Virtual Memory over FRAM on a microcontroller

Brendan Kristiansen

December 12, 2019

While most people may take for granted the level of complexity behind a RAM card or flash storage, there are thousands upon thousands of lines of code required to make each piece of hardware inside your computer behave the way it does. While this project started out as an ambitious attempt to create an optimized storage solution with a relatively polished interface, hardware complications quickly changed the course of my work into an exploration into the small nuances behind hardware and inside firmware that make the software we write work correctly.

While I was ultimately able to produce a working device that proved I had created a robust system for handling a virtual address space, it required a small number of deviations from what would be considered the "correct" way to make a device similar to this. The inability to use the microcontroller's built-in clock to run the SPI bus is perhaps the biggest issue with the system's current configuration. While the current solution (manually toggling and reading every one of the five total lines used in the SPI bus) did work for the purposes of this project, it is a dangerous way to implement SPI. It is much slower than using the system's built-in clock, and more importantly, in a more complicated hardware configuration, an interrupt in the middle of a "fake" SPI operation would easily corrupt the command being sent or the data being read from the FRAM. While it is possible to surround those operations with calls that disable interrupts, doing so could result in other malfunctions happening during those slow operations, such as data being lost coming in on a UART.

Ultimately, I am satisfied with how the project turned out. It was an opportunity for me to practice my hardware troubleshooting and embedded programming skills while still creating a product that ties back into the material learned in this class.