



Praktikum Mikrocomputertechnik V1

2018

Das MCT-Praktikum basiert auf dem Board STM32F (CortexM3 Architektur), das als Beispiel in der Vorlesung verwendet wurde. Im ersten Termin wird die General Purpose I/O (GPIO) und der SystemTickTimer verwendet, die ersten Programme werden in Assembler codiert.

Aufgabe 1_1: Lauflicht

An PortC (0-7) werden 8 Schalter, an PortB (8-15) 8 LEDs angeschlossen. In der ersten Aufgabe wird ein Lauflicht programmiert, die Geschwindigkeit wird durch die Funktion wait realisiert. Analysieren Sie das vorgegebene Programm V1.1_Lauflicht und lassen Sie dieses Programm im Debugger ablaufen.

Ablauf:

- In der Reset and Control Logik (RCC) muss jeder I/O- Baustein separat angeschaltet (enabled) werden.
Initialisierung RCC , Enable von PortB und PortC. Die Basisadresse des RCC-Registerblocks ist 0x40021000.
- Initialisierung des PortB (8-15) als Ausgang (General Purpose Push Pull, 50 MHz)
- Initialisierung des PortC (0-7) als Eingang (Floating Input)
Die Basisadresse des PortB-Blocks ist 0x40010C00, Basisadresse PortC ist 0x40011000.
- In einer Endlosschleife wird das Lauflicht ausgegeben

Ergänzen Sie nun das Programm so, dass das Lauflicht abhängig von der Schalterstellung langsamer oder schneller läuft. Verwenden Sie dazu beim Aufruf der Funktion wait einen Parameter, der die Länge der Wartezeit bestimmt. Initialisieren Sie den PortC (0-7) als Eingang (Floating Input), die verwendeten Schalter müssen ausmakiert werden.

Realisieren Sie 4 verschiedene Geschwindigkeiten.

Option: Verwenden Sie nun einen weiteren Schalter, um das Lauflicht vorwärts oder rückwärts laufen zu lassen.



Aufgabe 1_2 : Lauflicht mit SysTickTimer-Interrupt

Verwenden Sie nun den SysTickTimer, um die Geschwindigkeit des Lauflichts zu steuern. Vom SysTickTimer wird regelmäßig nach Ablauf des Timers ein Interrupt ausgelöst. Die Ausgabe an den PortB erfolgt nun in der Interruptroutine.

Verwenden Sie das Programmbeispiel V1.2_SysTickInt und ergänzen Sie es entsprechend.

Hinweis:

Das Beispielprojekt besteht aus zwei Programmteilen. Im Modul InitIO.s stehen die Unterprogramme Init_GPIO und Init_SysTick. Init_GPIO konfiguriert PortB und PortC, Init_SysTick soll von Ihnen so ergänzt werden, dass der SysTickTimer im Sekundenrhythmus einen Interrupt erzeugt. Die entsprechende Interruptroutine steht im Modul main.s und hat den Namen SysTick_Handler. Ergänzen Sie die Interruptroutine. Importieren Sie den SysTickhandler im das Modul STM32Init.s und tragen Sie den Systickhandler an der richtigen Position in die Vektortabelle (__Vectors) ein

Aufgabe 1_3 : Programm Lauflicht in C

Die Aufgabenstellung aus Aufgabe 1_1 (Lauflicht) soll nun mit der Programmiersprache C umgesetzt werden.

Setzen Sie nun Aufgabenstellung aus Aufgabe 1_1 (Lauflicht) und Aufgabe 1_2 (Lauflicht mit SysTickTimer) mit der Programmiersprache C um. Dazu muss mit der C-Syntax auf die absoluten Adressen der I/O-Register zugegriffen werden. Laden Sie dazu das Projekt „V1.3 Lauflicht_C“ und analysieren Sie die Typ-Definitionen und die #define-Anweisungen.

Beantworten Sie folgende Fragen:

- Welche I/O-Register werden durch die Typdefinition GPIO_TypeDef abgebildet?
- Welchen Wert hat die Konstante GPIOB_BASE?
- Welche Bedeutung erhält GPIOB durch die Anweisung #define GPIOB ((GPIO_TypeDef *) GPIOB_BASE)?
- Was passiert bei dieser C-Anweisung: `GPIOB->CRH = 0x33333333;`

Vervollständigen Sie nun das Programm V1.3 Lauflicht_C.



Aufgabe 1_4 : Lauflicht in C mit SysTickTimer -Interrupt

In der Aufgabe 1_2 wurde das Lauflicht mit Hilfe des SysTickTimer-Interrupts realisiert. Auch diese Aufgabenstellung soll nun in C programmiert werden. Legen Sie die Typdefinitionen für den Zugriff auf die I/O-Bausteine in einer eigenen Headerdatei an (ownIO.h).

Hinweis :

Die Interruptservice Routine muss folgenden Aufbau haben:

```
void SysTick_Handler(void)
{
    .....
}
```

Tipp: Testen Sie zuerst, ob der Interrupt überhaupt ausgelöst wird, indem Sie in den SysTick_Handler einen Breakpoint setzen. Wenn der Breakpoint erreicht wird, ist sichergestellt, dass die Interruptauslösung richtig funktioniert!

Der Name SysTick_Handler ist in der Vektortabelle im Modul „startup_stm32f10x_md.s“ bereits vordefiniert und wird vom Compiler entsprechend erkannt.