



The PHINIX+ System Architecture Documentation

Volume 0: Preface

Come discuss with us at the official
PHINIX+ Discord server:
<https://discord.gg/EFKDF3VE9C>

Version: 0.7.1
Date: 20th of February 2026
by Martin Andronikos



Licensed under CC BY-NC-SA 4.0
[https://creativecommons.org/licenses/
by-nc-sa/4.0/](https://creativecommons.org/licenses/by-nc-sa/4.0/)

Contents

| | | |
|-------|--------------------------|---|
| 1 | About the Documentation | 3 |
| 1.1 | Styling Decisions | 3 |
| 1.2 | Licensing Decisions | 3 |
| 2 | About the Author | 4 |
| 2.1 | On Hardware Engineering | 4 |
| 2.1.1 | Education | 4 |
| 2.1.2 | Attributions | 4 |
| 2.2 | On Typesetting | 4 |
| 3 | About the Project | 5 |
| 3.1 | The Catalyst | 5 |
| 3.1.1 | Expectations of the Past | 5 |
| 3.1.2 | Limitations of the Past | 5 |
| 3.2 | The Goal | 5 |

1 About the Documentation

This volume discusses the nature of the documentation itself, the scope and aim of the PHINIX+ project, and about the author as an individual and their motives. As a result, the use of the first person in the following chapters is unavoidable. The formal specification begins at Volume 1 if such details are irrelevant for the reader.

The purpose of the documentation is to describe all of the features of the PHINIX+ system with maximum possible detail. Therefore, it tries to conform to the typical requirements expected from technical documentation. The most important details to be transparent about regarding the documents themselves are thus the decisions about the look of the documents (styling) and about the licensing around the documents.

1.1 Styling Decisions

This documentation was written using the “[Typst](#)” typesetting program. If the source code of the used template is not available or the reader is not aware of Typst’s syntax, the decisions made regarding styling are hereby given:

- Pages are A4 sized with 25mm of vertical and 20mm of horizontal margins.
- For the bulk of the text, the serif font “[IBM Plex Serif](#)” was used.
- For the headings and for the title, the sans serif font “[IBM Plex Sans](#)” was used.
- For the code blocks the [Nerd Fonts](#) variant of the monospace font “[Inconsolata](#)” was used.
- Internal links (references) are in blue color with the exception of the contents page.
- External links (hyperlinks) are underlined and in blue color (as shown above).

In addition, the hereby given names for the varying-sized collections of text have been adopted:

- **Documentation** refers to the sum of the documents of the work.
- **Volume** refers to an individual, specific document of the work.
- **Chapter** refers to a collection of text under a first-level heading within a volume.
- **Section** refers to a collection of text under a second-level heading within a chapter.
- **Subsection** refers to a collection of text under a third-level heading within a section.

1.2 Licensing Decisions

This documentation is licensed under the Creative Commons [BY-NC-SA 4.0](#) license. This project is not currently intended to generate direct profit for the author and/or any other user of the project, focusing instead on educational and novelty value. If you are making a derivative of PHINIX+, you are kindly requested to retain this license per the requirements of the license and attribute the original author. The license only covers the architecture itself (this documentation) and not any implementations of the described architecture.

2 About the Author

This section talks about me, the author and individual who is behind the majority of the work behind PHINIX+. This section will come off as the most informal due to its nature, but I believe knowing the person behind it all might provide additional context. I will try to lay out my past as it relates to the project and my aspirations for the future.

2.1 On Hardware Engineering

I knew my passion was computer engineering ever since high school. Since then, I have been researching about the inner workings of computers and their operating principles. Due to the immense complexity of modern computational systems, the best introduction to the field has been looking at designs of the past, or, in other words, the field of “retrocomputing”. The constraints in the designs of that era were interesting to me, as well as being a source of influence in and of themselves.

2.1.1 Education

Even though I am in the process of attending a computer engineering course at a polytechnic university, most of my knowledge had been acquired before university. In other words, as I am writing this, I do not currently possess any formal experience or certification on the topic, and instead, I am self-taught. Most of my knowledge about this topic had been acquired from online sources, mostly Wikipedia and YouTube, and from direct assistance and guidance from people I met online, primarily on Discord.

2.1.2 Attributions

I would like to use this opportunity to point out a few of the exceptional people that have directly or indirectly helped me progress in this field:

- **Ben Eater:** The original homebrew computing YouTuber. His series on building a CPU on breadboards was the catalyst that got me interested in the first place almost 10 years ago. He has since moved on to other interesting topics in the field.
- **James Sharman:** His advanced homebrew computer series captured the imagination of many, myself included. His design explores more complex topics in the field, such as pipelining and bus timings, and portrays the immense software development effort that follows a completed build.
- **Open Redstone Engineers:** It might seem silly at first, but a Minecraft community of all places, ORE, has been the single most important gateway to people whose contributions to my journey cannot be understated. I joined this community about the same time as Ben Eater was creating his breadboard computer series.
- **Clamentos:** A member of ORE, he was the person that took time out of his days to explain to a stranger on the internet, me, the operation of and the principles behind pipelining, cache memory and branch prediction better than any professor at my university could.
- **LordDecapo:** A member of ORE, he recognized the potential in me and taught me how to program in SystemVerilog, thus allowing me to escape the virtual realm of Minecraft’s simulated wiring system. Later, as per his suggestion, I bought the very DE0 Nano FPGA development board that will ultimately be used to host PHINIX+.

2.2 On Typesetting

Typesetting is also an activity which I have had to teach myself. Even though my university did provide a lesson on technical document writing, it was not as useful as one would hope. Regardless, I was already trying to learn it myself before I even took that lesson. And, as per my choice of the typesetting program, I chose Typst simply because it was the easiest one for me to get into while providing a wide range of features and an expansive, complete syntax. I did look into LaTeX, but I quickly got overwhelmed and decided to not use it.

Since I am not confident in my writing abilities, I’d like to point out that if you would like to suggest something regarding the documents, don’t hesitate to reach out on our [Discord server](#) or make a pull request on the documentation’s [GitHub repository](#).

3 About the Project

This documentation and the overall design and direction of the PHINIX+ project is my personal endeavor, which I have been working on during my free time. Due to this, development naturally has been an ongoing effort, with a time span measured in years. It is not surprising, considering the scale of the project and that this is, at least for now, a one man show.

3.1 The Catalyst

I have consumed quite a lot of content regarding computer engineering over the years, mostly from various channels on YouTube. This gave me the valuable knowledge that made me curious enough, and made me feel confident enough to try to implement something of my own, inspired by systems from the mid 80s to the early 90s.

3.1.1 Expectations of the Past

I never got to experience early computing or the home computer revolution. To me, that period in the history of computing was the most interesting, as the systems of the time were still quite simple, enough so as to allow the owner to entirely understand how they would operate, yet make it into the hands of ordinary people. This understanding was also perpetuated by the fact that during that time, it was expected that the owner knew how to program their machine. Accumulation and handling of software was also much more intimate, since the internet did not exist. Everything was physical media.

Even though this did arguably begin with the MITS Altair 8800 during the mid 70s it really only reached the masses with machines like the Commodore 64 and the IBM Personal Computer (whose system architecture evolved into our modern computers) in the early 80s. To this day, there exists almost a cult following for these machines due to people's nostalgia and care for the past; the retro community.

3.1.2 Limitations of the Past

Apart from the difference in how people used computers in the past, there was also, of course, a big difference in the amount of performance that was available. With such limited resources, programmers had to be extra careful about what their operations would cost. This limiting environment resulted in numerous clever techniques, which took advantage of some quirk of the hardware to gain in performance, or to achieve things far above the expectation of the designers of the hardware.

As a result, I believe that software engineering under the limitations of systems of the past would better qualify as an art form compared to that of today. Programmers were (more than just) incentivized to come up with novel solutions to a problem, and in the process of this, being in more direct control and understanding of the hardware they were working on. It felt more authentic. It felt like a piece of software was genuinely crafted with care and attention, instead of just as a means to profits—believe it or not, malware included.

3.2 The Goal

So, having covered all of the most important factors which have contribute to me deciding to start this project, I shall finally explain what the purpose of PHINIX+ is. In short, it is my attempt to recreate that experience from the past, while teaching myself the practical principles behind it. The goal is first and foremost to educate myself, and secondarily to create at least a half-descent retro-style system; something to be proud about.

In practice, all that means is that PHINIX+ will be my honest attempt at a RISC-style CPU with a system around it to match, based on designs of the mid 80s to early 90s. I chose that time period and not something earlier, since my desired architecture (and implementation) is more similar to machines from then, instead of earlier so-called proto-CISC systems, like those based on 6502 or Z80 CPUs. However, since I am trying to design in such a way so as to allow others to implement their own systems based on the herein described architecture, I will be leaving implementation details mostly out of this documentation, and rather up to the implementer's discretion.