HP tech to somehow get around BIOS version downgrade prevention:



My speculation/research:

“Mysterious Black” (3 pin) Seems to be component 36 in the service manual. Listed as “ME/AMT Flash Override”! Could be very useful. I would guess that moving this header to the bottom 2 pins may allow use of fptw64.exe to flash the v8 ME firmware to the SPI chip from within the computer. Great discussion on updating Zx20 ME here.

I also noticed 2 unpopulated 2-pin headers, one on either side of the “Mysterious Black” jumper. I will thus call these mysterious black jumpers “left”, “centre”, “right”.

Left: (2pin) Not listed in service manual. It’s right under the clear CMOS button. Seems to be E14 BB.

Centre: (3pin, top 2 pins bridged by default) The ME/AMT Flash Override.

Right: (2pin) Listed as component 32, “Hard Disk Drive LED”. It doesn’t seem to be plugged in. Maybe HP planned to use it and changed their minds?

There’s one more mysterious 2-pin black header, off to the left of the 3rd PCIe slot. Not listed in the service manual.

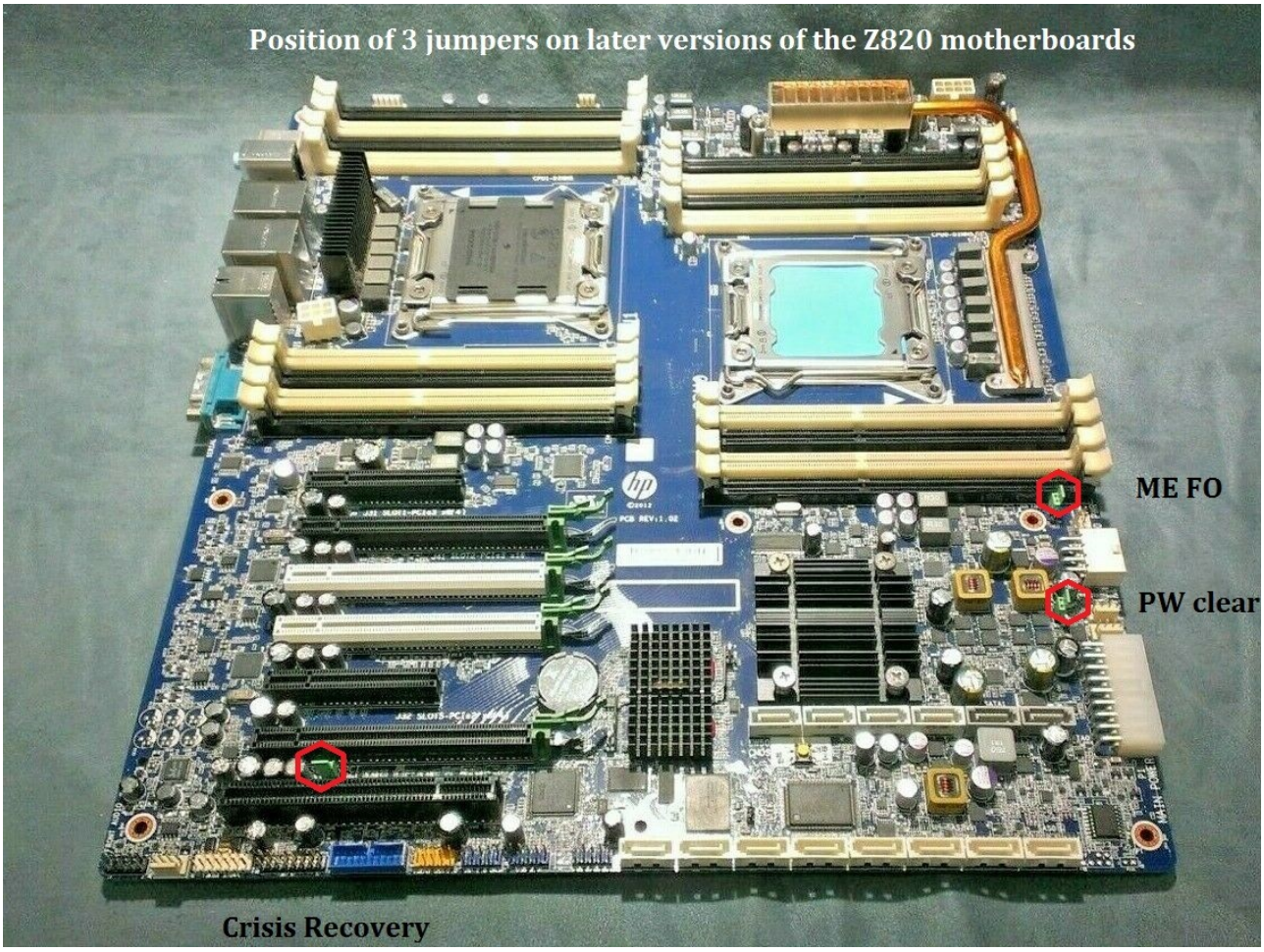
“Magic Green” (2 pin), in addition to being the “Password recovery” jumper (reset BIOS password when removed), can also disable BIOS downgrade prevention and (100% speculation) perhaps disables checks against modifying the BIOS in general. Listed as component 37 in the service manual.

Note that the crisis recovery jumper is also sometimes called “boot block recovery”. The boot block is used to flash a BIOS, it does not allow for flashing of the boot block.

The service manual says E14 on Z420/Z620 is the “ROM bootblock header”, with E15 being “ROM bootblock recovery header”, and E16 “ROM Recovery header/jumper”. So if E15 is the crisis recovery header inbetween the PCI slots, perhaps E14 can disable boot block protection in software?

Mtothaj posted “In the meantime I have experimented with the E14 header on the Z420 motherboard, referenced in the manual as 'ROM Bootblock Header'. I can confirm that **when a jumper cap is placed on this header, protection is removed from the bios boot block range (FF0000 - FFFFF)** as reported by the program prr2.exe”.

I found this diagram of the Z820 jumpers on a thread about upgrading the ME version to make a network card work after a BIOS update (for an HP 8200, later posts about Zx20):



There is also a trick where if you short pins (HDA_SDO) and (DVDD (3.3v)) of the HD Audio chip at boot, then some BIOS restrictions are removed. On some motherboards of some manufacturers this trick can allow flashing of the boot block.

Additional Resources

winbond 25Q128BV SPI flash chip data sheet:
https://www.winbond.com/resource-files/w25q128bv_revh_100313_wo_automotive.pdf

win-raid forums guide on programming SPI flash chips: https://www.win-raid.com/t4287f16-GUIDE-The-Beginners-Guide-to-Using-a-CH-A-SPI-Programmer-Flasher-With-Pictures.html

win-raid forums guide on “Intel Management Engine: Drivers, Firmware & System Tools”:
https://www.win-raid.com/t596f39-Intel-Management-Engine-Drivers-Firmware-amp-System-Tools.html

win-raid forums on “Unlock Intel Flash Descriptor Read/Write Access Permissions for SPI Servicing” (using HDA_SDO pinmod or **FDO/AMT Unlock**): https://www.win-raid.com/t3553f39-Guide-Unlock-Intel-Flash-Descriptor-Read-Write-Access-Permissions-for-SPI-Servicing.html

CHEF-KOCH on “HowTo flash Intel Management Engine (ME) firmware”: https://github.com/CHEF-KOCH/HowTo-flash-Intel-Management-Engine-ME-firmware

HP forums thread on this process for Z820 (in particular pages 5 to 10) “z820 e5-2600 v2 ivy bridge upgrade”: https://h30434.www3.hp.com/t5/Desktops-Archive-Read-Only/z820-e5-2600-v2-ivy-bridge-upgrade/td-p/5086052/page/5

HP forums BambiBoomZ “HP zX20 BIOS v 3.92 > Successful roll back to v 3.91” (the motherboard jumpers are discussed): https://h30434.www3.hp.com/t5/Business-PCs-Workstations-and-Point-of-Sale-Systems/HP-zX20-BIOS-v-3-92-gt-Successful-roll-back-to-v-3-91/td-p/6737848

HP forums Dan_WBGU tells you how to flash the ME on Zx20 “Z820 ME Firmware & Management is Disabled in BIOS”: https://h30434.www3.hp.com/t5/Desktops-Archive-Read-Only/Z820-ME-Firmware-amp-Management-is-Disabled-in-BIOS/td-p/5566025

Voodai’s Journal on “How to force a downgrade of the BIOS on a HP Z Workstation”

(specifically Z820 but should be broadly applicable): http://voodai.blogspot.com/2017/01/how-to-force-downgrade-of-bios-on-hp-z.html

stackexchange question on diagnosing issues with reading SPI flash:
https://electronics.stackexchange.com/questions/338633/can-not-interface-with-winbond-w25q16dvsig-over-spi

Acknowledgements

None of this would be possible without the posts by members of the HP forums, winraid forums, libreboot community, **and many others**. Specific thanks to:

- SDH (HP forums user)
- SalSimp (HP forums user)
- Dan_WBGU (HP employee and forums user)
- mthothaj (HP and other forums user)
- Agrawitt (HP forums user)
- Scoobis (HP forums user)
- Plutomaniac (winraid forums poster of many detailed guides)
- level1techs forum (discussion and advice on flashing SPI chips)
- Whoever decided the RPi3 should have a great 3.3v supply
- The inventor of the SOIC16 clamp
- HP, for providing full BIOS files and not being mean about the discussion on their forums