CT2 Praktikum

General Purpose Input/Output (GPIO)

1 Einleitung

GPIOs erlauben eine universelle Verwendung von Ein- und Ausgabepins. Dazu müssen Sie vor dem Zugriff konfiguriert werden. Lesen und schreiben von Werten geschieht in der Regel mittels Register (Abbildung 1).

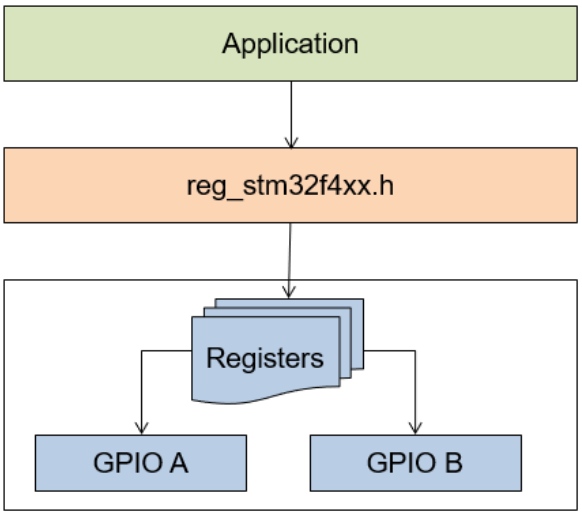


Abbildung 1: Zugriff auf GPIO mittels Register

In diesem Praktikum realisieren Sie die Konfiguration und die Ein- und Ausgabe mittels Registerzugriff für die GPIO der verwendeten Prozessorfamilie STM32F4xx.

2 Lernziele

- Sie können GPIO mit Hilfe von technischen Dokumentationen initialisieren.
- Sie sind in der Lage GPIO zur Ein- und Ausgabe zu verwenden.

3 Material

- 1x CT Board
- 1x Flachbandkabel 16 polig
- 1x Zusatzboard (GPIO) mit 2x Widerstand 580 Ohm

4 Vorbereitung

4.1 Fragen

Bereiten Sie sich mit der Beantwortung der folgenden Fragen auf das Praktikum vor. Laden und builden Sie das vorbereitete Projekt und analysieren Sie den gegebenen Sourcecode.

1. Ein registerzugriff erfolgt beispielsweise über **`GPIOB->MODER`**. Dabei gibt **`GPIOB`** die Basisadresse an und **`MODER`** den Offset. Wo sind diese zwei Symbole definiert (Dateiname und Zeile).

Sie können gegebene Definitionen direkt von Keil suchen lassen. Klicken Sie dazu mit der rechten Maustaste auf den Text und wählen Sie „Go to Definition of…“ aus. Voraussetzung ist, dass Sie mindestens einmal einen Build durchgeführt haben.

2. Was macht das folgende Macro?

```
#define GPIOB   ((reg_gpio_t *) 0x40020400)
```

---

3. Bitte beschreiben Sie in ein bis zwei Sätzen, was **`struct reg_gpio_t`** enthält?

---

4. Erklären Sie, wie das C-Statement
   `GPIOB->MODER = 0x00000280;`
   funktioniert. Wie erfolgt der Zugriff auf das Register?

4.2 Testaufbau mit dem GPIO Zusatzboard

GPIO B ist direkt auf Port P6 des CT Boards herausgeführt. Die Pinbelegung ist in Abbildung 2 dargestellt. Details finden sich im CT-Board Wiki / GPIO (https://ennis.zhaw.ch).

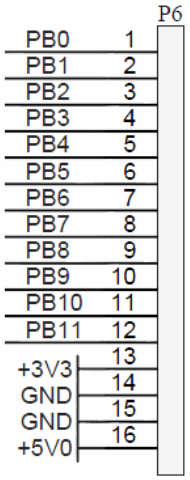


Abbildung 2: Pinbelegung GPIO B auf Stecker P6

Verbinden Sie das Zusatzboard mit Port P6 des CT Boards gemäss Abbildung 3. Benutzen Sie dazu das bereitgestellte Flachbandkabel.

Schliessen Sie ohne Anweisung auf dem Zusatzboard nichts kurz, die GPIO's des Microcontrollers könnten dabei zerstört werden.

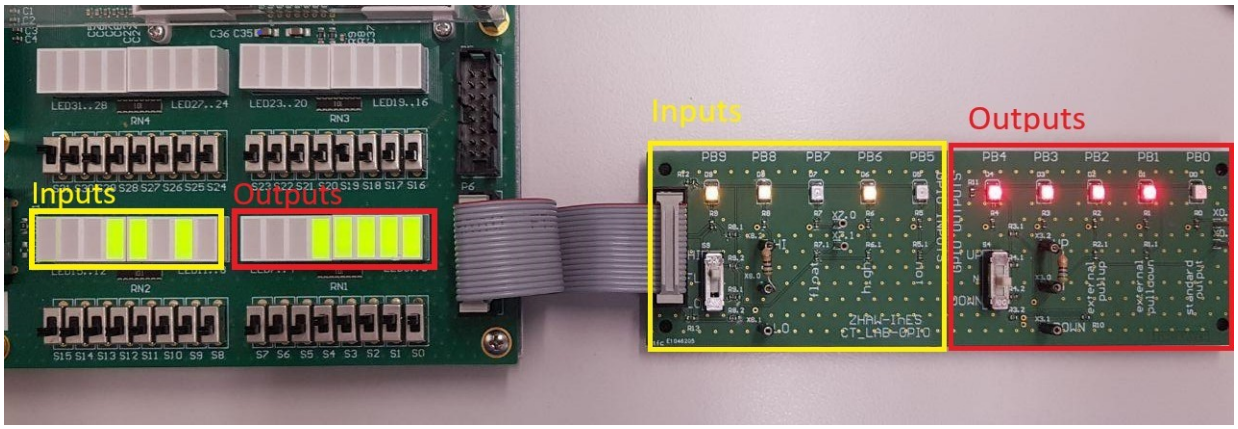


Abbildung 3: Verbinden des Zusatzboards mit GPIOB (P6) des CT Boards

Auf dem Zusatzboard sind die GPIO Pins PB9_PB0 verwendet. Die Beschriftung finden Sie jeweils über der LED. Die GPIO Pins PB10_PB11 sind nicht angeschlossen und werden in diesem Praktikum nicht verwendet.

PB4_PB0 werden als Outputs und PB9_PB5 als Inputs verwendet.

Studieren Sie das Schema des Zusatzboards im Anhang 7.3.

4.3 Programmrahmen

Die Stellung des Schiebeschalters S31 auf dem CT Board schaltet im vorgegebenen Code zwischen zwei Programmteilen um. Für die Aufgaben 5.1 bis 5.4 muss der Schalter auf '1', d.h. in der Stellung oben sein. Für die Aufgabe 5.5 auf '0', d.h. in der Stellung unten.

5 Aufgabe

Im Folgenden implementieren Sie schrittweise einen einfachen Registerzugriff für die GPIO der Microcontrollerfamilie STM32F4xx.

Ergänzen Sie den vorgegebenen Code an den dafür gekennzeichneten Stellen. Verwenden Sie die Konstanten, Macros und Structs aus **`reg_stm32f4xx.h`**, um auf die Register zugreifen zu können.

Aufbauend werden Sie die GPIO PB9_PB0 in ihrem Programm verwenden und mit Hilfe des Zusatzboards die korrekte Funktion Ihrer Implementation überprüfen.

5.1 Einlesen über GPIO

Implementieren Sie als erstes einen Registerzugriff, der PB9 als Input mit einem Pull-Up Widerstand konfiguriert. Die zu ändernden Bits müssen dazu zuerst gelöscht (auf 0 geschrieben) werden. Danach können die entsprechenden Bits gesetzt werden.

Zu setzende Register für eine GPIO Input Konfiguration:

- **`MODER`** **`// mode register`**
- **`PUPDR`** **`// pullup-pulldown register`**

Verifizieren Sie Ihre Funktion, indem Sie durch einen weiteren Registerzugriff im Hauptprogramm (While-Schleife weiter unten) das Input Data Register (**`IDR`**) von PB9 auslesen. Schieben Sie den ausgelesenen 16-bit Wert um 5 Stellen nach rechts[1] und zeigen Sie ihn mit dem Struct **`CT_LED->BYTE`** an LED15_8 an.

Schalten Sie den Schalter S9 auf dem Zusatzboard um und überprüfen Sie so die Korrektheit ihrer Implementation.

Da der Elementzugriff im HAL über Pointer (*) erfolgt, muss anstatt des Punktes (.) ein Pfeil (->) verwendet werden.

Beispiel: **`pointer->element`** entspricht **`(*pointer).element`**

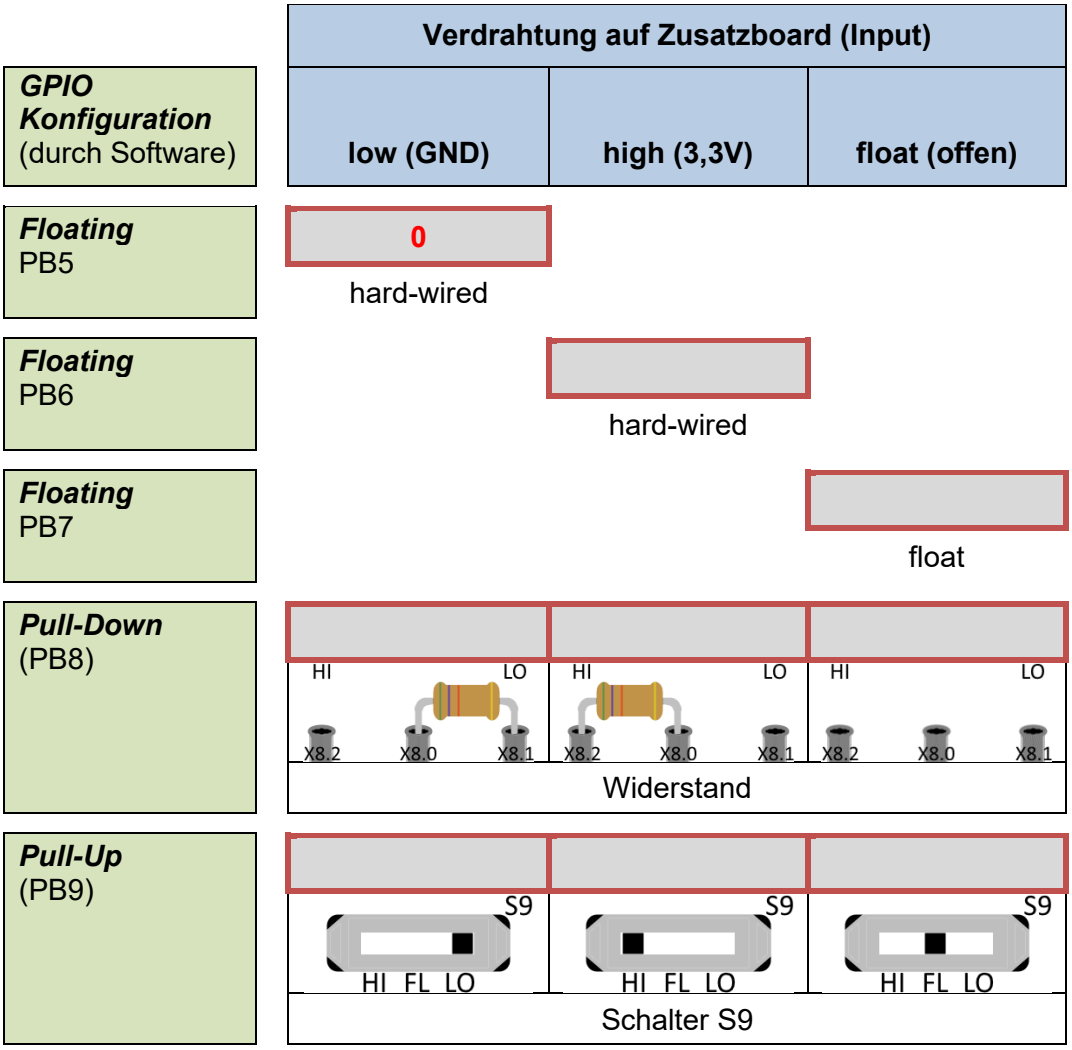
[1] Wir wollen erreichen, dass die eingelesenen Ports PB9-PB5 auf den LEDs 12 bis 8 angezeigt werden. Durch das Schieben nach rechts wird PB5 auf Bit 0 des Bytes LED15_8 angezeigt.

5.2 Konfigurieren und Testen von Inputs

Verwenden Sie weitere GPIO's als Input. Für diese Aufgabe müssen Sie das Schema des Zusatzboards verstanden haben.

a) Konfigurieren Sie die Pins PB9 bis PB5 in der Software gemäss der linken, hellgrünen Spalte der folgenden Tabelle.

Lesen Sie die Werte der Pins entsprechend auf LED15_8 ab und tragen Sie diese in die grauen Felder der Tabelle ein.



b) Wie erklären Sie sich die Unterschiede zwischen den Pins bei offenem Kontakt (letzte Spalte)?

c) Halten Sie beim offenen Kontakt PB7 ohne Pull Widerstand einen Finger (evtl. vorher benetzen) auf die Pins X7.0 und X7.1. Was stellen Sie fest? Warum ist das so? Achten Sie nochmals auf ihre ausgefüllte Tabelle und korrigieren Sie, falls notwendig.

5.3 Ausgabe über GPIO

In diesem Teil implementieren Sie die notwendigen Registerzugriffe für eine Output-Konfiguration.

Gleich wie bei der Konfiguration des Inputs, müssen auch hier die entsprechenden Registerbits zuerst gelöscht und anschliessend richtig beschrieben werden.

Zu setzende Register für eine GPIO Output Konfiguration:

- **`MODER`** `// mode register`
- **`PUPDR`** `// pullup-pulldown register`
- **`OSPEEDR`** `// output speed register`
- **`OTYPER`** `// output type register`

Achten Sie auf **`OTYPER`** (Output Type Register). Es hat eine andere Bitbreite als die übrigen Register.

Um die Output-Pins benutzen zu können, werden Registerzugriffe zum Lesen und Schreiben der Outputs benötigt. Bei diesem Zugriff müssen Sie den Inhalt des **`ODR`** (Output Data Register) beschreiben bzw. an den Aufrufer zurückgeben.

Testen Sie Ihre Implementation, indem Sie PB2_PB0 als Output wie folgt konfigurieren:

- Open Drain
- Kein Pull Widerstand
- Low speed

a) Setzen Sie die Outputs PB2 bis PB0 im **ODR** auf ‚1' (HIGH). Warum brennt nur die LED PB2 aber PB1 und PB0 nicht?

b) Benetzen Sie ihren Finger und halten ihn auf die zwei Stiftkontakte X0.0 und X0.1 (Schema, Anhang 7.2). Was beobachten Sie und warum?

c) Wie könnte man dieses Problem ohne Push-Pull Ausgang lösen?

5.4 Konfigurieren und Testen verschiedener Outputs

a) Konfigurieren Sie nun PB0 mit einem Pull-Up Widerstand und ändern Sie den Output PB1 in eine Push-Pull Stufe. Entspricht das Ergebnis nun ihren Erwartungen?

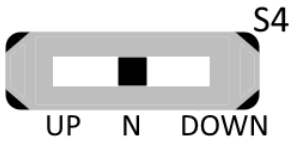
Wegen des sehr hochohmigen Pull-Up Widerstandes des Microcontrollers (unter anderem so gewählt wegen des Stromverbrauchs), reicht es fast nicht aus, eine LED zu treiben. Deswegen leuchtet die LED nur sehr schwach, wenn der Ausgang nur mit einem Pull-Up des Microcontrollers getrieben wird.

b) Konfigurieren Sie PB3 und PB4 als Output. Dabei soll PB3 als Open Drain Stufe ohne Pull Widerstand konfiguriert werden. PB4 soll eine Push-Pull Stufe sein.

Lesen Sie zusätzlich die DIP-Switchs S4_S0 ein und geben Sie ihre Zustände auf Output PB4_PB0 aus (**ODR**).

Zur Kontrolle lesen Sie die Zustände der Output-GPIOs zurück (indem Sie **ODR** auslesen) und geben diese dann auf die LED7_0 des CT Boards aus.

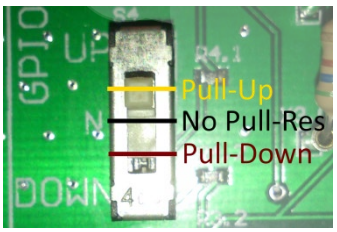
Schalten Sie den Schalter S4 auf dem Zusatzboard auf N und entfernen Sie den externen Pull Widerstand von PB3 (auf dem Zusatzboard).



Schalten Sie die DIP-Switchs S4_S0 auf HIGH. Warum brennt die LED D3 auf dem Zusatzboard nicht?

c) Fügen Sie nun auf dem Zusatzboard PB3 einen externen Pull Widerstand ein (X3.0 auf X3.1 bzw. X3.0 auf X3.2). Testen Sie über DIP-Switch S3 die Funktion des Outputs. Braucht es hier nun einen Pull-Up oder Pull-Down Widerstand?

d) Mit dem 3-stufigen DIP-Switch S4 auf dem Zusatzboard können Sie das gleiche machen, wie bei PB3 mit dem Pull Widerstand. Nach oben wird ein Pull-Up hinzu geschaltet, nach unten ein Pull-Down. In der Mittelstellung ist auf dem Zusatzboard kein Pull Widerstand dazu geschaltet.



Schalten Sie den DIP-Switch S4 auf dem Zusatzboard. Wieso macht die Schalterstellung keinen Unterschied auf das Ergebnis?

5.5 Zusatzaufgabe: Set / Reset / Toggle

Für diese Aufgabe schalten Sie den Dipswitch S31 auf dem CT Board in die Stellung unten. Implementieren Sie die Registerzugriffe zum bitweisen Setzen der GPIO:

- **`Reset -> Setzt den Output-Pin auf Low`**
- **`Set -> Setzt den Output-Pin auf High`**
- **`Toggle -> Toggelt den Output-Pin High->Low / Low->High`**

Prüfen Sie die Funktionen, indem Sie PB0 reseten, PB1 setzen und PB2 toggeln. Das vorgegebene Programm wartet jeweils auf das nächste Drücken des Buttons T0.

6 Bewertung

Bewertungskriterien	**Gewichtung**
Die Funktionen für die GPIO Inputs wurden gemäss Anforderungen in Aufgabe 5.1 in den HAL implementiert und können erklärt werden.	1/4
Das Programm erfüllt die in Aufgabe 5.2 geforderte Funktionalität und ist getestet.	1/4
Die Funktionen für die GPIO Outputs wurden gemäss Anforderungen in Aufgabe 5.3 in den HAL implementiert und können erklärt werden.	1/4
Das Programm erfüllt die in Aufgabe 5.4 geforderte Funktionalität und ist getestet.	1/4

7 Anhang

7.1 GPIO Zusatzboard

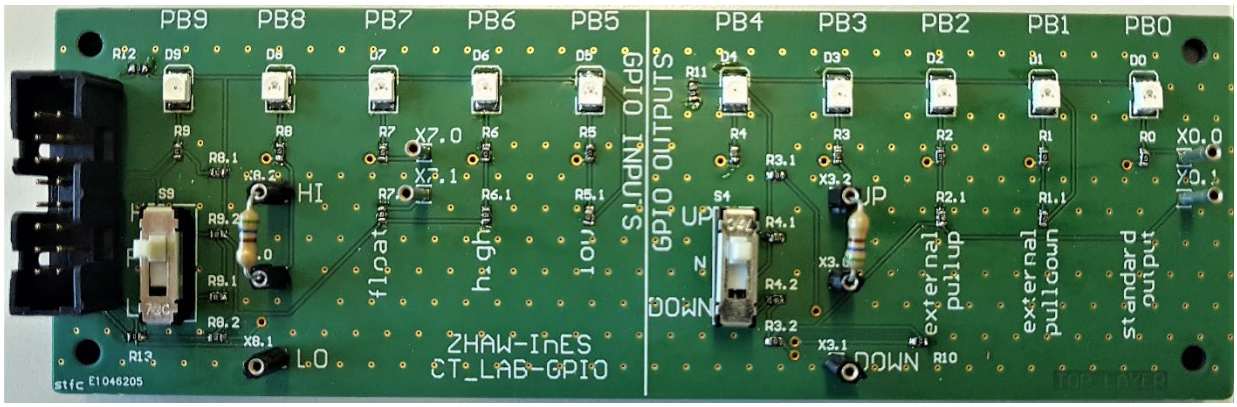


Abbildung 4: GPIO Zusatzboard mit 2 Widerständen

7.2 Schema GPIO Zusatzboard Pin X0.0 / X0.1

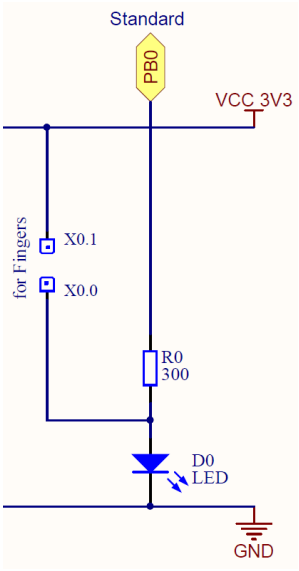


Abbildung 5: Fingerpins X0.0 / X0.1

7.3 Schema Zusatzboard

INPUT GPIO

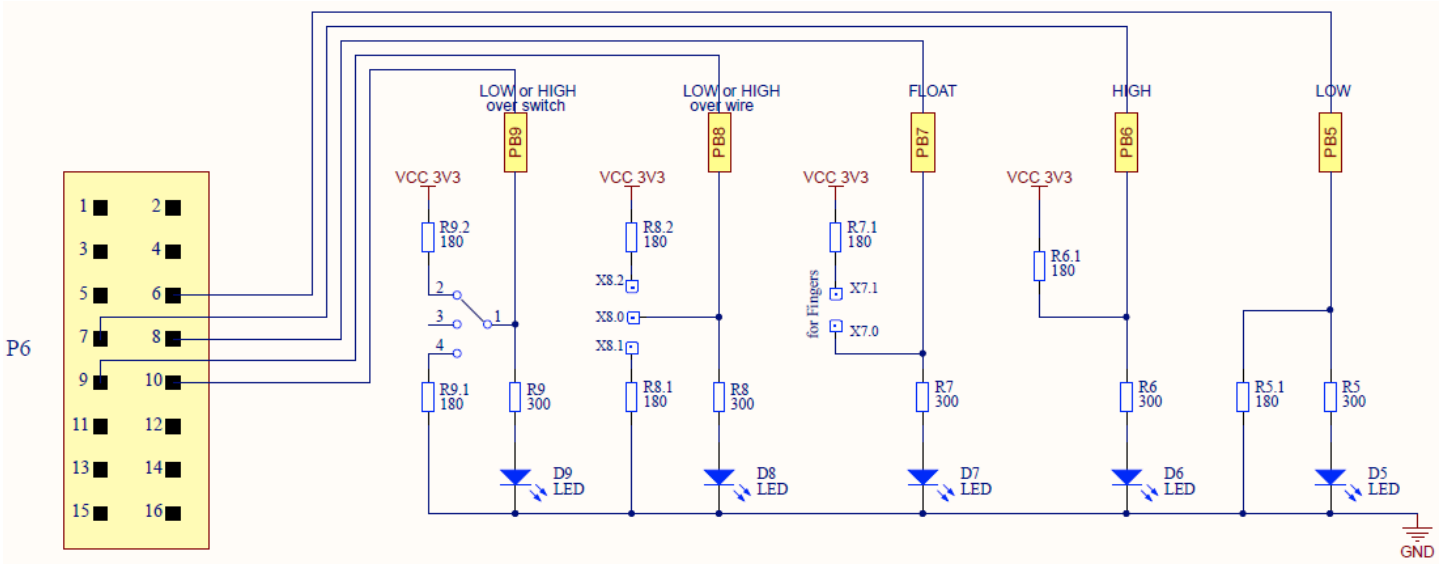


Abbildung 6: Zusatzboard Schema Input GPIO PB9_PB5

OUTPUT GPIO

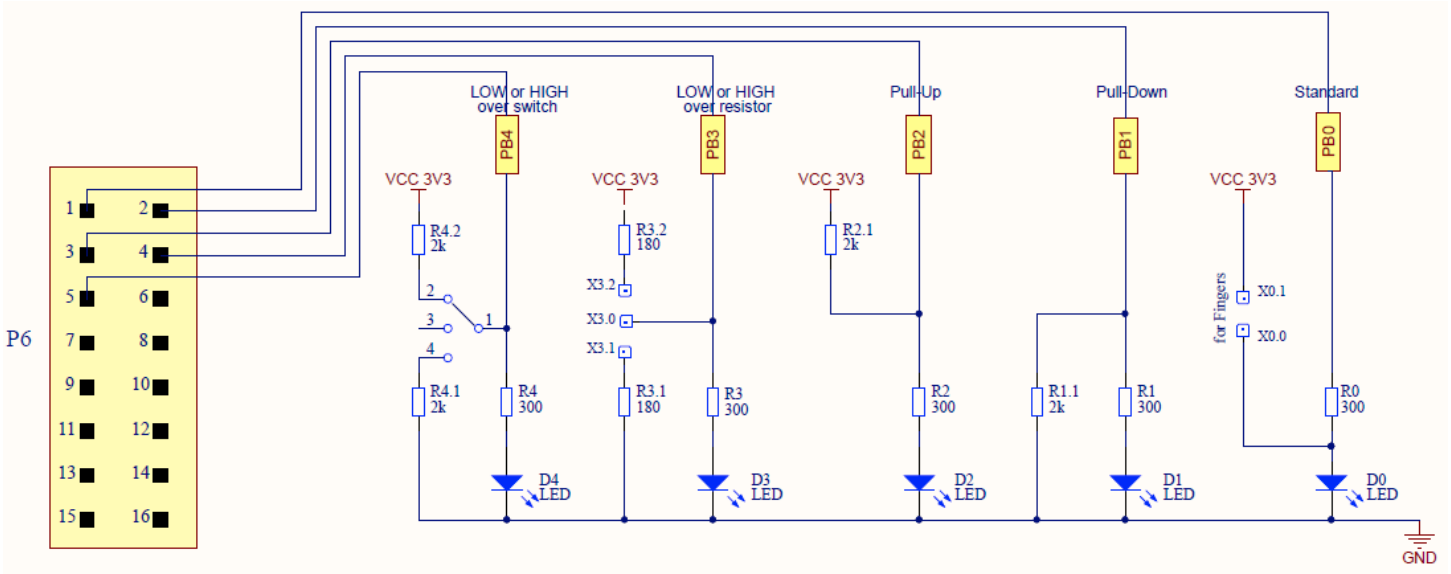


Abbildung 7: Zusatzboard Schema Output GPIO PB4_PB0

7.4 Debugging

Bei GPIOA und GPIOB sind nur die Pins 11-0 direkt abgreifbar. Die Pins 15-12 werden intern vom Discovery Board verwendet für z.B. Verbindung zum Debugger.

Falls die Pins 15-12 um konfiguriert werden, kann nicht mehr über die Debug-Schnittstelle auf den Microcontroller zugegriffen werden => Microcontroller nicht mehr um programmierbar. Um diesen Zustand zu verlassen, muss der Microcontroller anders booten (aus dem System Memory anstatt dem Flash Memory). Danach ist der Zugriff über die Debug-Schnittstelle wieder möglich.

Jetzt müssen die Registerzugriffe angepasst werden, die die Umkonfiguration der Pins 15-12 zufolge hatte, ansonsten ist die Debug-Schnittstelle beim nächsten Reset wieder nicht verfügbar!

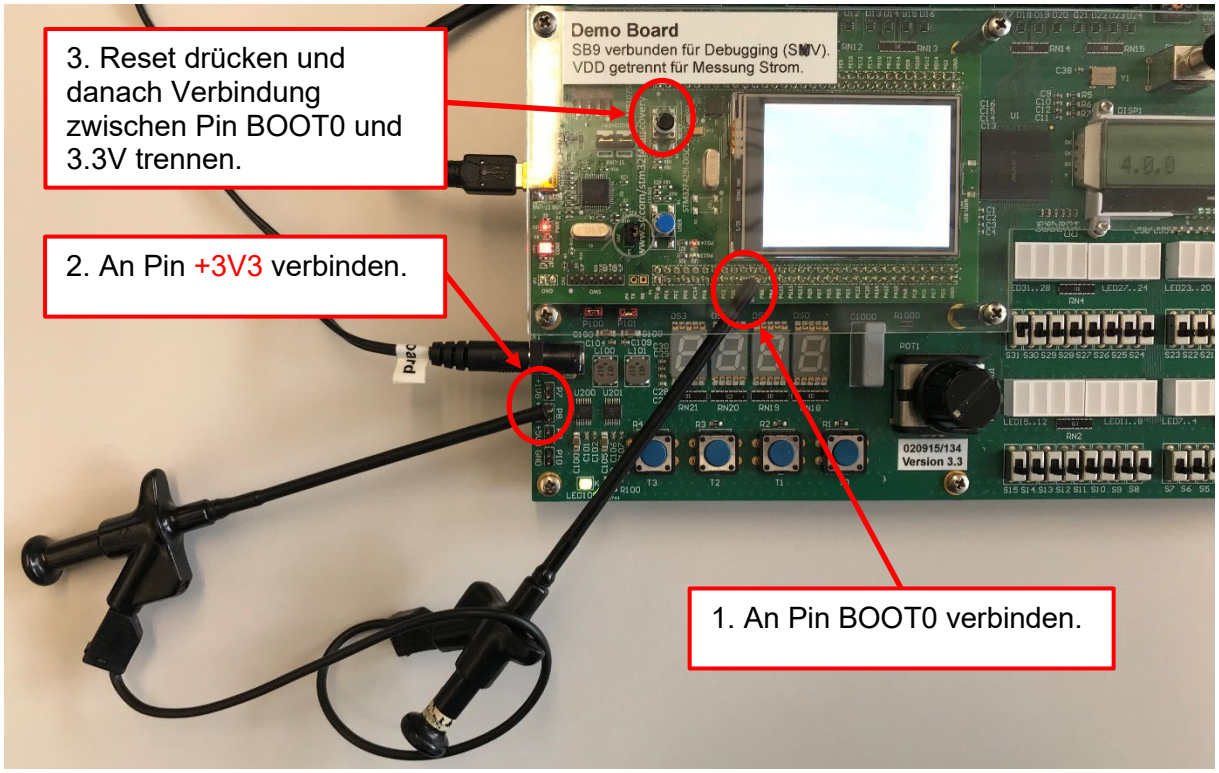


Abbildung 8: Notwendige Verbindung um aus dem System Memory zu booten