1. Externer-Interrupt

Ein digitaler Interrupt-Eingang, wird dazu verwendet, um das Hauptprogramm, wenn ein bestimmtes Ereignis, wie eine steigende oder eine fallende Flanke an einem Eingangspin erkannt wird. Siehe Abbildung 1.

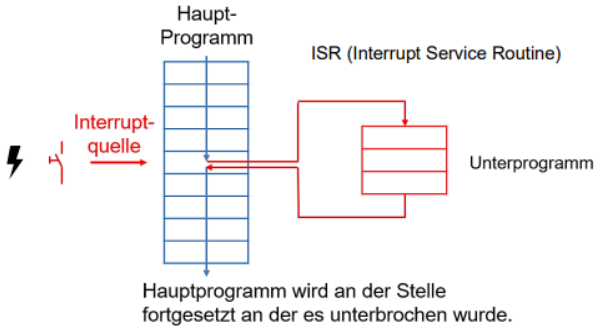

Abbildung 1: Hauptprogramm und Interrupt

Dies ermöglicht es, auf externe Signale in Echtzeit zu reagieren. Das Hauptprogramm wird dann bei einer steigenden Flanke oder einer fallenden Flank am Eingang unterbrochen. In Arduino-Framework verwendet man die Funktion **attachInterrupt()**, um einen Interrupt zuzuweisen. Die Signale werden durch folgende Modi erkannt:

Signalmodies exterer Interrupts:

- RISING: Reagiert auf eine steigende Flanke (Signalwechsel von LOW nach HIGH).
- FALLING: Reagiert auf eine fallende Flanke (Signalwechsel von HIGH nach LOW).
- CHANGE: Reagiert auf jede Flanke (Signalwechsel von HIGH nach LOW oder umgekehrt).

1.1. Beispiel eines externen Interrupts

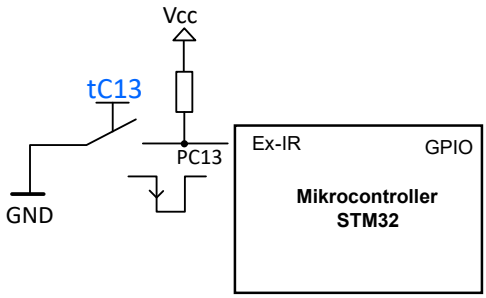

Abbildung 2: Blockschaltbild mitz fallender Flanke an PC13

```cpp
const int tPC13; // Blauer Taster Nucleo NH
void isr_InterruptServiceRoutine() {
  // Unterprogramm, in das bei einem Interrupt
  // gesprungen wird.
}

void setup() {
  // initialisiere Interrupt bei fallender Flanke an Pin PC13 und verknüpfe die ISR
  attachInterrupt(digitalPinToInterrupt(tPC13), isr_InterruptServiceRoutine, FALLING);
}

void loop() {
  // waiting for Interrupt
}
```

Listing 1: Konfiguration und Setup eines externen Interrupts.cpp

1.2. Aufgabe Interrupt Taster tC13 (toggeln)

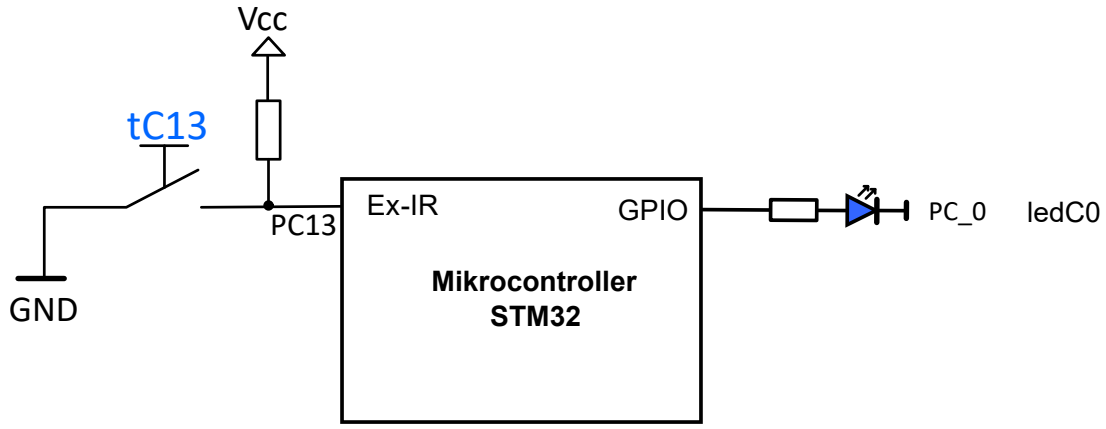

Abbildung 3: Blockschaltbild

Bei Betätigung der Nucleo Taste (Blau) an **PC13** soll bei **fallender Flanke**, die LED an **PC0** an bzw. ausgeschaltet werden (toggeln). Es soll das externe Interrupt eingesetzt werden. Der Zustand der LED (PC0) soll in der globalen Variable `volatile bool ledZustand = LOW` deklariert und mit LOW initialisiert werden.

Interrupt Service Routine (ISR): `void isr_tC13()`

Zweck von Volatile:

Normalerweise optimiert der Compiler den Code, indem er davon ausgeht, dass sich Variablen innerhalb eines Funktionsablaufs nicht unerwartet ändern. Wenn der Compiler jedoch nicht weiß, dass der Wert einer Variablen durch externe Ereignisse (z. B. Interrupts oder Hardwareregister) geändert werden kann, könnte er wichtige Aktualisierungen ignorieren, was zu unerwartetem Verhalten führt.

1.3. Aufgabe Interrupt Taster tC13 (wechseln)

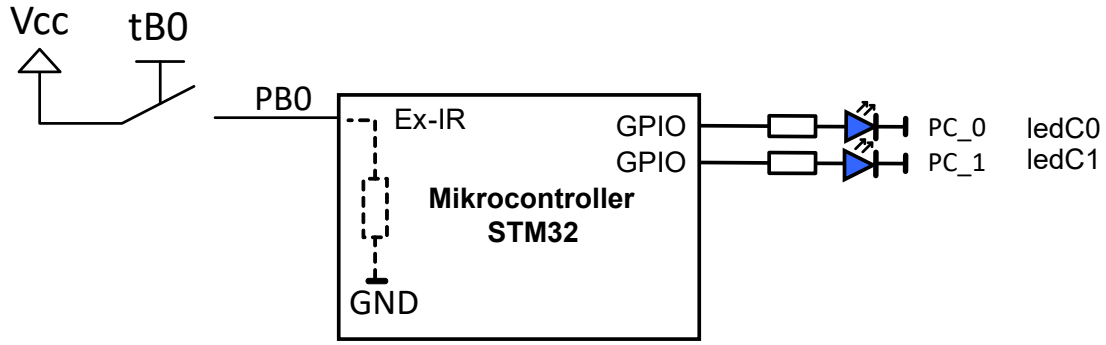

Abbildung 4: Blockschaltbild

Bei Betätigung des Tasters an **PB0** (**fallende Flanke**) soll abwechselnd **ledC0** bzw. **ledC1** aufleuchten. Bitte achten Sie auf die richtige Konfiguration des Eingangs. Es soll das externe Interrupt eingesetzt werden. Die globale umschaltvariable ist `volatile int zustand`.

ISR: `void isr_tB0()`

1.4. Aufgabe Interrupt Wechsel 2 Taster

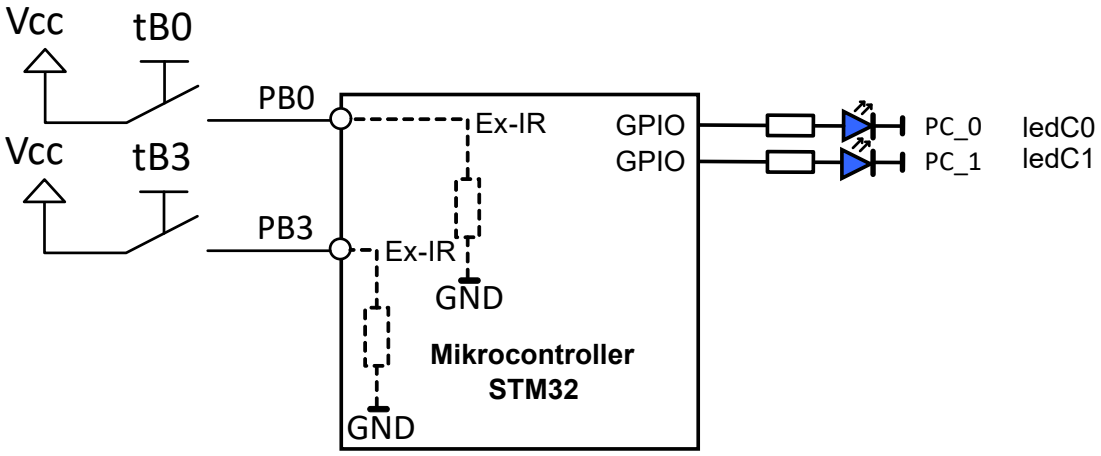

Abbildung 5: Blockschaltbild

Bei Betätigung des Tasters **tB0** (**fallende Flanke**) soll **ledC0** aufleuchten und **ledC1** ausgehen. Bei Betätigung des Tasters **tB3** (**fallende Flanke**) soll **ledC1** aufleuchten und **ledC0** ausgehen. Es soll das externe Interrupt eingesetzt werden. Bitte achten Sie auf die richtige Konfiguration der Eingänge. Die globalen umschaltvariablen sind **`volatile bool zustandLED0`** sowie **`volatile bool zustandLED0`**.

ISR: **`void isr_tB0()`**, **`void isr_tB3()`**

1.5. Aufgabe Interrupt Laufrichtungsänderung

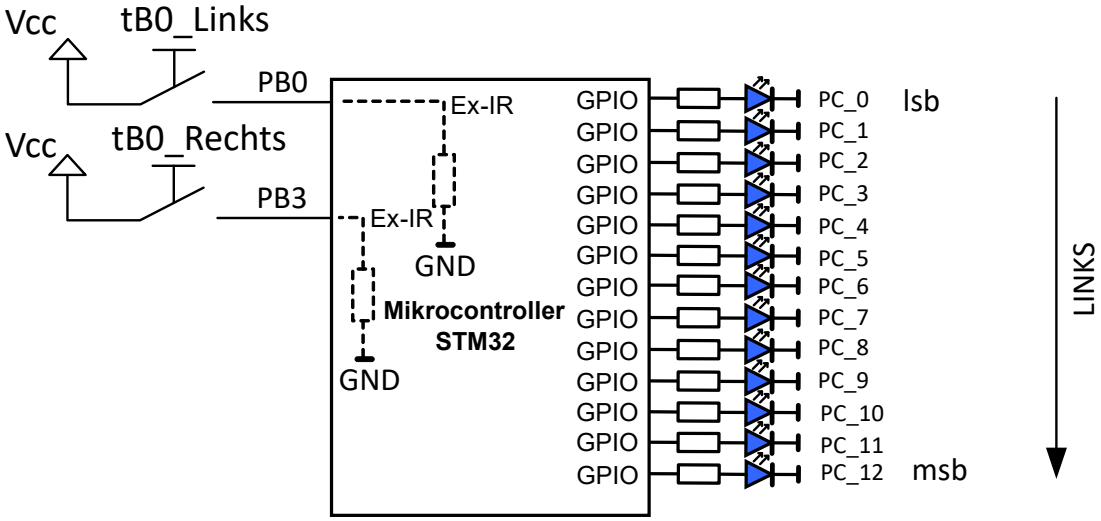

Abbildung 6: Blockschaltbild

Bei Betätigung des Tasters **tB0_Links** (**fallende Flanke**) soll ein Lauflicht mit einem Delay von 200 ms von **PC0, PC1…PC12…PC0…PC12..** durchlaufen. Bei Betätigung des **Tasters tB3_Rechts** (**fallende Flanke**) soll es in die andere Richtung laufen. Das Lauflicht soll portweise und mit Bitweisen Operatoren beschrieben werden. Bitte achten Sie auf die richtige Konfiguration der Eingänge. Die globale umschaltvariable ist `volatile int zustand`. Bitte nutzen Sie zu Unterscheidung zwischen **RECHTS** und **LINKS** einen sog. Enumerator.

Enumeratoren:

Definition für die Aufgabe: `enum zustaende {RECHTS=0, LINKS=1};` Die Variable zustand kann nun definiert werden in dem man sie setzt mit z.b. mit `zustand=RECHTS;`

ISR: **`isr_tB0Links()`**, **`isr_tB3Rechts()`**

1.6. Aufgabe Interrupt Laufrichtungsänderung - Aus

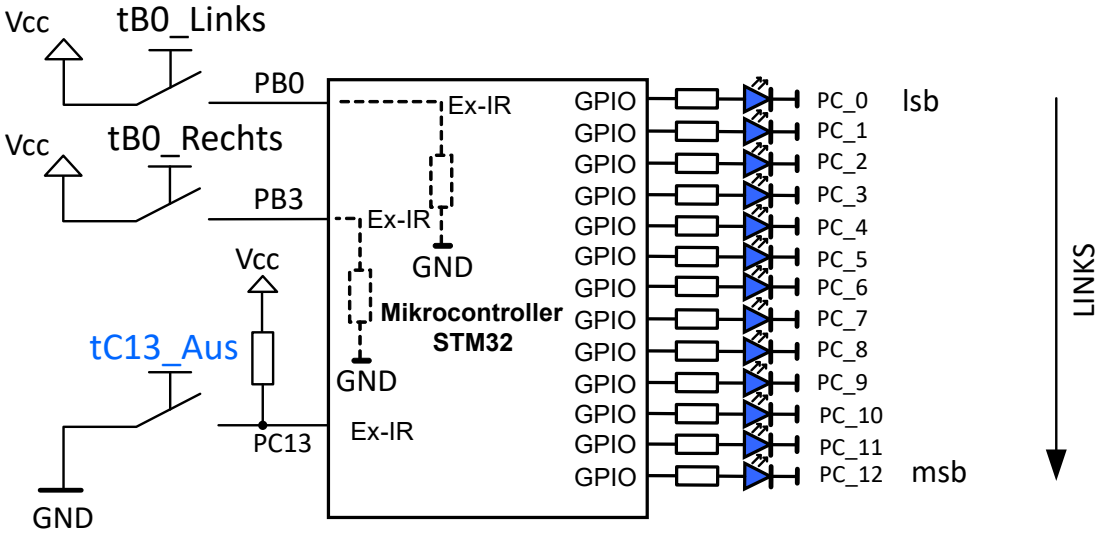

Abbildung 7: Blockschaltbild

Das Lauflicht soll jetzt abschaltbar sein über Taster **tC13_Aus** (**fallende Flanke**). Erweitern sie den Enumerator wie folgt `enum zustaende {RECHTS=0, LINKS=1, AUS=2};`

ISR: **`isr_tB0Links(), isr_tB3Rechts(), isr_tC13_Aus()`**

1.7. Aufgabe Interrupt Impulszähler

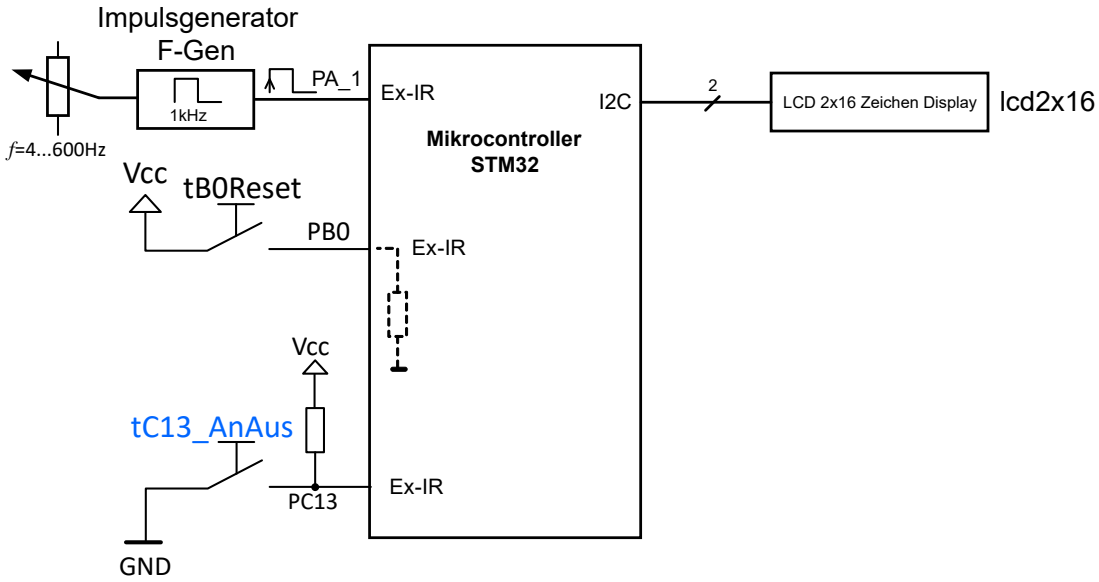

Abbildung 8: Blockschaltbild

Entwickeln Sie einen Impulszähler indem Sie den Frequenzgenerator F-GEN einschalten. Erweitern sie den Enumerator wie folgt enum zustaende **`{AUS=0, ZAEHLEN =1, RESET=2};`** Die Anzahl der gezählten Impulse soll auf dem LCD Display angezeigt werden.