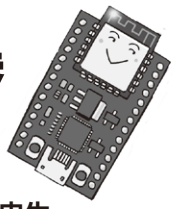
ダウンロード・データあります

700円マイコンESP32ではじめる 逆引きMicroPythonプログラム集



第10回 消費電流を半分以下に節約！ 省電力モードの使い方

角 史生

● ESP32に用意されているスリープ・モード

MicroPythonでは，ESP32用の省電力機能として，ライト・スリープ・モードとディープ・スリープ・モードの2つが用意されています．2つのスリープ・モードは，消費電力と復帰時の挙動が異なります．

▶ライト・スリープ・モード

ライト・スリープ・モードでは，ULP（Ultra Low Power）コプロセッサとリアルタイム・クロック（RTC）は稼働を続けます．CPUコアとメモリは，クロック・ゲーティングと呼ばれる手法により状態を保持したままで一時停止します．

このため，ライト・スリープから復帰する場合，リセットはされず，一時停止した次のプログラムから継続実行されます．

▶ディープ・スリープ・モード

ディープ・スリープ・モードでは，ULPコプロセッサとリアルタイム・クロック（RTC）のみ稼働し，CPUコアとメモリは電源OFFになります．

ディープ・スリープは，省電力効果が大きい反面，復帰後は電源OFF/ON時と同じようにハードウェア・リセットされた状態になります．このため，プログラムは初期状態に戻ります(1)(2)．

リスト1　一定時間後にライト・スリープ・モードから復帰するプログラム

```
import machine

print("start to sleep(1sec)")
machine.lightsleep(1000)  # 1000ms指定
print("wake up from sleep")
```

（a）ソースコード

```
start to sleep(1sec)
wake up from sleep
```

（b）REPLでの実行結果

リスト2　ライト・スリープ・モードから復帰した要因を特定するプログラム

```
import machine
def print_wakeup_reason():
    id = machine.wake_reason()
    print(id)
    if id == machine.EXT0_WAKE:
        print("EXT0_WAKE")
    elif id == machine.EXT1_WAKE:
        print("EXT1_WAKE")
    elif id == machine.PIN_WAKE:
        print("PIN_WAKE")
    elif id == machine.TOUCHPAD_WAKE:
        print("TOUCHPAD_WAKE")
    elif id == machine.TIMER_WAKE:
        print("TIMER_WAKE")
    else:
        print("unknown")
```

（a）ソースコード

```
>>> print_wakeup_reason()
4
TIMER_WAKE
```

（b）REPLでの実行結果

8-1　ライト・スリープ・モードの使い方

■ 基本…指定時間経過後に復帰

● タイマで復帰する方法

リスト1に示すのは，ライト・スリープ・モードを使ったプログラムの例と，それをREPLで実行した結果です．

一定時間経過後に復帰したい場合は，時間を指定してライト・スリープ関数を実行します．指定時間はms単位です．**リスト1**(a)の場合，1秒経過後に復帰し，ライト・スリープ関数lightsleep()の次の行から実行が継続されます．0を指定した場合，タイマでは起動しないので，外部からの割り込みで復帰する必要があります．

● スリープから復帰した要因を表示する方法

machineモジュールのwakeup_reason()関数を使うと，スリープから復帰した要因が特定できます．**リスト2**(a)に示すのが要因を表示する関数を使ったプログラムの例です．**リスト2**(b)に示すのは，**リスト2**(a)を実行した後に，print_wakeup_