スイッチ入力で LED 点灯

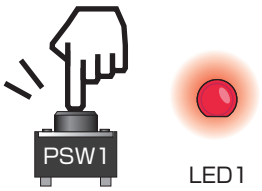
学習内容

マイコンボードの入力ピンを利用して，外部機器（スイッチ）の状態を取得します。
本 STEP で学習するスイッチの状態は ON か OFF です。

課題 07-1

以下のように LED の点灯をスイッチで操作してみましょう。

スイッチ ON で LED 点灯

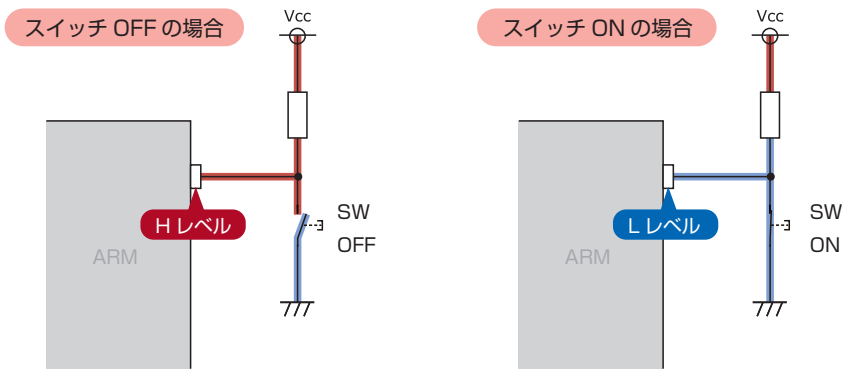


スイッチ OFF で LED 消灯

⚠ 本書では，スイッチ一般を SW，プッシュスイッチを PSW と表記しています。

スイッチの状態を取得するには

下図はスイッチをマイコンと接続した場合の回路です。
スイッチの状態により，回路上の電圧レベルが変化します。

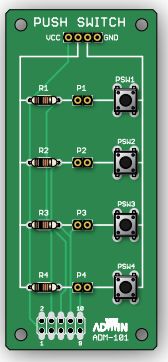


スイッチが接続された入力ピンの電圧が分かれば，スイッチの状態を取得することができます。

スイッチ入力で LED 点灯

配線 07-1

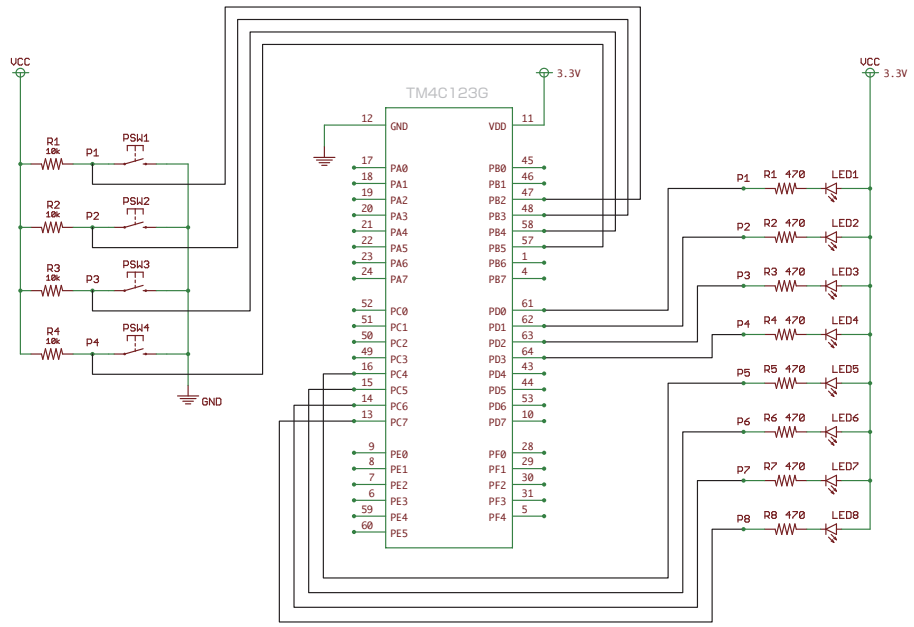
今回はマイコンボードとプッシュスイッチ(PSW)ボード, LED ボードを使用します。これらのボードはベースボード上で以下のように接続されています。



PSW ボード		マイコンボード		LED ボード
P1 - PSW1	⟷	2.02	PB2	
P2 - PSW2	⟷	4.03	PB3	
P3 - PSW3	⟷	1.07	PB4	
P4 - PSW4	⟷	1.02	PB5	
		3.03	PD0	⟷ P1 - LED1
		3.04	PD1	⟷ P2 - LED2
		3.05	PD2	⟷ P3 - LED3
		3.06	PD3	⟷ P4 - LED4
		4.04	PC4	⟷ P5 - LED5
		4.05	PC5	⟷ P6 - LED6
		4.06	PC6	⟷ P7 - LED7
		4.07	PC7	⟷ P8 - LED8
VCC	⟷	1.01	+3.3V	⟷ VCC
GND	⟷	2.01	GND	

マイコンボードと各ボードをベースボードから取り外し，ジャンパワイヤ（別売）で直接接続することも可能です。時間に余裕のある場合は，STEP 01 のピンアサインに注意しつつ，お好みのピンで課題を実現してみましょう。

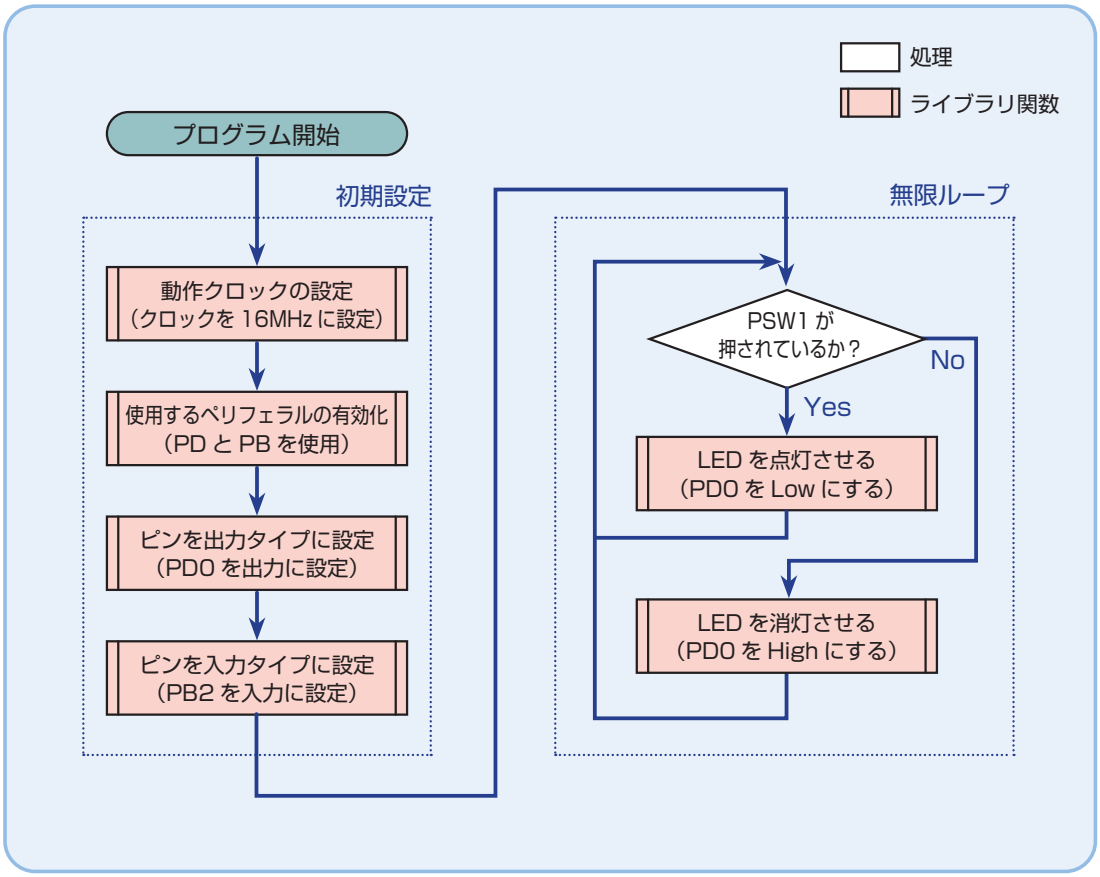
回路図



スイッチ入力で LED 点灯

フローチャート　07-1

フローチャートは以下のようになります。



スイッチ入力で LED 点灯

ライブラリ関数　07-1

STEP 07-1 で使用するライブラリ関数を解説します。
関数の引数は ui32 ～が「符号無し 32 ビット整数」，ui8 ～が「符号無し 8 ビット整数」です。
ライブラリ関数は TivaWare で提供されています。

ピンを入力タイプに設定　GPIOPinTypeGPIOInput(ui32Port, ui8Pins)

指定したピンを入力タイプに設定する。

```
GPIOPinTypeGPIOInput(GPIO_PORTB_BASE, GPIO_PIN_0);
```

設定項目 例

- GPIO_PORTB_BASE　：入力設定にしたいピンが属しているポートの設定値。
  例は B ポート。
- GPIO_PIN_0　：ピン番号の設定値。0 ～ 7 まで指定できる。
  例は 0 番ピン。

ピンの入力値を取得　GPIOPinRead(ui32Port, ui8Pins)

指定したピンの入力値を取得する。入力値が High(1) の場合は該当ビットが 1 の 32 ビット整数を返す。
入力値が Low(0) の場合は 0 を返すことになる。

```
GPIOPinRead(GPIO_PORTB_BASE, GPIO_PIN_0);
```

設定項目 例

- GPIO_PORTB_BASE　：状態を取得したいピンが属しているポートの設定値。
  例は B ポート。
- GPIO_PIN_0　：ピン番号の設定値。0 ～ 7 まで指定できる。
  例は 0 番ピン。

```
if (GPIOPinRead(GPIO_PORTB_BASE, GPIO_PIN_0) == 0)
{
    // PB0 が 0 (ON) の時
} else {
    // PB0 が 1 (OFF) の時
}
```

スイッチ入力で LED 点灯

コーディング 07-1

フローチャートを元に，ソースを記述してください。ソースが完成したら，実行して動作を確認しましょう。
以下に解答例ソースを示します。解答例やサンプルソースを参考に，皆さんで工夫してみてください。

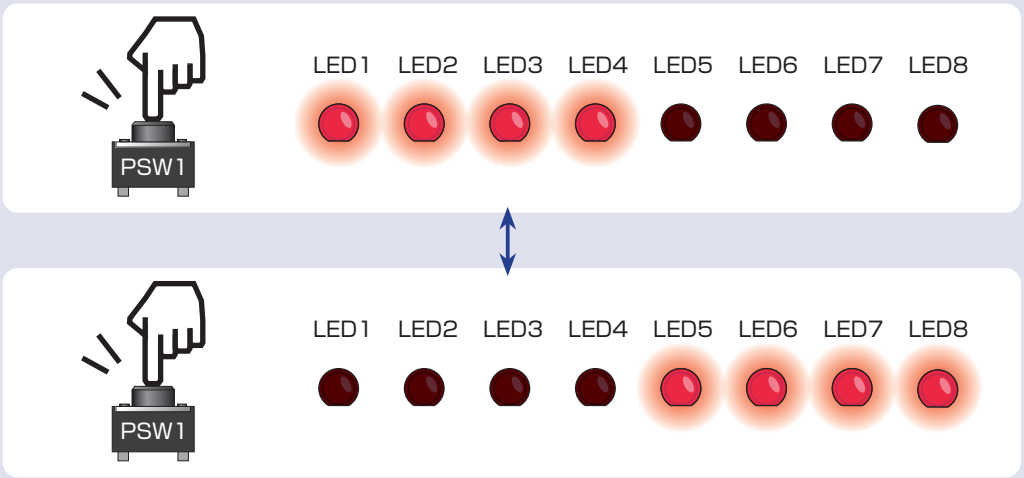
step07-1.c　CD-ROM の「サンプルソース」フォルダに、各ステップの c ファイルを収録しています

```
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_types.h"
#include "inc/hw_memmap.h"
#include "driverlib/gpio.h"
#include "driverlib/sysctl.h"

// LED 点灯 / 消灯用マクロ
#define ON  0x00
#define OFF 0xFF

// スイッチで LED を点灯させる
void main(void) {
        // 動作クロックの設定
        SysCtlClockSet(SYSCTL_SYSDIV_1 | SYSCTL_USE_OSC | SYSCTL_OSC_MAIN | SYSCTL_XTAL_16MHZ);

        // 使用するペリフェラルの有効化
        // : LED 用に I/O ポート D を使用
        SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
        // : スイッチ用に I/O ポート B を使用
        SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);

        // PD0 を出力タイプに設定
        GPIOPinTypeGPIOOutput(GPIO_PORTD_BASE, GPIO_PIN_0);

        // PB2 を入力タイプに設定
        GPIOPinTypeGPIOInput(GPIO_PORTB_BASE, GPIO_PIN_2);

        while (1) {
                // PSW1 が押されているか判定
                if (GPIOPinRead(GPIO_PORTB_BASE, GPIO_PIN_2) == 0) {
                        // LED を ON
                        GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_0, ON);
                } else {
                        // LED を OFF
                        GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_0, OFF);
                }
        }
}
```

スイッチ入力で LED 点灯

課題 07-2

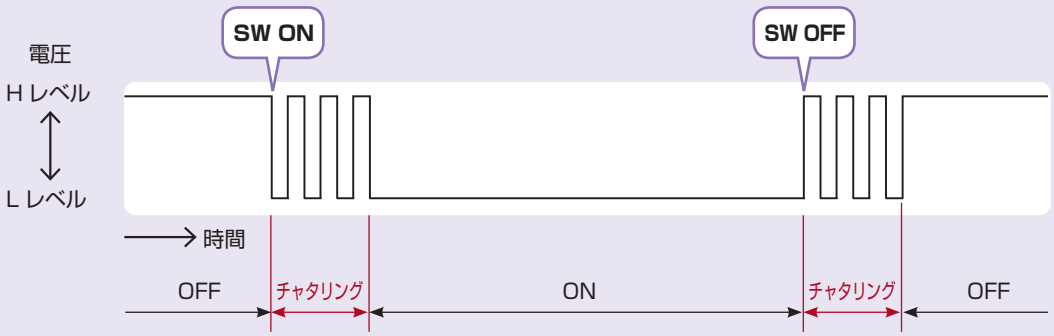
PSW1 を押すたびに，点灯パターンが交互に入れ替わるようにしましょう。
プログラムに チャタリング 対策が必要です。



解答は，巻末の解答例集参照

チャタリング

チャタリングとは，SW の ON/OFF が切り替わる際に，接点がバウンドしてしまう現象です。

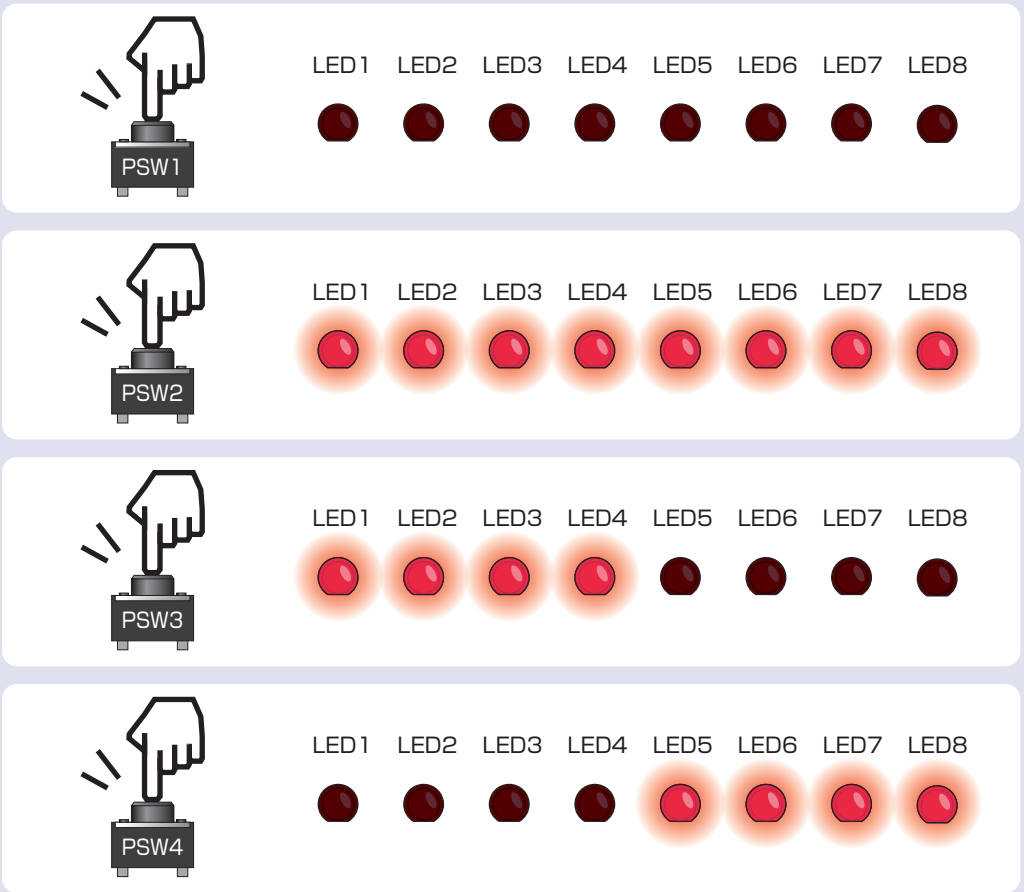


1 回だけ SW を ON/OFF したつもりでも，非常に短い時間内で何度も接点がバウンドし ON/OFF を繰り返してしまいます。これはハードの問題なのですが，プログラムで解決する方法がいくつかあります。
その 1 つが，チャタリングが起きている間，待ち時間を入れるという方法です。
エレモで使用している PSW の場合（機械的な劣化などにもよりますが）チャタリングの時間は 10 ～ 20 ミリ秒程度です。

スイッチ入力で LED 点灯

課題 07-3

PSW1，PSW2，PSW3，PSW4 を押すと，それぞれ以下のように点灯するようにしましょう。



解答は，巻末の解答例集参照