Hochschule Bremerhaven

Fakultät II – Management und Informationssysteme

Informatik

Modul Rechnerarchitektur

Protokoll zu Aufgabenblatt 13

von

André Jehsert Matrikel-Nr. 36270
Jan-Niklas Pauls Matrikel-Nr. 36268

02.07.2019

Inhaltsverzeichnis

1 Zusammenfassung

Der hier protokollierte fünfte und letzte Übungszettel besteht aus einer Aufgabe, welche Taster thematisiert. Bei dieser Aufgabe wird ein großer Teil auch auf die Planung des Programmes gelegt. Weitere Bonus-Aufgaben gibt es nicht wirklich. Die gegebene Bonus-Aufgabe weist auf die Aufgaben des vorherigen Übungszettels hin.

2 Einleitung

In dem folgenden Protokoll wird die Aufgabe eins des Übungszettels fünf bearbeitet. Bei besagter Aufgabe ging es um das Einbinden von Tastern, in das von uns erstellt Programm, welches Teil des letzten Übungsblattes war. Bevor dieser Taster allerdings implementiert werden soll, sollen Umsetzungsideen in Form eines Timing, Aktivitäts-, oder ähnliches Diagramms verdeutlicht werden.

3 Material und Methode

3.1 Evaluationsboard Amtel Ver.2.01

Das Evaluationsboard ist eine Platine auf dem viele kleine und uns noch unbekannte Bauteile verbaut sind. Es versorgt alle Bauteile mit Strom (bei uns VCC (Voltage at the common collector)=5V) und ermöglicht es diesen mit Leiterbahnen zu kommunizieren. Auf Grund der Aufgabenstellung sind uns schon einige Eigenschaften bekannt, welche wir nun vorausnehmen. So gab es eine von uns Identifizierte Bauteile:

- die Stromverbindung mithilfe eines Netzteiles
- der Summer
- LED´s
- Reset-Button und 3 weitere Taster
- mehrere Widerstände
- visuelle Schnittstelle

Zudem sind ist ein Mikrokontroller namens ÄTMEGA 32 16PU 01810D”verbaut. Das GND-Potential kann auf dem Board über den Pin 35, 37 und 39 abgegriffen werden. Das VCC-Potential über die Pin´s 36, 38 und 40.

3.2 Steckbrett

Das Steckbrett wird genutzt um Schaltkreise auf kleinem Raum nachbauen zu können. In einer regelmäßigen Gitterstruktur ist das Brett mit Einlassungen versehen, sodass Kontakte hier reingesteckt werden können um einen Stromfluss zu ermöglichen.

Links und Rechts am Rand verlaufen je 2 Bahnen, welche für GND und VCC vorgesehen sind. Diese Bahnen laufen Vertikal und lassen sich mithilfe von Kabeln an das Evaluationsboard anschliesen. Somit lässt sich das Brett mit Strom versorgen.

3.3 Kabel

Hier handelt es sich um vermutlich einfache 1 Milimeter dicke Kupferkabel, welche isoliert und mit einem Steckkopf versheen sind. Domit sind diese einfach in das Steckbrett einzusetzten.

3.4 Leuchtdioden

Es handelt sich um eine gelbe und grüne Leuchtdiode. Ihre Normalspannung liegt laut Recherche bei ca 2,1V und sie besitzen einen Gallium-Phospid Halbleiter und weisen zudem eine typische Lichtstärke von 18 mcd auf.

3.5 Widerstände

Insgesamt hatten wir 8 Widerstände. Jeweils 4x1k Ohmen und 4x10k Ohmen.

3.6 Transistor

Wir haben Transistoren vom Typen BC 547C

4 Ergebnisse

4.1 Aufgabe 01

4.1.1 Zusammenfassung

Die vorliegende Aufgabe eins handelt darum einen Taster des Entwicklungsbords so in unser Programm einzubauen, sodass dieser dann bei der Aktivierung unseren Morsecode des vorherigen Übungszettels ausgibt. Wie wir allerdings das Zusammenspiel zwischen unserer Morsenachricht und dem Taster implementieren wird in der Aufgaben freigegeben. Die Aufgabe fordert ebenfalls, dass vor unserer Implementierung wir unsere Idee in Form von Diagrammen verdeutlichen sollen.

4.1.2 Erwartung

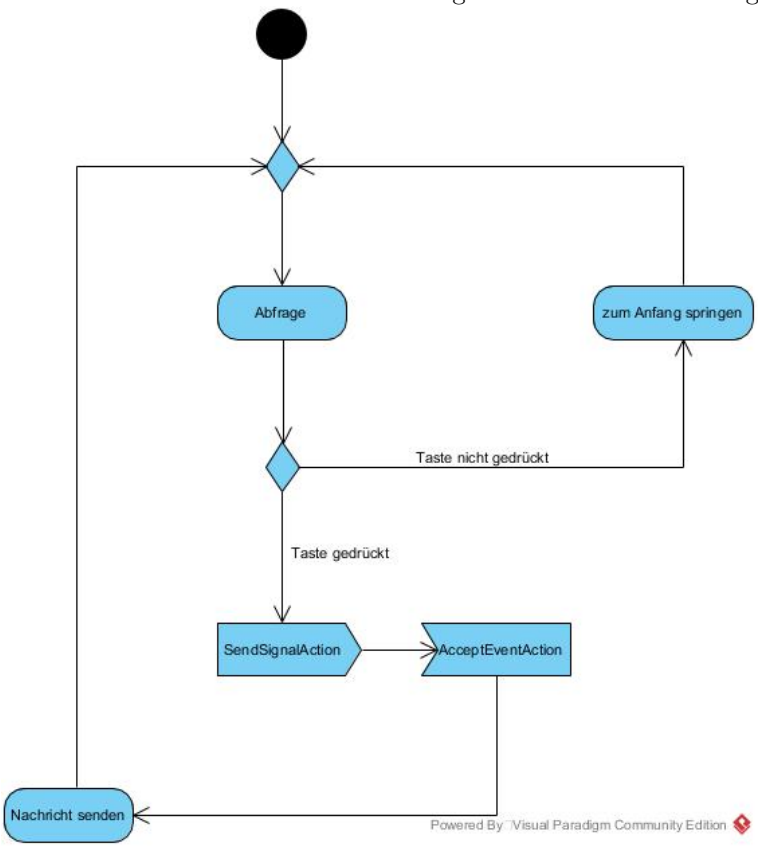
Die Erwartungen an diese Aufgabe beziehen sich vor allem auf unsere geplante Umsetztung. Unsere Überlegung war es, den Taster zu überprüfen. Dieser muss irgendeine Veränderung durch das drücken hervorrufen. Die Überprüfung sollte in einer Endlosschleife stattfinden und erst bei erfüllen der Bedingung (Taster gedrückt) zu der Ausgaberoutine springen. Ist diese einmal durchlaufen, soll das Programm wieder in die Abfrageschleife springen.

4.1.3 Materialien

- Laborrechner
- Morsecode-Programme vom letzen Übungsblatt
- Open-Collector-Schaltung
- Taster auf dem Mikrocontrollers

4.1.4 Detallierte Durchführung

Zu aller Erst haben wir unsere Gedanken der Erwartung mit hilfe eines UML-Diagrammes aufgezeichnet.



Nach unseren Recherchen haben wir unseren Registern Marker zugewiesen. Dieses taten wir in der 'init.S'. Folgende Register wurden gesetzt:

- Interrupt Knopf 1 = .EQU int0, 0x0001
- Interrupt Knopf 2 = .EQU int1, 0x0002

- General Interrupt Register = .EQU GICR, 0x3B
- MCU Controll Register = .EQU MCUCR, 0x35

Nachdem wir nun auf die Register zugreifen können, werden diese in der 'run.S' genutzt. Dafür schreiben wir die Zahl '0b00001010' in unsere MCUCR. Durch das Beschreiben des Registers mit dieser Zahl, können wir bestimmen ob wir eine Auslösung der Interruptroutine der Taster bei einer steigenden oder fallenden Flanke hervorrufen wollen. Dieses bedeuten nichts anderes als, eine Auslösung beim Drücken oder dem Loslassen des Tasters. Anschließend mussten wir das Register 'GICR' mit der '0b11000000' beschreiben. Dieses Schaltet das Bit an der Stelle 6 und 7 ein, was den Interrupt durch den externen Pin INT0 und INT1 zulässt. Nun muss nur mit 'SEI' der Global Interrupt eingeschaltet werden. Nachdem dieses passiert ist, wird eine kleine Endlosschleife gebaut. Diese soll nur verlassen werden, wenn das Vergleichsregister durch die Interruptroutine verändert wurde und damit r16 und r17 gleich sind. Dadurch springt die Schleife in die Morsecoderoutine und führt diesen einmal aus. Anschließend soll wieder in die Abfrgae gesprungen werden. Wie nun ein Interruptvektor definiert wird, haben wir durch die Hilfe von Herrn Lipskoch erhalten. Danach sollten wir den Code nach seinen Vorgaben in der 'interrupt.S' ergänzen. Im Groben wird die Lister der Interruptroutinen modifiziert und eine neue Routine für einen Interrupt der Pins INT0 und INT1 hinzugefügt. Dadurch wird dann beim Drücken der Taster ein Register unserer Whal beschrieben und wir können es Vergleichen.

4.1.5 Ergebnisse

Unsere Ergebnisse sind die Code-Snippets:

```
; Setzte die Interupts der Knöpfe
.EQU int0, 0x0001
.EQU int1, 0x0002

.EQU PORTB, 0x18
.EQU DDRB, 0x17

; General Interupt Register
.EQU GICR, 0x3B

; MCU Controll Register
.EQU MCUCR, 0x35
```

```
; Schreibe die Anweisung als Binärcode in ein Register da MCUCR nur mit Registern beschrieben werden kann
  ldi r18, 0b00001010

; Schreibe den Wert aus dem Register 18 in das MCUCR
  OUT MCUCR, r18

; Schreibe Anweisung als Binärcode in dein Register
  ldi r18, 0b11000000

; Schreibe den Wert aus dem Regoster 18 in das GICR
  OUT GICR, r18

; Set Global Interrupt Flag
  SEI


; Setzte Marke taster, setzt r16 auf 0, vergleicht Register mit dem Zähler, springt wenn nicht gleich in die loop
taster:
  ldi r16,0
  cp r16, r17
  brne loop

; Spring wieder in die Abfrage wenn cp not equal
jmp taster


loop:
; cleart, setzt, cleart um dem Ozilloskop zu zeigen wo der Anfang ist
  cbi PORTA,PA1
  sbi PORTA,PA1
  cbi PORTA,PA1

; ruft Funktionen auf um den Morsecode aufzurufen
  call dit
  call kpause
  call dah
  call kpause
  call dah
  call kpause
  call dah
  call kpause
  call dah
  call gpause
  call dah
  call kpause
  call dit
  call kpause
  call dit
  call kpause
  jmp taster
```

```
; Vektorraum Stelle 0 ist der Reset
; Schreibt im Vektorraum die Funktion des Interruptes an der Stelle 1 um. Stelle 1 ist für den Knopf zus
tändig
; wenn int0 durch das Drücken aufgerufen wird, wird die Routine int0 ausgeführt. Diese setzt Register 17
 auf 1 und returnt.
.GLOBAL __vector_1
.type __vector_1, @function
__vector_1:
int0:
  ldi r17,0
  reti
.SIZE __vector_1, .-__vector_1

; Schreibt im Vektorraum die Funktion des Interruptes an der Stelle 1 um. Stelle 2 ist für den Knopf zus
tändig
; wenn int1 durch das Drücken aufgerufen wird, wird die Routine int0 ausgeführt. Diese setzt Register 17
 auf 0 und returnt.
.GLOBAL __vector_2
.type __vector_2, @function
__vector_2:
int1:
  ldi r17,1
  reti
.SIZE __vector_2, .-__vector_2
```

Es ist nun nun möglich mit dem Taster1 den Morstecode zu senden und mit Taster2 diesen wieder auszuschalten.

4.1.6 Deutung

Bis auf das Einstellen des Interruptvektors, haben sich unsere Erwartungen mehr oder weniger bestätigt. Mithilfe der Implemetierung der 2 Taster ist es nun möglich den Morsecode zu senden. Da wir bereits 2 Taster impelmetiert haben, könnte man nun relative einfach die Interrupt routine anpassen und die Wahrheitstabelle mithilfe der Taster simulieren.

5 Diskussion

Zusammenfassend können wir für das fünfte Übungsblatt sagen, dass wir relativ schnell eine Idee für die Implementierung der Aufgabe eins hatten. Das Problem an dieser Implementierungsidee war allerdings, dass diese das Aktivieren des Tasters nicht jedes Mal gewiss registriert. Dann kam uns die Idee die Implementierung und das Registrieren des aktivierten Tasters mithilfe von Interrupts umzusetzen. Nach ein paar Recherchen und Nachschlagen im Handbuch des Mikrocontrollers haben wir die dafür erforderlichen Spezialregister her rausgefunden und somit die Implementierung umsetzen können.