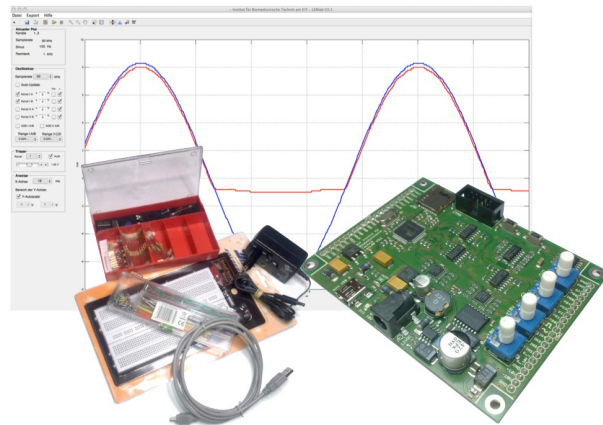


Kurs 4  
**Digitale Signalverarbeitung**



**Gruppe 89**

Vorname	Nachname	Matrikel-Nr.	u-Account	E-Mail
Max	Themistocli	2159422	uztld	uztld@student.kit.edu
Jan	Rösler	2096188	uclrb	uclrb@student.kit.edu
David	Ackermann	2121813	uepar	uepar@student.kit.edu

9. November 2018

## Inhaltsverzeichnis

<b>1</b>	<b>Interrupts</b>	<b>3</b>
<b>2</b>	<b>Diskretisierung</b>	<b>3</b>
<b>3</b>	<b>Erstes Programm</b>	<b>3</b>
3.1	Blinklicht . . . . .	3
3.2	RGB LED . . . . .	3
<b>4</b>	<b>Verwenden des TivaWare-Framework</b>	<b>5</b>
4.1	GPIO-Blinklicht . . . . .	5
4.2	Externe LED Schaltung . . . . .	5
<b>5</b>	<b>Warteschleifen und Taktfrequenz</b>	<b>5</b>
<b>6</b>	<b>Digitale Eingänge</b>	<b>5</b>
6.1	Taster . . . . .	5
6.2	Externer Interrupt . . . . .	5
<b>7</b>	<b>Analog-Digital-Umsetzer</b>	<b>5</b>
7.1	A/D-Umsetzer . . . . .	5
7.2	Widerstandsmessung . . . . .	5
<b>8</b>	<b>Lautstärkepegel</b>	<b>5</b>
8.1	Lautstärkepegel . . . . .	5
8.2	Eigene Verbesserungen . . . . .	5

## Abbildungsverzeichnis

## Tabellenverzeichnis

---

# 1 Interrupts

Ein Interrupt im Allgemeinen ist eine Unterbrechung eines laufenden Prozesses um zeitkritische Prozesse direkt abarbeiten zu können. Sie sind nicht Bestandteil des laufenden Programmes, sondern werden durch asynchrone, externe Ereignisse ausgelöst und benötigen eine extra Hardware. So wird beispielsweise bei einem Druck einer Taste einer Tastatur ein Interrupt ausgelöst um den jeweiligen Befehl direkt ausführen zu können. Eine Sonderklasse der Interrupts sind die non maskable Interrupts, kurz NMI. Diese sind höhergestellt als normale Interrupts und werden mit höchster Priorität direkt abgearbeitet und können nicht von dem Prozessor abgelehnt werden. Im Falle eines Stromausfalles greifen diese ein und sichern wichtige Daten bevor es zum Absturz kommt. Grundsätzlich unterscheidet man Interrupts weiter in Hardware, Software, präzise und unpräzise Interrupts. Hardware Interrupts sind hierbei Interrupts die von einer angeschlossenen Hardware an die CPU weitergeleitet werden. Dem entgegen stehen die Software Interrupts, welche wie ein Hardware Interrupt wirken, jedoch von einem Programm aufgerufen werden. Der Ablauf von Interruptzyklen sieht folgendermaßen aus: Wenn kein NMI vorliegt wird bei einem Interrupt bis Ende des Befehls gewartet. Nach abarbeiten des Befehls wird der Interruptzyklus der CPU gestartet. Hierbei liest der Datenbus den Interruptvektor und sperrt den maskierten Interrupteingang um ein überschneiden mehrerer Interrupts zu verhindern. Um im Anschluss wieder nahtlos das Programm fortführen zu können speichert die CPU den Befehlszähler im Stack ab. Mit Hilfe des Interruptvektors und einer Interrupttabelle kann der Interruptzeiger bestimmt werden. Anschließend läuft die Interrupt-Service-Routine ab. Hierbei ist zu beachten, dass alle Daten des vorherigen Programms zuvor in den Stack kopiert werden und nach Ablauf des Programms ebenfalls wieder zurück kopiert werden.

## 2 Diskretisierung

Ein Musiksignal wird mit einer Frequenz von 44,1 kHz abgetastet und hat eine Auflösung von 16 bit. Mit Gleichung 1 benötigt ein 10 s Musiksignal 7,056 Mbit Speicherplatz.

$$44,1\text{kHz} \cdot 10\text{s} \cdot 16\text{bit} = 7,056\text{Mbit} \quad (1)$$

## 3 Erstes Programm

### 3.1 Blinklicht

### 3.2 RGB LED

#### 3.2.1

Um die Hexadecimalzahl zu einer Binärzahl umzuwandeln wird diese durch 2 geteilt und der Rest notiert bis Null durch Zwei geteilt wird. Der Rest der ersten Division entspricht dem least

significant bit (LSB) und der Rest der letzten Division entspricht dem most significant bit (MSB). Dieser Vorgang wird in Gleichung 2 verdeutlicht.

$$\frac{4}{2} = 2, R = 0; \quad \frac{2}{2} = 1, R = 0; \quad \frac{1}{2} = 0, R = 1 \quad \rightarrow 0x04 = 0b100 \quad (2)$$

Port F hat eine Basis-Adresse von 0x4002.5000 und das Direction Register hat eine Offset Adresse von 0x400. Das setzen eines bits in dem Direction Register konfiguriert den jeweiligen Pin als einen Output. Die Addition der Basis- und Offset-Adressen führt zu einer Adresse bei der nun die Pins für Port F konfiguriert werden können. Durch das beschreiben dieses Registers mit 0x04, wird Pin PF2, die blaue LED, als Output konfiguriert.

#### 3.2.2

Sodass die grüne LED blinkt, muss Port PF3 als Output konfiguriert werden. Wie zuvor müssen die Offset- und Basis-Adresse addiert werden und darauf hin Pin PF3 als output definiert werden. PF3 entspricht bit Nummer 3 und kann mit dem Wert 0b00001000 beschrieben werden um dies zu bewirken. Der equivalente Hexadezimalwert wäre 0x08.

#### 3.2.3

Zurzeit wird nur die LED an PF2 zum blinken gebracht, indem der Inhalt von Register GPIO DATA auf 0x04 gesetzt wird. Um die LED an PF3 zum blinken zu bringen muss, wie in der letzten Aufgabe gezeigt, der Inhalt des GPIO DATA Registers auf 0x08 gesetzt werden. Sodass beide gleichzeitig an gehen, müssen die oben genannten Inhalte addiert werden. Dies würde einem Wert von 0x0C oder 0b00001100 entsprechen. Der Hexadezimalcode ist hier kürzer, aber bei dem Binären Code kann man sofort sehen welche bits gesetzt sind und man kann diese sofort ändern ohne Berechnungen durchführen zu müssen, wenn man ein zusätzliches bit setzen bzw. löschen will.

---

## **4 Verwenden des TivaWare-Framework**

### **4.1 GPIO-Blinklicht**

### **4.2 Externe LED Schaltung**

## **5 Warteschleifen und Taktfrequenz**

## **6 Digitale Eingänge**

### **6.1 Taster**

### **6.2 Externer Interrupt**

## **7 Analog-Digital-Umsetzer**

### **7.1 A/D-Umsetzer**

### **7.2 Widerstandsmessung**

## **8 Lautstärkepegel**

### **8.1 Lautstärkepegel**

### **8.2 Eigene Verbesserungen**