Unbricking/Flashing with a ch341a USB Programmer

Requirements

> **Tips**
>
> Most Skylake and older models (with a few exceptions) use a SOIC-8 flash chip which is easily clippable. Most if not all Kabylake/Apollolake and newer devices use a WSON-8 flash chip which can't be clipped, instead you need a WSON-8 probe. Check the part number of your flash chip to find the correct size needed.

1. A ChromeOS device
2. A device running Linux from which to run flashrom. For this guide, I will use a Ubuntu live USB.
3. A ch341a USB flash programmer
4. A 1.8v adapter

   > **Tips**
   >
   > The adapter is required for devices which use 1.8v flash chips. Some/Most Baytrail, Braswell, Skylake and many newer devices use a 1.8v flash chip. Baytrail is more reliable flashing at 3.3v though due to current leakage

5. Either a SOIC-8 chip clip or a WSON-8 probe

A ch341a programmer, 1.8v adapter, and a SOIC-8 clip are often bundled together at a lower cost, and if you're unsure if your device uses a 1.8v flash chip or a 3.3v one, it makes sense to have the adapter on hand if needed. You can look up the part number of your flash chip to determine which voltage it needs.

Hardware Disassembly

While this is somewhat device-specific, the main points are the same:

- Disconnect all external power
- Remove bottom cover (screws are often located under rubber feet or strips)
  - Some Chromebooks open up through the back and some through the keyboard, and as mentioned in Disabling write protect via Battery. On keyboard, you have to pry it out and remove a ribbon wire under the keyboard.
- Disconnect the internal battery (for Chromeboxes, disconnect the small CMOS battery)
- Locate the SPI flash chip

> **Caution**
>
> Most ChromeOS devices use a Winbond flash chip, though some use a compatible chip from another manufacturer, eg Gigadevices. It will be either an 8MB, 16MB, or 32MB chip, with the identifier W25Q64[xx] (8MB), W25Q128[xx] (16MB), or W25Q256[xx] (32MB) where [xx] is usually FV or DV. We do **not** want to touch the EC firmware chip, which is identified by W25X40[xx].

> **Warning**
>
> Unfortunately, many devices have the flash chip located on the top side of the main board, and require fully removing the main board in order to flash.

> **Tips**
>
> Pin 1 of the flash chip will be notated by a dot/depression on the chip; be sure to align this with pin 1 on the chip clip wiring.

Googling should locate a disassembly guide for most models. If you can't find one for your exact model, try to find one for another model of the same manufacturer as the bottom cover (or keyboard) removal tends to be very similar.

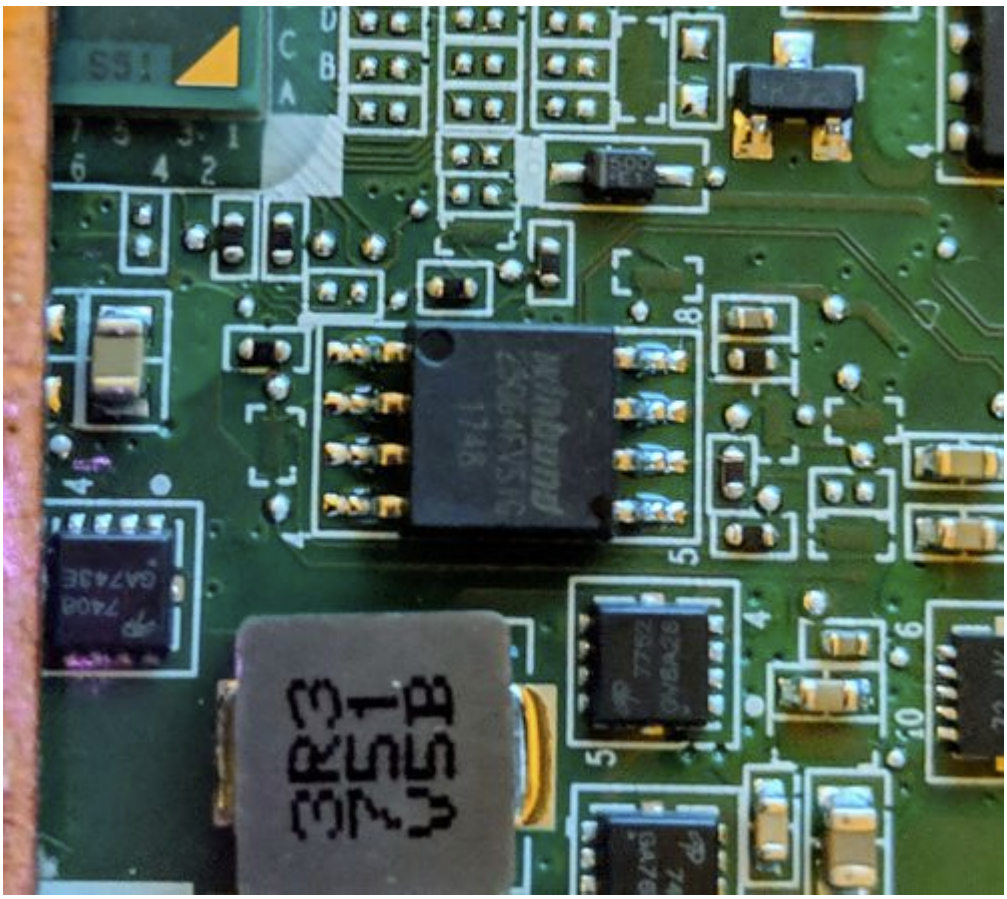
Prepping to Flash

Once you have your device disassembled and flash chip located, boot up the flashing environment. Most any Linux setup should do as long as either flashrom is available from the distro's software repositories, or it's 64-bit x86 (in which case you can download a statically compiled build of flashrom from mrchromebox.tech). This guide will use a Ubuntu live session booted from USB.

So let's get to it:

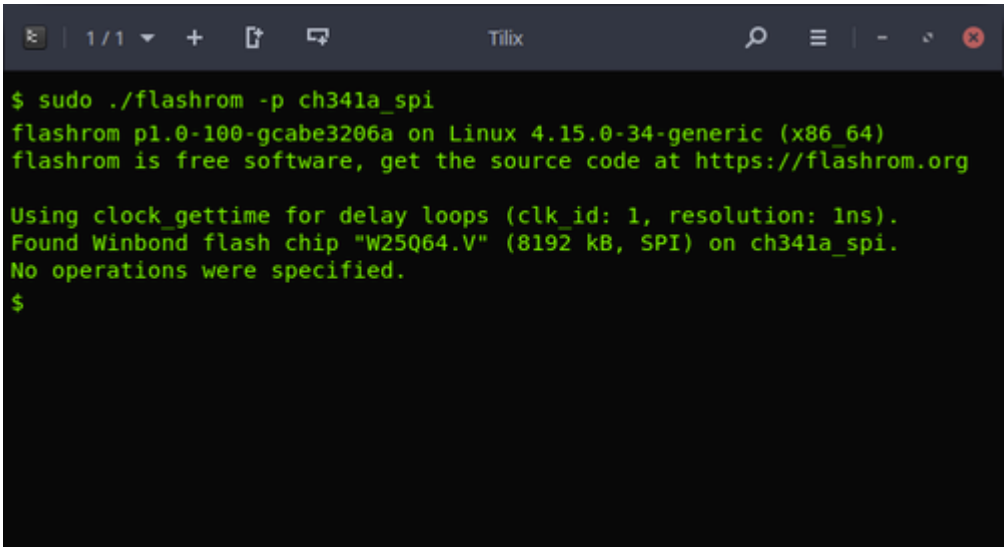
1. Boot your Linux environment (Ubuntu live USB or later recommended)
2. Connect to WiFi/internet
3. Open a (non-root) terminal/shell window, change to home directory
   - `cd;`
4. Install flashrom via apt:
   - `sudo apt update`
   - `sudo apt install flashrom`
5. Assemble ch341a programmer, 1.8v adapter (if needed), and chip clip/wiring. Ensure that pin 1 is correct and consistent.

6. Connect the chip clip to the SPI flash chip, or get ready to hold down your WSON-8 probe, rubber bands can be used to hold it down while flashing, then connect the CH341a to the Linux host machine. Note the dot/depression indicating pin 1.



7. Test connectivity and ensure the flash chip is properly identified:
   - `sudo flashrom -p ch341a_spi`

   Flashrom will produce output identifying the flash chip. If it doesn't, double check your connections to the programmer and the chip clip and retry.



8. Determine file to be flashed
   - Depending on your desired use for the device, you have 3 options for

flashing:

- The backup file of the stock firmware created by my Firmware Utility Script
  - If using this, simply copy the file from USB into the home directory of the live USB user
- The custom UEFI firmware for the device
  - If you were flashing the UEFI firmware when things went sideways, then that's the easiest way to proceed. You can download the UEFI firmware for your device by examining the sources.sh file from the Firmware Utility Script GitHub repo . Simply concatenate the device-specific filename to the Full ROM base path:
    - `wget <Full ROM base path><device specific filename>`
    - Example for the Acer Chromebook 14 CB3-431 (EDGAR): (Note this URL is years old and not actually valid)
    - `wget https://mrchromebox.tech/files/firmware/full_rom/coreboot_tiano-edgar-mrchromebox_20180827.rom`
    - Don't forget to get the SHA1 file for verification:
    - `wget https://mrchromebox.tech/files/firmware/full_rom/coreboot_tiano-edgar-mrchromebox_20180827.rom.sha1`
    - Then verify the download:
    - `sha1sum -c coreboot_tiano-edgar-mrchromebox_20180827.rom.sha1`
- The shellball firmware for the device
  - A shellball ROM can be downloaded via the coreboot `crosfirmware.sh` script from a Linux terminal:
    - `wget https://github.com/coreboot/coreboot/raw/refs/heads/main/util/chromeos/crosfirmware.sh`
    - `bash crosfirmware.sh <board name in all lowercase>`
    - Example for the Acer Chromebook 14 CB3-431 (EDGAR):
      - `wget https://github.com/coreboot/coreboot/raw/refs/heads/main/util/chromeos/crosfirmware.sh`
      - `bash crosfirmware.sh edgar`
    - This will produce a shellball firmware image named `coreboot-Google_Edgar.<version>.bin` . It will not have a valid HWID and will need the GBB flags reset after flashing to the device.

Tips

If you're not sure which file to use for your device / don't know your device's board name, you can reference the supported devices page.

Persisting the board's Vital Product Data (VPD) and Hardware ID (HWID)

The firmware in all ChromeOS devices contains a section (RO_VPD) which stores board-specific data, like the serial number, localization settings, and on many devices which have an Ethernet port, the LAN MAC address as well. When flashing via the Firmware Utility Script, the script will automatically extract this from the running firmware and inject it into the firmware to be flashed, so the device serial, LAN MAC address, etc are all maintained. Without this, the device will use a default/generic LAN MAC address set by coreboot. While not ideal, this is only really an issue if two or more of the same device are on the same LAN segment (or you're statically assigning IP addresses based on MAC). But for completeness, if flashing the UEFI firmware or shellball ROM, we'll extract the VPD (either from the board itself or a backup made by the script) and inject it into the firmware to be flashed.

> **NOTE**
>
> You don't need to do this if flashing a stock firmware backup created by the Firmware Utility Script; that image already contains the VPD.

1. For both the options below, we'll need to use the gbb_utility and cbfstool (coreboot filesystem) binaries, so let's download/extract those:
   - `wget https://mrchromebox.tech/files/util/cbfstool.tar.gz && tar -zxf cbfstool.tar.gz`
   - `wget https://mrchromebox.tech/files/util/gbb_utility.tar.gz && tar -zxf gbb_utility.tar.gz`
   - Option 1: Extract VPD and HWID from the firmware on device
     - `sudo flashrom -p ch341a_spi -r badflash.rom`
     - `./cbfstool badflash.rom read -r RO_VPD -f vpd.bin`
     - `./gbb_utility badflash.rom --get --hwid | sed 's/^hardware_id: //' > hwid.txt`
   - Option 2: Extract VPD and HWID from stock firmware backup created by Firmware Utility Script (this assumes the file has been copied into working directory)
     - `./cbfstool stock-firmware-<devicename>-<date>.rom read -r RO_VPD -f vpd.bin`

        - `./gbb_utility stock-firmware-<devicename>-<date>.rom --get --hwid | sed 's/^hardware_id: //' > hwid.txt`
2. Then we inject the VPD and HWID into the firmware image to be flashed.
    - `./cbfstool <Shellball ROM/UEFI Full ROM filename> write -r RO_VPD -f vpd.bin`
    - For UEFI Full ROM run
        - `./cbfstool <UEFI Full ROM filename> add -n hwid -f hwid.txt -t raw`
    - For Shellball run
        - `./gbb_utility <Shellball ROM> --set --hwid="$(cat hwid.txt)"`

Now the firmware image is ready to be flashed, and will maintain the device's unique serial, LAN MAC address, etc.

Flashing Your Device

Now that everything is prepped, time to flash the device. To be thorough, we'll perform a 2nd verification after flashing to ensure the integrity of the flashed firmware.

1. Flash the firmware:

    - For Intel-based ChromeOS devices, flash using
        - `sudo flashrom -p ch341a_spi -w <filename> --ifd -i bios`
    - For all other devices, use
        - `sudo flashrom -p ch341a_spi -w <filename>`

    Where `<filename\>` is the name of your backup file, UEFI firmware file, or shellball firmware file. This will usually take 30s-90s to complete; flashrom will first read the flash chip, determine which sectors differ, erase those sectors, write the new data, then verify the data written.

2. Verify the firmware

    Even though flashrom does this as part of the write process, verifying the entire flash chip is quick and an easy way to ensure everything went as it should:

    - As before, if flashing an Intel-based device, use
        - `sudo flashrom -p ch341a_spi -v <filename> --ifd -i bios`
    - Otherwise, use
        - `sudo flashrom -p ch341a_spi -v <filename>`

Using the same filename as before. If the verification passes, then disconnect the CH341a from the host machine, and then remove the chip clip.

Clean Up

Reassembly is the reverse of disassembly. Reconnect the internal battery and replace the bottom cover/keyboard. Flip over the device, connect external power, press the power button, and cross your fingers 😀

Last Updated:: 5/4/25, 6:51 PM