第1章　各コアに特定の処理を行わせる

ご購入はこちら

AMP方式「Try Kernel-A」の概要と実装

豊山 祐一

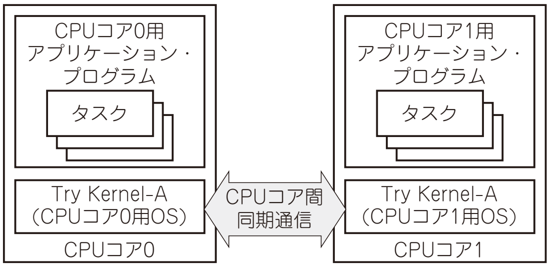

図1　Try Kernel-A（AMP方式）**のソフトウェア構成概要**
各CPUコア上で個別のOSのプログラムが動作し，このOSの上でそれぞれのCPUコア用のアプリケーション・プログラムが実行される

第3部ではAMP（Asymmetric Multi-Processing, 非対称マルチコア制御）方式のリアルタイムOSを作ります．AMP方式では，複数のCPUコアで独立にOSを実行し，CPUコア間の連携をとります．OSはTry Kernel 2.0をベースとし，名前はTry Kernel-Aと呼ぶことにします．

第1章ではシングルコアのTry KernelをAMP方式に対応させるための変更点と，ハードウェア・スピンロックやインターコアFIFOを使用したCPUコア間の低水準の同期通信機能について説明します．

第2章では各CPUコアでOSを実行するための起動処理と，Try Kernel-Aの起動までを説明します．

第3章ではTry Kernel-AのCPUコア間の同期通信の機能であるCPUコア間メッセージについて説明します．

シングルコアからマルチコアへ

● AMP方式OSのソフトウェア構成

PicoのマイコンRP2040はCortex-M0+のデュアルコアですので，AMP方式では2つのCPUコアの上でOSとアプリケーション・プログラムが実行されます．このAMP方式に対応したTry Kernel-Aのソフトウェア構成を**図1**に示します．

個々のCPUコアについて見れば，Try Kernel-Aのソフトウェア構成はシングルコアのTry Kernelと基本的に同じです．よって，基本的にはTry KernelにCPUコア間の同期通信の機能を拡張すれば，Try Kernel-Aが実現できます．

● 開発環境に合わせて実行コードは単一とする

AMP方式のプログラム開発は，CPUコアごとに独立した開発プロジェクトや実行コードを作成していくのが単純で明快です．ただし，これには開発環境がAMP方式に対応していなければなりません．

第1部から使ってきたPicoの一般的な開発環境は，開発プロジェクトと実行コードをCPUコアごとに分けることなく，全体で1つの開発プロジェクトと実行コードとします．

この方式は，複数のCPUコアのプログラムが単一の実行コードとして扱われるため，グローバルな関数や変数の名前を全体としてユニークとするなどの対応が必要となります．これは大規模なプログラム開発や多数のCPUコアを扱う際にはデメリットになるかもしれません．

一方で，実行コードの共通化が容易にできるので，メモリの使用量を節約しやすいというメリットがあります．

今回作成するTry Kernel-Aは，デュアルコア対応のシンプルなプログラムであり，単一の開発プロジェクトや実行コードでもさほど不便はありませんので，これまでのPicoの開発環境を使用することにします．これにより，シングルコアのプログラムや，この後に作成するSMP方式のOSと同じ開発環境を共有できます．

グローバル変数のマルチコア対応

● グルーバル変数の多重化

前述のように，Try Kernel-Aでは複数のCPUコアで動作するプログラムを単一の実行コードにまとめます．そのために，OSのグローバル変数をそれぞれのCPUコア用に用意しなければなりません．