**Flash: “Programmable” and “Read-Only” and Other Contradictions**

**CS-350**

Damean Murphy-Short

Southern New Hampshire University

January 13, 2025

There are many interesting and significant differences between embedded systems and desktop personal computers. This paper is tasked with describing the most important of those differences. Many examples will be given by referencing the ATMega32 microprocessor documentation (ATMel, Microchip.com, n.d.), which is featured in the Arduino Pro Micro microcontroller, which the author is familiar with to a novice degree.

The first difference we seek to explore is how embedded systems and personal computers utilize non-volatile memory. An embedded system will typically have an EEPROM (electronically erasable, programmable read-only memory) or some other form of ROM on the chip. This memory is intended, for the most part, to be written to once (or very few times) and contains the bootloader and main program instructions that will be read directly by the microprocessor.

A personal computer typically has a mechanical hard drive disk, solid state drive, or comparable storage device that is intended to be written to and read from continuously. Boot instructions and programs are written to RAM before being executed by the processor, differing from the direct reads that are performed on ROM in embedded systems. Typically, a desktop computer will also have ROM on the motherboard chipset that contains bootstrap instructions to fetch the OS bootloader from disk; this memory is practically the same as that found on embedded microcontrollers.

Our benevolent and oh-so-vague rubric asks of us to also ‘explain the differences between embedded systems and desktop systems’ and to do so in only 1-2 pages! We will begin by discussing the difference in the amount of power (literal, as measured in Watts) each will consume and what contributes to this difference. Most embedded systems require one or several orders of magnitude fewer Watts during operation than their desktop analogues. To begin with, the processors in desktop systems are generally running at higher clock speeds (several GHz rather than several MHz). It is also worth noting that these processors are typically multi-core, have a greater number of units performing operations or controlling memory, are physically much larger, have more cache memory to power, and in typical use-cases will be performing far more tasks while running a full operating system, where a microprocessor in an embedded system is smaller, more restricted, or lacking in all of these aspects. A desktop computer is designed to be general purpose, and so where an embedded system may only interact with a user through a single button or a simple display, a desktop computer will often support a variety of inputs and outputs, and therefore require a way to interface with and power them. These extra components, like graphical processing units, require additional power to provide this functionality. The scale of these systems also contributes to the difference in their power draw. A system with 1 TB of secondary memory will need a more memory complex controller and draw more power than a system with only several kilobytes of ROM.

This difference in power consumption means that embedded systems can be easily included in a variety of appliances without significantly increasing the amount of energy they use, which is important both for adhering to regulation and being responsible with our limited collective resources. The mentioned difference in size also contributes to the ease at which these systems can be incorporated as a part of a whole device. The lack of need to be as general-purpose also reduces the cost of embedded systems by reducing the amount and complexity of components that they are equipped with.

Below we will rapidly bullet point some other key differences between desktop and embedded systems that didn’t find their way into previous paragraphs:

* A desktop computer will often use an x86\_64 architecture processor (though not always, ARM laptops are becoming increasingly more common), whereas microcontrollers most commonly operate on a reduced instruction set architecture.
* A desktop computer has three types of volatile memory: CPU registers, SRAM in the form of CPU caches, and DRAM contained on memory modules that are connected to the CPU via the motherboard. A microcontroller may or may not have any DRAM, instead most RAM is commonly SRAM, as in the ATMega32 (ATMel, Microchip.com, n.d.). The microprocessor in an embedded system also typically has CPU registers. As in the ATMega32, some embedded systems will not have separate address spaces for “cache” and “RAM” due to these differences. The amount of volatile memory is typically far more limited in emebedded systems, meaning that embedded programs have a greater tendency to limit dynamic memory allocation or to not use dynamic memory allocation at all, instead using the “stack” only.

References

Atmel & Microchip.com. (n.d.). *ATmega32*. Microchip. Retrieved January 13, 2025, from https://www.microchip.com/en-us/product/ATmega32#Documentation