# RFID Praktikum 1

David Falk (736532), Christian Müller (737158)

19. Oktober 2015

## 1 OCR

Um ein Signal mit 50 Hz zu erhalten müssen wir zunächst die Werte für Register C und A des Timers berechnen.

Wir wählen 8 als Vorteiler. Zusammen mit der CPU-Frequenz von 25 MHz erhalten wir:

$$RC = \frac{25\,\text{MHz}}{8 \cdot 50\,\text{Hz}} = 62500$$

Da ein synchrones Rechtecksignal anliegen soll wählen wir für Register A:

$$RA = \frac{1}{2}RC = 31250$$

Diese Werte benutzen wir dann in der Funktion für die Timerinitialisierung:

```
void Timer3_init(void) {
  StructTC* timerbase3  = TCB3_BASE; // Basisadressse TC Block 1
  StructPIO* piobaseA   = PIOA_BASE; // Basisadresse PIO A

  timerbase3->TC_CCR = TC_CLKDIS;  // Disable Clock

  // Initialize the mode of the timer 3
  timerbase3->TC_CMR =
    TC_ACPC_CLEAR_OUTPUT  | //ACPC   : Register C clear TIOA
    TC_ACPA_SET_OUTPUT    | //ACPA   : Register A set TIOA
    TC_WAVE               | //WAVE   : Waveform mode
    TC_CPCTRG             | //CPCTRG : Register C compare trigger
    TC_CLKS_MCK8;           //TCCLKS : MCKI / 8
```

```
  // Initialize the counter:
  timerbase3->TC_RA = 31250; // 0.5 * TC_RC (synchrones Signal)
  timerbase3->TC_RC = 62500; // 25Mhz/(8*50Hz)

  // Start the timer :
  timerbase3->TC_CCR = TC_CLKEN;       // Enable the clock counter
  timerbase3->TC_CCR = TC_SWTRG;       // Trigger timer
  piobaseA->PIO_PER  = (1<<PIOTIOA3); // Enable output
  piobaseA->PIO_OER  = (1<<PIOTIOA3); // Block clock output
  piobaseA->PIO_CODR = (1<<PIOTIOA3); // by enabling perpheral output
}
```

## Aufgabe 2

Für die Abfrage der Tasten benutzen wir folgende Initialisierungsfunktion. Hier wird den Tasten 1-3 ein Interrupthandler zugewiesen.

```
void interrupt_init(void) {
  StructAIC* aicbase = AIC_BASE;    // Basisadresse AIC
  StructPIO* piobaseB = PIOB_BASE; // Basisadr. PIO B

  aicbase->AIC_IDCR = 1 << 14; // Interrupt PIOB ausschalten
  aicbase->AIC_ICCR = 1 << 14; // Interrupt PIOB loeschen

  aicbase->AIC_SVR[PIOB_ID] = (unsigned int)button_irq_handler;
  aicbase->AIC_SMR[PIOB_ID] = 0x7; // Priority
  aicbase->AIC_IECR = 1 << 14;      // Interrupt PIOB einschalten

  piobaseB->PIO_ODR = KEY1 | KEY2 | KEY3;
  piobaseB->PIO_IER = KEY1 | KEY2 | KEY3; // Interrupt innerhalb PIOB
      erlauben
}
```

Analog zu Praktikum 2 wird in unserem Interrupthandler anhand des Statusregisters abgefragt welche Tasten gedrückt sind und dann jeweils eine globale Variable auf *true* gesetzt.

```
volatile int button1pressed, button2pressed, button3pressed;

void button_irq_handler (void) __attribute__ ((interrupt));

// Interruptserviceroutine
void button_irq_handler (void) {
  StructPIO* piobaseB = PIOB_BASE; // Basisadresse PIO B
  StructAIC* aicbase  = AIC_BASE;  // Basisadresse AIC
    if ((piobaseB->PIO_PDSR & KEY1) == 0)
      button1pressed = 1;

    else if ((piobaseB->PIO_PDSR & KEY2) == 0)
      button2pressed = 1;

    else if ((piobaseB->PIO_PDSR & KEY3) == 0)
      button3pressed = 1;

  aicbase->AIC_EOICR = piobaseB->PIO_ISR; // End of interrupt
}
```

In der main-Funktion werden zunächst das PMC aktiviert, unsere Initialisierungsfunktionen aufgerufen, der Timer gestartet und die globalen Variablen initialisiert.

```
  pmcbase->PMC_PCER = 1<<14 | 1<<13 | 1<<9; // Power fuer PIOB, PIOA und
      TC3 einschalten
  Timer3_init();
  interrupt_init();
  piobaseA->PIO_PDR  = (1<<PIOTIOA3); // Deaktivieren um Timer die
      Kontrolle zu geben

  button1pressed = 0;
  button2pressed = 0;
  button3pressed = 0;
```

In einer Endlosschleife werden dann die globalen Variablen abgefragt und die Pumpe bzw. der Timer entsprechend aus- oder eingeschaltet.

```
  while(1) {
    if (button1pressed) {
      // Enabel peripheral output (disable timer)
```

```
      piobaseA->PIO_PER  = (1<<PIOTIOA3) ;
      button1pressed = 0;
    }
    if (button2pressed) {
      // Disable peripheral output (enable timer)
      piobaseA->PIO_PDR  = (1<<PIOTIOA3);
      button2pressed = 0;
    }
    if (button3pressed) {
      button3pressed = 0;
    }
  }
```