Name: **Marco Söllinger s2410306011**

Gruppe: 1 | 2

Punkte: / 24

ZÄHLER MIT VERZÖGERUNG

AUFGABE

Programmieren Sie einen 3-Bit-Zähler, der hochzählt, solange der Taster USER0 gedrückt ist und runterzählt, solange der Taster USER1 gedrückt ist. Der aktuelle Zählerwert soll an den 3 LEDs des APUS-EvalBoards ausgegeben werden.

Verwenden Sie für die Zugriffe auf die LEDs und die Taster wiederverwendbare Software-Module (.c/.h). Erstellen Sie diese, falls noch nicht vorhanden. Eine Vorlage für die LEDs ist vorgegeben.

Bei einem Überlauf (Zählerwert > 7) wird der Zähler auf 0 zurückgesetzt. Überlegen Sie sich, wie der Reset nach dem Überlauf mit geringstem Aufwand realisiert werden kann. Verwenden Sie eine beliebige Verzögerungsmethode, um die Zählgeschwindigkeit soweit zu verringern, dass die einzelnen Zählschritte erkennbar sind (<10Hz).

FRAGEN

- Was passiert bei Eingängen, bei denen kein Pullup/Pulldown—Widerstand vorhanden ist?
- Wie und an welcher Stelle im Code wird der Pullup/Pulldown Widerstand aktiviert? Welche GPIO-Register werden benötigt, um den GPIO-Pin auf Eingang zu konfigurieren und den Pullup/Pulldown zu aktivieren?
- Warum muss beim Zählen eine Verzögerung eingebaut werden?

ABGABE

- Kommentierter Sourcecode + Projekt (14 Punkte)
- ausgearbeitete Fragen (6 Punkte)
- Dokumentation (4 Punkte)

0.1 Aufgabe 1:

0.1.1 Loesungsidee

Der Button driver wurde aehnlich wie der LED driver implementiert. Es wurde ein Array mit den Button Pins und Ports erstellt.
In den einezlnen Funktionen wird mittels einer for-Schleife ueber die Buttons iteriert und der entsprechende Pin/Port angesprochen.
Wobei bei durch meiner Implementierung von uint32_t Button_GetState(void) gefordert wird, das die Definition der Buttons im Array gleich ist wie im Output dieser Funktion gefordert.
Also der erste GPIO in dem Array ist das bit 0 im Output der Funktion.

Der 3-bit Zahler wurde in der main.c implementiert.
Diese benutzt die LED_SetOut(counter) Funktion um den Zaehlerstand anzuzeigen. (Das 0. bit wird auf der Led3 angezeigt und das 3.bit auf der Led5)
Die Knopfe wurden wie in der Angabe verbunden mit dem incrementieren und decrementieren des Addierers.
Die Refreshrate des Addierers ist 2Hz.

0.1.2 Code

main.c

```c
/**
  ******************************************************************************
  * @file    main.c
  * @author  Marco Soellinger
  * @version V1.0
  * @date    14.10.2025
  * @brief   Main program body
  ******************************************************************************
  */

/* Includes ------------------------------------------------------------------*/
#include "stm32f0xx_conf.h"
#include "sysdelay.h"
#include "board_led.h"
#include "board_button.h"

/* Private typedef -----------------------------------------------------------*/
/* Private define ------------------------------------------------------------*/
/* Private macro -------------------------------------------------------------*/
/* Private variables ---------------------------------------------------------*/
/* Private function prototypes -----------------------------------------------*/
/* Private functions ---------------------------------------------------------*/

/**
  * @brief  Initialize the sys tick timer (M0 core) which is used for delay.
  * @param  None
  * @retval None
  */
void SysTick_Init(void)
{
    /* init the sys tick timer to be called every 1ms */
    SysTick_Config(SystemCoreClock / 1000);
}

/**
  * @brief  This is the SysTick interrupt handler which is called every 1ms.
            We have to increment the HAL tick counter which is used for SysDelay
  * @param  None
  * @retval None
  */
void SysTick_Handler()
{
    SysDelay_IncTicks();
}


/**
```

```
  * @brief  Main program
  * @param  None
  * @retval None
  */
int main(void)
{
    SysTick_Init();

    LED_Initialize();
    Button_Initialize();

    // counter for 3 bit Adder
    uint8_t counter = 0;

while (1){
        uint32_t state = Button_GetState();

        if(state & BUTTON_USER0_MASK){
            // increments adder and removes overflows
            counter = (counter+1) &0x7;
        }

        if(state & BUTTON_USER1_MASK){
            // dercrement adder and removes underflows
            counter = (counter-1) & (uint32_t)0x7;
        }

        LED_SetOut(counter);

        // 2 Hz freq
    SysDelay_Delay(500);

  }

}
```

board_button.h

```
/**
  ******************************************************************************
  * @file    board_button.h
  * @author  Andreas Oyrer
  * @version V1.0
  * @date    16-October-2017
  * @brief   APUS Board button control Header File.
  ******************************************************************************
  */

#ifndef __BOARD_BUTTON_H
#define __BOARD_BUTTON_H

/* Includes ------------------------------------------------------------------*/
#include <stdint.h>

/* Exported types ------------------------------------------------------------*/
#define BUTTON_USER0_MASK       ( 1 << 0)
#define BUTTON_USER1_MASK       ( 1 << 1)
#define BUTTON_WAKEUP_MASK      ( 1 << 2)


/* Exported functions ------------------------------------------------------- */

/**
  * @brief  Initialize buttons including GPIO ports.
  * @param  None
  * @retval None
*/
void Button_Initialize(void);

/**
  * @brief  Get buttons state.
  * @param  None
  * @retval The state of pressed buttons.
*/
uint32_t Button_GetState(void);

```

```
#endif /* __BOARD_BUTTON_H */
```

board_button.c

```
/**
  ******************************************************************************
  * @file    board_button.c
  * @author  Marco Soellinger
  * @version V1.0
  * @date    14-10-2025
  * @brief   APUS Board button control Source File.
  ******************************************************************************
  */

#include "stm32f0xx_gpio.h"
#include "board_button.h"

#define BUTTON_USER0_PIN       GPIO_Pin_0
#define BUTTON_USER1_PIN       GPIO_Pin_1
#define BUTTON_WAKEUP_PIN      GPIO_Pin_0

/* GPIO Pin identifier */
typedef struct _GPIO_PIN
{
    GPIO_TypeDef *port;
    uint16_t      pin;
} GPIO_PIN;

/* LED GPIO Pins */
/* Must be in the same order as the return mask */
/* first in array is on first position in return*/
static const GPIO_PIN BUTTON_PIN[] =
{
    { GPIOD, BUTTON_USER0_PIN },
    { GPIOD, BUTTON_USER1_PIN },
    { GPIOA, BUTTON_WAKEUP_PIN }
};


/**
  * @brief  Initialize buttons including GPIO ports.
  * @param  None
  * @retval None
*/
void Button_Initialize(void){
      GPIO_InitTypeDef initStruct;

    /* GPIO Ports Clock Enable */
        RCC->AHBENR |= RCC_AHBENR_GPIOAEN;
    RCC->AHBENR |= RCC_AHBENR_GPIODEN;
    /* initialize the GPIO struct */
    GPIO_StructInit(&initStruct);

    for (uint32_t idx = 0; idx < sizeof(BUTTON_PIN) / sizeof(GPIO_PIN); idx++)
    {
        /* Configure GPIO pin */
        initStruct.GPIO_Pin = BUTTON_PIN[idx].pin;
        initStruct.GPIO_Mode = GPIO_Mode_IN;
            initStruct.GPIO_PuPd = GPIO_PuPd_DOWN;

        GPIO_Init(BUTTON_PIN[idx].port, &initStruct);
    }
}

/**
  * @brief  Get buttons state.
  * @param  None
  * @retval The state of pressed buttons.
*/
uint32_t Button_GetState(void){
    uint32_t retVal = 0;

    for (uint32_t idx = 0; idx < sizeof(BUTTON_PIN) / sizeof(GPIO_PIN); idx++){
        if(GPIO_ReadInputDataBit(BUTTON_PIN[idx].port, BUTTON_PIN[idx].pin)){
            retVal |= (1 << idx);
        }
```

```
73
74    }
75
76        return retVal;
77  }
```

Fragen

- Was passiert bei Eingängen, bei denen kein Pullup/Pulldown—Widerstand vorhanden ist?
Dann ist der Pegel undefiniert, wenn nicht auf dem Schalter gedrueckt wird(Fall ABUS).
Man weiss nicht ob der Pegel hoch oder niedrig ist, da sich Signale aus der Umgebung einkoppeln koennen.
Je nach dem Potential der Umgebung kann der Pegel dann hoch oder niedrig sein, wenn die Schmittrigger Grenzen ueberschritten werden.

- Wie und an welcher Stelle im Code wird der Pullup/Pulldown Widerstand aktiviert?
Die Konfiguration wird im Initialisierungsstruktur des GPIOs vorgenommen:
initStruct.GPIO_PuPd = GPIO_PuPd_DOWN;
Diese Konfiguration wird dann mit GPIO_Init() uebernommen.

- Welche GPIO-Register werden benötigt, um den GPIO-Pin auf Eingang zu konfigurieren und den Pullup/Pulldown zu aktivieren?
GPIO Eingang: GPIOx_MODER (00 die einen Bits fuer den Pin um ihn als Eingang konfigurieren)
GPIO Pullup/Pulldown: GPIOx_PUPDR (10 fuer PullDown, 01 fuer PullUp)

- Warum muss beim Zählen eine Verzögerung eingebaut werden?
Weil der Mikrocontroller so schnell ist, dass er den Knopf mehrfach als gedrueckt erkennt, obwohl der Knopf nur einmal gedrueckt wurde.
Bzw. wenn der Knopf gehalten wird, sich die Werte so schnell aendern, dass man nur leuchtende Leds sieht.