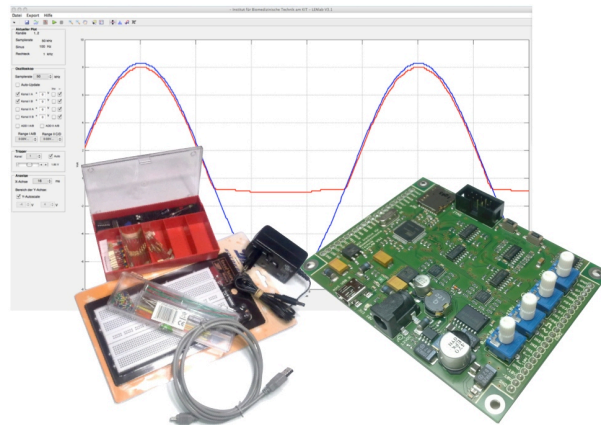


Kurs 4

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

17. November 2018

Inhaltsverzeichnis

1	Interrupts	3
2	Diskretisierung	3
3	Erstes Programm	3
3.1	Blinklicht	3
3.2	RGB LED	3
4	Verwenden des TivaWare-Framework	4
4.1	GPIO-Blinklicht	4
4.2	Externe LED Schaltung	5
5	Warteschleifen und Taktfrequenz	8
6	Digitale Eingänge	11
6.1	Taster	11
6.2	Externer Interrupt	14
7	Analog-Digital-Umsetzer	16
7.1	A/D-Umsetzer	16
7.2	Widerstandsmessung	16
8	Lautstärkepegel	16
8.1	Lautstärkepegel	16
8.2	Eigene Verbesserungen	16

Abbildungsverzeichnis

Tabellenverzeichnis

1	Flimmerverschmelzungsfrequenz	10
---	---	----

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 äquivalente 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

```
1 #include <stdint.h>
2 #include <stdbool.h> // definition of type "bool"
3 #include "inc/hw_memmap.h" // definition of memory addresses
4 #include "inc/hw_types.h" // definition of framework macros
5 #include "driverlib/gpio.h"
6 #include "driverlib/sysctl.h"
7
8 // Delay-Funktion
9
10 void delay(void)
```

```

11 {
12     uint32_t i=80000;
13     while(i) {i--;}
14 }
15
16
17 int main(void){
18     SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);           //Port F aktivieren
19     GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1);    //Pin 1 bis 3 ←
        initialisieren
20     GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_2);
21     GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_3);
22
23     while(1)         //unendliche Schleife fuer flimmern der LED in weiss
24     {
25         delay();
26         GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, 0xFF);
27         GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 0xFF);
28         GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_3, 0xFF);
29         delay();
30         GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, 0x00);
31         GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 0x00);
32         GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_3, 0x00);
33
34
35     }
36     return 0;
37 }

```

4.2 Externe LED Schaltung

4.2.1

```

1  /**
2   * main.c
3   */
4  #include <stdint.h>
5  #include <stdbool.h>           // definition ←
        of type "bool"
6  #include "inc/hw_memmap.h"    // definition ←
        of memory addresses
7  #include "inc/hw_types.h"     // definition ←
        of framework makros
8  #include "driverlib/gpio.h"
9  #include "driverlib/sysctl.h"
10
11 void delay(void)
12 {
13     uint32_t i=100000;

```

```

14  while(i) {i--;}
15  }
16
17
18  int main(void){
19      SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);           // GPIO Port B ←
20      aktivieren
21      GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_0);    // GPIO Port B ←
22      Pin 0–7 aktivieren
23      GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_1);
24      GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_2);
25      GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_3);
26      GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_4);
27      GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_5);
28      GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_6);
29      GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_7);
30
31      while(1)
32      {
33          delay();
34          GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_0, 0xFF);    // GPIO Pins ←
35          auf High stellen
36          GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_1, 0xFF);
37          GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_2, 0xFF);
38          GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_3, 0xFF);
39          GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_4, 0xFF);
40          GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_5, 0xFF);
41          GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_6, 0xFF);
42          GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_7, 0xFF);
43          delay();
44          GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_0, 0x00);    // GPIO Pins ←
45          auf Low stellen
46          GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_1, 0x00);
47          GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_2, 0x00);
48          GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_3, 0x00);
49          GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_4, 0x00);
50          GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_5, 0x00);
51          GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_6, 0x00);
52          GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_7, 0x00);
53      }
54      return 0;
55  }

```

4.2.2

```

1
2 /**
3  * main.c

```

```

4  */
5  #include <stdint.h>
6  #include <stdbool.h>                                // definition ←
7  #include "inc/hw_memmap.h"                          // definition ←
8  #include "inc/hw_types.h"                          // definition ←
9  #include "driverlib/gpio.h"
10 #include "driverlib/sysctl.h"
11
12 void delay(void)                                    // Blinklicht ←
13 {
14     uint32_t i=1000;
15     while(i) {i--;}
16 }
17
18 void delayPWM(void)                                // PWM Delay (←
19 {
20     uint32_t i=8000;
21     while(i) {i--;}
22 }
23
24
25 int main(void){
26     SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);    // GPIO Port B ←
27     GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_0); // GPIO Port B ←
28     GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_1);
29     GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_2);
30     GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_3);
31     GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_4);
32     GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_5);
33     GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_6);
34     GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_7);
35
36     while(1)
37     {
38         delay();
39
40         GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_0, 0xFF); // GPIO Pins ←
41         GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_1, 0xFF);
42         GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_2, 0xFF);
43         GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_3, 0xFF);
44         GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_4, 0xFF);
45         GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_5, 0xFF);

```

```
46  GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_6, 0xFF);
47  GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_7, 0xFF);
48
49  delayPWM();
50
51  GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_0, 0x00);           // GPIO Pins ←
    0–7 auf Low stellen
52  GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_1, 0x00);
53  GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_2, 0x00);
54  GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_3, 0x00);
55  GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_4, 0x00);
56  GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_5, 0x00);
57  GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_6, 0x00);
58  GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_7, 0x00);
59
60  }
61  return 0;
62 }
```

5 Warteschleifen und Taktfrequenz

5.1

Der Mikrokontroller besitzt einen 16 MHz Quarz, welcher mit Hilfe der PLL 400 MHz erreicht. Durch den Divisor Div_5 wird die gegebene Frequenz noch durch 5 dividiert. Somit ergibt sich mit der internen Halbierung eine Frequenz von 40 MHz. Die Berechnung ist in Gleichung 3 gezeigt.

$$\frac{400 \text{ MHz}}{2 \times 5} = 40 \text{ MHz} \quad (3)$$

5.2

```
1  #include <stdint.h>
2  #include <stdbool.h>
3  #include "inc/hw_ints.h"
4  #include "inc/hw_memmap.h"
5  #include "inc/hw_types.h"
6  #include "driverlib/gpio.h"
7  #include "driverlib/sysctl.h"
8  #include "driverlib/timer.h"
9  #include "driverlib/interrupt.h"
10
11
12 // stores ms since startup
13 volatile uint32_t systemTime_ms = 0;
```



```

14
15
16 void InterruptHandlerTimer0A (void)
17 {
18     // Clear the timer interrupt to prevent the interrupt function from ↵
19     // immediately being called again on exit
20     TimerIntClear(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
21     // count up one ms
22     systemTime_ms++;
23 }
24
25 void clockSetup(void)
26 {
27     uint32_t timerPeriod;
28     // Configure clock
29     SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|↵
30     SYSCTL_OSC_MAIN);
31     //enable peripheral for timer
32     SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
33     //configure timer as 32 bit timer in periodic mode
34     TimerConfigure(TIMER0_BASE, TIMER_CFG_PERIODIC);
35     //set timerPeriod to number of periods needed to generate a timeout with a ↵
36     // frequency of 1kHz (every 1ms)
37     timerPeriod = (SysCtlClockGet()/1000);
38     //set TIMER-0-A to generate a timeout after timerPeriod-1 cycles
39     TimerLoadSet(TIMER0_BASE, TIMER_A, timerPeriod-1);
40     //Register the function InterruptHandlerTimer0A to be called when an ↵
41     // interrupt from TIMER-0-A occurs
42     TimerIntRegister(TIMER0_BASE, TIMER_A, &(InterruptHandlerTimer0A));
43     //Enable the interrupt for TIMER-0-A
44     IntEnable(INT_TIMER0A);
45     //generate an interrupt, when TIMER-0-A sends a timeout
46     TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
47     //master interrupt enable for all interrupts
48     IntMasterEnable();
49     //Enable the timer to start counting
50     TimerEnable(TIMER0_BASE, TIMER_A);
51 }
52
53 void delay_ms(uint32_t waitTime)
54 {
55     uint32_t t = systemTime_ms;
56     while (systemTime_ms - t < waitTime);
57 }
58
59 int main(void)
60 {
61     clockSetup();

```

```

59     SysCtlPeripheralEnable (SYSCTL_PERIPH_GPIOB);           // GPIO Port
        B aktivieren
60     GPIOPinTypeGPIOOutput (GPIO_PORTB_BASE, GPIO_PIN_0);    // GPIO
        Port B Pin 0-7 aktivieren
61     GPIOPinTypeGPIOOutput (GPIO_PORTB_BASE, GPIO_PIN_1);
62     GPIOPinTypeGPIOOutput (GPIO_PORTB_BASE, GPIO_PIN_2);
63     GPIOPinTypeGPIOOutput (GPIO_PORTB_BASE, GPIO_PIN_3);
64     GPIOPinTypeGPIOOutput (GPIO_PORTB_BASE, GPIO_PIN_4);
65     GPIOPinTypeGPIOOutput (GPIO_PORTB_BASE, GPIO_PIN_5);
66     GPIOPinTypeGPIOOutput (GPIO_PORTB_BASE, GPIO_PIN_6);
67     GPIOPinTypeGPIOOutput (GPIO_PORTB_BASE, GPIO_PIN_7);
68
69     bool state = 0;
70     // Verbeseserte Variante des Codes der GPIO-Steuerung aus Aufgabe 4
71     while (1)
72     {
73         GPIOPinWrite (GPIO_PORTB_BASE, GPIO_PIN_0, GPIO_PIN_0*state)
            ;
74         GPIOPinWrite (GPIO_PORTB_BASE, GPIO_PIN_1, GPIO_PIN_1*state)
            ;
75         GPIOPinWrite (GPIO_PORTB_BASE, GPIO_PIN_2, GPIO_PIN_2*state)
            ;
76         GPIOPinWrite (GPIO_PORTB_BASE, GPIO_PIN_3, GPIO_PIN_3*state)
            ;
77         GPIOPinWrite (GPIO_PORTB_BASE, GPIO_PIN_4, GPIO_PIN_4*state)
            ;
78         GPIOPinWrite (GPIO_PORTB_BASE, GPIO_PIN_5, GPIO_PIN_5*state)
            ;
79         GPIOPinWrite (GPIO_PORTB_BASE, GPIO_PIN_6, GPIO_PIN_6*state)
            ;
80         GPIOPinWrite (GPIO_PORTB_BASE, GPIO_PIN_7, GPIO_PIN_7*state)
            ;
81
82         state = !state;
83         delay_ms (500/2);
84     }
85 }

```

5.3

Tabelle 1: Flimmerverschmelzungsfrequenz

Name	Frequenz [Hz]
David	50
Jan	47,6
Max	45

6 Digitale Eingänge

6.1 Taster

6.1.1

```
1  /**
2   * main.c
3   */
4  #include <stdint.h>
5  #include <stdbool.h>                                // definition ←
6  #include "inc/hw_memmap.h"                          // definition ←
7  #include "inc/hw_types.h"                          // definition ←
8  #include "driverlib/gpio.h"
9  #include "driverlib/sysctl.h"
10
11 void delay(void)                                    // Blinklicht ←
12     Delay
13 {
14     uint32_t i=500000;
15     while(i) {i--;}
16 }
17 void delayPWM(void)                                // PWM Delay (←
18     Hoher Wert -> Helle LED)
19 {
20     uint32_t i=10000;
21     while(i) {i--;}
22 }
23
24 int main(void){
25     SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);    // GPIO Port B ←
26     aktivieren
27     GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_0); // GPIO Port B ←
28     Pin 0-7 aktivieren
29     GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_1);
30     GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_2);
31     GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_3);
32     GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_4);
33     GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_5);
34     GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_6);
35     GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_7);
36
37     GPIOPinTypeGPIOInput(GPIO_PORTC_BASE, GPIO_PIN_7); // Schalter ←
38     input config
```

```

36  GPIOPadConfigSet (GPIO_PORTC_BASE, GPIO_PIN_7, GPIO_STRENGTH_2MA, ↵
    GPIO_PIN_TYPE_STD_WPU) ;
37
38  while(1)
39  {
40      delay() ;
41      if (GPIOPinRead (GPIO_PORTC_BASE, GPIO_PIN_7) == 0) {
42          GPIOPinWrite (GPIO_PORTB_BASE, GPIO_PIN_0, 0xFF) ;           // GPIO Pins ↵
            0-7 auf High stellen
43          GPIOPinWrite (GPIO_PORTB_BASE, GPIO_PIN_1, 0xFF) ;
44          GPIOPinWrite (GPIO_PORTB_BASE, GPIO_PIN_2, 0xFF) ;
45          GPIOPinWrite (GPIO_PORTB_BASE, GPIO_PIN_3, 0xFF) ;
46          GPIOPinWrite (GPIO_PORTB_BASE, GPIO_PIN_4, 0xFF) ;
47          GPIOPinWrite (GPIO_PORTB_BASE, GPIO_PIN_5, 0xFF) ;
48          GPIOPinWrite (GPIO_PORTB_BASE, GPIO_PIN_6, 0xFF) ;
49          GPIOPinWrite (GPIO_PORTB_BASE, GPIO_PIN_7, 0xFF) ;
50      }
51      else
52      {
53          GPIOPinWrite (GPIO_PORTB_BASE, GPIO_PIN_0, 0x00) ;           // GPIO Pins ↵
            0-7 auf Low stellen
54          GPIOPinWrite (GPIO_PORTB_BASE, GPIO_PIN_1, 0x00) ;
55          GPIOPinWrite (GPIO_PORTB_BASE, GPIO_PIN_2, 0x00) ;
56          GPIOPinWrite (GPIO_PORTB_BASE, GPIO_PIN_3, 0x00) ;
57          GPIOPinWrite (GPIO_PORTB_BASE, GPIO_PIN_4, 0x00) ;
58          GPIOPinWrite (GPIO_PORTB_BASE, GPIO_PIN_5, 0x00) ;
59          GPIOPinWrite (GPIO_PORTB_BASE, GPIO_PIN_6, 0x00) ;
60          GPIOPinWrite (GPIO_PORTB_BASE, GPIO_PIN_7, 0x00) ;
61      }
62  }
63  return 0;
64 }

```

6.1.2

```

1
2  /**
3   * main.c
4   */
5  #include <stdint.h>
6  #include <stdbool.h>           // definition ↵
    of type "bool"
7  #include "inc/hw_memmap.h"    // definition ↵
    of memory addresses
8  #include "inc/hw_types.h"     // definition ↵
    of framework makros
9  #include "driverlib/gpio.h"
10 #include "driverlib/sysctl.h"
11

```

```

12 void delay(void)
13 {
14     uint32_t i=200000;
15     while(i) {i--;}
16 }
17
18 int main(void){
19
20     SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);           // Port F ←
21     Aktivieren
22     GPIOPinTypeGPIOInput(GPIO_PORTF_BASE,GPIO_PIN_4);      // PF4 als ←
23     Input definieren
24     GPIOPadConfigSet(GPIO_PORTF_BASE,GPIO_PIN_4,GPIO_STRENGTH_2MA,←
25     GPIO_PIN_TYPE_STD_WPU);
26     // Bei der der Unteraufgabe 6.1.2 muesste der der Port B und Pin 7 gewaehlt←
27     werden
28
29     SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);           // GPIO Port B ←
30     aktivieren
31     GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_0);    // GPIO Port B ←
32     Pin 0-7 aktivieren
33     GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_1);
34     GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_2);
35     GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_3);
36     GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_4);
37     GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_5);
38     GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_6);
39     GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_7);
40
41     bool state = 0;
42
43     while(1)
44     {
45         delay();
46
47         if (GPIOPinRead(GPIO_PORTF_BASE,GPIO_PIN_4)==0)    // Wenn ←
48             Schalter geschlossen
49         {
50             state = !state;                                  // State ←
51             wird gewechselt
52         }
53
54         GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_0, GPIO_PIN_0*state); // GPIO ←
55         Output wird auf jeweiligen State geschaltet
56         GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_1, GPIO_PIN_1*state);
57         GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_2, GPIO_PIN_2*state);
58         GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_3, GPIO_PIN_3*state);
59         GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_4, GPIO_PIN_4*state);
60         GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_5, GPIO_PIN_5*state);
61         GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_6, GPIO_PIN_6*state);

```

```
53     GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_7, GPIO_PIN_7*state);
54 }
55     return 0;
56 }
```

6.1.3

Bei der der Unteraufgabe 6.1.2 müsste der der Port B und Pin 7 gewählt werden

6.2 Externer Interrupt

```
1  #include <stdint.h>
2  #include <stdbool.h>
3  #include "inc/hw_ints.h"
4  #include "inc/hw_memmap.h"
5  #include "inc/hw_types.h"
6  #include "driverlib/gpio.h"
7  #include "driverlib/sysctl.h"
8  #include "driverlib/timer.h"
9  #include "driverlib/interrupt.h"
10 // Adresse zum ansteuern der jeweiligen LED
11 volatile uint32_t LED = 1;
12
13 //Delay-Funktion  $f\tilde{A}\frac{1}{4}r$  Blinken
14 void delay(void)
15 {
16     uint32_t i=80000;
17     while(i) {i--;}
18 }
19
20 // Interruptroutine
21
22 void ex_int_handler(void) {
23
24     //Schalte vorherige LED aus
25
26     GPIOPinWrite(GPIO_PORTB_BASE, LED, 0x00);
27
28     // Interrupt-Flag  $I\tilde{A}\frac{1}{4}$ schen
29
30     GPIOIntClear(GPIO_PORTF_BASE, GPIO_PIN_4);
31
32     GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_4);
33
34     LED = LED<<1;
35     // Von 8. LED zur  $\tilde{A}\frac{1}{4}ck$  auf 1. springen
36     if ( LED >= 1<<8)
37     {
```

```

38 LED = 1;
39 }
40 }
41
42 void main (void){
43     SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);           // GPIO Port B ←
44     aktivieren
45     GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_0);    // GPIO Port B ←
46     Pin 0–7 aktivieren
47     GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_1);
48     GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_2);
49     GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_3);
50     GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_4);
51     GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_5);
52     GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_6);
53     GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_7);
54
55     SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);           // Port F ←
56     Aktivieren
57     GPIOPinTypeGPIOInput(GPIO_PORTF_BASE, GPIO_PIN_4);      // PF4 als ←
58     Input definieren
59     GPIOPadConfigSet(GPIO_PORTF_BASE, GPIO_PIN_4, GPIO_STRENGTH_2MA, ←
60     GPIO_PIN_TYPE_STD_WPU);
61
62     // Pin mit Interrupt
63     GPIOIntDisable(GPIO_PORTF_BASE, GPIO_PIN_4);
64     GPIOIntClear(GPIO_PORTF_BASE, GPIO_PIN_4);
65     GPIOIntTypeSet(GPIO_PORTF_BASE, GPIO_PIN_4, GPIO_FALLING_EDGE);
66     GPIOIntRegister(GPIO_PORTF_BASE, ex_int_handler);
67     GPIOIntEnable(GPIO_PORTF_BASE, GPIO_PIN_4);
68
69     while(1) {
70         delay();
71         GPIOPinWrite(GPIO_PORTB_BASE, LED, 0xFF);
72         delay();
73         GPIOPinWrite(GPIO_PORTB_BASE, LED, 0x00);
74     }
75 }

```

7 Analog-Digital-Umsetzer

7.1 A/D-Umsetzer

7.1.1

7.1.2

7.2 Widerstandsmessung

8 Lautstärkepegel

8.1 Lautstärkepegel

8.2 Eigene Verbesserungen