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 Registern (Abbildung 1). Die Datei reg_stm32f4xx.h definiert Strukturen und Makros für diese Registerzugriffe.

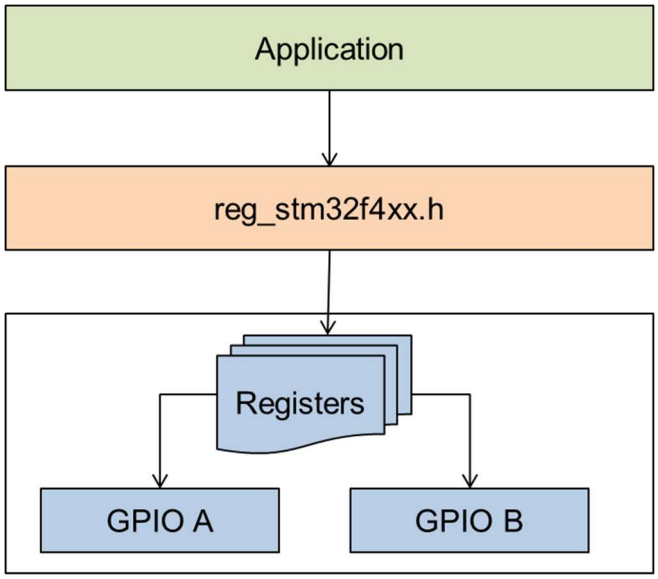


Abbildung 1: Zugriff auf GPIO mittels Register

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.
- Ihr Wissen zu den verschiedenen Konfigurationsoptionen ist gefestigt.
- 'Sie können die Strukturen des untersten HAL Layers (Basisadressen, Macros und structs in reg_stmf4xx.h) zur Konfiguration von Registern in eigenen Programmen einsetzen.

3 Material

- 1x CT Board
- 3x Verbindungsdrähte (female-female)

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?

   _____

5 Applikation und Testaufbau mit den Verbindungsdrähten

GPIO A und B sind direkt auf die Ports P5 und P6 des CT Boards herausgeführt. Die Pinbelegung für GPIO B / Port P6) ist in Abbildung 2 dargestellt, das Pin-Mapping von GPIO A auf P5 ist identisch. Details finden sich im CT-Board Wiki / GPIO (https://ennis.zhaw.ch).

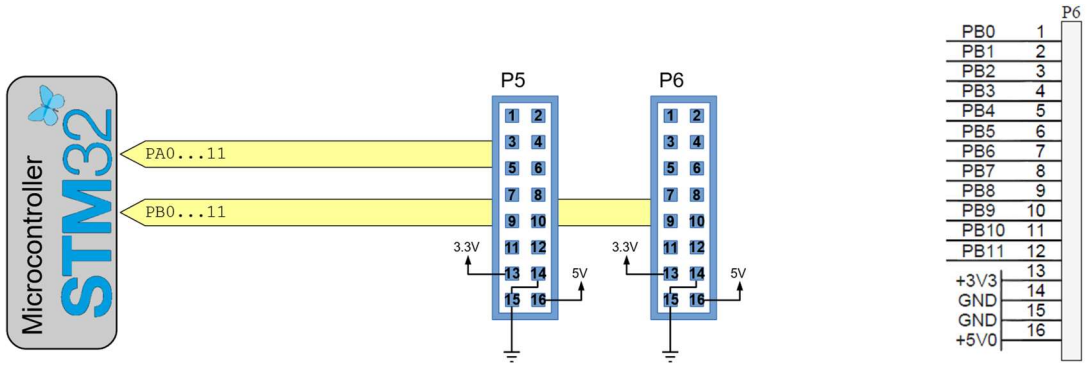


Abbildung 2: Pin-Mapping GPIO A und B auf Ports P5 und P6

In diesem Versuch werden Sie die Position der drei DIP-Schalter S10..S8 einlesen, die Werte über GPIO PB2 .. PB0 ausgeben und über GPIO PA2 .. PA0 wieder einlesen und die eingelesenen Informationen auf LED18 .. LED16 anzeigen. Zusätzlich verwenden wir LED10 .. LED8 und LED2 .. LED0 zu Debugging Zwecken. Abbildung 3 zeigt den Signalfluss

Hierzu müssen die GPIO Ports entsprechend als Inputs und Outputs konfiguriert werden.

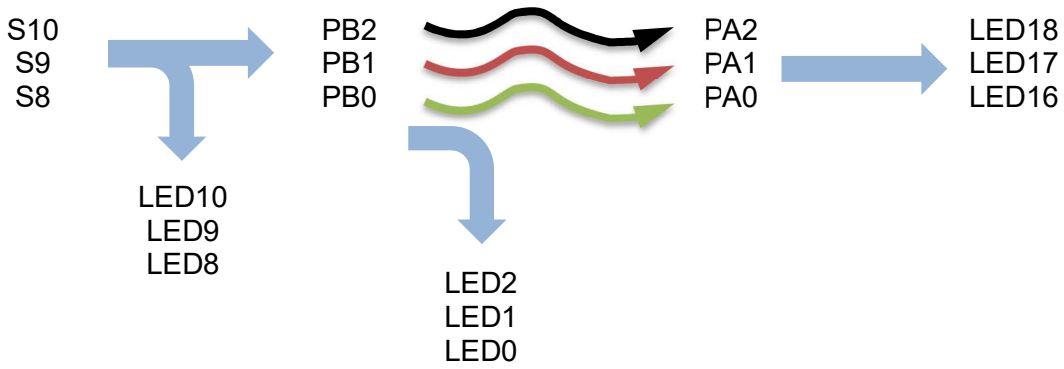


Abbildung 3: Signalfluss

Stellen Sie noch keine Verbindung zwischen den Pins auf dem CT Board her!

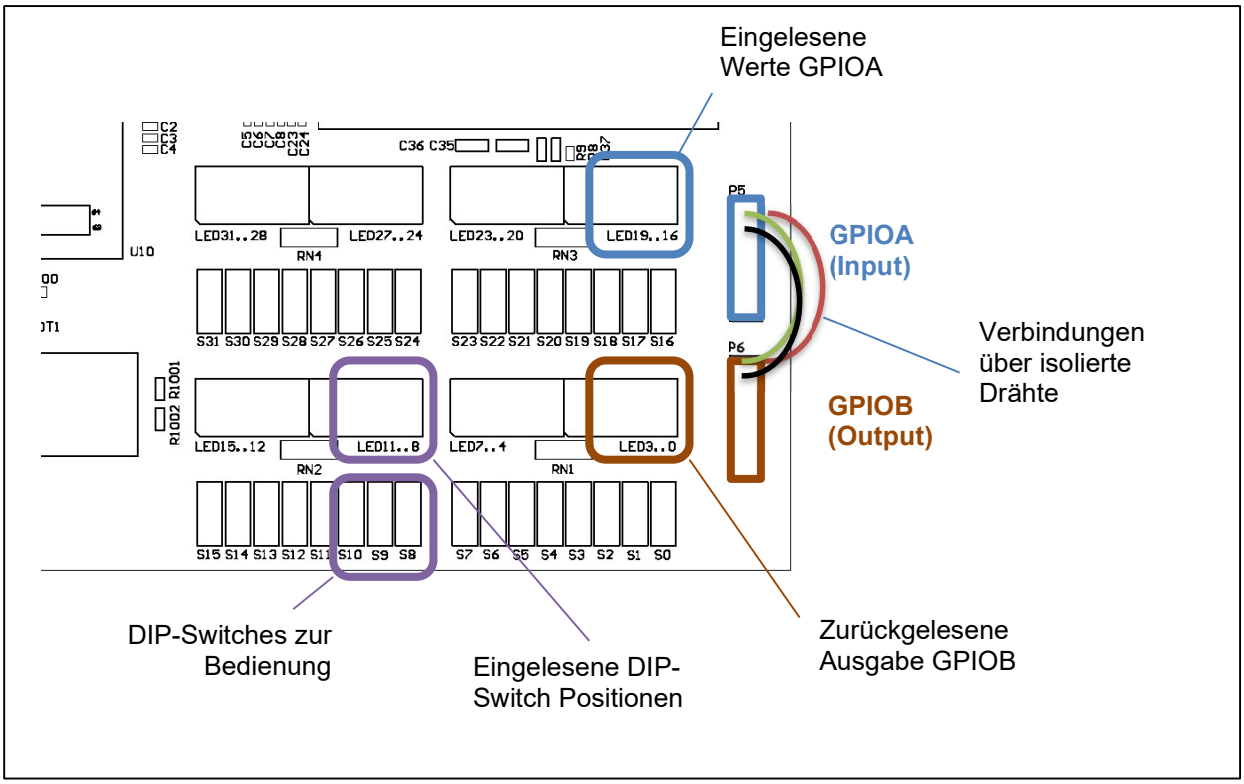


Abbildung 4: Im Versuch verwendete Elemente des CT-Boards

6 Aufgaben

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.

Um die Eigenschaften verschiedener Konfigurationen für GPIO zu untersuchen, sollen die Ein- und Ausgänge unterschiedlich konfiguriert werden:

PB0	Push-Pull Output	Kein Pull-up/-down	Low Speed
PB1	Open Drain Output	Kein Pull-up/-down	Medium Speed
PB2	Open Drain Output	Pull-up	High Speed
PA0	Input	Pull-down	
PA1	Input	Pull-up	
PA2	Input	Kein Pull-up/-own	

6.1 Konfiguration der Eingänge PA2 .. PA0

Überlegen Sie, welche Werte für die Konfiguration der Eingänge in die Konfigurationsregister von Port GPIO A geschrieben werden müssen.

Füllen Sie dazu die untenstehende Tabelle aus (Sie haben in der Vorlesungs-Übungsaufgabe zur GPIO Konfiguration zwei unterschiedliche Arten kennen gelernt, wie die Masken für das Löschen und Setzen der Bits definiert werden können).

	Port Register Name	Register Bits (Nr)	Bit Werte (Binär)	Clear/Set Maske (shift, invert)	Clear/Set Maske (absolut, hex)
Direction / Mode	PA0	Input			
	MODER	1..0	00	clr: ~(0x03 << 0)	clr: 0xFFFFFFFC
				set: 0x00 << 0	set: 0x00
	PA1	Input			
				clr:	clr:
				set:	set:
	PA2	Input			
				clr:	clr:
				set:	set:
Pull-up / Pull-down	PA0	Pull-down			
				clr:	clr:
				set:	set:
	PA1	Pull-up			
				clr:	clr:
				set:	set:
	PA2	No Pull-up/down			
				clr:	clr:
				set:	set:

Konfigurieren Sie die GPIOs im vorgegebenen Programmrahmen und lesen Sie in der Endlosschleife im Hauptprogramm die Werte von PA2 .. PA0 ein und zeigen diese auf LED18 bis LED16 an. Verwenden Sie hierzu **`struct CT_LED->BYTE`**

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

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

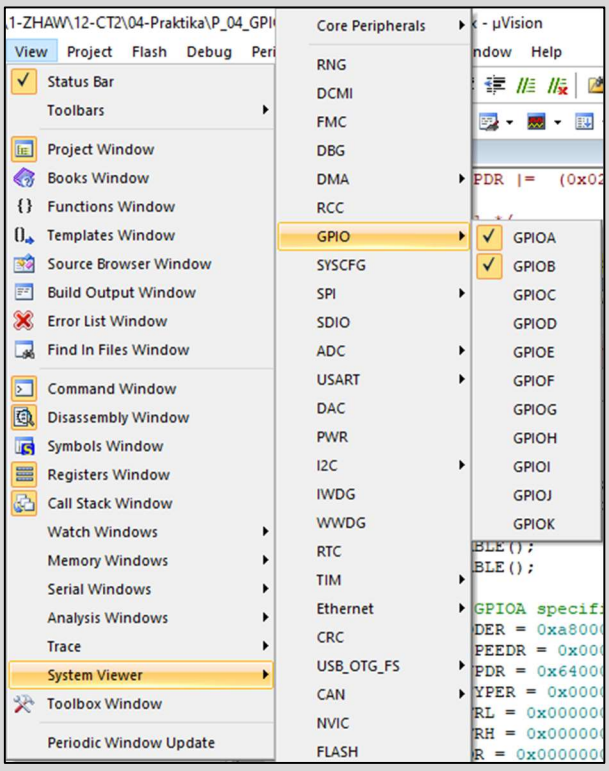
a) Bevor Sie das Programm ausführen: Welche LEDs erwarten sie bei der Ausführung hell, welche dunkel?

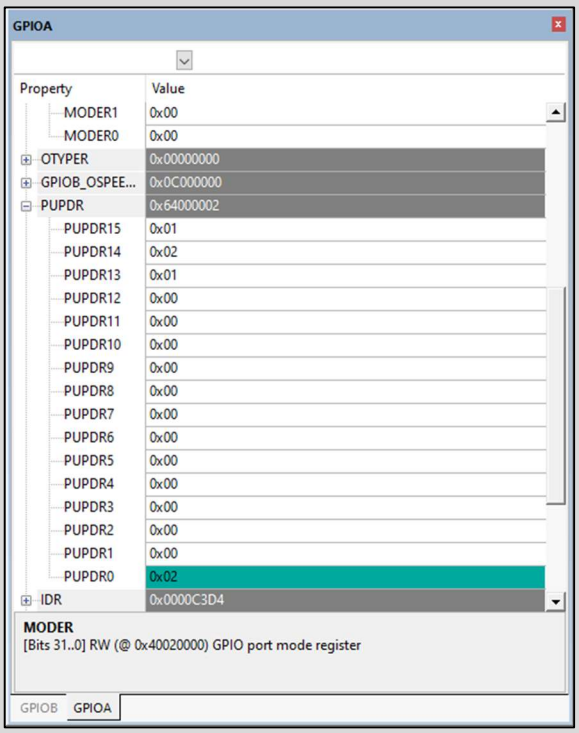
Hell: Dunkel:

b) Testen Sie Ihr Programm auf dem CT-Board. Entspricht das Ergebnis Ihren Erwartungen?

c) Streichen Sie mit dem Finger über die Pins von P5 und beobachten Sie das Verhalten der LEDs. Wie erklären Sie sich dieses Verhalten?

Sie können in der Keil uVision Entwicklungsumgebung die Werte der verschiedenen GPIO Register anschauen, indem Sie im Debugger Menü View -> System Viewer -> GPIO -> GPIOA auswählen und dann im Debug-Modus durch ihr Programm steppen. Im Beispiel ist für PA0 Pull-down gesetzt.





Achtung: Das Menu erscheint erst im Debugger.

6.2 Ausgabe über GPIO PB2 .. PB0

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.

	Port Register Name	Register Bits (Nr)	Bit Werte (Binär)	Clear/Set Maske (shift, invert)	Clear/Set Maske (absolut, hex)
Direction / Mode	PB0	Gen. Purpose Output			
				clr:	clr:
				set:	set:
	PB1	Gen. Purpose Output			
				clr:	clr:
				set:	set:
	PB2	Gen. Purpose Output			
				clr:	clr:
				set:	set:
Type	PB0	Push-Pull			
				clr:	clr:
				set:	set:
	PB1	Open Drain			
				clr:	clr:
				set:	set:
	PB2	Open Drain			
				clr:	clr:
				set:	set:
Pull-up / Pull-down	PB0	No Pull-up/-down			
				clr:	clr:
				set:	set:
	PB1	No Pull-up/-down			
				clr:	clr:
				set:	set:
	PB2	Pull-up			
				clr:	clr:
				set:	set:
Speed	PB0	Low			
				clr:	clr:
				set:	set:
	PB1	Medium			
				clr:	clr:
				set:	set:
	PB2	High			
				clr:	clr:
				set:	set:

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

Konfigurieren Sie PB2 ..PB0 als Output wie in der Tabelle vorbereitet.

Erweitern Sie die Implementierung in der Endlosscheife im Hauptprogramm so, das Sie die Wert der DIP Switches S10 .. S8 einlesen und sowohl auf LED10 .. LED8 ausgeben als auch in das **ODR** (Output Data Register) schreiben.

Lesen Sie den ausgegebenen Wert aus dem **ODR** wieder zurück und geben Sie ihn auf LED2 .. LED0 aus.

LED2 .. LED0 sollten nun die Position von S10 .. S8 anzeigen, gleich wie LED10 .. LED8.

6.3 Vervollständigen des Aufbaus

Trennen Sie während des Anbringens der Drähte das CT Board von der Versorgungsspannung.

Bei gröberen Fehlern kann Ihr CT-Board Board Schaden nehmen und zerstört werden. (s. auch Kapitel 7.1 - Debugging).

Verbinden Sie die jeweils entsprechenden Pins über die Verbindungsdrähte, also PA0 mit PB0, PA1 mit PB1 und PA2 mit PB2.

Schalten Sie die DIP Switches S10 - S8 und prüfen Sie, ob die Übertragung über die Verbindungsdrähte funktioniert. Stimmen die Anzeigen auf allen LEDs miteinander überein?

6.4 Untersuchung verschiedener I/O Kombinationen

Durch Umstecken der Verbindungsdrähte können Sie nun unterschiedlich konfigurierte Aus- und Eingänge miteinander verschalten. Prüfen Sie alle Kombinationen und tragen die Ergebnisse in die folgende Tabelle ein. Zur Stabilitätsprüfung können Sie wieder die «Fingerprobe» vornehmen (wenn ihr Finger schmal genug ist, dass sie die Pins von P5 von unten berühren können).. Für die Kombination Push-Pull / Pull-Down sind die Ergebnisse bereits eingetragen.

	LED16: Pull-Down	LED17: Pull-Up	LED18: No Pull
S8: Push-Pull, NO pull-up/down	0: **0** 1: **1**	0: 1:	0: 1:
S9: Open Drain, NO pull-up/down	0: 1:	0: 1:	0: 1:
S10: Open Drain, PULL-UP	0: 1:	0: 1:	0: 1:

Gibt es Kombinationen, die Sie kritisch sehen?

Bewertung

Bewertungskriterien	**Gewichtung**
Die Funktionen für die GPIO Inputs wurden gemäss Aufgabe implementiert und die Fragen können erklärt werden.	1/4
Die Output-Konfiguration ist implementiert und getestet.	1/4
Die Funktionen für die GPIO Outputs wurden gemäss Anforderungen getestet.	1/4
Die Untersuchung wurde komplett durch geführt und auffällige Ergebnisse können erklärt werden.	1/4

7 Anhang

7.1 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 umprogrammierbar. 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!

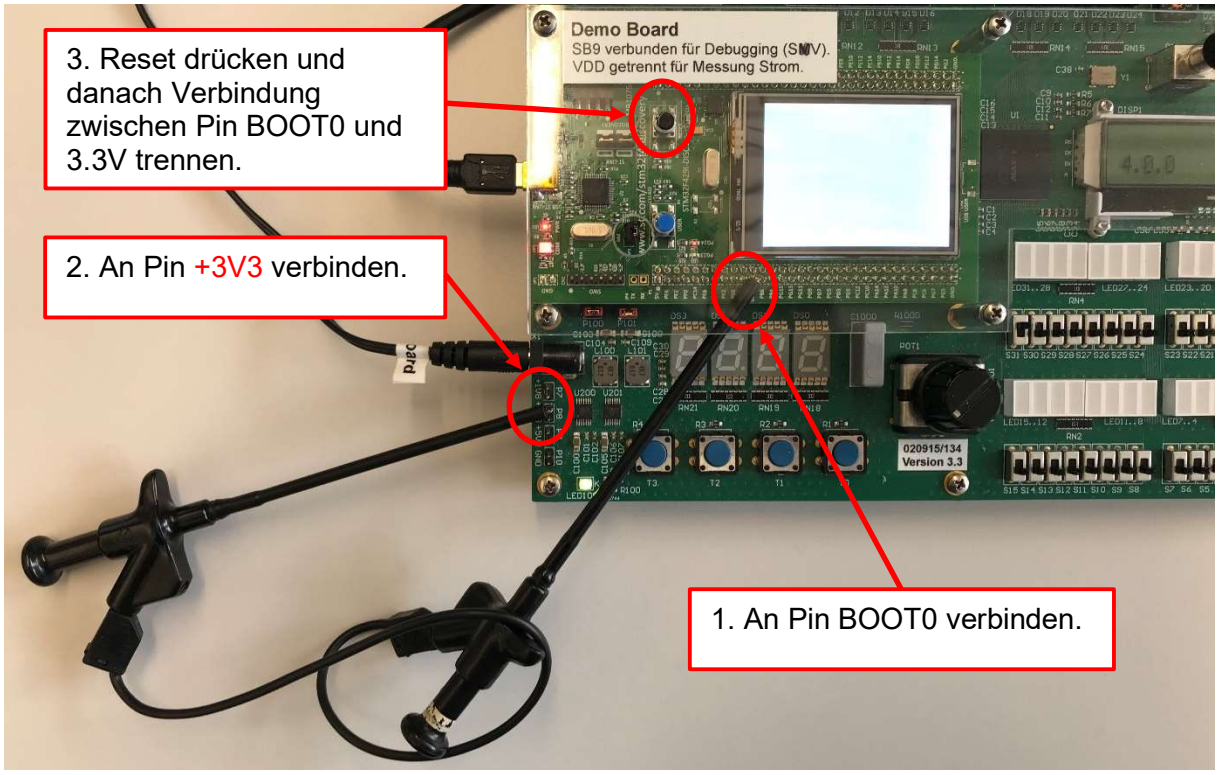


Abbildung 4: Notwendige Verbindung, um aus dem System Memory zu booten