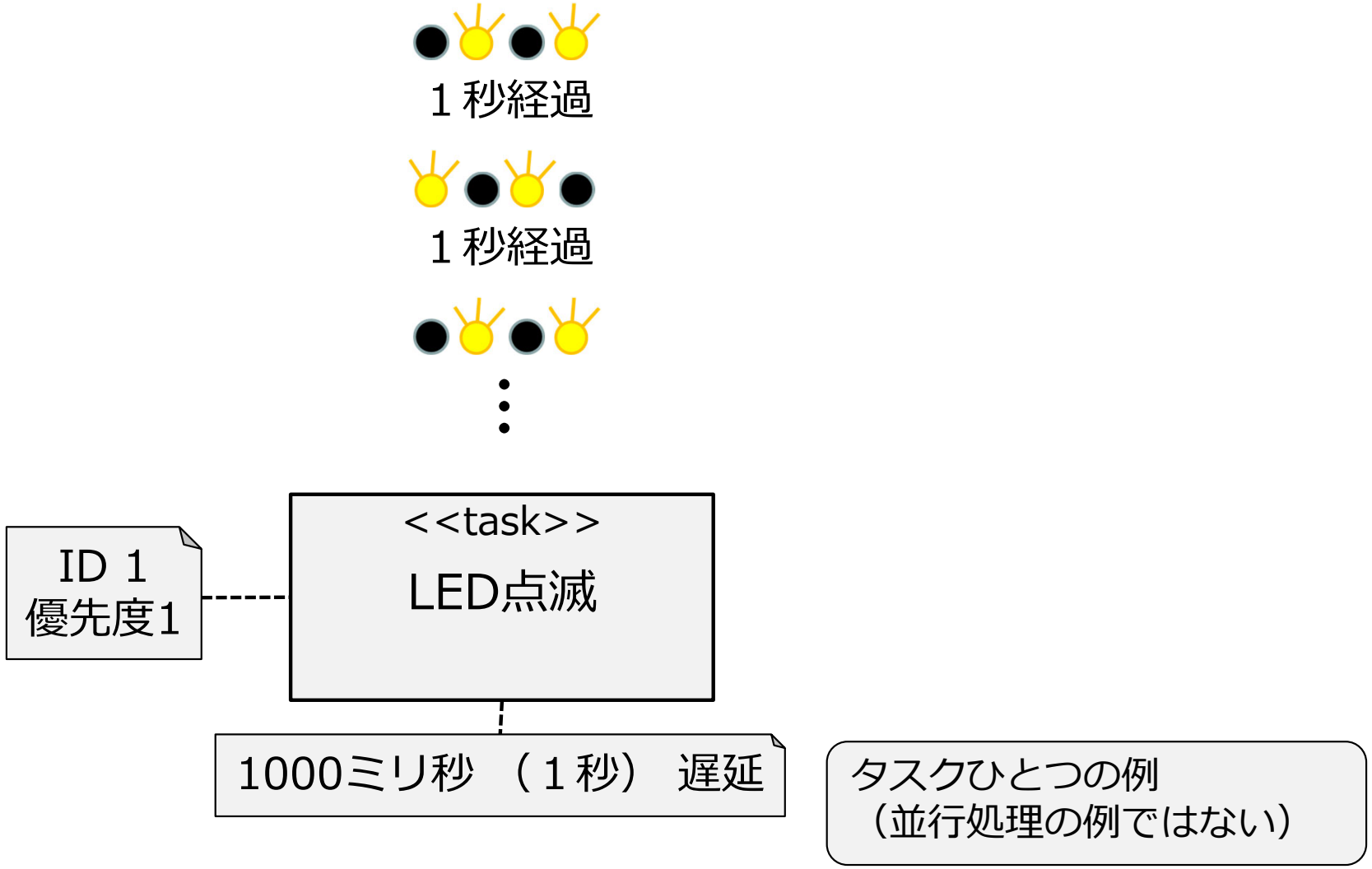
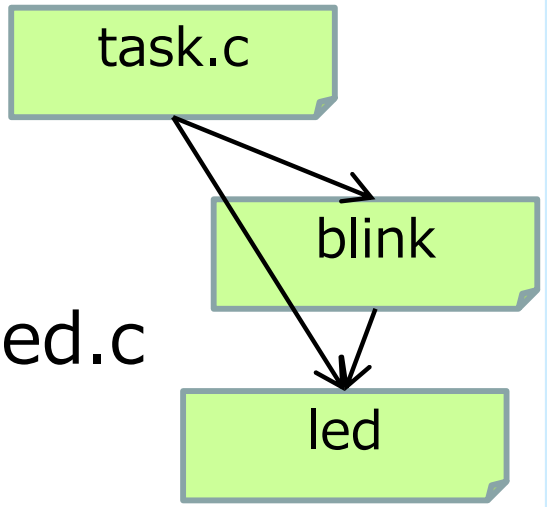
例題3　タスクの設定

- 1 秒間隔で LED の点灯パターンを反転する



ファイルの構造

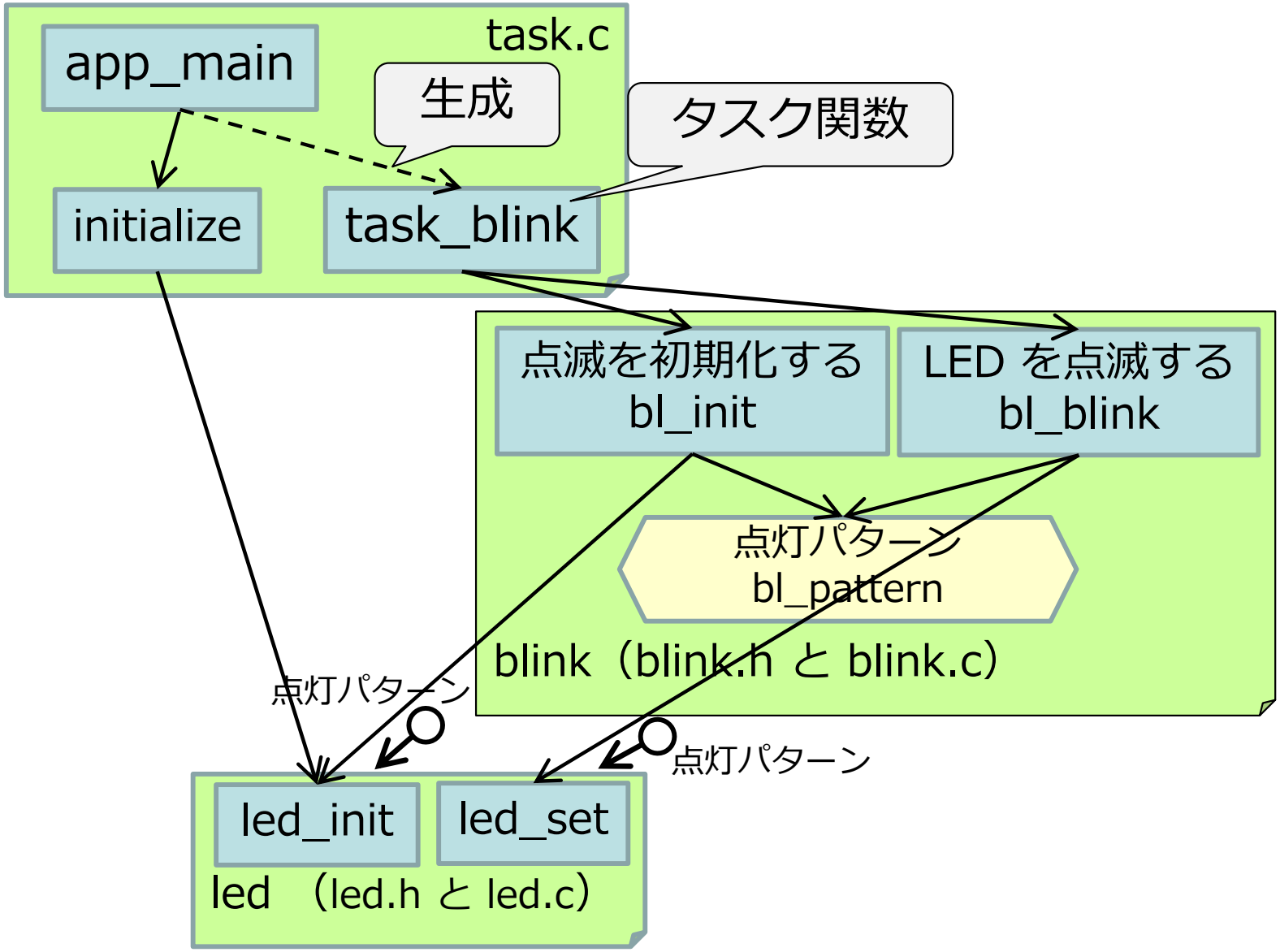


- 例題2 と同じファイル
  - LED 点滅：blink.h , blink.c , led.h , led.c
- 例題2 と異なるファイル
  - RTOS 使用： task.c （main.c ではなく）

pnshButton.h と
pushButton.c なし

ファイル	責務
task.c	システム動作 （タスクの生成、タスク関数、初期化）
blink.h と blink.c	LED 点滅
led.h と led.c	LED 出力

ファイルと関数の構造


task.c
app_main
生成
タスク関数
initialize
task_blink
点滅を初期化する
bl_init
LED を点滅する
bl_blink
点灯パターン
bl_pattern
blink（blink.h と blink.c）
点灯パターン
点灯パターン
led_init
led_set
led （led.h と led.c）

使用する API

- タスクの生成
  - xTaskCreate
    タスク関数などを指定してタスクを生成する
- 自タスクの遅延
  - vTaskDelay
    指定された時間、タスクをブロック状態に遷移し
    遅延させる

タスクの生成

xTaskCreate

引数で指定される情報に基づいてタスクを生成する

- 形式

```
BaseType_t xTaskCreate
(TaskFunction_t              pxTaskCode,
 const char *const           pcName,
 const uint32_t              ulStackDepth,
 void *const                 pvParameters,
 UBaseType_t                 uxPriority,
 TaskHandle_t *const         pxCreatedTask)
```

- 返却値
  - pdPASS　タスクを生成できたとき
  - エラー値 タスクを生成できなかったとき（projdefs.h で定義）

パラメータ

xTaskCreate

パラメータ		指定する内容
pxTaskCode	起動番地	タスクの開始の番地 タスクを実装する関数の関数名
pcName	名前	デバッグなどで使用できる名前 （RTOS は使用しない）
ulStackDepth	スタック領域のサイズ	バイト数で指定
pvParameters	タスクに渡す引数	タスクが引数を使用する場合 例題では使用していない（NULL）
uxPriority	優先度	値が大きい方が優先度高 アイドルタスクは優先度 0
pxCreatedTask	タスクのハンドルを 代入する領域	生成されたタスクのハンドルを 代入する領域

自タスクの遅延

vTaskDelay

引数で指定される時間、自タスクを遅延させる

- 形式

  void vTaskDelay (const TickType_t xTicksToDelay)
- 返却値　　なし
- パラメータ
  - xTicksToDelay　タスクを遅延させるティック数
    - マクロ pdMS_TO_TICKS などを使って実時間を指定できる
      例） pdMS_TO_TICKS(1000) は 1000 ミリ秒のティック数

ファイル task.c

```
// =====================================================
// File    : task.c
// Role    : Sample03 create task
// Date    : 2024.04.25
// Author  : Osaka Sangyo University
// =====================================================
// --- Header files (system)
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
// --- Header files (project)
#include "led.h"
#include "sevenSegmentLed.h"
#include "blink.h"

// --- macros
#define STACK_DEPTH     ((uint32_t) 4096)
#define PRIORITY_BLINK  (tskIDLE_PRIORITY + 1)
#define DELAY_BLINK     pdMS_TO_TICKS(1000)

// --- data (static)
static TaskHandle_t     taskHandleBlink = NULL;

// --- prototypes (static)
static void taskBlink(void *arg);
static void initialize(void);

// --- app_main function
// =====================================================
// Name       : app_main
// Function   : create tasks
// Parameters : none
// Return     : none
// notes      : called from Main Task
// =====================================================
void app_main(void)
{
    BaseType_t  pass;

    // initialize devices
    initialize();
    // create task
    pass = xTaskCreate(
                &taskBlink,
                "taskBlink",
                STACK_DEPTH,
                NULL,
                PRIORITY_BLINK,
                &taskHandleBlink
            );
    if (pass != pdPASS) {
        puts("cannot create taskBlink¥n");
    }
    return;
}
```

RTOS 利用のための
ヘッダファイルの #include

開発プログラムの
ヘッダファイル #include

マクロの定義（周期など）

タスクのハンドルを代入する変数

関数のプロトタイプ

app_main 関数
の定義

```
// --- task functions
// =====================================================
// Name       : taskBlink
// Function   : blinking LED
// Parameters : informations
// Return     : none
// notes      : none
// =====================================================
static void taskBlink(void *arg)
{
    bl_init();
    for (;;) {  // closed loop
        vTaskDelay(DELAY_BLINK);
        bl_blink();
    }
}

// --- functions (static)
// =====================================================
// Name       : initialize
// Function   : initialize devices
// Parameters : none
// Return     : none
// notes      : call before creating tasks
// =====================================================
static void initialize(void)
{
    led_init();
    sv_init();
    return;
}
```

タスク関数 taskBlink の
定義

初期化関数 initialize の
定義

ヘッダファイルとマクロ

■ ヘッダファイル

```
// --- Header files (system)
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
// --- Header files (project)
#include "led.h"
#include "sevenSegmentLed.h"
#include "blink.h"
```

PC にメッセージなどを表示するため

RTOS 利用のため

LED、7セグメント LED 初期化のため

点滅のため

■ マクロの定義

```
// --- macros
#define STACK_DEPTH      ((uint32_t) 4096)
#define PRIORITY_BLINK   (tskIDLE_PRIORITY + 1)
#define DELAY_BLINK      pdMS_TO_TICKS(1000)
```

タスクが使用するスタック（メモリ領域）のサイズ

タスクの優先度

周期（1000 ミリ秒）

タスクのハンドルとプロトタイプ

```
// --- data (static)
static TaskHandle_t     taskHandleBlink = NULL;

// --- prototypes (static)
static void taskBlink(void *arg);
static void initialize(void);
```

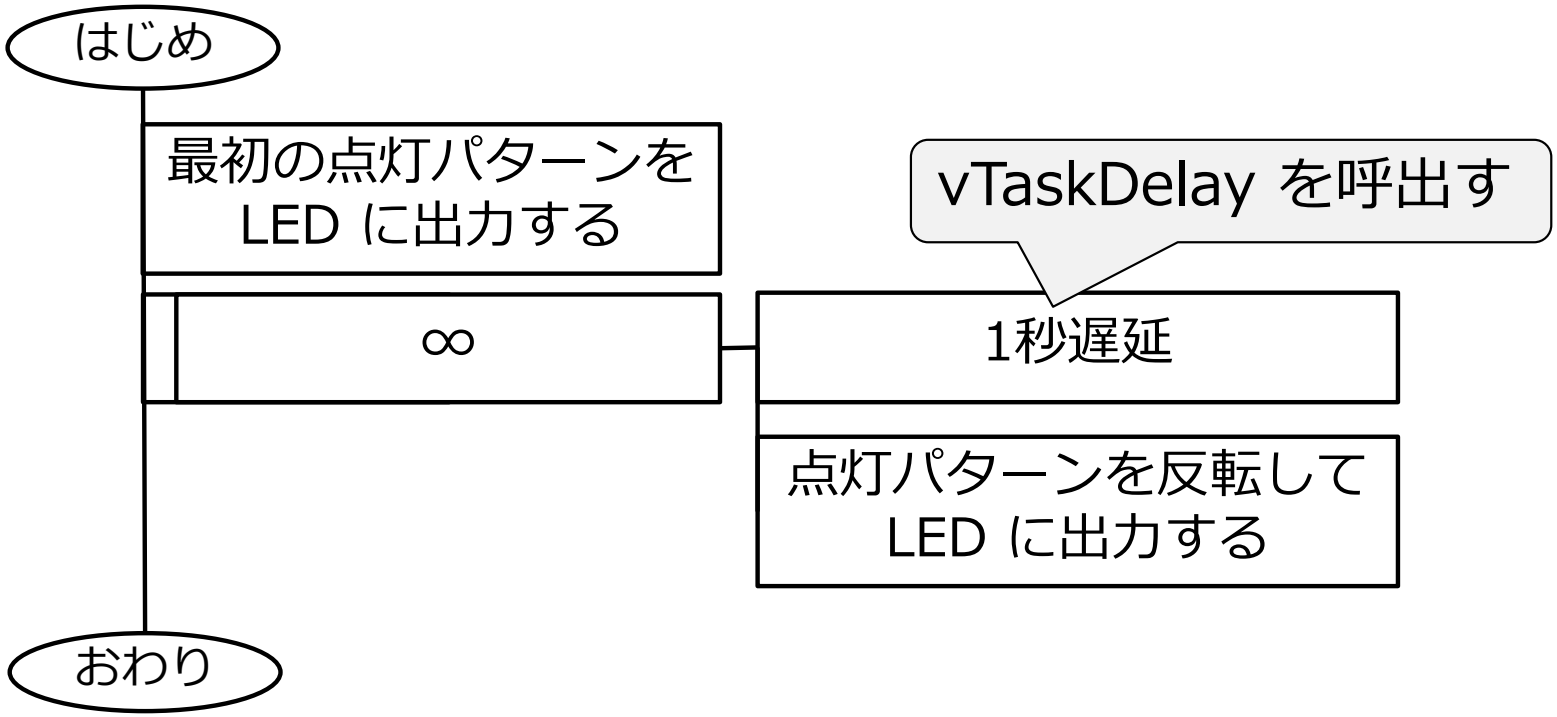
タスクのハンドルを
代入する変数

タスク関数 taskBlink のプロトタイプ

初期化関数 initialize のプロトタイプ

タスク関数のアルゴリズム

- 自タスクを 1 秒 （1000ミリ秒） 間ブロック状態に遷移させ、点灯パターンを反転する
  これを繰り返す



タスク関数 taskBlink

```
static void taskBlink(void *arg)
{
    bl_init();
    for (;;) {  // closed loop
        vTaskDelay(DELAY_BLINK);
        bl_blink();
    }
}
```

最初の点灯パターンを
LED に出力する

タスクを 1秒（1000ミリ秒）間
ブロック状態に遷移させる

点灯パターンを反転して
LED に出力する

app_main 関数

```
void app_main(void)
{
    BaseType_t  pass;

    // initialize devices
    initialize();
    // create task
    pass = xTaskCreate(
                &taskBlink,
                "taskBlink",
                STACK_DEPTH,
                NULL,
                PRIORITY_BLINK,
                &taskHandleBlink
            );
    if (pass != pdPASS) {
        puts("cannot create taskBlink¥n");
    }
    return;
}
```

関数 initialize

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

LED や 7セグメントLED の初期化

点滅タスクの生成

タスク関数 taskBlink のアドレス

タスクが使用するスタックのサイズや優先度を指定

生成したタスクのハンドルが代入される領域

タスクの生成に失敗したときPC にメッセージを表示する