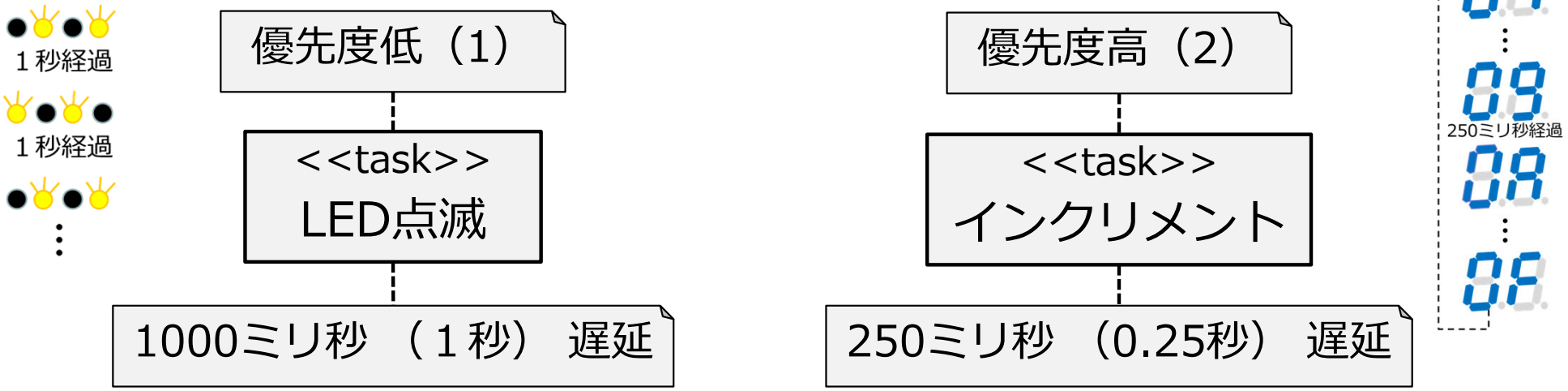
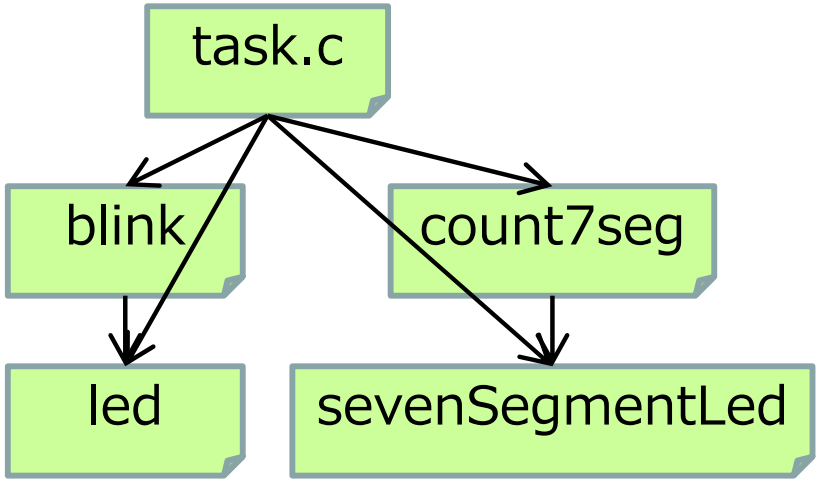
例題4　二つのタスク

- 二つのプログラムを並行動作させる
  - LED 点滅（例題3）
    - 1000 ミリ秒間隔で LED を点滅する
  - インクリメント
    - 250 ミリ秒間隔で数を数え、十六進数表記で右の 7 セグメント LEDに表示する（左の 7 セグメント LED はゼロ）



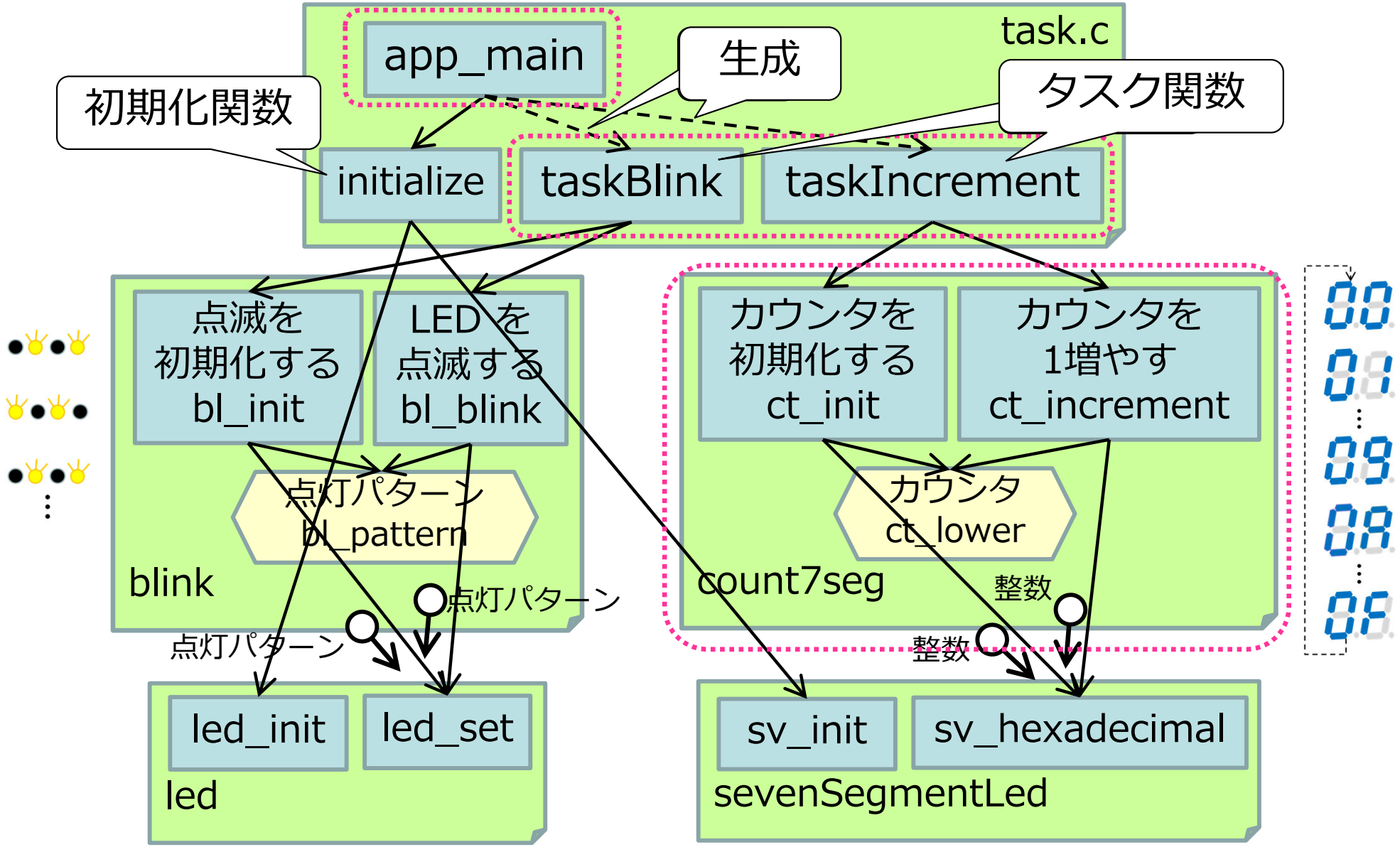
ファイルの構造



- 例題3 のファイルに
  カウントに関連する
  ファイルを追加
  - count7seg.h と count7seg.c
  - sevenSegmentLed.h と sevenSegmentLed.c

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）
count7seg	計数（7セグメントLED表示）
sevenSegmentLed	7セグメントLED表示
blink	LED 点滅
led	LED 出力

ファイルと関数の構造


task.c
app_main
生成
タスク関数
初期化関数
initialize
taskBlink
taskIncrement
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
点灯パターン
点灯パターン
led_init
led_set
led
カウンタを
初期化する
ct_init
カウンタを
1増やす
ct_increment
カウンタ
ct_lower
count7seg
整数
整数
sv_init
sv_hexadecimal
sevenSegmentLed

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

例題3と同じ

ファイル task.c のポイント

- ヘッダファイルの #include

```
// --- 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"
#include "count7seg.h"
```

点滅のため

カウントのため

- マクロの定義

```
// --- macros
#define STACK_DEPTH           ((uint32_t) 409[illegible])
#define PRIORITY_BLINK        (tskIDLE_PRIORITY + 1)     // lower
#define PRIORITY_INCREMENT    (tskIDLE_PRIORITY + 2)     // higher
#define DELAY_BLINK           pdMS_TO_TICKS(1000)
#define DELAY_INCREMENT       pdMS_TO_TICKS(250)
```

点滅タスクの優先度（低）
アイドルタスクは優先度ゼロ

カウントタスクの優先度（高）

それぞれのタスクの周期

タスクのハンドルとプロトタイプ

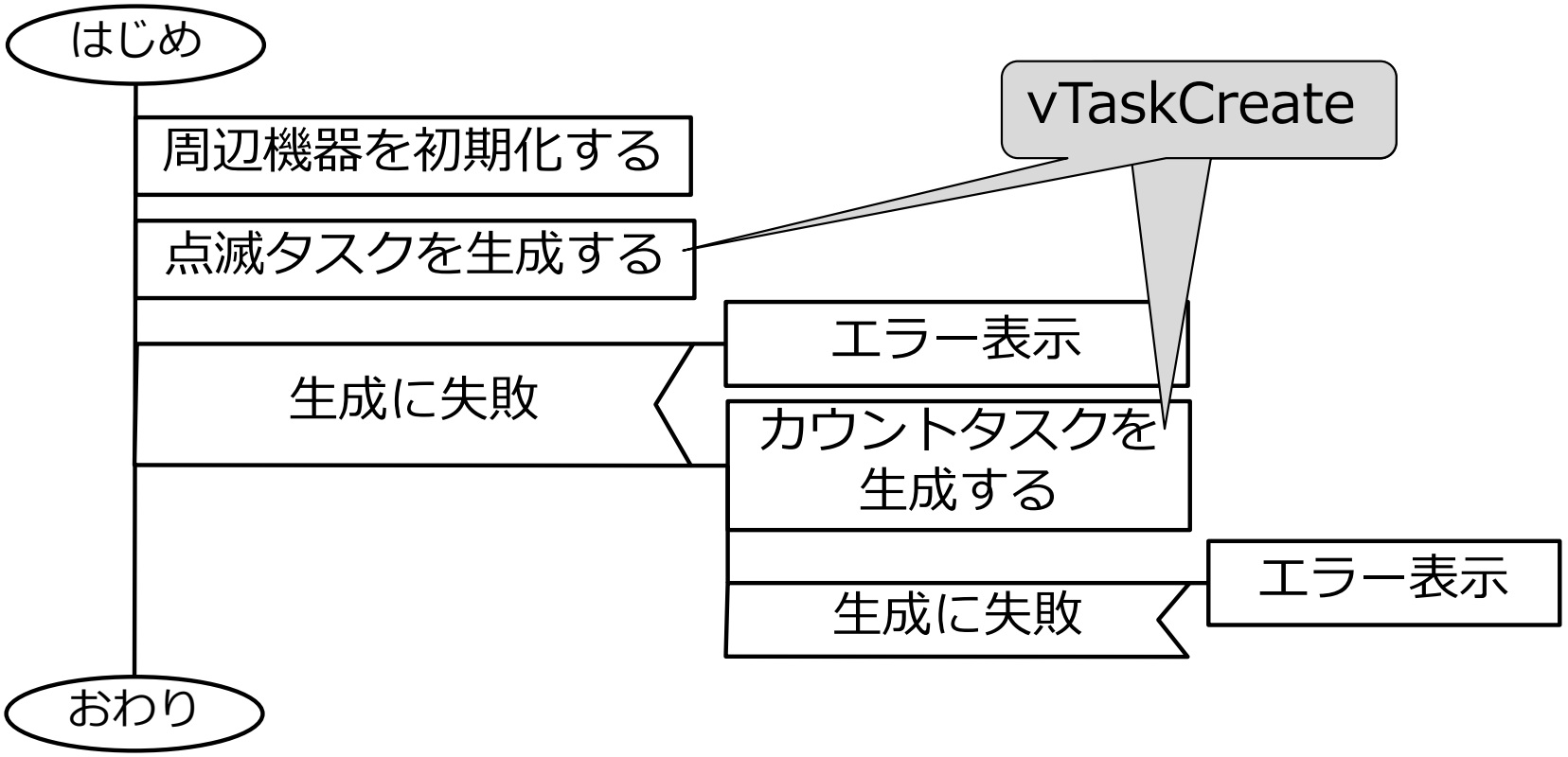
```
// --- data (static)
static TaskHandle_t     taskHandleBlink = NULL;
static TaskHandle_t     taskHandleIncrement = NULL;

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

各タスクのハンドルを代入する変数

各タスクのタスク関数 のプロトタイプ

app_main 関数のアルゴリズム

- 周辺機器を初期化しタスクを生成して終了する
  - 終わらない繰返しを実行しない



app_main 関数

関数 initialize

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

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

    // initialize devi
    initialize();
    // create tasks
    pass = xTaskCreate(
                &taskBlink,
                "taskBlink",
                STACK_DEPTH,
                NULL,
                PRIORITY_BLINK,
                &taskHandleBlink
            );
    if (pass != pdPASS) {
        // エラー表示（省略）
    } else {
        pass = xTaskCreate(
                    &taskIncrement,
                    "taskIncrement",
                    STACK_DEPTH,
                    NULL,
                    PRIORITY_INCREMENT,
                    &taskHandleIncrement
                );
        if (pass != pdPASS) {
            // エラー表示（省略）
        }
    }
    return;
}
```

点滅タスクの生成

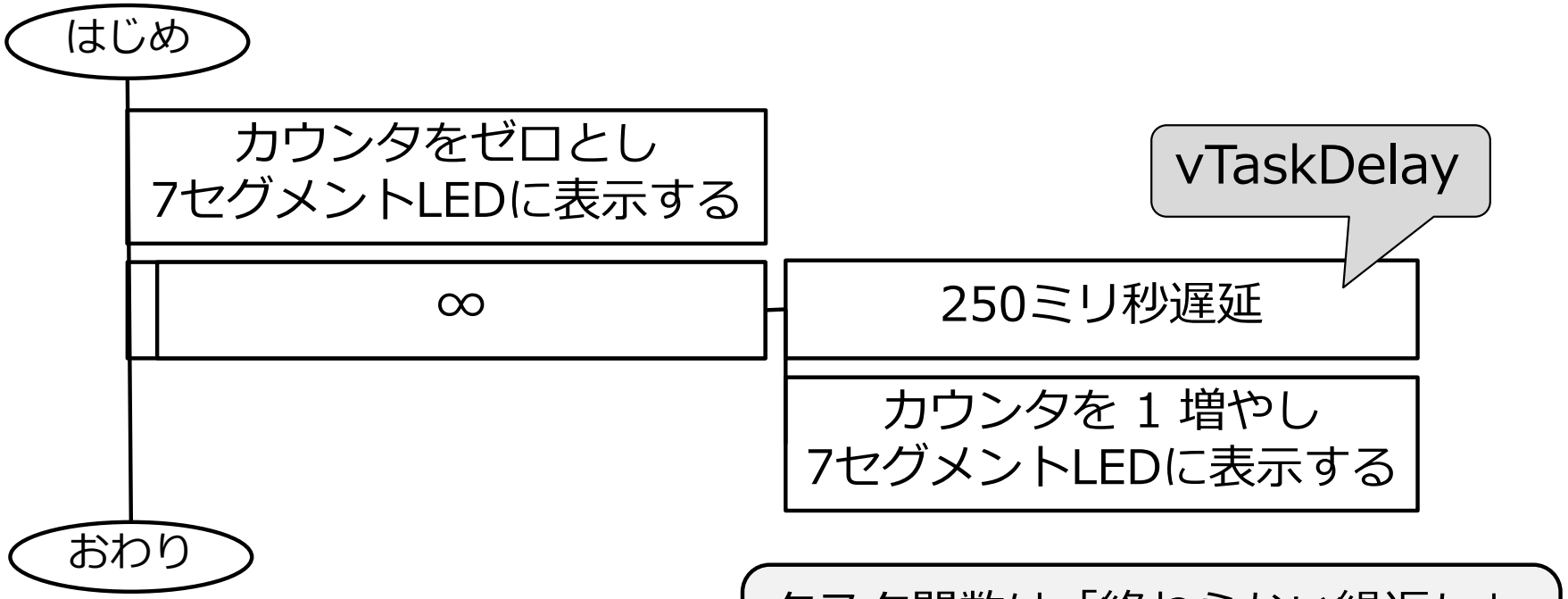
タスク関数 taskBlink のアドレス

カウントタスクの生成

タスク関数 taskIncrement のアドレス

タスク関数のアルゴリズム taskIncrement

- 0.25 秒 （250ミリ秒） 間隔でカウンタを 1 増やし 7セグメントLED に表示する
  これを繰り返す



タスク関数 taskIncrement

```
static void taskIncrement(void *arg)
{
    // monitor
    TaskHandle_t handle = xTaskGetCurrentTaskHandle();
    char* taskname = pcTaskGetName(handle);
    puts(taskname);

    ct_init();
    for (;;) {  // closed loop
        vTaskDelay(DELAY_INCREMENT);
        ct_increment();
    }
}
```

動作確認のための記述（後述）

カウンタをゼロとし
7セグメントLEDに表示する

タスクを 0.25秒（250ミリ秒）間
ブロック状態に遷移させる

カウンタを 1 増やし
7セグメントLEDに表示する

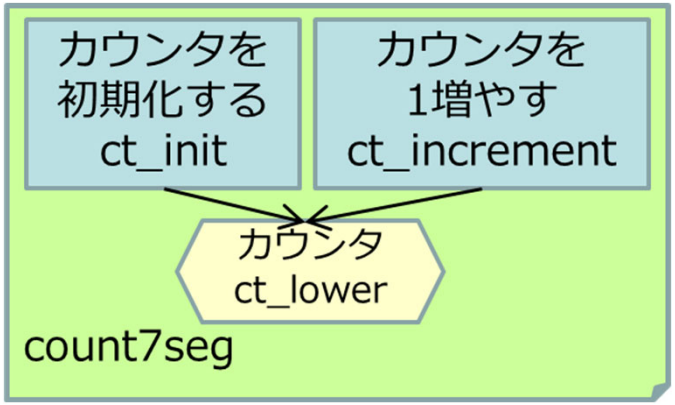
ファイル count7seg

- ヘッダファイル

```
extern void ct_init(void);
extern void ct_increment(void);
```

- 実装ファイル
  - ヘッダファイルの #include

```
// --- Header files (system)
#include <stdio.h>
// --- Header files (project)
#include "sevenSegmentLed.h"
#include "led.h"
// --- Header of own file
#include "count7seg.h"
```



  - マクロの定義

```
#define CT_OVER 0x10U
```

  - データの定義

```
static unsigned char    ct_lower;
```

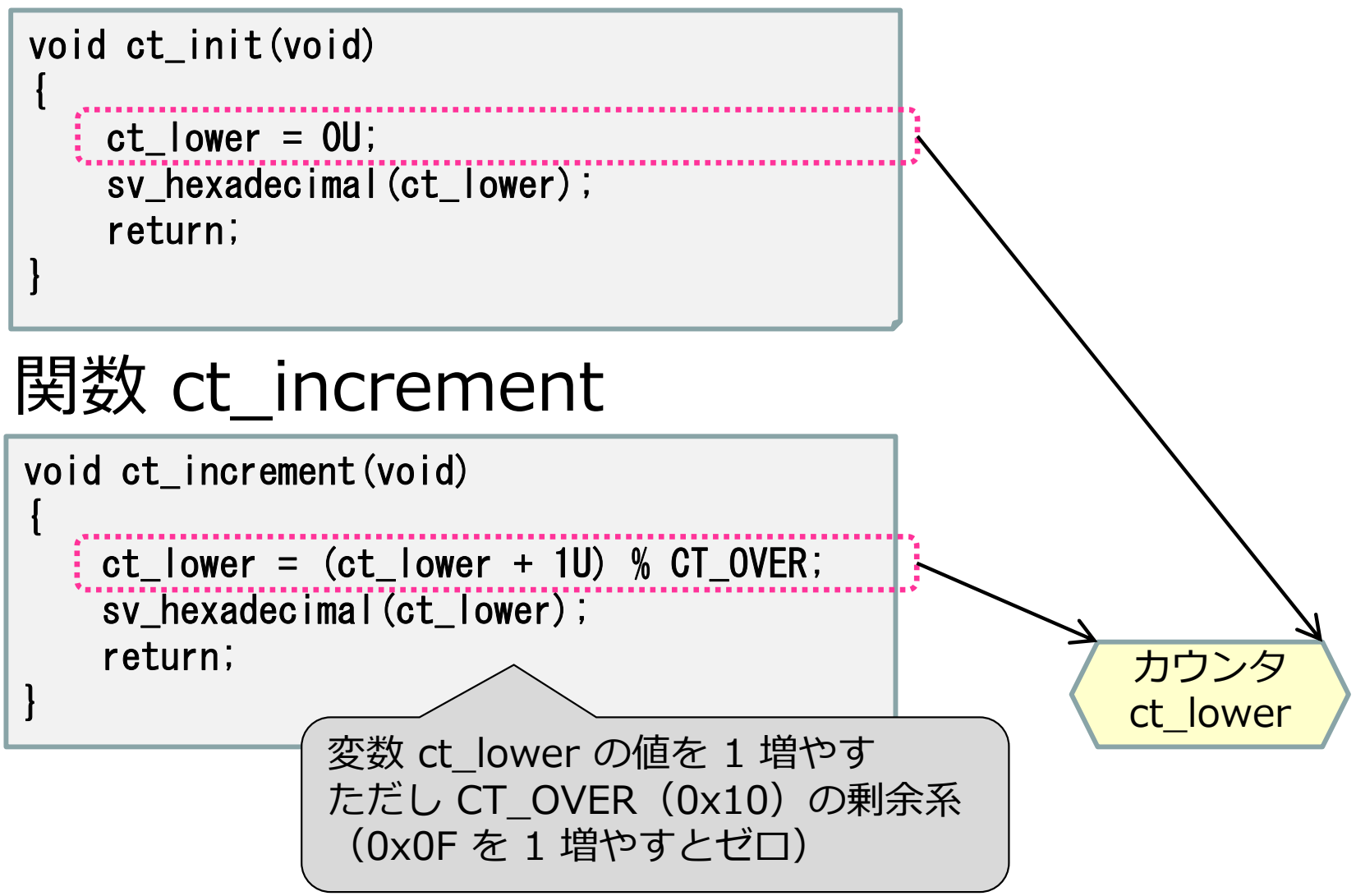
関数 ct_increment

- 関数 ct_init

```
void ct_init(void)
{
    ct_lower = 0U;
    sv_hexadecimal(ct_lower);
    return;
}
```

- 関数 ct_increment

```
void ct_increment(void)
{
    ct_lower = (ct_lower + 1U) % CT_OVER;
    sv_hexadecimal(ct_lower);
    return;
}
```



参考：動作確認

- tools フォルダの monitor.bat を利用すると開発 PC にメッセージなどを出力できる
  - 動作確認に利用できます

例）タスク関数 taskBlink

```
// --- Header files (system)
#include <stdio.h>

static void taskBlink(void *arg)
{
    // monitor
    TaskHandle_t handle = xTaskGetCurrentTaskHandle();
    char* taskname = pcTaskGetName(handle);
    puts(taskname);

    bl_init();
    for (;;) {  // closed loop
        vTaskDelay(DELAY_BLINK);
        bl_blink();
    }
}
```

puts などを使うため

実行中のタスクのハンドルと（handle）
タスクの名前を取得する（taskname）

実行中のタスクの名前を表示する

参考：動作確認の操作

① ./build.bat
ビルド

② ./flash.bat com〇
転送

③ ./monitor.bat com〇
モニタリング（監視）

④ Control+]
モニタリング停止

```
PS C:¥Espressif¥frameworks¥esp-idf-v5.0.2> cd C:¥work¥Sample04¥tools
PS C:¥work¥Sample04¥tools> ./build
```

```
or run 'idf.py -p (PORT) flash'

C:¥work¥Sample04>popd
PS C:¥work¥Sample04¥tools> ./flash.bat com
```

```
Hard resetting via RTS pin...
Done

C:¥work¥Sample04>popd
PS C:¥work¥Sample04¥tools> ./monitor.bat com
```

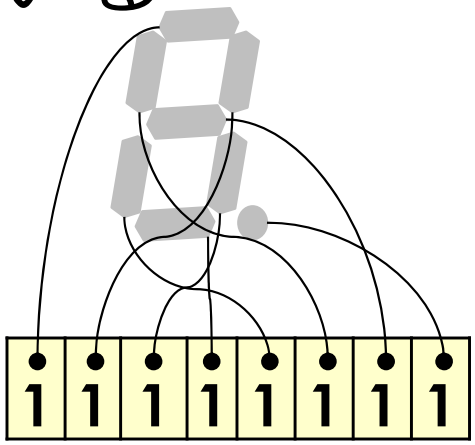
```
I (271) heap_init: At 3FF80000 len 00002000 (8 KiB): RTCRAM
I (279) spi_flash: detected chip: generic
I (282) spi_flash: flash io: dio
W (286) spi_flash: Detected size(4096k) larger than the size in the bina
 size in the binary image header.
I (300) cpu_start: Starting scheduler on PRO CPU.
taskIncrement
taskBlink
```

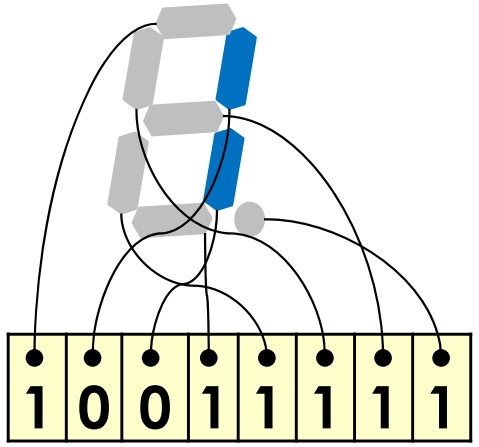
puts の出力

```
I (300) cpu_start: Starting scheduler on PRO CPU.
taskIncrement
taskBlink


C:¥work¥Sample04>popd
PS C:¥work¥Sample04¥tools>
```

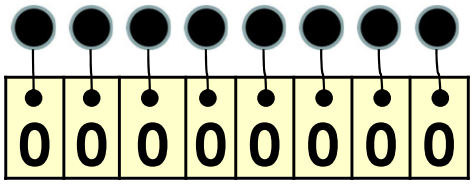
参考：7セグメントLED

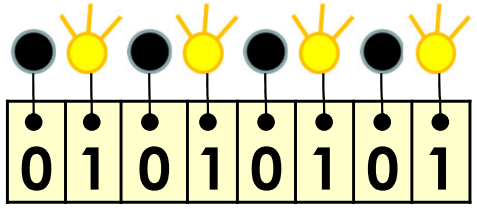
- 7 セグメント LED の各セグメントが、ポートの各ビットに対応している
  - 0　　点灯
  - 1　　消灯





- 8 つの LED の場合
  - 1　　消灯
  - 0　　点灯





参考：7セグメントLED の表示

- ファイル sevenSegmentLed のヘッダファイル

```
#ifndef SEVEN_SEGMENT_LED_H
#define SEVEN_SEGMENT_LED_H

// --- prototypes (extern)
extern void sv_init(void);
extern void sv_hexadecimal(unsigned char number);

#endif  // SEVEN_SEGMENT_LED_H
```

引数の８ビットのデータ（0～255）を
二つの7セグメントLEDに表示する

関数 sv_hexadecimal の定義

0 ～ F を構成する
８ビットのデータ

データを
7セグメントLED
に出力する関数