

**IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF TEXAS
MARSHALL DIVISION**

LIBERTY PATENTS, LLC,

Plaintiff,

v.

ACER INC.,

Defendant.

CIVIL ACTION NO. 2:20-cv-287

ORIGINAL COMPLAINT FOR
PATENT INFRINGEMENT

JURY TRIAL DEMANDED

ORIGINAL COMPLAINT FOR PATENT INFRINGEMENT

Plaintiff Liberty Patents, LLC (“Liberty Patents” or “Plaintiff”) files this original complaint against Defendant Acer Inc. (“Acer” or “Defendant”), alleging, based on its own knowledge as to itself and its own actions and based on information and belief as to all other matters, as follows:

PARTIES

1. Liberty Patents is a limited liability company formed under the laws of the State of Texas, with its principal place of business at 2325 Oak Alley, Tyler, Texas 75703.
2. Defendant Acer Inc. is a foreign company organized and existing under the laws of Taiwan, with a place of business located at 1F, 88, Sec. 1, Xintai 5th Road, Xizhi, New Taipei City 221, Taiwan. Acer Inc. may be served with process by serving the Texas Secretary of State, 1019 Brazos Street, Austin, Texas 78701, as its agent for service because it engages in business in Texas but has not designated or maintained a resident agent for service of process in Texas as required by statute. This action arises out of that business.

JURISDICTION AND VENUE

3. This is an action for infringement of a United States patent arising under 35 U.S.C. §§ 271, 281, and 284–85, among others. This Court has subject matter jurisdiction of the action under 28 U.S.C. § 1331 and § 1338(a).

4. This Court has personal jurisdiction over Acer pursuant to due process and/or the Texas Long Arm Statute because, *inter alia*, (i) Acer has done and continues to do business in Texas; and (ii) Acer has committed and continues to commit acts of patent infringement in the State of Texas, including making, using, offering to sell, and/or selling accused products in Texas, and/or importing accused products into Texas, including by Internet sales and sales via retail and wholesale stores, inducing others to commit acts of patent infringement in Texas, and/or committing a least a portion of any other infringements alleged herein. In addition, or in the alternative, this Court has personal jurisdiction over Acer pursuant to Federal Rule of Civil Procedure 4(k)(2).

5. Venue is proper as to Defendant Acer, which is organized under the laws of Taiwan and has its principal place of business in Taiwan. 28 U.S.C. § 1391(c)(3) provides that “a defendant not resident in the United States may be sued in any judicial district, and the joinder of such a defendant shall be disregarded in determining where the action may be brought with respect to other defendants.” *See also In re HTC Corp.*, 889 F.3d 1349 (Fed. Cir. 2018).

BACKGROUND

6. The three patents-in-suit cover technology used in computer systems, such as notebook computers, laptop computers, desktop computers, tablets, and other electronic devices. More particularly, the patents-in-suit describe key improvements to electronic devices in the areas of more efficient handling of computer instructions for faster processing, better power

distribution and power management, and a better process for retrieving automatic software updates.

7. U.S. Patent No. 6,535,959 (“the ’959 Patent”) discloses a processor that includes an instruction cache. The instruction cache is a set-associative cache that comprises multiple blocks. Claim 1 of the ’959 patent is directed to a processor that generates a power reduction signal, which indicates whether the subsequent instruction to be executed resides in the same block of the instruction cache as the current instruction that is being executed. This advantageously allows, for example, the processor to read consecutive instructions (or instructions that are in the same block) quickly, without multiple additional steps. The novel system results in a processor with increased operating speed and decreased power consumption.

8. The invention described in the ’959 Patent was the result of research conducted by two inventors at Conexant Systems, Inc., which was—at the time—the world’s largest, standalone communications-IC company. Conexant, itself, was a spin-off from the semiconductor division of the well-known and well-regarded Rockwell International Corp. Conexant was known as a leading supplier of innovative semiconductor solutions for imaging, audio, embedded modem, and video surveillance applications.¹ Recently, Conexant was acquired by Synaptics, the leading developer of human interface solutions for over \$300 million. Since its formation, Conexant has been an innovator in the semiconductor field (and others) with more than a thousand patents assigned to it.

¹ See Conexant’s Audio Solution Named CES Innovations 2011 Awards Honoree, BUSINESS WIRE (Nov. 9, 2010), www.businesswire.com/news/home/20101109005618/en/Conexant%E2%80%99s-Audio-Solution-Named-CES-Innovations-2011.

9. The '959 Patent has been cited by multiple technology companies—as recently as 2017—including, Apple, Fujitsu, IBM, Honeywell, Intel, Matsushita, Oracle, and Samsung.

10. U.S. Patent No. 6,920,573 (“the '573 Patent”) generally relates to a system for conserving energy in electronic systems. Specifically, the inventor developed a system that provides much-needed energy savings for computers, such as notebooks and laptops, by including various operating modes that limit power usage. In particular, the '573 Patent describes three operating modes. The first mode is a regular operating mode where the electronic device is fully powered on and where the main microprocessor is running. The second mode is a power-saving mode where the main microprocessor is not running, yet the system is still activated. The third mode is also a power-saving mode, and more specifically, a standby mode from which the first mode can be activated. The '573 Patent also discloses components to power the system, such as a rechargeable battery, and components to control the system, such as a power button.

11. Major companies in the electronics industry have cited the invention of the '573 Patent during patent prosecution. Indeed, the '573 Patent has been cited over fifty times by leading companies, including Google, Hewlett-Packard, Intel, Matsushita, Microsoft, NVIDIA, Sony, and Transmeta.

12. U.S. Patent No. 7,493,612 (“the '612 Patent”) discloses systems and methods for automatically updating the system software of an embedded system. Claim 1 describes an embedded system capable of automatically updating system software using update agent interface programming (UAIP)—code that initiates an update of the system software during the boot process. The embedded system includes first system software and a boot image. The system also includes a micro-controller capable of transforming the first system software into

system code and the boot image into boot code. The boot code includes update agent interface programming (UAIP) for initiating updating of the first system software before executing the system code. The system can be coupled to an external data storage device, which contains the second system software (i.e., the updated system code). If there is an update to the system software, the second system software is read from the external data storage device. As a result of the '612 Patent's inventive system, a computer can advantageously retrieve automatic updates during boot without loading its outdated OS—a more efficient, time-saving solution.

13. The '612 Patent's inventive system was developed by the Taiwanese company, Lite-On Technology Corp., which develops a wide range of consumer electronics products, such as semiconductors, monitors, motherboards, etc. Lite-On was originally founded in 1975 by former employees of Texas Instruments. While the company originally developed LEDs, it branched into other industries, such as embedded systems and related software, and stayed on the forefront of developing technologies. Lite-On was recently purchased by the Japanese company, Kioxia—a former division of Toshiba—for \$165 million.

14. The '612 Patent discloses a novel and important invention that is highly relevant to today's technology, which relies heavily on recurring updates to computer systems and IoT devices. It has been cited by major technology companies like Google, IBM, and Texas Instruments.

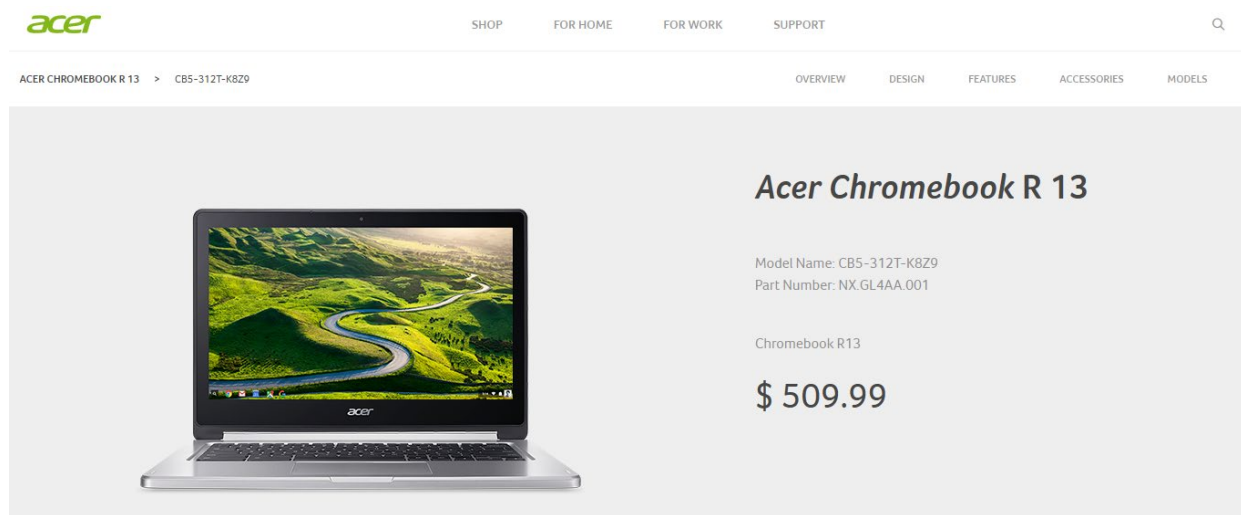
COUNT I

DIRECT INFRINGEMENT OF U.S. PATENT NO. 6,535,959

15. On March 18, 2003, the '959 Patent was duly and legally issued by the United States Patent and Trademark Office for an invention entitled "Circuit and Method for Reducing Power Consumption in an Instruction Cache."

16. Liberty Patents is the owner of the '959 Patent, with all substantive rights in and to that patent, including the sole and exclusive right to prosecute this action and enforce the '959 Patent against infringers, and to collect damages for all relevant times.

17. Acer made, had made, used, imported, provided, supplied, distributed, sold, and/or offered for sale products and/or systems including, for example, its Acer Chromebook R13 and other products (e.g., Chromebook Tab 10, Chromebook R13 CB5-312T-K5X4, Chromebook R13 CB5-312T-K8Z9, Chromebook R13 CB5-312T-K95W, etc.) that include processors with the capability to ignore reading the tag field when a sequential instruction is to be loaded (processors such as the ARM Cortex-A72, Cortex-A57, Cortex-A15, Cortex-A9, Cortex-R5, Cortex-R4, ARM11, etc.) (“accused products”):



Source: www.acer.com/ac/en/US/content/model/NX.GL4AA.001

MT8173

64-bit ARM Cortex-A72/A53 heterogenous multi-processor with CorePilot

MediaTek MT8173 is a highly integrated SOC which incorporates a 64-bit quad-core, with clusters of ARM Cortex-A53 and high performance Cortex-A72 processors operating at up to 2.0GHz. The Imagination PowerVR GX6250 GPU offers OpenGL ES 3.0. To complement, integrated is a high-end 20MP camera ISP, LPDDR3 at up to 933MHz, Ultra HD 2160p video decoding and WQXGA display capability. The MT8173 helps tablet manufacturers to build very high-performance multimedia tablets with a PC-like browser, close to console-level 3D gaming and cinema class home entertainment experiences.

Source: www.mediatek.com/products/tablets/mt8173

18. By doing so, Acer has directly infringed (literally and/or under the doctrine of equivalents) at least Claim 1 of the '959 Patent. Acer's infringement in this regard is ongoing.

19. For example, the ARM Cortex-A72 in the Acer Chromebook R 13 is a processor that includes an instruction cache. The instruction cache includes multiple cache lines or blocks.

Features

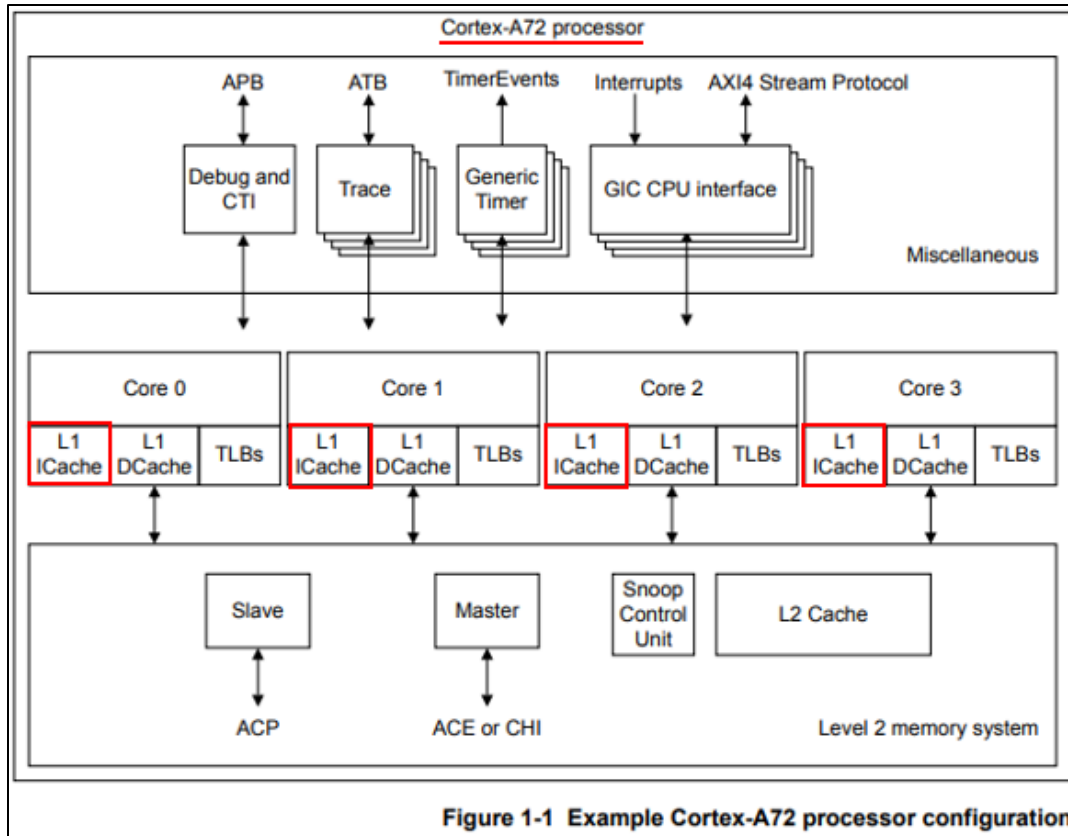
The Cortex-A72 processor includes the following features:

- Full implementation of the ARMv8-A architecture profile. See *1.2 Compliance* on page 1-15.
- Superscalar, variable-length, out-of-order pipeline.
- Dynamic branch prediction with *Branch Target Buffer* (BTB) and *Global History Buffer* (GHB) RAMs, a return stack, and an indirect predictor.
- 48-entry fully-associative L1 instruction *Translation Lookaside Buffer* (TLB) with native support for 4KB, 64KB, and 1MB page sizes.
- 32-entry fully-associative L1 data TLB with native support for 4KB, 64KB, and 1MB page sizes.
- 4-way set-associative unified 1024-entry *Level 2* (L2) TLB in each processor.
- Fixed 48K L1 instruction cache and 32K L1 data cache.
- Shared L2 cache of 512KB, 1MB, 2MB or 4MB configurable size.

Source:

https://static.docs.arm.com/100095/0002/cortex_a72_mpcore_trm_100095_0002_03_en.pdf

(Page 17).



Source:

https://static.docs.arm.com/100095/0002/cortex_a72_mpcore_trm_100095_0002_03_en.pdf

(Page 14).

Instruction fetch

The instruction fetch unit fetches instructions from L1 instruction cache and delivers up to three instructions per cycle to the instruction decode unit. It supports dynamic and static branch prediction.

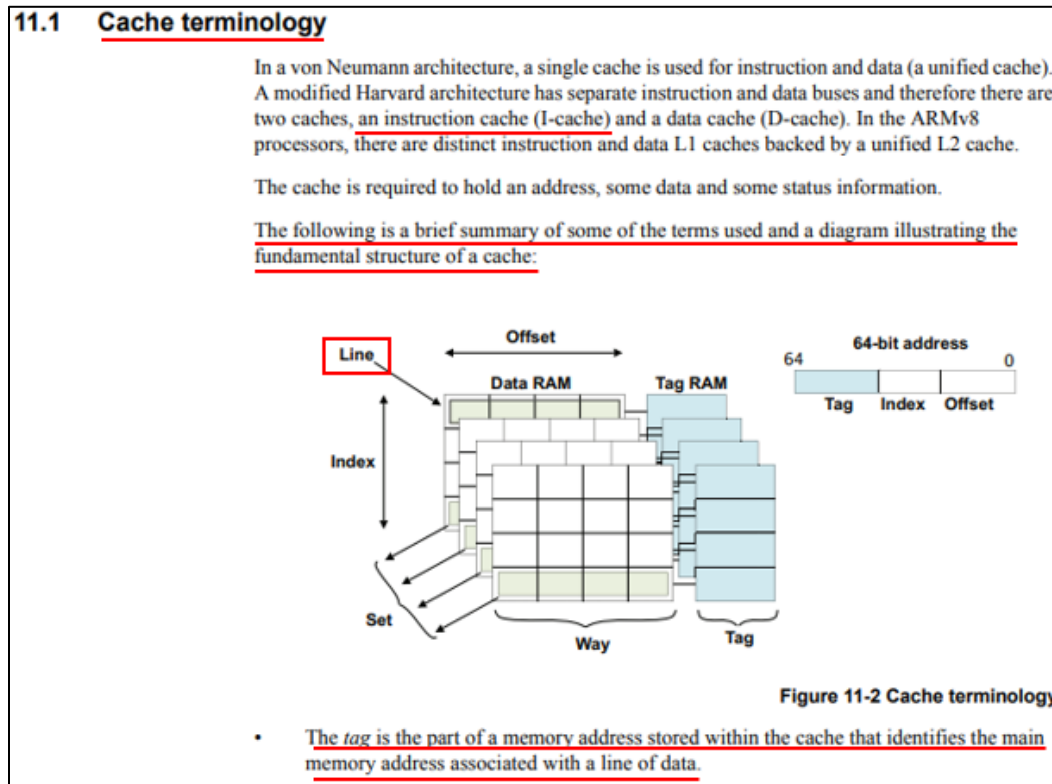
The instruction fetch unit includes:

- L1 instruction cache that is a 48KB 3-way set-associative cache with a 64-byte cache line and optional dual-bit parity protection per 32 bits in the Data RAM and 36 bits in the Tag RAM.
- 48-entry fully-associative L1 instruction *Translation Lookaside Buffer* (TLB) with native support for 4KB, 64KB, and 1MB page sizes.
- 2-level dynamic predictor with *Branch Target Buffer* (BTB) for fast target generation.
- Static branch predictor.
- Indirect predictor.
- Return stack.

Source:

https://static.docs.arm.com/100095/0002/cortex_a72_mpcore_trm_100095_0002_03_en.pdf

(Page 26).



Source:

https://static.docs.arm.com/den0024/a/DEN0024A_v8_architecture_PG.pdf?_ga=2.17157625.1756166971.1588761056-4692096.1569325365 (Page 145).

- It would be inefficient to hold one word of data for each tag address, so several locations are typically grouped together under the same tag. This logical block is commonly known as a cache line, and refers to the smallest loadable unit of a cache, a block of contiguous words from main memory. A cache line is said to be valid when it contains cached data or instructions, and invalid when it does not.

Source:

https://static.docs.arm.com/den0024/a/DEN0024A_v8_architecture_PG.pdf?_ga=2.17157625.1756166971.1588761056-4692096.1569325365 (Page 145).

Cache set	A cache set is a <u>group of cache lines (or blocks)</u> . A set contains all the ways that can be addressed with the same index. The number of cache sets is always a power of two. <i>See also</i> Cache terminology diagram on the last page of this glossary.
Cache way	A group of cache lines (or blocks). It is 2 to the power of the number of index bits in size. <i>See also</i> Cache terminology diagram on the last page of this glossary.

Source:

http://infocenter.arm.com/help/topic/com.arm.doc.ddi0301h/DDI0301H_arm1176jzfs_r0p7_trm.pdf (Page 746).

20. The ARM Cortex-A72 processor in the Acer Chromebook R 13 supports a power reduction method that is operational when an instruction is being accessed from the instruction cache. The instruction cache includes multiple cache lines or blocks, and each cache line or block is associated with a tag value. These tag values are stored in the tag RAM. The cache also includes data RAM for storing the instructions.

21. If a sequential (or subsequent) instruction to be read from the instruction cache is in the same cache line or block as the previous instruction, only the data RAM of the cache is accessed for the instruction, and the tag RAM is *not* accessed because the sequential instruction resides in the same cache line or block.

22. Accordingly, the Acer Chromebook R 13, which includes an ARM Cortex-A72, includes a circuit that sends a signal (“power reduction signal”) if a sequential instruction to be accessed from the instruction cache is identified as being in the same cache line or block.

L1 instruction memory system

The instruction cache can source up to 128 bits per fetch depending on alignment.

Sequential cache read operations reduce the number of full cache reads. This has the benefit of reducing power consumption. If a cache read is sequential to the previous cache read, and the read is within the same cache line, only the data RAM way that was previously read is accessed.

Source:

https://static.docs.arm.com/100095/0002/cortex_a72_mpcore_trm_100095_0002_03_en.pdf

(Page 287).

11.1 Cache terminology

In a von Neumann architecture, a single cache is used for instruction and data (a unified cache). A modified Harvard architecture has separate instruction and data buses and therefore there are two caches, an instruction cache (I-cache) and a data cache (D-cache). In the ARMv8 processors, there are distinct instruction and data L1 caches backed by a unified L2 cache.

The cache is required to hold an address, some data and some status information.

The following is a brief summary of some of the terms used and a diagram illustrating the fundamental structure of a cache:

The diagram illustrates the structure of a cache. It shows a grid of Data RAM cells and a corresponding Tag RAM. A 64-bit address is split into Tag, Index, and Offset. The diagram shows a grid of Data RAM cells and a corresponding Tag RAM. A 'Line' is highlighted in red, and a 'Set' is indicated by arrows. The 'Way' and 'Tag' are also labeled.

Figure 11-2 Cache terminology

- The tag is the part of a memory address stored within the cache that identifies the main memory address associated with a line of data.

Source:

https://static.docs.arm.com/den0024/a/DEN0024A_v8_architecture_PG.pdf?_ga=2.17157625.1756166971.1588761056-4692096.1569325365

(Page 145).

Instruction fetch

The instruction fetch unit fetches instructions from L1 instruction cache and delivers up to three instructions per cycle to the instruction decode unit. It supports dynamic and static branch prediction.

The instruction fetch unit includes:

- L1 instruction cache that is a 48KB 3-way set-associative cache with a 64-byte cache line and optional dual-bit parity protection per 32 bits in the Data RAM and 36 bits in the Tag RAM.
- 48-entry fully-associative L1 instruction *Translation Lookaside Buffer* (TLB) with native support for 4KB, 64KB, and 1MB page sizes.
- 2-level dynamic predictor with *Branch Target Buffer* (BTB) for fast target generation.
- Static branch predictor.
- Indirect predictor.
- Return stack.

Source:

https://static.docs.arm.com/100095/0002/cortex_a72_mpcore_trm_100095_0002_03_en.pdf

(Page 26).

- It would be inefficient to hold one word of data for each tag address, so several locations are typically grouped together under the same tag. This logical block is commonly known as a cache *line*, and refers to the smallest loadable unit of a cache, a block of contiguous words from main memory. A cache line is said to be valid when it contains cached data or instructions, and invalid when it does not.

Source:

https://static.docs.arm.com/den0024/a/DEN0024A_v8_architecture_PG.pdf?_ga=2.17157625.1756166971.1588761056-4692096.1569325365 (Page 145).

23. Acer has had knowledge of the '959 Patent at least as of the date when it was notified of the filing of this action.

24. Liberty Patents has been damaged as a result of the infringing conduct by Acer alleged above. Thus, Acer is liable to Liberty Patents in an amount that adequately compensates it for such infringements, which, by law, cannot be less than a reasonable royalty, together with interest and costs as fixed by this Court under 35 U.S.C. § 284.

25. Liberty Patents and/or its predecessors-in-interest have satisfied all statutory obligations required to collect pre-filing damages for the full period allowed by law for infringement of the '959 Patent.

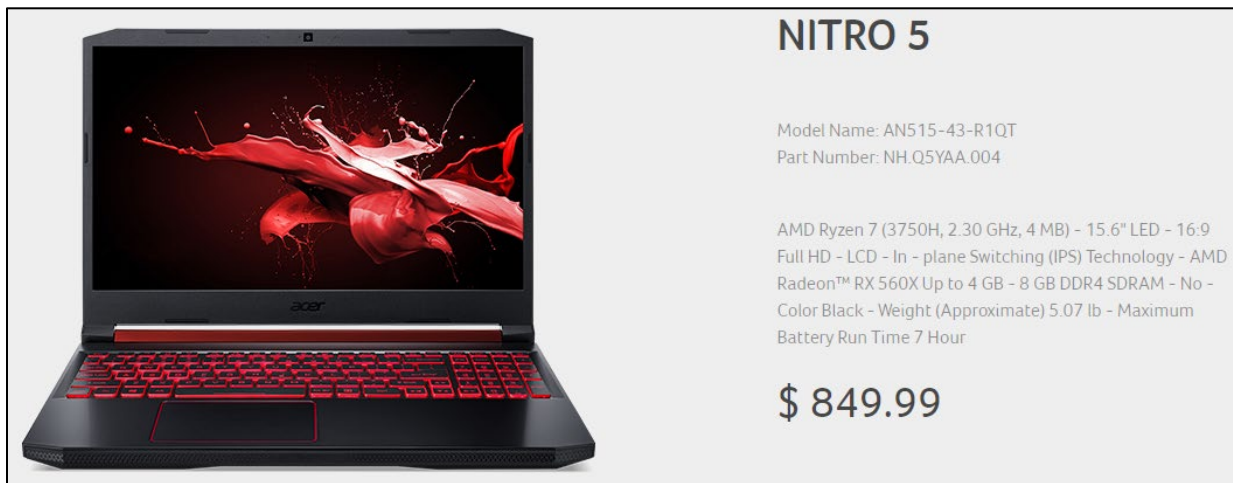
COUNT II

DIRECT INFRINGEMENT OF U.S. PATENT NO. 6,920,573

26. On July 19, 2005, the '573 Patent was duly and legally issued by the United States Patent and Trademark Office for an invention entitled "Energy-Conserving Apparatus and Operating System Having Multiple Operating Functions Stored in Keep-Alive Memory."


27. Liberty Patents is the owner of the '573 Patent, with all substantive rights in and to that patent, including the sole and exclusive right to prosecute this action and enforce the '573 Patent against infringers, and to collect damages for all relevant times.

28. Acer made, had made, used, imported, provided, supplied, distributed, sold, and/or offered for sale products and/or systems including, for example, its Acer Nitro 5 notebook and other products² including the "Power-Off Charging" feature ("accused products"):



² See, e.g., Acer Aspire 5; Acer Predator Helios 300 (PH317-53-77HB); Acer Aspire E 15 (E5-575G-57D4); Acer Aspire E 15 (E5-575-33BM); Acer Aspire R 14 Convertible; Acer Aspire 5 (A515-43-R19L).

Source: <https://www.acer.com/ac/en/US/content/model/NH.Q5YAA.004>

#	Icon	Item	Description
6		USB port with power-off charging	Connects to USB devices.

Source: <https://files.bbystatic.com/HRJdX5dzFTsthH%2B%2BTB1f2w%3D%3D/A9984950-7A67-43C6-877B-876F9DA38551.pdf> (Page 11)

29. By doing so, Acer has directly infringed (literally and/or under the doctrine of equivalents) at least Claim 13 of the '573 Patent. Acer's infringement of the '573 Patent is ongoing.

30. The Acer Nitro 5 is an information-processing apparatus with multiple operating functions. It includes a first group of circuitry that is actuatable to provide a first operating function. The first group of circuitry comprises main microprocessor circuitry.

31. For example, the Acer Nitro 5 includes a quad-core AMD Ryzen 7 processor. AMD Ryzen 7 includes four cores that further comprise Arithmetic Logic Units (ALU), Instruction and Data Caches, and other blocks. The processor also has different states like working state, sleeping state and off state etc., which correspond to the device's Power On mode, Sleep mode, and Shut Down mode, respectively. The processor operates differently depending on the current operating mode.

32. During Power On mode, the processor provides processing functions, including application processing, graphics processing, etc. ("first operating function"). The processing blocks like ALU, FPU, GPU etc. ("first group of circuitry") consume power and implement these required functions. These blocks are part of the core ("main microprocessor circuitry") of the processor.



NITRO 5

Model Name: AN515-43-R1QT
Part Number: NH.Q5YAA.004

AMD Ryzen 7 (3750H, 2.30 GHz, 4 MB) - 15.6" LED - 16:9 Full HD - LCD - In - plane Switching (IPS) Technology - AMD Radeon™ RX 560X Up to 4 GB - 8 GB DDR4 SDRAM - No - Color Black - Weight (Approximate) 5.07 lb - Maximum Battery Run Time 7 Hour

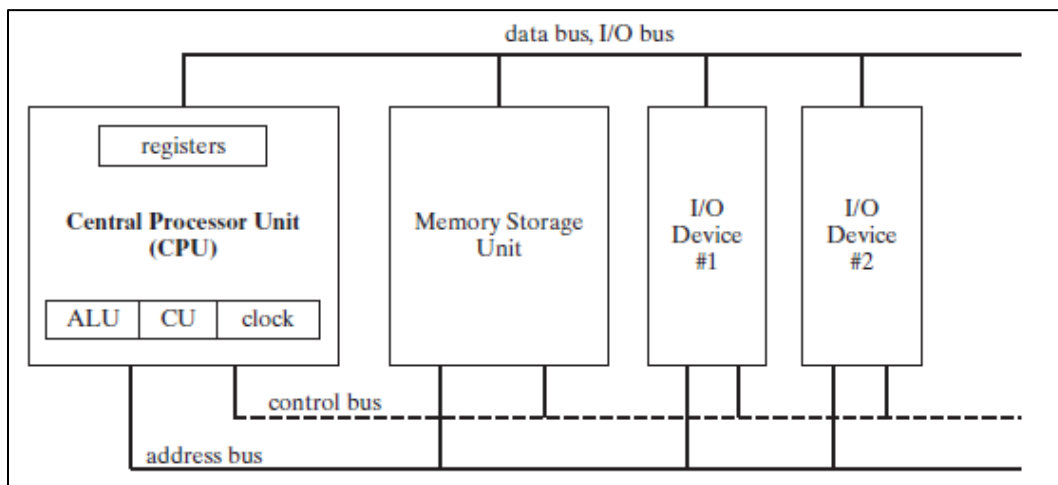
\$ 849.99

Source: <https://www.acer.com/ac/en/US/content/model/NH.Q5YAA.004>

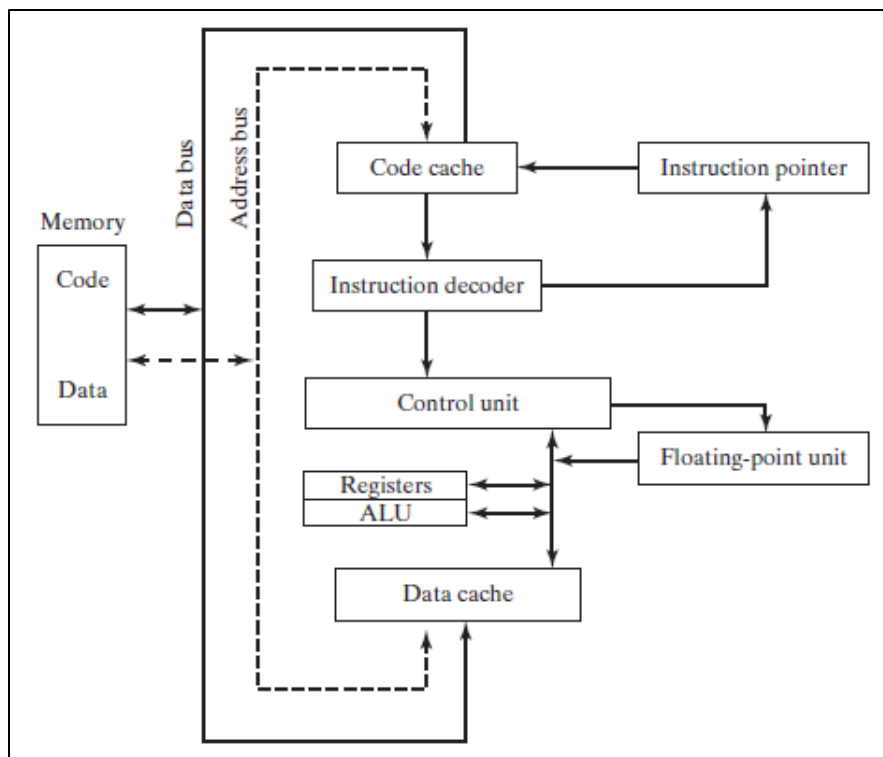
Visual Intensity

Explore games in greater detail with the sharp visuals of a 17.3 or 15.6-inch FHD IPS display. Enjoy smooth, blur-free gameplay with a 144Hz¹ refresh rate and a 3ms response time^{1,2}. We've boosted the screen-to-body ratio to 80% with narrow 7.02mm bezels¹ and provided lifelike colors using a 72% NTSC, 300 nits panel¹.

Source: <https://www.acer.com/ac/en/US/content/series/nitro5>



Source: <https://www.allaboutcircuits.com/technical-articles/an-introduction-to-x86-processor-architecture/>



Source: <https://www.allaboutcircuits.com/technical-articles/an-introduction-to-x86-processor-architecture/>

33. The following citations disclose different operating modes of the device, including Shut Down mode, Sleep mode, and Power On mode. The computer operates differently according to the current operating mode.

Turning your computer off

To turn the power off, do any of the following:

- Use the Windows shutdown command: Press the *Windows key* or select the *Windows Start button*, select **Power > Shut down**.
- Right-click the *Windows Start button* > **Shut down or sign out > Shut down**.

If you need to power down the computer for a short while, but don't want to completely shut it down, you can put it to Sleep by doing any of the following:

- Press the power button.
- Press the sleep hotkey.
- Press the *Windows key* or select the *Windows Start button*, select **Power > Sleep**.
- Right-click the *Windows Start button* > **Shut down or sign out > Sleep**.

Source: <https://files.bbystatic.com/HRJdX5dzFTsthH%2B%2BTB1f2w%3D%3D/A9984950-7A67-43C6-877B-876F9DA38551.pdf> (Page 6)

Disconnecting from the desktop

Follow these steps to disconnect your computer from external accessories:

1. Save any open files.
2. Remove discs from optical drive.
3. Shut down the computer or put it into Sleep or Hibernate mode.
4. Close the display cover.
5. Disconnect the cord from the AC adapter.

Source: <https://files.bbystatic.com/HRJdX5dzFTsthH%2B%2BTB1f2w%3D%3D/A9984950-7A67-43C6-877B-876F9DA38551.pdf> (Page 47)

34. The Acer Nitro 5 includes a second group of circuitry that is actuatable to provide a second operating function. During the second operating function, the system is not required to activate the main microprocessor circuitry.

35. For example, the Acer Nitro 5 has a USB Charge Manager program (which controls power-off charging) that allows the user to charge (“second operating function”) USB

connected devices (such as such as a mobile devices, cameras, activity trackers, smartwatches, etc.) even when the device is in Shut Down or Off mode. Mobile devices can be charged using the designated USB port having power-off charging capability without requiring the Acer Nitro 5 to be in working state (i.e., Power On mode). The corresponding USB charger IC/USB board circuit (“second group of circuitry”) can be actuated to provide the charging function during Shut Down mode.

USB Charge Manager

What is USB Charge Manager and how do I use it?

USB Charge Manager allows you to charge mobile devices any time the notebook is turned on or in Sleep mode. Power-off USB charging allows you to charge mobile devices using the designated USB port, even when the notebook is off or in Hibernation mode. USB Charge Manager is only available on systems with a USB 3.0 port designated by the graphic:




USB Charge Manager has two settings which control power-off USB charging. By default, the system will charge devices when the notebook is plugged in or on battery power. If the notebook is on battery power, charging will stop when the battery reaches 30%.

Source: https://us.answers.acer.com/app/answers/detail/a_id/29968/~/-usb-charge-manager



Source: <https://youtu.be/4JBi48wNTIk?t=42> (00:42 Mins)

#	Icon	Item	Description
6		USB port with power-off charging	Connects to USB devices.

Source: <https://files.bbystatic.com/HRJdX5dzFTsthH%2B%2BTB1f2w%3D%3D/A9984950-7A67-43C6-877B-876F9DA38551.pdf> (Page 11)

36. The Acer Nitro 5 includes a third group of circuitry that is actuatable to provide a standby function that allows the first group of circuitry (when deactivated) to be reactuable so that it can provide the first operating function. The third group of circuitry also comprises keep-alive memory circuitry for storing information needed for resuming the first operating function or the second operating function.

37. For example, the Acer Nitro 5 includes different operating modes like Sleep mode, Power On mode, and Shut down mode. The Sleep mode (“standby function”) can be activated and deactivated (i.e., to wake up the system) by pressing the Power button. The Acer Nitro 5 includes corresponding circuitry (“third group of circuitry”) that activates and deactivates the Sleep mode.

38. During Sleep mode, computational tasks are not performed, and the system consumes less power. The system retains enough context to return to a working state (“resuming said first operating function”) by storing or saving information in hardware memory, such as RAM or in a disk (“keep-alive memory circuitry”).

If you need to power down the computer for a short while, but don't want to completely shut it down, you can put it to *Sleep* by doing any of the following:

- Press the power button.
- Press the sleep hotkey.
- Press the *Windows key* or select the *Windows Start button*, select **Power > Sleep.**
- Right-click the *Windows Start button* > **Shut down or sign out > Sleep.**

Source: <https://files.bbystatic.com/HRJdX5dzFTsthH%2B%2BTB1f2w%3D%3D/A9984950-7A67-43C6-877B-876F9DA38551.pdf> (Page 6)

Preparing the computer

Before moving the computer, close and latch the display cover to place it in Sleep mode. You can now safely take the computer anywhere you go within the building. To wake the computer from Sleep mode, open the display and, if necessary, press and release the power button.

If you are taking the computer to a client's office or a different building, you may choose to shut down the computer:

Press the *Windows key*, click **Power** then select **Shut Down**

Or:

Source: <https://files.bbystatic.com/HRJdX5dzFTsthH%2B%2BTB1f2w%3D%3D/A9984950-7A67-43C6-877B-876F9DA38551.pdf> (Page 47)

39. The Acer Nitro 5 includes power providing means for providing power to the first group of circuitry, the second group of circuitry, and the third group of circuitry.

40. For example, the Acer Nitro 5 includes a rechargeable Lithium ion battery (“power providing means”) for providing power to the different circuits present in the system, including the CPU, memory, and I/O Peripherals (which include USB).

BATTERY PACK

The computer uses an embedded Lithium battery that gives you long use between charges.

Battery characteristics

The battery is recharged whenever you connect the computer to the AC adapter. Your computer supports charge-in-use, enabling you to recharge the battery while you continue to operate the computer. However, recharging with the computer turned off significantly reduces charge time.

The battery will come in handy when you travel or during a power failure.

Source: <https://files.bbystatic.com/HRJdX5dzFTsthH%2B%2BTB1f2w%3D%3D/A9984950-7A67-43C6-877B-876F9DA38551.pdf> (Page 44)

Battery Information	
Number of Cells	4-cell
Battery Chemistry	Lithium Ion (Li-Ion)
Maximum Battery Run Time	7 Hour

Source: <https://www.acer.com/ac/en/US/content/model/NH.Q5YAA.004>

41. The Acer Nitro 5 includes control means for controlling said power providing means to selectively activate said first group of circuitry, said second group of circuitry, and said third group of circuitry, so as to respectively provide said first operating function, said second operating function, and said standby function.

42. For example, the Acer Nitro 5 includes different operating modes like Power On, Sleep, and Shut down modes. Sleep mode (“standby function”), Shut Down mode, and Power On mode (which provides “first operating function”) can be activated using the Power button (“control means”). The USB port with power-off charging capability enables charging of a

mobile device through the designated USB port during Shut Down mode (“second operating function”).

43. The processor of the Acer Nitro 5 includes a Power Management Integrated Circuit (PMIC) that manages the power distribution in the processor system. The PMIC provides power to different circuits of the processor system. Further, the PMIC receives control inputs from the processor system, i.e., signals from the power button are used by PMIC as control inputs for enabling and disabling the power distribution for the circuits in the processor system.

If you need to power down the computer for a short while, but don't want to completely shut it down, you can put it to *Sleep* by doing any of the following:

- Press the power button.
- Press the sleep hotkey.
- Press the *Windows key* or select the *Windows Start button*, select **Power > Sleep.**
- Right-click the *Windows Start button* > **Shut down or sign out > Sleep.**

Source: <https://files.bbystatic.com/HRJdX5dzFTsthH%2B%2BTB1f2w%3D%3D/A9984950-7A67-43C6-877B-876F9DA38551.pdf> (Page 6)

Preparing the computer

Before moving the computer, close and latch the display cover to place it in Sleep mode. You can now safely take the computer anywhere you go within the building. To wake the computer from Sleep mode, open the display and, if necessary, press and release the power button.

If you are taking the computer to a client's office or a different building, you may choose to shut down the computer:

Press the *Windows key*, click **Power** then select **Shut Down**


Or:

Source: <https://files.bbystatic.com/HRJdX5dzFTsthH%2B%2BTB1f2w%3D%3D/A9984950-7A67-43C6-877B-876F9DA38551.pdf> (Page 47)

Note
 If you cannot power off the computer normally, press and hold the power button for up to ten seconds to shut down the computer. If you turn off the computer and want to turn it on again, wait at least two seconds before powering up.

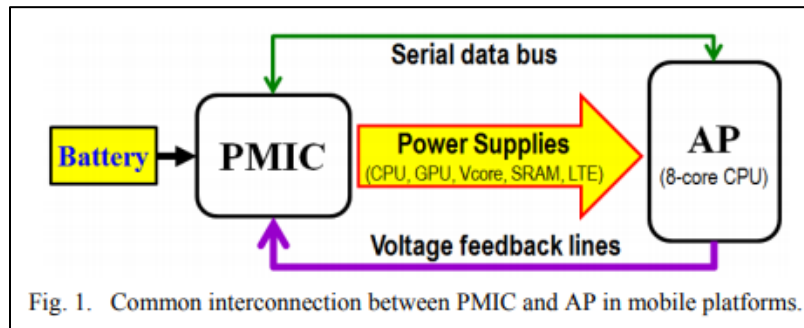
Source: <https://files.bbystatic.com/HRJdX5dzFTsthH%2B%2BTB1f2w%3D%3D/A9984950-7A67-43C6-877B-876F9DA38551.pdf> (Page 6)

USB Charge Manager allows you to charge mobile devices any time the notebook is turned on or in Sleep mode. Power-off USB charging allows you to charge mobile devices using the designated USB port, even when the notebook is off or in Hibernation mode. USB Charge Manager is only available on systems with a USB 3.0 port designated by the graphic:



USB Charge Manager has two settings which control power-off USB charging. By default, the system will charge devices when the notebook is plugged in or on battery power. If the notebook is on battery power, charging will stop when the battery reaches 30%.

Source: https://us.answers.acer.com/app/answers/detail/a_id/29968/~usb-charge-manager



Source: <https://ieeexplore.ieee.org/document/7237388>

44. Acer has had knowledge of the '573 Patent at least as of the date when it was notified of the filing of this action.

45. Liberty Patents has been damaged as a result of Acer's infringing conduct alleged above. Thus, Acer is liable to Liberty Patents in an amount that adequately compensates it for such infringements, which, by law, cannot be less than a reasonable royalty, together with interest and costs as fixed by this Court under 35 U.S.C. § 284.

46. Liberty Patents and/or its predecessors-in-interest have satisfied all statutory obligations required to collect pre-filing damages for the full period allowed by law for infringement of the '573 Patent.

COUNT III

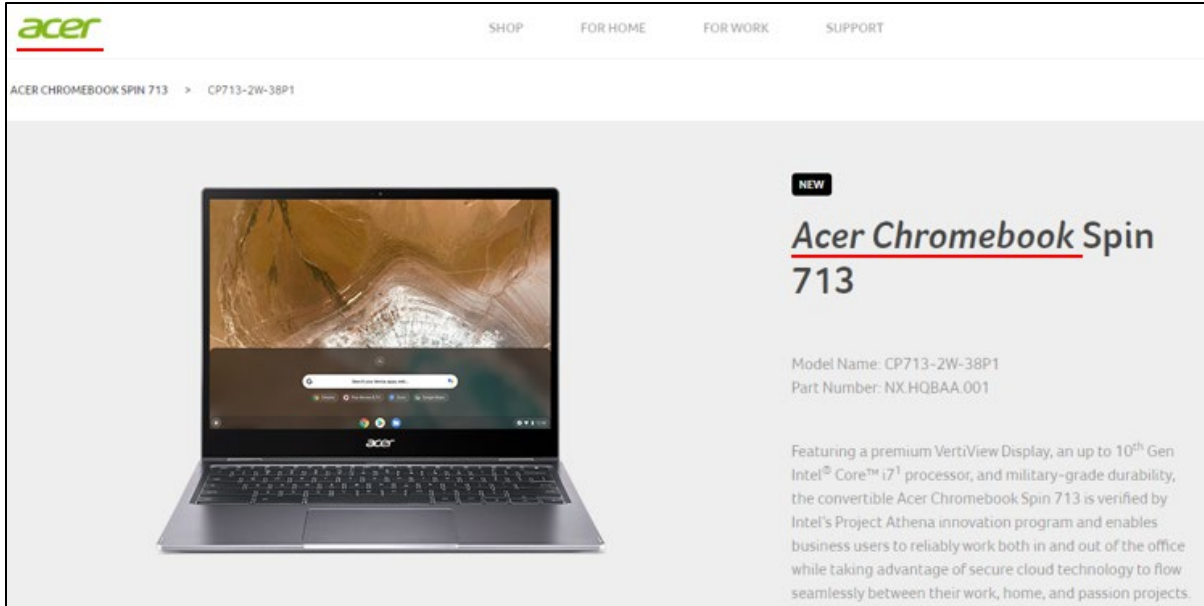
DIRECT INFRINGEMENT OF U.S. PATENT NO. 7,493,612

47. On February 17, 2009, the '612 Patent was duly and legally issued by the United States Patent and Trademark Office for an invention entitled "Embedded System and Related Method Capable of Automatically Updating System Software."

48. Liberty Patents is the owner of the '612 Patent, with all substantive rights in and to that patent, including the sole and exclusive right to prosecute this action and enforce the '612 Patent against infringers, and to collect damages for all relevant times.

49. Acer made, had made, used, imported, provided, supplied, distributed, sold, and/or offered for sale products and/or systems including, for example, its Chromebook family of products running Chrome OS,³ including the Acer Chromebook Spin 713 ("accused products"):

³ See, e.g., Chromebook Tab 10; Chromebook 11 (C720, C720P); Chromebook 11 (C740); ; Chromebook 11 (CB3-111, C730, C730E); Chromebook 11 (CB3-131, C735); Chromebook 11 N7 (C731, C731T, CB311-7H); Chromebook 11 (CB311-8H, CB311-8HT); Chromebook 11 (C732, C732T, C732L, C732LT); Chromebook 11 (C771, C771T); Chromebook 311 (C721); Chromebook 311 (C733, C733U, C733T, CB311-9H, CB311-9HT) ; Chromebook 314 (C933, C933T, CB314-1H, CB314-1HT); Chromebook 315 (CB315-2H, CB315-3H, CB315-3HT); Chromebook 512 (C851, C851T); Chromebook 13 (CB5-311, C810); Chromebook 13 (CB713-



Source: <https://www.acer.com/ac/en/US/content/model/NX.HQBAA.001>


1W); Chromebook Enterprise 13 (CB713-1W); Chromebook 14 (CB3-431); Chromebook 14 for Work (CP5-471); Chromebook 514; Chromebook 15 (CB5-571, C910); Chromebook 15 (CB3-531); Chromebook 15 (CB3-532); Chromebook 15 (CB315-1H, CB315-1HT); Chromebook 15 (CB515-1H, CB515-1HT); Chromebook 712 (C871); Chromebook 714 (CB714-1W); Chromebook Enterprise 714 (CB714-1WT); Chromebook 715 (CB715-1W); Chromebook Enterprise 715 (CB715-1WT) ; Chromebook R11 (CB5-132T, C738T); Chromebook R13 (CB5-312T); Chromebook Spin 13 (CP713-1WN); ; Chromebook Enterprise Spin 13 (CP713-1WN); Chromebook Spin 15 (CP315); Chromebook Spin 11 (R751T); Chromebook Spin 11 (CP311-H1, CP311-1HN); Chromebook Spin 311 (CP311-2H, CP311-2HN, R721T); Chromebook Spin 511 (R752T, R752TN); Chromebook Spin 512 (R851TN); Chromebook Spin 713 (CP713).

ACER CHROMEBOOK SPIN 713 > CP713-2W-38P1	
Operating System	
Operating System	Chrome OS™
Processor	
Processor Manufacturer	Intel®
Processor Type	Core™ i3
Processor Model	i3-10110U
Processor Speed	2.10 GHz
Processor Core	Dual-core (2 Core™)

Source: <https://www.acer.com/ac/en/US/content/model/NX.HQBAA.001>

50. By doing so, Acer has directly infringed (literally and/or under the doctrine of equivalents) at least Claim 1 of the '612 Patent. Acer's infringement in this regard is ongoing.

51. For example, the Acer Chromebook Spin 713 runs Chrome OS, which can automatically update the device's firmware over the internet. Chromebooks include an Embedded Controller (EC), which is responsible for power sequencing of the main CPU or the Application Processor (AP), keyboard control, thermal control, battery charging control, verified boot, etc. The EC—together with the components it controls—is an embedded system capable of automatically updating the system software. Specifically, the system includes two types of firmware: RO firmware and RW firmware. The RW firmware ("system software") is stored in the updateable section of the firmware and can be automatically updated.



Introduction

Chrome devices are computers developed by Google that run Chrome OS. What makes these computers unique is that they run in a pure web environment—they automatically update—you don't have to regularly install patches or re-image machines regularly. They boot quickly and have several security features built in.

Chrome devices can be centrally managed by the Google Admin console. You can configure over 200 settings from this web-based console, such as Wi-Fi settings, selecting apps to be pre-installed, and forcing devices to auto-update to the latest version of Chrome OS.

Source: https://services.google.com/fh/files/misc/chrome_device_deployment_guide.pdf (Page 4)

Auto Update policy

Overview

Chrome devices (e.g. Chromebook, Chromebox, Chromebase, Chromebit) receive automatic updates that enhance both the device and its software. Device updates provide the latest features and keep the device secure, and are applied across the operating system, browser and hardware. These updates depend on many device specific non-Google hardware and software providers that work with Google to provide the highest level of security and stability support. For this reason, older Chrome devices cannot receive updates indefinitely to enable new OS and browser features.

Source: <https://support.google.com/chrome/a/answer/6220366?hl=en>

ChromeOS Firmware Updater

This repository contains the firmware updater (`chromeos-firmwareupdate`) that will update firmware images related to verified boot, usually host (also known as AP, BIOS or MAIN) and EC (Embedded Controller).

Contents

- Introduction
- Using Firmware Updater
 - Update manually
 - Simulating ChromeOS Auto Update
- Building Firmware Updater
- Manipulating Firmware Updater Packages
 - CROS_FIRMWARE_MAIN_IMAGE
 - CROS_FIRMWARE_MAIN_RW_IMAGE
 - CROS_FIRMWARE_EC_IMAGE
- Technical Details
 - Packaging
 - Updater logic

Introduction

Auto update is one of the most important feature in Chrome OS. Updating firmware is one of the most complicated process, since all Chromebooks come with firmware that implemented verified boot and must be able to update in background silently.

Source:

<https://chromium.googlesource.com/chromiumos/platform/firmware/+/master/README.md>

What's an Embedded Controller anyway?

- A tiny SoC that manages battery charging, fans, keyboard, LEDs, etc.
- Typically runs even when the main system processor is off
 - We call the main system CPU the "AP" (for Application Processor)
- Most laptops have them
- Most Chromebooks do too
- Ours is open source, which is unusual

Source: https://www.coreboot.org/images/5/50/An_Open_Source_EC.pdf (Page 4).

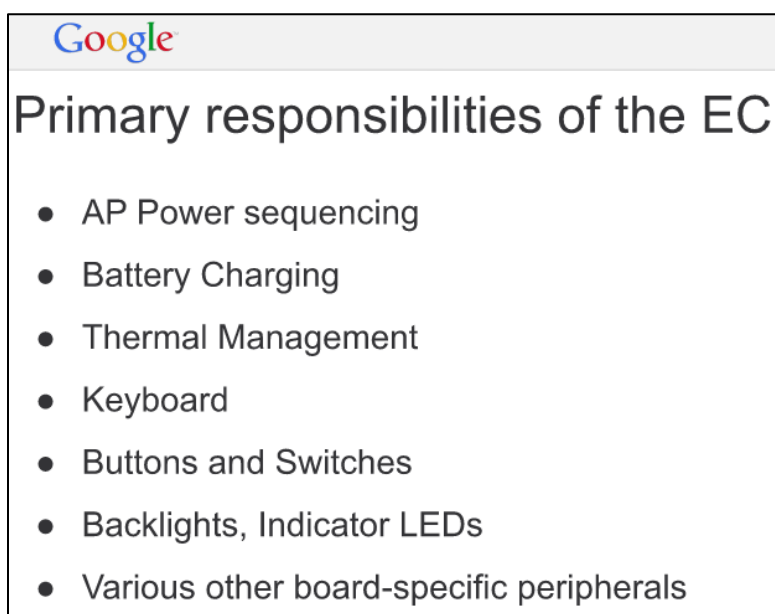
Introduction

The Chromium OS project includes open source software for embedded controllers (EC) used in recent ARM and x86 based Chromebooks. This software includes a lightweight, multitasking OS with modules for power sequencing, keyboard control, thermal control, battery charging, and verified boot. The EC software is written in C and supports [a variety of micro-controllers.](#)

This document is a guide to help make you familiar with the EC code, current features, and the process for submitting code patches.

For more see the Chrome OS Embedded Controller [presentation](#) and [video](#) from the [2014 Firmware Summit.](#)

Source: <https://chromium.googlesource.com/chromiumos/platform/ec/>



Source: https://docs.google.com/presentation/d/1Xa_Z5SjW-soPvkugAR8_TEJFrJpzoZUa9HNR14Sjs8/pub?start=false&loop=false&delayms=3000&slide=id.g2bc16935c_0142 (Slide 20).

Chrome EC

- Embedded Controllers are vital but closed
- Chrome EC is open source
 - chromiumos/platform/ec.git
- Chrome EC is designed for security
 - RO and RW regions
 - RW update is signed and handled by host firmware
 - EC Software Sync is part of Verified Boot
- Support for different ARM SOCs
 - Texas Instruments Stellaris Cortex-M4
 - ST Micro STM32 Cortex-M3
 - More in progress...

Source: https://docs.google.com/presentation/d/1h-nsDGIQmYI21dr95nYgLmyCYDgBIpJWSt9b7AqTZaw/pub?start=false&loop=false&delayms=3000&slide=id.g2b77a1dcf_298 (Slide 29).

Chrome EC Software Sync

It is important that the AP firmware (coreboot) and the EC firmware remain compatible through upgrades. During every Normal Mode boot, the EC firmware is verified by the AP firmware and updated, if required. In Recovery Mode, the EC and AP firmware stay in read-only mode.

Source: <https://link.springer.com/content/pdf/10.1007%2F978-1-4842-0070-4.pdf> (Page 119).

Terminology

RO and RW

MCUs running the EC code have read-only (RO) and read-write (RW) firmware. Coming out of reset, the MCU boots into its RO firmware.

In the case of the EC, the RO firmware boots the host and asks it verify a hash of the RW firmware (software sync). If the RW firmware is invalid, it is updated from a copy in the host's RW firmware.

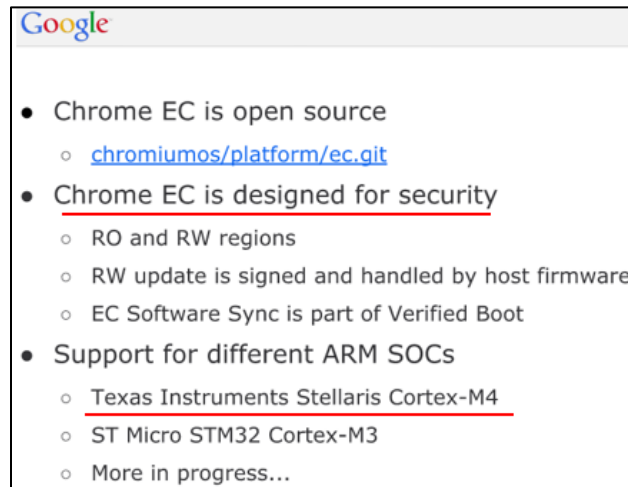
Source:

https://chromium.googlesource.com/chromiumos/platform/ec/+master/docs/write_protection.md

A feature called "Software Sync" keeps a copy of the read-write (RW) EC firmware in the RW part of the system firmware image. At boot, if the RW EC firmware doesn't match the copy in the system firmware, the EC's RW section is re-flashed.

Source: <https://chromium.googlesource.com/chromiumos/platform/ec/>

52. The Acer Chromebook Spin 713 includes a first storage device for storing a first system software and a boot image. For example, the device includes flash memory (“first storage device”) that stores the RO firmware (“boot image”) and RW firmware (“first system software”).



Source: https://docs.google.com/presentation/d/1Xa_Z5SjW-soPvkugAR8_TEJFrJpzoZUa9HNR14Sjs8/pub?start=false&loop=false&delayms=3000&slide=id.g2bbed09ac_111 (Slide 2).

Changing Software Write Protection with ectool

ectool flashprotect

Print out current flash protection state.

```
Flash protect flags: 0x0000000f wp_gpio_asserted ro_at_boot ro_now all_now
Valid flags:       0x0000003f wp_gpio_asserted ro_at_boot ro_now all_now STUCK INCONSISTENT
Writable flags:    0x00000000
```

Flash protect flags - Current flags that are set.

Valid flags - All the options for flash protection.

Writable flags - The flags that currently can be changed. (In this case, no flags can be changed).

Flags:

- `wp_gpio_asserted` - Whether the hardware write protect GPIO is currently asserted (read only).
- `ro_at_boot` - Whether the EC will write protect the RO firmware on the next boot of the EC.
- `ro_now` - Protect the read-only portion of flash immediately. Requires hardware WP be enabled.
- `all_now` - Protect the entire flash (including RW) immediately. Requires hardware WP be enabled.
- `STUCK` - Flash protection settings have been fused and can't be cleared (should not happen during normal operation. Read only.)
- `INCONSISTENT` - One or more banks of flash is not protected when it should be (should not happen during normal operation. Read only.).

Source:

https://chromium.googlesource.com/chromiumos/platform/ec/+/master/docs/write_protection.md

Firmware Image

The Chrome OS firmware image has two main sections: Read-Only (RO) and Read-Write (RW). The RO firmware is set at the factory and cannot be updated after manufacturing. The RW firmware can be updated during Chrome OS auto-update (AU).

If a problem is found in RO firmware, Google creates an update and places it in the RW firmware. During the boot process, the RO firmware checks whether there is an update in the RW section and, if so, jumps to the RW update to execute the new boot code.

The RO firmware contains the following code:

- U-Boot, including the device tree for this system
- On x86 systems: coreboot
- Google Binary Block (GBB), which contains the following:
 - Recovery screen images
 - Public keys needed to verify the RW firmware
- Firmware ID (a string with the version number and device type)

The RW firmware contains two sections: A and B. Each section contains the following:

- U-Boot, including the device tree for this system (identical to the U-Boot images in RO firmware)
- VBlock, which contains the signatures used to verify the kernel before loading and running it
- Firmware ID
- Embedded Controller image
- Fmap, a data structure that describes the layout and contents of the SPI Flash. This structure is required by the Flashrom tool.

Source: <https://www.chromium.org/chromium-os/firmware-porting-guide/2->

[concepts?tmpl=%2Fsystem%2Fapp%2Ftemplates%2Fprint%2F&showPrintDialog=1](https://www.chromium.org/chromium-os/firmware-porting-guide/2-concepts?tmpl=%2Fsystem%2Fapp%2Ftemplates%2Fprint%2F&showPrintDialog=1)

53. The RO firmware and RW firmware are stored in system memory.

BASE	SIZE	SECTION	DESCRIPTION
0x000000	0x200000	SI_ALL	Descriptor + ME
<u>0x200000</u>	<u>0x0f0000</u>	<u>RW_SECTION_A</u>	<u>Read-Write Firmware A</u>
<u>0x2f0000</u>	<u>0x0f0000</u>	<u>RW_SECTION_B</u>	<u>Read-Write Firmware B</u>
0x3e0000	0x010000	RW_MRC_CACHE	Memory Training Cache
0x3f0000	0x004000	RW_ELOG	Event Log
0x3f4000	0x004000	RW_SHARED	Shared Data
0x3f8000	0x002000	RW_VPD	Read-Write VPD
0x400000	0x200000	RW_LEGACY	Legacy Firmware
0x600000	0x004000	RO_VPD	Read-Only VPD
0x610000	0x000800	FMAP	Flash Map
0x610800	0x000040	RO_FRID	RO Firmware ID
0x611000	0x0ef000	GBB	Google Binary Block
<u>0x700000</u>	<u>0x100000</u>	<u>BOOT_STUB</u>	<u>Read-Only Firmware</u>

Source: https://docs.google.com/presentation/d/1h-nsDGIQmYI21dr95nYgLmyCYDgBIpJWSt9b7AqTZaw/pub?start=false&loop=false&delayms=3000&slide=id.g2b77a1dcf_1128 (Slide 33).

54. The Acer Chromebook Spin 713 includes a micro-controller that is coupled to the first storage device for respectively transforming the first system software and the boot image into system code and boot code. The micro-controller orderly executes the boot code and the system code to control booting of the embedded system.

55. For example, the EC (“micro-controller”) of the Acer Chromebook Spin 713 is connected to flash memory (“first storage device”) that stores both the RO firmware (“boot image”) and RW firmware (“first system software”). The firmware can have different configurations, which relates to how it uses flash and RAM memory for system code and boot code execution. For example, firmware images in Chrome OS support configurations like Internal Mapped Storage, Code Copied to RAM for Use, etc.

56. During system boot, linker scripts provide information as to how different sections of the RO firmware (“boot image”) and RW firmware (“first system software”) map to different sections of memory. The system provides linker scripts for both the RO firmware and the RW firmware in all the supported memory configurations. Before booting, a small program called Boot Loader (i.e., start-up code) uses linker scripts to load sections of the RO firmware and RW firmware into different parts of memory. The EC executes the mapped RO firmware (“boot code”) and the mapped RW firmware (“system code”) only after they are loaded into specified sections of memory determined by linker scripts. Accordingly, the RO firmware (“boot image”) and RW firmware (“first system software”) are transformed into memory-mapped executable code (i.e., “boot code” and “first system code”) after being mapped and loaded into their respective sections of memory.

EC Image Geometry Spec

Introduction

The EC codebase currently supports the following chips:

lm4, stm32: Internal memory-mapped flash storage

cr50: Internal memory-mapped flash storage, with image signature preceding RO image

npcx: External memory-mapped flash storage dedicated for EC, code copied to SRAM before execution

mec1322: External unmapped flash storage dedicated for EC, code is copied to SRAM before execution

For memory-mapped flash storage, contents are read from a chip-defined region in memory. Contents are written through SPI commands.

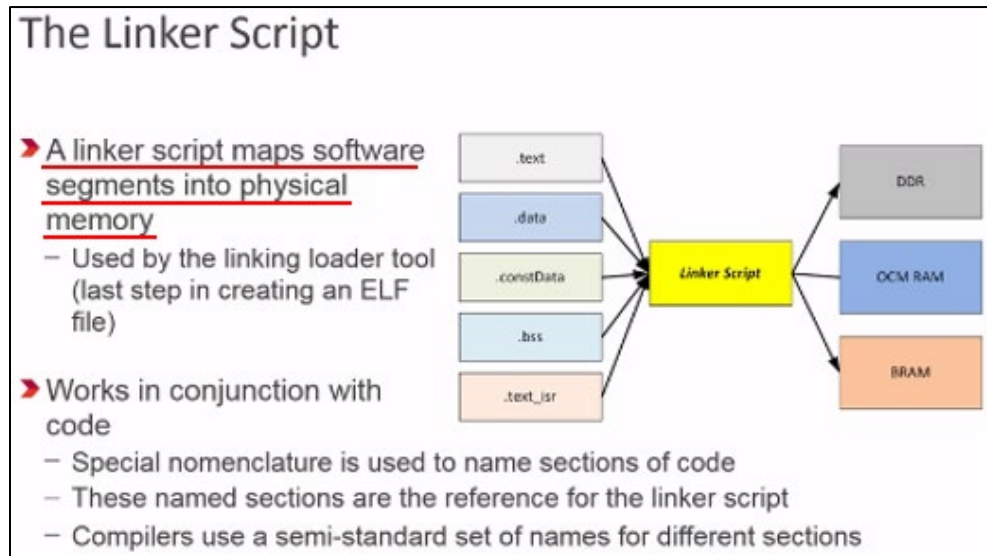
Source: <https://www.chromium.org/chromium-os/ec-development/ec-image-geometry-spec>

Supported Configurations

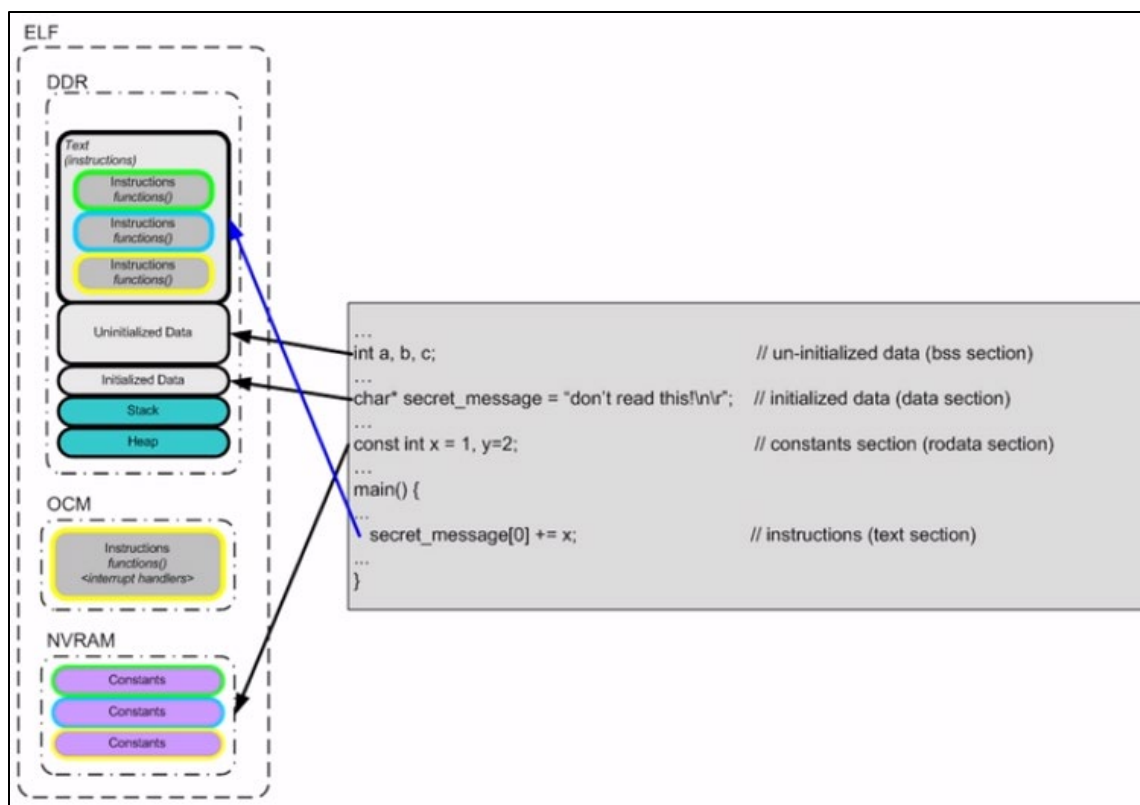
With the changes proposed here, we aim to support ECs with the following configurations:

- Internal mapped storage
 - External mapped storage (shared or dedicated)
 - External unmapped storage (shared or dedicated)
 - Code executed directly from mapped storage
 - Code copied to SRAM before use
-
- One RO image
 - One contiguous region of storage belonging to the EC, containing the RO image, that can be write protected
 - Up to one RW image
 - Up to one contiguous region of storage belonging to the EC, containing the RW image, that will not be write protected
 - Support for any number of chip-specific non-image data pieces (*NIDs*) as part of the storage regions (loaders, headers, etc - any piece of data on EC storage that isn't part of the RO or RW image)
 - Write protected and writable EC storage regions need not be mutually contiguous

Source: <https://www.chromium.org/chromium-os/ec-development/ec-image-geometry-spec>



Source: <https://www.xilinx.com/training/customer-training/using-linker-scripts.html> (2:20)



Source: <https://www.xilinx.com/training/customer-training/using-linker-scripts.html> (9:52)

Linking

Our linker scripts were originally written to support internal memory-mapped storage with only RO and RW images (no NIDs). The scripts will be extended to support the new configs defined above. For example, the linker will produce a unified image with the RW image at the appropriate offset, as defined by CONFIG_RW_STORAGE_OFF / CONFIG_RW_STORAGE_SIZE.

Source: <https://www.chromium.org/chromium-os/ec-development/ec-image-geometry-spec>

The microcontroller boot process starts by simply applying power to the system. Once the voltage rails stabilize, the microcontroller looks to the reset vector for the location in flash where the start-up instruction can be found. The reset vector is a special location within the flash memory

Source: <https://www.benigo.com/understanding-the-microcontroller-boot-process/>

The address that is stored at the reset vector is loaded by the microcontroller and the instructions that are contained there are then loaded and executed by the CPU. Now these first instructions aren't the start of main that the developer created. Instead, these are instructions on how to start-up the microcontroller.

The first thing that usually occurs is that the vector tables that are stored in flash are copied to RAM. They are copied from and to the location that is specified in the linker file at the time the executable program is created. One reason for copying the vector tables to RAM is that it is faster to execute from RAM than flash. This helps to decrease the latency of any interrupt calls within the system. Depending on the particular architecture of the microcontroller there may then be an instruction to update a vector table register so that the microcontroller knows where the start of the RAM table is.

Next the initialized data sections are copied into RAM. This is usually variables that are stored in the .data section of the linker. Examples of initialized data would be static, global and static local variables that have been provided with an initialization value during compile time. These are explicit definitions such as `int Var = 0x32;`

Following the copy of the data section, the .bss section is also copied. The .bss section contains variables that are not initialized explicitly or that have been initialized to a value of zero. A simple example is that the variable `static int Var;` would be contained within this section.

Finally, the microcontroller will copy any RAM functions from flash to RAM. Once again it is sometimes worthwhile to execute certain functions out of RAM rather than flash due to the execution speed being slightly faster. These are functions usually decided upon by the developer and purposely placed there in the linker file prior to compiling the program.

Source: <https://www.benigo.com/understanding-the-microcontroller-boot-process/>

57. During initialization and booting of the EC (“micro-controller”), the executable version of the RO firmware (“boot code”) is executed first. The RO firmware then runs the executable version of the RW firmware (“first system code”). Accordingly, the boot code and system code are orderly executed to control booting of the embedded system.

EC Software Sync

It is important that the AP firmware (BIOS) and the EC firmware remain compatible through upgrades. At every* cold boot/reset of the EC

1. The EC boots its RO firmware, and powers on the AP.
2. The AP boots its RO firmware.
3. The AP verifies its RW firmware and jumps to it.
4. The EC computes a hash of its RW firmware.
5. The AP RW firmware contains a copy of the EC's RW firmware. The AP compares its hash with the EC's hash.
6. If they differ, the AP gives the EC the correct RW firmware, which the EC writes to its flash.
7. The EC jumps to its RW firmware.

There also are a few other tricks to ensure the EC isn't lying about its hash

*Normal mode, anyway. In recovery mode both AP and EC stay in their RO firmware

Source: https://www.coreboot.org/images/5/50/An_Open_Source_EC.pdf (Page 22).

Firmware Image

The Chrome OS firmware image has two main sections: Read-Only (RO) and Read-Write (RW). The RO firmware is set at the factory and cannot be updated after manufacturing. The RW firmware can be updated during Chrome OS auto-update (AU).

If a problem is found in RO firmware, Google creates an update and places it in the RW firmware. During the boot process, the RO firmware checks whether there is an update in the RW section and, if so, jumps to the RW update to execute the new boot code.

The RO firmware contains the following code:

- U-Boot, including the device tree for this system
- On x86 systems: coreboot
- Google Binary Block (GBB), which contains the following:
 - Recovery screen images
 - Public keys needed to verify the RW firmware
- Firmware ID (a string with the version number and device type)

The RW firmware contains two sections: A and B. Each section contains the following:

- U-Boot, including the device tree for this system (identical to the U-Boot images in RO firmware)
- VBlock, which contains the signatures used to verify the kernel before loading and running it
- Firmware ID
- Embedded Controller image
- Fmap, a data structure that describes the layout and contents of the SPI Flash. This structure is required by the Flashrom tool.

Source: <https://www.chromium.org/chromium-os/firmware-porting-guide/2-concepts?tmpl=%2Fsystem%2Fapp%2Ftemplates%2Fprint%2F&showPrintDialog=1>

BASE	SIZE	SECTION	DESCRIPTION
0x000000	0x200000	SI_ALL	Descriptor + ME
0x200000	0x0f0000	RW_SECTION_A	Read-Write Firmware A
0x2f0000	0x0f0000	RW_SECTION_B	Read-Write Firmware B
0x3e0000	0x010000	RW_MRC_CACHE	Memory Training Cache
0x3f0000	0x004000	RW_ELOG	Event Log
0x3f4000	0x004000	RW_SHARED	Shared Data
0x3f8000	0x002000	RW_VPD	Read-Write VPD
0x400000	0x200000	RW_LEGACY	Legacy Firmware
0x600000	0x004000	RO_VPD	Read-Only VPD
0x610000	0x000800	FMAP	Flash Map
0x610800	0x000040	RO_FRID	RO Firmware ID
0x611000	0x0ef000	GBB	Google Binary Block
0x700000	0x100000	BOOT_STUB	Read-Only Firmware

Source: https://docs.google.com/presentation/d/1h-nsDGIQmYI21dr95nYgLmyCYDgBIpJWSt9b7AqTZaw/pub?start=false&loop=false&delayms=3000&slide=id.g2b77a1dcf_1128 (Slide 33).

58. The Acer Chromebook Spin 713 includes a connecting interface that is coupled to the micro-controller and further coupled to an external data storage device through a data transmission media. The external data storage device stores a second system software.

59. For example, the EC (“micro-controller”), which controls peripheral connections of the Acer Chromebook Spin 713 like USB, Wi-Fi, etc., receives an updated version of the RW firmware (“second system software”) from the device’s Application Processor (AP). Specifically, the updated RW firmware (“second system software”) is read from the network server when the device is connected to the internet through a Wi-Fi network interface. Accordingly, the EC (“micro-controller”) is configured to be coupled to the network server (“external data storage device”) through the Wi-Fi network interface (“connecting interface”).

The network server stores the RW firmware update (“second system software”), which can be read by the device via the internet (“data transmission media”).

EC Software Sync


It is important that the AP firmware (BIOS) and the EC firmware remain compatible through upgrades. At every* cold boot/reset of the EC

1. The EC boots its RO firmware, and powers on the AP.
2. The AP boots its RO firmware.
3. The AP verifies its RW firmware and jumps to it.
4. The EC computes a hash of its RW firmware.
5. The AP RW firmware contains a copy of the EC's RW firmware. The AP compares its hash with the EC's hash.
6. If they differ, the AP gives the EC the correct RW firmware, which the EC writes to its flash.
7. The EC jumps to its RW firmware.

There also are a few other tricks to ensure the EC isn't lying about its hash

*Normal mode, anyway. In recovery mode both AP and EC stay in their RO firmware

Source: https://www.coreboot.org/images/5/50/An_Open_Source_EC.pdf (Page 22).



Power Sequencing

- Each AP family has its own
 - Power states
 - Voltage regulators
 - Control GPIOs (both input and output)
 - Transition rules
 - Timing requirements
 - Trigger events
- The EC must manage and respond to all those requirements as the AP boots, sleeps, idles, or transitions between various subtle states.
- It must also ensure that certain peripherals are brought up and down accordingly (USB, WiFi, etc.)

Source: https://docs.google.com/presentation/d/1Xa_Z5SjW-soPvkugAR8_TEJFrJpzoZUa9HNR14Sjs8/pub?start=false&loop=false&delayms=3000&slide=id.g2bbed09ac_142 (Slide 21).

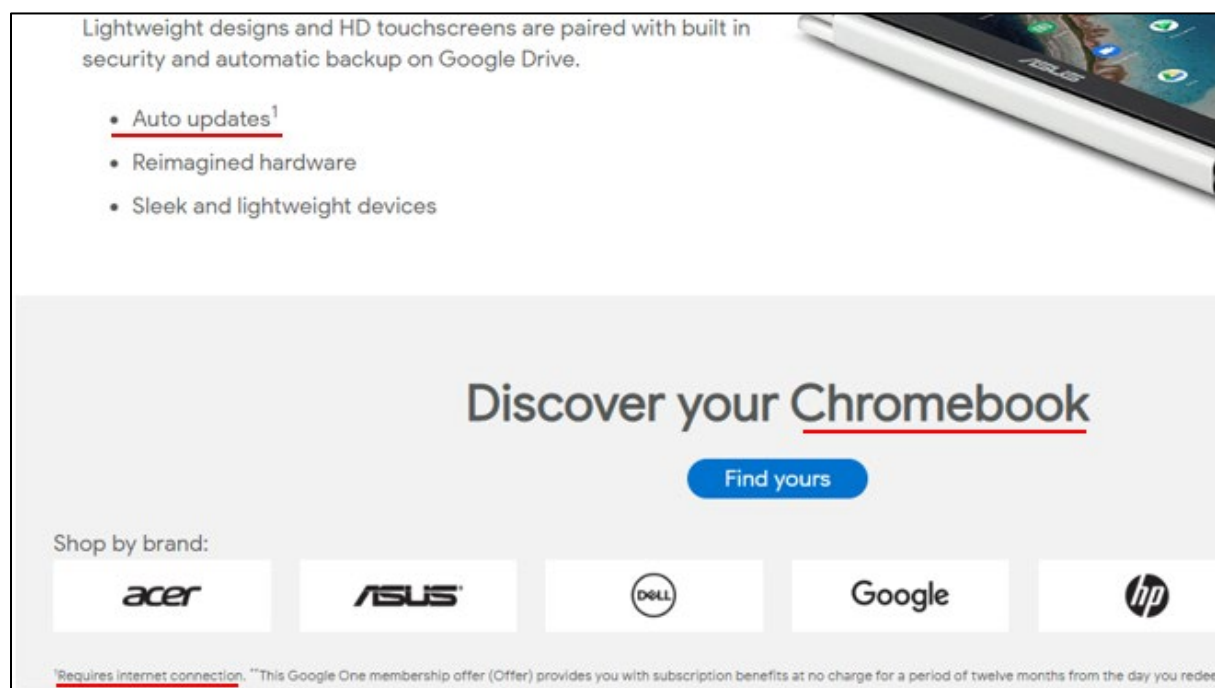
3. Verified Boot

Chromebook's startup is very different from Windows or Mac machines. When Chrome OS boots, it compares every component of the operating system with the current version verified by Google through the Internet. If there is a discrepancy, it will replace with the up-to-date version. Every time the Chromebook starts up, it does the self-check called "Verified Boot."

The self-check ensures Chrome OS in the right shape; it plays a fundamental role in Chromebook security mechanism.

- Drive automatic update: download new updates of Chrome OS when Verified Boot;
- Repair corrupted OS: take Chrome OS back if malware manages to escape the Sandbox;

Source: <https://www.keepds.com/tool/list?os=c>



Lightweight designs and HD touchscreens are paired with built in security and automatic backup on Google Drive.

- Auto updates¹
- Reimagined hardware
- Sleek and lightweight devices

Discover your Chromebook

Find yours

Shop by brand:

acer ASUS DELL Google hp

¹Requires internet connection. ²This Google One membership offer (Offer) provides you with subscription benefits at no charge for a period of twelve months from the day you redeem.

Source: https://www.walmart.ca/en/electronics/laptops-computers/laptops-notebooks/chromebooks/google/N-1990+1000268?mtr=mdv_00439&icid=electronics_wmg_display_walmart_l4hb_wk16_google_chromebook_en

Google rolls out security updates as soon as they're ready and applies them when a Chromebook boots up. In other words, during the boot sequence, Chrome OS checks to see if a new update is available. If yes, installs it without interrupting the user. At this point, your Internet has to be available.

Source: <https://www.keepds.com/tool/list?os=c>

If peer-to-peer (P2P) networking is available, devices can automatically update Chrome from nearby devices of the same model. This option reduces external network traffic. If P2P automatic updating fails or isn't possible on your network, devices update as usual. They either download the update from Google's servers or an intermediate web-caching proxy server.

Source: <https://support.google.com/chrome/a/answer/3168106?hl=en>

Firmware Image

The Chrome OS firmware image has two main sections: Read-Only (RO) and Read-Write (RW). The RO firmware is set at the factory and cannot be updated after manufacturing. The RW firmware can be updated during Chrome OS auto-update (AU).

If a problem is found in RO firmware, Google creates an update and places it in the RW firmware. During the boot process, the RO firmware checks whether there is an update in the RW section and, if so, jumps to the RW update to execute the new boot code.

The RO firmware contains the following code:

- U-Boot, including the device tree for this system
- On x86 systems: coreboot
- Google Binary Block (GBB), which contains the following:
 - Recovery screen images
 - Public keys needed to verify the RW firmware
- Firmware ID (a string with the version number and device type)

The RW firmware contains two sections: A and B. Each section contains the following:

- U-Boot, including the device tree for this system (identical to the U-Boot images in RO firmware)
- VBlock, which contains the signatures used to verify the kernel before loading and running it
- Firmware ID
- Embedded Controller image
- Fmap, a data structure that describes the layout and contents of the SPI Flash. This structure is required by the Flashrom tool.

Source: <https://www.chromium.org/chromium-os/firmware-porting-guide/2-concepts?tmpl=%2Fsystem%2Fapp%2Ftemplates%2Fprint%2F&showPrintDialog=1>

60. The Acer Chromebook Spin 713 includes boot code, which includes update agent interface programming (UAIP). The micro-controller is capable of executing the update agent interface programming to read the second system software from the external data storage device through the connecting interface before executing the system code.

61. During the boot process of the system, the executable version of the RO firmware (“boot code”) is executed before the RW firmware. And before the RW firmware (“first system code”) is executed, the EC initiates a Software Sync process by powering on and booting the device’s AP. The system verifies and if needed, updates the RW firmware by comparing the version of the RW firmware. If there is an available update to the RW firmware (“second system software”), the update is sent to flash memory. Accordingly, the RO firmware (“boot code”) includes code or programming (“update agent interface programming”) to initiate the software sync process that updates the RW firmware.

EC Software Sync

It is important that the AP firmware (BIOS) and the EC firmware remain compatible through upgrades. At every* cold boot/reset of the EC

1. The EC boots its RO firmware, and powers on the AP.
2. The AP boots its RO firmware.
3. The AP verifies its RW firmware and jumps to it.
4. The EC computes a hash of its RW firmware.
5. The AP RW firmware contains a copy of the EC’s RW firmware. The AP compares its hash with the EC’s hash.
6. If they differ, the AP gives the EC the correct RW firmware, which the EC writes to its flash.
7. The EC jumps to its RW firmware.

There also are a few other tricks to ensure the EC isn’t lying about its hash

*Normal mode, anyway. In recovery mode both AP and EC stay in their RO firmware

Source: https://www.coreboot.org/images/5/50/An_Open_Source_EC.pdf (Page 22).

U-Boot and Embedded Controller

U-Boot performs the device initialization for the system, as follows:

1. U-Boot calls the Vblnit() function to start verified boot.
2. In the simplest case, the boot consists of one step: running the code located in the Read-Only (RO) section of the firmware. During this process, U-Boot checks whether there are any updates in the Read/Write (RW) section of the firmware. If updates are present, U-Boot loads and runs RW firmware.
3. The main firmware performs a software sync that checks whether the EC code needs to be updated. If so, the update is sent over I2C, SPI, or LPC to the EC.
4. Once the EC code has been sync’ed, execution jumps to the EC code in the RW firmware.
5. The kernel is loaded and verified.
6. The correct device tree file is selected for the system (ARM only).
7. In addition, the kernel contains command line information that U-Boot picks up and passes to the kernel for use during boot. The command line tells the kernel which device to boot from (eMMC, SD, USB) and also contains the Verity parameters that are passed in to the kernel at boot time.
8. The system boots the kernel, which uses the root hash (contained in the Verity parameters) to open the root filesystem.
9. The system initializes user space and runs X and Chrome.

Source: <https://www.chromium.org/chromium-os/firmware-porting-guide/2-concepts?tmpl=%2Fsystem%2Fapp%2Ftemplates%2Fprint%2F&showPrintDialog=1>

3. Verified Boot

Chromebook's startup is very different from Windows or Mac machines. When Chrome OS boots, it compares every component of the operating system with the current version verified by Google through the Internet. If there is a discrepancy, it will replace with the up-to-date version. Every time the Chromebook starts up, it does the self-check called "Verified Boot."

The self-check ensures Chrome OS in the right shape; it plays a fundamental role in Chromebook security mechanism.

- Drive automatic update: download new updates of Chrome OS when Verified Boot;
- Repair corrupted OS: take Chrome OS back if malware manages to escape the Sandbox;

Source: <https://www.keepds.com/tool/list?os=c>

Terminology

RO and RW

MCUs running the EC code have read-only (RO) and read-write (RW) firmware. Coming out of reset, the MCU boots into its RO firmware.

In the case of the EC, the RO firmware boots the host and asks it verify a hash of the RW firmware (software sync). If the RW firmware is invalid, it is updated from a copy in the host's RW firmware.

Source:

https://chromium.googlesource.com/chromiumos/platform/ec/+master/docs/write_protection.md

A feature called "Software Sync" keeps a copy of the read-write (RW) EC firmware in the RW part of the system firmware image. At boot, if the RW EC firmware doesn't match the copy in the system firmware, the EC's RW section is re-flashed.

Source: <https://chromium.googlesource.com/chromiumos/platform/ec/>

62. During the Verified Boot process, the system checks for updates to the RW firmware. The RO firmware includes programming that verifies and compares the version of the RW firmware on the Acer Chromebook Spin 713 with the version of the RW firmware on the

network server. The RO firmware further includes programming that updates the RW firmware based on that comparison. The Software Sync process is a part of the Verified Boot process. It includes programming that verifies and compares version of the RW firmware.

Chrome EC

- Embedded Controllers are vital but closed
- Chrome EC is open source
 - chromiumos/platform/ec.git
- **Chrome EC is designed for security**
 - RO and RW regions
 - RW update is signed and handled by host firmware
 - EC Software Sync is part of Verified Boot
- Support for different ARM SOCs
 - Texas Instruments Stellaris Cortex-M4
 - ST Micro STM32 Cortex-M3
 - More in progress...

Source: https://docs.google.com/presentation/d/1h-nsDGIQmYI21dr95nYgLmyCYDgBIpJWSt9b7AqTZaw/pub?start=false&loop=false&delayms=3000&slide=id.g2b77a1dcf_298 (Slide 29).


```

VbError_t VbEcSoftwareSync(VbCommonParams *cparams)
{
    VbSharedDataHeader *shared =
        (VbSharedDataHeader *)cparams->shared_data_blob;
    int in_rw = 0;
    int rv;
    const uint8_t *ec_hash = NULL;
    int ec_hash_size;
    const uint8_t *rw_hash = NULL;
    int rw_hash_size;
    const uint8_t *expected = NULL;
    int expected_size;
    uint8_t expected_hash[SHA256_DIGEST_SIZE];
    int need_update = 0;
    int i;

    /* Determine whether the EC is in RO or RW */
    rv = VbExEcRunningRW(&in_rw);

```

Source: https://chromium.googlesource.com/chromiumos/platform/vboot_reference/+/factory-spring-4262.B/firmware/lib/vboot_api_kernel.c

```

/* Get hash of EC-RW */
rv = VbExEcHashRW(&ec_hash, &ec_hash_size);
if (rv) {
    VBDEBUG(("VbEcSoftwareSync() - "
            "VbExEcHashRW() returned %d\n", rv));
    VbSetRecoveryRequest(VBNV_RECOVERY_EC_HASH_FAILED);
    return VBERROR_EC_REBOOT_TO_RO_REQUIRED;
}
if (ec_hash_size != SHA256_DIGEST_SIZE) {
    VBDEBUG(("VbEcSoftwareSync() - "
            "VbExEcHashRW() says size %d, not %d\n",
            ec_hash_size, SHA256_DIGEST_SIZE));
    VbSetRecoveryRequest(VBNV_RECOVERY_EC_HASH_SIZE);
    return VBERROR_EC_REBOOT_TO_RO_REQUIRED;
}

VBDEBUG(("EC hash:"));
for (i = 0; i < SHA256_DIGEST_SIZE; i++)
    VBDEBUG(("0x%02x", ec_hash[i]));
VBDEBUG((" \n"));

```

Source: https://chromium.googlesource.com/chromiumos/platform/vboot_reference/+/factory-spring-4262.B/firmware/lib/vboot_api_kernel.c

```

/*
 * Get expected EC-RW image if we're sure we need to update (because the
 * expected hash didn't match the EC) or we still don't know (because
 * there was no expected hash and we need the image to compute one
 * ourselves).
 */
if (need_update || !rw_hash) {
    /* Get expected EC-RW image */
    rv = VbExEcGetExpectedRW(shared->firmware_index ?
                            VB_SELECT_FIRMWARE_B :
                            VB_SELECT_FIRMWARE_A,
                            &expected, &expected_size);

    if (rv) {
        VBDEBUG(("VbEcSoftwareSync() - "
                "VbExEcGetExpectedRW() returned %d\n", rv));
        VbSetRecoveryRequest(VBNV_RECOVERY_EC_EXPECTED_IMAGE);
        return VBERROR_EC_REBOOT_TO_RO_REQUIRED;
    }
    VBDEBUG(("VbEcSoftwareSync() - expected len = %d\n",
            expected_size));

    /* Hash expected image */
    internal_SHA256(expected, expected_size, expected_hash);
    VBDEBUG(("Computed hash of expected image:"));
    for (i = 0; i < SHA256_DIGEST_SIZE; i++)
        VBDEBUG(("0x", expected_hash[i]));
    VBDEBUG((" \n"));
}

```

Source: https://chromium.googlesource.com/chromiumos/platform/vboot_reference/+/factory-spring-4262.B/firmware/lib/vboot_api_kernel.c

```

/*
 * We need to update, but the expected EC image doesn't match
 * the expected EC hash we were given.
 */
VBDEBUG(("VbEcSoftwareSync() - "
        "VbExEcGetExpectedRW() returned %d\n", rv));
VbSetRecoveryRequest(VBNV_RECOVERY_EC_HASH_MISMATCH);
return VBERROR_EC_REBOOT_TO_RO_REQUIRED;

```

Source: https://chromium.googlesource.com/chromiumos/platform/vboot_reference/+/factory-spring-4262.B/firmware/lib/vboot_api_kernel.c


```

/* Update EC if necessary */
if (need_update) {
    VBDEBUG(("VbEcSoftwareSync() updating EC-RW...\n"));

    if (shared->flags & VBSD_EC_SLOW_UPDATE) {
        VBDEBUG(("VbEcSoftwareSync() - "
            "EC is slow. Show WAIT screen.\n"));

        /*
         * FIXME(crosbug.com/p/12257): Ensure the VGA Option
         * ROM is loaded!
         */
        VbDisplayScreen(cparams, VB_SCREEN_WAIT, 0, &vnc);
    }


    rv = VbExEcUpdateRW(expected, expected_size);
    if (rv == VBERROR_EC_REBOOT_TO_RO_REQUIRED) {
        /*
         * Reboot required. May need to unprotect RW before
         * updating, or may need to reboot after RW updated.
         * Either way, it's not an error requiring recovery
         * mode.
         */
        VBDEBUG(("VbEcSoftwareSync() - "
            "VbExEcUpdateRW() needs reboot\n"));
    }
    return rv;
}

```

Source: https://chromium.googlesource.com/chromiumos/platform/vboot_reference/+/factory-spring-4262.B/firmware/lib/vboot_api_kernel.c

63. The EC controls peripheral connections like USB, Wi-Fi, etc. During the Verified Boot process, the Acer Chromebook Spin 713 receives updates of the RW firmware from the network server (“external data storage device”) when connected to the internet through Wi-Fi, for example. The RW firmware is updated during the Verified Boot process.

64. The EC initiates the process of updating the RW firmware by activating the Wi-Fi network interface, connecting to the internet, and starting the Software Sync process. Accordingly, the system includes code or programming (“update agent interface programming”) to read the RW firmware update (“second system software”) from the network server (“external data storage device”) before executing the current version of the RW firmware (“first system code”).



Power Sequencing

- Each AP family has its own
 - Power states
 - Voltage regulators
 - Control GPIOs (both input and output)
 - Transition rules
 - Timing requirements
 - Trigger events
- The EC must manage and respond to all those requirements as the AP boots, sleeps, idles, or transitions between various subtle states.
- It must also ensure that certain peripherals are brought up and down accordingly (USB, WiFi, etc.)

Source: https://docs.google.com/presentation/d/1Xa_Z5SjW-soPvkugAR8_TEJFrJpzoZUa9HNR14Sjs8/pub?start=false&loop=false&delayms=3000&slide=id.g2bbed09ac_142 (Slide 21).

3. Verified Boot

Chromebook's startup is very different from Windows or Mac machines. When Chrome OS boots, it compares every component of the operating system with the current version verified by Google through the Internet. If there is a discrepancy, it will replace with the up-to-date version. Every time the Chromebook starts up, it does the self-check called "Verified Boot."

The self-check ensures Chrome OS in the right shape; it plays a fundamental role in Chromebook security mechanism.

- Drive automatic update: download new updates of Chrome OS when Verified Boot;
- Repair corrupted OS: take Chrome OS back if malware manages to escape the Sandbox;

Source: <https://www.keepds.com/tool/list?os=c>

Lightweight designs and HD touchscreens are paired with built in security and automatic backup on Google Drive.

- Auto updates¹
- Reimagined hardware
- Sleek and lightweight devices

Discover your Chromebook

Find yours

Shop by brand:

acer ASUS DELL Google hp

¹Requires internet connection. **This Google One membership offer (Offer) provides you with subscription benefits at no charge for a period of twelve months from the day you redeem

Source: www.walmart.ca/en/electronics/laptops-computers/laptops-notebooks/chromebooks/google/N-1990+1000268?mtr=mdv_00439&icid=electronics_wmg_display_walmart_l4hb_wk16_google_chromebook_en

Google rolls out security updates as soon as they're ready and applies them when a Chromebook boots up. In other words, during the boot sequence, Chrome OS checks to see if a new update is available. If yes, installs it without interrupting the user. At this point, your Internet has to be available.

Source: <https://www.keepds.com/tool/list?os=c>

If peer-to-peer (P2P) networking is available, devices can automatically update Chrome from nearby devices of the same model. This option reduces external network traffic. If P2P automatic updating fails or isn't possible on your network, devices update as usual. They either download the update from Google's servers or an intermediate web-caching proxy server.

Source: <https://support.google.com/chrome/a/answer/3168106?hl=en>

Firmware Image

The Chrome OS firmware image has two main sections: Read-Only (RO) and Read-Write (RW). The RO firmware is set at the factory and cannot be updated after manufacturing. The RW firmware can be updated during Chrome OS auto-update (AU).

If a problem is found in RO firmware, Google creates an update and places it in the RW firmware. During the boot process, the RO firmware checks whether there is an update in the RW section and, if so, jumps to the RW update to execute the new boot code.

The RO firmware contains the following code:

- U-Boot, including the device tree for this system
- On x86 systems: coreboot
- Google Binary Block (GBB), which contains the following:
 - Recovery screen images
 - Public keys needed to verify the RW firmware
- Firmware ID (a string with the version number and device type)

The RW firmware contains two sections: A and B. Each section contains the following:

- U-Boot, including the device tree for this system (identical to the U-Boot images in RO firmware)
- VBlock, which contains the signatures used to verify the kernel before loading and running it
- Firmware ID
- Embedded Controller image
- Fmap, a data structure that describes the layout and contents of the SPI Flash. This structure is required by the Flashrom tool.

Source: <https://www.chromium.org/chromium-os/firmware-porting-guide/2-concepts?tmpl=%2Fsystem%2Fapp%2Ftemplates%2Fprint%2F&showPrintDialog=1>

65. Acer has had knowledge of the '612 Patent at least as of the date when it was notified of the filing of this action.

66. Liberty Patents has been damaged as a result of the infringing conduct by Acer alleged above. Thus, Acer is liable to Liberty Patents in an amount that adequately compensates it for such infringements, which, by law, cannot be less than a reasonable royalty, together with interest and costs as fixed by this Court under 35 U.S.C. § 284.

67. Liberty Patents and/or its predecessors-in-interest have satisfied all statutory obligations required to collect pre-filing damages for the full period allowed by law for infringement of the '612 Patent.

ADDITIONAL ALLEGATIONS REGARDING INFRINGEMENT

68. Acer has also indirectly infringed the '959 Patent, the '573 Patent, and the '612 Patent by inducing others to directly infringe the '959 Patent, the '573 Patent, and the '612 Patent. Acer has induced the end-users, Acer's customers, to directly infringe (literally and/or under the doctrine of equivalents) the '959 Patent, the '573 Patent, and the '612 Patent by using the accused products.

69. Acer took active steps, directly and/or through contractual relationships with others, with the specific intent to cause them to use the accused products in a manner that infringes one or more claims of the patents-in-suit, including, for example, claim 1 of the '959 Patent, claim 13 of the '573 Patent, and claim 1 of the '612 Patent.

70. Such steps by Acer included, among other things, advising or directing customers and end-users to use the accused products in an infringing manner; advertising and promoting the use of the accused products in an infringing manner; and/or distributing instructions that guide users to use the accused products in an infringing manner.

71. Acer performed these steps, which constitute induced infringement, with the knowledge of the '959 Patent, the '573 Patent, and the '612 Patent and with the knowledge that the induced acts constitute infringement.

72. Acer was and is aware that the normal and customary use of the accused products by Acer's customers would infringe the '959 Patent, the '573 Patent, and the '612 Patent. Acer's inducement is ongoing.

73. Acer has also induced its affiliates, or third-party manufacturers, shippers, distributors, retailers, or other persons acting on its or its affiliates' behalf, to directly infringe (literally and/or under the doctrine of equivalents) the '959 Patent, the '573 Patent, and the '612 Patent by importing, selling or offering to sell the accused products, including, for example, Amazon, Best Buy, CDW, East Texas Copy Systems, Fry's Electronics, Micro Center, Office Depot, Staples, Target, Walmart, and others. *See, e.g.,* www.acer.com/ac/en/US/content/online-stores; www.acer.com/ac/en/US/content/wheretobuy.

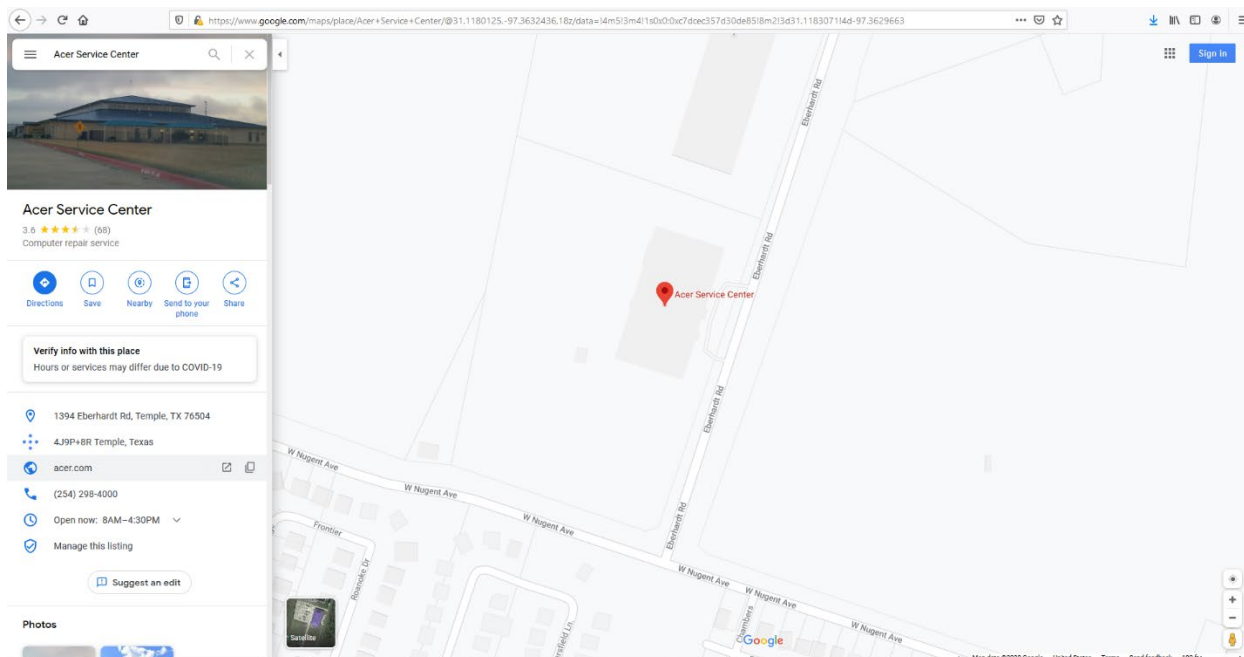
74. Acer has a significant role in placing the accused products in the stream of commerce with the expectation and knowledge that they will be purchased by consumers in Texas and elsewhere in the United States.

75. Acer purposefully directs or controls the making of accused products and their shipment to the United States, using established distribution channels, for sale in Texas and elsewhere within the United States.

76. Acer purposefully directs or controls the sale of the accused products into established United States distribution channels, including sales to nationwide retailers. Acer's established United States distribution channels include one or more United States based affiliates (e.g., Acer America Corporation).

77. Acer purposefully directs or controls the sale of the accused products online and in nationwide retailers, including for sale in Texas and elsewhere in the United States, and expects and intends that the accused products will be so sold.

78. Acer purposefully places the accused products—whether by itself or through subsidiaries—into an international supply chain, knowing that the accused products will be sold in the United States, including Texas. Therefore, Acer also facilitates the sale of the accused products in Texas. For example, Acer operates, through its affiliate, a service center in this state, located at 1394 Eberhardt Rd., Temple, Texas, 76504:



Source: <https://goo.gl/maps/FjJHwcmQhUupVKy26>



Source: <https://goo.gl/maps/3LzoSQJUobagAmF68>



Source: <https://goo.gl/maps/3LzoSQJUobagAmF68>

79. Acer took active steps, directly and/or through contractual relationships with others, with the specific intent to cause such persons to import, sell, or offer to sell the accused products in a manner that infringes one or more claims of the patents-in-suit, including, for example, claim 1 of the '959 Patent, claim 13 of the '573 Patent, and claim 1 of the '612 Patent.

80. Such steps by Acer included, among other things, making or selling the accused products outside of the United States for importation into or sale in the United States, or knowing that such importation or sale would occur; and directing, facilitating, or influencing its affiliates, or third-party manufacturers, shippers, distributors, retailers, or other persons acting on its or their behalf, to import, sell, or offer to sell the accused products in an infringing manner.

81. Acer performed these steps, which constitute induced infringement, with the knowledge of the '959 Patent, the '573 Patent, and the '612 Patent and with the knowledge that the induced acts would constitute infringement.

82. Acer performed such steps in order to profit from the eventual sale of the accused products in the United States.

83. Acer's inducement is ongoing.

84. Acer has also indirectly infringed by contributing to the infringement of the '959 Patent, the '573 Patent, and the '612 Patent. Acer has contributed to the direct infringement of the '959 Patent, the '573 Patent, and the '612 Patent by the end-user of the accused products.

85. The accused products have special features that are specially designed to be used in an infringing way and that have no substantial uses other than ones that infringe the '959 Patent, the '573 Patent, and the '612 Patent, including, for example, claim 1 of the '959 Patent, claim 13 of the '573 Patent, and claim 1 of the '612 Patent.

86. The special features include, for example, executing computer instructions in an instruction cache used in a manner that infringes the '959 Patent; power distribution and power management techniques used in a manner that infringes the '573 Patent; and retrieving automatic software updates in an embedded system used in a manner that infringes the '612 Patent.

87. These special features constitute a material part of the invention of one or more of the claims of the '959 Patent, the '573 Patent, and the '612 Patent and are not staple articles of commerce suitable for substantial non-infringing use.

88. Acer's contributory infringement is ongoing.

89. Acer has had actual knowledge of the '959 Patent, the '573 Patent, and the '612 Patent at least as of the date when it was notified of the filing of this action. Since at least that time, Acer has known the scope of the claims of the '959 Patent, the '573 Patent, and the '612 Patent; the products that practice the '959 Patent, the '573 Patent, and the '612 Patent; and that Liberty Patents is the owner of the '959 Patent, the '573 Patent, and the '612 Patent.

90. By the time of trial, Acer will have known and intended (since receiving such notice) that its continued actions would infringe and actively induce and contribute to the infringement of one or more claims of the '959 Patent, the '573 Patent, and the '612 Patent.

91. Furthermore, Acer has a policy or practice of not reviewing the patents of others (including instructing its employees to not review the patents of others), and thus has been willfully blind of Liberty Patents' patent rights. *See, e.g.*, M. Lemley, "Ignoring Patents," 2008 Mich. St. L. Rev. 19 (2008).

92. Acer's actions are at least objectively reckless as to the risk of infringing valid patents, and this objective risk was either known or should have been known by Acer. Acer has knowledge of the '959 Patent, the '573 Patent, and the '612 Patent.

93. Acer's customers have infringed the '959 Patent, the '573 Patent, and the '612 Patent. Acer has encouraged its customers' infringement.

94. Acer's direct and indirect infringement of the '959 Patent, the '573 Patent, and the '612 Patent has been, and/or continues to be willful, intentional, deliberate, and/or in conscious disregard of Liberty Patents' rights under the patents-in-suit.

95. Liberty Patents has been damaged as a result of Acer's infringing conduct alleged above. Thus, Acer is liable to Liberty Patents in an amount that adequately compensates it for such infringements, which, by law, cannot be less than a reasonable royalty, together with interest and costs as fixed by this Court under 35 U.S.C. § 284.

JURY DEMAND

Liberty Patents hereby requests a trial by jury on all issues so triable by right.

PRAYER FOR RELIEF

Liberty Patents requests that the Court find in its favor and against Acer, and that the

Court grant Liberty Patents the following relief:

a. Judgment that one or more claims of the '959 Patent, the '573 Patent, and the '612 Patent have been infringed, either literally and/or under the doctrine of equivalents, by Acer and/or all others acting in concert therewith;

b. A permanent injunction enjoining Acer and its officers, directors, agents, servants, affiliates, employees, divisions, branches, subsidiaries, parents, and all others acting in concert therewith from infringement of the '959 Patent, the '573 Patent, and the '612 Patent; or, in the alternative, an award of a reasonable ongoing royalty for future infringement of the '959 Patent, the '573 Patent, and the '612 Patent by such entities;

c. Judgment that Acer account for and pay to Liberty Patents all damages to and costs incurred by Liberty Patents because of Acer's infringing activities and other conduct complained of herein, including an award of all increased damages to which Liberty Patents is entitled under 35 U.S.C. § 284;

d. That Liberty Patents be granted pre-judgment and post-judgment interest on the damages caused by Acer's infringing activities and other conduct complained of herein;

e. That this Court declare this an exceptional case and award Liberty Patents its reasonable attorney's fees and costs in accordance with 35 U.S.C. § 285; and

f. That Liberty Patents be granted such other and further relief as the Court may deem just and proper under the circumstances.

Dated: September 2, 2020

Respectfully submitted,

/s/ Zachariah S. Harrington

Matthew J. Antonelli
Texas Bar No. 24068432
matt@ahtlawfirm.com

Zachariah S. Harrington
Texas Bar No. 24057886
zac@ahtlawfirm.com
Larry D. Thompson, Jr.
Texas Bar No. 24051428
larry@ahtlawfirm.com
Christopher Ryan Pinckney
Texas Bar No. 24067819
ryan@ahtlawfirm.com
Rehan M. Safiullah
Texas Bar No. 24066017
rehan@ahtlawfirm.com

ANTONELLI, HARRINGTON
& THOMPSON LLP
4306 Yoakum Blvd., Ste. 450
Houston, TX 77006
(713) 581-3000

Stafford Davis
State Bar No. 24054605
sdavis@stafforddavisfirm.com
Catherine Bartles
Texas Bar No. 24104849
cbartles@stafforddavisfirm.com
THE STAFFORD DAVIS FIRM
815 South Broadway Avenue
Tyler, Texas 75701
(903) 593-7000
(903) 705-7369 fax

Attorneys for Liberty Patents, LLC