Mikroprozessorpraktikum

Konstantin Bork, Kean Seng Liew, & Oliver Stein, Gruppe A, HWP8

01-01 GPIO Ausgang

A01-01.1

1. PMOS und NMOS steuern den Output des Prozessors.
2. Der PMOS ist aktiviert und der Output ist somit HIGH.
3. Der NMOS ist aktiviert und der Output ist somit LOW.
4. Nur der NMOS steuert den Output des Prozessors.
5. Der NMOS ist deaktiviert und der Output ist HIGH.
6. Der NMOS ist aktiviert und der Output ist LOW.

A01-01.2

Am besten eignet sich Push/Pull mit NoPull, weil man damit den Output des Prozessors steuern kann.

A01-01.3

aufgabe.h

```
#ifndef __aufgabe_h__
#define __aufgabe_h__

//############################################################################
//########## cmsis_lib include
//############################################################################
#include "stm32f4xx_gpio.h"
#include "stm32f4xx_rcc.h"

//############################################################################
//########## mpp_lib include
//############################################################################
#include "led.h"

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

//============================================================================
// Macros
//============================================================================

//============================================================================
// Variablen
//============================================================================

//============================================================================
```

```
// Funktionen
//=============================================================================
extern void init_leds();

//=============================================================================
#endif
```

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
    GPIO_SetBits(GPIOB, GPIO_Pin_2);

    // LED wurde initialisiert, LED ausschalten
    GPIO_ResetBits(GPIOB, GPIO_Pin_2);
}
```

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();

    // Initialisierung aller Portleitungen und Schnittstellen
    // Freigabe von Interrupten
    init_board();

    // Start der RTC  falls diese noch
    // nicht initialisiert war wird
    // die RTC mit der LSE-Taktquelle aktiviert
    start_RTC();

    // Initialisiere die grüne LED
    init_leds();

    while(1)
    {
        wait_mSek(500);
    }
}
```

A01-01.4

aufgabe.h

```
#ifndef __aufgabe_h__
#define __aufgabe_h__

//#############################################################################
//########## cmsis_lib include
//#############################################################################
#include "stm32f4xx_gpio.h"
#include "stm32f4xx_rcc.h"

//#############################################################################
//########## mpp_lib include
//#############################################################################
#include "led.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))

//===========================================================================
// Variablen
//===========================================================================

//===========================================================================
// Funktionen
//===========================================================================
extern void init_leds();

//===========================================================================
#endif
```

aufgabe.c

```
#include "aufgabe.h"
#include "test.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
    GPIO_SetBits(GPIOB, GPIO_Pin_2);

    // LED wurde initialisiert, LED ausschalten
    GPIO_ResetBits(GPIOB, GPIO_Pin_2);
}
```

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();

    // Initialisierung aller Portleitungen und Schnittstellen
    // Freigabe von Interrupten
    init_board();

    // Start der RTC  falls diese noch
    // nicht initialisiert war wird
    // die RTC mit der LSE-Taktquelle aktiviert
    start_RTC();

    // Initialisiere die grüne LED
    init_leds();

    while(1)
    {
        wait_mSek(500);
        GR_LED_ON;
        wait_mSek(500);
        GR_LED_OFF;
    }
}
```