第4章 実機やエミュレータでHello Worldからデバッグまで

Arm & RISC-Vマイコンを Rustで動かす

中林 智之

ここでは，Rustのマイコン・ファームウェア用パッケージ（Rustではクレートという）をビルドして，マイコン上で動かしてみます（**写真1**）．

まず，Rustのクロス・ビルド環境と，マイコンごとのデバッグ環境をセットアップします．次に，quickstartクレートを使って，RISC-VとCortex-Mとで，ファームウェアをビルド，実行してみましょう．

Rustマイコン開発環境の準備

● ツールチェーンの追加

rustupのデフォルト・インストールでは，ネイティブ環境のツールチェーンしかインストールされません．そこで，今回の実験内で利用するCortex-MとRISC-Vのツールチェーンを追加します．

GCCではターゲットごとにコンパイラを構築しなければなりませんが，Rustではrustupでターゲットに必要なコンポーネント（ビルド済みの標準ライブラリ）をインストールするだけです．Cortex-M系で困らないように，今回の実験で利用するCortex-M3とCortex-M4FやRISC-Vのターゲット向けにツールチェーンをインストールします．

▶ Cortex-M3とCortex-M4F

- thumbv6m-none-eabi # Cortex-M0およびCortex-M0+
- thumbv7m-none-eabi # Cortex-M3
- thumbv7em-none-eabi # Cortex-M4およびCortex-M7（FPUなし）
- thumbv7em-none-eabihf # Cortex-M4FおよびCortex-M7F（FPUあり）

▶ RISC-V

- riscv32imac-unknown-none-elf

次のコマンドで上記ツールチェーンを全てインストールできます．

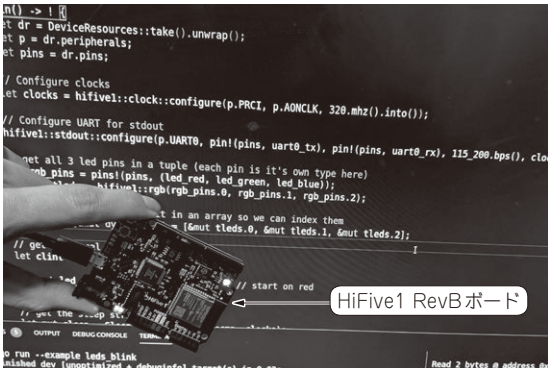
```
$ rustup target add thumbv6m-none-eabi thumbv7m-none-eabi thumbv7em-none-eabi thumbv7em-none-eabihf riscv32imac-unknown-none-elf
```

Rustでクロス・ビルドするための準備はこれで全てです．

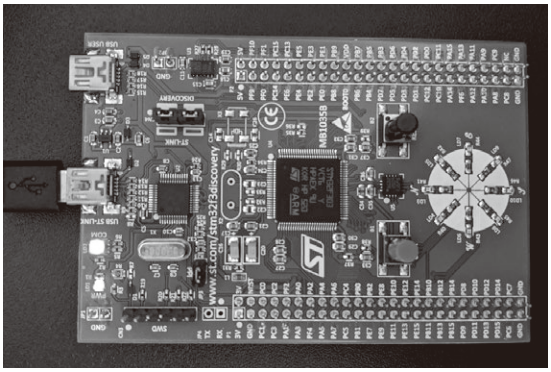
● Cargoサブコマンドのインストール

開発を便利にするためのCargoのサブコマンドをインストールします．Cargoにはサード・パーティ製のサブコマンドを追加できます．ここでは2つのサブコマンドを追加しておきましょう．

- cargo-generate
- cargo-binutils



（a）モダンな組み合わせ…RISC-Vマイコン×Rust



（b）定番STM32マイコンを動かす（STM32F3Discoveryボード）

写真1　Armマイコンや RISC-V マイコンを Rust プログラムで動かす