OS起動処理の実装

豊山 祐一

本章では，それぞれのCPUコアで，第1章で作成したTry Kernel-Aを実行させる起動処理について説明します．

シングルコアの起動処理の流れ

OSの起動処理は，マイコンのハードウェア仕様に強く依存します．Picoの場合は外部のフラッシュ・メモリからブートするなど，一般的なマイコンとは少々異なった起動処理となります．そこで，まずはシングルコアでの起動処理を説明します．**図1**にシングルコアのTry Kernelの起動処理の流れを示します．処理の概要を順番に説明します．

▶①ファースト・ステージ・ブートローダ

Picoのマイコン RP2040のROM上のプログラムであり，ユーザが変更することはできません．

このプログラムの処理によりCPUコア1は休止状態となります．そして，外部フラッシュ・メモリ上のセカンド・ステージ・ブートローダの実行コードをSRAM上に転送して実行します．

▶②セカンド・ステージ・ブートローダ

フラッシュ・メモリ上のプログラムを直接実行可能に設定したのち，フラッシュ・メモリ上のユーザ定義のリセット・ハンドラを実行します．

▶③ユーザ定義リセット・ハンドラ

Try Kernelでは，このリセット・ハンドラで，必要最小限のハードウェアの初期化，グローバル変数などの初期化を行った後，Try Kernelのメイン関数を実行します．

▶④Try Kernelのメイン関数

Try Kernelのメイン関数は，最初に実行される特別なタスクである初期タスクの生成と実行を行います．

▶⑤初期タスク

初期タスクから，アプリケーション・プログラムのユーザ・メイン関数が実行されます．

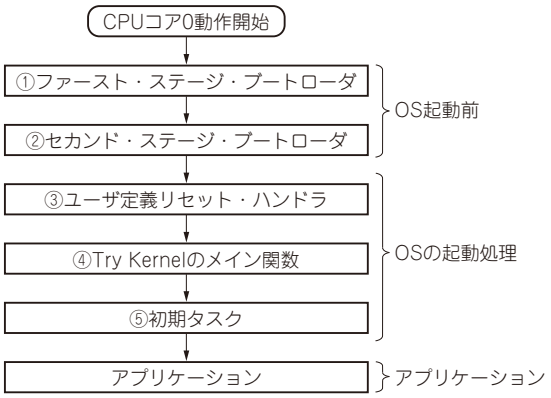

図1　シングルコアのTry Kernelの起動処理の流れ

Try Kernel-Aの起動処理の流れ

シングルコアのTry Kernelの起動処理を元にTry Kernel-Aの起動処理を作成していきます．

①および②のブートローダは，ハードウェアの仕様とも密接であり，特に①は変更できませんので，そのまま使用します．

③のユーザ定義リセット・ハンドラの処理において，休止状態のCPUコア1を起床させます．起床したCPUコア1は，CPUコア1用のユーザ定義リセット・ハンドラを実行します．

以降はそれぞれのCPUコアで④以降の起動処理を継続していきます．

Try Kernel-Aの起動処理の流れをまとめると，**図2**に示すようになります．それぞれの処理を順番に説明します．

● CPUコア0のユーザ定義リセット・ハンドラの処理

起動処理の流れの中で，シングルコアのTry KernelとTry Kernel-Aで違いがでるのは，ユーザ定義リセット・ハンドラの処理からです．