```c
/*
 * main.c
 * Created: 26/09/2019
 * Author:  Cornelis Peter Hiemstra, Noel Janes & Flavia Pérez Cámara
 * Platform: Arduino Due / Atmel SAM3X8E
 * Purpose:  Main function declaration, initializes board peripherals, calls
 *           all other init functions and starts the FreeRTOS scheduler.
 */

#include <asf.h>
#include <FreeRTOS.h>
#include <task.h>

#include <blink.h>
#include <command_po.h>
#include <tc.h>


int main (void)
{
    /* System clock initialization */
    sysclk_init();

    /* Board peripherals initialization */
    board_init();

    /* Init tasks */
    init_blink();
    init_cmd();
    init_tc();

    /* Start scheduler */
    vTaskStartScheduler();

    /* Should never reach here ... */
    return 1;
}
```

```c
/*
 * command_po.c
 *
 * Created: 26/09/2019
 * Author:  Cornelis Peter Hiemstra, Noel Janes & Flavia Pérez Cámara
 * Platform: Arduino Due / Atmel SAM3X8E
 * Purpose:  Protected object for current command
 */


#include <asf.h>
#include <FreeRTOS.h>
#include <task.h>

#include <command_po.h>

#include <stdlib.h>
#include <stdio.h>
#include <blink.h>

/* Prototypes */
unsigned char cmd;

void set_cmd ( unsigned char c) {
    cmd = c; /* sets the current command value to new the value*/
}


unsigned char get_cmd() {
    return cmd; /* returns the current command value*/
}

/* Initializes command object
*/
void init_cmd() {

    const usart_serial_options_t usart_serial_options = {
        .baudrate   = CONF_UART_BAUDRATE,
        .charlength = CONF_UART_CHAR_LENGTH,
        .paritytype = CONF_UART_PARITY,
        .stopbits   = CONF_UART_STOP_BITS
    };

    stdio_serial_init(CONF_UART, &usart_serial_options);
}
```

```c
/*
 * command_po.h
 *
 * Created: 26/09/2019
 * Author:  Cornelis Peter Hiemstra, Noel Janes & Flavia Pérez Cámara
 * Platform: Arduino Due / Atmel SAM3X8E
 * Purpose:  Protected object for current command
 */

#ifndef COMMAND_PO_H_
#define COMMAND_PO_H_


/**
 * Initializes command object
 */
void init_cmd ( void );

/**
 * Returns current command, ensures mutual exclusion
 */
unsigned char get_cmd( void );


/**
 * Sets current command, ensures mutual exclusion
 */
void set_cmd ( unsigned char );



#endif /* COMMAND_PO_H_ */
```

```c
/*
 * tc.c
 * Created: 26/09/2019
 * Author:  Cornelis Peter Hiemstra, Noel Janes & Flavia Pérez Cámara
 * Platform: Arduino Due / Atmel SAM3X8E
 * Purpose:  Receives char via UART in an interrupt and interprets them
 */


#include <FreeRTOS.h>
#include <task.h>
#include <queue.h>
#include <conf_uart_serial.h>
#include <stdio_serial.h>
#include <command_po.h>
#include <tc.h>

/* Prototypes */
static unsigned char blink_cmd;
static unsigned char TC_input;

/* Declaration of the Queue*/
QueueHandle_t xQueueTC;


/**
 * Initializes UART console, UART interrupt, command queue
 * and creates TC task
 */
void init_tc( )
{
    /* Configure UART communication */
    const usart_serial_options_t usart_serial_options = {
        .baudrate   = CONF_UART_BAUDRATE,
        .charlength = CONF_UART_CHAR_LENGTH,
        .paritytype = CONF_UART_PARITY,
        .stopbits   = CONF_UART_STOP_BITS
    };

    /* redirect printf / scanf function calls to UART */
    stdio_serial_init(CONF_UART, &usart_serial_options);

    /* Enable UART Interrupt for RX ready */
    uart_enable_interrupt( CONF_UART, UART_IER_RXRDY );

    /* Enable Interrupt Handler */
    NVIC_EnableIRQ((IRQn_Type) CONF_UART_ID);

    /* ATTENTION!
     * It is very important to manually set the priority of the interrupt
     * to a higher value, as per default all interrupts have priority 0,
     * which is the highest priority. This causes problems when calling
     * FreeRTOS APIs that end in ...fromISR( ), since they check if they
     * were not called from an ISR with higher or equal priority as
     * configMAX_SYSCALL_INTERRUPT_PRIORITY
    */
```

```
    NVIC_SetPriority(CONF_UART_ID, configMAX_PRIORITIES);

    xQueueTC = xQueueCreate( 4, sizeof( char ) );

}
/**
 * TC task function, interprets received data from UART
 */
void handleInput() {

    switch (TC_input) {
        case ('a'):
        set_cmd('0'); /* sets the command value to 0 */
        break;
        case ('b'):
        set_cmd('1'); /* sets the command value to 1 */
        break;
        case ('c'):
        set_cmd('2'); /* sets the command value to 2 */
        break;
    }

}


/**
 * Interrupt service routine for UART RXTX
 */
void UART_Handler( )
{
    /* The UART interrupt is triggered both for RX and TX, therefore
       we have to see if RXRDY is set in the UART status register */
    if((CONF_UART->UART_SR & UART_SR_RXRDY) == UART_SR_RXRDY)
    {
        TC_input=CONF_UART->UART_RHR;
        handleInput();
    }
}
```

```c
/*
 * tc.h
 * Created: 26/09/2019
 * Author:  Cornelis Peter Hiemstra, Noel Janes & Flavia Pérez Cámara
 * Platform: Arduino Due / Atmel SAM3X8E
 * Purpose:  Receives char via UART in an interrupt and interprets them
 */


#ifndef TC_H_
#define TC_H_

/**
 * Initializes TC object
 */
void init_tc( void );
/* TC task function, interprets received data from UART
*/
void handleInput (void);

#endif /* TC_H_ */
```

```c
/*
 * blink.c
 * Created: 26/09/2019
 * Author:  Cornelis Peter Hiemstra, Noel Janes & Flavia Pérez Cámara
 * Platform: Arduino Due / Atmel SAM3X8E
 * Purpose:   Holds the blinking task and init function
 */

#include <asf.h>
#include <FreeRTOS.h>
#include <task.h>

#include <blink.h>
#include <command_po.h>
#include <tc.h>

/* Prototypes */
int delay = 0;
bool ready = false;

void blink() {

    TickType_t xLastWakeTime;       /* Last wake time of the task, needed for periodic execution */
    xLastWakeTime = xTaskGetTickCount();

    for( ;; ) {

        switch (get_cmd()) {    /* Get the command value */
            case ('0'):
            ready = true;       /* Ready state */
            break;
            case ('1'):
            delay = 250;        /* Low frequency blinking state */
            break;
            case ('2'):
            delay = 50;         /* High frequency blinking state */
            break;
        }


        if(!ready) {
            /* If not ready -> keep LED off */
            PIOB->PIO_CODR = 1 << 27;
        } else if(delay == 0) {
            /* If ready -> keep LED on continuously */
            PIOB->PIO_SODR = 1 << 27;
        } else {
            if((PIOB->PIO_ODSR & (1 << 27)) > 0) {
                /* If pin 27 is active -> turn off via Clear Output Data Register (CODR) */
                PIOB->PIO_CODR = 1 << 27;
                } else {
                /* If pin 27 is not active -> turn on via Set Output Data Register (SODR) */
                PIOB->PIO_SODR = 1 << 27;
```

```
            }
            vTaskDelayUntil(&xLastWakeTime, delay/portTICK_RATE_MS); /* Absolute
              delay */
        }
        vTaskDelay(10);

    }

    /* Tasks must not attempt to return from their implementing
    function or otherwise exit.  In newer FreeRTOS port
    attempting to do so will result in an configASSERT() being
    called if it is defined.  If it is necessary for a task to
    exit then have the task call vTaskDelete( NULL ) to ensure
    its exit is clean. */
    vTaskDelete( NULL );
}



void init_blink( )
{
    /* Initialize LED0 on Port B Pin 27*/
    PIOB->PIO_PER = 1 << 27;        /* Pin Enable Register (PER) */
    PIOB->PIO_OER = 1 << 27;        /* Output Enable Register (OER) */
    PIOB->PIO_OWER = 1 << 27;       /* Output Write Enable Register (OWER) */

        /* Create task */
        xTaskCreate(
        blink,              /* Function that implements the task. */
        "Blink Task",       /* Text name for the task. */
        250,                /* Stack size in words, not bytes. */
        NULL,               /* Parameter passed into the task. */
        1,                  /* Priority at which the task is created. */
        NULL                /* Used to pass out the created task's handle. */
        );
}
```

```c
/*
 * blink.h
 * Created: 26/09/2019
 * Author:  Cornelis Peter Hiemstra, Noel Janes & Flavia Pérez Cámara
 * Platform: Arduino Due / Atmel SAM3X8E
 * Purpose:   Holds the blinking task
 */
#ifndef BLINK_H_
#define BLINK_H_

/* Initializes and creates blink task */
void init_blink( void );
void blink(void);

#endif /* BLINK_H_ */
```