Mikroprozessorpraktikum

Konstantin Bork & Kean Seng Liew, Gruppe A, HWP8

05-01 Interrupt extern

A05-01.1

> Die Beispielanwendung soll im Kern aus einer leeren Endlosschleife bestehen. In der Ausgangssituation sollen die grüne LED ausgeschaltet sein. Beide Taster an PC8 und PC5 sollen interruptfähig sein - PC8 Taster1 liefert bei eine HL-Flanke einen Interrupt und - PC5 Taster2 liefert bei eine LH-Flanke einen Interrupt. Die notwendige Konfiguration der Register der Portleitungen und die Freigabe des Interrupts müssen vor der Endlosschleife erfolgen. Für die Konfiguration der Portleitungen an denen die Taster 1 und 2 angeschlossen sind wurden ja schon entsprechende Funktionen erstellt. Den Code für die Konfiguration der Interruptfähigkeit, binden Sie bitte in die Funktionen init_taste_1_irq() und init_taste_2_irq() ein.
> Die ISR für den Taster1 an PC8 soll die grüne LED an PB2 einschalten.
> Die ISR für den Taster2 an PC5 soll die grüne LED an PB2 ausschalten.

Auszug aufgabe.h

```
#ifndef __aufgabe_h__
#define __aufgabe_h__

//############################################################################
//########## cmsis_lib include
//############################################################################
#include "stm32f4xx.h"
#include "misc.h"
#include "stm32f4xx_exti.h"
#include "stm32f4xx_gpio.h"
#include "stm32f4xx_iwdg.h"
#include "stm32f4xx_rcc.h"
#include "stm32f4xx_rtc.h"
#include "stm32f4xx_sdio.h"
#include "stm32f4xx_usart.h"
#include "stm32f4xx_wwdg.h"

//############################################################################
//########## mpp_lib include
//############################################################################
#include "global.h"
#include "init.h"
#include "interrupts.h"
#include "led.h"
#include "taster.h"
#include "usart.h"
```

```
//#############################################################################
//########## Eigene Funktionen, Macros und Variablen
//#############################################################################

//=============================================================================
// Macros
//=============================================================================
#define GR_LED_ON        (GPIO_SetBits(GPIOB, GPIO_Pin_2))
#define GR_LED_OFF       (GPIO_ResetBits(GPIOB, GPIO_Pin_2))
#define GR_LED_Toggle    (GPIO_ToggleBits(GPIOB, GPIO_Pin_2))

//=============================================================================
// Funktionen
//=============================================================================
// Aufgabe A01-01
extern void init_leds();

// Aufgabe A01-02
extern void init_taste_1();
extern void init_taste_2();
extern int led_steuerung();

// Aufgabe A05-01
extern void init_taste_1_irq();
extern void init_taste_2_irq();
//=============================================================================
#endif
```

Auszug aufgabe.c

```
#include "aufgabe.h"

// Aufgabe A01-01.3
// Initialisiert die Portleitung der grünen LED und schaltet diese ein
void init_leds() {
    // Setzt GPIO Port auf den Reset Zustand zurück
    GPIO_DeInit(GPIOB);

    // Taktquelle für die Peripherie aktivieren
    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOB, ENABLE);

    // Struct anlegen
    GPIO_InitTypeDef GPIO_InitStructure;

    // Struct Initialisieren setzt alle Leitungen auf
    // Eingang ohne PushPull
    GPIO_StructInit(&GPIO_InitStructure);

    // Die Funktionalität der Portleitungen festlegen

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2;

    // Auswahl des I/O Mode
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT; //GPIO Output Mode

    // Auswahl der Speed
```

```
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz; // Low speed

    // Auswahl des Output Typs
    GPIO_InitStructure.GPIO_OType = GPIO_OType_PP; // PushPull

    // Auswahl des Push/Pull Typs
    GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL; // NoPull

    // PortLeitungen initialisieren
    GPIO_Init(GPIOB, &GPIO_InitStructure);

    // Schaltet die LED ein
    GR_LED_ON;

    // LED wurde initialisiert, LED ausschalten
    GR_LED_OFF;
}

// Aufgabe A01-02.2
// Funktion zur Initialisierung beider Tasten
void init_taste(uint16_t GPIO_Pin) {
    // Setzt GPIO Port auf den Reset Zustand zurück
    GPIO_DeInit(GPIOC);

    // Taktquelle für die Peripherie aktivieren
    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOC, ENABLE);

    // Struct anlegen
    GPIO_InitTypeDef GPIO_InitStructure;

    // Struct Initialisieren setzt alle Leitungen auf
    // Eingang ohne PushPull
    GPIO_StructInit(&GPIO_InitStructure);

    // Die Funktionalität der Portleitungen festlegen

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin;

    // Auswahl des I/O Mode
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN; // GPIO Input Mode

    // Auswahl der Speed
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz; // Low speed

    // Auswahl des Output Typs
    GPIO_InitStructure.GPIO_OType = GPIO_OType_PP; // PushPull

    // Auswahl des Push/Pull Typs
    GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL; // NoPull

    // PortLeitungen initialisieren
    GPIO_Init(GPIOC, &GPIO_InitStructure);
}

// Initialisierung von Taste 1
void init_taste_1() {
    init_taste(GPIO_Pin_8);
}
```

```
// Initialisierung von Taste 2
void init_taste_2() {
    init_taste(GPIO_Pin_5);
}

// Aufgabe A05-01
void init_exti(uint8_t EXTI_PinSource, uint32_t EXTI_Line, EXTITrigger_TypeDef
EXTI_Trigger)
{
    //=============================================================
    //========= Interrupt Konfiguration
    //=============================================================
    // Bindet Port A Leitung 0 an die EXTI_Line0 Leitung
    SYSCFG_EXTILineConfig(EXTI_PortSourceGPIOA, EXTI_PinSource);

    // Struct anlegen
    EXTI_InitTypeDef EXTI_InitStructure;

    // EXTI_Line zweisen
    EXTI_InitStructure.EXTI_Line = EXTI_Line;

    // Interrupt Mode setzen
    EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;

    // Triggerbedingung setzen
    EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger;

    // Interrupt erlauben
    EXTI_InitStructure.EXTI_LineCmd = ENABLE;

    // Regiser aus dem Struct heraus setzen
    EXTI_Init(&EXTI_InitStructure);
}

void init_nvic(uint8_t NVIC_IRQChannel)
{
    //=============================================================
    //========= Interruptcontroller Konfiguration
    //=============================================================

    // Anlegen eines NVIC Struct
    NVIC_InitTypeDef NVIC_InitStructure;

    // Festlegung der Interruptquelle
    NVIC_InitStructure.NVIC_IRQChannel = NVIC_IRQChannel;

    // Festlegung der Priorität entweder in 5 Gruppen
    //=====================================================
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_4);
    //=====================================================

    // oder feiner gegliedert in Priorität und Subpriorität
    //=====================================================
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
    //=====================================================

    // Interruptkanal Freigabe
```

```
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;

    // Register aus dem Struct heraus schreiben
    NVIC_Init(&NVIC_InitStructure);
}

// Initialisiere Taste 1
void init_taste_1_irq()
{
    // Initialisiere GPIO
    init_taste_1();

    // Initialisiere External Interrupt auf EXTI_Line0 mit HL-Flanke gemäß Aufgabe
    init_exti(EXTI_PinSource0, EXTI_Line0, EXTI_Trigger_Falling);

    //
    init_nvic(EXTI0_IRQn);
}

// Initialisiere Taste 2
void init_taste_2_irq()
{
    // Initialisiere GPIO
    init_taste_2();

    // Initialisiere External Interrupt auf EXTI_Line1 mit LH-Flanke gemäß Aufgabe
    init_exti(EXTI_PinSource1, EXTI_Line1, EXTI_Trigger_Rising);

    //
    init_nvic(EXTI1_IRQn);
}
```

Auszug interrupts.c

```
void EXTI9_5_IRQHandler(void)
{

    //SystemInit();
    //===== Taster2
    if (EXTI_GetITStatus(EXTI_Line5) == SET)
        {
            EXTI_ClearFlag(EXTI_Line5);
            EXTI_ClearITPendingBit(EXTI_Line5);
            // Code
            GR_LED_OFF;
        }
    //===== nicht belegt
    if (EXTI_GetITStatus(EXTI_Line6) == SET)
        {
            EXTI_ClearFlag(EXTI_Line6);
            EXTI_ClearITPendingBit(EXTI_Line6);
            usart2_send("EXTI6_IRQn\r\n");
        }
    //===== nicht belegt
    if (EXTI_GetITStatus(EXTI_Line7) == SET)
        {
```

```
            EXTI_ClearFlag(EXTI_Line7);
            EXTI_ClearITPendingBit(EXTI_Line7);
            usart2_send("EXTI7_IRQn\r\n");
        }
    //===== Taster 1
    if (EXTI_GetITStatus(EXTI_Line8) == SET)
        {
            EXTI_ClearFlag(EXTI_Line8);
            EXTI_ClearITPendingBit(EXTI_Line8);
            // Code
            GR_LED_ON;
        }
    //===== nicht belegt
    if (EXTI_GetITStatus(EXTI_Line9) == SET)
        {
            EXTI_ClearFlag(EXTI_Line9);
            EXTI_ClearITPendingBit(EXTI_Line9);
            usart2_send("EXTI9_IRQn\r\n");
        }

}
```

main.c

```
#include "main.h"
#include "aufgabe.h"

int main(void)
{
    // Initialisierung des Systems und des Clocksystems
    SystemInit();

    // SysTick initialisieren
    // jede ms erfolgt dann der Aufruf
    // des Handlers fuer den Interrupt SysTick_IRQn
    InitSysTick();

    // Initialisiere die grüne LED
    init_leds();

    // Initialisiere beide Tasten
    init_taste_1_irq();
    init_taste_2_irq();

    init_nvic();

    while(1)
    {
        // Leere Schleife, da die gesamte Funktionalität in den Interrupt-Handlern
        // vorhanden ist
    }
}
```

Die LED reagiert direkt auf den Druck von Taste 1, anders gesagt, man muss die Taste nicht

loslassen, damit die LED leuchtet.

A05-01.2

> Aufgabenstellung wie in A05-01.1. Allerdings sollen folgende Änderungen eingeführt werden. - PC8 Taster1 liefert bei eine LH-Flanke einen Interrupt und - PC5 Taster2 liefert bei eine HL-Flanke einen Interrupt.
>
> Testen Sie das Verhalten der Taster.

```
#### Auszug aufgabe.c

// Initialisierung von Taste 1
void init_taste_1_irq()
{
    // GPIO Konfiguration von taste_1
    init_taste_1();

    init_exti(EXTI_PinSource8, EXTI_Line8, EXTI_Trigger_Rising);
}

// Initialisierung von Taste 2
void init_taste_2_irq()
{
    init_taste_2();

    init_exti(EXTI_PinSource5, EXTI_Line5, EXTI_Trigger_Falling);
}
```

Der Unterschied zum ersten Programm besteht darin, dass die Interrupt-Auslöser bei beiden Tasten vertauscht sind. Der Interrupt wird erst gesendet, wenn Taste 1 losgelassen wird. Auf den Tastendruck direkt reagiert das Board nicht.

A05-01.3

> Die Aufgabenstellung ist wie in A05-01.1 beschrieben, allerdings soll folgende Änderungen eingeführt werden. - Nach zehnmaliger Betätigung des Taster1 ist dieser nicht mehr interruptfähig. - Erst wenn ab diesem Zeitpunkt der Taster2 zweimal betätigt wurde, ist Taster1 wieder interruptfähig. - Geben Sie bei jeder Tastenbetätigung die betätigte Taste und die Anzahl der Tastenbetätigungen beider Tasten auf der USART aus.
>
> Beachten Sie, daß die Taste1 nie länger als 2 Sekunden gedrückte wird da sonst die Stromversorgung Ausschaltet wird.

Auszug aufgabe.c

```
void deaktiviert_exti()
{
        //=============================================================
        //========= Interrupt Konfiguration
        //=============================================================
        // Bindet Port C Leitung 8 an die EXTI_Line8 Leitung
        SYSCFG_EXTILineConfig(EXTI_PortSourceGPIOC, EXTI_PinSource8);

        // Struct anlegen
        EXTI_InitTypeDef EXTI_InitStructure;

        // EXTI_Line zuweisen
        EXTI_InitStructure.EXTI_Line = EXTI_Line8;

        // Interrupt Mode setzen
        EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;

        // Triggerbedingung setzen
        EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling;

        // Interrupt verbieten
        EXTI_InitStructure.EXTI_LineCmd = DISABLE;

        // Register aus dem Struct heraus setzen
        EXTI_Init(&EXTI_InitStructure);

        EXTI_ClearFlag(EXTI_Line8);
        EXTI_ClearITPendingBit(EXTI_Line8);
}
```

Auszug interrupts.c

```
void EXTI9_5_IRQHandler(void)
{
    static unsigned int taste_1_gedrueckt = 0;
    static unsigned int taste_2_gedrueckt = 0;
    static unsigned char interrupt_deaktiviert = 0;

    //SystemInit();
    //===== Taster2
    if (EXTI_GetITStatus(EXTI_Line5) == SET)
        {
            EXTI_ClearFlag(EXTI_Line5);
            EXTI_ClearITPendingBit(EXTI_Line5);
            // Code
            GR_LED_OFF;

            // Nur wenn der Interrupt von Taste 1 deaktiviert ist und Taste 2
gedrückt wurde,
            // führe die nächsten Schritte aus
            if(interrupt_deaktiviert == 1)
            {
                taste_2_gedrueckt++;
```

```
                // Ausgabe der Betätigungen von Taste 2
                sprintf(usart2_tx_buffer, "\r\nTaster 2: %d\r\n",
taste_2_gedrueckt);
                usart_2_print(usart2_tx_buffer);

                if(taste_2_gedrueckt == 2)
                {
                    // Initialisiere wieder den Interrupt, da Taste 2 zweimal
gedrückt wurde
                    init_taste_1_irq();

                    // Reset aller Werte
                    taste_1_gedrueckt = 0;
                    taste_2_gedrueckt = 0;
                    interrupt_deaktiviert = 0;
                }
            }
        }
    //===== nicht belegt
    if (EXTI_GetITStatus(EXTI_Line6) == SET)
        {
            EXTI_ClearFlag(EXTI_Line6);
            EXTI_ClearITPendingBit(EXTI_Line6);
            usart2_send("EXTI6_IRQn\r\n");
        }
    //===== nicht belegt
    if (EXTI_GetITStatus(EXTI_Line7) == SET)
        {
            EXTI_ClearFlag(EXTI_Line7);
            EXTI_ClearITPendingBit(EXTI_Line7);
            usart2_send("EXTI7_IRQn\r\n");
        }
    //===== Taster 1
    if (EXTI_GetITStatus(EXTI_Line8) == SET)
        {
            EXTI_ClearFlag(EXTI_Line8);
            EXTI_ClearITPendingBit(EXTI_Line8);
            // Code
            GR_LED_ON;
            taste_1_gedrueckt++;

            // Ausgabe der aktuellen Betätigungen von Taste 1
            sprintf(usart2_tx_buffer, "\r\nTaster 1: %d\r\n", taste_1_gedrueckt);
            usart_2_print(usart2_tx_buffer);

            // Wenn Taste 1 zehnmal betätigt wurde, deaktiviere den Interrupt
            if(taste_1_gedrueckt == 10)
            {
                deaktiviert_exti();
                interrupt_deaktiviert = 1;
            }
        }
    //===== nicht belegt
    if (EXTI_GetITStatus(EXTI_Line9) == SET)
        {
            EXTI_ClearFlag(EXTI_Line9);
            EXTI_ClearITPendingBit(EXTI_Line9);
            usart2_send("EXTI9_IRQn\r\n");
```

```
        }
}
```

Der restliche Code ist identisch mit dem in Aufgabe 5.1.1 gezeigten Code, insbesondere die main.c und die Initialisierungsmethoden für beide Tasten.