STEP 06

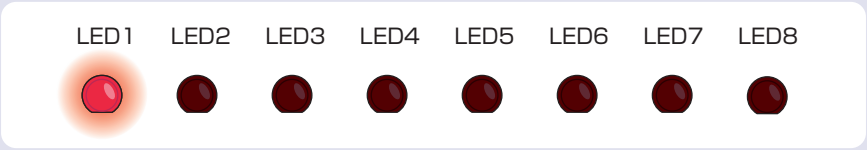
LED 点灯

学習内容

マイコンボードの入出力ピンを利用して，外部機器 (エレモ LED) を制御します。
本 STEP では LED の点灯および消灯を学習します。

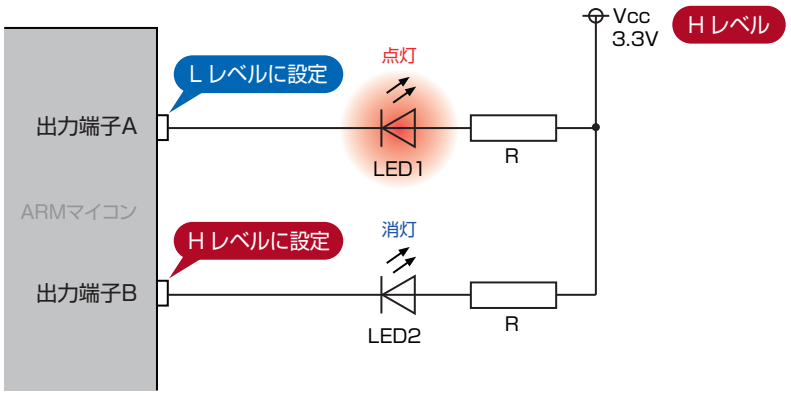
課題 06-1

LED を下記のパターンで点灯させましょう。



LED を点灯させるには

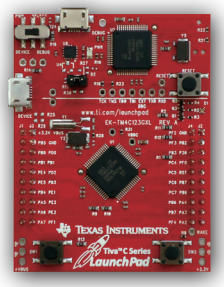
本書で利用するマイコンは，最大 43 個の端子を入出力端子として利用できます。入出力端子とは，入力端子または出力端子として使用可能な端子です。
入力端子とは，端子の電圧レベルをマイコンに伝えることができる端子で，出力端子とは，マイコンで設定した電圧レベルに端子の電圧レベルを設定できる端子です。電圧レベルには，High レベル (H レベル，ここでは +3.3V) と Low レベル (L レベル，0V = GND) の2つがあります。
LED を制御する場合，接続する端子を出力端子に設定します。



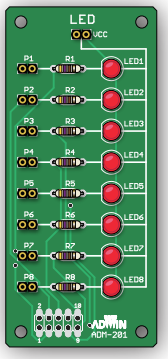
上記のような回路では，
出力端子 A を L レベル，出力端子 B を H レベルにすると，LED1 は点灯し，LED2 は点灯しません。
出力端子の電圧レベル設定をプログラムで行うことにより，LED の点灯消灯を制御できるのです。

配線 06-1

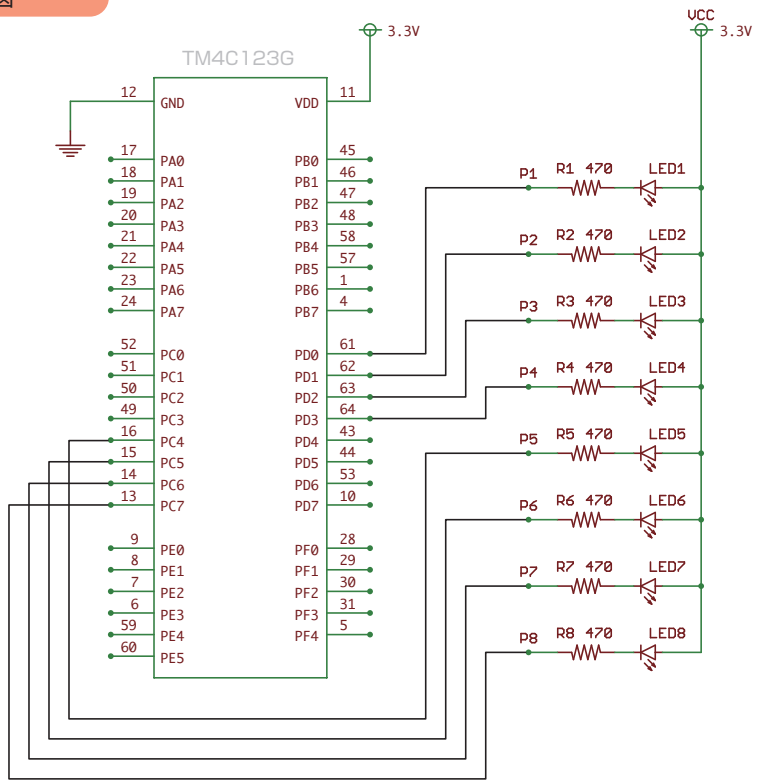
今回はマイコンボードと LED ボードを使用します。これらのボードはベースボード上で以下のように接続されています。



マイコンボード			LED ボード
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
1.01	+3.3V	⟷	VCC



回路図



LED 点灯

配線 06-1

LED ボードには 8 つの LED があり，これらを操作するには 8 つの入出力端子が必要です。

TM4C123GH6PM には A ～ F の 6 つの 8 ビット汎用入出力ポートが備わっており，各 I/O ポートが原則として 8 個の入出力端子として使用可能です。しかし，STEP 01 のピンアサインによれば，実際に使用可能な端子には制約があります。マイコンボード上のピンに 8 個の端子すべてが接続されているのはポート B のみですが，これも PB0，PB1 の利用に一定の注意が必要です（このほかに PB6 と PD0，PB7 と PD1 がそれぞれ基板上でショートされていることにも注意してください）。本書の範囲内ではポート B の 8 個の端子すべてを LED 操作に利用することも可能ですが，特に事情が無ければ PB0，PB1 の利用は避け，複数の I/O ポートを併用して LED 操作を行うことが望ましいでしょう。

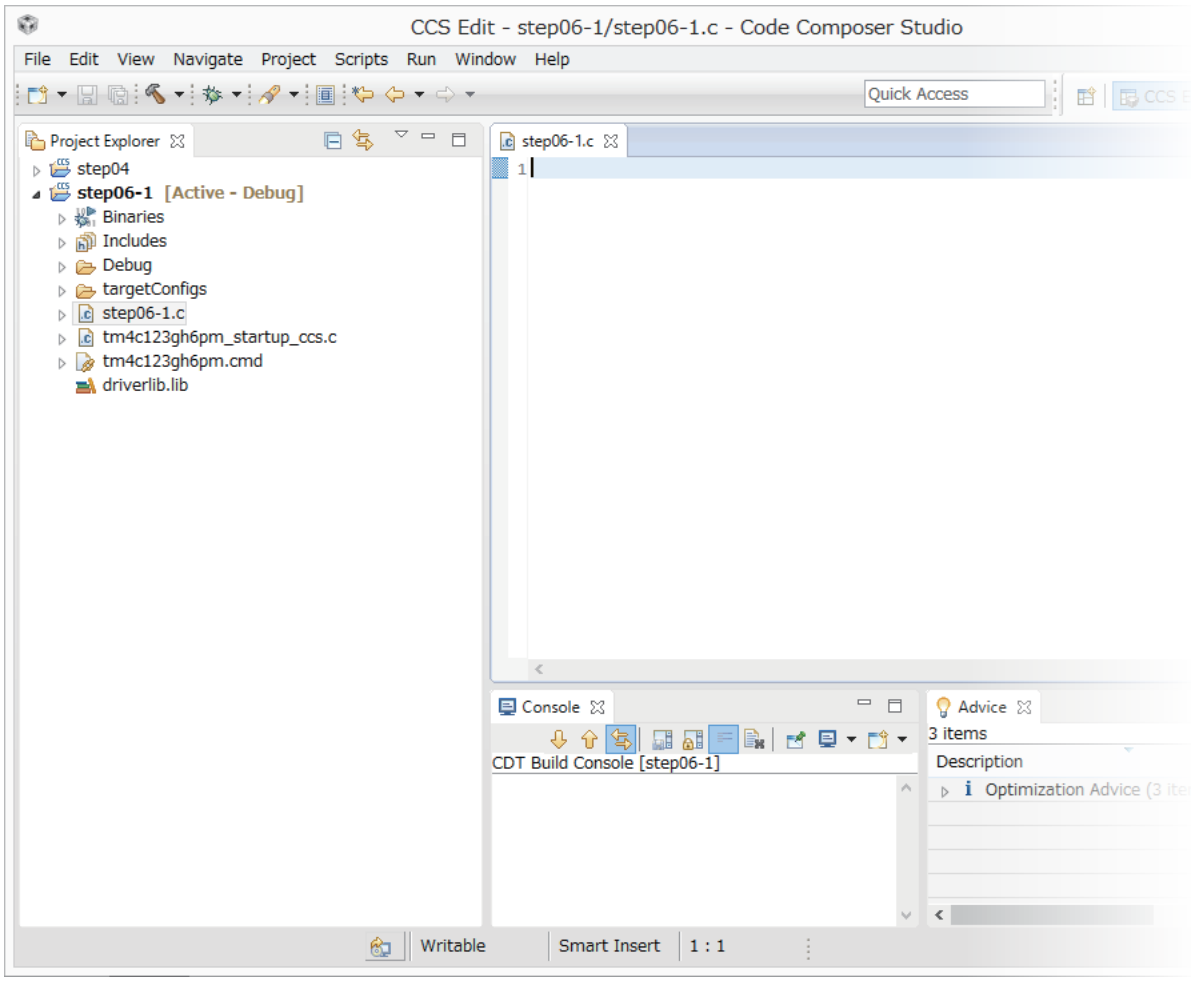
実際に使用する I/O ポートの端子は，利用可能なものであればどれでもかまいませんが，本書付属のベースボードではポート D とポート C を併用し，LED1 ～ LED4 の操作を PD0 ～ PD3 端子で，LED5 ～ LED8 の操作を PC4 ～ PC7 端子で行うようにしています。

ここでは，マイコンボードの PD0 で LED ボードの LED1 を操作することになります。また，LED への電源はマイコンボードの +3.3V (1.03) から LED ボードの VCC へと供給しています。

マイコンボードと LED 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。LED ボードの LED2 ～ LED8 は今回は使用しないので，接続しなくてもかまいません。

プロジェクト作成　06-1

「step04」プロジェクトと同じワークスペースに「step06-1」というプロジェクトを作成しましょう。



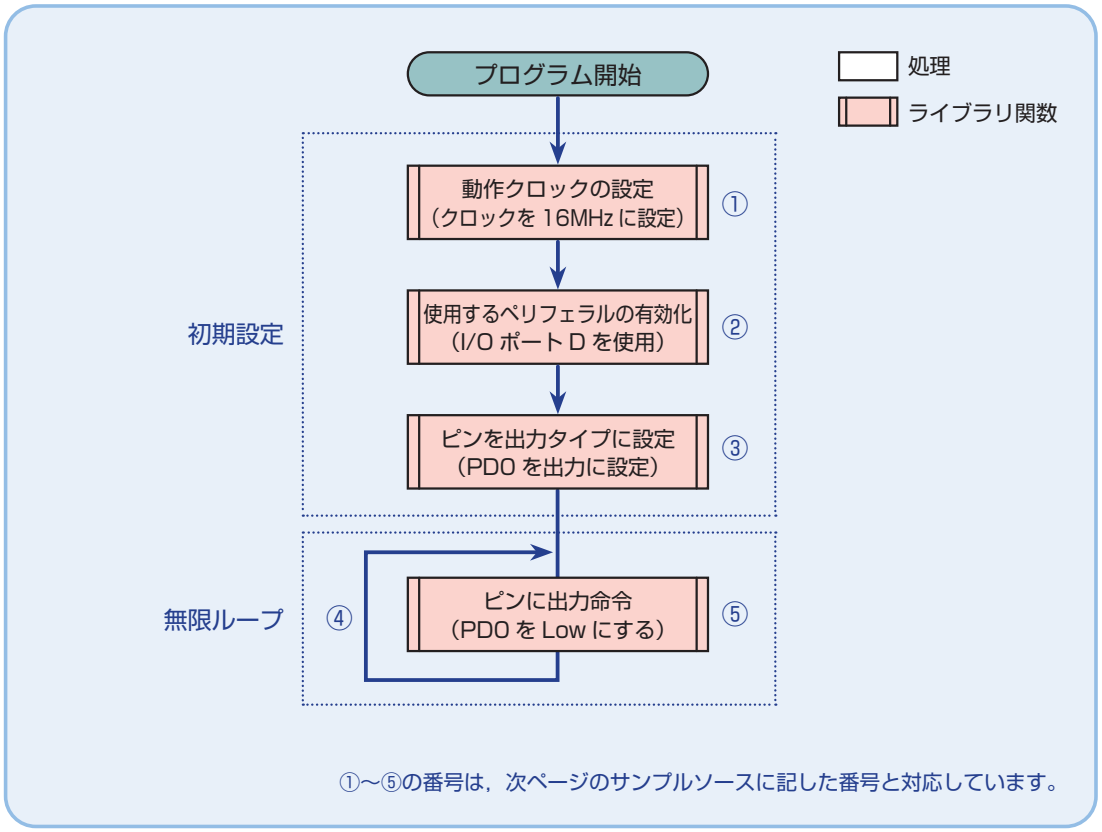
新規作成でも step04 の複製でも構いません。分からない方は STEP 04 を参考にしてください。
「step06-1」プロジェクト作成後，プロジェクトビューに上図のように表示されていれば OK です。

同じワークスペースに複数のプロジェクトがある場合，編集中のファイルの属するプロジェクトまたは Project Explorer ビューで選択されているファイルの属するプロジェクトが自動的にアクティブプロジェクト（ビルドやデバッグを行う対象のプロジェクト）になります。

LED 点灯

フローチャート　06-1

配線の段階で，どの端子を出力端子にして LED を制御するかというハードウェア設計は済んでいます。後はソフトウェア設計ですが，コーディングを始める前に，マイコンに行わせる処理の流れをフローチャートでまとめておきましょう。
フローチャートは以下のようになります。
プログラムは，初期設定部分と無限ループ部分からなります。ここでは「ピンに命令出力」を無限ループの中に置いていますが，無限ループの手前（初期設定部分の最後）でもかまいません。



①～⑤の番号は，次ページのサンプルソースに記した番号と対応しています。

コーディング　06-1

フローチャートを元に，ソースを記述してください。
step04.c のソースを元に，自力で考えてみると良いでしょう。また，使用するライブラリ関数を次ページから紹介していますので，ライブラリ関数を学習してからソースを考えてみてもいいでしょう。
以下に，解答例ソースを示します。

step06-1.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

```
1  #include <stdint.h>
2  #include <stdbool.h>
3  #include "inc/hw_types.h"
4  #include "inc/hw_memmap.h"
5  #include "driverlib/gpio.h"
6  #include "driverlib/sysctl.h"
7
8  // LED 点灯 / 消灯用マクロ
9  #define ON  0x00
10 #define OFF 0xFF
11
12 // LED1 を点灯させる
13 void main(void) {
14         // 動作クロックの設定
15         SysCtlClockSet(SYSCTL_SYSDIV_1 | SYSCTL_USE_OSC | SYSCTL_OSC_MAIN | SYSCTL_XTAL_16MHZ); ①
16
17         // 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
18         SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD); ②
19
20         // PD0 ピンを出力タイプに設定
21         GPIOPinTypeGPIOOutput(GPIO_PORTD_BASE, GPIO_PIN_0); ③
22
23         // 無限ループ
24         while (1) {
25                 // LED を ON
26                 GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_0, ON); ⑤
27         }
28 }
```

インクルードおよびマクロ（コンパイル時に必要）

④

このソースは解答例です。あくまで例ですから，この通りでなければならないことはありません。
また，目的を実現するフローチャートやプログラムの正解は 1 つではありません。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。

LED 点灯

インクルードファイル　06-1

STEP 06-1 で使用するインクルードファイルを解説します。

標準ヘッダファイル　stdint.h

```
#include <stdint.h>
```

C 言語の標準ヘッダファイルのひとつで，指定幅の整数型を宣言しています。今回のプログラムでは直接は使っていませんが，Tiva のヘッダファイル内で使用されるため，プログラム冒頭でインクルードする必要があります。

標準ヘッダファイル　stdbool.h

```
#include <stdbool.h>
```

C 言語の標準ヘッダファイルのひとつで，論理型（ブーリアン型）を宣言しています。今回のプログラムでは直接は使っていませんが，Tiva のヘッダファイル内で使用されるためプログラム冒頭でインクルードする必要があります。

Tiva ヘッダファイル　inc/hw_types.h

```
#include "inc/hw_types.h"
```

Tiva のヘッダファイルで，ハードウェアにアクセスするためのマクロが宣言されています。プログラム中では直接は使用しません。

Tiva ヘッダファイル　inc/hw_memmap.h

```
#include "inc/hw_memmap.h"
```

Tiva のヘッダファイルで，デバイスのメモリマップのマクロが宣言されています。今回のプログラムでは，以下のマクロがここで宣言されています。

- GPIO_PORTD_BASE

Tiva ヘッダファイル　driverlib/gpio.h

```
#include "driverlib/gpio.h"
```

Tiva のヘッダファイルで，GPIO ピンを操作するためのマクロおよび関数が宣言されています。今回のプログラムでは，以下のマクロおよび関数がここで宣言されています。

- GPIO_PIN_0
- GPIOPinTypeGPIOOutput(ui32Port, ui8Pins)
- GPIOPinWrite(ui32Port, ui8Pins, ui8Val)

Tiva ヘッダファイル　driverlib/sysctl.h

```
#include "driverlib/sysctl.h"
```

Tiva のヘッダファイルで，システム制御のためのマクロおよび関数が宣言されています。今回のプログラムでは，以下のマクロおよび関数がここで宣言されています。

- SYSCTL_SYSDIV_1
- SYSCTL_USE_OSC
- SYSCTL_OSC_MAIN
- SYSCTL_XTAL_16MHZ
- SYSCTL_PERIPH_GPIOD
- SysCtlClockSet(ui32Config)
- SysCtlPeripheralEnable(ui32Peripheral)

LED 点灯

ライブラリ関数 06-1

STEP 06-1 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。

動作クロックの設定 SysCtlClockSet (ui32Config)

ARM マイコンの動作クロックを設定する。動作クロックの設定を行わなかった場合は，16MHz ± 3% のマイコン内蔵クロックが使用される。

```
SysCtlClockSet(SYSCTL_SYSDIV_1 | SYSCTL_USE_OSC | SYSCTL_OSC_MAIN | SYSCTL_XTAL_16MHZ);
```

設定項目 例

- SYSCTL_SYSDIV_1 ：システムクロックの分周器の設定値。1/1 ～ 1/64 まで設定できる。例は 1/1 の設定。
- SYSCTL_USE_OSC ：システムクロックに外部クロックを使用する設定値。
- SYSCTL_OSC_MAIN ：外部クロックの用途の設定。例はメインで使用する設定値。
- SYSCTL_XTAL_16MHZ ：外部クロックの周波数の設定値。本書で使用するマイコンボードには 16MHz ± 0.005% の外部クロックが実装されている。

使用するペリフェラルの有効化 SysCtlPeripheralEnable (ui32Peripheral)

指定したペリフェラル（ここでは I/O ポート）を有効にする。I/O ポートを使用するにはこの関数で有効にする必要がある。有効にせずにポートを使用すると，実行時にエラー (fault) が起こるので注意すること。

```
SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
```

設定項目 例

- SYSCTL_PERIPH_GPIOD ：例は GPIO（汎用入出力）ポートの PD を使用するときの設定値。

この他にも以下のような設定がある。

SYSCTL_PERIPH_GPIOA ：GPI/O ポートの PA を有効化
SYSCTL_PERIPH_ADC0 ：A/D 変換器 0 を有効化
SYSCTL_PERIPH_TIMER0 ：タイマ 0 を有効化

ペリフェラル

Peripheral を直訳すると「周辺機器」。本来は外部に接続された機器を指すが，マイコンの場合は内蔵された機器（I/O ポートや A/D 変換機，タイマなど）もペリフェラルと呼ばれる。

ピンを出力タイプに指定　GPIOPinTypeGPIOOutput (ui32Port, ui8Pins)

指定したピンを出力タイプに設定する。

```
GPIOPinTypeGPIOOutput(GPIO_PORTD_BASE, GPIO_PIN_0);
```

設定項目 例

- GPIO_PORTD_BASE　：出力設定にしたいピンが属しているポートの設定値。
  例はDポート。
- GPIO_PIN_0　：ピンを指定する。0 ～ 7 まで指定できる。
  例は 0 番ピン。

ピンの出力設定　GPIOPinWrite()

指定したピンの状態を 0(Low) または 1(High) にする。

```
GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_0, ON);
```

設定項目 例

- GPIO_PORTD_BASE　：状態を変更したいピンが属しているポートの設定値。
  例はDポート。
- GPIO_PIN_0　：ピンを指定する。0 ～ 7 まで指定できる。
  例は 0 番ピン。
- ON　：ピンの状態を指定する。
  0(Low) にしたい時は 0 を，1(High) にしたいときは該当ビットが 1 の値を設定する。
  例の ON はソース内で　`#define ON  0x00`　と定義しているので，0 (Low) になる。

ライブラリ関数を詳しく調べる

「TivaWare Peripheral Driver Library　User's Guide」を参照してください。
関数名で検索すると素早くアクセスできます。PDF は付属 CD-ROM の「参考資料」フォルダに収録されています。

LED 点灯

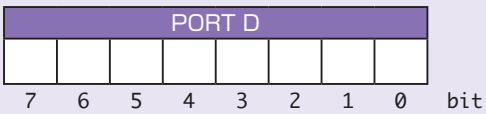
ピンの出力設定値について

以下の３つの関数は，いずれもポート D の３番ピンを 1 (High) に設定する正しい引数が設定されています。

```
GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_3, 0x08);
GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_3, 0xFF);
GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_3, GPIO_PIN_3);
```

どうして３番ピンを 1 (High) にするのに「0x08」や「0xFF」を設定するのでしょうか。

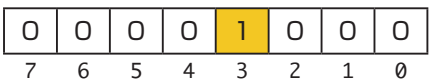
ポート D は 8 ビットのレジスタで，
8 個のマスのある右図のようなイメージです。



GPIOPinWrite( ) は，1 番目の引数で指定したポート（ポート D）に，2 番目の引数で指定したビット（3 番目）のみ有効にして，3 番目の引数で指定した値を書き込む関数です。
下図の黄枠は，指定したビット（3 番目）の指定した値です。

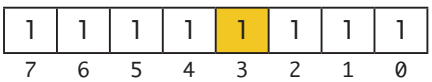
A

16 進数：0x08 を指定する
＝ 2 進数：0000 1000



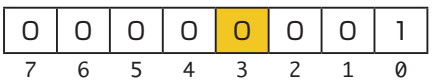
B

16 進数：0xFF を指定する
＝ 2 進数：1111 1111



C

1 を指定する ＝ 16 進数：0x01
＝ 2 進数：0000 0001



レジスタ

コンピュータのプロセッサが内部に保持する少量で高速な記憶装置。メモリはプロセッサ外にある記憶装置。

A は，0x08 による指定が正しく 1 (High) になることがわかります。
B は，step06-1.c の #define OFF 0xFF による指定例で，これも OK です。
C は正しくない例です。C はよくある間違いなのでご注意ください。

2番目の引数の GPIO_PIN_3 は，3ビット目だけに値を書き込むという設定ですが，この GPIO_PIN_3 は gpio.h の中で以下のように定義されています。

```
#define GPIO_PIN_0    0x00000001  // GPIO pin 0
#define GPIO_PIN_1    0x00000002  // GPIO pin 1
#define GPIO_PIN_2    0x00000004  // GPIO pin 2
#define GPIO_PIN_3    0x00000008  // GPIO pin 3
#define GPIO_PIN_4    0x00000010  // GPIO pin 4
#define GPIO_PIN_5    0x00000020  // GPIO pin 5
#define GPIO_PIN_6    0x00000040  // GPIO pin 6
#define GPIO_PIN_7    0x00000080  // GPIO pin 7
```

32 ビットで定義されていますが，上位ビットは 0 で下位 2 桁（8 ビット分）しか使われておらず
GPIO_PIN_3 = 0x08 と同じです。
この定義を利用すると，ピンごとに指定する値を考えなくても

```
GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_3, GPIO_PIN_3);
```

と書けば指定ピンを 1(High) にすることができるのです。プログラム的には A の例と同じ意味になります。
ピンの出力値設定やその他の設定は，最終的にはポートのレジスタに 8 ビットの数値を書き込むことで行われますが，ライブラリ関数を使うことで分かりやすい形でのピンの設定が可能になります。
なお，設定するピンを指定する GPIOPinWrite() の第 2 引数や GPIOPinTypeGPIOPinOutput( ) の第 2 引数も，実は同様に設定するピンをビットで指定しています。そのため，例えば

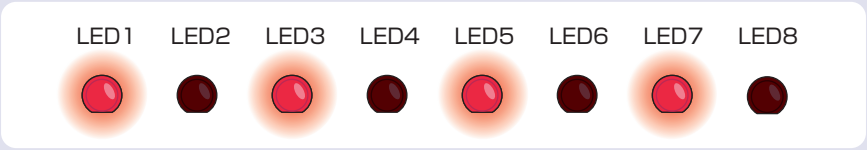
```
GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_2 | GPIO_PIN_3, 0xFF);
```

のように設定すれば，ポート D の 2 番ピンと 3 番ピンを同時に 1(High) にすることができます。

LED 点灯

課題 06-2

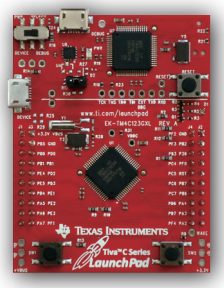
LED を下記のパターンで点灯させましょう。



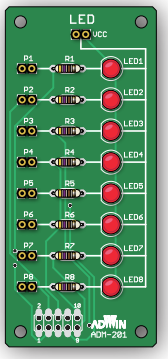
解答は，巻末の解答例集参照

配線 06-2

今回は配線 06-1 で示した配線のうち，PD0 - LED1 に加えて PD2- LED3，PC4 - LED5，PC6 - LED7 も利用します。



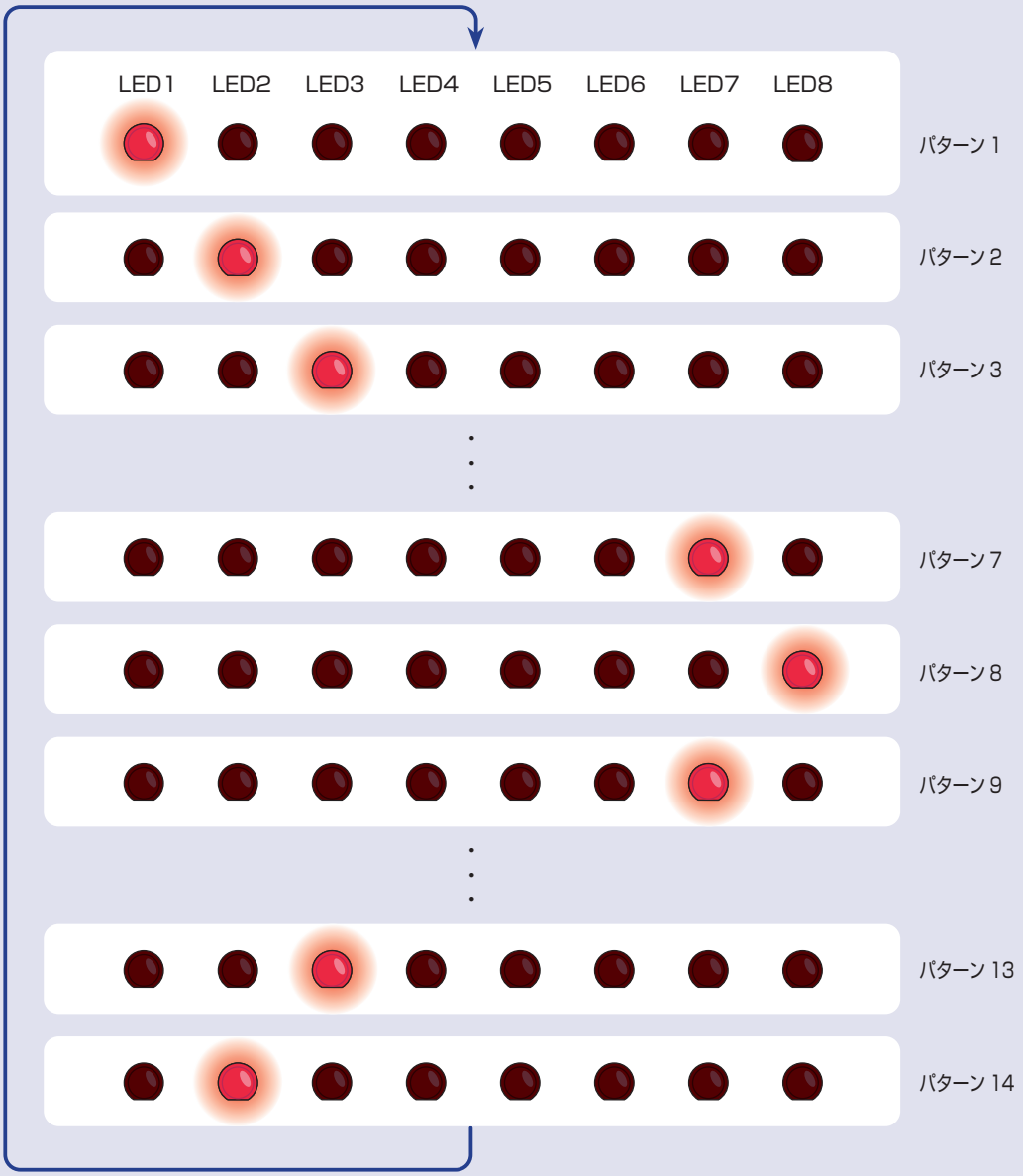
マイコンボード			LED ボード
3.03	PD0	⟷	P1 - LED1
3.04	PD1	⟷	P2 - LED2
3.05	PD2	⟷	P3 - LED3
3.06	PD3	⟷	P4 - LED4
4.04	PC4	⟷	P5 - LED5
4.05	PC5	⟷	P6 - LED6
4.06	PC6	⟷	P7 - LED7
4.07	PC7	⟷	P8 - LED8
1.01	+3.3V	⟷	VCC



ベースボードを用いず，ジャンパワイヤ（別売）でマイコンボードと LED ボードを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。LED1，LED3，LED5，LED8 は今回は使用しませんが，これらにも配線を行っておくと課題 06-3 にもそのまま対応できます。

課題 06-3

LED を下記のパターンで繰り返し点灯させましょう。点灯 LED が左右に移動する点灯パターンです。



解答は，巻末の解答例集参照

LED 点灯

配線 06-3

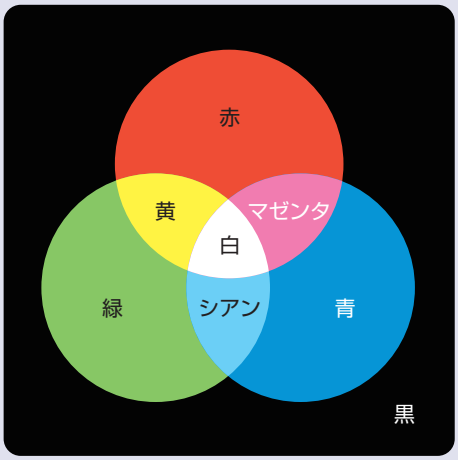
今回は配線 06-1 で示した配線のすべてを利用します。
ベースボードを用いず，ジャンパワイヤ（別売）でマイコンボードと LED ボードを接続することも可能です。
時間に余裕のある場合は，STEP 01 のピンアサインに注意しつつ，お好みのピンで課題を実現してみましょう。

課題 06-4

この課題は応用課題です。時間に余裕がある場合はチャレンジしてみてください。

マイコンボードには赤（R）・緑（G）・青（B）のユーザ LED が実装されており，赤は PF1，緑は PF3，青は PF2 で制御されます。この LED を使い，黒（消灯），赤，緑，青，黄，シアン，マゼンタ，白を順に繰り返し点灯させるようにしましょう。
LED ボードは今回は使用しません。マイコンボードのユーザ LED は H レベルで点灯，L レベルで消灯し，LED ボードとは逆なので注意しましょう。
課題 06-3 が理解できていれば簡単です。

ピン番号	GPIO	ボード上の機能等	MPU ピン
3.10	PF1	LED（赤）	29
4.01	PF2	LED（青）	30
4.02	PF3	LED（緑）	31



参考：光の３原色と組み合わせ

解答は，巻末の解答例集参照