Faculté des Sciences et Ingénierie
Master Informatique
Systèmes Électroniques, Systèmes Informatiques

Laboratoire d'Informatique Paris 6 - CIAN

Hardware initialization of modern computers

A review on the importance of firmware in modern computing and a documentation on the Asus KGPE-D16 RAM initialization

August, 2024

Adrien 'neox' Bourmault (neox@gnu.org)

Under the supervision of Franck WAJSBÜRT (franck.wajsburt@lip6.fr)

This is Edition 0.0.

Contents

Abstract

The global trend is towards the scarcity of free software-compatible hardware, and soon there will be no computer that will work without software domination by big companies, especially involving BIOSes. A Basic Input Output System (BIOS) was originally a set of low-level functions contained in the read-only memory of a computer's mainboard, enabling it to perform basic operations when powered up. However, the definition of a BIOS has evolved to include what used to be known as Power On Self Test (POST) for the presence of peripherals, allocating resources for them to avoid conflicts, and then handing over to an operating system boot loader. Nowadays, the bulk of the BIOS work is the initialization and training of RAM. This means, for example, initializing the memory controller and optimizing timing and read/write voltage for optimal performance, making the code complex, as its role is to optimize several parallel buses operating at high speeds and shared by many CPU cores, and make them act as a homogeneous whole.

This documentation is the product of a project hosted by the *LIP6 laboratory* and supported by the *GNU Boot Project* and the *Free Software Foundation*, delves into the importance of firmware in the hardware initialization of modern computers. It explores various aspects of firmware, such as Intel Management Engine (ME), AMD Platform Security Processor (PSP), Advanced Configuration and Power Interface (ACPI), and System Management Mode (SMM). Additionally, it provides an in-depth look at memory initialization and training algorithms, highlighting their critical role in system stability and performance.

Examples of the implementation in the Asus KGPE-D16 mainboard are presented, describing its hardware characteristics, topology, and the crucial role of firmware in its operation after the mainboard architecture is examined. Practical examples illustrate the impact of firmware on hardware initialization, memory optimization, resource allocation, power management, and security. Specific algorithms used for memory training and their outcomes are analyzed to demonstrate the complexity and importance of firmware in achieving optimal system performance.

Furthermore, the article explores the relationship between firmware and hardware virtualization, discussing how modern firmware supports and enhances virtualized environments. Security considerations and future trends in firmware development are also addressed, emphasizing the need for continued research and advocacy for free software-compatible hardware. The article concludes with a call to action, urging the development of libre firmware solutions to ensure greater control and security in computing.

Chapter 1

Introduction to firmware and BIOS evolution

1.1 Historical context of BIOS

1.1.1 Definition and origin

The BIOS (Basic Input/Output System) is firmware used to perform hardware initialization during the booting process and to provide runtime services for operating systems and programs. Being a critical component for the startup of personal computers, acting as an intermediary between the computer's hardware and its operating system, the BIOS is embedded on a chip on the motherboard and is the first code that runs when a PC is powered on. The concept of BIOS has its roots in the early days of personal computing. It was first developed by IBM for their IBM PC, which was introduced in 1981. The term BIOS itself was coined by Gary Kildall, who developed the CP/M (Control Program for Microcomputers) operating system. In CP/M, BIOS was used to describe a component that interfaced directly with the hardware, allowing the operating system to be somewhat hardware-independent.

IBM's implementation of BIOS became a de facto standard in the industry, as it was part of the IBM PC's open architecture, which refers to the design philosophy adopted by IBM when developing the IBM Personal Computer (PC), introduced in 1981. This architecture is characterized by the use of off-the-shelf components and publicly available specifications, which allowed other manufacturers to create compatible hardware and software. It was in fact a departure from the proprietary systems prevalent at the time, where companies closely guarded their designs to maintain control over the hardware and software ecosystem. For example, IBM used the Intel 8088 CPU, a well-documented and widely available processor, and also the Industry Standard Architecture (ISA) bus, which defined how various components like memory, storage, and peripherals communicated with the CPU. This open architecture allowed other manufacturers to create IBM-compatible computers, also known as "clones", which further popularized the BIOS concept. As a result, the IBM PC BIOS set the stage for a standardized method of interacting with computer hardware, which has evolved over the years but remains fundamentally the same in principle. IBM also published detailed technical documentation at that time, including circuit diagrams, BIOS listings, and interface specifications. This transparency allowed other companies to understand and replicate the IBM PC's functionality.

1.1.2 Functionalities and limitations

The Basic Input/Output System (BIOS) is a foundational firmware component in early personal computers, responsible for initializing hardware and booting the operating system. Developed as part of IBM's original PC design, the BIOS provided essential functionalities.

When a computer is powered on, the BIOS executes a Power-On Self-Test (POST), a diagnostic sequence that verifies the integrity and functionality of critical hardware components such as the CPU, RAM, disk drives, keyboard, and other peripherals. This process ensures that all essential hardware components are operational before the system attempts to load the operating system. If any issues are detected, the BIOS generates error messages or beep codes to alert the user. Following the successful completion of POST, the BIOS runs the bootstrap loader, a small program that identifies the operating system's bootloader on a storage device, such as a hard drive, floppy disk, or optical drive. The bootstrap loader then transfers control to the OS bootloader, initiating the process of loading the operating system into the computer's memory and starting it. This step effectively bridges the gap between hardware initialization and operating system execution. The BIOS also provides a set of low-level software routines known as interrupts. These routines enable software to perform basic input/output operations, such as reading from the keyboard, writing to the display, and accessing disk drives, without needing to manage the hardware directly. By providing standardized interfaces for hardware components, the BIOS simplifies software development and improves compatibility across different hardware configurations.

Despite its essential role, the early BIOS had several limitations. One significant limitation was its limited storage capacity. Early BIOS firmware was stored in Read-Only Memory (ROM) chips with very limited storage, often just a few kilobytes. This constrained the complexity and functionality of the BIOS, limiting it to only the most essential tasks needed to start the system and provide basic hardware control. The original BIOS was also non-extensible. ROM chips were typically soldered onto the motherboard, making updates difficult and costly. Bug fixes, updates for new hardware support, or enhancements required replacing the ROM chip, leading to challenges in maintaining and upgrading systems. Furthermore, the early BIOS was tailored for the specific hardware configurations of the initial IBM PC models, which included a limited set of peripherals and expansion options. As new hardware components and peripherals were developed, the BIOS often needed to be updated to support them, which was not always feasible or timely. Performance bottlenecks were another limitation. The BIOS provided basic input/output operations that were often slower than direct hardware access methods. For example, disk I/O operations through BIOS interrupts were slower compared to later direct access methods provided by operating systems, resulting in performance bottlenecks, especially for disk-intensive operations. This inflexibility restricts the ability to support new hardware and technologies efficiently. Early BIOS implementations also had minimal security features. There were no mechanisms to verify the integrity of the BIOS code or to protect against unauthorized modifications, leaving systems vulnerable to attacks that could alter the BIOS and potentially compromise the entire system, such as rootkits and firmware viruses.
Added to that, the traditional BIOS operates in 16-bit real mode, a constraint that limits the amount of code and memory it can address. This limitation hinders the performance and complexity of firmware, making it less suitable for modern computing needs [11]. Additionally, BIOS relies on the Master Boot Record (MBR) partitioning scheme, which supports a maximum disk size of 2 terabytes and allows only four primary partitions [13][33]. This constraint has become a significant drawback as storage capacities have increased. Furthermore, the traditional BIOS has limited flexibility and is challenging to update or extend. This inflexibility restricts the ability to support new hardware and technologies efficiently [35][1].

1.2 Modern BIOS and UEFI

1.2.1 Transition from traditional BIOS to UEFI (Unified Extensible Firmware Interface)

All the limitations listed earlier have necessitated a transition to a more modern firmware interface, designed to address the shortcomings of the traditional BIOS. This section delves into the historical context of this shift, the driving factors behind it, and the advantages UEFI offers over the traditional BIOS.
The development of UEFI began in the mid-1990s as part of the Intel Boot Initiative, which aimed to modernize the boot process and overcome the limitations of the traditional BIOS. By 2005, the Unified EFI Forum, a consortium of technology companies including Intel, AMD, and Microsoft, had formalized the UEFI specification [13]. UEFI was designed to address the shortcomings of the traditional BIOS, providing several key improvements.

One of the most significant advancements of UEFI is its support for 32-bit and 64-bit modes, allowing it to address more memory and run more complex firmware programs. This capability enables UEFI to handle the increased demands of modern hardware and software [11][34]. Additionally, UEFI uses the GUID Partition Table (GPT) instead of the MBR, supporting disks larger than 2 terabytes and allowing for a nearly unlimited number of partitions [12][33]. Improved boot performance is another driving factor. UEFI provides faster boot times compared to the traditional BIOS, thanks to its efficient hardware and software initialization processes. This improvement is particularly beneficial for systems with complex hardware configurations, where quick boot times are essential [11]. UEFI's modular architecture makes it more extensible and easier to update compared to the traditional BIOS. This design allows for the addition of drivers, applications, and other components without requiring a complete firmware overhaul, providing greater flexibility and adaptability to new technologies [35][1]. UEFI also includes enhanced security features such as *Secure Boot*, which ensures that only trusted software can be executed during the boot process, thereby protecting the system from unauthorized modifications and malware [7][10].

The industry-wide support and standardization of UEFI have accelerated its adoption across various platforms and devices. Major industry players, including Intel, AMD, and Microsoft, have adopted UEFI as the new standard for firmware interfaces, ensuring broad compatibility and interoperability [13].

1.2.2 An other way with coreboot

While UEFI has become the dominant firmware interface for modern computing systems, it is not without its critics. Some of the primary concerns about UEFI include its complexity, potential security vulnerabilities, and the degree of control it provides to hardware manufacturers over the boot process. As an alternative to UEFI, coreboot offers a different approach to firmware that aims to address some of these concerns and continue the evolution of BIOS. *coreboot*, originally known as LinuxBIOS, is a free firmware project initiated in 1999 by Ron Minnich and his team at the Los Alamos National Laboratory. The project's primary goal was to create a fast, lightweight, and flexible firmware solution that could initialize hardware and boot operating systems quickly, while remaining transparent and auditable[28].

One of the main advantages of *coreboot* over UEFI is its simplicity. *coreboot* is designed to perform only the minimal tasks required to initialize hardware and pass control to a payload, such as a bootloader or operating system kernel. This minimalist approach reduces the attack surface and potential for security vulnerabilities, as there is less code that could be exploited by malicious actors [32]. Another significant benefit of *coreboot* is its libre nature. Unlike UEFI, which is controlled by a consortium of hardware and software vendors, *coreboot*'s source code is freely available and can be audited, modified, and improved by anyone. This transparency ensures that security researchers and developers can review the code for potential vulnerabilities and contribute to its improvement, fostering a community-driven approach to firmware development[28]. *coreboot* also supports a wide range of payloads, allowing users to customize their boot process to suit their specific needs. Popular payloads include SeaBIOS, which provides legacy BIOS compatibility, and Tianocore, which offers UEFI functionality within the *coreboot* framework. This flexibility allows *coreboot* to be used in a variety of environments, from embedded systems to high-performance servers[27].

Despite its advantages, *coreboot* is not without its challenges. The project relies heavily on community contributions, and support for new hardware often lags behind that of UEFI. Additionally, the minimalist design of *coreboot* means that some advanced features provided by UEFI, such as Secure Boot, are not available by default. However, the *coreboot* community continues to work on adding new features and improving compatibility with modern hardware[23]. However, it's important to note that *coreboot* is not entirely free in all aspects. Many modern processors and chipsets require proprietary binary blobs for certain functionalities, such as memory initialization and hardware management. These blobs are necessary for *coreboot* to function correctly on a wide range of hardware, but they compromise the goal of having a fully free firmware one day[20]. To address these concerns, the GNU Project has developed GNU Boot, a fully free distribution of firmware, including *coreboot*, that aims to be entirely free by avoiding the use of proprietary binary blobs. GNU Boot is committed to using only free software for all aspects of firmware, making it a preferred choice for users and organizations that prioritize software freedom and transparency[21].

1.3 Shift in firmware responsibilities

Initially, we saw that the BIOS's primary function was to perform the Power-On Self-Test (POST), a basic diagnostic testing process to check the system's hardware components and ensure they were functioning correctly. This included verifying the CPU, memory, and essential peripherals before passing control to the operating system's bootloader. This process was relatively simple, given the limited capabilities and straightforward architecture of early computer systems[35]. As computer systems advanced, particularly with the advent of more sophisticated memory technologies, the role of the BIOS expanded significantly. An example is that modern memory modules operate at much higher speeds and capacities than their predecessors, requiring precise configuration to ensure stability and optimal performance. We'll see in following sections how memory is taken care by firmware, since the memory controller, a critical component in modern computer systems, manages the data flow between the processor and memory modules. Firmware then plays a crucial role in configuring this controller during the boot process. This configuration includes setting memory frequencies, voltage levels, and timing parameters to match the specifications of the installed memory[13]. The enhanced role of firmware in memory training and optimization directly impacts system performance and stability. For example, overclocking involves configuring the system to run at higher speeds than manufacturer-specified limits. Firmware plays a key role in enabling and managing overclocking, particularly for the memory subsystem. By allowing adjustments to memory frequencies, voltages, and timings, it provides tools for performance tuning while including safeguards to manage the risks of instability and hardware damage [7].

Chapter 2

Characteristics of Asus KGPE-D16 Mainboard

2.1 Overview of Asus KGPE-D16 Hardware

- Description of the mainboard's hardware components
  - CPU: Support for AMD Opteron 6000 series processors
  - RAM: 16 DDR3 DIMM slots supporting up to 256GB of memory
  - Expansion Slots: Multiple PCIe slots for expandability
  - Storage: SATA ports and potential for RAID configurations
  - Networking: Integrated dual gigabit Ethernet ports
  - Other Peripherals: USB ports, audio outputs, and additional I/O ports
- Topology and Layout
  - Physical layout of the mainboard
  - Placement of key components and their interactions
  - Cooling and power distribution

2.2 Firmware's Role in Asus KGPE-D16

- Initial hardware setup
- Memory training and optimization
- Resource allocation and conflict resolution
- Power management and efficiency
- Security features and updates

Chapter 3

Key Components in Modern Firmware

3.1 Advanced Configuration and Power Interface (ACPI)

- Detailed explanation of ACPI
- Role in power management and system configuration
- Implementation in modern operating systems
- **Asus KGPE-D16 Example**: ACPI utilization in power management and device configuration on the mainboard

3.2 System Management Mode (SMM)

- Definition and significance
- How SMM enhances system security
- Examples of SMM applications in real-world systems
- **Asus KGPE-D16 Example**: SMM features and their impact on system security and functionality in the KGPE-D16

3.3 AMD Platform Security Processor (PSP) and Intel Management Engine (ME)

- Overview and purpose
- Security implications, concerns and controversies
- Interaction with system firmware
- Differences between Intel ME and AMD PSP

Chapter 4

Memory Initialization and Training Algorithms

4.1 Importance of Memory Initialization

- Steps involved in initializing the memory controller
- Critical role in system stability and performance
- **Asus KGPE-D16 Example**: Memory initialization process on the KGPE-D16 mainboard

Memory training involves several steps: 1. **Detection and Initialization**: The BIOS detects the installed memory modules, determining their size, speed, and type. 2. **Configuration and Timing Setup**: The BIOS configures the memory controller settings, including timings for memory access such as CAS latency, RAS to CAS delay, and other parameters [11]. 3. **Training and Calibration**: The BIOS performs tests and adjustments to calibrate the memory system, ensuring stable operation at optimal speeds by adjusting signal voltages and testing data integrity [56].

These steps are crucial for modern systems, where improper memory configuration can lead to instability, data corruption, or suboptimal performance.

Memory timings, such as CAS latency, RAS to CAS delay, and others, must be finely tuned to ensure optimal performance. The BIOS uses a combination of predefined profiles and dynamic adjustments to achieve the best balance between speed and stability. Advanced timing optimization involves setting these parameters to ensure that memory operations are performed with minimal latency and maximum throughput [33].

4.2 Memory Training Algorithms

- Techniques used for training memory
- Optimization of timings and voltage settings
- Challenges in multi-core CPU environments
- **Asus KGPE-D16 Example**: Specific algorithms used for memory training in the mainboard and their performance outcomes

To optimize memory performance, the BIOS employs various training algorithms and calibration techniques. These methods test the memory under different conditions and make necessary adjustments to improve stability and efficiency. Key techniques include voltage adjustments, data integrity testing, and signal timing calibration [34].

Voltage adjustments involve tweaking the power supplied to the memory modules to ensure reliable operation. Data integrity testing checks that data can be accurately read and written, while signal timing calibration fine-tunes the delays between different memory operations to minimize latency.

4.3 Practical Examples

- Real-world scenarios where firmware played a crucial role in system performance
- Analysis of firmware updates and their impact on the KGPE-D16 mainboard
- User experiences and testimonials highlighting the importance of firmware
- **Asus KGPE-D16 Example**: Specific case studies and firmware updates for the mainboard

Chapter 5

Firmware and Hardware Virtualization

5.1 Introduction to Hardware Virtualization

- Definition and purpose of virtualization
- How firmware interacts with virtualized environments
- **Asus KGPE-D16 Example**: Virtualization capabilities and performance on the mainboard

5.2 Role of BIOS/UEFI in Virtualization

- Initialization and configuration for virtual machines
- Resource allocation and management
- **Asus KGPE-D16 Example**: BIOS/UEFI settings and their impact on virtualization efficiency on the KGPE-D16

5.3 Security and freedom considerations

- Security risks associated with virtualization
- Measures taken by firmware to mitigate risks
- **Asus KGPE-D16 Example**: Security measures implemented in the mainboard's firmware to support secure virtualization

5.4 Future Trends in Firmware and Virtualization

- Emerging advancements and their impact on firmware
- Predictions for the evolution of BIOS/UEFI in virtualization
- **Asus KGPE-D16 Example**: Potential future firmware updates and their expected impact on the mainboard's virtualization capabilities

Conclusion

5.5 Summary of Key Points

- Recap of the evolution and current state of firmware
- Importance of understanding modern BIOS functionalities
- **Asus KGPE-D16 Example**: Summary of the mainboard's features and firmware contributions

5.6 Call for Action

- Advocacy for free software-compatible hardware
- Encouraging research and development in libre firmware solutions

Bibliography

[1] ACMCS. “The Evolution of Firmware: BIOS to UEFI”. In: *ACM Computing Surveys* 47.4 (2015), pp. 55–61. DOI: 10.1145/2766462.

[2] ACPI. *ACPI Specification*. https://www.acpi.info/spec.htm. Accessed: 2024-07-05.

[3] AMD. *AMD Platform Security Processor (PSP)*. https://www.amd.com/en/technologies/security. Accessed: 2024-07-05.

[4] AMD. “BIOS and Kernel Developers Guide (BKDG) for AMD Family 15h Models 00h-0Fh Processors Rev 3.14”. In: 42301. Jan. 2013.

[5] AMD. “SR5690/5670/5650 BIOS Developers Guide 3.00”. In: 43870. Nov. 2010.

[6] AMD. “SR5690/5670/5650 Register Programming Requirements 3.05”. In: 43872. Aug. 2012.

[7] T. Anderson. *BIOS vs. UEFI: Understanding the Modern Boot Environment*. https://www.pcworld.com/article/3171322/bios-vs-uefi-understanding-the-modern-boot-environment.html. 2018.

[8] IBM Archives. *IBM Personal Computer*. https://www.ibm.com/history/personal-computer. 2024.

[9] Asus. *Asus KGPE-D16 Mainboard Documentation and User Manuals*. https://www.asus.com/Commercial-Servers-Workstations/KGPE-D16/HelpDesk_Manual/. Accessed: 2024-07-05.

[10] H. Chang and A. Smith. “UEFI Secure Boot in Modern Computing”. In: *International Journal of Information Security* 12.3 (2013), pp. 231–241. DOI: 10.1007/s10207-013-0191-1.

[11] Intel Corporation. *Unified Extensible Firmware Interface (UEFI)*. https://www.intel.com/content/www/us/en/architecture-and-technology/unified-extensible-firmware-interface.html. 2020.

[12] Microsoft Corporation. *UEFI Firmware*. https://docs.microsoft.com/en-us/windows-hardware/drivers/bringup/uefi-firmware. 2019.

[13] UEFI Forum. *UEFI Specification*. https://uefi.org/specifications. 2021.

[14] UEFI Forum. *Unified Extensible Firmware Interface*. https://uefi.org/. 2024.

[15] Jimmy Grewal. *The Creation of the IBM PC*. Armonk Institute. 2024.

[16] Micron Technology Inc. *Technical Note: DDR3 ZQ Calibration*. 2008.

[17] Intel Corporation. *Intel Management Engine (Intel ME)*. https://www.intel.com/content/www/us/en/architecture-and-technology/intel-management-engine.html. Accessed: 2024-07-05.

[18] Christoph Lameter. “NUMA (Non-Uniform Memory Access): An Overview”. In: *Queue* 11 (July 2013). DOI: 10.1145/2508834.2513149.

[19] Samsung Electronics Co. Ltd. *DDR3 SDRAM Specification Rev 1.4*. Nov. 2011.

[20] GNU Boot project maintainers. *Frequently Asked Questions*. https://www.gnu.org/software/gnuboot/web/faq.html. Accessed: 2024-07-23.

[21] GNU Boot project maintainers. *GNU Boot — Free your BIOS today!* [Online; accessed 7-May-2024]. 2024. URL: https://www.gnu.org/software/gnuboot/.

[22] GNU Boot project maintainers. *GNU Boot — Status*. [Online; accessed 7-May-2024]. 2024. URL: https://www.gnu.org/software/gnuboot/web/status.html.

[23] R. Minnich and E. Hendricks. “Challenges and Progress in Coreboot Development”. In: *Journal of Open Source Software* 3.29 (2018), pp. 1–6. DOI: 10.21105/joss.00429.

[24] Computer History Museum. *The Evolution of the BIOS*. https://computerhistory.org/. 2024.

[25] Author Names. “Title of the Paper on Hardware Virtualization and Firmware”. In: *Journal Name* Volume.Number (Year), Pages.

[26] Author Names. “Title of the Paper on Memory Training Algorithms”. In: *Journal Name* Volume.Number (Year), Pages.

[27] coreboot project. *coreboot Payloads*. https://www.coreboot.org/Payloads. Accessed: 2024-07-23.

[28] coreboot project. *coreboot: Open Source Firmware*. https://www.coreboot.org/. Accessed: 2024-07-23.

[29] Raptor Engineering LLC. *Raptor Engineering website*. [Online; accessed 8-May-2024]. 2009-2024. URL: https://raptorengineering.com/.

[30] Felix Richter et al. “BIOS and UEFI firmware analysis”. In: *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*. 2011, pp. 7–16.

[31] Ronald H Rosenberg. *Open architecture computer systems*. IEEE Computer Society Press, 1994.

[32] M. Rudolph. “LinuxBIOS: Open Source Boot Firmware”. In: *Proceedings of the Linux Symposium*. 2007, pp. 159–167. URL: https://ols.fedoraproject.org/OLS/Reprints-2007/rudolph-Reprint.pdf.

[33] M. E. Russinovich, D. A. Solomon, and A. Ionescu. *Windows Internals, Part 1*. 6th. Microsoft Press, 2012.

[34] M. Shin and K. Lee. “Design and Implementation of a UEFI-Compliant Firmware Platform”. In: *Journal of Computer Science and Technology* 26.2 (2011), pp. 219–230. DOI: 10.1007/s11390-011-0121-8.

[35] R. Smith. *UEFI vs. BIOS: Whats the Difference?* https://www.techradar.com/news/uefi-vs-bios-whats-the-difference. 2017.

[36] Sorbonne Université/CNRS. *Annuaire LIP6*. [Online; accessed 7-May-2024]. 2024. URL: https://www.lip6.fr/recherche/resultat.php?keyword=&find=Rechercher+au+LIP6.

[37] Sorbonne Université/CNRS. *Laboratoire d’Informatique de Paris 6*. [Online; accessed 7-May-2024]. 2024. URL: https://www.lip6.fr/.

[38] Wikipedia contributors. *AGESA — Wikipedia, The Free Encyclopedia*. [Online; accessed 8-May-2024]. 2023. URL: https://en.wikipedia.org/w/index.php?title=AGESA&oldid=1166805057.

[39] Wikipedia contributors. *AMD Platform Security Processor — Wikipedia, The Free Encyclopedia*. [Online; accessed 7-May-2024]. 2024. URL: https://en.wikipedia.org/w/index.php?title=AMD_Platform_Security_Processor&oldid=1216563013.

[40] Wikipedia contributors. *DDR3 SDRAM — Wikipedia, The Free Encyclopedia*. [Online; accessed 8-May-2024]. 2024. URL: https://en.wikipedia.org/w/index.php?title=DDR3_SDRAM&oldid=1207641521.

[41] Wikipedia contributors. *Free software — Wikipedia, The Free Encyclopedia*. [Online; accessed 30-January-2024]. 2024. URL: https://en.wikipedia.org/w/index.php?title=Free_software&oldid=1196006316.

[42] Wikipedia contributors. *Free Software Foundation — Wikipedia, The Free Encyclopedia*. [Online; accessed 7-May-2024]. 2024. URL: https://en.wikipedia.org/w/index.php?title=Free_Software_Foundation&oldid=1222269091.

[43] Wikipedia contributors. *Free software movement — Wikipedia, The Free Encyclopedia*. [Online; accessed 29-January-2024]. 2024. URL: https://en.wikipedia.org/w/index.php?title=Free_software_movement&oldid=1197710495.

[44] Wikipedia contributors. *GNU Free Documentation License — Wikipedia, The Free Encyclopedia*. [Online; accessed 30-January-2024]. 2024. URL: https://en.wikipedia.org/w/index.php?title=GNU_Free_Documentation_License&oldid=1193649968.

[45] Wikipedia contributors. *GNU General Public License — Wikipedia, The Free Encyclopedia*. [Online; accessed 30-January-2024]. 2024. URL: https://en.wikipedia.org/w/index.php?title=GNU_General_Public_License&oldid=1199241605.

[46] Wikipedia contributors. *GNU GRUB — Wikipedia, The Free Encyclopedia*. [Online; accessed 7-May-2024]. 2024. URL: https://en.wikipedia.org/w/index.php?title=GNU_GRUB&oldid=1217643156.

[47] Wikipedia contributors. *GNU Project — Wikipedia, The Free Encyclopedia*. [Online; accessed 7-May-2024]. 2024. URL: https://en.wikipedia.org/w/index.php?title=GNU_Project&oldid=1205139455.

[48] Wikipedia contributors. *Intel Management Engine — Wikipedia, The Free Encyclopedia*. [Online; accessed 7-May-2024]. 2024. URL: https://en.wikipedia.org/w/index.php?title=Intel_Management_Engine&oldid=1216703991.

[49] Wikipedia contributors. *Laboratoire d'Informatique de Paris 6 — Wikipedia, The Free Encyclopedia*. [Online; accessed 7-May-2024]. 2024. URL: https://en.wikipedia.org/w/index.php?title=Laboratoire_d%27Informatique_de_Paris_6&oldid=1222525180.

[50] Wikipedia contributors. *Non-disclosure agreement — Wikipedia, The Free Encyclopedia*. [Online; accessed 8-May-2024]. 2023. URL: https://en.wikipedia.org/w/index.php?title=Non-disclosure_agreement&oldid=1183749255.

[51] Wikipedia contributors. *OpenBMC — Wikipedia, The Free Encyclopedia*. [Online; accessed 8-May-2024]. 2023. URL: https://en.wikipedia.org/w/index.php?title=OpenBMC&oldid=1183698628.

[52] Wikipedia contributors. *SeaBIOS — Wikipedia, The Free Encyclopedia*. [Online; accessed 7-May-2024]. 2023. URL: https://en.wikipedia.org/w/index.php?title=SeaBIOS&oldid=1179465237.

[53] Wikipedia contributors. *The Free Software Definition — Wikipedia, The Free Encyclopedia*. [Online; accessed 29-January-2024]. 2023. URL: https://en.wikipedia.org/w/index.php?title=The_Free_Software_Definition&oldid=1192713194.

[54] Wikipedia contributors. *The Open Source Definition — Wikipedia, The Free Encyclopedia*. [Online; accessed 30-January-2024]. 2023. URL: https://en.wikipedia.org/w/index.php?title=The_Open_Source_Definition&oldid=1191447775.

[55] Wikipedia contributors. *X86 — Wikipedia, The Free Encyclopedia*. [Online; accessed 7-May-2024]. 2024. URL: https://en.wikipedia.org/w/index.php?title=X86&oldid=1221800539.

[56] K. Wolf. "Modern Boot Firmware: Moving from BIOS to UEFI". In: *IEEE Computer Society* 39.5 (2006), pp. 42–47. DOI: 10.1109/MC.2006.156.

GNU Free Documentation License

Version 1.3, 3 November 2008

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "**Document**", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "**you**". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "**Modified Version**" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "**Secondary Section**" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "**Invariant Sections**" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "**Cover Texts**" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "**Transparent**" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "**Opaque**".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "**Title Page**" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "**publisher**" means any person or entity that distributes copies of the Document to the public.

A section "**Entitled XYZ**" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "**Acknowledgements**", "**Dedications**", "**Endorsements**", or "**History**".) To "**Preserve the Title**" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.
You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.
The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.
The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.
In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.
You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.
If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See `https://www.gnu.org/licenses/`.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy’s public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

“Massive Multiauthor Collaboration Site” (or “MMC Site”) means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A “Massive Multiauthor Collaboration” (or “MMC”) contained in the site means any set of copyrightable works thus published on the MMC site.

“CC-BY-SA” means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

“Incorporate” means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is “eligible for relicensing” if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with ... Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.