Mikroprozessorpraktikum

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

02-01 Taktfrequenz

A02-01.1

> Machen Sie sich mit der Konfiguration der Portleitung PC09 und der Nutzung der Alternativfunktion vertraut. Zielstellung ist die Portleitung so zu konfigurieren, daß die SYSCLK Taktquelle über das MCO2 Pin an PC09 ausgegeben wird. Dazu finden Sie Informationen an folgenden Stellen: - Einführungstext zum Aufgabenkomplex - Referenz Manual - I/O Pin Multiplexer - Referenz Manual - MCO2
>
> Erstellen Sie dazu die notwendige Sequenz an Codezeilen innerhalb der Funktion init_PC09(). Kommentieren Sie jede Quellcodezeile.

Auszug aufgabe.h

```
// Aufgabe A02-01.1
// Initialisierung von Portleitung 9 für die SYSCLK
void init_PC09() {
    // 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);

    // Benutze Pin 9 gemäß Aufgabe
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;

    //
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;

    // Setze den Modus auf Push/Pull
    GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;

    // Verwende NoPull
    GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;

    // Setze die Frequenz des Ports auf 50 MHz
```

```
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;

    // Initialisiere den GPIO Port
    GPIO_Init(GPIOC, &GPIO_InitStructure);

    //
    GPIO_PinAFConfig(GPIOC, GPIO_Pin_9, GPIO_AF_MCO);

    //
    RCC_MCO2Config(RCC_MCO2Source_SYSCLK, RCC_MCO2Div_1);
}
```

A02-01.2

> Bis auf den Funktionsaufruf SystemInit() in der main() müssen alle Funktionsaufrufe im Startprojekt auskommentiert werden. Fügen Sie vor der Endlosschleife den Funktionsaufruf init_PC09() aus der Aufgabe A02-01.1 ein. Dadurch wird an der Portleitung PC09 das SYSCLK-Taktsignal ausgegeben. Lassen Sie auf Basis der Funktion wait_uSek(unsigned long us) die grüne LED im Sekundentakt in einer Endlosschleife blinken. Vergessen Sie nicht die LED Portleitung entsprechend zu initialisieren. Bestimmen Sie mit dem Oszillographen und mit dem Counter die Taktfrequenz an der Portleitung PC09. Notieren Sie sich den Stromverbrauch des Boards um ihn mit dem Verbrauch in der folgenden Aufgabe zu vergleichen.

aufgabe.h

```
#ifndef __aufgabe_h__
#define __aufgabe_h__

//############################################################################
//########## cmsis_lib include
//############################################################################
#include "stm32f4xx.h"
#include "stm32f4xx_gpio.h"h"
#include "stm32f4xx_rcc.h"

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

#define SEC_IN_USEC       1000000

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

//============================================================================
// 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 A02-01
extern void init_PC09();

//============================================================================
#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
    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 f�r 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 A01-02.3
// Programm zum Kontrollieren der grünen LED mit den beiden Tasten
int led_steuerung()
{
    uint8_t     Byte = 0;

    Byte = GPIO_ReadInputDataBit(GPIOC, GPIO_Pin_5);

    if(Byte == Bit_SET)
    {
        return 1;
    }

    Byte = GPIO_ReadInputDataBit(GPIOC, GPIO_Pin_8);

    if(Byte != Bit_SET)
    {
        return -1;
    }

    return 0;
}

// Aufgabe A02-01.1
// Initialisierung von Portleitung 9 für die SYSCLK
void init_PC09() {
    // 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);

    // Benutze Pin 9 gemäß Aufgabe
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;

    //
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;

    // Setze den Modus auf Push/Pull
    GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;

    // Verwende NoPull
    GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;

    // Setze die Frequenz des Ports auf 50 MHz
```

```
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;

    // Initialisiere den GPIO Port
    GPIO_Init(GPIOC, &GPIO_InitStructure);

    //
    GPIO_PinAFConfig(GPIOC, GPIO_Pin_9, GPIO_AF_MCO);

    //
    RCC_MCO2Config(RCC_MCO2Source_SYSCLK, RCC_MCO2Div_1);
}
```

main.c

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

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

    // Initialisiere die grüne LED
    init_leds();

    // Initialisiere PC09 für die SYSCLK
    init_PC09();

    while(1)
    {
        GR_LED_ON;
        wait_uSek(SEC_IN_USEC);
        GR_LED_OFF;
        wait_uSek(SEC_IN_USEC);
    }
}
```

Es liegt Strom mit einer Stärke von 75 mA bei einer Spannung von 4V an. Somit hat das Board eine Leistung von 0,3W.

A02-01.3

> Kommentieren Sie nun den Funktionsaufruf SystemInit() aus. Laden Sie das Programm erneut und stellen Sie fest mit welcher Taktfrequenz der Mikrocontroller jetzt läuft. In diesem Fall läuft das System jetzt auf Basis des HSI mit 16MHz. Die Taktfrequenz SYSCLK kann wieder an der Portleitung PC09 ermittelt werden. Wie verhält sich jetzt die LED und wie hat sich der Stromverbrauch des Boards geändert? Erklären Sie die Beobachtung.

Die LED blinkt jetzt in einem längeren Zeitraum, wir haben 10 Sekunden gemessen statt 1 Sekunde.

Dafür sinkt die Stromstärke auf 30 mA und das Board hat somit eine Leistung von 0,12W.

A02-01.4

In dieser Aufgabe soll der Stromverbrauch und die Taktfrequenz des STM32 für unterschiedliche Modi ermittelt werden. Bis auf den Funktionsaufruf SystemInit() in der main() müssen alle Funktionsaufrufe im Startprojekt auskommentiert werden. Dazu muß vor der while(1) Schleife der Aufruf der folgenden Funktionen erfolgen.

```
SystemInit();
init_PC09();
init_taste_1();
init_taste_2();
```

Innerhalb der Schleife soll per Tastendruck der Aufruf folgender Funktionen erfolgen.

```
Taste 1 gedrückt ruft slowMode() auf.
Taste 2 gedrückt ruft fastMode() auf.
```

Der Quellcode dieser Funktionen muß in das Projekt aufgenommen werden.
Messen Sie für jeden Zustand den Stromverbrauch des Mikrocontrollers und die Taktfrequenz (SYSCLK).
Im nächsten Schritt wird der Funktionsaufruf init_PC09() entfernt und die gleiche Messung wiederholt. Vergleichen Sie die Messergebnisse. Welche Schlussfolgerung für die Programmierung einer Anwendung unter dem Gesichtspunkt des Energieverbrauchs ziehen Sie aus der Beobachtung?
Informativ sollten Sie sich an dieser Stelle einen Überblick über den Strombedarf der einzelnen Periperieeinheiten und dessen Abhängigkeit von der Taktfrequenz anhand des Datenblatts verschaffen.

aufgabe.h

```
#ifndef __aufgabe_h__
#define __aufgabe_h__

//############################################################################
//########## cmsis_lib include
//############################################################################
#include "stm32f4xx.h"
#include "stm32f4xx_gpio.h"
#include "stm32f4xx_rcc.h"
```

```
//#############################################################################
//########## mpp_lib include
//#############################################################################
#include "led.h"
#include "taster.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))

#define SEC_IN_USEC       1000000

//=============================================================================
// 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 A02-01
extern void init_PC09();
extern void fastMode();
extern void slowMode();

//=============================================================================
#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
    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 A01-02.3
// Programm zum Kontrollieren der grünen LED mit den beiden Tasten
int led_steuerung()
{
    uint8_t     Byte = 0;

    Byte = GPIO_ReadInputDataBit(GPIOC, GPIO_Pin_5);

    if(Byte == Bit_SET)
    {
        return 1;
    }

    Byte = GPIO_ReadInputDataBit(GPIOC, GPIO_Pin_8);

    if(Byte != Bit_SET)
    {
        return -1;
    }

    return 0;
}

// Aufgabe A02-01.1
// Initialisierung von Portleitung 9 für die SYSCLK
void init_PC09() {
    // 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);
```

```
    // Benutze Pin 9 gemäß Aufgabe
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;

    // Setze den Modus auf Alternate Function, da wir keine "richtigen" I/O-
Aufgaben lösen wollen
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;

    // Setze den Output Typ auf Push/Pull
    GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;

    // Verwende PullUp
    GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;

    // Setze die Frequenz des Ports auf 50 MHz
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;

    // Initialisiere den GPIO Port
    GPIO_Init(GPIOC, &GPIO_InitStructure);

    // Konfiguriert PC09 für den Alternate Function Modus, verbindet MCO mit PC09
    GPIO_PinAFConfig(GPIOC, GPIO_Pin_9, GPIO_AF_MCO);

    // Konfiguriert MCO2 und PC09 so, dass die SYSCLK Taktquelle über MCO2 an PC09
ausgegeben wird
    RCC_MCO2Config(RCC_MCO2Source_SYSCLK, RCC_MCO2Div_1);
}

// Aufgabe A02-01.4
// PLLStartUp Hilfsfunktion
void RCC_WaitForPLLStartUp(void) {
    while ( (RCC->CR & RCC_CR_PLLRDY) == 0 ) {
        __NOP();
    }
}

//==== Taktfrequenz 24MHz mit HSE-OSC=16MHz
void slowMode(void) {
    RCC_DeInit();

    RCC_HSEConfig(RCC_HSE_ON);
    if (RCC_WaitForHSEStartUp() == ERROR) {
        return;
    }
    // HSEOSC=16MHz SYSCLK=24MHz HCLK=24MHz
    // PCLK1=24 PCLK2=24MHz
    RCC_PLLConfig(RCC_PLLSource_HSE,    //RCC_PLLSource
                  16,     //PLLM
                  192,    //PLLN
                  8,      //PLLP
                  4       //PLLQ
    );
    RCC_PLLCmd(ENABLE);
    RCC_WaitForPLLStartUp();

    // Configures the AHB clock (HCLK)
    RCC_HCLKConfig(RCC_SYSCLK_Div1);
    // Low Speed APB clock (PCLK1)
    RCC_PCLK1Config(RCC_HCLK_Div1);
```

```
    // High Speed APB clock (PCLK2)
    RCC_PCLK2Config(RCC_HCLK_Div1);

    // select system clock source
    RCC_SYSCLKConfig(RCC_SYSCLKSource_PLLCLK);
}

//==== Taktfrequenz 168MHz mit HSE-OSC=16MHz
void fastMode(void) {
    RCC_DeInit();

    RCC_HSEConfig(RCC_HSE_ON);
    if (RCC_WaitForHSEStartUp() == ERROR) {
        return;
    }
    // HSEOSC=16MHz SYSCLK=168MHz HCLK=168MHz
    // PCLK1=42MHz PCLK2=84MHz
    RCC_PLLConfig(RCC_PLLSource_HSE,    //RCC_PLLSource
                  16,     //PLLM
                  336,    //PLLN
                  2,      //PLLP
                  7       //PLLQ
    );
    RCC_PLLCmd(ENABLE);
    RCC_WaitForPLLStartUp();

    // Configures the AHB clock (HCLK)
    RCC_HCLKConfig(RCC_SYSCLK_Div1);
    // High Speed APB clock (PCLK1)
    RCC_PCLK1Config(RCC_HCLK_Div4);
    // High Speed APB clock (PCLK2)
    RCC_PCLK2Config(RCC_HCLK_Div2);


    // select system clock source
    RCC_SYSCLKConfig(RCC_SYSCLKSource_PLLCLK);
}
```

main.c

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

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

    // Initialisiere beide Tasten
    init_taste_1();
    init_taste_2();

    // Initialisiere PC09 für die SYSCLK
    init_PC09();

    while(1)
```

```
    {
        int taste_2_gedrueckt = led_steuerung();

        if(taste_2_gedrueckt == 1) {
            fastMode();
        } else if (taste_2_gedrueckt == -1) {
            slowMode();
        }
    }
}
```

Die Eingangsspannung beträgt immer 4V. Folgende Werte bei der Stromstärke und der Taktfrequenz haben wir gemessen: - Mit init_PC09() haben wir im Slowmode 28mA bei 24MHz Takt gemessen, folglich hat das Board eine Leistung von 112mW. Im Fastmode haben wir 64mA bei 168 MHz gemessen und somit eine Leistung von 272mW. - Ohne init_PC09() haben wir im Slowmode 27mA bei 24MHz Takt gemessen, folglich hat das Board eine Leistung von 108mW. Im Fastmode haben wir 58mA bei 168 MHz gemessen und somit eine Leistung von 232mW.

Wenn der Energieverbrauch wichtig ist und die Rechenleistung nicht so wichtik, kann man eine geringe Taktfrequenz verwenden und sollte MCO2 nicht als Taktquelle verwenden. Wenn der Energieverbrauch egal ist oder man eine zeitkritische Anwendung hat, sollte man eine hohe Taktfrequenz und MCO2 als Taktquelle verwenden.