School of Engineering
InES Institute of Embedded Systems

MC1

Power Modes

1 Einleitung

Sie werden in diesem Praktikum die Power Modi des Microcontrollers und deren Initialisierung kennenlernen. Informationen zum Microcontroller, insbesondere der in diesem Praktikum verwendeten Power Modi, finden Sie auf https://ennis.zhaw.ch.

- Verwenden Sie wenn immer möglich die Funktionen aus dem HAL (Hardware Abstraction Layer). Wo keine Funktionen im HAL vorhanden sind, schreiben oder lesen Sie direkt von den Registern.

2 Lernziele

- Sie können verschiedene Power Modi in eigenen Programmen anwenden.
- Sie verstehen, wie die Power Modi die Energieaufnahme beeinflussen.
- Sie können den Wake-up Timer im RTC einsetzen, um in regelmässigen Abständen aus einem Low-power Mode aufzuwachen.

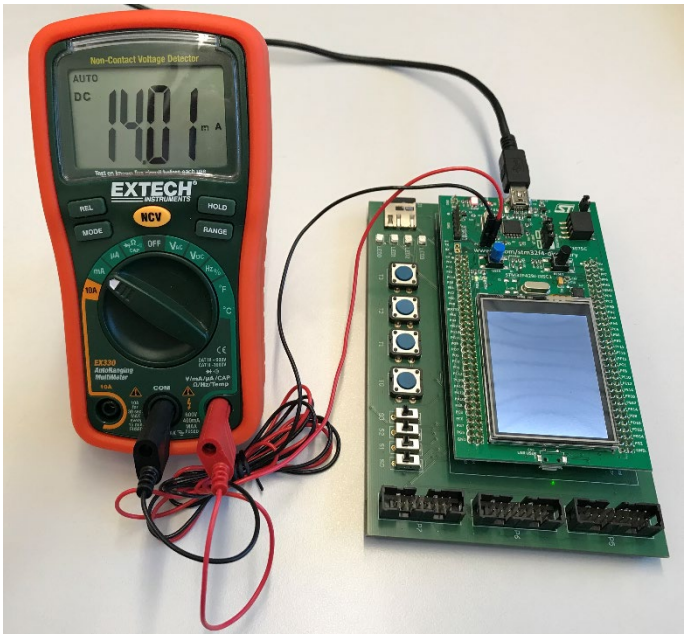

Abbildung 1: Messaufbau

3 Aufgabe – Stromaufnahme messen

Verwenden Sie den vorgegebenen Programmrahmen *code_power* und das bereitgestellte Discovery Board Modul. In dieser Aufgabe soll die Stromaufnahme des Microcontrollers gemessen werden.

User Button	GPIO Port A.0
LED Grün (LD3)	GPIO Port G.13
LED Rot (LD4)	GPIO Port G.14

Auf dem Discovery Board ist ein Anschluss vorhanden, um eine Strommessung durchzuführen. Entfernen Sie den Jumper JP3 und schliessen Sie das Multimeter an diesen Pins an (Siehe Abbildungen 1 und 2).

Schalten Sie das Multimeter ein und stellen Sie den Bereich auf *mA*.

Achten Sie darauf, den Jumper nach der Messung wieder einzusetzen. Der Flash Download funktioniert nur mit eingesetztem Jumper JP3 oder mit angeschlossenem Multimeter.

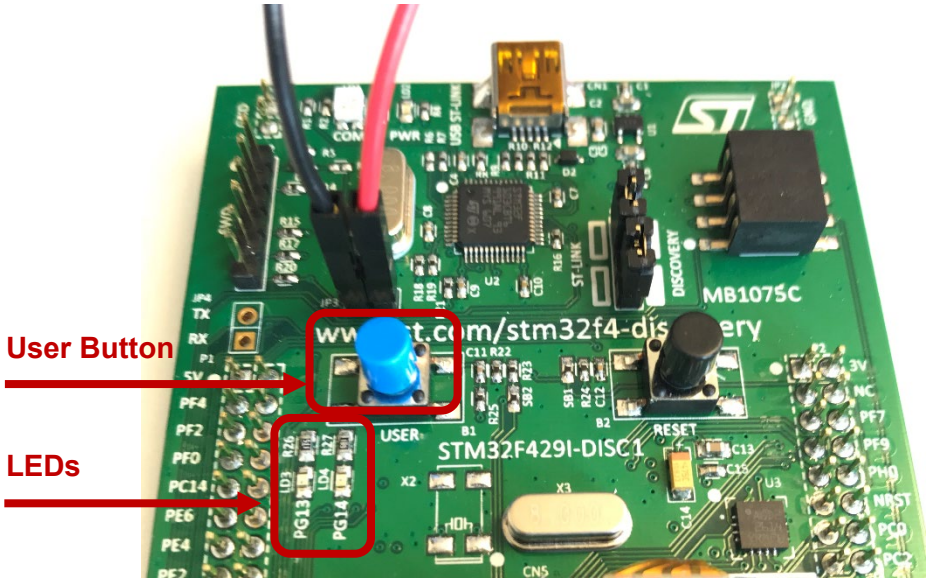

Abbildung 2: Anschluss Strommessung

3.1 Basisfunktionalität Finite State-Machine (FSM)

Der gegebene Programmrahmen *code_power* enthält eine FSM. Wir verwenden diese FSM, um den Stromverbrauch in unterschiedlichen Power Modi zu messen.

Die FSM soll zyklisch bei jedem Tastendruck auf dem User Button den Zustand wechseln:

ON_STATE → RUN_ STATE → SLEEP_ STATE → STOP_ STATE → ON_ STATE ...

Bei jedem Zustandswechsel werden die beiden LEDs umgeschaltet um den aktuellen Zustand anzuzeigen.

Für die Abfrage des Tastendrucks im main-loop ist in *user_button.h* eine Funktion deklariert. Implementieren Sie diese und den zugehörigen Interrupt Handler im Modul *user_button.* Vergessen Sie nicht, das Interrupt Flag im Interrupt Handler zu löschen.

Messen Sie die Stromaufnahme in jedem Zustand und tragen Sie die Resultate in der Tabelle im Anhang ein.

3.2 Sleep Mode

Implementieren Sie im Modul *power_mode* die vorbereitete Funktion, um in den *'Sleep Mode'* zu gelangen.

Anschliessend messen Sie die Stromaufnahme im *'Sleep Mode'* und überprüfen, dass diese signifikant tiefer ist als vorher.

- Selektieren Sie den *'Sleep Mode'* indem Sie das SLEEPDEEP Bit im System Control Register (SCR) des System Control Blocks (SCB) auf '0' setzen. Zugriff über **`SCB->SCR`**. Beim SCB handelt es sich um ein sogenanntes *'Core Peripheral'* des ARM Cortex-M. D.h. der Block stammt nicht von STMicroelectronics sondern gehört zum ARM Prozessorkern. Sie finden die Beschreibung des Registers (SCR) deshalb im ***'Programming Manual'*** und nicht im ***'Reference Manual'***.
- Der Wechsel in den Low Power Mode erfolgt mit der Assembler Instuktion **`wfi`**. Verwenden Sie folgenden Ausdruck:
  ```
  __asm volatile („wfi“);
  ```
- Der Wakeup erfolgt "automatisch" beim nächsten Drücken des User Buttons über den Interrupt an Port A.0.

3.3 Stop Mode

Implementieren Sie im Modul *power_mode* die vorbereitete Funktion um in den *'Stop Mode'* zu gelangen.

- Selektieren Sie den *'Stop Mode'* indem Sie das SLEEPDEEP Bit auf '1' setzen.
- Setzen/löschen Sie die Bits im *'Configuration Register'* im *'Power Controller'* (**`PWR->CR`**), so dass im *'Stop Mode'* gilt
  - Low power regulator on
  - Flash power down
  - Stop (not standby) when SLEEPDEEP Bit is set

- Nach dem Aufwachen aus dem *'Stop Mode'* muss die System Clock wieder neu konfiguriert werden, da der Microcontroller automatisch den HSI RC Oszillator als Clock Quelle auswählt. Das Konfigurieren ist vorgegeben und erfolgt über den Aufruf **`power_set_clock()`** in der FSM.

3.4 Anpassen der Clock Frequenz

Implementieren Sie im Modul *power_mode* die Einstellungen für die weiteren Frequenzen. Messen Sie die Ströme in allen Zuständen und Clockeinstellungen und füllen Sie die zweite Tabelle im Anhang aus.

4 Aufgabe – Wake Up Timer

In dieser Aufgabe messen Sie die Zeit, welche die CPU zum Erledigen einer bestimmten Arbeit braucht. Zusammen mit den Messungen aus Aufgabe 3 bestimmen Sie anschliessend die Energieaufnahme. Um automatisiert alle 100 ms aus dem Low Power Zustand aufzuwachen, wird der 'Wake Up Timer' verwendet.

4.1 Konfiguration Wake Up Timer

Implementieren Sie die Funktion **`wakeup_init()`** im Modul *wakeup_timer*. In dieser Funktion wird der 'Wake Up Timer' so konfiguriert, dass er alle 100 ms ein Interrupt erzeugt (WUTF) um das System aufzuwecken.

- Der *'Wake Up Timer'* ist Teil des *'Real Time Clock'* (RTC). Verwenden Sie für die Programmierung des 'Wake Up Timers' die Beschreibung auf https://ennis.zhaw.ch

Implementieren Sie ebenfalls im Modul *wakeup_timer* den Interrupt Handler für den Wake Up Interrupt. Einzige Funktion des Interrupt Handlers ist das Zurücksetzen der Interrupt-Bedingung. Dafür müssen sowohl das Interrupt Flag im *'Wake Up Timer'* als auch das Pending Register (PR) Flag im *'EXTI'* Block gelöscht werden.

4.2 Implementation Hauptprogramm

Implementieren Sie am vorbereiteten Ort in *main.c* die folgende Endlos-Schleife:

- Grüne LED ein, rote LED aus
- Ausführen von **`work_hard()`** aus Modul *work*
- Grüne LED aus
- Wechsel in Stop Mode
- Aufwachen aus dem Stop Mode (über den Interrupt des 'Wake Up Timers')
- Setzen der Clocks

4.3 Messen

Am Pin der grünen LED können Sie mit dem Oszilloskop die Wachphase sehen. Das Discovery sollte alle 100 ms aufwachen, **`work_hard()`** ausführen (Run Mode) und dann wieder in den Stop Mode gehen.

Messen Sie die Zeiten für verschiedene System Clocks und tragen Sie die Ergebnisse in Tabelle 3 ein.

Tragen Sie die Formel für die Energieberechnung ein.

Vervollständigen Sie diese Tabelle indem Sie die Ergebnisse aus Aufgabe 3 für Ihre Berechnungen verwenden.

5 Bewertung

Das Praktikum wird mit maximal 3 Punkten bewertet:

- Aufgaben 3.1 und 3.2 (FSM und Sleep Mode) 1 Punkt
- Aufgaben 3.3 und 3.4 (Stop Mode und Clock Frequenzen) 1 Punkt
- Aufgabe 4 ('Wake Up Timer') 1 Punkt

Punkte werden nur gutgeschrieben, wenn die folgenden Bedingungen erfüllt sind:

- Der Code muss sauber strukturiert und kommentiert sein.
- Das Programm ist softwaretechnisch sauber aufgebaut.
- Die Funktion des Programmes wird erfolgreich vorgeführt.
- Der/die Studierende muss den Code erklären und zugehörige Fragen beantworten können.

6 Anhang

6.1 Messergebnisse Stromaufnahme

System Clock [MHz]	Stromaufnahme [mA] bei 3V			
	ON_STATE	RUN_ STATE	SLEEP_ STATE	STOP_ STATE
168	34.14	33.39	33.22	32.14

Tabelle 1: Stromaufnahme FSM (Aufgabe 3.1)

System Clock [MHz]	Stromaufnahme [mA] bei 3V			
	ON_ STATE	RUN_ STATE	SLEEP_ STATE	STOP_ STATE
168	34.35	33.33	20.14	3.67
120	30.88	29.78	16.49	3.68
84	23.93	22.75	13.01	3.66
60	19.14	17.88	10.71	3.66

Tabelle 2: Stromaufnahme Aufgabe 3.3 und 3.4

6.2 Ergebnisse 'Wake Up Timer'

Notieren Sie die Formel für den Energiebedarf für einen Zyklus:

$E_{TOT} =$ (t_arbeit*i_run+t_stop*i_stop)*U

System Clock [MHz]	Arbeitszeit [ms]	I_{Run} [mA]	Zeit in Stop [ms]	I_{STOP} [mA]	Energie [mWs]
168	28.4	31.1	99.2	3.6	4.093
120	40.8	24.7	88.4	3.6	4.376
84	56.4	19.0	72.0	3.6	4.392
60	80.0	15.0	48.0	3.6	4.53

Tabelle 3: Energieaufnahme