Should you Rust in embedded yet?

Who am I?

- Simonas Kazlauskas;
- Member of the Rust compiler team;
- Day job: firmware for electricity meters;
- kazlauskas.me;
- @nagisa on GitHub, @simukis on Twitter.

Why does embedded matter?

Embedded is virtually everywhere, including safety-critical applications:

- Automotive;
- Alarms;
- Medical machines (e.g. pacemakers);
- and many more.

Most of the embedded firmware is still written in C or C++, very little in Ada.

Why does embedded matter?

Most common bug classes are the same for both embedded and user-space firmware:

- Memory bugs (use-after-free, out-of-bounds reads and writes, ...);
- Data races;
- Logic errors;

Most of the bugs are more difficult to discover and debug in embedded firmware compared to user-space software.
Rust is well positioned to "solve" some of them in embedded-space the same way it did in the user-space programs.

Hardware support

It is imperative that Rust supports emiting code for the architecture used in the embedded project.

Architecture	Status
ARM	Built-in
MSP430	Built-in

Hardware support

It is imperative that Rust supports emiting code for the architecture used in the embedded project.

Architecture	Status
ARM	Built-in
MSP430	Built-in
AVR	Fork (avr-rust)
RISC-V	Fork (riscv-rust-toolchain)
32-bit PIC	No built-in target

Hardware support

It is imperative that Rust supports emiting code for the architecture used in the embedded project.

Architecture	Status
ARM	Built-in
MSP430	Built-in
AVR	Fork (avr-rust)
RISC-V	Fork (riscv-rust-toolchain)
32-bit PIC	No built-in target
16 & 8-bit PIC	No LLVM support
8051	No LLVM support
Custom ISAs	No LLVM support
⋮	⋮

Ecosystem

svd2rust, dslite2svd

Manufacturers provide headers and libraries to manipulate all the (usually memory-mapped) registers and functions.
Usually C/C++.

Ecosystem

svd2rust, dslite2svd

Manufacturers provide headers and libraries to manipulate all the (usually memory-mapped) registers and functions.
Usually C/C++.
Manufacturers also tend to provide descriptions in machine readable (e.g. XML-based) formats.
svd2rust generates Rust code to access registers described in SVD files.
m-labs/dslite2svd converts from Texas Instruments DSLite to SVD.

Ecosystem

embedded-hal

svd2rust does not directly or conveniently expose features in a convenient way. embedded-hal will provide traits to use common concepts (I2C, SPI, DMA, ...) in a safe manner.

Ecosystem

embedded-hal

svd2rust does not directly or conveniently expose features in a convenient way.
embedded-hal will provide traits to use common concepts (I2C, SPI, DMA, ...) in a safe manner.
Traits still need to be implemented for each microcontroller family, manually. Enables implementation of portable embedded drivers.

Ecosystem

Libraries

Catching up in coverage quickly. Fairly comprehensive already.

- TockOS a safety-focused real-time OS, less widespread than, say, FreeRTOS.
- smoltcp well documented, implemented and tested, but less tried competitor of lwIP.
- rtfm a nice framework for real-time embedded applications.
- Support crates for $\approx$ 40 MCUs.

Stability story

Not feasible to use stable compiler for `no_std`, yet alone embedded development in general. A number of important features are still unstable:

- `lang_items` (for `panic_fmt`, but also sometimes `sized`, `copy`, etc);
- `asm`, `core_intrinisics`;
- `start`;
- `linkage`...

Stability story

Embedded development needs stability of currently untracked parts of compilation pipeline – linking, target specification, et cetera.
Embedded development reacts strongly to code-size changes. Not technically breaking, but an increase in code size may cause code to not fit into MCU storage anymore.

Summary

Is it feasible to use Rust in embedded yet?

Yes.

Summary

Is it feasible to use Rust in embedded yet?

Yes. As long as it supports your hardware and you are willing to put up with a less stable toolchain and less mature ecosystem in exchange for the safety guarantees.