INTERRUPTS

INDINF08

12. JANUAR 2015

STEFAN ERCEG
4AHITT

Inhalt

1. Aufgabenstellung 2
2. Zeitabschätzung 2
3. Tatsächlicher Zeitaufwand 2
4. Arbeitsdurchführung 3
4.1 Allgemeines 3
4.2 Änderungen an der Datei „main.c“ 4
4.3 Änderungen an der Datei „stm32f30x_it.c“ 5
4.4 Anzeige des Ergebnisses 6
5. Lessons learned 7
6. Quellenangaben 7

Github-Link: https://github.com/serceg-tgm/4AHITT-SYT/tree/indinf/IndInf08

Github-Tag: erceg_indinf08

1. Aufgabenstellung

Schreiben Sie die Aufgabe IndInf07 um, sodass Interrupts verwendet werden und die Hauptroutine nur while(1); als "wiederkehrende" Anweisung enthält.

2. Zeitabschätzung

Teilaufgabe	benötigte Zeit
Programm implementieren	100 Minuten
Protokoll schreiben	30 Minuten
Gesamt	**130 Minuten (2 h 10 min)**

3. Tatsächlicher Zeitaufwand

Teilaufgabe	benötigte Zeit
Programm implementieren	60 Minuten
Protokoll schreiben	35 Minuten
Gesamt	**95 Minuten (1 h 35 min)**

Der tatsächliche Zeitaufwand für die Übung betrug um 35 Minuten weniger als der geplante. Da uns bereits einiges an Wissen über Interrupts während des Unterrichts beigebracht wurde, fiel es nicht bedeutend schwer, diese Übung durchzuführen.

4. Arbeitsdurchführung

4.1 Allgemeines

Zu Beginn wurde das Github-Repository vom Herr Professor Borko [1] geklont:

```
git clone https://github.com/mborko/stm32f3-template.git
```

In diesem Repository befindet sich bereits im Verzeichnis `src/interrupt` eine Interrupt-Übung, bei der nach dem Drücken des Buttons 4 LEDs, bei denen der LED-Grad %90° gleich 0 ergibt, eingeschaltet sind. Erst nach dem erneuten Drücken des Buttons werden diese 4 LEDs wieder ausgeschaltet.

Diese Übung wird nun so umgeändert, dass beim Starten des Programms die 4 LEDs, bei denen der LED-Grad %90° gleich 0 ergibt, leuchten. Solange man auf den Button drückt, werden die 4 LEDs, bei denen der LED-Grad %90° ungleich 0 ergibt, leuchten und die anderen 4 ausgeschaltet.

Als erstes habe ich mal einen eigenen Ordner `indinf08` in meinem Projekt-Ordner erstellt, wo ich das Makefile aus dem Repository und alles, was sich im Verzeichnis `src/interrupt` befindet, kopiert habe.

Somit musste man im Makefile die Zeile `PROJ := src/blink` auf `PROJ := .` ändern, damit das richtige Programm ausgeführt wird.

4.2 Änderungen an der Datei „main.c“

```
int main(void) {

	int i;
	for(i = 0; i <= 7; i++) STM_EVAL_LEDInit(i);

	STM_EVAL_LEDToggle(LED3);
	STM_EVAL_LEDToggle(LED6);
	STM_EVAL_LEDToggle(LED7);
	STM_EVAL_LEDToggle(LED10);

	STM_EVAL_PBInit(BUTTON_USER,BUTTON_MODE_EXTI);
	EXTI0_Config();

	while (1);

}
```

Zu Beginn wurden in der main-Funktion mittels einer for-Schleife alle 8 LEDs initialisiert.

Danach werden die 4 LEDs aktiviert, bei denen der LED-Grad %90° gleich 0 ergibt. Dies wird gemacht, damit diese sofort nach dem Starten des Programms leuchten. Dazu wird die Funktion `STM_EVAL_LEDToggle(LED)` 4-mal aufgerufen.

Als nächstes wird PA0 als Interrupt-Modus aktiviert und die Funktion `EXTI0_Config()` aufgerufen, welche zur Konfiguration des Interrupts dient.

Zum Schluss der main-Funktion wird noch die wiederkehrende Anweisung `while(1);` durchgeführt.

In der Funktion `EXTI0_Config(void)`, die, wie vorher erwähnt, zur Konfiguration des Interrupts dient, wird in der Zeile

```
	EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Rising;
```

noch ein `Falling` hinzugefügt, daher sieht die Zeile folgendermaßen aus:

```
	EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Rising_Falling;
```

Somit wird das Interrupt bei Falling Edge (sinkende Flanke; von 1 auf 0) und Rising Edge (steigende Flanke; von 0 auf 1) ausgelöst und der Zustand des Buttons überprüft.

4.3 Änderungen an der Datei „stm32f30x_it.c“

```
void EXTI0_IRQHandler(void) {

	int i;
	for(i = 0; i <= 7; i++) STM_EVAL_LEDToggle(i);

	/* Clear the EXTI line 0 pending bit */
	EXTI_ClearITPendingBit(USER_BUTTON_EXTI_LINE);

}
```

In der Funktion `EXTI0_IRQHandler(void)` werden zu Beginn alle 8 LEDs mittels einer for-Schleife initialisiert.

Mit der Zeile `EXTI_ClearITPendingBit(USER_BUTTON_EXTI_LINE);` wird das für das Interrupt zuständige Bit gecleart. Somit kann beim Drücken des Buttons erneut reagiert werden.

4.4 Anzeige des Ergebnisses

Nach dem Builden und Ausführen des Programms, sind 4 LEDs, bei denen der LED-Grad %90° gleich 0 ergibt, eingeschaltet.

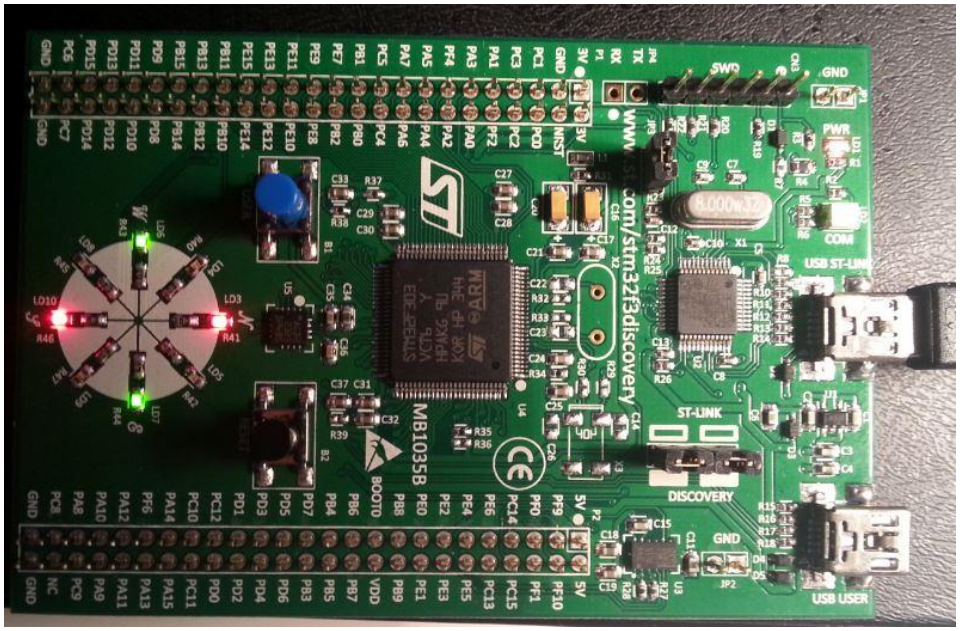

Abbildung 1: Anzeige des Mikrocontrollerboards, auf dem 4 LEDs, bei denen der LED-Grad %90° gleich 0 ergibt, leuchten [Stefan Erceg, fotografiert am 11.12.2014]

Wenn der blaue Button „USER“ nun gedrückt wird, leuchten die 4 LEDs, bei denen der LED-Grad %90° ungleich 0 ergibt. Diese 4 LEDs leuchten solange man auf den Button drückt, nach dem Loslassen leuchten die anderen 4 LEDs erneut.

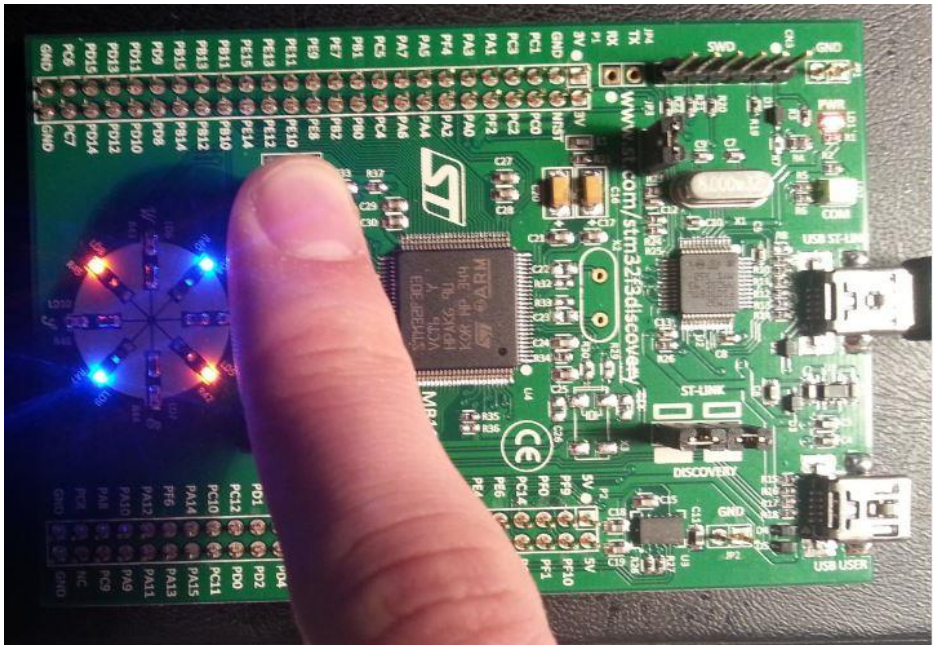

Abbildung 2: Anzeige des Mikrocontrollerboards, auf dem die anderen 4 LEDs beim Drücken des Buttons leuchten [Stefan Erceg, fotografiert am 11.12.2014]

5. Lessons learned

- Verwendung von Interrupts
- Makefiles genauer durchlesen

6. Quellenangaben

[1] Github-User „mborko” (2014, 2015). Github-Repository „stm32f3-template” [Online]. Available at: https://github.com/mborko/stm32f3-template [zuletzt abgerufen am 12.01.2015]