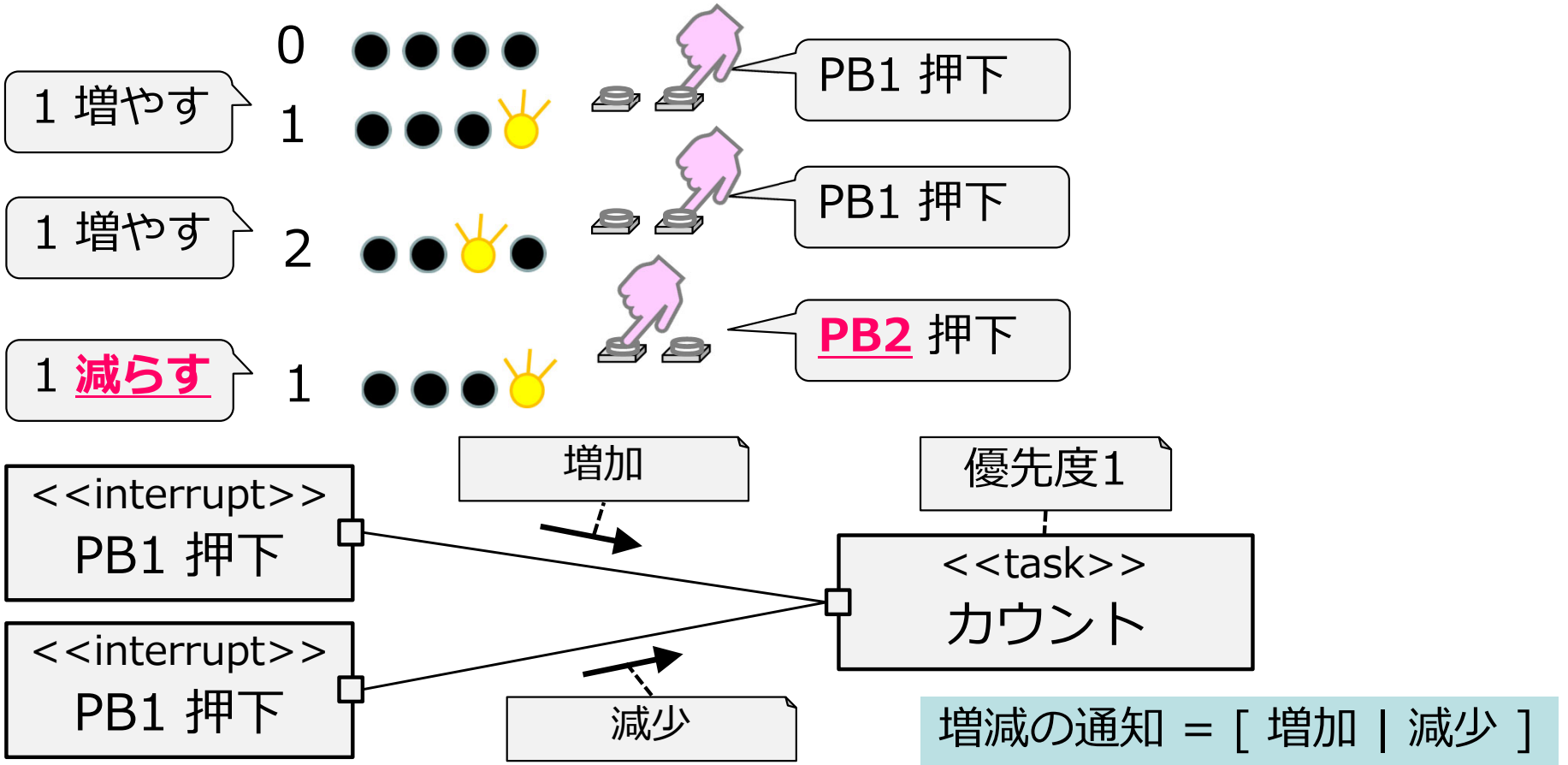
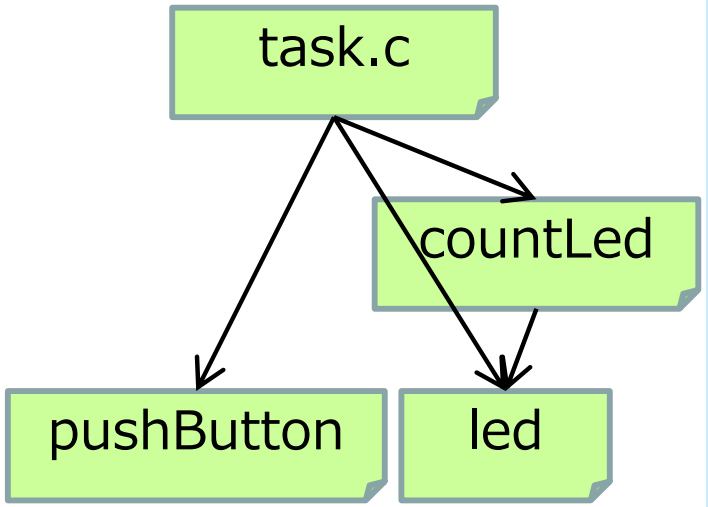
例題8　二つの事象の通知

- プッシュボタン PB1 が押されたとき数を 1 増やし、PB2 が押されたとき数を 1 減らし二進数のパターンで LED を点灯する



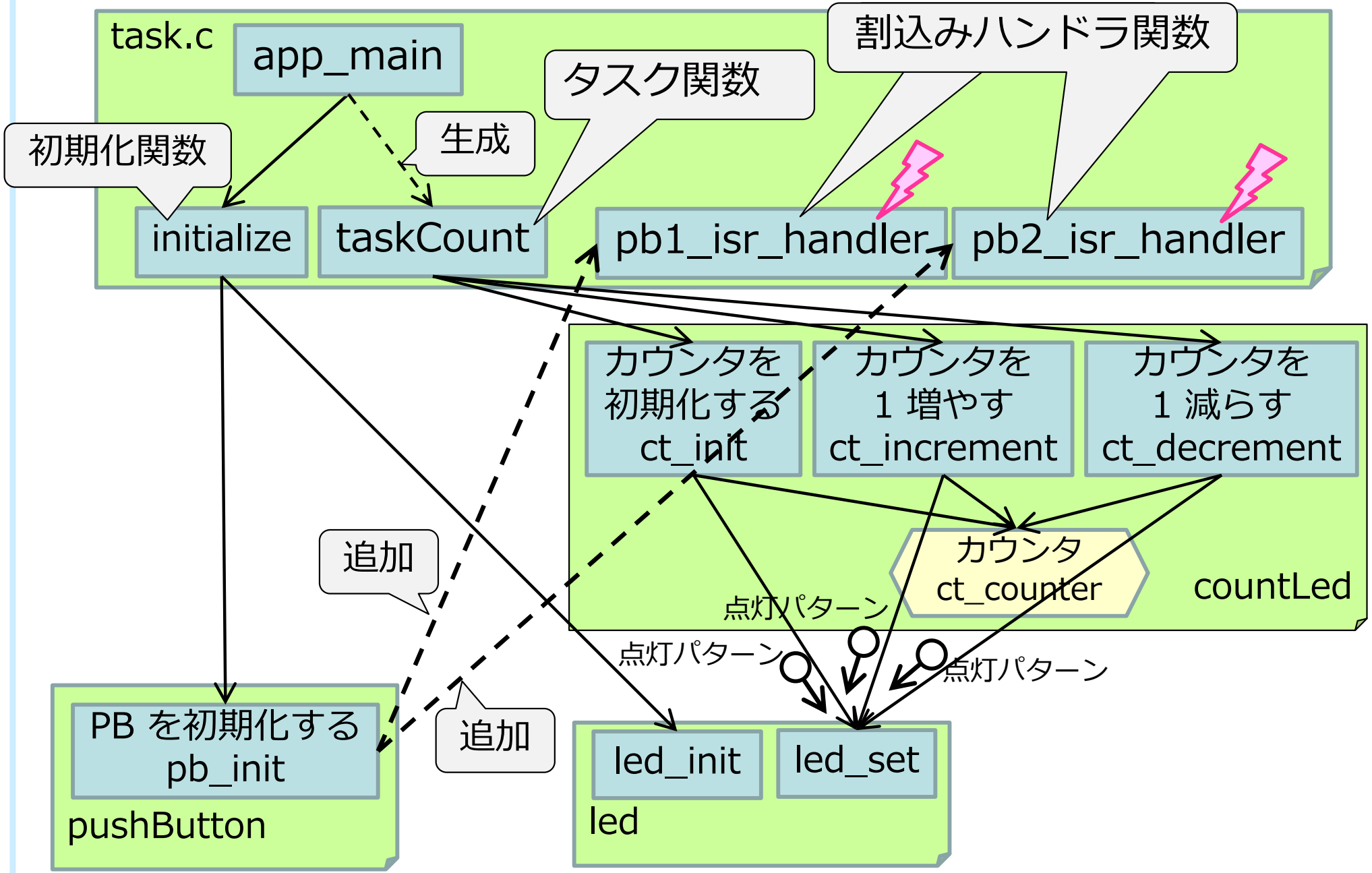
ファイルの構造



- ファイル countLed
  - 数を数え、二進数のパターンで LED を点灯する
- ファイル pushButton
  - プッシュボタンを初期化し、割込みハンドラ関数を追加する

ファイル	責務
task.c	システム動作 ・タスクの生成（app_main 関数） ・タスク関数、割込みハンドラ関数 ・初期化関数
countLed	数を数えて LED に表示
led	LED 出力
pushButton	プッシュボタン

ファイルと関数の構造


task.c
app_main
生成
タスク関数
割込みハンドラ関数
初期化関数
initialize
taskCount
pb1_isr_handler
pb2_isr_handler
カウンタを
初期化する
ct_init
カウンタを
1 増やす
ct_increment
カウンタを
1 減らす
ct_decrement
カウンタ
ct_counter
countLed
追加
点灯パターン
点灯パターン
点灯パターン
追加
PB を初期化する
pb_init
pushButton
led_init
led_set
led

使用する API

- タスクの生成（省略）
- タスクへの通知
  - xTaskNotifyWait（既出）
    自タスクへの通知を待つ（ブロック状態に遷移）
    - ブロック状態が解除されたときの通知値を受け取ることができる
  - xTaskNotifyFromISR（既出）
    割込み処理からタスクに通知する
    - 通知先のタスクの通知値のビットを設定することができる

自タスクへの通知待ち

xTaskNotifyWait

再掲

自タスクへの通知を待つ

- 形式

```
BaseType_t xTaskNotifyWait
(uint32_t   ulBitsToClearOnEntry,
 uint32_t   ulBitsToClearOnExit,
 uint32_t   *pulNotificationValue,
 TickType_t xTicksToWait)
```

通知値を受け取る引数

- 返却値
  - pdPASS　通知を受理した
  - pdFAIL　通知を受理していない

パラメータ

xTaskNotifyWait

再掲

パラメータ		指定する内容
ulBitsToClearOnEntry	待ち開始時にクリアするビット	通知値のビットのうち待ち開始時にクリアするビット（補足参照）
ulBitsToClearOnExit	待ち終了時にクリアするビット	通知値のビットのうち待ち終了時にクリアするビット（補足参照）
pulNotificationValue	待ち解除時の値を代入する領域	待ちが解除されたときの通知値を代入する領域（呼出しもとに返す）
xTicksToWait	最大待ち時間	ブロック状態で通知を待つ最大待ち時間ティック数で指定

通知値を受け取る引数

補足）　タスクは、内部に通知を受け取るためのデータ（通知値/notifications）を持っている
xTaskNotifyWait は待ち開始/終了時に、通知値をビット単位でクリアできる

タスクへの通知

再掲

xTaskNotifyFromISR

タスクに通知する（割込みコンテキスト用）

- 形式

```
BaseType_t xTaskNotifyFromISR
(TaskHandle_t    xTaskToNotify,
 uint32_t        ulValue,
 eNotifyAction   eAction,
 BaseType_t      *pxHigherPriorityTaskWoken )
```

通知値を設定する引数

- 返却値
  - pdPASS　通知は通知先のタスクに受理された
    引数 eAction が eNoAction/eSetBit/eIncrement
    のとき、返却値は常に pdPASS

再掲

パラメータ

xTaskNotifyFromISR

パラメータ		指定する内容
xTaskToNotify	タスクのハンドル	通知先のタスクのハンドル
ulValue	通知に関する値	RTOS によって値がどのように使われるかは、引数 eAction との関連で決まる
eAction	通知時のアクション	eNoAction 何もしない eSetBits ulVasue のビットを設定する eIncrement 通知値を 1 増やす など
pxHigherPriorityTaskWoken 通知先のタスクの優先度の情報		pdTRUE 通知によってブロック状態が解除されるタスクの優先度が実行中のタスクよりも高いとき（コンテキストの切り替えを要求する必要がある） pdFALSE　上記以外

引数 eAction が eSetBits のとき 通知値に設定するビット

引数 ulValue で指定されているビットを通知値に設定する

ファイル task.c のポイント

#include 、マクロ、変数、プロトタイプ

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

// --- macros
// task
#define STACK_DEPTH      ((uint32_t) 4096)
#define PRIORITY_COUNT   (tskIDLE_PRIORITY + 1)
// notify
#define CLEAR_NONE       ((uint32_t) 0)
#define CLEAR_ALL        ULONG_MAX
#define TICKS_TO_WAIT    pdMS_TO_TICKS(1000 * 60 * 60)    // 1 hour
#define VALUE_NONE       ((uint32_t) 0)
#define VALUE_INCREMENT  ((uint32_t) 0x01)
#define VALUE_DECREMENT  ((uint32_t) 0x02)
```

通知を待つ時間
1 時間 PB1、PB2 を操作しないと
タイムアウトする

タスクのハンドルを
代入する変数

タスク関数の
プロトタイプ

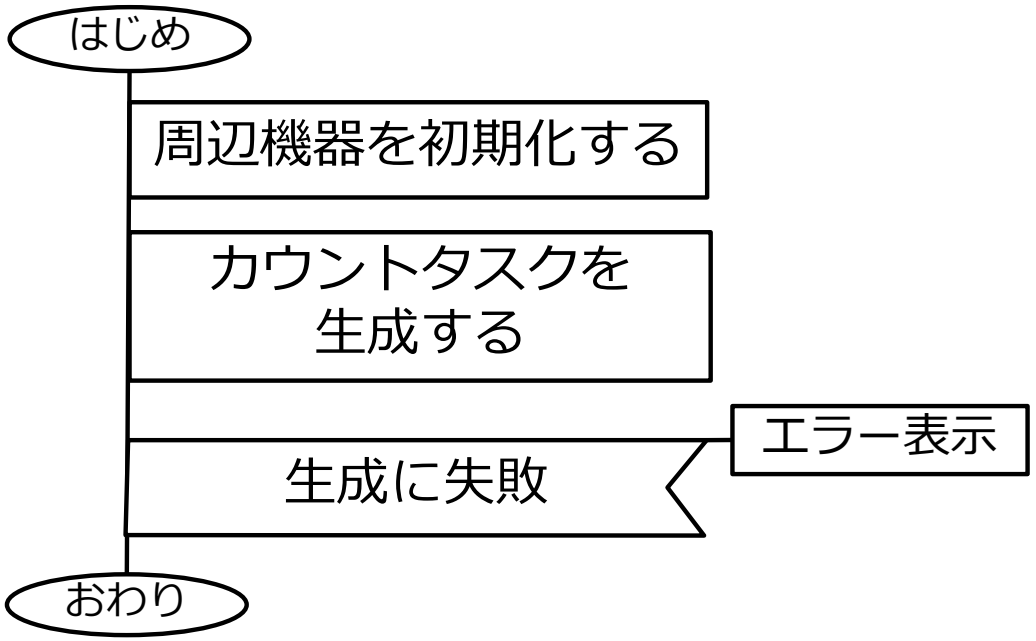
通知値に設定するビット

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

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

app_main 関数

■ アルゴリズム



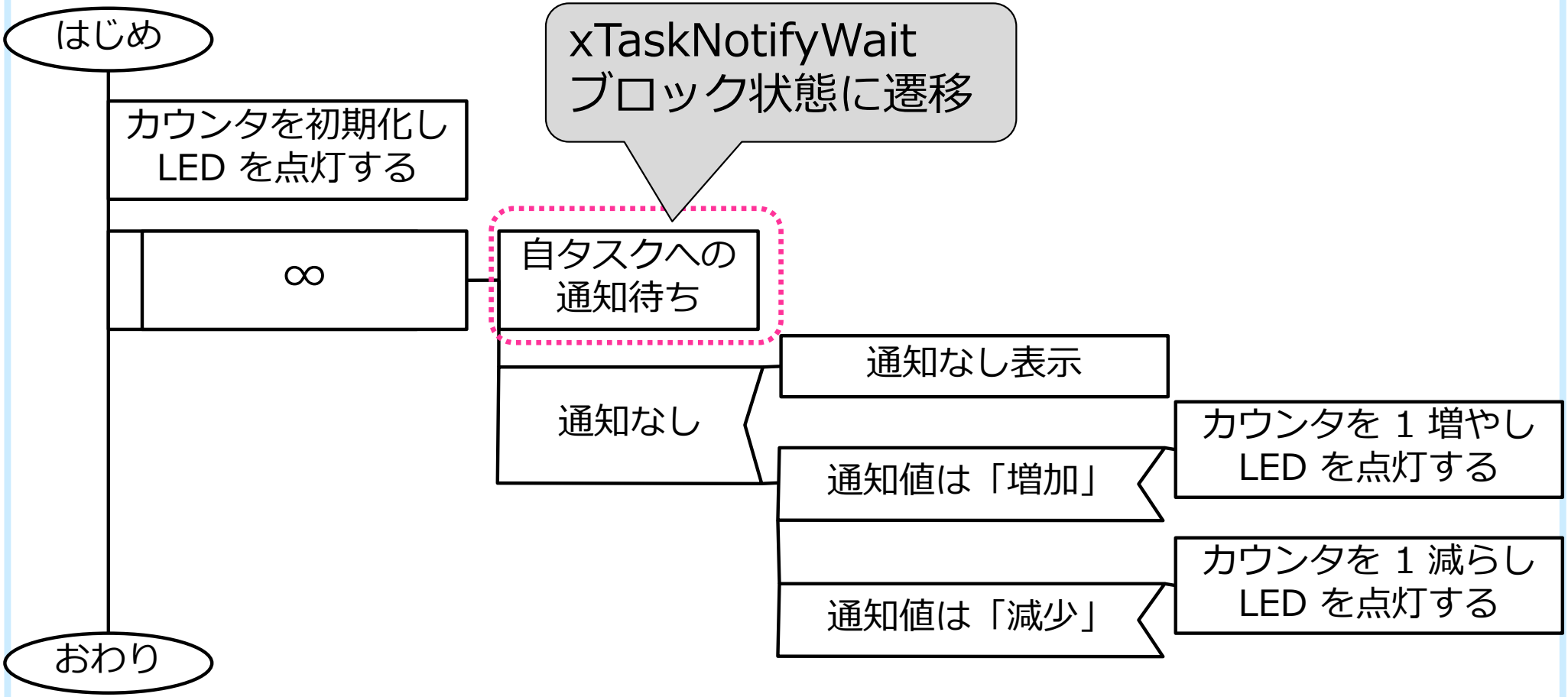
■ プログラム

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

    // initialize devices
    initialize();
    // create task
    pass = xTaskCreate(
        &taskCount,
        "taskCount",
        STACK_DEPTH,
        NULL,
        PRIORITY_COUNT,
        &taskHandleCount
    );
    if (pass != pdPASS) {
        puts("cannot create taskCount");
    }

    return;
}
```

タスク関数 taskCount のアルゴリズム



タスク関数のプログラム

```
static void taskCount(void *arg)
{
    // 動作確認（省略）
    ct_init();
    for (;;) {  // closed loop
        BaseType_t  pd;
        uint32_t    notifiedValue;
        pd = xTaskNotifyWait(
            CLEAR_NONE,
            CLEAR_ALL,
            &notifiedValue,
            TICKS_TO_WAIT
        );
        if (pd != pdPASS) {
            puts("not notified");
        } else {
            if ( (notifiedValue & VALUE_INCREMENT) != VALUE_NONE ) {
                ct_increment();
            }
            if ( (notifiedValue & VALUE_DECREMENT) != VALUE_NONE) {
                ct_decrement();
            }
        }
    }
}
```

通知値を代入するための変数

ブロック状態が解除されたとき
すべてのビットをクリアする

通知値を受け取る

通知を
待つ

通知値のビット0
（VALUE_INCREMENT）を調べる

通知値のビット1
（VALUE_DECREMENT）を調べる

受取った通知値への対応

- ビット0 が 0 か 1 か調べる

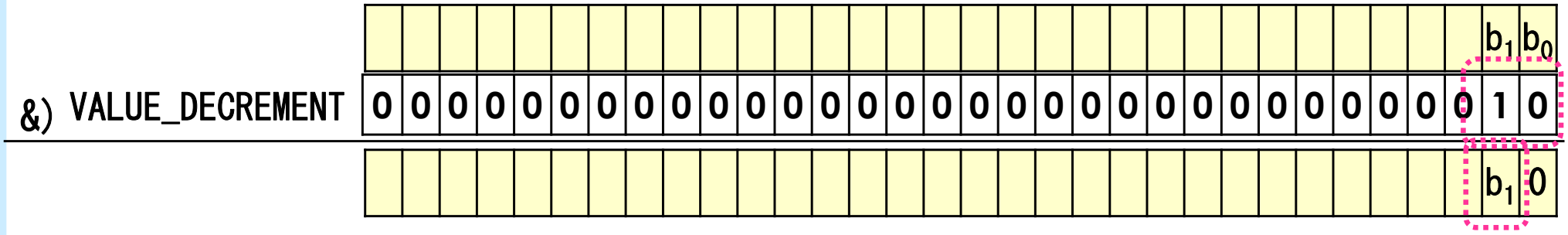
```
if ((notifiedValue & VALUE_INCREMENT) != VALUE_NONE ) {
    ct_increment();
}
```

ビット0 が 1 （!= 0）であれば、「増加」の通知

- ビット1 が 0 か 1 か調べる

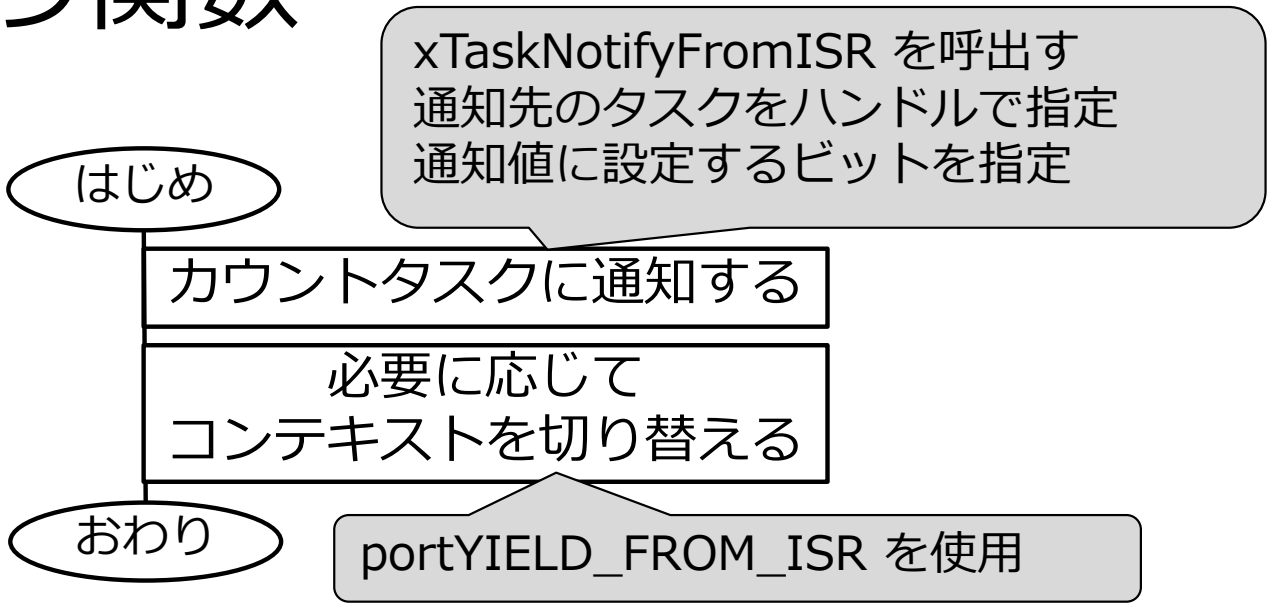
```
if ( (notifiedValue & VALUE_DECREMENT) != VALUE_NONE) {
    ct_decrement();
}
```

ビット1 が 1 （!= 0）であれば、「減少」の通知



割込みハンドラ関数

- アルゴリズム



- プログラム

```
void pb1_isr_handler(void *arg)
{
    BaseType_t  higherPriorityTaskWoken = pdFALSE;
    (void)xTaskNotifyFromISR(
        taskHandleCount,
        VALUE_INCREMENT,
        eSetBits,
        &higherPriorityTaskWoken
    );
    portYIELD_FROM_ISR(higherPriorityTaskWoken);
    return;
}
```

割込みコンテキスト用のAPI

通知先のタスクの通知値のビットの中で
VALUE_INCREMENT で 1 となっている
ビット（ビット0）を 1 とする

pb2_isr_handler 省略