

第1章

STM32向けを流用して、
ラズベリー・パイPicoとRZマイコンで試す

開発の準備…環境構築と ビルド&拡張の手順

関本 健太郎



写真1 ラズベリー・パイPicoを使ってMicroPythonの拡張にトライ

第1章ではデバッグできるようになるところまで

● 速度や使い勝手が完璧ではないMicroPython

MicroPythonを使うと、素早くプログラムを作成し実行できる反面、特定のハードウェア・リソースにアクセスできなかったり、MicroPythonの中間言語によるオーバーヘッドで実行速度が期待に及ばなかったりすることがあります。

このような場合に、MicroPythonのモジュールの一部を、アセンブラやC、C++で記述することで、アクセスできるハードウェア・リソースを追加したり、処理速度を改善したりできます。

そこで本稿では、ラズベリー・パイPico(以降、Pico)を例に、MicroPythonの拡張モジュールの仕組みを解説し、拡張モジュールの作成方法について、実装例を示しながら説明します(写真1)。

さらに、第5章ではRZマイコン(ルネサス エレクトロニクス)を搭載したマイコン・ボード GR-MANGO向けのMicroPythonに、CPUレジスタ・アクセス・モジュール、LCDおよびカメラ処理クラスの実装方法を紹介します。実装したソースコードは本書ウェブ・ページから提供します。

MicroPythonの内部アーキテクチャ

MicroPythonの作者のYouTube⁽¹⁾によると、STM32向けのMicroPythonの内部アーキテクチャは、図1のようになります。ちなみにこの動画にはアーキテクチャの説明だけでなく、MicroPythonを高速化するヒントがたくさん説明されています。

● 幾つかのブロックに分けて考える

MicroPythonの内部アーキテクチャは、大きく分けると次の通りです。

- マイコンのブート機能
- MicroPythonのコンパイラ機能
- バイト・コードを実行するMicroPythonの仮想マシン
- ネイティブ・コードを直接実行するJust-In-Time機能(一部のマイコンでのみサポートされている)
- MicroPythonのランタイム・ライブラリ機能
- 外部バインディング・モジュール

● 第9部でトライすること

本稿では、図1の右側に示すモジュール追加による機能拡張をどのように行うのかを、幾つかの例を示しながら説明していきます。ちなみに、本稿の範囲外ですが、マイコン依存部は

- マイコンのブート機能
- ランタイム・ライブラリ中のMCU周辺機能

です。従ってMicroPythonを特定のマイコンに移植する際は、大きく分けてこの2つの部分を実装する作業になります。

特定のマイコンでMicroPythonの仮想マシン機能を動作させるだけなら、マイコンのブート部分を実装するだけなので、半日もあれば実装できるでしょう。例えばRP2040マイコン向けのrp2ポート(特定のマイコン向けに移植されたMicroPythonをポートと呼ぶ)では、MicroPythonは、Pico用のSDKをベースに、ベアメタル、つまりOSのない環境で動作しています。

■ダウンロード・データについて■

データは当時のMicroPythonのビルド全体をzipファイルにしたので、ビルド、実行はそのまま動作するはずですが、ただし、最新のMicroPythonとはファイル構成は一部変更されたり、APIが変更されたりしていますので、現状のものとは詳細部分で異なっています。