BUTTONSTEUERUNG

INDINF07

11. DEZEMBER 2014

STEFAN ERCEG
4AHITT

Inhalt

Github-Link: https://github.com/serceg-tgm/4AHITT-SYT/tree/indinf/IndInf07

Github-Tag: erceg_indinf07

1. Aufgabenstellung

Aktivieren Sie den Pin PA0 auf Port A um den User-Button verwenden zu können. Beachten Sie, dass dieser Pin als Input-Pin konfiguriert werden muss (Pull-Up/Down nicht vergessen!). Schreiben Sie Ihren Code der Aufgabe IndInf06 um und reagieren Sie folgendermaßen auf das Drücken des Buttons:

Button != gedrückt -> Lichter° %90° == 0 -> Licht an, Andere aus
Button == gedrückt -> Lichter° %90° == 0 -> Licht aus, Andere an

Implementieren Sie auch eine Funktion zum Toggeln der LEDs!

Dokumentieren Sie alle notwendigen Schritte und Einstellungen und schreiben Sie ein Protokoll.
Dies ist keine Gruppenarbeit!

2. Zeitabschätzung

Teilaufgabe	benötigte Zeit
Programm implementieren	60 Minuten
Protokoll schreiben	60 Minuten
Gesamt	**120 Minuten (2 h)**

3. Tatsächlicher Zeitaufwand

Teilaufgabe	benötigte Zeit
Programm implementieren	120 Minuten
Protokoll schreiben	55 Minuten
Gesamt	**175 Minuten (2h 55 min)**

Der tatsächliche Zeitaufwand für die Übung betrug aufgrund der Implementierung des Programms 55 Minuten länger als der geplante. Welche Probleme und Missverständnisse bei der Implementierung aufgetaucht sind, können Sie beim Punkt 5.2 nachlesen.

4. Designüberlegung

Folgende Designüberlegungen kamen zu Stande:

Für das Umschalten der LEDs wäre es auf jeden Fall sinnvoll, eine „toggle_LEDs“-Funktion zu implementieren, bei der im Parameter der Pin angegeben und ein Pointer der Struct des GPIO Port E übergeben wird. In der Funktion wird dann die LED auf dem Pin, welcher angegeben wurde, mit dem Output Data Register (ODR) eingeschaltet.

Nach einigen Schlussfolgerungen und Problemen (siehe Punkt 5.2) kam ich ebenfalls zum Entschluss eine „toggle_allLEDs“-Funktion zu implementieren, bei der im Parameter nur der Pointer der Struct des GPIO Port E übergeben wird. In der Funktion werden dann mittels einer for-Schleife alle LEDs mit dem Output Data Register (ODR) eingeschaltet.

In der Main-Funktion wird eine Variable „buttonState“ erstellt, welche den aktuellen Status des Buttons speichert. Falls sich der Status ändert, wird erkannt, dass der Button gedrückt wurde und die anderen LEDs auf dem Microcontrollerboard anfangen sollen zu leuchten. Ebenfalls wird der Pin PA0 auf Port A als Input-Pin konfiguriert.

5. Arbeitsdurchführung

5.1 allgemeine Implementierung

Das Projekt von IndInf06 wurde genommen und kopiert:

```
cp -r indinf06 indinf07
```

Danach wurden in der Main-Datei des IndInf07-Projekts alle Funktionen, welche für diese Aufgabe nicht notwendig sind, entfernt.

Zu Beginn der Implementierung wurden die Prototypen für die beiden Funktionen „toggle_LEDs“ und „toggle_allLEDs“ erstellt.

Die vorher erwähnten Funktionen wurden danach folgendermaßen implementiert:

```
void toggle_LEDs (int bit, GPIO_TypeDef *PE) {
	PE->ODR ^= (1 << bit);
}

void toggle_allLEDs(GPIO_TypeDef *PE) {

	int i;

	for (i = 8; i <= 15; i++) {
		PE->ODR ^= (1 << i);
	}

}
```

Bei beiden Funktionen wurden lediglich die beim Punkt 4 beschriebenen Überlegungen umgesetzt.

Danach folgte die Implementierung der Main-Funktion. Der finale Code sah folgendermaßen aus:

```
int main (void) {


	GPIO_TypeDef *PE = GPIOE;

	GPIO_TypeDef *PA = GPIOA;

	PA->PUPDR = (1 << 1);

	int i;

	for (i = 8; i <= 15; i++) {
		PE->MODER |= (1 << (i*2));
	}

	uint16_t buttonState = PA->IDR & (1 << 0);

	toggle_LEDs(9, PE);
	toggle_LEDs(11, PE);
	toggle_LEDs(13, PE);
	toggle_LEDs(15, PE);


	while(1) {

		if((PA->IDR & (1 << 0)) != buttonState) {
			toggle_allLEDs(PE);
			buttonState = PA->IDR & (1 << 0);
		}

	}

}
```

Folgende Schritte wurden durchgeführt:

- ein Pointer der Struct des GPIO Port E wurde erstellt und initialisiert
- ein Pointer der Struct des GPIO Port A wurde erstellt und initialisiert
- PUPDR, welcher als Pull-Up bzw. Pull-Down Register dient, wurde verwendet, um den Pin A0 auf Port A auf Pull-Down einzustellen [1]
- MODER, welcher als Port Mode Register dient und somit den Modus (Input/Output) einstellt, wurde verwendet, um alle LEDs als Output-Pins zu definieren
- IDR, welcher als Input Data Register dient, wurde verwendet, um den Pin A0 auf Port A als Input-Pin zu definieren [1] – dabei wird auch der Status des Buttons gespeichert, um später abfragen zu können, ob der Button gedrückt worden ist oder nicht
- die LEDs auf Pin 9, 11, 13 und 15 wurden eingeschaltet
- In der while-Schleife, welche eine Endlosschleife darstellt, wurde mit einer if-Abfrage überprüft, ob sich der Status des Buttons geändert hat. Falls dies geschehen ist, wurde die Funktion „toggle_allLEDs“ angewendet und alle LEDs eingeschaltet. Da 4 LEDs jedoch vorher schon eingeschaltet waren, werden dadurch diese 4 LEDs ausgeschaltet, die anderen 4 fangen jedoch zu leuchten an.

5.2 Probleme bei der Implementierung

Nachdem ich dachte, dass ich das Programm fertig implementiert habe, buildete ich das Programm und führte es aus. Das Problem, das auftauchte, war jedoch, dass beim Drücken des Buttons alle LEDs leuchteten.

Ein Teil des Codes in der Main-Funktion sah zu dem Zeitpunkt folgendermaßen aus:

```
// the current button state is saved
uint16_t buttonState = PA->IDR & (1 << 0);

while(1) {

	// the LEDs will toggle, when the button is pressed
	if((PA->IDR & (1 << 0)) != buttonState) {

		toggle_LEDs(8, PE);
		toggle_LEDs(10, PE);
		toggle_LEDs(12, PE);
		toggle_LEDs(14, PE);
		// the new state of the button is saved
		buttonState = PA->IDR & (1 << 0);

	} else {

		toggle_LEDs(9, PE);
		toggle_LEDs(11, PE);
		toggle_LEDs(13, PE);
		toggle_LEDs(15, PE);

	}

}
```

Das Problem war jenes: Nachdem für die ersten 4 LEDs die Funktion „toggle_LEDs“ aufgerufen wurden, besaß ich am Ende eine Binärdarstellung von 1010 1010 (jede 2. LED leuchtete). Wenn der Button dann gedrückt ist, wird diese „toggle_LEDs“-Funktion erneut mehrmals aufgerufen, allerdings genau umgekehrt, und somit besaß man eine Binärdarstellung von 0101 0101. Als diese beiden Binärdarstellungen mit einem XOR (exklusivem Oder) verknüpft worden sind, ergab das Ergebnis 1111 1111 und somit leuchteten alle LEDs [2]. Somit musste man vor dem Drücken der Buttons den alten Zustand der LEDs „vergessen lassen“.

Deshalb wurden die Befehle, die im else-Zweig vorhanden waren, vor der while-Schleife durchgeführt und eine Funktion „toggle_allLEDs“, welche alle LEDs einschaltet, implementiert, die in der if-Abfrage ausgeführt wird. Somit wurden die 4 LEDs, die vor dem Drücken des Buttons eingeschaltet waren, ausgeschaltet werden.

Der Code sah am Ende folgendermaßen aus:

```
uint16_t buttonState = PA->IDR & (1 << 0);

// the LEDs on pin 9, 11, 13 and 15 are switched on
toggle_LEDs(9, PE);
toggle_LEDs(11, PE);
toggle_LEDs(13, PE);
toggle_LEDs(15, PE);


while(1) {
	// the LEDs will toggle, when the button is pressed
	if((PA->IDR & (1 << 0)) != buttonState) {
		/* because 4 LEDs are already switched on, the use of the
function toggle_allLEDs will deactivate this 4 LEDs
		   and switch on the other 4 LEDs */
		toggle_allLEDs(PE);
		// the new state of the button is saved
		buttonState = PA->IDR & (1 << 0);
	}

}
```

5.3 Anzeige des Ergebnisses

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

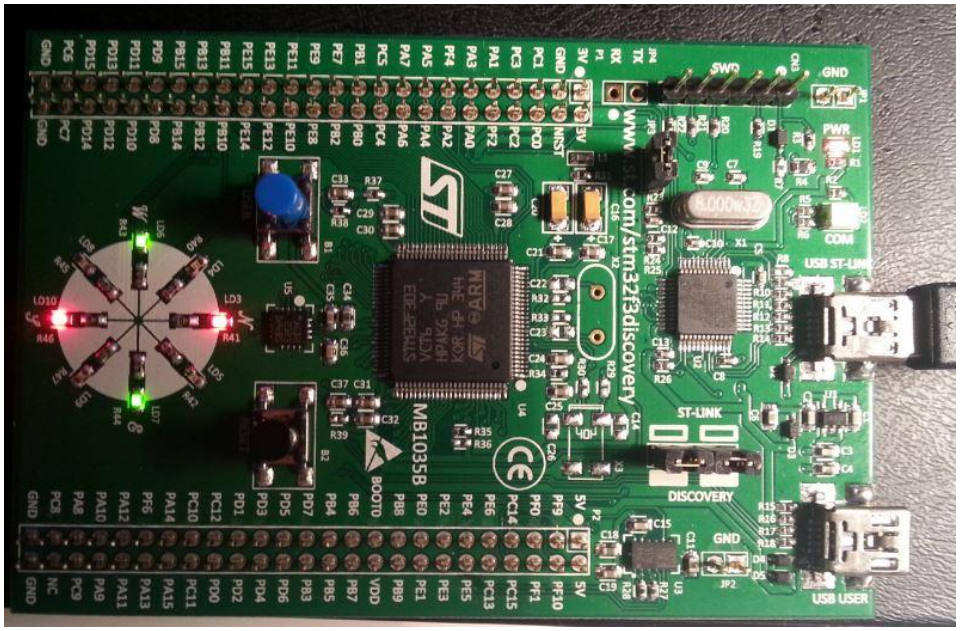

Abbildung 1: Anzeige des Mikrocontrollerboards, auf dem 4 LEDs, bei denen der LED-Grad %90° gleich 0 ergibt, leuchten [Stefan Erceg, fotografiert am 11.12.2014]

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.

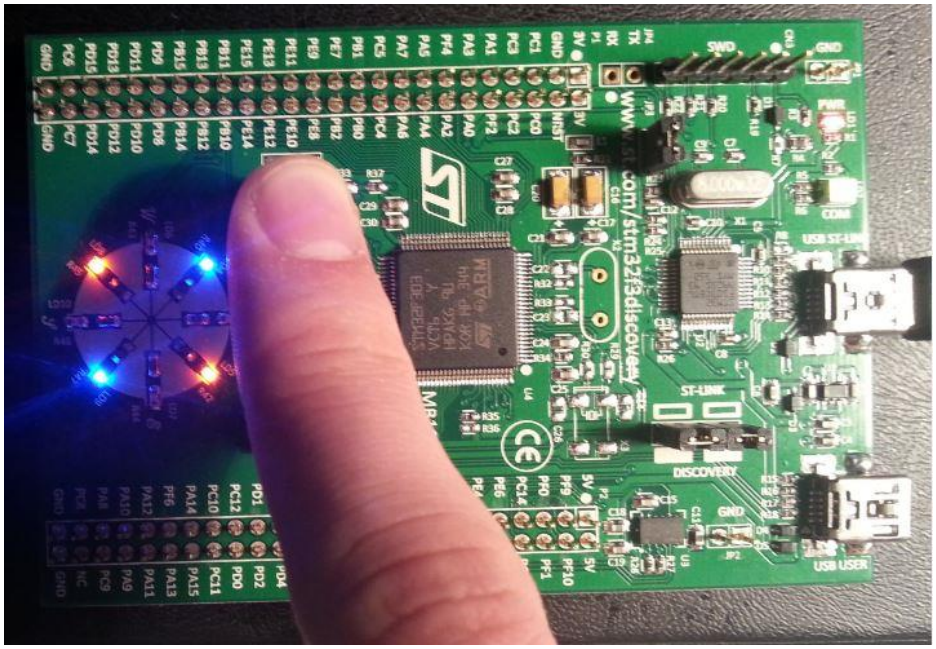

Abbildung 2: Anzeige des Mikrocontrollerboards, auf dem die anderen 4 LEDs beim Drücken des Buttons leuchten [Stefan Erceg, fotografiert am 11.12.2014]

6. Lessons learned

- Anwendung eines XOR-Operators
- PUPDR (Pull-Up/Pull-Down Register) und IDR (Input Data Register) genauer kennengelernt
- Setzen des Pull-Up/Pull-Down Registers
- Ruhe beherrschen bei bestimmten Fehlermeldungen ☺

7. Quellenangaben

[1] STMicroelectronics (May 2014). RM0316 Reference manual (S. 157) [Online]. Available at: http://www.st.com/web/en/resource/technical/document/reference_manual/DM00043574.pdf [zuletzt abgerufen am 11.12.2014]

[2] J. Wolf (2009). Openbook „C von A bis Z“ – Kap. 6.5.3 Bitweises XOR [Online]. Available at: http://openbook.galileo-press.de/c_von_a_bis_z/006_c_operatoren_005.htm#mj47da0e8682ea7d892f3113146cd64cb1 [zuletzt abgerufen am 11.12.2014]