Board:lenovo/x1 carbon gen1

The wiki is being retired!

Documentation is now handled by the same processes we use for code: Add something to the Documentation/ directory in the coreboot repo, and it will be rendered to https://doc.coreboot.org/. Contributions welcome!

Contents

Status

Intel_Native_Raminit has it's own status page.

Thanks for your interest in the Lenovo X1 Carbon 1st gen port. The X1 Carbon 1st gen is very similar to the Lenovo X230. See Board:lenovo/x230.

Issues

- Powered USB 2.0 port isn't powered in power-off state.
- S3 Suspend when resuming may cause black screen when not using a VGA option rom.
- SeaBIOS flickers when not using a VGA option rom.
- Bluetooth fails to load firmware. Unknown if related to coreboot. Needs more testing.
- Keyboard backlight (fn + spacebar) has 4 modes instead of three. Two modes do not illuminate the keyboard (should be one), the remaining two modes illuminate the keyboard with low and high brightness as expected.

Tested

- S3 (Suspend to memory)
- S4 (Suspend to disk)
- USB (both 2.0 and 3.0 ports)
- ThinkPad USB 3.0 Ethernet Adapter (Device ID 17ef:7205)
- Video (internal)
- Sound (integrated speakers, integrated mic, external headphones)
- WLAN (Centrino Advanced-N 6205)
- WLAN toggle switch
- Linux Boot / Install (SeaBIOS)
- SD card reader (Ricoh Co Ltd PCIe SDXC/MMC Host Controller [1180:e823] (rev 07))
- Thermal management
- Fingerprint reader (147e:2020 Upek TouchChip Fingerprint Coprocessor)
- Webcam
- Trackpoint
- Touchpad
- Fn hotkeys
- Physical hotkeys (power, audio mute, mic mute, volume)
- Nvramcui as secondary payload
- Keyboard backlight
- Screen backlight
- mSATA

Untested

- Displayport
- WWAN

Proprietary Components Status

- CPU Microcode
- VGA Option ROM (optional): you need it if you want graphics in SeaBIOS but most payloads should work without it (text mode or corebootfb mode)
- ME (Management Engine) => you do not have to touch it (just leave it where it is)
- EC (Embedded Controller) => you do not have to touch it (just leave it where it is)

Code

{{ #if: | * [{{{review_url}}} The code has been merged into coreboot master]: | * The code has been merged into coreboot master:}}

```
$ git clone https://review.coreboot.org/coreboot.git
```

Make sure to check out 3rd party blobs.

```
cd ./coreboot/3rdparty
git clone http://review.coreboot.org/p/blobs.git
```

Please have a look at Build_HOWTO. Start compiling crossgcc right now! It will take a while. You will build coreboot itself at a later time.

Preparation

Required Tools

- External flash programmer. A CH341A flash programmer was used for this guide. You can purchase these for under $10.00 USD.
- SOIC8 clip for externally reading and flashing. This eliminates the need for any soldering. You can purchase these for around $12.00 USD.
- Secondary system with flashrom installed. Another laptop with Linux installed was used while constructing this guide.




Update BIOS

It is a good idea to make sure you're on the latest firmware from Lenovo before flashing coreboot. This guide was written using the 2.75 bios.

Highly suggested: Install Linux and configure SSH before flashing coreboot to your X1 Carbon. This will allow you to troubleshoot from a running system if your display does not initialize after flashing.

- Bios Update CD Image (http://pcsupport.lenovo.com/us/en/products/laptops-and-netbooks/thinkpad-x-series-laptops/thinkpad-x1-carbon-type-34xx/3460/7zg/downloads/ds030685)
- Bios Update Windows Utility (http://pcsupport.lenovo.com/us/en/products/laptops-and-netbooks/thinkpad-x-series-laptops/thinkpad-x1-carbon-type-34xx/3460/7zg/downloads/ds030684)

Hardware Teardown

Proceeds as follows:

1. Turn off your laptop, remove AC adapter.
2. Detach the keyboard bezel assembly (https://www.youtube.com/watch?v=rBnEHuJLhb4). You can keep the ribbon cable connected and pivot the keyboard assembly against the lid of the laptop.
3. Remove main battery conectoy and cmos battery connector.


Mainboard Flash Layout - MX25L3273E and MX25L6406E

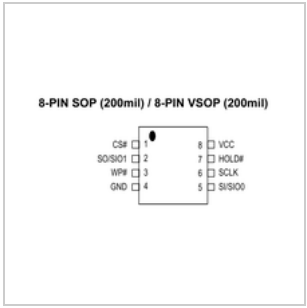
The X1 Carbon 1st gen has two flash chips of 4M (MX25L3273E) and 8M (MX25L6406E). The flash chips

are located just above the mSATA SSD. They're concatenated to one virtual flash chip of 12M which is itself subdivided in roughly in 3 parts:

- Descriptor (12K)
- ME firmware (5M-12K)
- System flash (7M)

ME firmware is not readable. Vendor firmware locks the flash and so you need to flash externally until you re-flash unlocked firmware.

When flashing coreboot you technically only need to flash the 4M MX25L3273E flash chip. You could follow this guide and flash coreboot without ever reading or writing to the 8M MX25L6406E, however you will need to read the 8M chip if you intend to extract the VGA BIOS.



Lenovo Firmware Backup

Before building and flashing coreboot we need to make a copy of the existing firmware.

MX25L3273E Backup

Connect your SOIC clip to the MX25L3273E flash chip (the left one). Insert the CH341A flash programmer into a USB port. Read the stock firmware twice from the 4M MX25L3273E flash chip:

```
flashrom -p ch341a_spi -c MX25L3273E -r x1c_MX25L3273E_original_bios_1.bin
flashrom -p ch341a_spi -c MX25L3273E -r x1c_MX25L3273E_original_bios_2.bin
```

Compare the two images to ensure a successful read.

```
sha256sum x1c_MX25L3273E_original_bios_1.bin  x1c_MX25L3273E_original_bios_2.bin
```

sha256sum output:

```
b5bc2c096d53ed15fdbfe99c41bbb7ae9311366329cc19f611ba3f743c09b8e4  x1c_MX25L3273E_original_bios_1.bin
b5bc2c096d53ed15fdbfe99c41bbb7ae9311366329cc19f611ba3f743c09b8e4  x1c_MX25L3273E_original_bios_2.bin
```

The hashes should match. If not, repeat the above steps. Your SOIC clip may require slight downward pressure during read and write operations. You'll need to get a feel for the clip. Some are made better than

others.

Consolidate and rename your extracted firmware images (MX25L3273E)

```
rm x1c_MX25L3273E_original_bios_2.bin
mv x1c_MX25L3273E_original_bios_1.bin x1c_MX25L3273E_original_bios.bin
```

MX25L6406E Backup

Connect your SOIC clip to the MX25L6406E flash chip (the right one). Insert the CH341A flash programmer into a USB port. Read the stock firmware two separate times from the 8M MX25L6406E flash chip:

```
flashrom -p ch341a_spi -c "MX25L6406E/MX25L6408E" -r x1c_MX25L6406E_original_bios1.bin
flashrom -p ch341a_spi -c "MX25L6406E/MX25L6408E" -r x1c_MX25L6406E_original_bios2.bin
```

Compare the two images to ensure a successful read.

```
sha256sum x1c_MX25L6406E_original_bios1.bin x1c_MX25L6406E_original_bios2.bin
```

sha256sum output:

```
affd1adb758e971650c81a495fff5c5cfd88a206e2da6cf81a1032835a5d52c7  x1c_MX25L6406E_original_bios1.bin
affd1adb758e971650c81a495fff5c5cfd88a206e2da6cf81a1032835a5d52c7  x1c_MX25L6406E_original_bios2.bin
```

The hashes should match. If not, repeat the above steps.

Consolidate and rename your extracted firmware images (MX25L6406E)

```
rm x1c_MX25L6406E_original_bios2.bin
mv x1c_MX25L6406E_original_bios1.bin x1c_MX25L6406E_original_bios.bin
```

12M Virtual Flash Image

After successfully extracting the firmware from both flash chips, combine the 4M and 8M firmware images into the full 12M virtual flash image.

```
cat x1c_MX25L3273E_original_bios.bin x1c_MX25L6406E_original_bios.bin > x1c_original_bios.bin
```

Extract VGA BIOS - UEFITool (https://github.com/LongSoft/UEFITool)

Pleaes see VGA_support.

Use git to clone the UEFITool repository and build UEFITool. Requires qt5base-devel package in Ubuntu 16.04. See the included README.

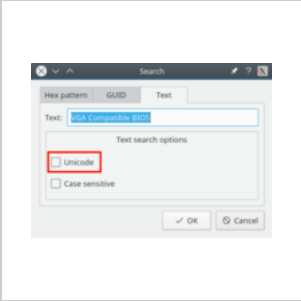
```
git clone https://github.com/LongSoft/UEFITool.git
cd ./UEFITool
```

Build UEFITool

```
/usr/lib/x86_64-linux-gnu/qt5/bin/qmake /uefitool.pro
make release
```

Launch UEFITool

1. Open the 12M combined bios file (x1c_original_bios.bin) - ***File -> Open image file…***
2. Search for text "VGA Compatible BIOS" (uncheck unicode) - ***File -> Search***
3. Double click search the search result. This will highlight the raw section.
4. Right click highlighted raw section and select ***Extract Body...***. Save the extracted file. This is your extracted vga bios - 'x1c_vga_bios.img'.



Run sha256sum to calculate the checksum:

```
sha256sum x1c_vga_bios.img
```

Output:

```
2ffad3deec22bde663721c496de5792a0429f5732145f0782bb389dda3ac7ed6  x1c_vga_bios.img
```

We'll come back to this vga bios file when it's time to build coreboot.

Build Coreboot

Building Firmware

Please have a look at Intel_Sandybridge_Build_Tutorial.

In the coreboot directory you can configure the build-time options of coreboot with nconfig.

```
make nconfig
```

Expand this to view recommended nconfig options: [Expand]

Recommended '.config' file included for reference with the above configuration (SeaBios, nvramcui, VGA BIOS image).

Expand this to view .config contents: [Expand]

Build coreboot rom using make.

```
make
```

The coreboot rom is saved to ./build/coreboot.rom.

Flashing Coreboot

First we need to split the coreboot.rom file. Remember, you only need to flash coreboot to the 4M

MX25L3273E (left) flash chip. Since you have to write only top 4M, split the file with dd:

```
dd of=./x1_carbon_MX25L3273E_coreboot.rom bs=1M if=./build/coreboot.rom skip=8
```

Connect your SOIC clip to the MX25L3273E flash chip (the left one). Insert the CH341A flash programmer into a USB port. Flash the 4M coreboot rom:

```
flashrom -p ch341a_spi -c MX25L3273E -w ./x1_carbon_MX25L3273E_coreboot.rom
```

Congrats! You can now boot your X1 Carbon from coreboot!

If you're successful you will be greeted by SeaBIOS (or your preferred payload).

Internal Flashing

Please see Intel_Sandybridge_Build_Tutorial#Flashing_coreboot.

Unlock and Re-Flash MX25L6406E

Unlocking the 8M MX25L6406E flash image enables you to use flashrom internally without resorting to the SOIC clip and external flash programmer.

Build ifdtool.

```
cd ./coreboot/util/ifdtool
make
```

Use ifdtool to unlock the 8M bios image. Your paths may be slightly different.

```
./ifdtool -u ../x1c_MX25L6406E_original_bios.bin ../x1c_MX25L6406E_original_bios_unlocked.bin
```

Flash the unlocked 8M image back to the MX25L6406E.

```
sudo flashrom -p ch341a_spi -c "MX25L6406E/MX25L6408E" -w x1c_MX25L6406E_original_bios_unlocked.bin
```

Get X1 Carbon Flash Layout

Extract the flash layout using ifdtool on the 8M MX25L6406E backup image.

```
./util/ifdtool/ifdtool -f ./x1c.layout ./x1_carbon_MX25L6406E_original_bios.bin
```

It creates a layout file with the following contents:

```
00000000:00000fff fd
00500000:007fffff bios
00003000:004fffff me
00001000:00002fff gbe
```

Flashing Coreboot Internally

Use the layout file to flash only the BIOS flash region, leaving IFD, ME, and GBE alone. Use the full 12M coreboot.rom image.

```
flashrom -p internal:laptop=force_I_want_a_brick -l ./x1c.layout -i bios -w ./build/coreboot.rom
```

If you are receiving errors during flashing you most likely need to pass `iomem=relaxed` as a kernel parameter at boot time before using flashrom.

Successful flash output:

```
flashrom v0.9.9-rc1-r1942 on Linux 4.8.0-52-generic (x86_64)
flashrom is free software, get the source code at https://flashrom.org

Using region: "bios".
Calibrating delay loop... OK.
coreboot table found at 0xbff5e000.
===========================================================================
WARNING! You seem to be running flashrom on an unsupported laptop.
Laptops, notebooks and netbooks are difficult to support and we
recommend to use the vendor flashing utility. The embedded controller
(EC) in these machines often interacts badly with flashing.
See the manpage and https://flashrom.org/Laptops for details.

If flash is shared with the EC, erase is guaranteed to brick your laptop
and write may brick your laptop.
Read and probe may irritate your EC and cause fan failure, backlight
failure and sudden poweroff.
You have been warned.
===========================================================================
Proceeding anyway because user forced us to.
Found chipset "Intel QS77".
This chipset is marked as untested. If you are using an up-to-date version
of flashrom *and* were (not) able to successfully update your firmware with it,
then please email a report to flashrom@flashrom.org including a verbose (-V) log.
Thank you!
Enabling flash write... Enabling hardware sequencing due to multiple flash chips detected.
OK.
Found Programmer flash chip "Opaque flash chip" (12288 kB, Programmer-specific) mapped at physical
address 0x0000000000000000.
Reading old flash chip contents... done.
Erasing and writing flash chip... Erase/write done.
Verifying flash... VERIFIED.
```

This page was last edited on 4 October 2017, at 21:59.