Component Analysis

Year: Spring 2025 Team: 15 Project: αCassiopeiae 8800

Creation Date: 2/7/2025 Last Modified: February 8, 2025

Member 1: Brian Lu Email: lu964@purdue.edu

Member 2: Seth Deegan Email: sdeegan@purdue.edu

Member 3: Caleb Shinkle Email: cshinkle@purdue.edu

Member 4: Nathan Huang Email: huan1811@purdue.edu

Assignment Evaluation: See Rubric on Brightspace Assignment

1.0 Component Analysis:

There are 3 major components in our design. They are described in detail as seen below:

1. Microcontroller:

We are using a Raspberry Pi 2350 to emulate the functionality of the Intel 8080 processor that was initially in the Altair 8800, but in a com

pact way.

1. Power Converter:

We are using a buck converter with a linear regulator to maintain a 3.3V/0.5A level output from a 5V/2A wall input, hoping for high efficiency.

1. LEDS:

We are using reverse mounted LEDs to light up the front panel, to obtain an aesthetic like the original Altair; if unsuccessful or uncertain about the reverse mounting, we will switch to the original red lamps the Altair had.

1.1 Analysis of Component 1: Microcontroller

Four our choice of microcontroller, we were looking mainly for a microcontroller with enough I/O pins to support our implementation of the S100 bus, as well as a fast enough clock speed for our emulator to execute 8080 instructions at the desired 2 MHz that the original Altair 8800 had. Additionally, we wanted to write our emulator in Rust, so we wanted our microcontroller to support Rust to avoid writing our emulator in C. Finally, we want good UART support so we can interface two machines to communicate over UART. We are going to use a micro for emulation of the Intel 8080 CPU as well as for memory, so we needed to make sure that our micros would have enough memory to house programs and easily support an implementation of the S100-inspired bus for communication.

The first option that came to mind was some kind of STM microcontroller, specifically the STMF722IE. Since we all have some experience working with this family of microcontrollers, we thought it would be a natural choice. After looking at the datasheet, the STMF722IE has various form factors with different numbers of GPIO pins, up to 140. Additionally, its cores can clock up to 216 MHz, it contains 512 KB of SRAM, and 4 UART channels, which is everything that we need for the project [1]. With Rust support, the STMF722IE seemed like a pretty decent choice for our project, the only thing it lacked was the Programmable IO (PIO) provided by the RP2350B.

Another micro we looked at was an ESP32 series microcontroller, as those are another popular option. For this project specifically, we looked at the ESP32-DOWD-V3 micro. It provides extremely similar specs to the STM micro, including a slightly faster clock (240 MHz), 520 KB of SRAM, and 3 UART channels [3]. Even though it has Rust support, it provides only 34 GPIO pins, which we didn’t think would be enough for our bus implementation. The ESP micro also provides on-board Wi-Fi and Bluetooth support, but we will not be needing it for our project.

Finally, we looked at the RP2350B microcontroller, which has 512 KB of SRAM, similar to the other two micros. Its cores also have 150 MHz clocks, and it provides two ARM cores and two RISC-V cores, which can be used to enhance performance if necessary. It also has 48 GPIO pins, which is plenty to support our bus implementation. It also supports Rust and 2 two UART interfaces. The main draw for the RP2350B was its PIO feature, which would make our bus implementation easier as we could use the 12 PIO state machines provided to control the bus.

All three of these micros appeared to have the right specs for our project, so we ended up going with the RP2350B for its PIO for our bus implementation, as well as its RISC-V cores for better performance if needed. Some sample specs we looked at when making our decision can be seen in the table below:

*Table 1: Microcontroller Specs Comparison*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Clock Speed (MHz) | SRAM (KB) | GPIO Pins | Rust Support | UART |
| RP2350B | 150 | 512 | 48 | Yes | 2 |
| STMF722IE | 216 | 512 | Up to 140\* | Yes | 4 |
| ESP32-DOWD-V3 | 240 | 520 | 34 | Yes | 3 |

1.2 Analysis of Component 2: Buck Converter

For powering our system, we were looking at implementing all circuitry with 3.3V DC. Although the original bus used slightly more exotic voltage levels (+/- 8V, +/- 16V), we plan on doing everything with 3.3V DC. Since our system will be wall-powered, we will need a converter from wall power to a barrel jack (120V AC to 5V DC), then some kind of buck converter or linear voltage regulator to step down from 5V to 3.3V.

Our first option that we found was the MCP16331 High Voltage Input Integrated Switch Step-Down Regulator. It has everything that we were looking for, as it takes an input voltage range of 4.4V to 50V, provides an output voltage range of 2V to 24V, and has an output current of up to 1.2A when operating at 5V [5].

We also looked at the LM1117, which has an input range of 2.6-15V and an output range of 1.25-13.8V, depending on how 2 external resistors are configured. It provides slightly less output current, at only 800 mA [6].

The last regulator we looked at was the LT3045, with an input range of 1.8-20V and an output range of 0-15V. Once again, this model provides less current at only 500 mA [7].

After considering that we are likely need a good amount of current draw to power the several microcontrollers we are using, we decided to go with the MCP16331 since it has the highest output current and supports the voltage ranges we will be using.

*Table 2: Voltage Converter/Regulator Comparison*

|  |  |  |  |
| --- | --- | --- | --- |
|  | Input Voltage Range (V) | Output Voltage Range (V) | Output Current (A) |
| MCP16331 | 4.4 - 50 | 2 - 24 | Up to 1.2 |
| LM1117 | 2.6 - 15 | 1.25 - 13.8 | 0.8 |
| LT3045 | 1.8 - 20 | 0 - 15 | 0.5 |

1.3 Analysis of Component 3: LEDS or Reverse Mounted LEDs

The original Altair used the red lamps found in [4]. However, we aim to minimize the area required by the LEDs, and hence would like to use reverse mounted LEDs for displaying the state of the computer, memory, etc. We could still maintain the red lamp aesthetic that the old Altair had, with 36 red LEDs.

The first component we looked at was the SunLED XZM2CRK45WT-9. SunLED had reverse-mount PLCC devices that we perfect for our situation, but they only provided 11 variants. There were several regions of the datasheet that had to be heavily considered, including the power dissipation, forward voltage, and reverse current.

SunLED had two potential candidates for the surface mounted LEDs. This is because they were the primary vendor we considered, because Digikey and Adafruit, two highly reliable vendors, both pointed us in the direction of SunLED. The other component we considered, which was linked to DigiKey’s website, was the XZM2CRKM2DG45WT-9, which had the capability to flash in both green and red.

When comparing the power dissipation, we needed to consider the absolute maximum ratings. Because our Altair would be most likely operating at room temperature (25°C), we needed to account for maximum standards within that range, because if the voltage or current were to exceed their maximum reverse/forward values, they could potentially cause a short circuit or catch fire. We favored a higher value of the reverse current, although that came with a higher power dissipation.

The last component that was analyzed was the original red LED lamp that was on the Altair, the HLMP-D155A. This is a common low current red lamp, which dissipates little power while operating a steady red light. This is likely why it was used on the original Altair. It has a maximum current rating of 300 mA, which is much higher than either of the reverse mounted LEDs. However, because of its convex shape and the fact we are trying to minimize the volume of the Altair, it may not be the best option. Comparisons can be seen below in the table.

*Table 3: Comparisons of LEDs*

|  |  |  |  |
| --- | --- | --- | --- |
| Component | XZM2CRK45WT-9 | XZM2CRKM2DG45WT-9 | HLMP-D155A |
| Forward Voltage (V) | 2.2 | 2.2 | 1.6 |
| Reverse Voltage (V) | 5 | 5 | 5 |
| Forward  Current (mA) | 50 | 150 | 300 |
| Operating  Temperature (°C) | -40 < T < 85 | -40 < T < 85 | -20 < T < 100 |
| Power (mW) | 140 | 140 or 120 | 87 |
| SMD? | Yes | Yes (Chosen) | No |

2.0 Sources Cited:

[1] “STM32F722IE - STMicroelectronics,” STMicroelectronics, 2025. Available: <https://www.st.com/en/microcontrollers-microprocessors/stm32f722ie.html#documentation> (accessed Feb. 07, 2025).

[2] “RP2350 Datasheet,” *Raspberry Pi Ltd*, Oct. 2024. Available: <https://datasheets.raspberrypi.com/rp2350/rp2350-datasheet.pdf> (accessed Feb. 07, 2025.)

[3] Espressif, “ESP32 Series Datasheet Including,” 2024. Available: <https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf> (accessed Feb. 07, 2025).

‌[4] Broadcom, *HLMP-D,K LED Data Sheet*. [Online]. Available: <https://mm.digikey.com/Volume0/opasdata/d220001/medias/docus/4128/HLMP-D%2CK.pdf>. (Accessed: Feb. 7, 2025)

[5] Microchip, ”High Voltage Input Integrated Switch Step-Down Regulator,” *June 2014*. Available: <https://ww1.microchip.com/downloads/en/DeviceDoc/20005308C.pdf> (Accessed Feb. 8, 2025)

[6] “LM1117 800-mA, Low-Dropout Linear Regulator.” *February 2000*. Available: <https://www.ti.com/lit/ds/symlink/lm1117.pdf> (Accessed Feb. 8, 2025)

[7] Analog Devices, ”LT3045 20 V, 500 mA, Ultralow Noise, Ultrahigh PSRR Linear Regulator” Available: <https://www.analog.com/media/en/technical-documentation/data-sheets/lt3045.pdf> (Accessed Feb. 8, 2025)

‌[8] “Part Number: XZMDKCBDDG45S-9,” 2023. Accessed: Feb. 09, 2025. [Online]. Available: <https://www.sunledusa.com/products/spec/XZMDKCBDDG45S-9.pdf> (Accessed Feb. 8, 2025)

[9] “XDSB8625 V4-X Layout: Maggie L. Part Number: XZMDKVG45WT-9.” [Online]. Available: <https://www.sunledusa.com/products/spec/XZMDKVG45WT-9.pdf> (Accessed: Feb. 08, 2025)

‌