C言語
設計実装エクササイズ［静的］

ビースラッシュ株式会社

コース概要と到達目標

コース 概要	ソフトウェア設計と実装の基本を習得する自習コースです。 動くコードから始めて、関数化／構造化／部品化の３つのプログラミングを行います。 プログラミング結果のソースコードを図面化して、設計意図を説明します。
特徴	コードの実装を行い、その図面を作り、解答図面と比較検討することで、実践力を高めます。 完全な自習コースです。隙間時間もしくは集中的に時間確保して実施いただけます。 １カ月の期間中は質問ができます。
前提条件	C言語の文法を学習したことがある方。 （文法が分からないときは書籍などで確認することができれば充分です）
到達目標	本講座では、次のことを目標としています。 １．設計の基本に従ってプログラミングできる　→ 箱と線と配置 ２．ソースコードから構造図を作成できる　→ 図面化 ３．良い設計とは何かを構造図で説明できる　→ 設計原則 ４．設計意図を他者に説明できる　→ 設計意図

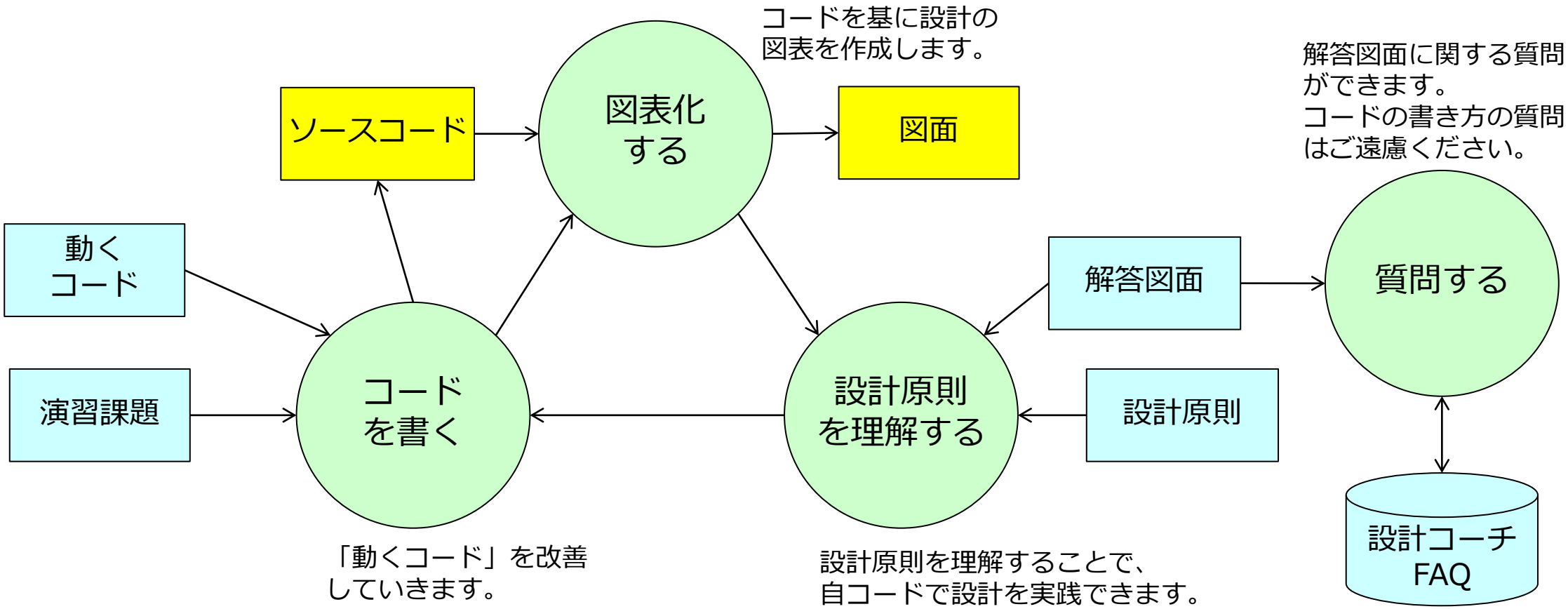
習得できる設計原則

- 設計原則を理解すれば、ソースコードの設計品質が判断できます
  - 生成AIコードも設計品質を高めることができます

ID	設計原則		要点
G1	関数化	関心事の分離	高凝集
G2		インタフェース定義	疎結合
G3		レイヤー化	アプリ層とドライバ層の分離、ミドル層の作り方
G4		IO分離	入力（センサー）と出力（アクチュエータ）の分離
G5	構造化	抽象データ型	データ構造を公開せず、手続きを定義して公開する
G6		BOSS-STS構造	構造設計の基本形
G7		求心遠心	物理と論理を往来するデータの流れ
G8	部品化	インタフェースと実装の分離	ヘッダファイルでインタフェース定義、実装ファイルは置換可能
G9		変数のカプセル化	変数をファイル内スコープにする
G10		単方向依存	上位が下位のサービスを使う

狙い：コードと設計原則を関連付けて理解

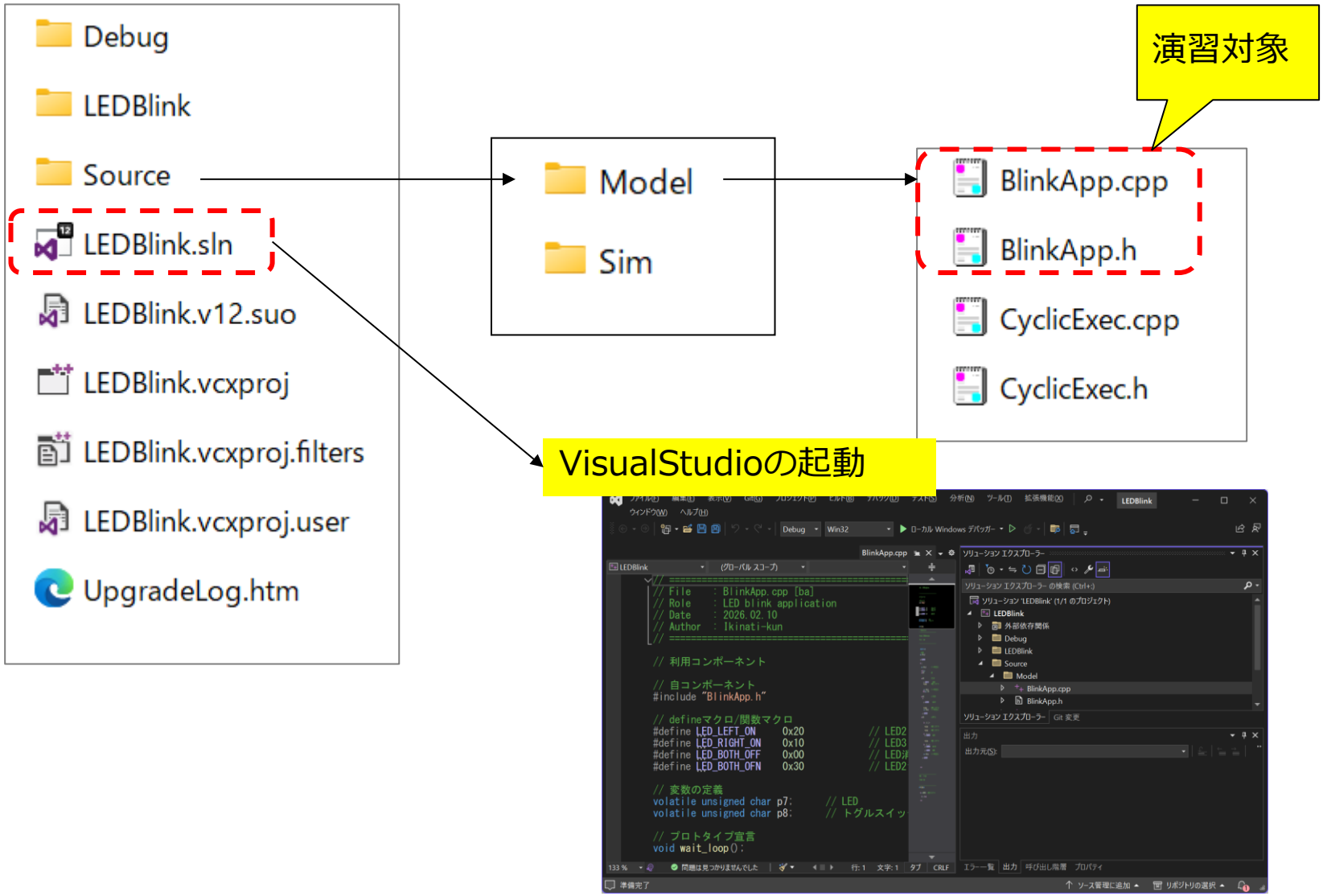
- コードを書いて、それを図面化することで「良い設計」を体得します
- 図表化することで開発現場での適用が容易になります
- 設計原則と照らし合わせることで、実践と理論がつながります



事前準備

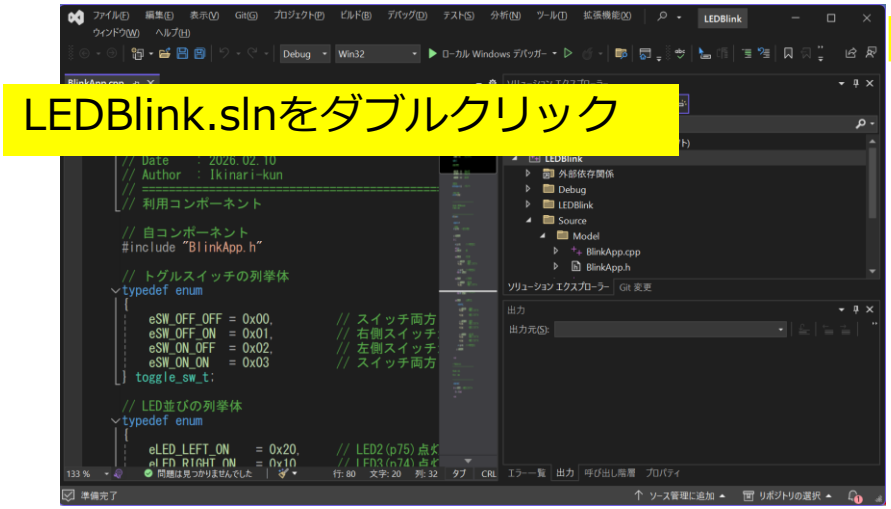
まずは動作確認をしてください

フォルダ構成



シミュレータ動作の確認

- ソリューションファイル(LEDBlink.sln)をダブルクリックして「ビルド」→「デバッグの開始」でトグルボタン操作→LEDが光ることを確認してください
  - VisualStudio2026で動作します



ここをクリックすると
見呉横のLEDが光ります

要求仕様と動くコード

要求仕様と動作環境（想定ハードウェアとPCシミュレーション）

要求仕様

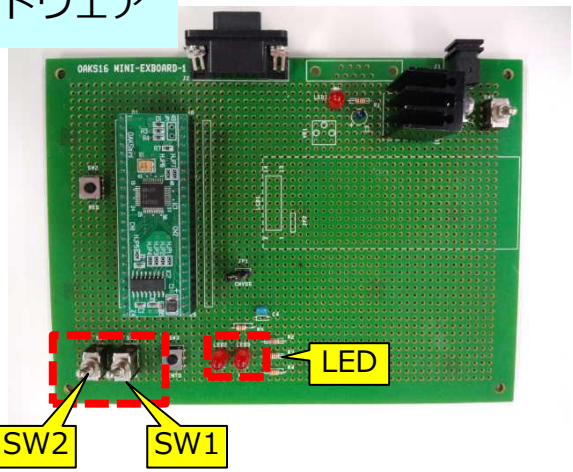
2つのトグルスイッチの状態に応じて、
2つのLEDの点滅パターンを変える

点滅仕様

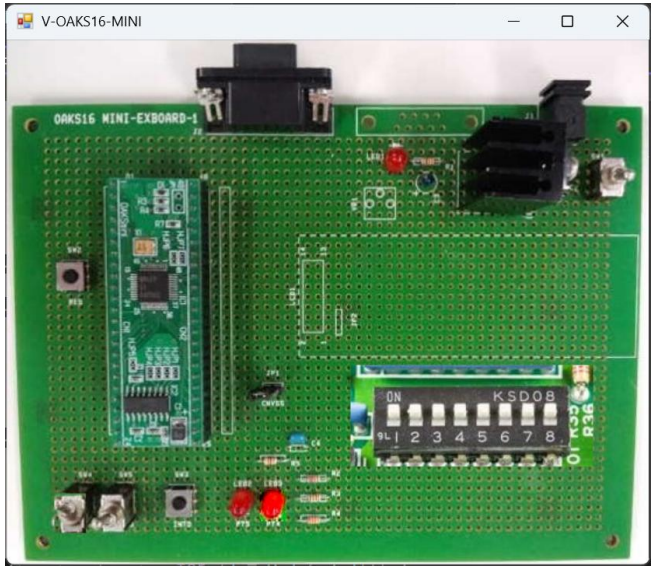
2つのトグルスイッチで点滅パターンを切り替える

SW2	SW1	点滅パターン
OFF	OFF	２つ同時に点滅
OFF	ON	交互に点灯
ON	OFF	2進数で0～３
ON	ON	三三七拍子のリズム

想定ハードウェア



PCシミュレータ

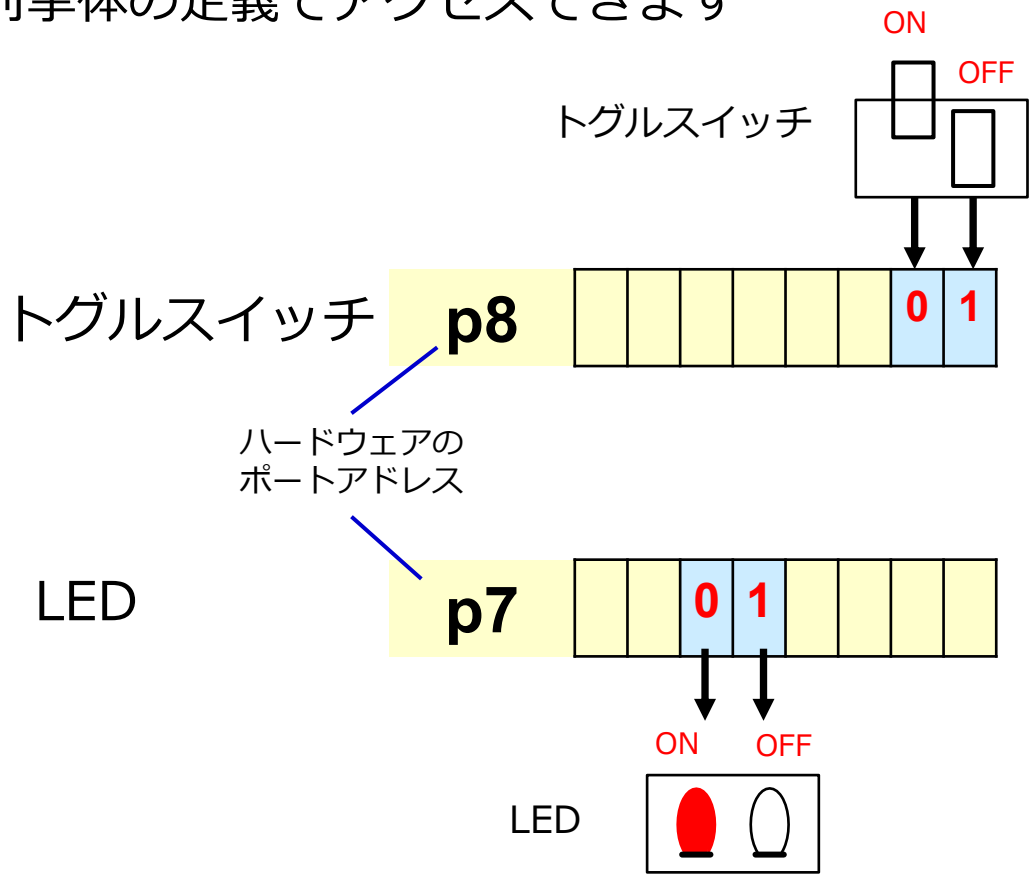


Visual Studioで動作します
こちらからダウンロードして動作確認疎お願いします
↓

https://www.bslash.co.jp/AtMuseum/faq_cstm/exercisesim/
atd20260227bs

想定ハードウェア：トグルスイッチとLED

- p8とp7がハードウェアポートです
  - 回路図は省略します
  - 列挙体の定義でアクセスできます



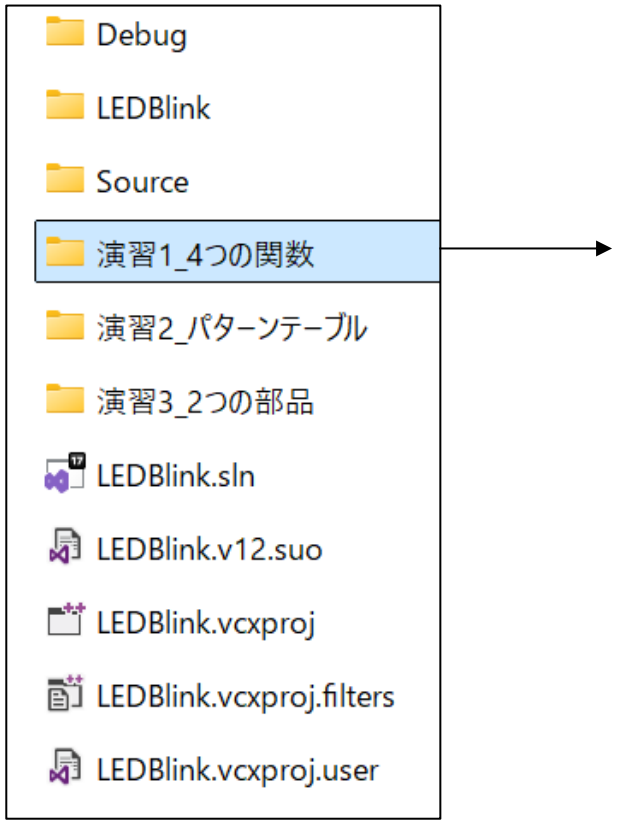

```
// トグルスイッチの列挙体
typedef enum
{
    eSW_OFF_OFF = 0x00,     // スイッチ両方ともオフ
    eSW_OFF_ON  = 0x01,     // 右側スイッチがオン
    eSW_ON_OFF  = 0x02,     // 左側スイッチがオン
    eSW_ON_ON   = 0x03      // スイッチ両方ともオン
} toggle_sw_t;/
}
// 変数の定義
volatile unsigned char p8;          // トグルスイッチ
```

```
// LED並びの列挙体
typedef enum
{
    eLED_LEFT_ON     = 0x20,    // LED2(p75)点灯
    eLED_RIGHT_ON    = 0x10,    // LED3(p74)点灯
    eLED_BOTH_OFF    = 0x00,    // LED消灯
    eLED_BOTH_ON     = 0x30,    // LED2つ点灯
} led_line_t;

// 変数の定義volatile unsigned char p7;          // LED
```

演習1：関数化

関数呼出しと図面化

- ４つの関数を使ってプログラミングしてください
- その結果を図面化してください



Funcs.c

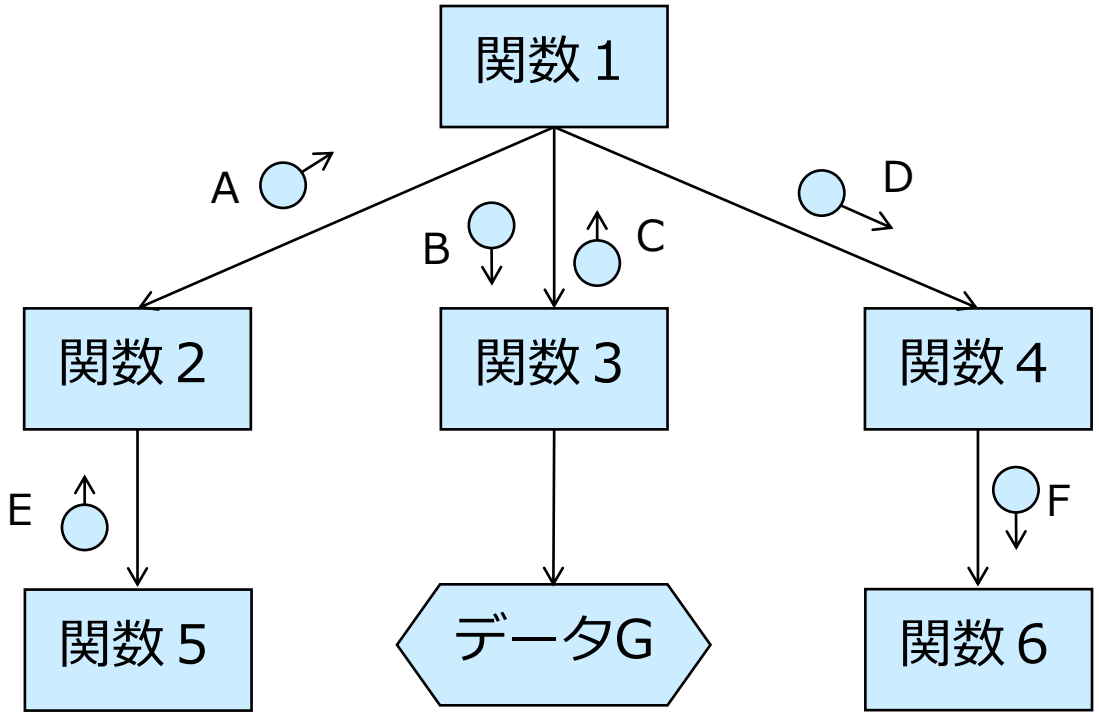
4つの関数

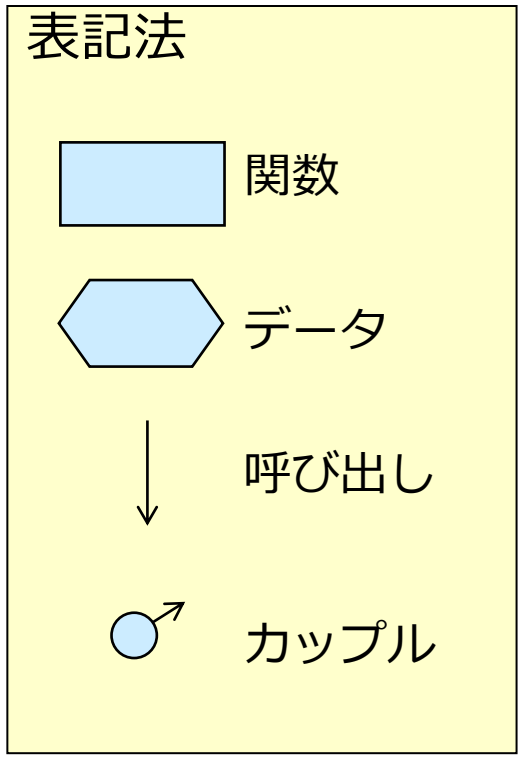
初期化関数とドライバ関数

```
// プロトタイプ宣言
static void init_sw(void);
static void init_led(void);
static toggle_sw_t gpio_sw_read(void);
static void gpio_led_write(led_line_t);
```

表記法 関数構造図（引数と戻り値）

- 関数の呼び出し構造の図解
- 関数間を流れるデータの表記



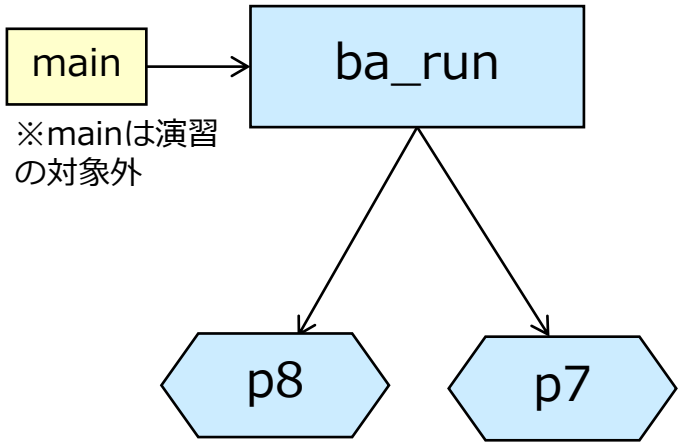


カップルは関数への引数と戻り値に相当します
上位から下位へ流れるカップルが引数
下位から上位へ流れるカップルが戻り値

起点コードと演習1の関数構造図

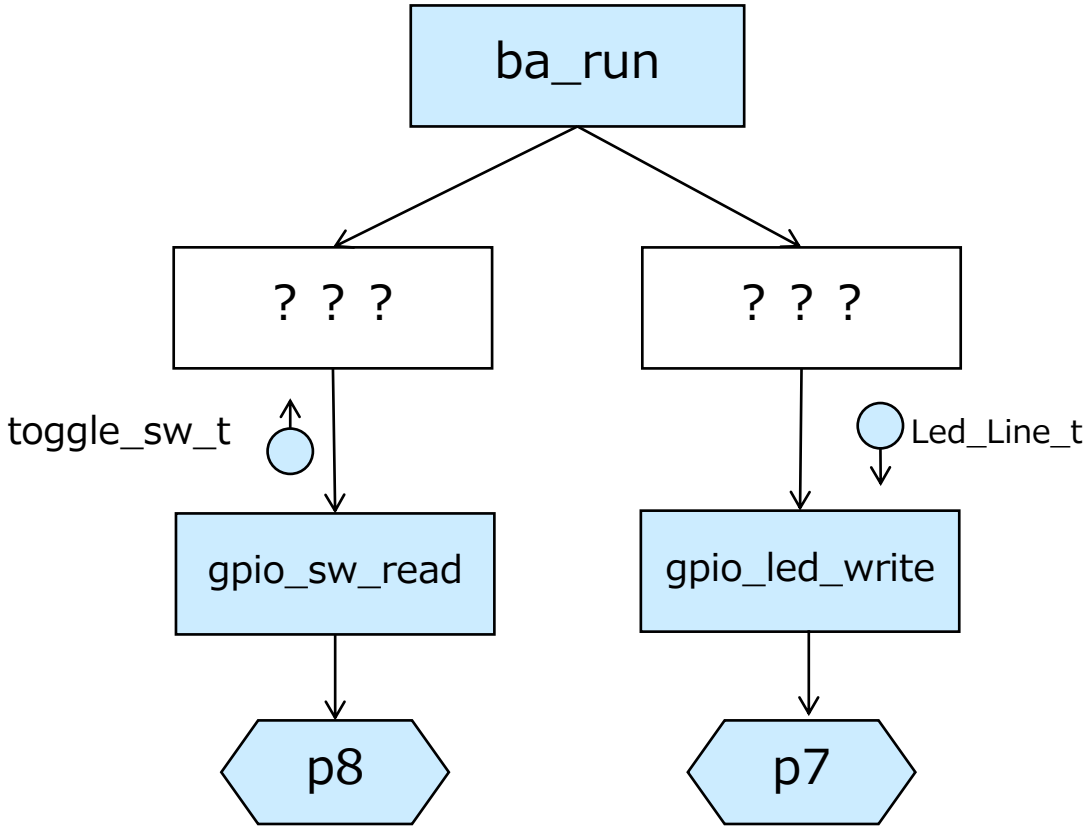
起点コード

関数呼出しの構造はなく、bs_run()から
直接p8とp7にアクセスしている
→「一筆書き」プログラムと呼ばれる



演習1

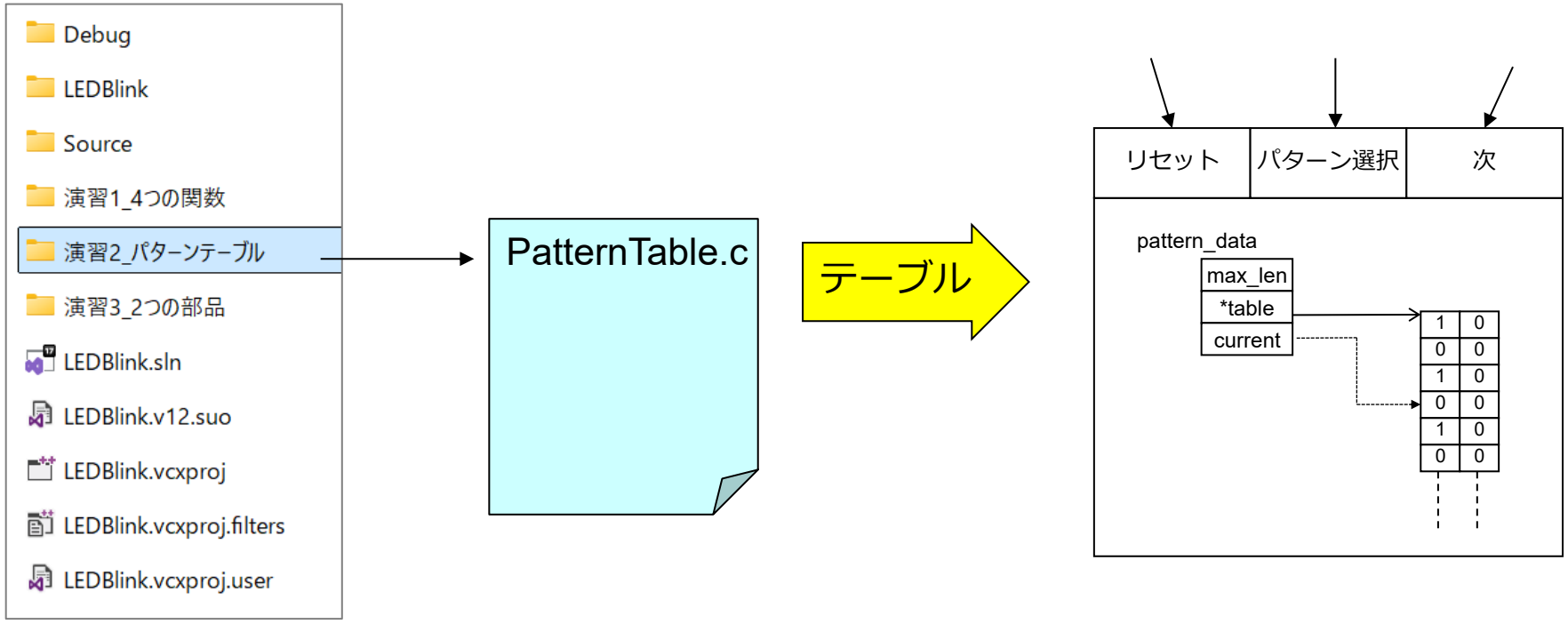
ba_run()内部を関数化して、関数の呼び出し構造を創ってください
その際に、bs_run()から直移設ドライバ関数路呼び出すのではなく、
中間に関数を挟んでください



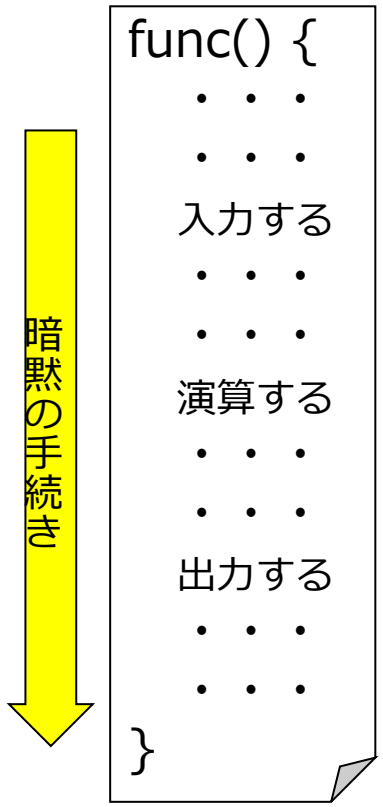
演習2：構造化

パターンテーブルを使って関数化

- 点滅パターンの切り替えをテーブルで実施
- 「リセット」と「パターン選択」と「次データの取り出し」の関数を作って、上位から呼び出してください
- その関数構造図を作成してください ⇒ BOSS-STS構造を目指す（次ページ）

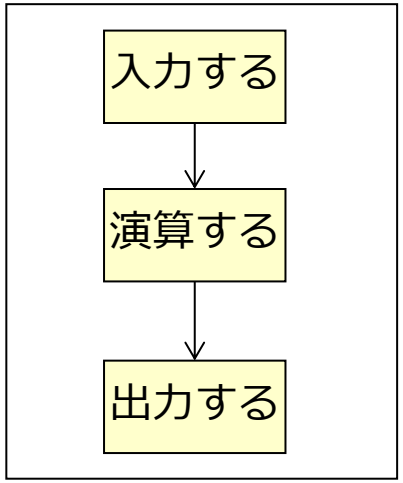


構造設計の基本：BOSS-STS構造

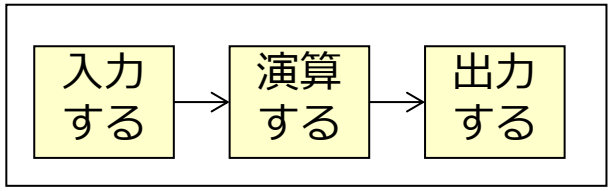


一筆書き
設計ではない

✕ フローチャート

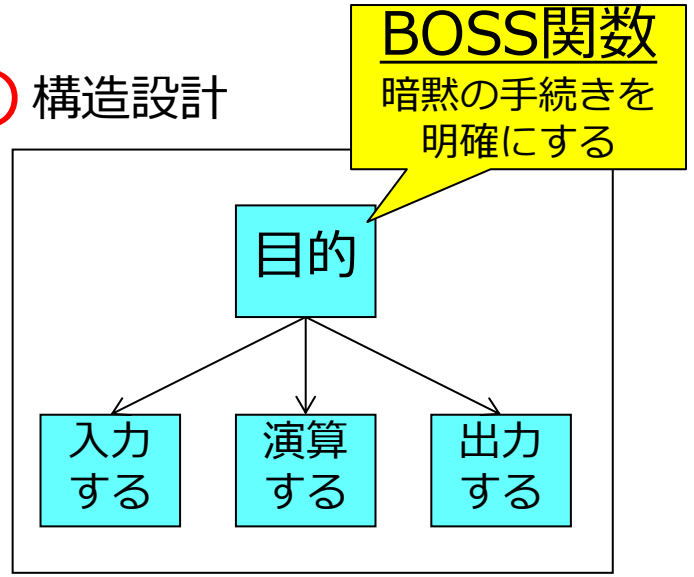


✕ コールグラフ



フローチャート／コールグラフ
モジュール化はできているが、
レベル化はできていない
⇒設計ではない

○ 構造設計

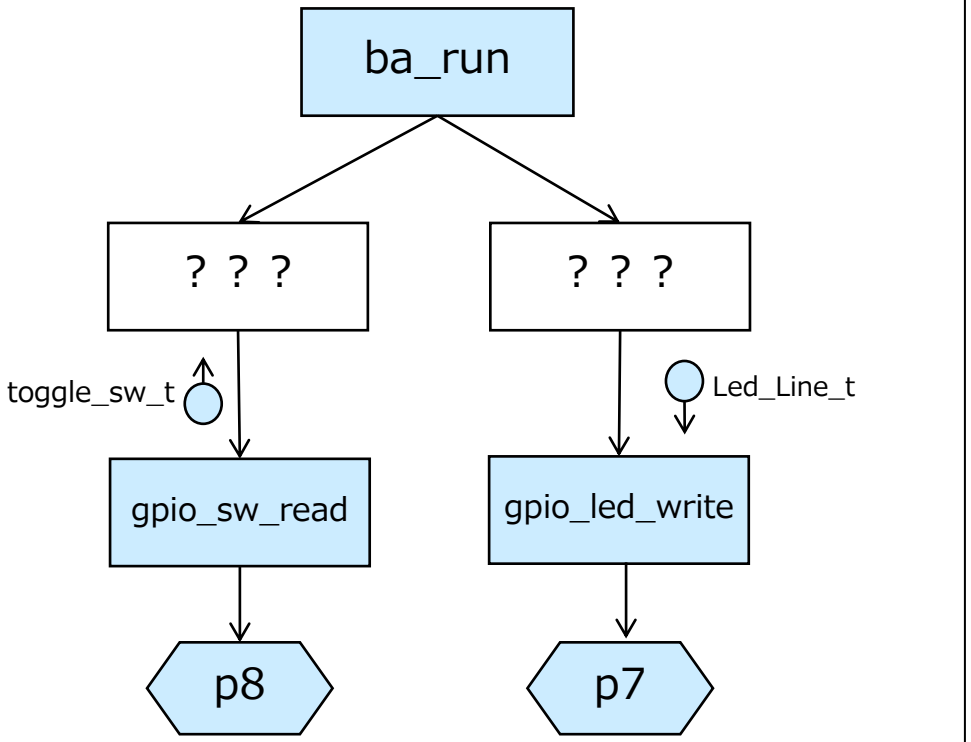


構造設計
箱：What名称
線：利用関係
配置：水平垂直分割
　水平：レイヤー化
　垂直：IO分離、STS分割

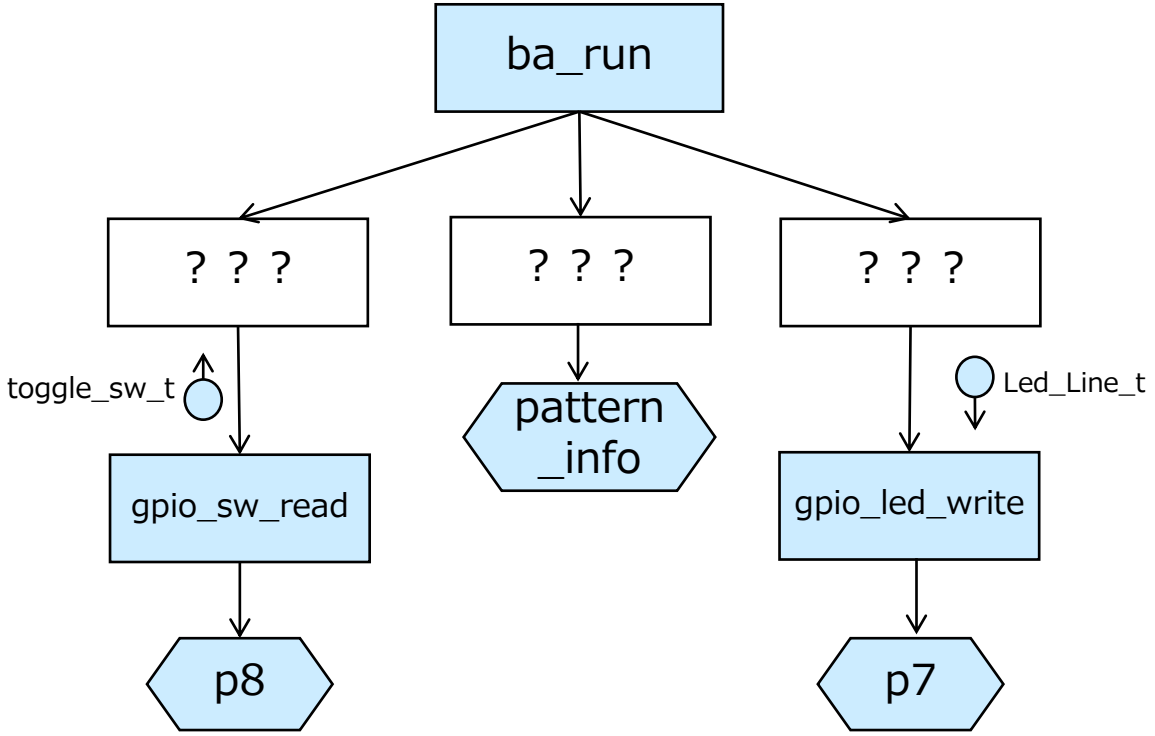
STS：Source-Transform-Sink
源泉 ― 変換 ― 吸収

演習1と演習2の関数構造図

演習1
入出力分離した構造図



演習2
BOSS-STS構造の構造図

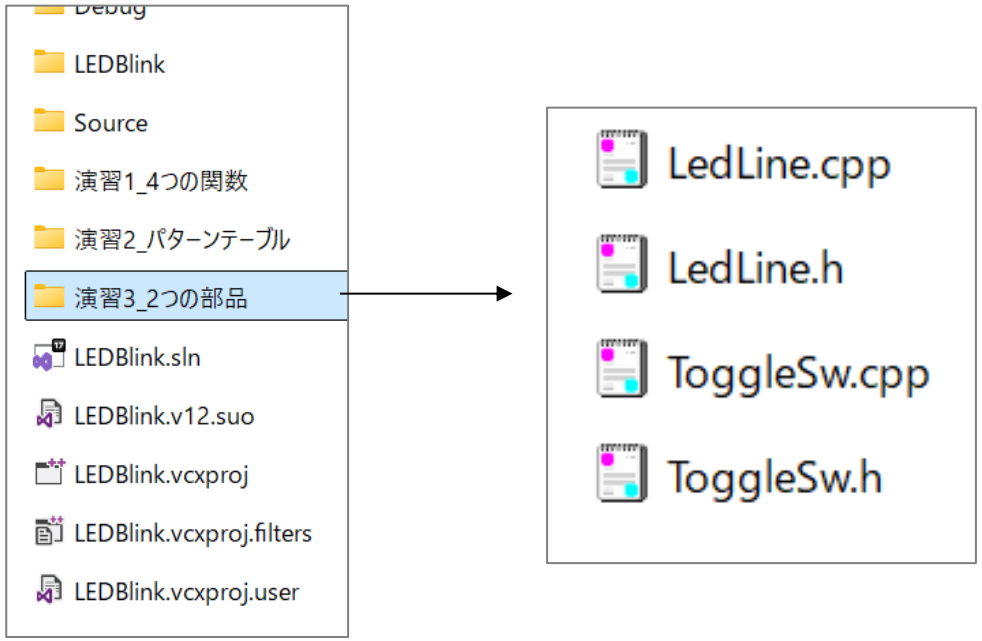


演習３：部品化

部品化

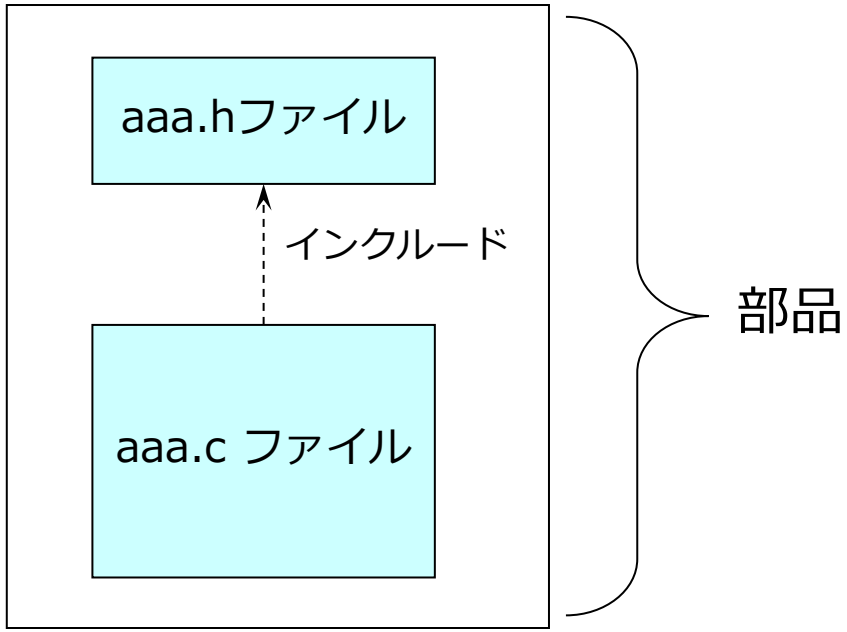
- ヘッダファイルと実装ファイルのペア（部品）を作ってください
- 部品間の呼び出し構造でファイル構造図を作成してください

- 2つの部品化例があります



部品化

- 部品とは、付け替え、組合せによってソフトウェアを動かすことが出来る管理単位
- 部品は、データと手続きを有する単位
  - C言語の場合、ヘッダファイル（拡張子がh）と実装ファイル（拡張子がc）のペア
  - オブジェクト指向言語の場合、クラス
  - 『最小の管理単位』となる
  - データをカプセル化
  - 公開／非公開を明示的に



部品の例（トグルスイッチ）

Toggle_sw.h

```
// ================================================
// File    : ToggleSw.h [ts]
// Role    : Toggle Switch Driver
// Date    : 2026.02.10
// Author  : Kouzou-san
// ================================================
#pragma once

// トグルスイッチの列挙体
typedef enum
{
    eSW_OFF_OFF = 0x00,         // スイッチ両方ともオフ
    eSW_OFF_ON = 0x01,          // 右側スイッチがオン
    eSW_ON_OFF = 0x02,          // 左側スイッチがオン
    eSW_ON_ON = 0x03            // スイッチ両方ともオン
} toggle_sw_t;

// 公開インタフェース定義
extern void         init_sw(void);
extern toggle_sw_t gpio_sw_read(void);
```

③外部へ公開するものは
ヘッダファイルでextern宣言

公開関数の引数や戻り値は
enum定義で公開

Toggle_sw.c

実装ファイルで利用ファイル
と自ファイルをinclude

公開関数

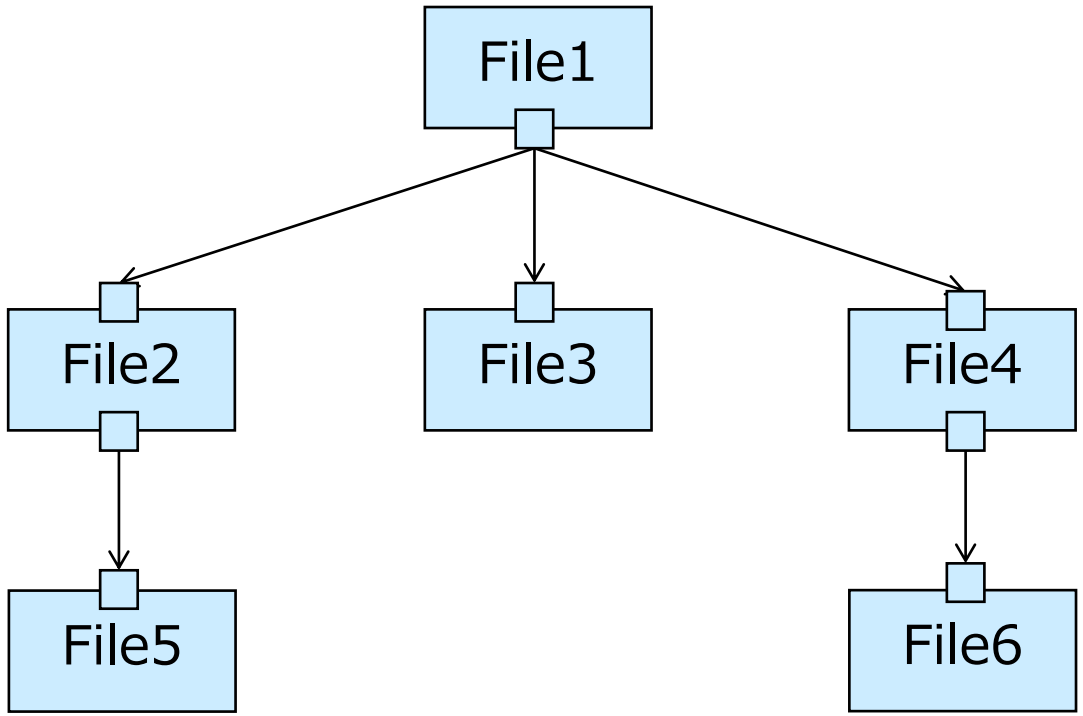
```
// ================================================
// File    : ToggleSw.cpp [ts]
// Role    : Toggle Switch Driver
// Date    : 2026.02.10
// Author  : Kouzou-san
// ================================================
// 利用コンポーネント

// 自コンポーネント
#include "ToggleSw.h"

// 変数の定義
static volatile unsigned char p8;     // トグルスイッチ

// プロトタイプ宣言
// --- functions (extern)
// ================================================
// Name        : init_sw
// Function    : トグルスイッチの初期化
// Parameters  : none
// Return      : none
// notes       :
// ================================================
void init_sw() {
    toggle_sw_t sw;

    // 一度読むことで初期化
    sw = gpio_sw_read();

    return;
}

// ================================================
// Name        : gpio_sw_read
// Function    : トグルスイッチを読む
// Parameters  : none
// Return      : トグルスイッチの値（2ビット）
// notes       :
// ================================================
toggle_sw_t gpio_sw_read() {

    toggle_sw_t sw;

    sw = (toggle_sw_t)(0x03 & p8);

    return sw;
}
```

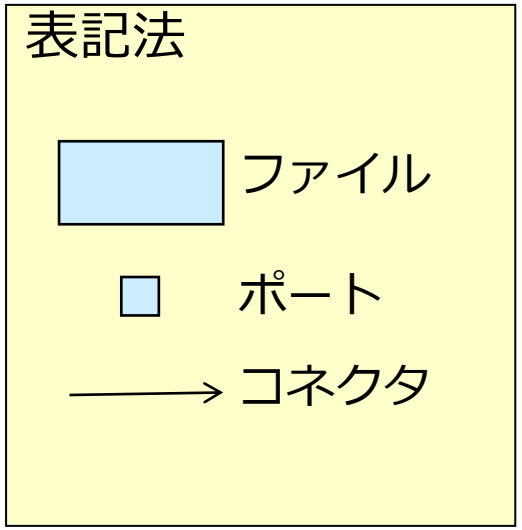
static宣言により変数をカプセル化
※今回はsim側で使っているので、staticはつけていません
※staticを付けるとリンクエラーになってしまいます

公開する初期化関数
※ファイルごとに初期化関数を作る

表記法 ファイル構造図

- ファイル間の呼び出し構造の図解
- 関数集合がインタフェース（ポート）になります





本資料の一部または全部を、ビースラッシュ株式会社の
許可なしに複写、複製、再配布することを禁じます。