Each RTOS task has an array of *task notifications*. Each task notification has a *notification state* that can be either 'pending' or 'not pending', and a 32-bit *notification value*. The constant configTASK_NOTIFICATION_ARRAY_ENTRIES sets the number of indexes in the task notification array. Prior to FreeRTOS V10.4.0 tasks only had a single task notification, not an array of notifications.

A *direct to task notification* is an event sent directly to a task, rather than indirectly to a task via an intermediary object such as a queue, event group or semaphore. Sending a direct to task notification to a task sets the state of the target task notification to 'pending'. Just as a task can block on an intermediary object such as a semaphore to wait for that semaphore to be available, a task can block on a task notification to wait for that notification's state to become pending.

Sending a direct to task notification to a task can also optionally update the value of the target notification in one of the following ways:

- Overwrite the value regardless of whether the receiving task has read the value being overwritten or not.
- Overwrite the value, but only if the receiving task has read the value being overwritten.
- Set one or more bits in the value.
- Increment (add one to) the value.

Use of xTaskNotifyGive() and ulTaskNotifyTake()

Task 1 gives Notification to Task 2 : Try single Notification and multiple notifications

Task2 Takes Notification : Change the xClearCountOnExit as pdTRUE and pdFALSE and see change in output

Try changing waiting time for task *xTicksToWait* in NotifyTake API

```
#include<stdio.h>
#include<FreeRTOS.h>
#include<task.h>
#include<queue.h>


TaskHandle_t mytaskHandle1 = NULL;
TaskHandle_t mytaskHandle2 = NULL;
TickType_t t;

void mytask2(void* p)
{
   int NotificationValue;
   while (1)
```

```
    {
        printf("\nWelcome to KJSCE from Task2:");
        /* Query the time */
        t = xTaskGetTickCount();
        printf("\tAt time = %d", t);
            NotificationValue = ulTaskNotifyTake(pdTRUE, (TickType_t)portMAX_DELAY);
        if (NotificationValue > 0)
                        {
            printf("\nNotification received % d", NotificationValue);
            /* Query the time */
            t = xTaskGetTickCount();
            printf("\tAt time = %d", t);
        }
        vTaskDelay(100);

    }
    fflush(stdout);
}

void mytask1(void* p)
{
    while (1)
    {
        printf("\nWelcome to KJSCE from Task1");
            printf("\nTask1 Gives Notification to Task 2");
            /* Query the time */
            t = xTaskGetTickCount();
            printf("\tAt time = %d", t);
        xTaskNotifyGive(mytaskHandle2);
        vTaskDelay(1000);
        }
    fflush(stdout);

}
void main_blinky(void)
{
    int pass = 10;
    xTaskCreate(mytask1, "task1", 128, (void*)pass, 1, &mytaskHandle1);
    xTaskCreate(mytask2, "task2", 128, (void*)pass, 1, &mytaskHandle2);
    vTaskStartScheduler();
    while (1)
    {
        printf("My idle task");
    }
}
```