UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TECHNISCHE INFORMATIK

ÜBUNG ZU „TECHNISCHE GRUNDLAGEN DER INFORMATIK“

SS 2012

Übung 12

Assemblerprogrammierung

Aufgabe 12.1: Ein-/Ausgabe *(10 Punkte)*

Mit Hilfe eines ATmega16 soll ein elektrisches Schloss für einen Safe gebaut werden. Die Pin-Nummer des Safes soll dabei über ein Tastenfeld (Abb. 4.1) eingegeben werden. Die Pin-Eingabe wird mit der #-Taste abgeschlossen und ist somit nicht auf eine feste Länge festgesetzt. Nach drei falschen Versuchen soll die Eingabe von Pin-Nummern für eine Stunde gesperrt sein.

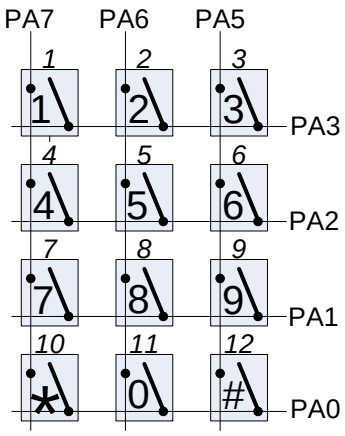

Abbildung 12.1: Tastenfeld

Um einen Tastendruck zu erkennen, ist die Tastenmatrix an PORTA angeschlossen, wobei PA7-PA5 als Ausgang (**Low**), und PA3-PA0 als Eingang zu konfigurieren sind (**Pull-Up**). Ist keine Taste gedrückt, so liegt an PA3-PA0 ein High an. Erst wenn eine Taste gedrückt wird (und damit der entsprechende Schalter geschlossen ist), so wird die entsprechende Zeile auf Low gezogen. Gehen Sie davon aus, dass nicht mehrere Tasten gleichzeitig gedrückt werden. Um nun die gedrückte Taste zu detektieren, soll wie folgt vorgegangen werden: Sobald eine Taste gedrückt wurde (eine Leitung von PA3-PA0 ist nicht mehr High), sollen zwei der drei Spalten auf High gelegt werden. Anschließend soll das gegebene Unterprogramm `get_row` aufgerufen werden. Liegt die gedrückte Taste in der Spalte, die auf Low gelegt ist, so wird die Zeilennummer (1-4) in R16 zurückgegeben, ansonsten eine 0. Dieses muss im Anschluss auch noch für die anderen beiden Kombinationen durchgeführt werden. Aus der Spalte und der Zeile kann nun die Tastennummer (1-12) errechnet werden.

Da der einzugebende Pin beliebig lang sein kann, soll die eingegebene Sequenz zuerst im RAM unter dem Label Code gespeichert und erst nach der Eingabe der #-Taste (12) mit der Überprüfung begonnen werden. Der Pin selber ist im ROM unter dem Label PIN gespeichert und schließt auch mit einer # (12) ab. Wurde der Code richtig eingegeben, so soll das gegebene Unterprogramm `granted` aufgerufen werden. Im Falle einer Falscheingabe soll das gegebene Unterprogramm `denied` aufgerufen, und wieder zum Anfang gesprungen werden. Wurde der Code jedoch 3x falsch eingegeben, so soll anschließend das gegebene Unterprogramm `wait_1_hour` aufgerufen werden, welches eine erneute Eingabe erst nach einer Stunde wieder erlaubt.

Aufgaben

Schreiben Sie aussagekräftig kommentierten ATmega16-Assembler-Code!

a) Schreiben Sie, beginnend mit dem Label `init`, Code zur Initialisierung des Stackpointers sowie des verwendeten I/O-Ports.

b) Schreiben Sie das Unterprogramm `read`, welches wartet, bis eine Taste gedrückt wurde. Anschließend soll, wie oben beschrieben, die gedrückte Taste ermittelt, und in `R16` gesichert werden. Erst, wenn die Taste wieder losgelassen wurde, soll das Unterprogramm wieder verlassen werden.

c) Schreiben Sie das Hauptprogramm `main`, welches in einer Endlosschleife läuft. Die eingelesenen Zeichen sollen im RAM unter dem Label Code gespeichert und nach dem Erkennen der #-Taste mit dem im ROM abgelegten PIN verglichen werden. Je nach Korrektheit soll hier entweder das Unterprogramm `granted` oder `denied` aufgerufen, und im Falle von drei Falscheingaben zusätzlich das Unterprogramm `wait_1_hour` aufgerufen werden, bevor wieder zum Anfang gesprungen wird.

Aufgabe 12.2: Ein-/Ausgabe, Timer/Counter & Interrupts *(10 Punkte)*

Ein ATmega16 soll in einem einfachen Stimmgerät für Gitarren zum Einsatz kommen (Abbildung 12.2). Hinter dem Mikrofon ist ein Verstärker sowie ein so genannter Schmitt-Trigger geschaltet, der das eingehende Sinussignal in ein Rechtecksignal umformt (Abbildung 12.3). Der Ausgang des Schmitt-Triggers ist an INT0 (PD2) angeschlossen. Zum Anzeigen der zu stimmenden Saite sind sechs Leuchtdioden (LEDs) eingebaut (PA5-PA0), wobei PA5 der tiefen E-Saite zugeordnet ist, und PA0 der hohen e-Seite. Die oberen drei LEDs zeigen zusätzlich an, ob die Saite zu tief (PD7), zu hoch (PD5) oder richtig (PD6) gestimmt ist.

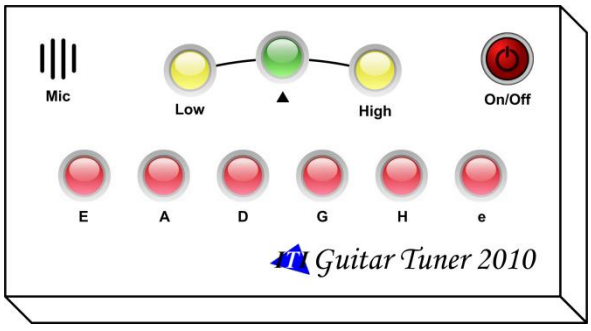

Abbildung 12.2

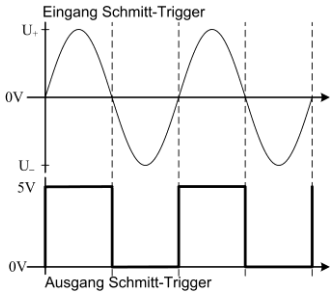

Abbildung 12.3

Saite	Frequenz	Timer
E	82,4 Hz	190
A	110 Hz	142
D	146,83 Hz	106
G	196 Hz	80
H	246,94 Hz	63
e	329,62 Hz	47

Tabelle 12.4

Um die Frequenz der Saite zu bestimmen soll der Timer/Counter 0 mit einem Prescaler von 64 verwendet werden. Zur Erhöhung der Messgenauigkeit soll dabei aber nicht nur ein Wert gemessen werden, sondern der Mittelwert über fünf Messungen errechnet werden. Diese sollen im Datenbereich in einer Variablen `Messungen` (5 Byte) abgelegt werden.

Wird ein Interrupt an INT0 bei einer steigenden Flanke ausgelöst, so soll der Zählerwert des Timer/Counter 0 auf 0 gesetzt werden. Bei der nächsten steigenden Flanke soll der Wert des Timer/Counter 0 dann an der entsprechenden Stelle in der Variablen `Messungen` gespeichert werden und der Timer/Counter 0 wieder auf 0 gesetzt werden. Nach den fünf Messungen wird dann der Mittelwert der Messwerte berechnet.

Anschließend soll die LED derjenigen Saite eingeschaltet werden, deren Frequenz der gemessenen Frequenz am nächsten ist. Die Tabelle der Frequenzen ist bereits im ROM unter dem Namen `Frequenzen` abgelegt, wobei der erste Eintrag den Vergleichswert für die tiefe E-Saite enthält. Zusätzlich soll mit den drei oberen LEDs angezeigt werden, ob die gemessene Frequenz für die Saite zu niedrig, zu hoch oder richtig ist.

Aufgaben
Verwenden Sie aussagekräftig kommentierten ATmega16-Assembler-Code!
Ihnen steht ein Unterprogramm `mittelwert` zur Verfügung, welches im `Z`-Register die Adresse der ersten gemessenen Frequenz und die Anzahl der Messungen in `R16` erwartet, und den Mittelwert in `R16` zurückliefert.

d) Geben Sie eine Folge von Assemblerdirektiven an, mit der die Interrupt-Vektor-Tabelle initialisiert wird, so dass nach einem Reset zum Label `init` und bei einem externen Interrupt über `INT0` zum Label `isr_int0` gesprungen wird.

e) Geben Sie eine Folge von Assemblerdirektiven an, mit denen am Anfang des SRAM Platz für die Variable `Messungen` reserviert wird.

f) Schreiben Sie, beginnend mit dem Label `init`, Code zur Initialisierung des Stackpointers, der verwendeten I/O-Ports sowie der Initialisierung der Timer und des externen Interrupts. Anschließend soll zum Label `main` gesprungen werden.

g) Schreiben Sie die Interrupt-Service Routine `isr_int0` für das Speichern des Timerwertes und das Rücksetzen des Wertes für eine neue Messung.

h) Schreiben Sie ein Unterprogramm `anzeigen`, welches den errechneten Mittelwert in `R16` übergeben bekommt und die LED der entsprechenden Saite, sowie die LED der aktuellen Stimmung (Low, Tuned, High) einschaltet. Vergessen Sie dabei nicht, eventuell vorher eingeschaltete LEDs wieder auszuschalten.

i) Schreiben Sie das Hauptprogramm `main`, welches in einer Endlosschleife läuft. Aktivieren Sie zuerst die globalen Interrupts. Nachdem die fünf Messungen ausgeführt wurden sollen die Interrupts wieder deaktiviert werden. Anschließend soll der Mittelwert berechnet, und die Stimmung angezeigt werden.

Geben Sie bitte **zusätzlich** zum Ausdruck alle im AVR Studio erarbeiteten Lösungsteile per eMail an die Adresse tgi@iti.uni-luebeck.de ab.

Abgabe bis spätestens Freitag, 06.07.2012, 12.00 Uhr