RP2040 Forth

M. Edward (Ed) Borasky

2022-07-21

Table of contents

Preface

This is a Quarto book.

To learn more about Quarto books visit https://quarto.org/docs/books.

1 Introduction

1.1 What is this?

As the name implies, *Raspberry Pi Pico Forth* is a Forth system for the Raspberry Pi Pico micro-controller. There are other Forth systems for the Raspberry Pi Pico, most notably *Mecrisp* (Koch 2022). So why write my own?

- The goal of this project is a live-coding environment for algorithmic composition and digital sound synthesis. It's not clear how to do that in Mecrisp.
- Mecrisp is complex - there's a large documentation site, but finding your way around it is daunting. I want something simpler, along the lines of FIG-Forth (Pintaske, Teal, and Ting 2020) *eForth* (Pintaske and Ting 2020a) or *CamelForth* (Pintaske and Rodriguez 2018).
- Although the initial system will be designed for the Raspberry Pi Pico, I want to run it on other ARM Cortex micro-controllers. The Pico is at its heart a dual-core ARM Cortex M0+. That is pretty much the simplest ARM Cortex micro-controller around.

  Most other micro-controllers aimed at digital audio have hardware floating point and some even have digital signal processing assist functionality. To be fair, Mecrisp does run on many other ARM Cortex micro-controllers.
- Raspberry Pi Pico C/C++ SDK (Raspberry Pi Ltd. (2022a); Raspberry Pi Ltd. (2022b))- I want to build using the excellent tooling and documentation the Raspberry Pi Foundation provides; my guidebook is Smith (2021). Mecrisp has its own build process.
- Integration with C/C++ libraries - there are numerous C and C++ libraries available for digital audio, and I don't want to re-invent them if I don't need to do so.

1.2 Why Forth?

There are a few other ways to program a Raspberry Pi Pico. Other than the C/C++ SDK, the two main ones are *MicroPython / CircuitPython* and the *Arduino IDE*. Both are capable of generating code that runs on a variety of micro-controllers.

The problem with MicroPython / CircuitPython is that I don't know Python. I can read Jupyter notebooks, but to actually write professional-level Python code, I'd need to learn more Python.

The Arduino IDE is more promising at first glance. It's designed for beginners, it runs on all the micro-controllers I'd want to use, including, of course, the Raspberry Pi Pico. There are numerous audio projects for the Arduino; see, for example, Edstrom (2016) and Cook (2015). And there's even a port of eForth to the Arduino IDE (Pintaske and Ting 2020b).

But the native language of the Arduino IDE is Arduino's own dialect of C++, and I'm no more capable of writing professional-level C++ than I am of writing professional-level Python. And the Raspberry Pi Pico C/C++ SDK tooling and documentation are better than those of the Arduino IDE.

Finally, there's no obvious way to write assembly code in the Arduino IDE. Writing an efficient Forth system requires either some complex C code, a complex meta-compiler written in Forth, or a small and simple assembly language kernel.

Forth is an interactive interpreter like Python, but it's a good bit faster for two reasons:

- The parser is much simpler; the syntax is Reverse Polish notation, and
- It's built from the ground up from assembly language primitive operations.

For a first course in Forth, see Brodie (2022).

Ultimately, writing a Forth system from the ground up in assembly language is more fun than wrangling a few thousand lines of someone else's code or learning a complex language that I have no other use for. I already know Forth, and I earned a living for many years writing assembly code.

1.3 Target hardware

The first version will run on the Pimoroni Pico Audio Pack. Later versions will run on the Teensy 4.1 and the Electro-Smith Daisy.

1.4 Initial feature set

- Assembly-language kernel ("inner interpreter" in Forth jargon). The threading model will be dictated by the requirement to run on all ARM Cortex-M micro-controllers.
- A Forth text interpreter ("outer interpreter" and "colon compiler") written in Forth. This will communicate with the user via the USB serial interface. There is also a serial interface on the Pico but I'm keeping that free for low-level debugging access.

- A Forth-style assembler for writing ARM Cortex M0+ code and code for all the specialized co-processors on the RP2040 CPU. See Smith (2021) for the details.
- Integration with the Pimoroni C/C++ audio pack libraries.

References

Brodie, Leo. 2022. "Starting Forth." Forth, Inc. https://www.forth.com/starting-forth/.

Cook, M. 2015. *Arduino Music and Audio Projects*. Technology in Action Series. Apress. https://books.google.com/books?id=kK1PCwAAQBAJ.

Edstrom, B. 2016. *Arduino for Musicians: A Complete Guide to Arduino and Teensy Microcontrollers*. Oxford University Press. https://books.google.com/books?id=WYLnCwAAQBAJ.

Koch, Matthias. 2022. "Mecrisp - Native code Forth for MSP430 and ARM Cortex." http://mecrisp.sourceforge.net/.

Pintaske, J., and B. Rodriguez. 2018. *Moving Forth - Internals and TTL Processor: Forth Internals*. Amazon Digital Services LLC - Kdp Print Us. https://books.google.com/books?id=RGqhxgEACAAJ.

Pintaske, J., S. Teal, and C. H. Ting. 2020. *FIG-Forth Manual: Documentation and Test in 1802 IP*. Independently Published. https://books.google.com/books?id=oFCVzQEACAAJ.

Pintaske, J., and C. H. Ting. 2020a. *EForth and Zen - 3rd Edition 2017: With 32-Bit 86eForth V5.2 for Visual Studio 2015*. Independently Published. https://books.google.com/books?id=ipi9zQEACAAJ.

———. 2020b. *EForth as Arduino Sketch: No Extra Programmer*. Independently Published. https://books.google.com/books?id=FPGazQEACAAJ.

Raspberry Pi Ltd. 2022a. "Getting started with Raspberry Pi Pico." https://datasheets.raspberrypi.com/pico/getting-started-with-pico.pdf.

———. 2022b. "Raspberry Pi Pico C/C++ SDK." https://datasheets.raspberrypi.com/pico/raspberry-pi-pico-c-sdk.pdf.

Smith, S. 2021. *RP2040 Assembly Language Programming: ARM Cortex-M0+ on the Raspberry Pi Pico*. Apress. https://books.google.com/books?id=lUyazgEACAAJ.