Übung 03: Interrupts, EEPROM

Hardware:

- Basis: Arduino, Steckbrett, Kabel
- Taster
- Widerstand: 1 kΩ

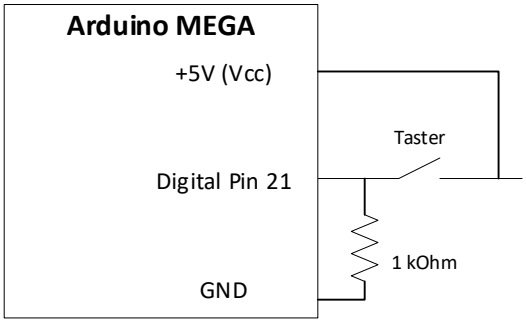
Informationen:

- Datenblatt: Kapitel 8 „AVR Memories“, Kapitel 14 „Interrupts“ und Kapitel 15 „Externe Interrupts“
- Arduino Pin Mapping: https://www.arduino.cc/en/Hacking/PinMapping2560

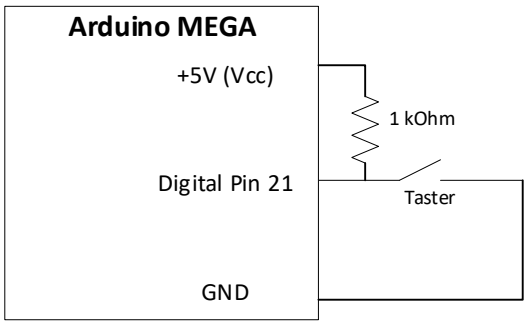
Aufgabe 1: Externe Interrupts mit der Arduino Library

Ziel-Anwendung: Der Mikrocontroller zählt per Interrupt mit, wie oft ein angeschlossener Taster bereits gedrückt wurde. Bei ***geschlossenem*** Tasters soll ***HIGH*** eingelesen werden, bei ***offenem*** Taster ***LOW***. Der aktuelle „Zählerstand“ soll über die serielle Konsole ausgegeben werden.

a) Benötigen Sie die linke oder die rechte Schaltung? Begründen!



Pull-Down, Active High



Pull-Up, Active Low

b) **Polling:** Überlegen Sie vorab, was problematisch am rechten Programm ist? Begründen Sie!
Tipp: `delay(.)`

c) Bauen Sie die korrekte Schaltung aus a) nach!

d) Anstatt über *Polling*, soll nun ein **externer Interrupt** eingesetzt werden. Welcher externe Interrupt (`INT<X>`) ist eine Alternate Function des Digital Pin 21?
Hinweis: Pin Mapping

```
int counter = 0;

void setup() {
   Serial.begin(9600);
   pinMode(21, INPUT);
}
void loop() {
   if (digitalRead(21) == HIGH)
      counter++;
   Serial.println(counter) ;
   delay(3000);
}
```

e) Modifizieren Sie das vorgegebene Programm, so dass der Tastendruck per externem Interrupt erkannt wird. Es gelten folgende Anforderungen / Hinweise:

- Beachten Sie die Unterlagen der Vorlesung.
- Verwenden Sie das Kommando `attachInterrupt()`[1]. Bei einer *steigenden* Flanke soll der Zähler um 1 erhöht werden.
- Fügen Sie eine weitere Methode hinzu, die als Interrupt Service Routine (ISR) fungiert.
- Sie dürfen `digitalRead(.)` **nicht** mehr verwenden.
- In der `loop`-Methode(.) soll weiterhin ein `delay(3000)` stehen. Trotz dieses „Busy Waitings“ soll ein Tastendruck jederzeit erkannt werden.

[1] https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/

f) **Entprellung:** Wird jeder Tastendruck nur einmal erkannt?
Fügen Sie Ihrem Programm eine SW-Entprellung wie folgt hinzu: Jede ISR erfasst anfangs die aktuelle Zeit mit `millis()`. Merken Sie sich diese Zeit bis zum nächsten Aufruf der ISR. Sind seit dem letzten Aufruf weniger als 200 *ms* vergangen, soll der Zähler nicht erhöht werden.

Wichtiger Hinweis: In einer ISR funktioniert kein `delay` und auch der Wert von `millis()` wird nicht korrekt aktualisiert. Der Grund ist, dass beide intern mit Timer-Interrupts arbeiten und weitere Interrupts sind innerhalb einer ISR gesperrt. Setzt man wie in f) `millis()` ein, so sollte die ISR so kurz wie möglich sein. Allgemeiner Rat: Verwenden Sie zur Abfrage von Tastern besser Polling mit einer SW-Entprellung per `delay`-Kommando oder fragen Sie einfach den Taster nicht zu häufig ab, siehe Übung 2. Für andere Ereignisse wie z.B. Unterbrechen einer Lichtschranke sind Interrupts jedoch ideal.

Aufgabe 2: Externe Interrupts mit der AVR-Libc

Die Schaltung und das Programmverhalten bleiben unverändert. Die Kommandos `pinMode`, `digitalRead` und `attachInterrupt` dürfen aber **nicht** mehr verwendet werden!
Hinweise:

- Die ISR kann wie folgt definiert werden: `ISR (INT0_vect) {/* ISR handling*/}`
  Details siehe: http://www.nongnu.org/avr-libc/user-manual/group__avr__interrupts.html
- Datenblatt S. 110: Register *EIMSK* und *EICRA* konfigurieren.
- Datenblatt S. 13: `I`-Bit im SREG-Register setzen. Am einfachsten durch Kommando `sei()`. Dadurch werden Interrupts global aktiviert. `cli()` würde deaktivieren.

Aufgabe 3: Theorie – Betriebsstundenzähler und Wear Leveling

Für Wartungs- oder Servicezwecke möchte ein Hersteller wissen, wie viele Stunden der Kunde sein Gerät angeschaltet hat (*Betriebsstundenzähler)*. In das Gerät ist ein Atmega2560 Mikrocontroller eingebaut. Der Mikrocontrollers protokolliert die Betriebsstunden im internen EERPOM[2], damit die Information auch während eines Reboots erhalten bleibt. Das Least Significant Byte (LSB) wird somit einmal jede Stunde verändert bzw. geschrieben.

a) Lesen Sie den ersten Absatz von 8.3 „EEPROM Data Memory“ (Handbuch S. 22). Warum könnte die beschriebene Anwendung problematisch sein?

b) Darf das Gerät unter oben beschriebenen Anforderungen 20 Jahre im Einsatz sein?

c) Abhilfe schafft „*Wear Leveling*“! Lesen Sie sich die folgende Application Note durch: http://ww1.microchip.com/downloads/en/Appnotes/doc2526.pdf

d) Die Arrays, die die beiden Ringpuffer repräsentieren, sehen wie folgt aus:
   - *Parameter Buffer*: `[ a | b | c | d | e | f ]`
   - *Status Buffer*: `[ 96 | 97 | 98 | 93 | 94 | 95 ]`

   Wie oft kann man jetzt den Wert des Zählerstands modifizieren?
   Was ist der aktuell gültige Wert im *Parameter Buffer*? (`a, b, c, d, e` oder `f`)?

[2] Im EERPOM werden häufig auch Kalibrierungsdaten und/oder Konfigurationsdaten gespeichert.