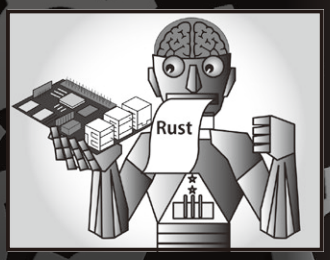
[特設記事] ダウンロード・データあります ご購入はこちら

ラズパイのデバイス・ドライバ作りを例に

安全性の高いRustをChatGPTで書いてしまおう



第2章 メモリ安全なRustでデバイス・ドライバ作り

高野 祐輝

表1 ファイルの一覧

ファイル	内 容
asm/boot.S	AArch64のブート・コード
src/main.rs	mainファイル
src/driver.rs	ドライバ定義ファイル
src/driver/gpio.rs	GPIOドライバ
src/driver/pl011.rs	UART PL011ドライバ
Cargo.toml	Cargoの設定ファイル
aarch64-custom.json	Cargoのカスタム・ビルド設定用
.cargo/config.toml	Cargoのエイリアス設定用
Makefile	Makefile
link.lds	リンカ・スクリプト

注：このうちデバイス・ドライバ本体であるgpio.rsとpl011.rsをChatGPTで生成した

ここからはベアメタル（OSなしの環境）で動作するGPIOと，UARTのRust版デバイス・ドライバをChatGPTで自動生成してみます．

ベアメタル環境で動作するデバイス・ドライバを作成するにはハードウェアの知識が必要です．また，C言語でデバイス・ドライバを書いたことがある方でも，Rustではまだ書いたことがない方も多いのではないでしょうか．そんな方でも，ChatGPTの助けがあればRust版デバイス・ドライバを作ることができます．（編集部）

ChatGPTで作るHello，world!

● やること…ラズパイ上で動作するプログラム作成

第1章でChatGPTの得意不得意を理解したところで，本格的なソフトウェアを実装していきます．ここでは，Rust言語を用いて，QEMU上のラズベリー・パイ3をベアメタルで利用してUARTシリアル・コンソールからHello，world!を出力します．ベアメタルとは，OSなしでプログラムを実行させることです．ここで実行するコードとコメントは極力ChatGPTに生成してもらいます．

本章で利用するファイルの一覧は**表1**の通りです．

リスト1 コンパイルと実行に必要なライブラリのインストール

```
$ sudo apt install build-essential clang qemu-
                                        system-arm
$ rustup toolchain install nightly
$ rustup default nightly
$ rustup component add rust-src llvm-tools-preview
$ rustup target add aarch64-unknown-none-softfloat
$ cargo install cargo-binutils
```

リスト2 Rustの設定ファイルCargo.toml

```
[package]
name = "rpi_baremetal"
version = "0.1.0"
edition = "2021"

# See more keys and their definitions at https://
    doc.rust-lang.org/cargo/reference/manifest.html

[dependencies]

[features]
raspi3 = []
raspi4 = []

[profile.dev]
panic = "abort"          ← 開発時のビルドの設定

[profile.release]
panic = "abort"          ← リリース時のビルドの設定
```

プログラムの作成

● 環境構築…ライブラリのインストール

作成に当たり，コンパイルと実行に必要なライブラリをインストールします．DebianかUbuntu Linuxを利用している場合，**リスト1**のようにしてインストールします．なお，Rustコンパイラのインストールは既に行っているものとします．

● ビルド設定Cargo.tomlの内容

ソースコードを生成する前に，必要な各種ファイルを準備します．Rustのビルドに必要なCargo.tomlは**リスト2**のようにします．重要なのは，パニック（復帰不可能な致命的エラー）が発生したときにアボートすることです（開発ビルドとリリース・ビルドの両

ダウンロード・データはこちら．
https://www.cqpub.co.jp/interface/download/contents.htm