Board:lenovo/x60/Installation

From coreboot

These Coreboot/Libreboot flashing instructions are designed for the **Lenovo X60**, **X60s**, **X60 tablet**, **T60** and **T60p**.

Warning: The **ThinkPad T60**/T60p Series must be reconfigured to work with **Libreboot**. If you have an ATI GPU, *you must replace it* with an Intel motherboard. (http://libreboot.org/docs/hcl/index.html#t60_ati_intel) You will most likely need to swap the screen and/or swap the motherboard, in mast cases. Follow the steps in the Lenovo HMM (http://support.lenovo.com/us/en/docs/migr-62733) to replace them.

Note: ThinkPad T60/T60p systems with an ATI GPU are compatible with **Coreboot** *as long as proprietary VGABIOS blobs are extracted and installed. Other T60 laptops are also compatible, with the same configuration in coreboot.* See this guide (https://github.com/bibanon/Coreboot-ThinkPads/wiki/ThinkPad-T60p) for more information.

Contents

Back up the original proprietary firmware

Warning: It is *STRONGLY RECOMMENDED* to create a backup of the vendor BIOS firmware; each vendor BIOS image has a unique, unrecoverable ID. Do not use another laptop's vendor BIOS image.

1. Download and extract the latest Libreboot binaries (http://www.libreboot.org/download/).
2. From the `libreboot_bin/` or `libreboot_util/` directory:
3. Run *both* of these commands to create a backup of the original firmware, creating a new file containing the data called `factory.bin` (don't panic, nothing is being installed):

```
sudo ./flashrom/i686/flashrom_lenovobios_sst -p internal -r factory.bin
sudo ./flashrom/i686/flashrom_lenovobios_macronix -p internal -r factory.bin
```

4. This will have created a file called "factory.bin", which is the contents of the flash chip while it had the original "BIOS" firmware. Back this up to a safe place (it can be useful for potential reverse engineering work, or for other testing).
5. The source code for this version of flashrom can also be found on the libreboot website. It is a patched flashrom executable, to work around security restrictions in the original Lenovo BIOS (it will also be used for flashing,

later on in this guide); the patches for it in libreboot can be found under resources/flashrom/patch/.
6. There are two flash chips for the X60/T60. The instructions above simply used flashrom executables for both; one failed, and one should have succeeded. This is much easier than finding out what flash chip you have before hand (which involved elaborate hacks in flashrom, reading output, or taking apart the laptop and physically looking at the chip).

Video BIOS (VGA option ROM)

For those systems with Intel graphics, native graphics initialization code exists in coreboot. However, if you want to use the Intel (proprietary) Video BIOS, you should extract this from the factory.bin dump that you created earlier. If your system has ATI graphics (common on the T60), this is **required**.

VGA support (http://www.coreboot.org/VGA_support#RECOMMENDED:_Extracting_from_your_vendor_bios_image) page on the coreboot wiki tells you how to extract it.

Place this inside the coreboot/ directory, and in menuconfig enable it under **Devices** if you are using payloads other than SeaBIOS (for SeaBIOS, you should configure SeaBIOS to run it. Coreboot's default SeaBIOS configuration will use it). In either scenario, you will want to include it in the ROM image, so make sure to specify the path "vgabios.bin" (or whatever the path to your VBIOS image is) in menuconfig, making sure to enter the correct numbers for the PCI ID (get this using **lspci -nn | grep VGA**).

Patch the coreboot ROM image for bucts

Failure to follow this will result in a bricked laptop.

BUC.TS

Back**u**p **C**ontrol **T**op **S**wap.

1. Run the `dd` command below to shift the first 64K of data from `coreboot.rom`

```
dd if=coreboot.rom of=top64k.bin bs=1 skip=$[$(stat -c %s coreboot.rom) - 0x10000] count=64k
```

2. Run the dd command below to display the first 64k of `coreboot.rom`

```
dd if=coreboot.rom bs=1 skip=$[$(stat -c %s coreboot.rom) - 0x20000] count=64k | hexdump
```

3. Verify that the complete range is filled with `ff` bytes before proceeding.

   The output of the `dd` command above must EXACTLY match the text below. If not, the coreboot image needs to be rebuilt with the second-to-last 64kbyte block unused.

```
0000000 ffff ffff ffff ffff ffff ffff ffff ffff *0010000
```

4. Run the `dd` command below:

```
dd if=top64k.bin of=coreboot.rom bs=1 seek=$[$(stat -c %s coreboot.rom) - 0x20000] count=64k conv=notrunc
```

What this did was copy the upper 64KiB section of the ROM image, into the section below. This is the boot block; lenovobios prevents writing to the upper 64KiB block (by default, it prevents all other regions but a patched flashrom binary can flash those regions in software). A utility called **bucts** will be used later on to set the system up so that it boots from the before-final 64KiB block during the initial installation of coreboot.

- Source: gmane.linux.bios Mailing List - LinuxBIOS on T60 (http://comments.gmane.org/gmane.linux.bios/69354) - Peter Stuge's Method of installing Coreboot on the X60.

Install Coreboot (First Flash)

The initial flash will write coreboot to the flash chip, but with the final 64KiB boot block from lenovobios (which is write-protected) intact. bucts will be used to make the system boot from the lower 64KiB boot block (before the final one) where you previously copied it to using dd.

As before, you will be using the patched version of flashrom distributed in libreboot. The binary releases of libreboot come with both bucts and flashrom already compiled (you can also build them from source, if you like). You can also get bucts and flashrom from upstream (optional):

- http://git.stuge.se/?p=bucts.git
- http://flashrom.org/ - apply the lenovobios patches from libreboot src - resources/flashrom/patch/lenovobios_* - there are 2 patches, for different chips. Build 2 executables, each with one of the patches (but not the other)

The libreboot project also distributes ROM images already compiled for the X60/T60 (dd modification for bucts already applied on all ROM images), if you prefer (for T60, please avoid these if you have either a 1024x768 screen and/or ATI graphics; also, the 15.4" widescreen T60 laptops are untested in libreboot):

1. Run `su` to become root.

2. You **must** run bucts, flipping the register so that the value is high (1) (as explained before):
3. Run `./bucts/i686/bucts 1`
    1. It should have said **Updated BUC.TS=1** for the above command. If not, please do NOT continue; get help.
4. Flash Coreboot (run both of these commands, whichever works first):

```
sudo ./flashrom/i686/flashrom_lenovobios_sst -p internal -w coreboot.rom
sudo ./flashrom/i686/flashrom_lenovobios_macronix -p internal -w coreboot.rom
```

    - This will take a while, and will spit out a few errors (since the upper 64KiB region of the flash is write-protected).
5. You'll see a lot of error output, but relax: this is normal. It will look something like this:

```
Reading old flash chip contents... done.
Erasing and writing flash chip... spi_block_erase_20 failed during command execution at address 0x0
Reading current flash chip contents... done. Looking for another erase function.
spi_block_erase_52 failed during command execution at address 0x0
Reading current flash chip contents... done. Looking for another erase function.
Transaction error!
spi_block_erase_d8 failed during command execution at address 0x1f0000
Reading current flash chip contents... done. Looking for another erase function.
spi_chip_erase_60 failed during command execution
Reading current flash chip contents... done. Looking for another erase function.
spi_chip_erase_c7 failed during command execution
Looking for another erase function.
No usable erase functions left.
FAILED!
Uh oh. Erase/write failed. Checking if anything has changed.
Reading current flash chip contents... done.
Apparently at least some data has changed.
Your flash chip is in an unknown state.
```

1. If the errors are like that then, contrary to the error output, the image was flashed successfully.
    - If they don't match, **DO NOT TURN OFF YOUR LAPTOP**; get help instead.
2. Shut down the laptop (fully shut it down, as in, turn it off), and then boot it again in a few seconds. Your laptop will boot into Coreboot.

Note: SeaBIOS will not display anything without a proprietary VGABIOS blob (Intel or ATI GPU) or (Intel GPU only) the free SeaVGABIOS ("coreboot linear framebuffer" in SeaBIOS menuconfig) option ROM in SeaBIOS combined with native graphics initialization, but GNU/Linux should work fine in any case.

- Sources: Peter's mail (http://www.flashrom.org/pipermail/flashrom/2012-April/009124.html) - Mailing List Thread 1 (http://thread.gmane.org/gmane.linux.bios/69354) - Mailing List Thread 2 (http://thread.gmane.org/gmane.linux.bios.flashrom/575)

Install Coreboot (Second Flash)

Next, flash Coreboot a second time to overwrite the original boot block. This time, you can (and should) used an unpatched version of flashrom. Libreboot also comes with this, or (once again) you can also use upstream if you like.

1. Run `su` to become root, and change to the libreboot_bin or libreboot_util directory.
    1. Run `./flashrom/i686/flashrom -p internal -w coreboot.rom`
    2. It should say **Verifying flash... VERIFIED** at the end of the output. If not, get help.
2. Reset bucts back to normal (only if the step above worked):
    1. Run `bucts 0`
3. Reboot the laptop. Coreboot has been successfully installed.

Recovery with a Hardware Firmware Flasher

If you had a bad flash you will need to use a hardware flasher to reflash the BIOS.

If you want something that's easy to follow, the libreboot project shows how to flash externally using a BeagleBone Black

- Unbricking the X60 (http://libreboot.org/docs/install/x60_unbrick.html)
- Unbricking the X60 Tablet (http://libreboot.org/docs/install/x60tablet_unbrick.html)
- Unbricking the T60 (http://libreboot.org/docs/install/t60_unbrick.html)
- How to flash using the BBB (http://libreboot.org/docs/install/bbb_setup.html)

Coreboot standard configuration

- It's now the default that when running SeaBios, that it (instead of coreboot) runs the VGA option rom.

See VGA_support for details on how to include the VGA BIOS image.

Recently tested revisions on the X60

See the most recent board-status submissions (http://review.coreboot.org/gitweb?p=board-status.git;a=tree;f=lenovo/x60;hb=HEAD)

970ad7076388b3ef98988121170df86196d493b4 coreboot-4.0-5534-g970ad70

8496c4dbec41b3a9284fc29b0dcd97fc8946223b coreboot-4.0-5045-g9bf05de

Recently tested revisions on the T60

See the most recent board-status submisssions (http://review.coreboot.org/gitweb?p=board-status.git;a=tree;f=lenovo/t60;hb=HEAD)

a172ea546992c3f6f6a99b4dbaabbdae4c959707 4.0-5611-ga172ea5

9bf05de5ab2842fc83cea8da5e9058417fc4bc24 4.0-5045-g9bf05de

Status

- Thinkpad X60 Status
- Thinkpad T60 Status

OLD INFO

The status and installation pages used to be one in the same.

This page underwent massive changes, some of which weren't good. Below is a copy of what used to be on the old page:

Flashing on the laptop instructions.

Lenovo X60, **X60s**, **T60** and **T60p** flashing instructions.

These Lenovo laptops have a register that must be flipped before coreboot can be flashed.

For those/some models with SPI flash chips you have also to modify flashrom. Because the chipset locks down the available commands that flashrom can send to the flash chip, you also need to change the flashrom source in a way that is not suitable to upstream. Flash chips can be identified by various commands (REMS*, RDID etc.). Some of them reply with an ID for the vendor and the exact chip model; others just reply with a single byte which is fine if there is only a small number of chips to distinguish, but won't work for the huge number of flash chips known to flashrom. The problem with the vendor BIOS is that it forbids the higher quality identification commands, so you need to force flashrom to use the lower quality opcode for the chip in your Thinkpad. You have to know the chip model beforehand (e.g. by inspection). Known models on the x60s are SST25VF016B, MX25L1605D and maybe others.

You will need: the flashrom source (http://flashrom.org/Download#Installation_from_source) (at least r1613 to make sure the laptops are whitelisted to work with flashrom), a small modification of it (as explained below in detail), and the bucts utility (http://git.stuge.se/?p=bucts.git).

1. Patch flashrom to use RES SPI identification and spi_chip_write_1 for your flash chip, as well as change the flash chip model id to fit the RES opcode.
   - Find the definition of your flash chip in flashrom's flashchips.c
     - Optionally, you can copy the existing definition as it is done in this patch (http://patchwork.coreboot.org/patch/3621/). This will allow to switch between the two definitions with the -c parameter. Be sure to change the `.name` field in that case (e.g. `.name = "SST25VF016B-RES",`).
   - Change the .probe field to probe_spi_resN where N equals the number of ID bytes the flash replies to the RES ID command (e.g. `.probe = probe_spi_res2`, if the chip replies with one byte vendor ID and one byte model ID)
   - Change the .model_id field to the RES model ID given in the datasheet of the flash chip (e.g. `.model_id = 0x14,`)
   - Change the .write field to spi_chip_write_1 (i.e. `.write = spi_chip_write_1,`)
2. Run `flashrom -p internal -r factory.bin`

   This step is IMPORTANT since the factory BIOS in your machine is tied to your particular system board (or "planar" in IBM FRU terms) with a unique ID not present in factory BIOS updates.

3. Run `dd if=coreboot.rom of=top64k.bin bs=1 skip=$[$(stat -c %s coreboot.rom) - 0x10000] count=64k`
4. Run `dd if=coreboot.rom bs=1 skip=$[$(stat -c %s coreboot.rom) - 0x20000] count=64k | hexdump`

   Verify that the complete range is filled with ff bytes before proceeding! The above command must output:
   ```
   0000000 ffff ffff ffff ffff ffff ffff ffff ffff
   *
   0010000
   ```
   If this is not the case, the coreboot image needs to be rebuilt with the second-to-last 64kbyte block unused.

5. Run `dd if=top64k.bin of=coreboot.rom bs=1 seek=$[$(stat -c %s coreboot.rom) - 0x20000] count=64k conv=notrunc`
6. Run `bucts 1`
7. Run `flashrom -p internal -w coreboot.rom`

   This will be slow, it will output errors for addresses 0x0 and 0x1f0000 when working with a 2 Mbyte flash chip, and it will say "FAILED!" at the end, see Peter's mail (http://www.flashrom.org/pipermail/flashrom/2012-April/009124.html) before you panic.

8. Power cycle the machine (i.e. a cold boot, not just a reboot), now starting with coreboot
9. Revert all changes made to flashrom (maybe backup the binary for later experiments)
10. Run `flashrom -p internal -w coreboot.rom`.

    This will successfully overwrite the entire flash chip, including the last 64k that were write protected with the factory BIOS.

11. Run `bucts 0`

See also http://thread.gmane.org/gmane.linux.bios/69354 http://thread.gmane.org/gmane.linux.bios.flashrom/575

Recovery

If you had a bad flash you will need a recovery method.

If you only set bucts, then rebooted without doing any flash writes, things might be easier: bucts sets a register that lives on the RTC well, ie. it is powered by the same source that keeps the clock alive. Usually that's a battery on the mainboard, and often there's some way to cut the source (by removing the battery, a jumper, or pads that can be shorted). After doing that (for a few seconds, there might be some capacitors in the way that keep power stable), the register should be reset and the system should boot as normal.

On the x60x, bucts issues might also be solved by "discarging RTC", which is done by pressing the power button 5 times for 10 seconds.

Required/advised hardware and informations

- X60 Hardware Maintenance Manual (http://download.lenovo.com/ibmdl/pub/pc/pccbbs/mobiles_pdf/42x3550_04.pdf) or T60 Hardware Maintenance Manual (http://download.lenovo.com/ibmdl/pub/pc/pccbbs/mobiles_pdf/42t7844_04.pdf) for disassembling the laptop
- An SO-8 IC clip, like the Pomona 5250 (http://www.tme.eu/en/details/pom-5250/test-clips/pomona/5250/) for instance.
- An external flashrom programmer

Howto

0. wire the pomona clip to a programmer that way:

From the #coreboot IRC Channel on FreeNode servers:

```
Oct 01 15:35:48 <CareBear\>     one important thing is that when you connect the clip to the X60 you should not connect all pins
[...]
Oct 01 15:36:22 <CareBear\>     only connect these pins: 1, 2, 4, 5, 6
[...]
Oct 01 15:37:21 <CareBear\>     also important: first connect charger to laptop, then connect the clip
[...]
Oct 01 17:49:41 <CareBear\>     GNUtoo-desktop : the mainboard must be powered off, but with the charger connected
[...]
Oct 01 17:50:39 <CareBear\>     um, that way there is no way anything will break
[...]
Oct 01 17:51:00 <CareBear\>     it is important not to connect 3v3 from the outside
Oct 01 17:51:39 <CareBear\>     because the correct power sequencing is not known, and if any other rail must come on before the standby 3v3 t
hen the machine may well break when 3v3 is applied from the outside
[...]
Oct 01 17:52:48 <CareBear\>     it may also be fine - but it is unknown what happens
[...]
Oct 01 17:53:47 <CareBear\>     not supplying 3v3 from the outside is safer
Oct 01 17:54:25 <CareBear\>     and because the machine is powered off, there is no risk of the chipset accessing the flash chip
```

In another hand I didn't follow that and wired it without powering the mainboard(mainboard disconnected from power plug, no battery in) and with all pins and it worked…

1. Disassemble carefully the laptop, the SO-8 chip is on the bottom of the mainboard…
2. connect the pomona clip to the BIOS chip
3. flash coreboot or the BIOS
4. remount the laptop

Coreboot standard configuration

- It's advised to make SeaBios(instead of coreboot) run the VGA option rom by disabling CONFIG_VGA_ROM_RUN:

```
[ ] Run VGA Option ROMs
```

in make menuconfig. Note that you still need to include the option rom in coreboot:

```
[*] Add a VGA BIOS image
```

See VGA_support for details on how to include the VGA BIOS image.

- Also disable CONFIG_S3_VGA_ROM_RUN which is for really old linux kernels(2.4) (which is disabled automatically if you don't select CONFIG_VGA_ROM_RUN).

From the #coreboot IRC Channel on FreeNode servers:

```
Oct 04 13:47:09 <patrickg>      that's about running vga init on s3 wakeup - required for some older linux kernels
[...]
Oct 04 13:47:25 <patrickg>      BIOSes call it "POST on wakeup" or sth like that
Oct 04 13:47:30 <patrickg>      older ~ 2.4 class ;)
```

Status

- Thinkpad X60s Status
