UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TECHNISCHE INFORMATIK

Übungsblatt 10

Timer und Interrupts

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

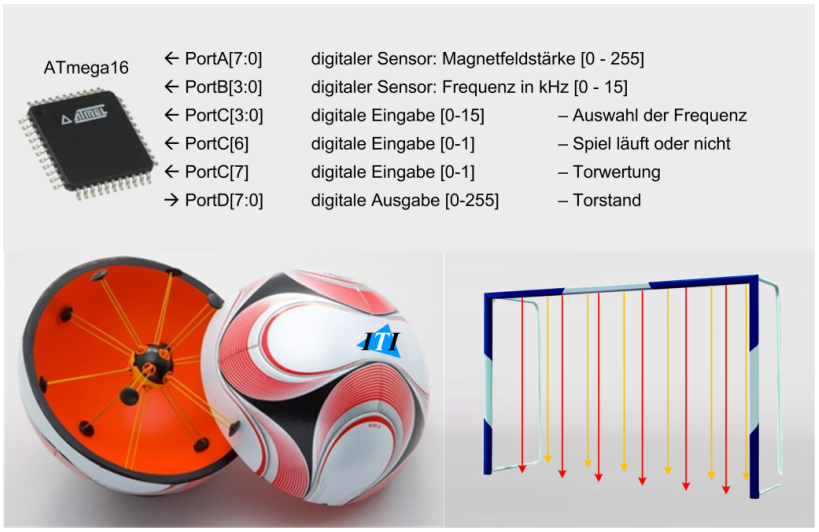
Verwenden Sie für Ihre Implementierungen einen beliebigen Editor und simulieren Sie ihren Assembler-Code anschließend im **ITImega16-Simulator**. Ihnen steht im Moodle für die Übung entsprechendes **Template** zur Verfügung.

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 - Smart-Ball

Mit Hilfe eines ATmega16, einem Magnetfeldsensor, einem Frequenzempfänger und einigen Eingabemöglichkeiten soll eine vereinfachte Version der GoalRef Torlinientechnologie des Fraunhofer IIS entwickelt werden. Dabei wird in einem Fußballtor ein magnetisches Feld erzeugt, an dessen Stärke gemessen werden kann, ob ein Ball die Torlinie überquert hat oder nicht. Der Ball selber funkt zusätzlich mit einer eigenen Frequenz zwischen 1 und 15kHz, anhand derer er im Tor identifiziert werden kann, um sicher zu gehen, dass ein Tor mit einem zulässigen Ball erzielt worden ist.



Die Funktionsweise des Systems kann wie folgt beschrieben werden: Wenn das Spiel läuft (PortC[6]=1), wird mittels eines Timers rund viermal pro Sekunde überprüft, ob ein Ball im Tor ist, oder nicht. Dafür wird die Stärke des Magnetfelds über den an PortA angeschlossenen Sensor eingelesen. Ist kein Ball im Tor, ist der Wert größer als 150 $\frac{A}{m}$ – ist ein Ball

im Tor, wird nur noch eine Stärke von höchstens 127 $\frac{A}{m}$ gemessen. Nachdem ein Ball erkannt worden ist, wird dessen Frequenz über den an PortB[3:0] angeschlossenen Empfänger eingelesen. Diese wird mit der anfangs über PortC[3:0] vor-eingestellten Frequenz im Bereich von 1 bis 15kHz verglichen. Erst nachdem beide Überprüfungen positiv verlaufen sind, kann der interne Torzähler goals um eins erhöht werden. Danach wird blockierend auf eine Betätigung des an PortC[7] angeschlossenen Tasters gewartet, bevor das nächste Tor gezählt werden kann. Parallel dazu läuft ein Hauptprogramm in einer Endlosschleife, welches lediglich den aktuellen Torstand goals über den PortD[7:0] ausgibt.

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 goals genutzt werden kann.

(b) Geben Sie eine Folge von Assembleroperationen an, mit der die Interrupt-Vektor-Tabelle initialisiert wird, sodass nach einem RESET zum Label init, und bei einem Timer/Counter0-Overflow-Interrupt zum Label CHECK4BALL_ISR gesprungen wird.

(c) Initialisieren Sie unter dem Label init den Stack-Pointer und die Ports explizit (Ports A-C=Eingang; D= Ausgang). Der Zustand der Eingänge bis auf PC7 sei tri-state und der Ausgang wird auf 0 gesetzt. Initialisieren Sie weiterhin den Timer/Counter0 mit einem Prescaler von clk/1024 und aktivieren Sie den Timer/Counter-Overflow-Interrupt. Aktivieren sie global die Interrupts und springen Sie zur main.

(d) Implementieren Sie das Hauptprogramm main, welches anfangs das Zählregister für die Tore goals auf 0 setzt und dann in einer Endlosschleife den Wert von goals auf den PortD[7:0] ausgibt. Dieser Wert wird unabhängig vom Hauptprogramm in der Interrupt-Service-Routine check4ball_isr manipuliert.

(e) Implementieren Sie die Interrupt-Service-Routine check4ball_isr wie folgt: Läuft das Spiel aktuell nicht (PortC[6] = 0), wird der Torzähler goals auf 0 gesetzt und zurück gesprungen. Läuft das Spiel (PortC[6] = 1), wird überprüft, ob aktuell ein Ball im Tor ist (PinA < 128) und falls ja, das Unterprogramm check4freq aufgerufen. Achten Sie wie immer darauf, dass die Zustände der Register und Flags erhalten bleiben.

(f) Implementieren Sie das Unterprogramm check4freq, welches zuerst den Wert aus PortC[3:0] ausliest, um diesen mit dem an PortB[3:0] gemessenen Wert zu vergleichen. Stimmen die Frequenzen nicht überein, wird zum Ende des Unterprogramms gesprungen. Ansonsten wird der Torzähler goals um 1 erhöht, und blockierend auf eine Betätigung (von 1 nach 0) des an PortC[7] angeschlossenen Tasters gewartet, bis zurück gesprungen wird. Dies verhindert, dass ein Tor vom System mehrfach gezählt wird.