第4章

ライブラリ一覧とプログラム開発のポイント

PIOプログラミング［導入編］

宮田 賢一

本稿では，ラズベリー・パイPico（以降，Pico）の大きな特徴の1つであるプログラマブルI/O（PIO）を，MicroPythonを使ってプログラミングする方法について紹介します．

MicroPythonにはPIO用のライブラリも用意されている

Picoに搭載されているマイコンRP2040は，Cortex-M0+のプロセッサ・コアとは独立に，プログラマブルI/O（Programmable IO）と呼ばれる処理ユニットを2基搭載しています．

プログラマブルI/Oは独自の命令セットと専用のメモリを持ち，GPIOの信号を1サイクルの単位で処理可能な一種のプロセッサです．

そしてこのプログラマブルI/Oを使った処理を自由にプログラミングできるのが，Picoの大きな特徴の1つです．

PicoのMicroPythonでは，rp2モジュールにプログラマブルI/Oを使うためのライブラリが用意されていますので，ここではその使い方を詳しく説明します．

なお本稿執筆時点では，まだrp2モジュールの詳細な仕様が公開されていません．そこで公式ドキュメント(1)の情報に加え，筆者が独自にrp2モジュールのソースコードを解析した結果をもとに説明します．

表1　PIO命令のオペランドで使用可能な引数

引　数	意　味
x	スクラッチ・レジスタ（32ビット）
y	スクラッチ・レジスタ（32ビット）
pins	プログラマブルI/Oに割り当てたGPIOピンの値
pndirs	プログラマブルI/Oに割り当てたGPIOピンの方向
osr	出力シフト・レジスタ
isr	入力シフト・レジスタ
pc	プログラム・カウンタ．このレジスタに値をセットすると次のサイクルでレジスタの値のアドレスに分岐する
exec	次実行命令．このレジスタに値をセットすると，次のサイクルでレジスタの内容をPIO命令とみなして実行する
null	（out命令の場合）副作用のみ実行． （in命令の場合）定数ゼロを返す

● PIOは独立した小さなプロセッサ

本書第2部第1章の図1を見ると，Cortex-M0+コアとプログラマブルI/Oとは，AHB-Liteバスを通して接続されており，プログラマブルI/O内の送信FIFO/受信FIFOを介して，CPUとデータをやり取りします．

プログラマブルI/O内の4つの独立したステート・マシンは，独自の命令セットを持つプロセッサ・コアです．そしてステート・マシンはステート・マシン間で共有する命令メモリから命令を読み出し，必要に応じてFIFOのデータを送受信しながら，GPIO信号の入出力を行います．

また，FIFOやステート・マシンから割り込みを発生させ，RP2040の外部にあるデバイスとの同期処理を行えます．

● PIO命令に対応する関数

プログラマブルI/Oで使用できる命令は9種類であり，全ての命令は1サイクルで実行可能です．命令からアクセスできるレジスタの一覧を**表1**に示します．

MicroPythonでは各命令を関数呼び出しの形式で記述します．以降にPIO命令に対応するMicroPythonの関数仕様を記します．

▶命令の分岐：JMP

```
jmp(cond, label=None)
```

条件を満たしたときに，labelのアドレスに分岐します．condには**表2**（a）のいずれかを指定します．

▶処理待ち：WAIT

```
wait(polarity, src, index)
```

条件にマッチするまで処理をストールさせます．引数の意味を**表2**（b）に示します．

▶入力シフト・レジスタに書き込む：IN

```
in(src, bitcount)
```

srcで指定したレジスタの値をbitcountで指定しただけビット・シフトし，入力シフト・レジスタに書き込みます．srcに指定できるレジスタはpins，x，y，null，isr，osrです．各レジスタの意味は**表1**を参照してください．