

例題1 トグルスイッチと LED

- 4つのトグルスイッチの ON/OFF のとおりに 4つの LED を点灯する



全ての LED が OFF



全ての TSW が OFF

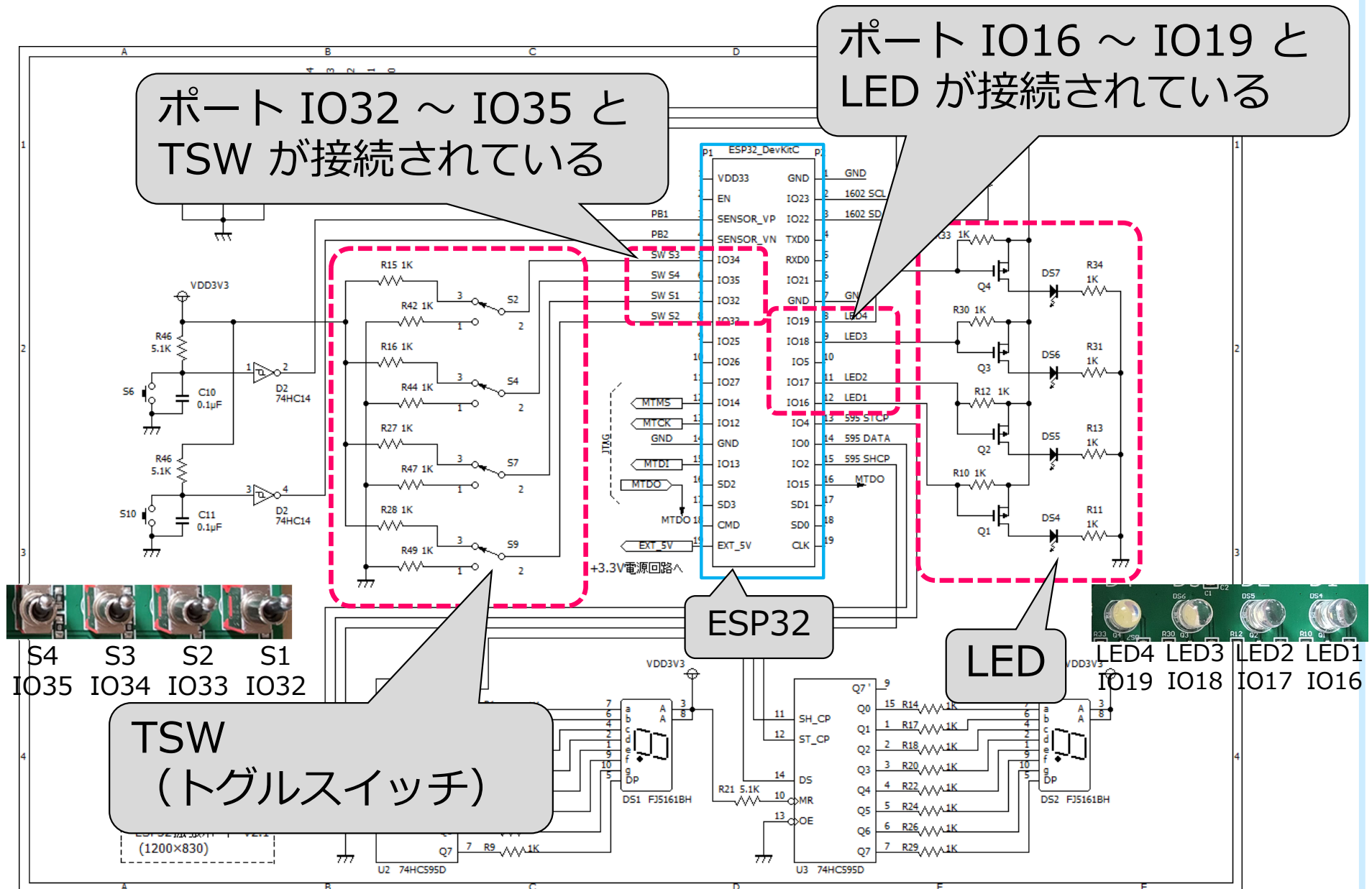


LED1とLED3 がON



TSW1とTSW3 がON

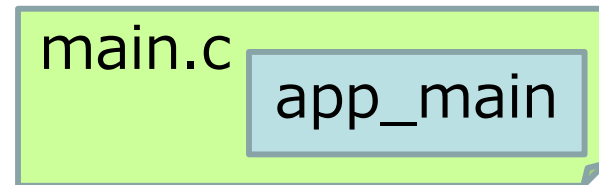
参考：スマートパネルの回路図



紹介するプログラム

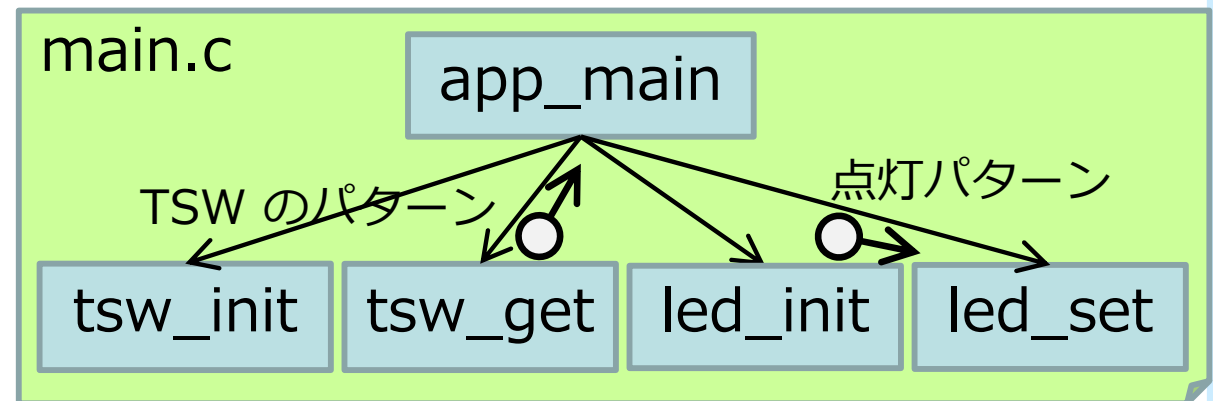
■ 例題1-1

- app_main 関数のみ



■ 例題1-2

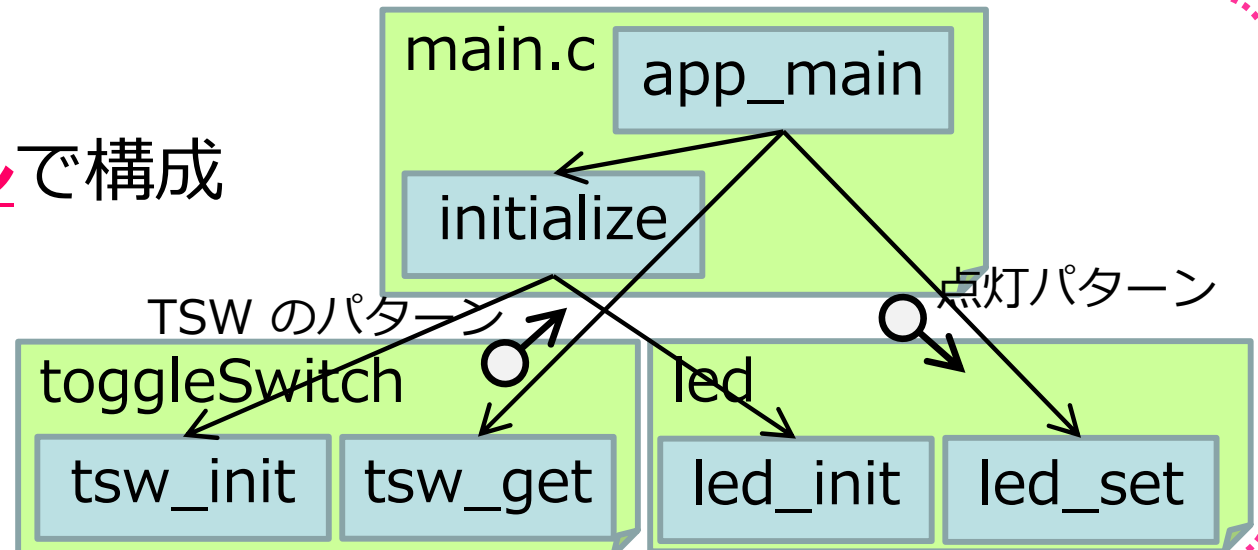
- 複数の関数で構成



■ 例題1-3

- 複数のファイルで構成

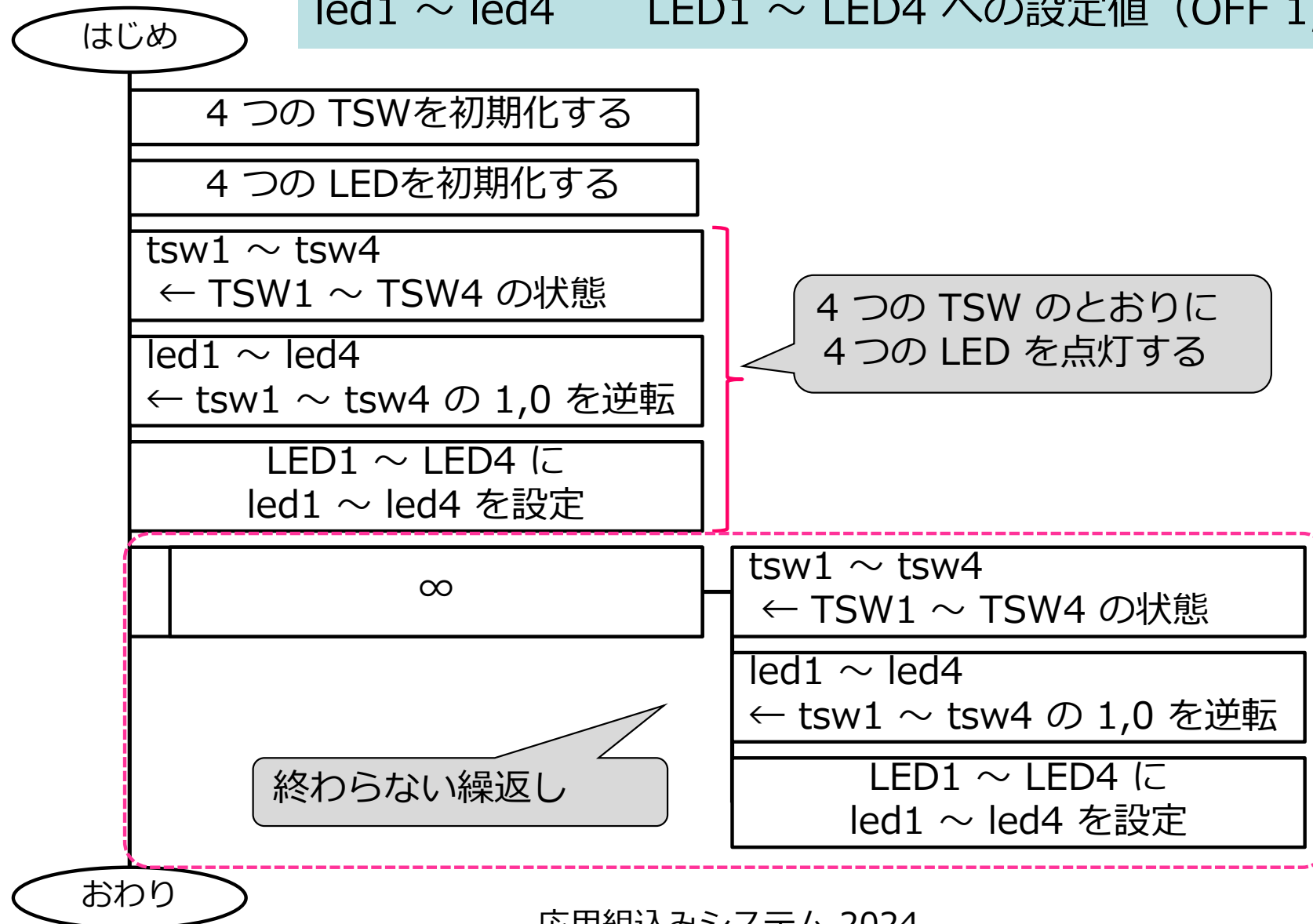
第2回以降は
例題1-3 の形を
使います



例題1-1 アルゴリズム

データ

tsw1 ~ tsw4 TSW1 ~ TSW4 の状態 (OFF 0, ON 1)
led1 ~ led4 LED1 ~ LED4 への設定値 (OFF 1, ON 1)



例題1-1 プログラム

app_main 関数

ファイルコメント
(ファイルの情報)

関数コメント
(関数の情報)

ヘッダファイルの
#include

変数の宣言

マクロの定義

TSW の初期化

プロトタイプ
(関数 sv_init)

LED の初期化

7 セグメント LED を初期化
する関数 sv_init の定義

関数 sv_init の呼出し

tsw1 ~ tsw4
← TSW1 ~ TSW4 の状態

led1 ~ led4
← tsw1 ~ tsw4 の 1,0 を逆転

LED1 ~ LED4 に
led1 ~ led4 を設定

終わらない繰返し

```
// File : main.c
// Role : Sample01-1 toggle switch and LED
// Date : 2024.04.11
// Author : Osaka Sangyo University
// =====
// --- Header Files (system)
#include "sdkconfig.h"
#include "hal/gpio_types.h"
#include "driver/gpio.h"

// --- macros
#define TSW1 GPIO_NUM_32
#define TSW2 GPIO_NUM_33
#define TSW3 GPIO_NUM_34
#define TSW4 GPIO_NUM_35
#define LED1 GPIO_NUM_16
#define LED2 GPIO_NUM_17
#define LED3 GPIO_NUM_18
#define LED4 GPIO_NUM_19

// --- prototypes (static)
static void sv_init(void);
```

```
-----
// Name : sv_init
// Function : initialize seven segment LED
// Parameters : none
// Return : none
// notes : call before closed loop
// =====
static void sv_init(void)
{
    unsigned char i;
    unsigned char j;

    esp_rom_gpio_pad_select_gpio(GPIO_NUM_0);
    ESP_ERROR_CHECK(gpio_set_level(GPIO_NUM_0, 1U));
    ESP_ERROR_CHECK(gpio_set_direction(GPIO_NUM_0, GPIO_MODE_OUTPUT));
    esp_rom_gpio_pad_select_gpio(GPIO_NUM_2);
    ESP_ERROR_CHECK(gpio_set_level(GPIO_NUM_2, 0U));
    ESP_ERROR_CHECK(gpio_set_direction(GPIO_NUM_2, GPIO_MODE_OUTPUT));
    esp_rom_gpio_pad_select_gpio(GPIO_NUM_4);
    ESP_ERROR_CHECK(gpio_set_level(GPIO_NUM_4, 0U));
    ESP_ERROR_CHECK(gpio_set_direction(GPIO_NUM_4, GPIO_MODE_OUTPUT));
    // seven segments all off
    for (i = 0; i < 2U; ++i) {
        for (j = 0; j < (8U + 1U); ++j) {
            ESP_ERROR_CHECK(gpio_set_level(GPIO_NUM_0, 1U));
            ESP_ERROR_CHECK(gpio_set_level(GPIO_NUM_2, 0U));
            ESP_ERROR_CHECK(gpio_set_level(GPIO_NUM_4, 1U));
        }
        ESP_ERROR_CHECK(gpio_set_level(GPIO_NUM_4, 0U));
        ESP_ERROR_CHECK(gpio_set_level(GPIO_NUM_4, 1U));
    }
    return;
}
```

```
--- app_main function
// Name : app_main
// Function : run the system
// Parameters : none
// Return : none
// notes : called from Main Task
=====
void app_main(void)
{
    unsigned char tsw1;
    unsigned char tsw2;
    unsigned char tsw3;
    unsigned char tsw4;
    unsigned char led1;
    unsigned char led2;
    unsigned char led3;
    unsigned char led4;

    // initialize toggle switch
    esp_rom_gpio_pad_select_gpio(TSW1);
    ESP_ERROR_CHECK(gpio_set_direction(TSW1, GPIO_MODE_INPUT));
    ESP_ERROR_CHECK(gpio_pull_up_en(TSW1));
    esp_rom_gpio_pad_select_gpio(TSW2);
    ESP_ERROR_CHECK(gpio_set_direction(TSW2, GPIO_MODE_INPUT));
    ESP_ERROR_CHECK(gpio_pull_up_en(TSW2));
    esp_rom_gpio_pad_select_gpio(TSW3);
    ESP_ERROR_CHECK(gpio_set_direction(TSW3, GPIO_MODE_INPUT));
    ESP_ERROR_CHECK(gpio_pull_up_en(TSW3));
    esp_rom_gpio_pad_select_gpio(TSW4);
    ESP_ERROR_CHECK(gpio_set_direction(TSW4, GPIO_MODE_INPUT));
    ESP_ERROR_CHECK(gpio_pull_up_en(TSW4));

    // initialize led
    esp_rom_gpio_pad_select_gpio(LED1);
    ESP_ERROR_CHECK(gpio_set_level(LED1, 1U));
    ESP_ERROR_CHECK(gpio_set_direction(LED1, GPIO_MODE_OUTPUT));
    esp_rom_gpio_pad_select_gpio(LED2);
    ESP_ERROR_CHECK(gpio_set_level(LED2, 1U));
    ESP_ERROR_CHECK(gpio_set_direction(LED2, GPIO_MODE_OUTPUT));
    esp_rom_gpio_pad_select_gpio(LED3);
    ESP_ERROR_CHECK(gpio_set_level(LED3, 1U));
    ESP_ERROR_CHECK(gpio_set_direction(LED3, GPIO_MODE_OUTPUT));
    esp_rom_gpio_pad_select_gpio(LED4);
    ESP_ERROR_CHECK(gpio_set_level(LED4, 1U));
    ESP_ERROR_CHECK(gpio_set_direction(LED4, GPIO_MODE_OUTPUT));

    // initialize seven segment LED
    sv_init();

    // read toggle switch
    tsw1 = (unsigned char)gpio_get_level(TSW1);
    tsw2 = (unsigned char)gpio_get_level(TSW2);
    tsw3 = (unsigned char)gpio_get_level(TSW3);
    tsw4 = (unsigned char)gpio_get_level(TSW4);

    // set led
    led1 = tsw1 ^ BIT0;
    led2 = tsw2 ^ BIT0;
    led3 = tsw3 ^ BIT0;
    led4 = tsw4 ^ BIT0;

    ESP_ERROR_CHECK(gpio_set_level(LED1, led1));
    ESP_ERROR_CHECK(gpio_set_level(LED2, led2));
    ESP_ERROR_CHECK(gpio_set_level(LED3, led3));
    ESP_ERROR_CHECK(gpio_set_level(LED4, led4));

    for (;;) { // closed loop
        // read toggle switch
        tsw1 = (unsigned char)gpio_get_level(TSW1);
        tsw2 = (unsigned char)gpio_get_level(TSW2);
        tsw3 = (unsigned char)gpio_get_level(TSW3);
        tsw4 = (unsigned char)gpio_get_level(TSW4);
        // TSW OFF(0U)/ON(1U) -> LED OFF(1U)/ON(0U)
        led1 = tsw1 ^ BIT0;
        led2 = tsw2 ^ BIT0;
        led3 = tsw3 ^ BIT0;
        led4 = tsw4 ^ BIT0;
        // set LED
        ESP_ERROR_CHECK(gpio_set_level(LED1, led1));
        ESP_ERROR_CHECK(gpio_set_level(LED2, led2));
        ESP_ERROR_CHECK(gpio_set_level(LED3, led3));
        ESP_ERROR_CHECK(gpio_set_level(LED4, led4));
    }
}
```

例題1-1 マクロなど

■ 定数に「意味を表す名前」をつける

```
// =====  
// File      : main.c  
// Role      : Sample01-1 toggle switch and LED  
// Date      : 2024.04.11  
// Author    : Osaka Sangyo University  
// =====  
// --- Header files (system)  
#include "sdkconfig.h"  
#include "hal/gpio_types.h"  
#include "driver/gpio.h"  
  
// --- macros  
#define TSW1    GPIO_NUM_32  
#define TSW2    GPIO_NUM_33  
#define TSW3    GPIO_NUM_34  
#define TSW4    GPIO_NUM_35  
#define LED1    GPIO_NUM_16  
#define LED2    GPIO_NUM_17  
#define LED3    GPIO_NUM_18  
#define LED4    GPIO_NUM_19
```

TSW1 ~ TSW4 の
ポートを指定するマクロ

LED1 ~ LED4 の
ポートを指定するマクロ

例題1-1 app_main 関数

■ 変数の宣言

```
// --- app_main function
// =====
// Name      : app_main
// Function   : run the system
// Parameters : none
// Return     : none
// notes      : called from Main Task
// =====
void app_main(void)
{
    unsigned char tsw1;
    unsigned char tsw2;
    unsigned char tsw3;
    unsigned char tsw4;
    unsigned char led1;
    unsigned char led2;
    unsigned char led3;
    unsigned char led4;
    // 次のページ以降へ
}
```

トグルスイッチの状態を
代入する変数の宣言

LED に設定する値を
代入する変数の宣言

例題1-1 周辺機器の初期化

```
void app_main(void)
```

```
{
```

```
// 前ページより
```

```
// initialize toggle switch
```

```
esp_rom_gpio_pad_select_gpio(TSW1);
```

```
ESP_ERROR_CHECK(gpio_set_direction(TSW1, GPIO_MODE_INPUT));
```

```
ESP_ERROR_CHECK(gpio_pullup_en(TSW1));
```

```
// TSW2 ~ TSW4 の初期化省略
```

```
// initialize LED
```

```
esp_rom_gpio_pad_select_gpio(LED1);
```

```
ESP_ERROR_CHECK(gpio_set_level(LED1, 1U));
```

```
ESP_ERROR_CHECK(gpio_set_direction(LED1, GPIO_MODE_OUTPUT));
```

```
// LED2 ~ LED4 の初期化省略
```

```
// initialize seven segment LED
```

```
sv_init();
```

```
// 次のページ以降へ
```

```
}
```

TSW1 を初期化する

LED1 を初期化する

関数 sv_init を呼出して
7 セグメント LED を初期化する
(点灯してしまうので)

例題1-1 TSW を読み LED に設定

```
void app_main(void)
{
    // 前ページより
    // read toggle switch
    tsw1 = (unsigned char)gpio_get_level(TSW1);
    tsw2 = (unsigned char)gpio_get_level(TSW2);
    tsw3 = (unsigned char)gpio_get_level(TSW3);
    tsw4 = (unsigned char)gpio_get_level(TSW4);
    // TSW OFF (0U) / ON (1U) -> LED OFF (1U) / ON (0U)
    led1 = tsw1 ^ BIT0;
    led2 = tsw2 ^ BIT0;
    led3 = tsw3 ^ BIT0;
    led4 = tsw4 ^ BIT0;
    // set LED
    ESP_ERROR_CHECK(gpio_set_level(LED1, led1));
    ESP_ERROR_CHECK(gpio_set_level(LED2, led2));
    ESP_ERROR_CHECK(gpio_set_level(LED3, led3));
    ESP_ERROR_CHECK(gpio_set_level(LED4, led4));
    // 次ページ以降へ
}
```

変数 tsw1 ~ tsw4 に
TSW1 ~ TSW4 の状態を代入

変数 tsw1 ~ tsw4 の 1,0 を
逆転して変数 led1 ~ led4 に代入

LED1 ~ LED4 に
led1 ~ led4 を設定

例題1-1 終わらない繰返し

```
void app_main(void)
{
    // 前ページより
    for (;;) { // closed loop
        // read toggle switch
        tsw1 = (unsigned char)gpio_get_level(TSW1);
        tsw2 = (unsigned char)gpio_get_level(TSW2);
        tsw3 = (unsigned char)gpio_get_level(TSW3);
        tsw4 = (unsigned char)gpio_get_level(TSW4);
        // TSW OFF(0U)/ON(1U) -> LED OFF(1U)/ON(0U)
        led1 = tsw1 ^ BIT0;
        led2 = tsw2 ^ BIT0;
        led3 = tsw3 ^ BIT0;
        led4 = tsw4 ^ BIT0;
        // set LED
        ESP_ERROR_CHECK(gpio_set_level(LED1, led1));
        ESP_ERROR_CHECK(gpio_set_level(LED2, led2));
        ESP_ERROR_CHECK(gpio_set_level(LED3, led3));
        ESP_ERROR_CHECK(gpio_set_level(LED4, led4));
    }
}
```

終わらない繰返し

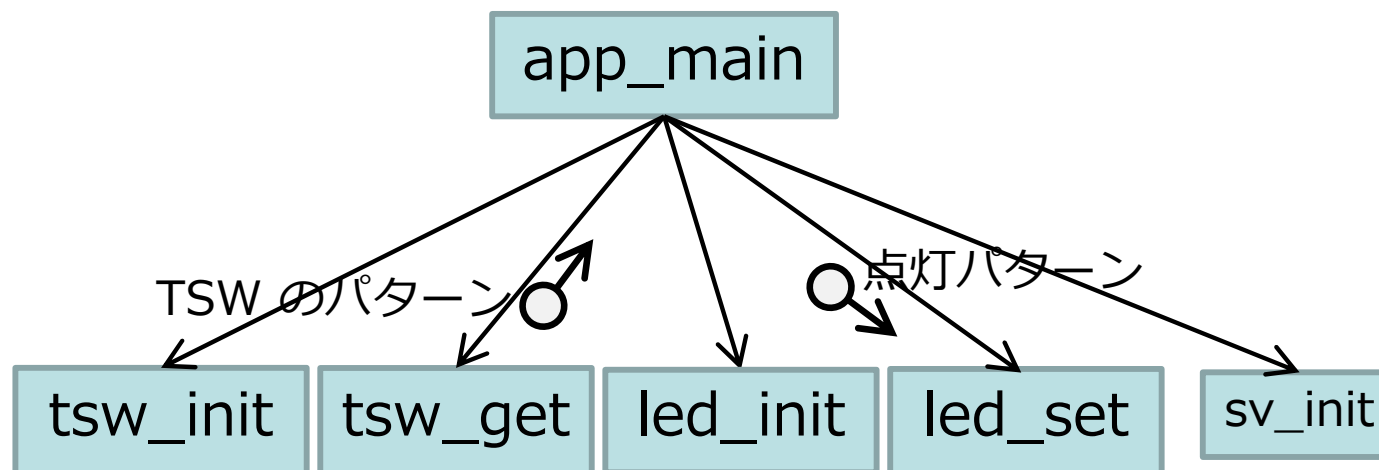
TSW を読み LED に設定

例題1-1 関数 sv_init の定義

```
// =====  
// Name      : sv_init  
// Function   : initialize seven segment LED  
// Parameters : none  
// Return    : none  
// notes     : call before closed loop  
// =====  
static void sv_init(void)  
{  
    unsigned char  i;  
    unsigned char  j;  
  
    esp_rom_gpio_pad_select_gpio(GPIO_NUM_0);  
    ESP_ERROR_CHECK(gpio_set_level(GPIO_NUM_0, 1U));  
    ESP_ERROR_CHECK(gpio_set_direction(GPIO_NUM_0, GPIO_MODE_OUTPUT));  
    esp_rom_gpio_pad_select_gpio(GPIO_NUM_2);  
    ESP_ERROR_CHECK(gpio_set_level(GPIO_NUM_2, 0U));  
    ESP_ERROR_CHECK(gpio_set_direction(GPIO_NUM_2, GPIO_MODE_OUTPUT));  
    esp_rom_gpio_pad_select_gpio(GPIO_NUM_4);  
    ESP_ERROR_CHECK(gpio_set_level(GPIO_NUM_4, 0U));  
    ESP_ERROR_CHECK(gpio_set_direction(GPIO_NUM_4, GPIO_MODE_OUTPUT));  
    // seven segments all off  
    for (i = 0; i < 2U; ++i) {  
        for (j = 0; j < (8U + 1U); ++j) {  
            ESP_ERROR_CHECK(gpio_set_level(GPIO_NUM_0, 1U));  
            ESP_ERROR_CHECK(gpio_set_level(GPIO_NUM_2, 0U));  
            ESP_ERROR_CHECK(gpio_set_level(GPIO_NUM_2, 1U));  
        }  
        ESP_ERROR_CHECK(gpio_set_level(GPIO_NUM_4, 0U));  
        ESP_ERROR_CHECK(gpio_set_level(GPIO_NUM_4, 1U));  
    }  
  
    return;  
}
```

7セグメントLEDの初期化

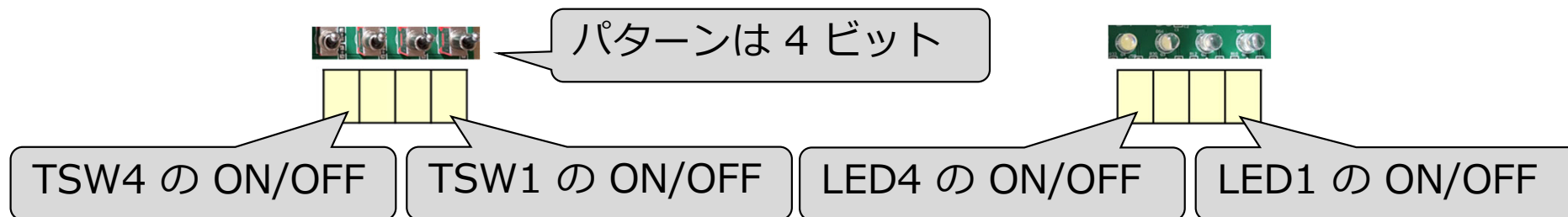
例題1-2 構造図



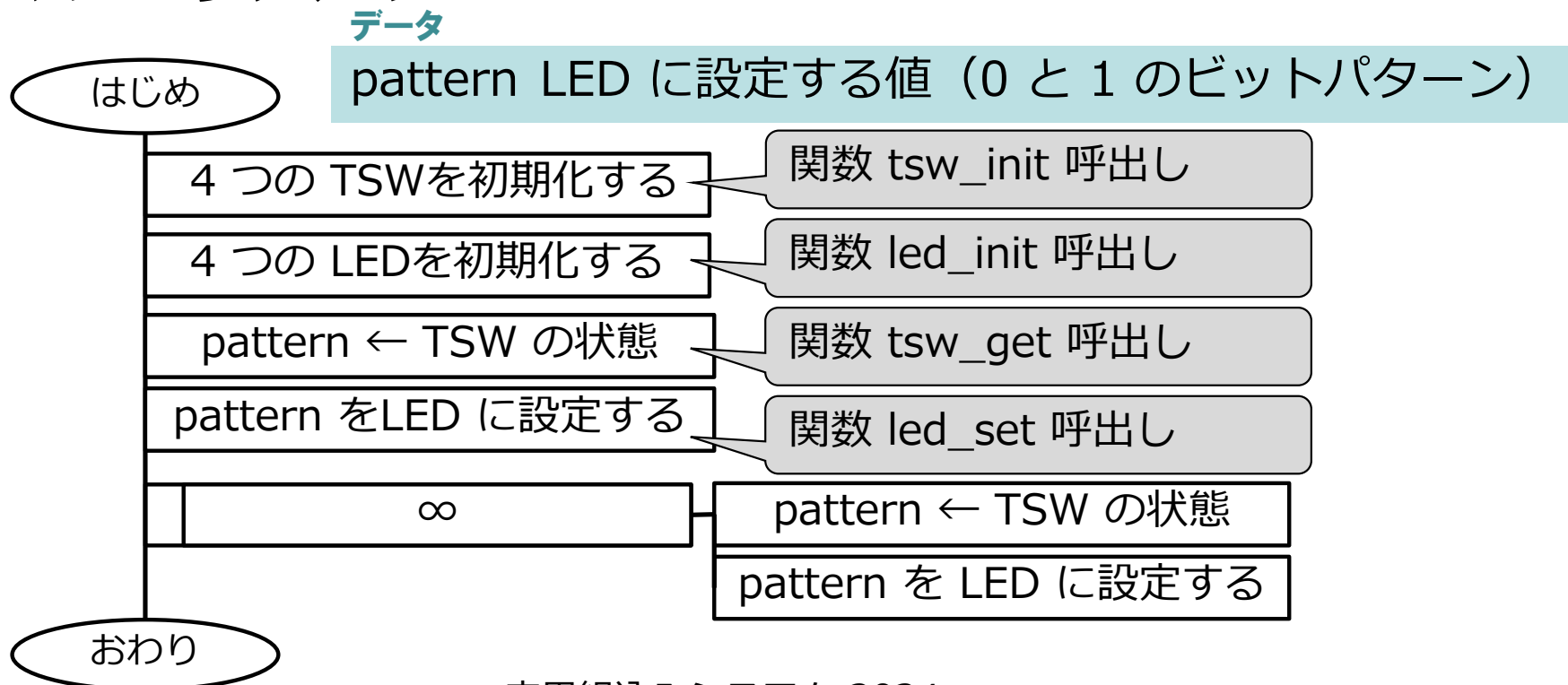
入出力装置	関数	機能	入力（引数）	出力（返却値）
TSW トグルスイッチ	tsw_init	TSW を初期化する	なし	なし
	tsw_get	TSW を読む	なし	TSW のパターン
LED	led_init	LED を初期化する	なし	なし
	led_set	LED に設定する	点灯パターン	なし

例題1-2 アルゴリズム

■ TSW,LED とビットパターン



■ アルゴリズム



例題1-2 プログラム

```
// =====  
// File   : main.c  
// Role   : Sample01-2 toggle switch and LED (functions)  
// Date   : 2024.04.11  
// Author : Osaka Sangyo University  
// =====  
// --- Header files (system)  
#include "sdkconfig.h"  
#include "hal/gpio_types.h"  
#include "driver/gpio.h"  
  
// --- macros  
#define TSM1 GPIO_NUM_32  
#define TSM2 GPIO_NUM_33  
#define TSM3 GPIO_NUM_34  
#define TSM4 GPIO_NUM_35  
#define LED1 GPIO_NUM_16  
#define LED2 GPIO_NUM_17  
#define LED3 GPIO_NUM_18  
#define LED4 GPIO_NUM_19
```

プロトタイプ

```
// --- prototype (static)  
static void tsw_init(void);  
static unsigned char tsw_get(void);  
static void led_init(void);  
static void led_set(unsigned char led);  
static void sv_init(void);  
  
// --- app_main (static)  
// Name : app_main  
// Function : run the system  
// Parameters : none  
// Return : none  
// Notes : called from Main Task  
// =====  
void app_main(void)  
{  
    unsigned char pattern;  
  
    // initialize toggle switch  
    tsw_init();  
    // initialize LED  
    led_init();  
  
    // initialize seven segment LED  
    sv_init();  
  
    // read toggle switch  
    pattern = tsw_get();  
    // set LED  
    led_set(pattern);  
    for (;;) { // closed loop  
        // read toggle switch  
        pattern = tsw_get();  
        // set LED  
        led_set(pattern);  
    }  
}
```

app_main 関数

```
// --- functions (static)  
// Name : tsw_init  
// Function : initialize toggle switch  
// Parameters : none  
// Return : none  
// Notes : call before closed loop  
// =====  
static void tsw_init(void)  
{  
    esp_rom_gpio_pad_select_gpio(TSM1);  
    ESP_ERROR_CHECK(gpio_set_direction(TSM1, GPIO_MODE_INPUT));  
    ESP_ERROR_CHECK(gpio_pull_up_en(TSM1));  
    esp_rom_gpio_pad_select_gpio(TSM2);  
    ESP_ERROR_CHECK(gpio_set_direction(TSM2, GPIO_MODE_INPUT));  
    ESP_ERROR_CHECK(gpio_pull_up_en(TSM2));  
    esp_rom_gpio_pad_select_gpio(TSM3);  
    ESP_ERROR_CHECK(gpio_set_direction(TSM3, GPIO_MODE_INPUT));  
    ESP_ERROR_CHECK(gpio_pull_up_en(TSM3));  
    esp_rom_gpio_pad_select_gpio(TSM4);  
    ESP_ERROR_CHECK(gpio_set_direction(TSM4, GPIO_MODE_INPUT));  
    ESP_ERROR_CHECK(gpio_pull_up_en(TSM4));  
    return;  
}
```

関数 tsw_init
の定義

```
// Name : tsw_get  
// Function : read toggle switch  
// Parameters : none  
// Return : tsw (data of toggle switch)  
//          OFF 0, ON 1  
//          bit0 of tsw is TSM1  
//          bit1 of tsw is TSM2  
//          bit2 of tsw is TSM3  
//          bit3 of tsw is TSM4  
// Notes : none  
// =====  
static unsigned char tsw_get(void)  
{  
    unsigned char tsw;  
    unsigned char tsw1;  
    unsigned char tsw2;  
    unsigned char tsw3;  
    unsigned char tsw4;  
  
    // read each toggle switch  
    tsw1 = (unsigned char)gpio_get_level(TSM1);  
    tsw2 = (unsigned char)gpio_get_level(TSM2);  
    tsw3 = (unsigned char)gpio_get_level(TSM3);  
    tsw4 = (unsigned char)gpio_get_level(TSM4);  
    // set each bit of tsw  
    tsw = tsw1; // bit0  
    tsw = tsw | (tsw2 << 1); // bit1  
    tsw = tsw | (tsw3 << 2); // bit2  
    tsw = tsw | (tsw4 << 3); // bit3  
  
    return tsw;  
}
```

関数 tsw_get
の定義

```
// =====  
// Name : led_init  
// Function : initialize LED  
// Parameters : none  
// Return : none  
// Notes : call before closed loop  
// =====  
static void led_init(void)  
{  
    esp_rom_gpio_pad_select_gpio(LED1);  
    ESP_ERROR_CHECK(gpio_set_level(LED1, 1));  
    ESP_ERROR_CHECK(gpio_set_direction(LED1, GPIO_MODE_OUTPUT));  
    esp_rom_gpio_pad_select_gpio(LED2);  
    ESP_ERROR_CHECK(gpio_set_level(LED2, 1));  
    ESP_ERROR_CHECK(gpio_set_direction(LED2, GPIO_MODE_OUTPUT));  
    esp_rom_gpio_pad_select_gpio(LED3);  
    ESP_ERROR_CHECK(gpio_set_level(LED3, 1));  
    ESP_ERROR_CHECK(gpio_set_direction(LED3, GPIO_MODE_OUTPUT));  
    esp_rom_gpio_pad_select_gpio(LED4);  
    ESP_ERROR_CHECK(gpio_set_level(LED4, 1));  
    ESP_ERROR_CHECK(gpio_set_direction(LED4, GPIO_MODE_OUTPUT));  
    return;  
}
```

関数 led_init
の定義

```
// =====  
// Name : led_set  
// Function : set LED  
// Parameters : led  
//          OFF 0, ON 1  
//          set bit0 of led to LED1  
//          set bit1 of led to LED2  
//          set bit2 of led to LED3  
//          set bit3 of led to LED4  
// Return : none  
// Notes : none  
// =====  
static void led_set(unsigned char led)  
{  
    unsigned char led1;  
    unsigned char led2;  
    unsigned char led3;  
    unsigned char led4;  
  
    led1 = (led & BIT0) ^ BIT0; // LED1 bit0 of led  
    led2 = ((led >> 1) & BIT0) ^ BIT0; // LED2 bit1 of led  
    led3 = ((led >> 2) & BIT0) ^ BIT0; // LED3 bit2 of led  
    led4 = ((led >> 3) & BIT0) ^ BIT0; // LED4 bit3 of led  
    // set each LED  
    ESP_ERROR_CHECK(gpio_set_level(LED1, led1));  
    ESP_ERROR_CHECK(gpio_set_level(LED2, led2));  
    ESP_ERROR_CHECK(gpio_set_level(LED3, led3));  
    ESP_ERROR_CHECK(gpio_set_level(LED4, led4));  
    return;  
}
```

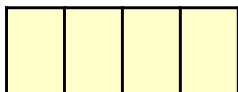
関数 led_set
の定義

例題1-2 プロトタイプ

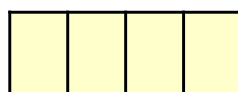
```
// --- prototypes (static)
static void          tsw_init(void);
static unsigned char tsw_get(void);
static void          led_init(void);
static void          led_set(unsigned char led);
static void          sv_init(void);
```

ファイル内で定義され
ファイル内のみで使用する関数の
プロトタイプ

入出力装置	関数	機能	入力（引数）	出力（返却値）
TSW トグルスイッチ	tsw_init	TSW を初期化する	なし	なし
	tsw_get	TSW を読む	なし	TSW のパターン
LED	led_init	LED を初期化する	なし	なし
	led_set	LED に設定する	点灯パターン	なし



TSW のパターン



点灯パターン

例題1-2 app_main 関数

```
void app_main(void)
{
```

変数の宣言

```
    unsigned char    pattern;
```

```
    // initialize toggle switch
```

```
    tsw_init();
```

```
    // initialize LED
```

```
    led_init();
```

```
    // initialize seven segment LED
```

```
    sv_init();
```

周辺機器の初期化
(関数呼出し)

pattern ← TSW の状態
(関数呼出し)

pattern を LED に設定
(関数呼出し)

```
    // read toggle switch
```

```
    pattern = tsw_get();
```

```
    // set LED
```

```
    led_set(pattern);
```

```
    for (;;) { // closed loop
```

```
        // read toggle switch
```

```
        pattern = tsw_get();
```

```
        // set LED
```

```
        led_set(pattern);
```

```
    }
```

```
}
```

終わらない繰返し

例題1-2 関数 tsw_init

```
static void tsw_init(void)
{
    esp_rom_gpio_pad_select_gpio(TSW1);
    ESP_ERROR_CHECK(gpio_set_direction(TSW1, GPIO_MODE_INPUT));
    ESP_ERROR_CHECK(gpio_pullup_en(TSW1));
    esp_rom_gpio_pad_select_gpio(TSW2);
    ESP_ERROR_CHECK(gpio_set_direction(TSW2, GPIO_MODE_INPUT));
    ESP_ERROR_CHECK(gpio_pullup_en(TSW2));
    esp_rom_gpio_pad_select_gpio(TSW3);
    ESP_ERROR_CHECK(gpio_set_direction(TSW3, GPIO_MODE_INPUT));
    ESP_ERROR_CHECK(gpio_pullup_en(TSW3));
    esp_rom_gpio_pad_select_gpio(TSW4);
    ESP_ERROR_CHECK(gpio_set_direction(TSW4, GPIO_MODE_INPUT));
    ESP_ERROR_CHECK(gpio_pullup_en(TSW4));
    return;
}
```

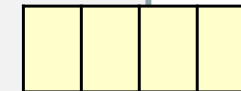
例題1-2 関数 tsw_get

```
static unsigned char tsw_get(void)
{
    unsigned char    tsw;
    unsigned char    tsw1;
    unsigned char    tsw2;
    unsigned char    tsw3;
    unsigned char    tsw4;

    // read each toggle switch
    tsw1 = (unsigned char)gpio_get_level(TSW1);
    tsw2 = (unsigned char)gpio_get_level(TSW2);
    tsw3 = (unsigned char)gpio_get_level(TSW3);
    tsw4 = (unsigned char)gpio_get_level(TSW4);
    // set each bit of tsw
    tsw = tsw1;                // bit0
    tsw = tsw | (tsw2 << 1U); // bit1
    tsw = tsw | (tsw3 << 2U); // bit2
    tsw = tsw | (tsw4 << 3U); // bit3

    return tsw;
}
```

変数 tsw の各ビットに
tsw1 ~ tsw4 の値を設定



変数 tsw

例題1-2 関数 led_init

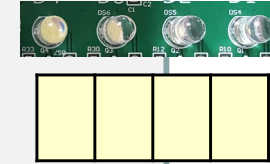
```
static void led_init(void)
{
    esp_rom_gpio_pad_select_gpio(LED1);
    ESP_ERROR_CHECK(gpio_set_level(LED1, 1U));
    ESP_ERROR_CHECK(gpio_set_direction(LED1, GPIO_MODE_OUTPUT));
    esp_rom_gpio_pad_select_gpio(LED2);
    ESP_ERROR_CHECK(gpio_set_level(LED2, 1U));
    ESP_ERROR_CHECK(gpio_set_direction(LED2, GPIO_MODE_OUTPUT));
    esp_rom_gpio_pad_select_gpio(LED3);
    ESP_ERROR_CHECK(gpio_set_level(LED3, 1U));
    ESP_ERROR_CHECK(gpio_set_direction(LED3, GPIO_MODE_OUTPUT));
    esp_rom_gpio_pad_select_gpio(LED4);
    ESP_ERROR_CHECK(gpio_set_level(LED4, 1U));
    ESP_ERROR_CHECK(gpio_set_direction(LED4, GPIO_MODE_OUTPUT));
    return;
}
```

例題1-2 関数 led_set

```
static void led_set(unsigned char led)
{
    unsigned char    led1;
    unsigned char    led2;
    unsigned char    led3;
    unsigned char    led4;

    led1 = (led & BIT0) ^ BIT0;          // LED1 bit0 of led
    led2 = ((led >> 1U) & BIT0) ^ BIT0; // LED2 bit1 of led
    led3 = ((led >> 2U) & BIT0) ^ BIT0; // LED3 bit2 of led
    led4 = ((led >> 3U) & BIT0) ^ BIT0; // LED4 bit3 of led
    // set each LED
    ESP_ERROR_CHECK(gpio_set_level(LED1, led1));
    ESP_ERROR_CHECK(gpio_set_level(LED2, led2));
    ESP_ERROR_CHECK(gpio_set_level(LED3, led3));
    ESP_ERROR_CHECK(gpio_set_level(LED4, led4));

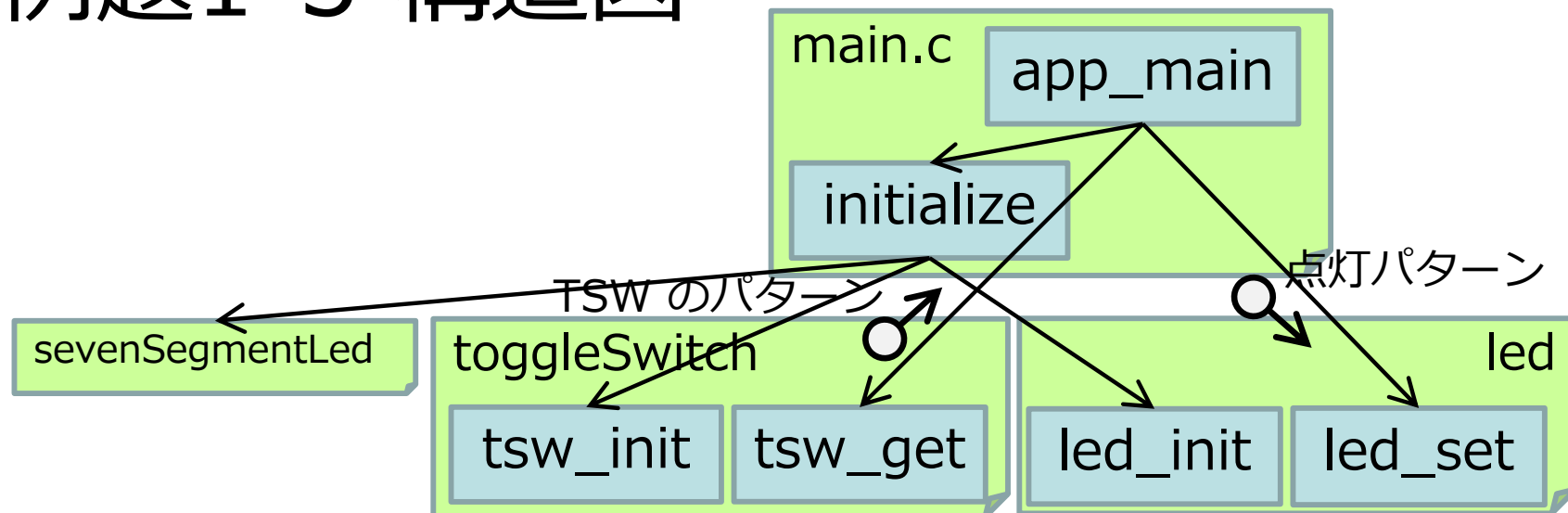
    return;
}
```



引数 led

引数 led の各ビットを
変数 led1 ~ led4 に代入

例題1-3 構造図



装置（ファイル）	関数	機能	入力（引数）	出力（返却値）
トグルスイッチ toggleSwitch.h toggleSwitch.c	tsw_init	TSW を初期化する	なし	なし
	tsw_get	TSW を読む	なし	TSW のパターン
LED led.h led.c	led_init	LED を初期化する	なし	なし
	led_set	LED に設定する	点灯パターン	なし
7セグメントLED sevenSegmentLed.h sevenSegmentLed.c	sv_init	7セグメントLED を初期化する	なし	なし

ファイル toggleSwitch.h

```
#ifndef TOGGLE_SWITCH_H
#define TOGGLE_SWITCH_H

// --- prototypes (extern)
extern void      tsw_init(void);
extern unsigned char tsw_get(void);

#endif // TOGGLE_SWITCH_H
```

実装ファイル（拡張子 .c のファイル）
で定義され
他のファイルから使用される関数の
プロトタイプ

ファイル toggleSwitch.c

```
// --- Header files (system)
#include "sdkconfig.h"
#include "hal/gpio_types.h"
#include "driver/gpio.h"
// --- Header of own file
#include "toggleSwitch.h"
```

自身のヘッダファイルを
#include

```
// --- macros
#define TSW1    GPIO_NUM_32
#define TSW2    GPIO_NUM_33
#define TSW3    GPIO_NUM_34
#define TSW4    GPIO_NUM_35
```

関数の定義
内容は同じだが
宣言に static をつけない

```
void tsw_init(void)
{
    // 省略
    return;
}

unsigned char tsw_get(void)
{
    unsigned char    tsw;
    // 省略
    return tsw;
}
```

ファイル led.h

```
#ifndef LED_H
#define LED_H

// --- prototypes (extern)
extern void led_init(void);
extern void led_set(unsigned char led);

#endif // LED_H
```

実装ファイル（拡張子 .c のファイル）
で定義され
他のファイルから使用される関数の
プロトタイプ

ファイル led.c

```
// --- Header files (system)
#include "sdkconfig.h"
#include "hal/gpio_types.h"
#include "driver/gpio.h"
// --- Header of own file
#include "led.h"
```

自身のヘッダファイルを
#include

```
// --- macros
#define LED1    GPIO_NUM_16
#define LED2    GPIO_NUM_17
#define LED3    GPIO_NUM_18
#define LED4    GPIO_NUM_19
```

関数の定義
内容は同じだが
宣言に static をつけない

```
void led_init(void)
{
    // 省略
    return;
}

void led_set(unsigned char led)
{
    // 省略
    return;
}
```

ファイル main.c

```
// --- Header files (project)
#include "toggleSwitch.h"
#include "led.h"
#include "sevenSegmentLed.h"
```

利用する関数のプロトタイプのあるヘッダを
#include

```
// --- prototypes (static)
static void initialize(void);
```

ファイル内で定義され
ファイル内のみで使用する関数の
プロトタイプ

```
void app_main(void)
{
    unsigned char    pattern;

    // initialize devices
    initialize();

    // 省略 (TSW ごとに LED を点灯)
    for (;;) { // closed loop
        // 省略 (TSW ごとに LED を点灯)
    }
}
```

```
static void initialize(void)
{
    tsw_init();
    led_init();
    sv_init();
    return;
}
```