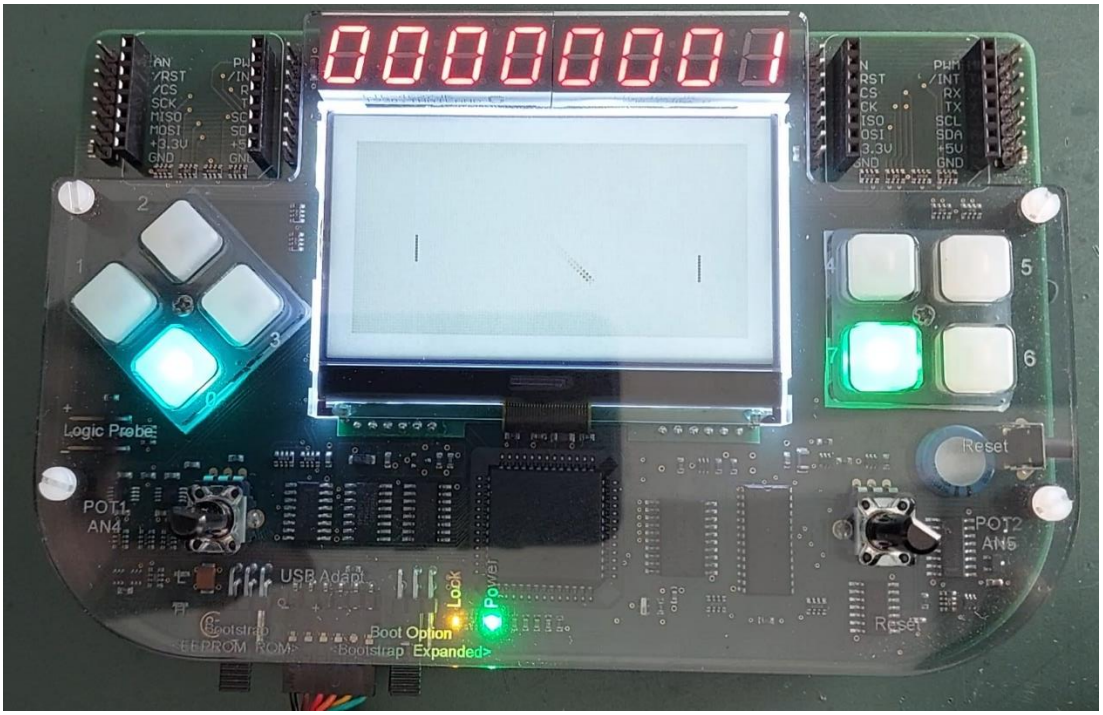
Bildungseinrichtung:	FH Wedel
Ort:	22880 Wedel
Semester:	Wintersemester 2022

Pong

Workshop Microcontroller



Verfasser:

Name:	Krause
Vorname	Luca Manuel
Fachrichtung:	Technische Informatik
Matrikelnummer:	104236
Fachsemester:	5
Verwaltungssemester:	5

Inhalt

Aufgabe

Abstrakte Erläuterung

Ziel der Aufgabe ist eine Umsetzung eines komplexeren Programms auf dem trainer11-Board. Die Schwerpunkte liegen vor allem bei der klar strukturierten Programmierung und der Verständlichkeit der Dokumentation.

Umsetzung der Aufgabe

Die Aufgabe ist realisiert in dem Programm Pong, einem 1972 von Atari veröffentlichten Videospiel, dass als weltweit erstes beliebtestes Videospiel gilt.

Elemente auf dem Board

Um das Spiel Pong auf dem Board zu realisieren wird das LCD-Display für das Spielfeld genutzt, die 7-Segment-Anzeige für den Punktestand und für einen Reset des Spielstandes eine Taste (Taste 0).
Die acht Elemente der Sieben-Segment-Anzeige werden in zwei mal vier Elemente aufgeteilt, sodass die ersten vier die Punkte des linken Spielers und die rechten vier die Punkte des rechten Spielers anzeigt.

Entwicklungsumgebung

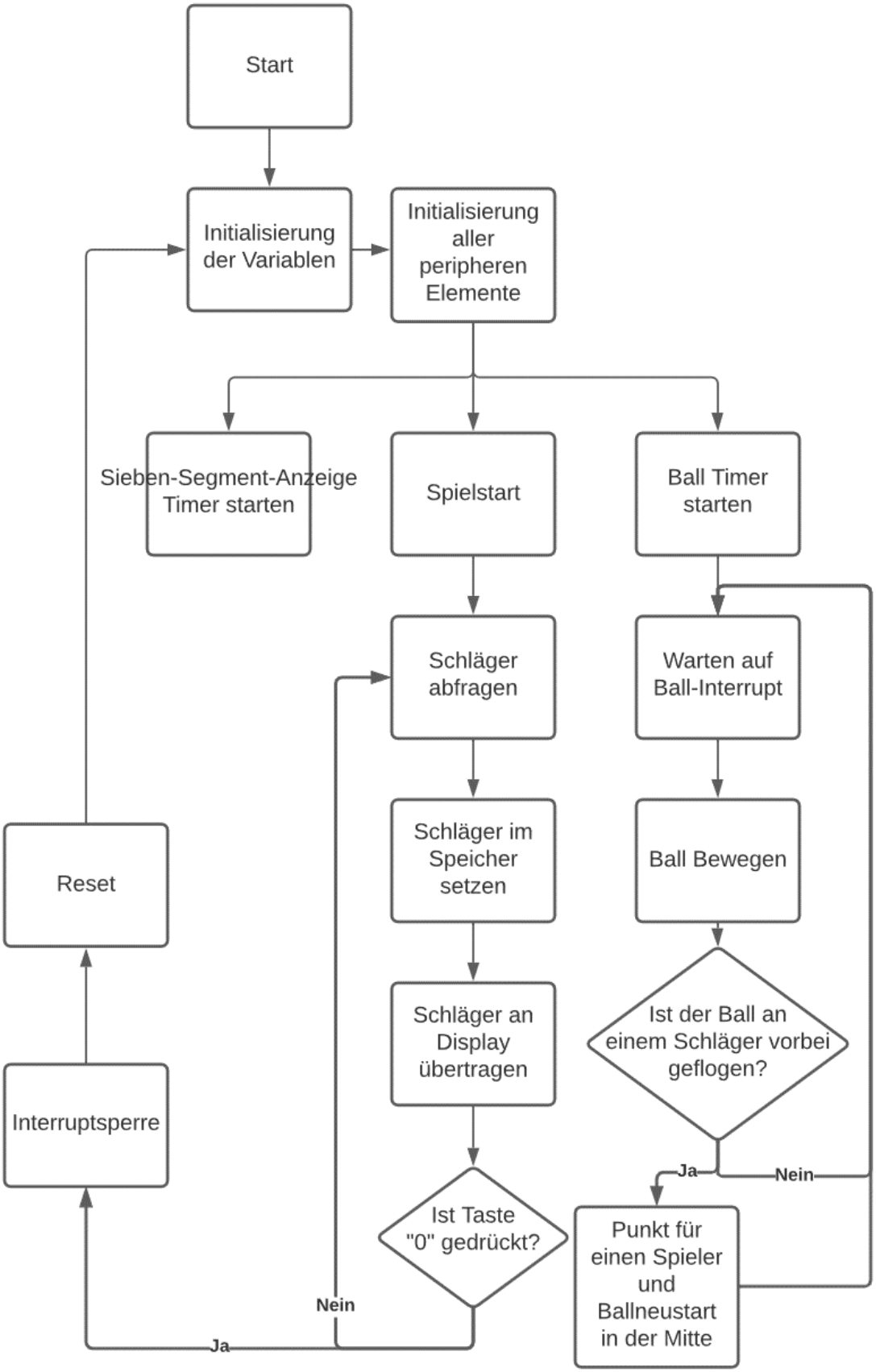
Betriebssystem	Windows 10
Software	Geany IDE, MiniIDE
Sprache	ASM11

Installation Geany und MiniIDE

- Bei der Nutzung der Geany IDE sollte bei jedem Dokument bei >Dokument>Dateityp festlegen>Kompilersprachen>Assembler ausgewählt werden.
- Anschließend in dem Reiter >Erstellen>Kommandos links „Assembler“ und rechts „C:\miniIDE\ASM11 %f -l“ eintragen.
- Nun im gleichen Fenster unter „Befehle zum Ausführen“ bei links „Ausführen“, rechts „C:\Realterm\realterm "first" "display=1" "rows=40" "baud=9600" "sendfile=%d\%e.s19"“ eintragen.
    - Eventuell kann hier auch noch der Standardport mit “port=COMPort“ eingetragen werden.
- Die miniIDE von Mgtek sollte hierzu im Hauptverzeichnis C:/miniIDE installiert werden. Diese IDE nutzt nur den Assembler ASM11
- Realterm sollte auch im Hauptverzeichnis als C:/realterm installiert werden.[12]

[2] a: Diese Beschreibungen entsprechen im Wesentlichen den Beschreibungen von Herrn Völkers Folien

Programmorganisationsplan



Spielablauf / Beschreibung des POP

Nach dem Starten des Microcontrollers und Überspielen der Dateien werden die Variablen und peripheren Elemente, wie das Display, die Potentiometer und die 7-Segment-Anzeige initialisiert. Die Variablen werden vollständig mit Nullen gefüllt und im Anschluss von den eigenen Initialisierungsprogrammen, wie dem Setzen des Balles, auf spezielle Werte gesetzt.

Bei der Initialisierung werden außerdem die beiden Timer gestartet, wobei der eine Timer für die Aktualisierung des Balls und der andere für die Aktualisierung der 7-Segment-Anzeige zuständig ist. Der Timer für die Anzeige der Punkte liest nur die Punkte aus und veröffentlicht diese kontinuierlich.

Bei dem Timer des Balls wird zudem noch geprüft, ob er die Grenzen des Spielfeldes erreicht und deshalb entweder von Ihnen abprallen muss (obere und untere Grenze) oder an den Schlägern vorbeifliegen würde und deshalb einen Spieler einen Punkt erhält. In dem Moment wird dann auch die Anzahl der Punkte aktualisiert.

Nach den Initialisierungen wird die Hauptschleife des Spiels gestartet. In dieser wird immer die Position der Schläger abgefragt, diese auf einen Wertebereich transformiert und im Anschluss im Speicher abgelegt, der jedes Byte der Schläger beinhaltet. Anschließend werden die Bytes der Schläger an das Display übertragen. Als letzter Teil der Schleife wird die Taste „0“ überprüft, die eine Reset-Taste darstellt. Mit dieser Taste werden der Spielstand und das Spielfeld zurückgesetzt, wobei im Anschluss wieder mit der Initialisierung gestartet wird.

Beschreibung wichtigster Dateien/Unterprogramme

Ball.inc

In der Datei zum Ball werden alle Dinge zum Ball geklärt. Er wird hier initialisiert und bewegt.
Bei der Bewegung des Balls gibt es verschiedene Fälle. Im einfachsten Fall wird der Ball von seiner Bewegung her nur einen Punkt weitergesetzt. Fliegt der Ball also nach rechts unten, nimmt dessen x- und y-Position um jeweils +1 zu. Trifft der Ball die obere Grenze, muss er nach unten abprallen, trifft er die untere, muss er nach oben abprallen. Das Abprallen nach oben und unten geschieht im 45°-Winkel.

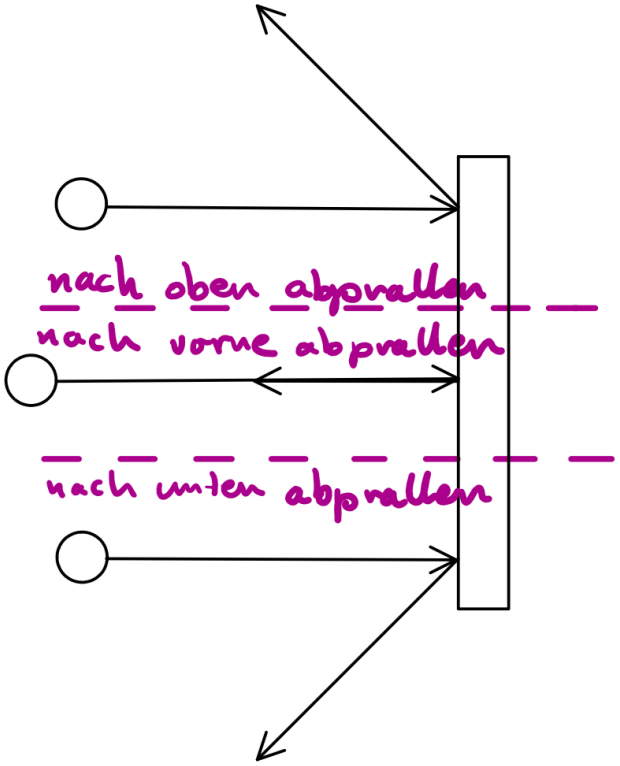
Pseudocode:

```
//Konstanten
POS_BAT_1 = 1; POS_BAT_2 = 126;
//Variablen
move_x = 1; move_y = 1;
x_pos = Width/2; y_pos = Height/2;
x_pos += move_x; y_pos += move_y;

If (y_pos == 0) {
        move_y = 1
} else if (y_pos == Height -1) {
        move_y = -1
}

//Befindet sich der Ball vor einem der Schläger?
If (x_pos == POS_BAT_1 + 1) {
        //Werte des ersten Schlägers
        If (y_pos >= SchlägerPosStart && y_pos < SchlägerPosEnd) {
                //Schläger getroffen
                move_x = 1
        } else {
                //Punkte verteilen
        }
} else if (x_pos == POS_BAT_2 - 1) {
        //Werte des zweiten Schlägers
        If (y_pos >= SchlägerPosStart && y_pos < SchlägerPosEnd) {
                //Schläger getroffen
                move_x = -1
        } else {
                //Punkte verteilen
        }
}
```

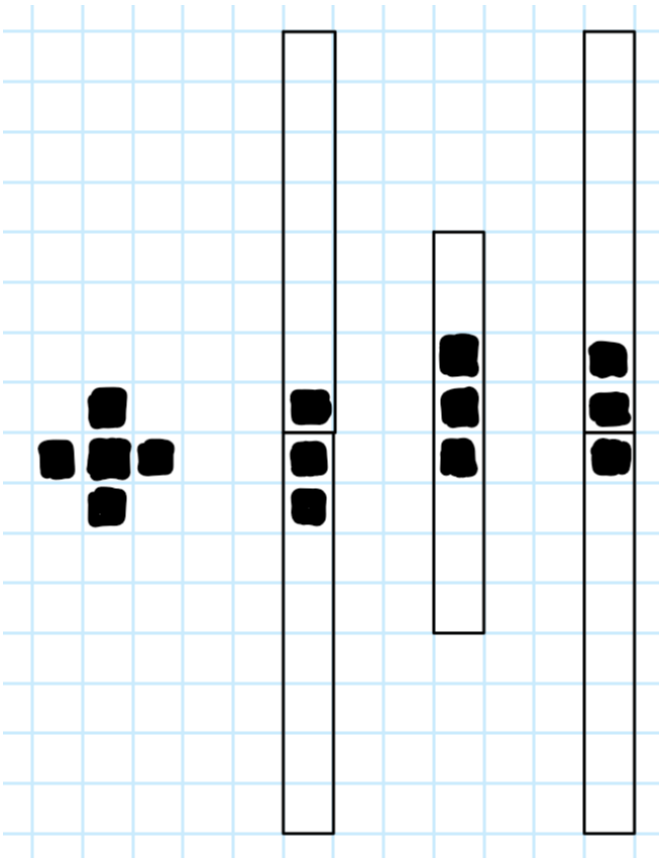
Für das Treffen des Schlägers wurde später noch eine komplexere Art der Berechnung eingeführt, um das Abprallen vom Schläger variabel zu gestalten. Hierbei kann der Ball im oberen Drittel des Schlägers nach oben abprallen, in der Mitte nach vorne und im unteren Drittel nach unten. Diese Ausgangswinkel sind unabhängig davon, aus welcher Richtung der Ball auf den Schläger getroffen ist. Hier ist die einfache Art zu sehen, wenn der Ball horizontal auf den Schläger trifft:



LCD.inc

In der LCD-Datei sind aller Unterroutinen zum LCD-Display untergebracht, die nicht mit der Initialisierung und der Kommunikation zu tun haben. Somit werden hier die Schläger und der Ball auf das Spielfeld gesetzt.
Die Schläger werden hier außerdem im Speicher gesetzt. Hat der AD-Wandler neue Werte erhalten, werden die acht Bytes im Speicher so verändert, dass die Bits, wo der Schläger in der Vertikalen zu sehen ist, gesetzt sind. Diese Schläger werden dann als ganzes aus dem Speicher an das LCD-Display übertragen.
Wird der Ball bewegt (Unterroutine moveBall in Ball.inc), muss vor der Ballbewegung die alte Ballposition gelöscht werden und ein/zwei leere/s Byte/s an die alte/n LCD-Display Position/en übertragen. Nach der Bewegung kann der Ball dann mit der Unterroutine loadBall auf das Spielfeld gesetzt werden. Für das Setzen des Balls ist es aufgrund seiner Größe wichtig, ob mehr als ein Byte übertragen werden muss. Ist mehr als ein Byte benötigt, sind bei dem Einen Byte immer die zwei LSB/MSB und bei dem anderen ein MSB/LSB gesetzt.

Mögliche Ball-Positionen im Zusammenhang mit den Bytes für den Ball, die an das Display übertragen werden. Hier ist links der Ball und in den drei Balken rechts der mittlere vertikale Streifen des Balls zu sehen:



Pseudocode:

```
int PAGES = 8;
void loadBall (byte BallPosition) {
        byte modBallPosition = BallPosition % PAGES;
        switch (modBallPosition) {
                case 0:
                        byte firstByte = 0b10000000;
                        byte secondByte = 0b00000011;
                        break;
                case 7:
                        byte firstByte = 0b00000011;
                        byte secondByte = 0b10000000;
                        break;
                default:
                        byte Byte = 0b00000111;
                        //Schiebe nun die Bits soweit, bis sie sich an der richtigen Stelle befinden
                        break;
        }
        //Übertrage das/die Byte/s durch eine zusätzliche Page-Berechnung an das Display
}
```

Programmtests

Das Programm wurde auf zwei verschiedene Weisen getestet. Zum Einen wurden Debug-Ausgaben auf der Textausgabe benutzt. Zum Anderen wurde das Spiel Pong gespielt und nach speziellen Fehlern gesucht.

Debug-Tests

Testfall	Erwartetes Ergebnis	Ergebnis
Vollständiges Drehen beider Potentiometer	Die Werte 0 – 55 werden nach der Wertetransformation ausgegeben	Nur die Werte 0 – 55 sind in der Ausgabe zu sehen

Spieltests

Testfall	Erwartetes Ergebnis	Ergebnis
Ball auf dem oberen Rand des Schlägers aufkommen lassen.	Der Ball prallt ab.	Der Ball prallt ab.
Ball auf dem unteren Rand des Schlägers aufkommen lassen.	Der Ball prallt ab.	Der Ball prallt ab.
Der Ball fliegt oben direkt am Schläger vorbei.	Der Gegner bekommt einen Punkt und der Ball wird zurückgesetzt.	Der Gegner bekommt einen Punkt und der Ball wird zurückgesetzt.
Der Ball fliegt unten direkt am Schläger vorbei.	Der Gegner bekommt einen Punkt und der Ball wird zurückgesetzt.	Der Gegner bekommt einen Punkt und der Ball wird zurückgesetzt.
Der Ball kann sich überall auf dem Spielfeld bewegen	Der Ball bewegt sich überall.	Der Ball bewegt sich überall.
Den Schläger mit dem Ball am oberen Ende treffen	Der Ball prallt am oberen Ende (obere drei Pixel) des Schlägers nach oben ab.	Der Ball prallt am oberen Ende (obere drei Pixel) des Schlägers nach oben ab.
Den Schläger mit dem Ball in der Mitte treffen	Der Ball prallt in der Mitte (mittlere drei Pixel) des Schlägers nach vorne ab.	Der Ball prallt in der Mitte (mittlere drei Pixel) des Schlägers nach vorne ab.
Den Schläger mit dem Ball am unteren Ende treffen	Der Ball prallt am unteren Ende (untere drei Pixel) des Schlägers nach unten ab.	Der Ball prallt am unteren Ende (untere drei Pixel) des Schlägers nach unten ab.

Probleme und Lösungen

Sichtbarkeit des Balls

Der Ball ist durch eine zu hohe Schnelligkeit des Balls leider nur stark gedimmt zu erkennen. Dazu führte erstmal eine Begrenzung des Hardware-Interrupt-Timers für den Ball, der um maximal den Wert $2^{16} – 1 = 65.535$ geändert werden kann.

Der erste Lösungsansatz bestand darin, dass es eine Konstante gibt, mit Hilfe derer nun dieser maximale Wert mit einer ganzzahligen Nummer multipliziert werden kann. Der Timer für die Aktualisierung des Balls kann jetzt zum Beispiel bei jedem zweiten Timer-Interrupt (Konstante = 2) den Ball auf dem Display aktualisieren. Sollte man eine Geschwindigkeit zwischen den Ganzzahligen Werten anstreben, kann man außerdem die Konstante für den Interrupt Timer ändern, bei der der Interrupt-Timer aufgerufen wird. Als Beispiel könnte man den Wert von 65.000 auf 40.000 ändern. Der Ball ist somit fast 40 % schneller.

Diese Lösung wurde allerdings verworfen, da dies zu Problemen mit der Sieben-Segment-Anzeige führte (siehe Aktualisierung Sieben-Segment-Anzeige).

Um den Ball sichtbarer zu machen, ist er nun ein wenig größer, wodurch die einzelnen Pixel länger sichtbar sind. Zusätzlich ist der Ball nun durch den Timer der Sieben-Segment-Anzeige gesteuert. Dieser Timer zählt eine Variable hoch, die im Hauptprogramm ausgelesen und auf Null zurückgesetzt wird, sobald eine bestimmte Konstante überschritten wird. Dies ist nun viel einfacher und zusätzlich der Ball nun einfacher von der Geschwindigkeit her durch eine einzige Konstante angepasst werden.

Aktualisierung Sieben-Segment-Anzeige

Leider war die Aktualisierung der Sieben-Segment-Anzeige nicht konstant, beziehungsweise konnte nicht so schnell durchgeführt werden, damit das menschliche Auge kein Flackern wahrnimmt. Das liegt an der Aktualisierung des Balls und des Schlägers auf dem Display, dessen Kommunikation viel langsamer als der Prozessor sind.

Als Lösung dafür wurde der zweite Timer, der eigentlich die Aktualisierung des Balls auf dem Display vornehmen soll, gelöscht, weil die Interrupts, die durch den Timer gesetzt werden zu lange aktiv waren und so der 7-Segment-Anzeige in die Quere kamen.

Ghosting in der Sieben-Segment-Anzeige

Bei der Sieben-Segment-Anzeige kam es zu einem Ghosting-Effekt, der dadurch geschuldet war, dass die Anzeige und das LCD-Display durch die gleiche Variable PIO_C angesprochen werden. Dadurch kam es zu einer Racing Condition, die zu Fehlern führte, wenn ein Interrupt für die Sieben-Segment-Anzeige kam, während das LCD-Display angesprochen wurde.

Gelöst wurde dieses Problem durch eine Interrupt-Sperre um die Nutzung von PIO_C beim Ansprechen des LCD-Displays (in der Datei LCD_communication.inc bei der Unterroutine LCDsend).

Literaturverzeichnis

1. https://stud.fh-wedel.de/handout/Voelker_Joerg/:
wesentliche Grundbausteine zum Erstellen des Codes und für das Verständnis der Kommunikation zwischen den peripheren Einheiten
    a. https://stud.fh-wedel.de/handout/Voelker_Joerg/Folien/T2ErsteBefehle.pdf

2. https://de.wikipedia.org/wiki/Pong:
Grundinformationen zum Spiel Pong

3. https://www.lucidchart.com/pages/:
Zum Erstellen des Programmorganisationsplans