# Informationstechnik-Praxis / AVR-Übungsblatt 4

## Lernziele / Tipps / allgemeine Voraussetzungen:

Verbinden Sie für alle Aufgaben die rote, gelbe und grüne LED mit **Port C, Bit 3, 4 und 5**.
Verbinden Sie außerdem die beiden Taster mit **Port B, Bit 4 und 5**.

**Lassen Sie für ALLE Versuche diese Verbindungen genau SO gesteckt!**

Wir verwenden auch wieder „Zeitschleifen“ zur Zeitverzögerung (siehe letzte Praktika).

Wir verwenden nun auch **Unterprogramme (Routinen)**!

### Was ist zu beachten?

Unterprogramme (Routinen) sind wie Funktionen im Fach „Programmiertechnik“ zu verstehen:

- Wir schreiben Unterprogramm an das Ende unseres Quelltextes.
- Markieren Sie bitte Anfang und Ende von Unterprogrammen mit einer Kommentarzeile!
- Die erste Zeile eines Unterprogramms beginnt mit einem Sprungziel (Label).
- Zurück kommen wir mit dem Return-Befehl **`ret`**
- Das Unterprogramm wird aufgerufen durch den Befehl **`rcall Sprungziel`**

Allgemeingültiges Beispiel:

```
...   .....
rcall Marke1
...   .....        ; hier hin springt das Programm zurück
; ---------------------------------------------------------------------
Marke1:            ; hier beginnt das Unterprogramm
        nop        ; hier natürlich irgendwas „sinnvolles“...
        nop
        ret        ; zurück hinter die aufrufende Stelle
; ---------------------------------------------------------------------
```

In der Regel wird irgendein Zustand (eine Bedingung) geprüft – zum Beispiel, ob ein Taster gedrückt wurde oder nicht – und abhängig davon ein Unterprogramm aufgerufen (oder eben auch nicht).

Dies bekommen sie am besten hin mit den Branch-Verzweigungsbefehlen (siehe Referenz, da stehen etliche Verzweigungsbefehle zur Verfügung) oder indem Sie direkt ein Bit in einem Register oder einem PORT prüfen, und zwar mit dem **`sbis`** oder **`sbic`**-Befehl, bedeutet: **s**kip next instruction if **b**it **i**s **s**et (1)/ **s**kip next instruction if **b**it **i**s **c**leared (0)):

Beispiel:

```
sbis  PINx, Bit    ; ein Bit am Eingang direkt abfragen und
                     – wenn gesetzt – den nächsten Befehl auslassen!
rcall Marke1       ; das Unterprogramm wird also aufgerufen,
                     wenn das Bit NICHT gesetzt ist
nop                ; an diese Stelle kehrt das Unterprogramm zurück
```

**Achtung: Wegen der Pullup-Widerstände haben die Taster-Zustände folgende Pegel:**

- **Taster ist gedrückt: Bit ist 0**
- **Taster nicht gedrückt: Bit ist 1**

**Im obigen Beispiel wird das Unterprogramm NICHT aufgerufen, wenn das zu prüfende Bit (auf 1) gesetzt ist (also der Taster NICHT gedrückt ist)!**

**Also genau überlegen: wann ist welches Bit 0 oder 1 und welchen Befehl (sbis oder sbic) muss ich verwenden? „Beliebter“ Gedankenfehler (auch bei Ihrem Dozenten…)**

## Aufgabe 1 – (optional):

Stellen Sie die Aufgaben aus dem Übungsblatt 3 fertig (falls noch nicht geschehen)!

Warum? Weil das Verständnis der DDRs, der PORTs, der Ein- und Ausgabefunktionen (in, out, PINx) und der Behandlung und Prüfung einzelner Bits doch irgendwie enorm hilfreich ist…

## Aufgabe 2 – Taster 2 ruft Unterprogramm (Routine) auf:

Wenn Taster 2 nicht gedrückt ist, soll die **grüne** LED in Ruhe leuchten.

Wird Taster 2 gedrückt, soll ein Unterprogramm aufgerufen werden, das die **grüne** LED ausschaltet und stattdessen die **gelbe** und die **rote** LED für einige Sekunden im Wechsel (gut sichtbar) blinken lässt.

Dieses Unterprogramm läuft also auch weiter, wenn der Taster 2 wieder losgelassen wird.
Ein kurzes Antippen sollte also genügen, um das Wechsellicht zu starten.

Danach beenden die **rote** und **gelbe** LED ihre Show und die **grüne** LED geht wieder an.

TIPP: Verwenden Sie wieder eine (mehrfach geschachtelte) Zeitschleife, einen Zähler in einem eigenen Register, um die Anzahl der erfolgten Wechsel von einem Startwert bis auf 0 zu zählen (dekrementieren) und ein weiteres, eigenes Register, um ein vorbereitetes Bitmuster für eine Exklusiv-Oder-Verknüpfung zu haben.

## Aufgabe 3 – Taster ruft Unterprogramm mit Unterprogramm auf:

Auch aus einem Unterprogramm („Routine“) heraus lässt sich ein weiteres Unterprogramm aufrufen.
Wir müssen nur die Übersicht behalten, welches Unterprogramm wann aufgerufen wird und vor Allem: welche Register werden wo verwendet?

Alle Register verhalten sich wie „globale Variablen“, können also überall verwendet werden und somit auch in einem Unterprogramm mit neuen Werten versehen werden, was im aufrufenden Programm unerwartete Effekte bewirken kann…

### Übung:

Schreiben Sie die Zeitverzögerung in eine eigene Routine „Zeit01“, die Ihnen eine Zeitverzögerung zwischen 2 Anzeigewechseln der LEDs erzeugt – also die Zeit, die zwischen zwei unterschiedlichen LED-Mustern vergehen soll.

## Aufgabe 4 – Taster ruft nacheinander unterschiedliche Blinkmuster auf:

Die Blinkmuster von Übungsblatt 2 (Lauflicht, Wechsellicht, Binärmuster) sollen jeweils für etwa 2 bis 5 Sekunden sichtbar werden. Bei jedem Drücken des Tasters 2 soll das nächste Blinkmuster für 2 – 5 Sekunden erscheinen und danach wieder die grüne LED leuchten.

TIPP1: Legen Sie ein eigenes Register fest, in das Sie die „Nummer“ des aktuellen Blinkmusters ablegen. Beim nächsten Tasterdruck wird das nächste Blinkmuster aufgerufen. Sind alle Blinkmuster durch, wird wieder das allererste Blinkmuster erzeugt.

TIPP2: Speichern Sie die Anweisungen für die einzelnen Blinkmuster in eigenen Routinen und rufen, je nach „Nummer“ des Blinkmusters, die entsprechende Routine auf.

## Aufgabe 5 - 2 Taster rufen Blinkmuster aufsteigend oder absteigend ab:

Wie Aufgabe 4, nur können Sie hier mit dem Taster 2 (wie gehabt) das nächste Blinkmuster aufrufen, mit Taster 1 wird jedoch das letzte Blinkmuster aufgerufen (inkrementieren bzw. dekrementieren der „Blinkmuster-Nummer“.).