UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TECHNISCHE INFORMATIK

Übungsblatt 11

Timer und Interrupts, A/D-Wandler

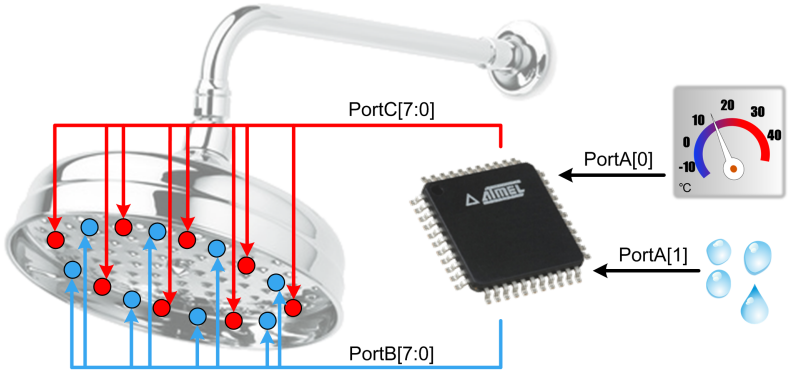
Vorlesung *Technische Grundlagen der Informatik 1*, Sommersemester 2020
Erstellt von Dr.-Ing. Kristian Ehlers

Verwenden Sie für Ihre Implementierungen einen beliebigen Editor und simulieren Sie ihren Assembler-Code anschließend im **ITImega16-Simulator**. Ihnen steht im Moodle für die Übung entsprechendes **Template** zur Verfügung.

Es seien hiermit noch folgende Anmerkung zur Nutzung des **ITImega16-Simulators** gegeben:

- Um die Namensdefinitionen wie z.B. DDRA, PORTA, PINA usw. nutzen zu können, müssen diese mithilfe des Includes `.include "m16def.inc"` bereitgestellt werden.
- Die Konfigurationsdateien der Peripherie müssen auf `layout.js` enden.
- Die Direktiven `.list` bzw. `.nolist` erlauben es, nachfolgenden Quellcode im Simulator anzuzeigen oder auszublenden.

Aufgabe 1 Assemblerprogrammierung - Duschkopf



AD-Wert	Codierung	LEDs	°C
0 – 23	0000 0000	○○○○○○○○	5°C
24 – 47	0000 0001	○○○○○○○●	10°C
48 – 71	0000 0011	○○○○○○●●	15°C
72 – 95	0000 0111	○○○○○●●●	20°C
96 –119	0000 1111	○○○○●●●●	25°C
120–143	0001 1111	○○○●●●●●	30°C
144–167	0011 1111	○○●●●●●●	35°C
168–191	0111 1111	○●●●●●●●	40°C
192–255	1111 1111	●●●●●●●●	45°C

Unter Verwendung von einem ATmega16, einem Temperatursensor, einem Feuchtigkeitsschalter und jeweils acht blauen und roten LEDs soll ein beleuchteter Duschkopf entwickelt werden. Dabei werden die unterschiedlichen LEDs abhängig von der aktuellen Wassertemperatur angesteuert und erlauben somit eine visuelle Einschätzung der Temperatur. Die Funktionsweise des Systems kann wie folgt beschrieben werden: Wenn das Wasser angestellt worden ist (PortA1 = 1), liefert der Temperatursensor (PortA0) einen korrespondierenden analogen Wert, aus dem die aktuelle Temperatur berechnet werden kann. Abhängig davon wird eine bestimmte Anzahl an roten (PortC) und blauen (PortB) LEDs vom ATmega16 angeschaltet. Dabei sind die blauen und roten LEDs invers zueinander beschaltet, d.h. eine rote LED leuchtet, sobald ein HIGH Signal

über den Port ausgegeben wird, eine blaue LED leuchtet, sobald ein LOW Signal über den Port ausgegeben wird. Diese Beschaltung erlaubt es, den gleichen Bitvektor auf die unterschiedlichen Ausgabe-Ports (PortB und PortC) auszugeben. Hat das Wasser bspw. eine Temperatur von 32°C, liefert der Temperatursensor den Wert 155 und auf PortB und PortC wird der Bitvektor 00111111 ausgegeben. Eine Übersicht über die Zusammenhänge der Analogwerte und der leuchtenden LEDs gibt die oben dargestellte Tabelle.

Verwenden Sie das im Moodle bereitgestellte Template zu dieser Aufgabe und schreiben Sie aussagekräftig kommentierten ATmega16-Assembler-Code!

(a) Sorgen Sie dafür, dass Ihnen im Code die Namensdefinitionen für die einzelnen I/O-Register zur Verfügung stehen und R22 unter dem Namen cnt genutzt werden kann.

(b) Geben Sie eine Folge von Assembleroperationen an, sodass nach einem RESET zum Label init gesprungen wird. Weiterhin soll beim Auslösen des Interrupts für Fertigstellung der A-D-Wandlung die ADC_ISR aufgerufen werden. Reservieren Sie im RAM ab der Adresse 0x0060 neun Bytes unter dem Namen codes zur Speicherung der verschiedenen LED Codierungen. Speichern Sie außerdem im Programmspeicher (ROM) ab der Adresse 0x0030 unter dem Namen ad_values die Schwellwerte der A-D-Wandlung (23, 47, 71, …).

(c) Implementieren Sie das Unterprogramm init, welches zuerst den Stackpointer initialisiert. Konfigurieren Sie weiterhin den PortA als Eingang im Zustand tri-state und PortB und PortC als Ausgang mit Initialzuständen PortB = 11111111 und PortC = 00000000. Initialisieren Sie weiterhin den A-D-Wandler mit einer Referenzspannung von 5V (Vcc) und einer linksbündigen Speicherung auf dem Kanal 0. Der A-D-Wandler soll mit einem Prescaler von clk/32 betrieben werden. Vergessen Sie auch nicht, den A-D-Wandler zu aktivieren und das entsprechende Interrupt Flag zu setzen! Aktivieren Sie die Interrupts auch global. Zum Schluss rufen Sie das Unterprogramm init_codes (d) auf und springen zum Hauptprogramm (main).

(d) Schreiben Sie das Unterprogramm init_codes, welches die Codes aus der obigen Tabelle nacheinander an die reservierten Speicherstellen codes im RAM schreibt. Beginnen Sie mit dem ersten Code „00000000“.

(e) Implementieren Sie das Unterprogramm read_adc, welches zuerst eine A-D-Wandlung startet, indem es den ATmega in den SLEEP-Modus versetzt. Nach Abschluss der A-D-Wandlung wird das HIGH-Byte des Ergebnisses in das Register R16 geschrieben und zurückgegeben.

(f) Die Interrupt Service Routine ADC_ISR soll den SLEEP-Modus des ATmegas wieder beenden. Sorgen Sie dafür, dass alle genutzten Register und auch der interne Zustand des ATmegas unverändert bleiben.

(g) Entwickeln Sie das Hauptprogramm main, welches in einer Endlosschleife abgearbeitet wird. Dabei wird zuerst ausgewertet, ob das Wasser angestellt ist (PortA[1]=1) oder nicht. Läuft momentan kein Wasser (PortA[1]=0), werden alle LEDs ausgeschaltet und zurückgesprungen. Ist das Wasser angestellt, wird als erstes eine A-D-Wandlung durchgeführt, um den aktuellen Temperaturwert zu erhalten. Dieser wird dann in einer Schleife (nutzen sie den definierten Schleifenzähler cnt) mit den Schwellwerten ad_values aus dem Programmspeicher (ROM) verglichen, um dann abhängig davon die richtige unter codes im RAM abgelegte LED Codierung zu laden. Diese muss dann nur noch auf den beiden Ausgabe-Ports ausgegeben werden, bevor zum Anfang zurück gesprungen wird.