Firmware Write Protection on ChromeOS Devices

The firmware write protect mechanism is one of the most misunderstood aspects of ChromeOS devices; many guides/blogs/how-to's incorrectly tell users to disable it as part of whatever process they are attempting to guide the user through. Hopefully this page can clear some of that up 😃

How Does Firmware Write Protection Work?

On a typical ChromeOS device, the system (or application processor/AP) firmware is stored on a SPI flash chip, often in a SOIC-8 package. As part of the ChromeOS security model, certain parts of the device firmware are set to be read-only. The protection of these read-only regions is implemented by a combination of hardware and software measures.

Software write protection

The software write protection is implemented via special registers on the firmware chip. These registers allow for the software write-protect to be enabled or disabled, as well as one or more ranges of addresses to be protected / marked as read-only. This allows for parts of the chip to be protected (e.g., the RO firmware, or RO_VPD regions) and parts to be system (or user) writable (e.g., the RW_LEGACY region).

Hardware Write Protection

The hardware write protection is an electrical circuit which prevents writing to the software protection special registers; it's normally enforced by the grounding of the !WP pin on the firmware flash chip. Thus, the hardware write protection not only protects directly these special registers, but indirectly also the data in the firmware chip.

HW WP Implementation

- Early Chromebook models (2012-2013): use a jumper or switch to implement hardware write protection. All models prior to the 2013 Chromebook Pixel fall into this group.
- Pre-CR50 models (2014-2017): use a screw to complete the ground; removing it leaves the !WP pin floating, effectively disabled. The 2013 Chromebook Pixel was the first device to use a WP screw; all Haswell, Broadwell, Baytrail, Skylake, and Braswell-based devices do as well.
- CR50 models (2017+): On all Chromebook models with the CR50/Google Security Chip (all Kabylake/Apollolake and newer models), the !WP pin is controlled by the CR50.

On most early CR50 platforms, the CR50 sets the WP state to follow the battery sense line, so disconnecting the battery cable **from the mainboard** will disable the hardware write protect.

On some newer platforms, WP cannot be disabled by disconnecting the battery; instead there is an unpopulated jumper on the board which must be bridged.

On all CR50 devices, it is also possible to change the WP state using the closed-case debugging (CCD) features of the CR50, along with a special USB-C debug cable (called a SuzyQable). See the CCD section under Disabling FW WP.

Why Disable Firmware Write Protection?

There are only two real reasons to disable the firmware write protect on your ChromeOS device:

- To change the Google Binary Block (GBB) flags
- To flash custom firmware which modifies or overwrites the RO portions of the stock firmware

Flashing firmware which only modifies an RW portion of the firmware (like RW_LEGACY, for Legacy Boot Mode) does not require the firmware write protect to be disabled.

GBB Flags

The GBB Flags are used to modify the boot behavior of a ChromeOS device in Developer Mode. There is a wide range of functions offered by these flags, but they are most commonly used to:

- Shorten the Developer Mode boot screen timeout (from 30s to 1s, and remove the beep)
- Prevent accidental disablement of Developer Mode (via spacebar)
- Force enablement of Legacy Boot Mode (regardless of crossystem flag value)
- Set Legacy Boot Mode as the default boot path (negates the need to use CTRL+L)

For most users, there's no need to set these flags manually, as the Firmware Utility Script provides the functionality to set the desired timeout and default boot option, while setting the other flags to sane defaults.

NOTE

The GBB flags are a construct of the stock ChromeOS device firmware. They do not exist / cannot be set when running custom firmware which replaces the stock firmware (e.g., MrChromebox's UEFI Firmware).

Custom Firmware

Flashing custom firmware which completely replaces the stock firmware requires disabling the firmware write protect, since all RO and RW portions of the chip are overwritten.