Name: **Marco Söllinger, s2410306011, Gruppe 1**

Gruppe: 1 | 2

Punkte: / 24

ALARMANLAGE + TAKTQUELLE

AUFGABE

Programmieren Sie eine Alarmanlage, welche 2 unabhängige Alarmzonen überwacht. Die Alarmzonen werden z.B. mit Bewegungsmelder, Glasbruchsensoren oder Fensteröffnungskontakten versehen.

Wird ein Alarm ausgelöst (Taster wird gedrückt), so wird die zugewiesene Alarmzonen-LED eingeschaltet. Die Sirenen-LED soll nun blinken (Frequenz egal, sollte aber erkennbar sein).

Der Alarm kann durch Drücken der Alarm-Aus-Taste deaktiviert werden. Dadurch werden alle LEDs ausgeschaltet. Ändern Sie beim Drücken des Alarm-Aus-Tasters zusätzlich einmalig die Frequenz des Prozessors (SYSCLK) auf 24MHz. Prüfen Sie, welche Schritte zum Ändern des Takt notwendig sind und lesen Sie die Details dazu im Datenblatt nach.

BELEGUNG

Funktion	Bezeichnung
Alarmzone 1 Eingang	Taster USER0
Alarmzone 2 Eingang	Taster USER1
Alarmzone 1 LED	LED4
Alarmzone 2 LED	LED5
Sirenen-LED (soll blinken wenn Alarm aktiv)	LED3
Alarm-Aus-Taste	Taster WAKEUP

FRAGEN

- Was ändert sich mit der neu eingestellten Frequenz? Warum?
- Mit welcher Frequenz wird der Prozessor betrieben, wenn das Hauptprogramm gestartet wird. Prüfen Sie also, welche Initialisierung im Startup-Code durchgeführt wird. Beschreiben Sie die einzelnen Schritte im Detail und kennzeichnen Sie den Takt-Pfad in der Takt-Übersichtsgrafik (siehe Seite 104 im STM32F0x2 Reference Manual). Diese Grafik soll ebenfalls in der Abgabe angefügt werden.

ABGABE

- Kommentierter Sourcecode (16 Punkte)
- Ausgearbeitete Fragen und Dokumentation (8 Punkt)

Aufgabe 1:

1.1 Loesungsidee

Es wurde die Alarmanlagenlogic in der main.c implementiert.
Fuer das periodische Blinken der LED wurden die SysTick verwendet.
Fuer das Ansteuern der Peripherie wurde die gegebenen Board Treiber verwendet.

Fuer das, wechseln der Systemclock Frequenz wurde die Funktion SetSysClock in system_stm32f0xx.c kopiert und angepasst.
Das wechseln zur HSE Clock und ausschalten des PLL wurde noch in die standert SetSysClock Funktion hinzugefuegt.
Die neue Implementierung heisst SwitchClockTo24Mhz() und liegt in der main.c Datei.
Ausserdem wurde das Aktualisieren der SysTick Zeit in dieser Funktion hinzugefuegt.

1.2 Code

Die Board treiber wurden nicht veraendert und sind in den Board Unterordnern zu finden.

main.c

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

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

#include "board_button.h"
#include "board_led.h"
#include "stm32f0xx_conf.h"
#include "stm32f0xx_flash.h"
#include "stm32f0xx_rcc.h"
#include "stm32f0xx_tim.h"
#include "sysdelay.h"

/* Private typedef -----------------------------------------------------------*/
/* Private define ------------------------------------------------------------*/
#define ALARMZONE1_BUTTON BUTTON_USER0_MASK
#define ALARMZONE2_BUTTON BUTTON_USER1_MASK
#define ALARM_OFF_BUTTON BUTTON_WAKEUP_MASK

#define ALARMZONE1_LED LED4
#define ALARMZONE2_LED LED5
#define ALARM_SIREN LED3

#define SIREN_TOGGLE_MS 100u

/* Private macro -------------------------------------------------------------*/
/* Private variables ---------------------------------------------------------*/
static volatile uint32_t g_ms = 0;
static bool zone1_active = false;
static bool zone2_active = false;

/* Private function prototypes -----------------------------------------------*/
static void SysTick_Init(void);
void SysTick_Handler(void); // <<< muss global & exakt so hei en
static void SwitchClockTo24Mhz(void);

/* Private functions ---------------------------------------------------------*/

/**
 * @brief  Initialize the sys tick timer (M0 core) which is used for delay.
 * @param  None
 * @retval None
```

```
 */
static void SysTick_Init(void) {
  SystemCoreClockUpdate();
  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() {
  g_ms++;
  SysDelay_IncTicks();
}

/**
 * @brief  Umschalten auf SYSCLK = 24 MHz (HSI/2 * 6  ber PLL)
 *         Modfifyed from sytem configuration
 */
static void SwitchClockTo24Mhz(void) { // Copied from System_stm32f and modified
  __IO uint32_t StartUpCounter = 0, HSEStatus = 0;

  // Make sure HSI is on
  RCC->CR |= RCC_CR_HSION;
  while ((RCC->CR & RCC_CR_HSIRDY) == 0) {
  }

  // Switch SYSCLK to HSI
  RCC->CFGR &= ~RCC_CFGR_SW;
  RCC->CFGR |= RCC_CFGR_SW_HSI;
  while ((RCC->CFGR & RCC_CFGR_SWS) != RCC_CFGR_SWS_HSI) {
  }

  // Turn off PLL
  RCC->CR &= ~RCC_CR_PLLON;
  while (RCC->CR & RCC_CR_PLLRDY) {
  }

  // From SystemInit()

  /* SYSCLK, HCLK, PCLK configuration ----------------------------------------*/
  /* Enable HSE */
  RCC->CR |= ((uint32_t)RCC_CR_HSEON);

  /* Wait till HSE is ready and if Time out is reached exit */
  do {
    HSEStatus = RCC->CR & RCC_CR_HSERDY;
    StartUpCounter++;
  } while ((HSEStatus == 0) && (StartUpCounter != HSE_STARTUP_TIMEOUT));

  if ((RCC->CR & RCC_CR_HSERDY) != RESET) {
    HSEStatus = (uint32_t)0x01;
  } else {
    HSEStatus = (uint32_t)0x00;
  }

  if (HSEStatus == (uint32_t)0x01) {
    /* Enable Prefetch Buffer and set Flash Latency */
    FLASH->ACR = FLASH_ACR_PRFTBE | FLASH_Latency_0; // Modified

    /* HCLK = SYSCLK */
    RCC->CFGR |= (uint32_t)RCC_CFGR_HPRE_DIV1;

    /* PCLK = HCLK */
    RCC->CFGR |= (uint32_t)RCC_CFGR_PPRE_DIV1;

    /* PLL configuration */
    RCC->CFGR &= (uint32_t)((uint32_t)~(RCC_CFGR_PLLSRC | RCC_CFGR_PLLXTPRE |
                                        RCC_CFGR_PLLMULL));
    RCC->CFGR |= (uint32_t)(RCC_CFGR_PLLSRC_PREDIV1 |
                            RCC_CFGR_PLLXTPRE_PREDIV1 | RCC_CFGR_PLLMULL3);

    /* Enable PLL */
    RCC->CR |= RCC_CR_PLLON;
```

```
127
128     /* Wait till PLL is ready */
129     while ((RCC->CR & RCC_CR_PLLRDY) == 0) {
130     }
131
132     /* Select PLL as system clock source */
133     RCC->CFGR &= (uint32_t)((uint32_t)~(RCC_CFGR_SW));
134     RCC->CFGR |= (uint32_t)RCC_CFGR_SW_PLL;
135
136     /* Wait till PLL is used as system clock source */
137     while ((RCC->CFGR & (uint32_t)RCC_CFGR_SWS) != (uint32_t)RCC_CFGR_SWS_PLL) {
138     }
139   } else { /* If HSE fails to start-up, the application will have wrong clock
140                 configuration. User can add here some code to deal with this
141               error */
142   }
143
144   // If this is not done the system clock is not correct
145   // and the blinking of the led gets slower
146   SysTick_Init();
147 }
148
149 int main(void) {
150   SysTick_Init();
151   LED_Initialize();
152   Button_Initialize();
153
154   uint32_t lastToggle = 0;
155   bool switchedHSI = false;
156
157   while (1) {
158     uint32_t state = Button_GetState();
159
160     if (state & ALARMZONE1_BUTTON) {
161       zone1_active = true;
162       LED_On(ALARMZONE1_LED);
163     }
164
165     if (state & ALARMZONE2_BUTTON) {
166       zone2_active = true;
167       LED_On(ALARMZONE2_LED);
168     }
169
170     if (state & ALARM_OFF_BUTTON) {
171       zone1_active = false;
172       zone2_active = false;
173
174       LED_Off(ALARMZONE1_LED);
175       LED_Off(ALARMZONE2_LED);
176       LED_Off(ALARM_SIREN);
177
178       if (!switchedHSI) {
179         SwitchClockTo24Mhz();
180         switchedHSI = true;
181       }
182     }
183
184     bool alarmActive = (zone1_active || zone2_active);
185     if (alarmActive) {
186       uint32_t now = g_ms;
187       if ((uint32_t)(now - lastToggle) >= SIREN_TOGGLE_MS) {
188         LED_Toggle(ALARM_SIREN);
189         lastToggle = now;
190       }
191     } else {
192       LED_Off(ALARM_SIREN);
193     }
194
195     SysDelay_Delay(1);
196   }
197 }
```

1.3 Fragen

- Was ändert sich mit der neu eingestellten Frequenz? Warum?
Wenn man meinen Code ausfuehrt, wie oben gezeigt, aendert das Aufrufen der SwitchClockTo24Mhz Funktion die Systemclock Frequenz von 48Mhz auf 24Mhz.
Das Blinken der Led wird dadurch nicht beeinflusst, da die SysTick Zeit ebenfalls angepasst wird mittels:
- SystemCoreClockUpdate();
- SysTick_Config(SystemCoreClock / 1000);
Diese Aufrufe aktualisieren die SystemTick Zeit auf 1ms bei 24Mhz.

Wenn diese Aktualisierung nicht gemacht wird, blinkt die LED langsamer, da die SysTick Zeit immer noch auf 1ms bei 48Mhz eingestellt ist.

- Mit welcher Frequenz wird der Prozessor betrieben, wenn das Hauptprogramm gestartet wird?
Wenn der Microcontroller gestartet wird, wird er mit der internen HighSpeed Clock HSI (8Mhz) betrieben. (Hardware default)
In der startup.s wird der Reset Handler aufgerufen, welcher die SystemInit Funktion in system_stm32f0xx.c aufruft.
Diese Funktion setzt ein paar Initalzustand und ruft anschliessend die SetSysClock Funktion auf.
SetSysClock schaltet den High speed external Clock HSE (8Mhz Quarz) ein, konifiguriert den PLL (8Mhz /1 *6 = 48Mhz), schaltet diese ein und wechselt die Systemclock auf die PLL als source.
Ausserdem wird die Flash latency richtig modifiziert, damit der Prozessor bei 48Mhz stabil laeuft.
Macht man das nicht, haengt sich der Microcontroller nach einiger Zeit auf.

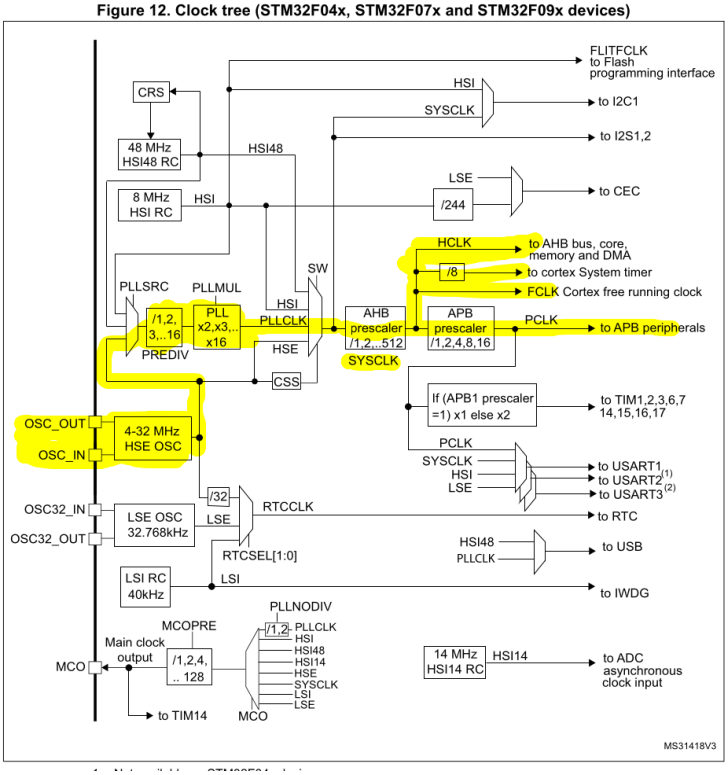

Figure 1: Pfad der Sytemclocks