UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TECHNISCHE INFORMATIK

Übungsblatt 4 (praktisch)

Digitales Zahlenschloss auf einem ATmega16

Vorlesung: Technische Grundlagen der Informatik 1, Sommersemester 2020

Dozent: Dr.-Ing. Kristian Ehlers

Ziele und Umfeld des Versuchs

Das Ziel dieses Versuchs ist die Implementierung eines digitalen Zahlenschlosses mit Hilfe eines ATmega16 Mikrocontrollers. Dieser ist Ihnen aus der praktischen Übung 3 bereits bekannt. Ein wichtiger Schwerpunkt dieser Übung betrifft die Programmstrukturierung und Wiederverwendung von Code mit Hilfe von Unterprogrammen.

Vorbereitungsaufgaben

Die folgenden Vorbereitungsaufgaben dienen dazu, die Aufgabe genau zu spezifizieren und Ihnen eine Schritt-für-Schritt Anleitung zur Lösung des Problems zu geben. Bearbeiten Sie die Aufgaben vollständig zu Hause, mit dem Ziel, beim Versuch in unserem Labor nur noch den möglichst lauffähigen Code eingeben zu müssen. Nutzen Sie den *ITImega16-Simulator*, um Ihren Code möglichst vollständig zuhause zu testen.

Aufgabenstellung und Gesamtverhalten des Systems

In diesem Versuch sollen Sie auf dem ATmega16 Mikrocontroller ein digitales 4-Ziffern-Zahlenschlosses vergleichbar mit der Pin-Eingabe auf einem Mobiltelefon entwerfen und implementieren. Ein Taster an PA7 dient zur Einstellung einer Ziffer von 0 bis 9, die dann mit einem weiteren Taster an PA0 bestätigt wird. Die auszuwählende Ziffer wird dabei auf dem PortD als Binärzahl ausgegeben. Weiterhin werden sechs farbige LEDs an das System angeschlossen, welche den aktuellen Zustand des Systems anzeigen. Dabei dienen vier gelbe LEDs zur Visualisierung, welche Ziffer des Codes gerade eingelesen wird. Eine grüne LED visualisiert die gegebenenfalls erfolgreiche Eingabe des Codes wohingegen eine rote LED die fehlerhafte Eingabe des Codes visualisieren soll. Die einzelnen Ziffern der sequentiell einzugebenden vierstelligen Kombination werden unter dem Label CODE im ROM abgelegt.
Die folgende Abbildung 1 verdeutlicht das Verhalten des Systems bei einer vorgegebenen Ziffernkombination von „0-8-1-5“. Dabei wird der Pin-Code zuerst richtig und dann einmal falsch eingegeben.

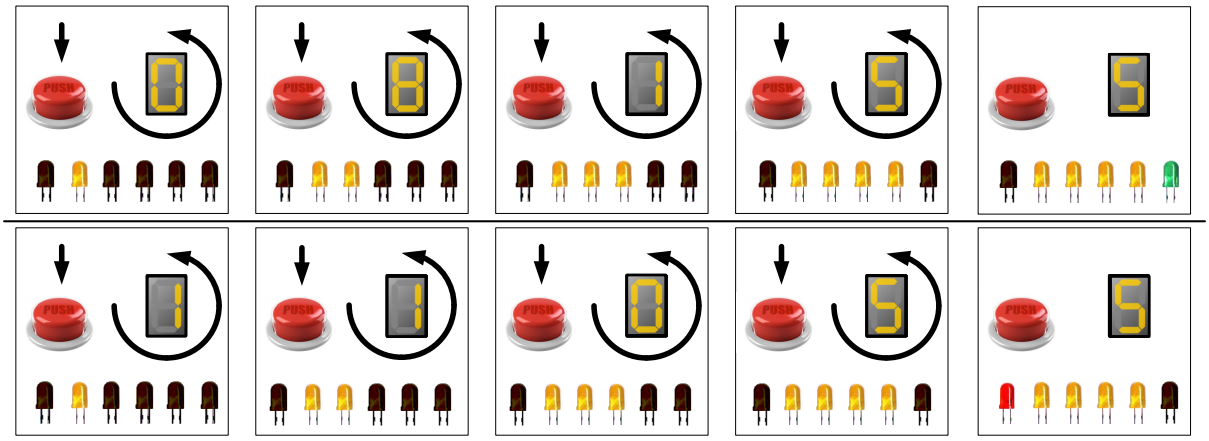

Abbildung 1: Zweimalige Eingabe eines Codes. Beim ersten Mal (obere Zeile) wird der Code akzeptiert, beim zweiten Mal (untere Zeile) verworfen.

Gesamtverhalten

Das Gesamtverhalten des Systems wird folgendermaßen beschrieben. Der Nutzer hat die Möglichkeit, mit Hilfe eines Schalters eine Ziffer von 0-9 einzustellen. Diese wird auf der 7-Segment-Anzeige dargestellt. Jede durch eine Betätigung des Schalters erzeugte fallende Flanke (Wechsel von High - 1 auf Low - 0) sorgt für das Inkrementieren des Ziffernwertes. Sollte der Wert den Zahlenwert 10 erreichen, wird dieser auf 0 zurückgesetzt (0, 1, 2, …, 8, 9, 0, 1, …). Hat der Nutzer eine gewünschte Ziffer eingestellt, soll diese durch die Betätigung des zweiten Schalters bestätigt und vom System als Eingabe übernommen werden. Auch bei diesem Schalter soll die entsprechende Reaktion des Systems nur bei der fallenden Flanke erfolgen. Auf diese Weise kann der Nutzer sequentiell vier Ziffern eingeben. Die vier gelben LEDs visualisieren die Stelle der aktuellen Ziffer des vierstelligen Codes. Die Eingabe der Ziffern erfolgt stets in der sequentiellen Reihenfolge von links nach rechts. Sollte die Kombination also wie im Beispiel „0-8-1-5“ lauten, so muss erst die Eingabe der 0, dann die Eingabe der 8 bis hin zur Eingabe der 5 erfolgen. Während der Einstellung der ersten Ziffer leuchtet die erste LED. Während der Auswahl der zweiten Ziffer leuchten die erste und die zweite LED usw. Bei einer korrekten Eingabe aller vier Ziffern leuchtet zusätzlich zu den vier gelben LEDs die grüne LED anderenfalls die rote LED. In diesem Zustand soll das System solange verharren, bis der Schalter zur Übernahmen einer Ziffer ein weiteres Mal betätigt wird. Dann springt das System zur Eingabe der ersten Ziffer und das Verhalten beginnt von vorn.

Implementierung

Nutzen Sie für diesen Versuch das im Moodle gegebene Template. Weiterhin geben wir Ihnen nachfolgend schrittweise Hinweise für die Implementierung. Es steht Ihnen frei, diese zu befolgen oder nicht. Es ist jedoch Pflicht, das Unterprogramm `POLL_BUTTONS` mit der beschriebenen Funktionalität zu implementieren und die vier Ziffern der Kombination im ROM abzulegen.

Nehmen Sie zu Beginn des Programms alle von Ihnen genutzten Namensdefinitionen vor. Wir empfehlen Ihnen, die aktuell eingestellte Ziffer im Register `R20` unter dem Namen `Digit` zu sichern, den Status, ob bereits eine fehlerhafte Eingabe erfolgt ist, im Register `R21` unter dem Namen `Fail` festzuhalten und für die Detektion der fallenden Flanken an den Schaltern das Register `R22` unter dem Namen `ButtonStates` aufrufbar zu machen. Nehmen Sie ferner alle notwendigen Initialisierungen für ihr Programm vor. Initialisieren Sie den Stack Pointer und stellen Sie gemäß Tabelle 1 alle benötigten Ein- und Ausgabegabe-Ports des ATmega16 ein.

ATmega I/O Port	I/O Richtung	Anschluss
PB0	Ausgang	LED Gelb1
PB1	Ausgang	LED Gelb2
PB2	Ausgang	LED Gelb3
PB3	Ausgang	LED Gelb4
PB4	Ausgang	LED Grün
PB5	Ausgang	LED Rot
PA7	Eingang	Taster (Inkrementieren der Ziffern)
PA0	Eingang	Taster (Eingabe)
PD3 - PD0	Ausgang	Anzeige der aktuellen Ziffer

Tabelle 1: Portbelegung des ATmega16 mit Ein- und Ausgaberichtung

Das Basisverhalten des Systems läuft in einer Endlosschleife und soll unter dem Label `MAIN` implementiert werden. Im Hauptprogramm werden das Register `Fail` mit 0 initialisiert, die erste LED angeschaltet und anschließend sequentiell die vier Ziffern jeweils durch einen Aufruf des Unterprogramms `POLL_BUTTONS` eingelesen. Nach jedem Aufruf erfolgt direkt der Vergleich mit der korrespondierenden Ziffer der Kombination und die entsprechenden gelben LEDs werden angeschaltet. Sollten die Ziffern nicht übereinstimmen, so wird das Register `Fail` inkrementiert. Auf diese Weise kann nach der Eingabe der letzten Ziffern das Register `Fail` überprüft werden, um herauszubekommen, ob der komplette Code richtig eingegeben wurde oder nicht. Anhand des Resultats wird dann zusätzlich die rote oder grüne LED angeschaltet. Das System soll solange in diesem Zustand verharren, bis erneut der Schalter zur Übernahme einer Ziffer betätigt wird und anschließend von vorn beginnen.

Das Unterprogramm `POLL_BUTTONS` soll in einer Schleife die Eingänge, an der die beiden Schalter angeschlossen sind, auf eine fallende Flanke überprüfen. Wird diese an dem Eingang des Schalters für das Inkrementieren der Ziffer detektiert, so soll das Register Digit inkrementiert werden. Sollte der Wert in dem Register den Zahlenwert 9 überschreiten, so wird er auf null zurückgesetzt. Weiterhin wird die 7-Segement-Anzeige aktualisiert. Sollte der Schalter für die Übernahme einer Ziffer betätigt worden sein, ist das Unterprogramm zu verlassen. Nach dem Rückkehren aus dem Unterprogramm befindet sich also im Register `Digit` die aktuell eingestellte und bestätigte Ziffer.
Hinweis: Für die Detektion der fallenden Flanken empfehlen wir nach jedem Einlesen der Eingänge im Register `ButtonStates` den Zustand der Buttons zu speichern. Sollte dann der aktuelle Wert des Eingangs 0 und der im vorherigen Durchlauf 1 gewesen sein, ist eine fallende Flanke detektiert worden.

Testen Sie Ihren Assemblercode zu Hause ausführlich. Nutzen Sie dafür den *ITImega16-Simulator*.

Die nachfolgenden Ausführungen dienen lediglich der Information und müssen im Rahmen dieser Praktischen Übung bei der Bearbeitung nicht berücksichtigt werden. Es erfolgt eine Einführung in die Hard- und Softwareumgebung des Labors sowie die Durchführungsbeschreibung im Labor. Diese Information sollen das Verständnis für das spätere Tutorial zu dieser Übung fördern und den Hardwarebezug in der Laborumgebung aufrechthalten.

Das verwendete Experimentiersystem

Für die Durchführung des Versuchs sind die in Abbildung 2 dargestellten Bauteile (ein Schalterbaustein, ein LED-Baustein, eine 7-Segment-Anzeige und ein ATmega16) auf dem Steckbrett zu platzieren.

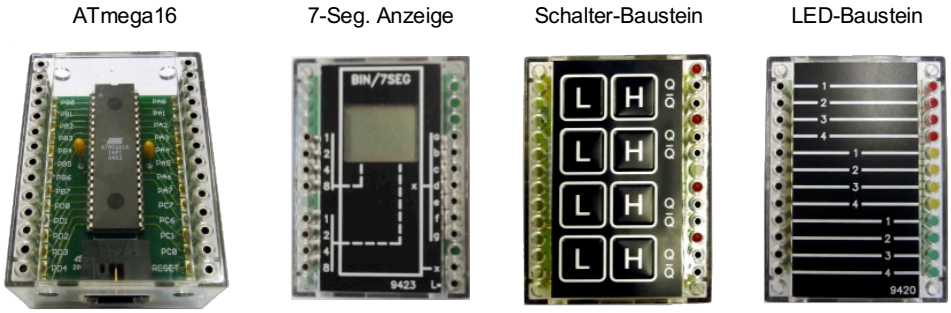

Abbildung 2: Für das Experiment genutzte Bauteile

Versuchsdurchführung

Bevor Sie ihren Versuch auf dem Steckbrett aufbauen, sollten Sie das von Ihnen in der Versuchsvorbereitung geschriebene Programm eingeben und mit Hilfe des *ITImega16-Simulators* überprüfen.

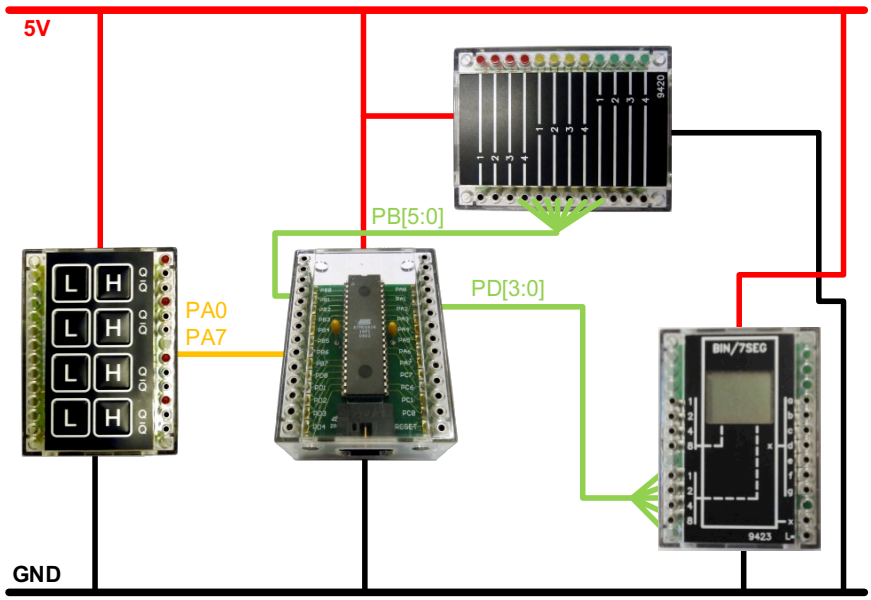

Abbildung 3: Schaltplan des Versuchsaufbaus

Bauen Sie nun den gesamten Versuchsaufbau auf dem Steckbrett auf, wobei Sie währenddessen auf keinen Fall die Spannungsversorgung einschalten! Die Abbildung 3 verdeutlicht den Versuchsaufbau grafisch.