Mikroprozessortechnik

Laborbericht 2

(LED Pendel)

|  |  |
| --- | --- |
| **Name, Vorname** | **Matrikel Nr.:** |
| Reger, Eugen | 2259469 |
| Busse, Jan-Henrik | 2258292 |
| Schurr, Lucien | 2319848 |

Leitender Professor: Prof. Dr. Pawel Buczek

Abgabedatum: 14.05.2018

Inhalt

[Übersicht 3](#_Toc514058483)

[1. Aufgabe 1 4](#_Toc514058484)

[1.1. Lösungsansatz 4](#_Toc514058485)

[1.2. Code 4](#_Toc514058486)

[1.3. Code Auswertung 5](#_Toc514058487)

[2. Aufgabe 2 6](#_Toc514058488)

[2.1. Benutzen Sie für die Herstellung der Verzögerungszeiten eine Softwareverzögerung (Riemschneider, 2016). 6](#_Toc514058489)

[2.2. Aufgabe 8](#_Toc514058490)

[2.3. Code Auswertung 10](#_Toc514058491)

[3. Aufgabe 3 12](#_Toc514058492)

[3.1. Lösungsansatz 12](#_Toc514058493)

[3.2. Code 12](#_Toc514058494)

[3.3. Code Auswertung 15](#_Toc514058495)

[4. Aufgabe 4 17](#_Toc514058496)

[4.1. Lösungsansatz 17](#_Toc514058497)

[4.2. Code Auswertung 17](#_Toc514058498)

[Fazit 18](#_Toc514058499)

[Abbildungsverzeichnis 18](#_Toc514058500)

[Literaturverzeichnis 18](#_Toc514058501)

[Anhang 19](#_Toc514058502)

## Übersicht

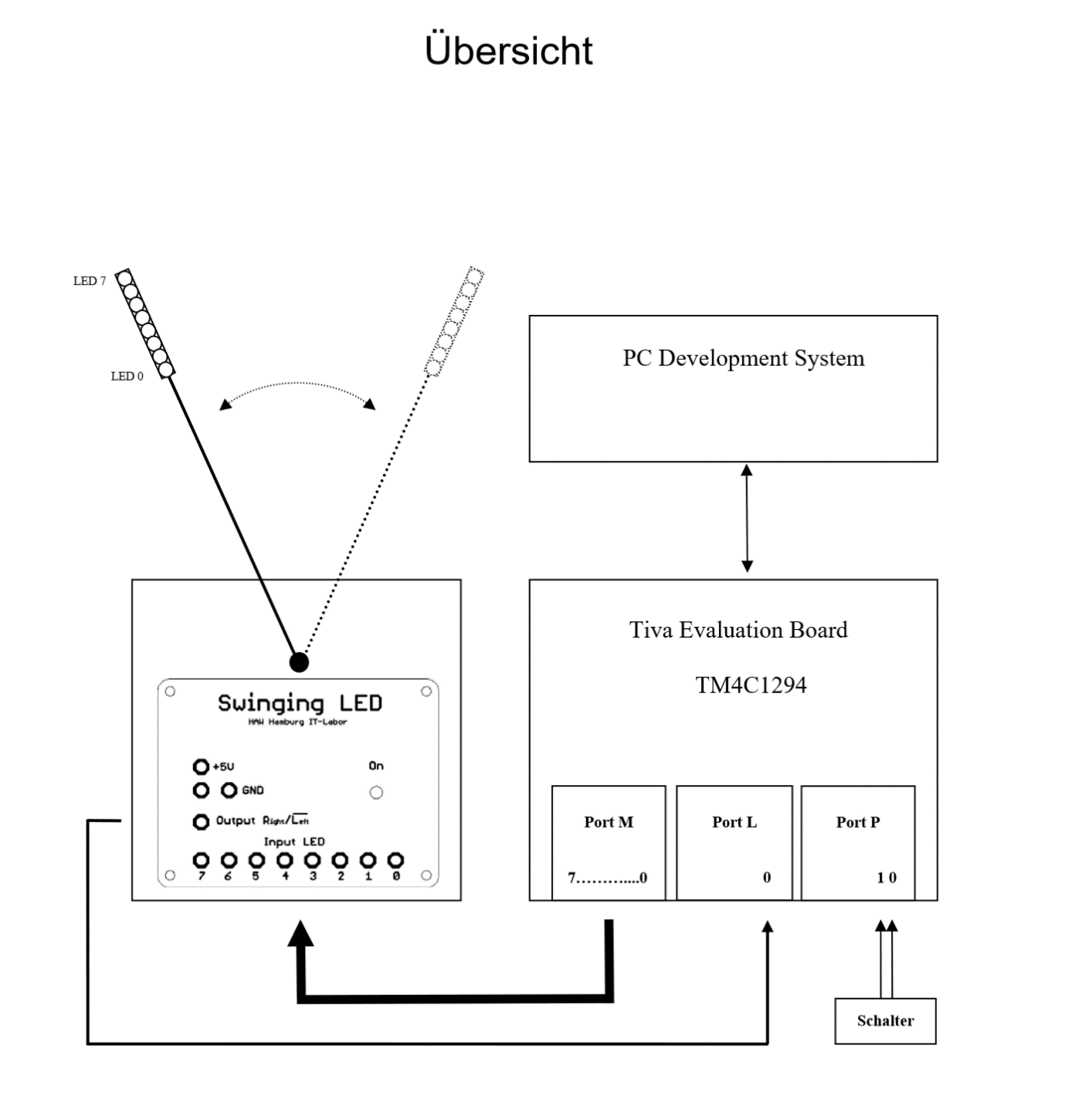


Abbildung Systemübersicht mit einer möglichen Anschlussweise der LED-Pendel Hardware (Riemschneider, 2016)

Im folgenden Versuch soll die zeitliche Verzögerung betrachtet werden. Die Verzögerung soll mithilfe einer for-Schleife, als auch mit einem Timer realisiert werden. Für den Versuch wird ein LED-Pendel mit 8 LED´s zur Verfügung gestellt. Die Pendelfrequenz beträgt nach dem Einschalten ca. 8,3 Hz und ist nicht veränderbar (Riemschneider, 2016). Mithilfe des Ports L kann der Umkehrzeitpunkt ermittelt werden (Abfragen des Signals - Potenzial von Low auf High 🡪 linker Umkehrpunkt; Potenzial von High auf Low 🡪 rechter Umkehrpunkt).

## Aufgabe 1

Schreiben Sie ein Programm, welches an dem linken und rechten Rand des Pendelbereichs jeweils einen schmalen Balken darstellt. Die Einschaltzeit der LEDs soll mit einer Softwareverzögerung (for-Schleife) realisiert werden. Experimentieren Sie mit der Einschaltzeit der LEDs, um ein ansprechendes Aussehen der Balken zu erhalten. Ermitteln Sie die sinnvolle Einschaltzeit der LEDs und den Zeitbedarf für eine halbe bzw. vollständige Pendelbewegung. Erkennen Sie die Pendel Umkehrzeitpunkte durch kontinuierliche Abfrage (polling) des Signals (Riemschneider, 2016).

### Lösungsansatz

Um die schmalen Balken an den Umkehrzeitpunkten darstellen zu können, wurde der Output () vom Pendel an unserem Port L geschaltet. Durch kontinuierliche Abfrage (polling), wird dann die Flanke erkannt. Bei Flankenänderung wurden alle 8 LEDs eingeschaltet. Nach einer kurzen Wartezeit (Verzögerungsschleife) wurden die LEDs wieder ausgeschaltet. Die Schleifendurchläufe wurden solange variiert, bis die gewünschte Balkenbreite an den Rändern eingeblendet wurde.

### Code

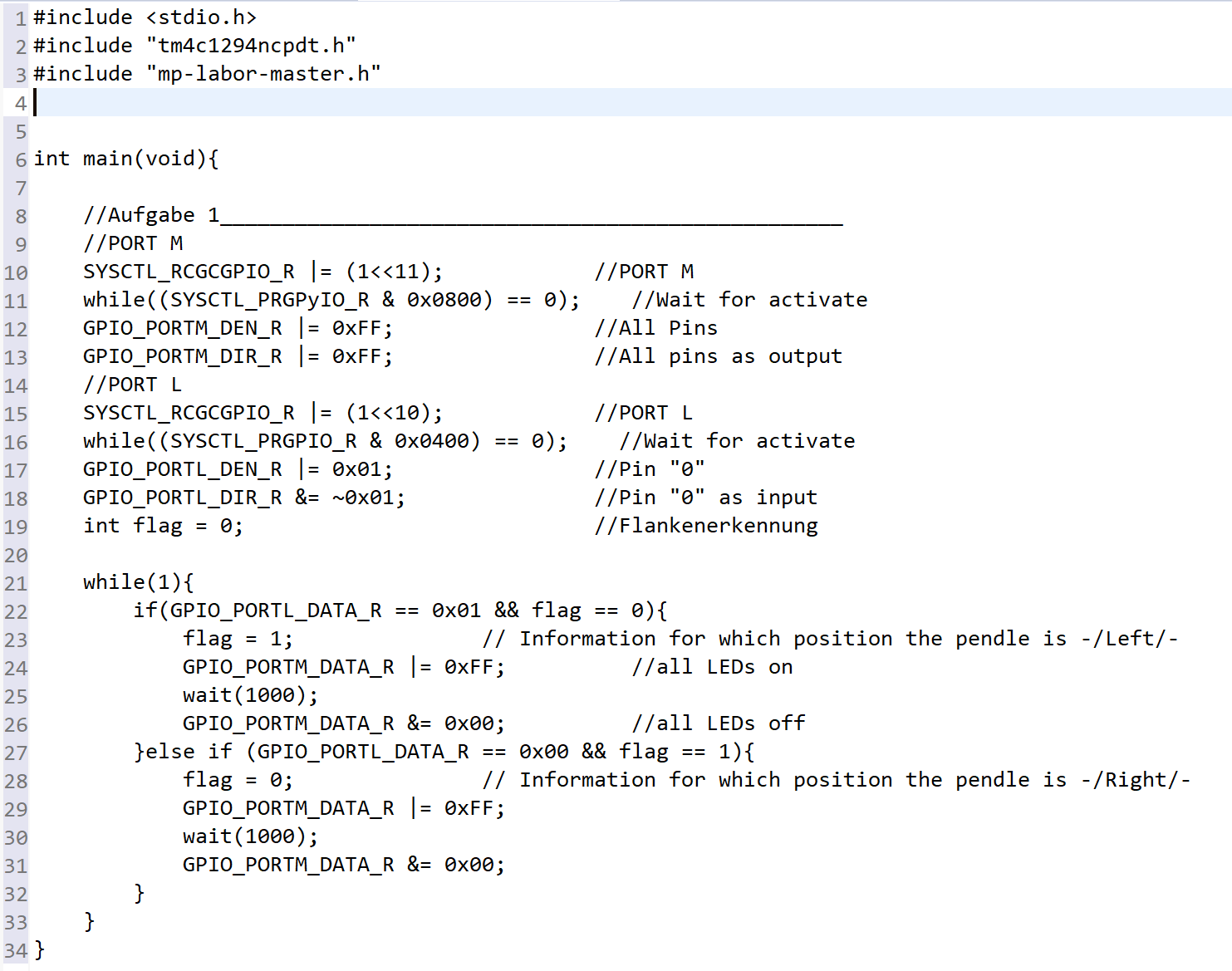


Abbildung Code Composer Studio - Main Aufgabe 1

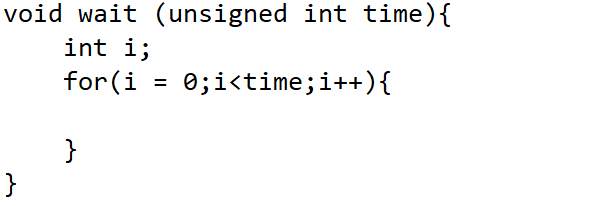


Abbildung Code Composer Studio - Ausgelagerte Softwareverzögerung

### Code Auswertung

Nach variieren der Schleifendurchgänge wurde der Ausschaltzeitpunkt so verzögert (1000 Schleifendurchläufe), das ein deutlicher Balken am Umkehrpunkt zu sehen ist. Nach der Berechnung der 1000 Schleifendurchläufe kam eine Verzögerung von 643µs raus. Zur Erfassung der halben Schwingung wurde die Ausschaltzeit solange verzögert, bis der Balken bei der Hälfte ausgeht. Die Schwierigkeit bestand darin mit bloßem Auge, während der Pendelschwingung, zu sehen wann die LEDs ausgehen. Nach 47500 Schleifendurchläufen befand sich, unsere Ansicht nach, das Pendel in der Mitte.

Die Verzögerung beträgt dann:

47500 Schleifendurchläufe = 6,43·10−4 x 45,5 = 0.03054s = 30,54ms

Angegeben wurde die halbe Periode mit ungefähr 60 ms (Riemschneider, 2016).

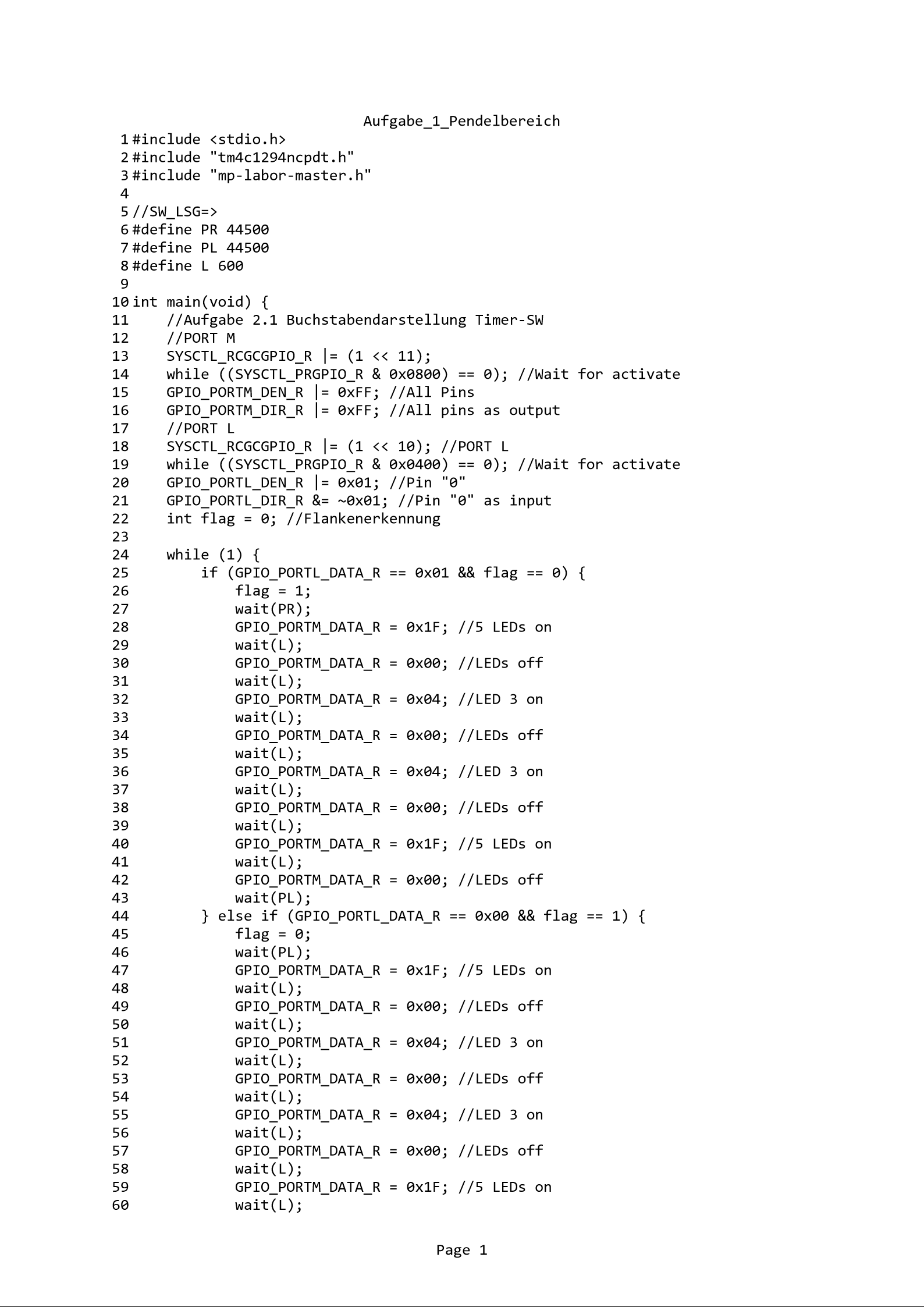
## Aufgabe 2

Schreiben Sie ein Programm zur Darstellung eines beliebigen Zeichens. Überlegen Sie sich eine sinnvolle Anordnung der Bildpunkte für ein Zeichen (Riemschneider, 2016).

### Benutzen Sie für die Herstellung der Verzögerungszeiten eine Softwareverzögerung (Riemschneider, 2016).

Um ein beliebiges Zeichen darstellen zu können, müssen die einzelnen LEDs über den Port M angesteuert werden. Nach der Abfrage ob das Pendel Links oder Rechts steht, wird mittels einer for-Schleife bis kurz vor dem Mittelpunkt gewartet. Anschließend werden die einzelne Kombination für ein H, mit einem kleinen Abstand, ausgegeben.

#### Code



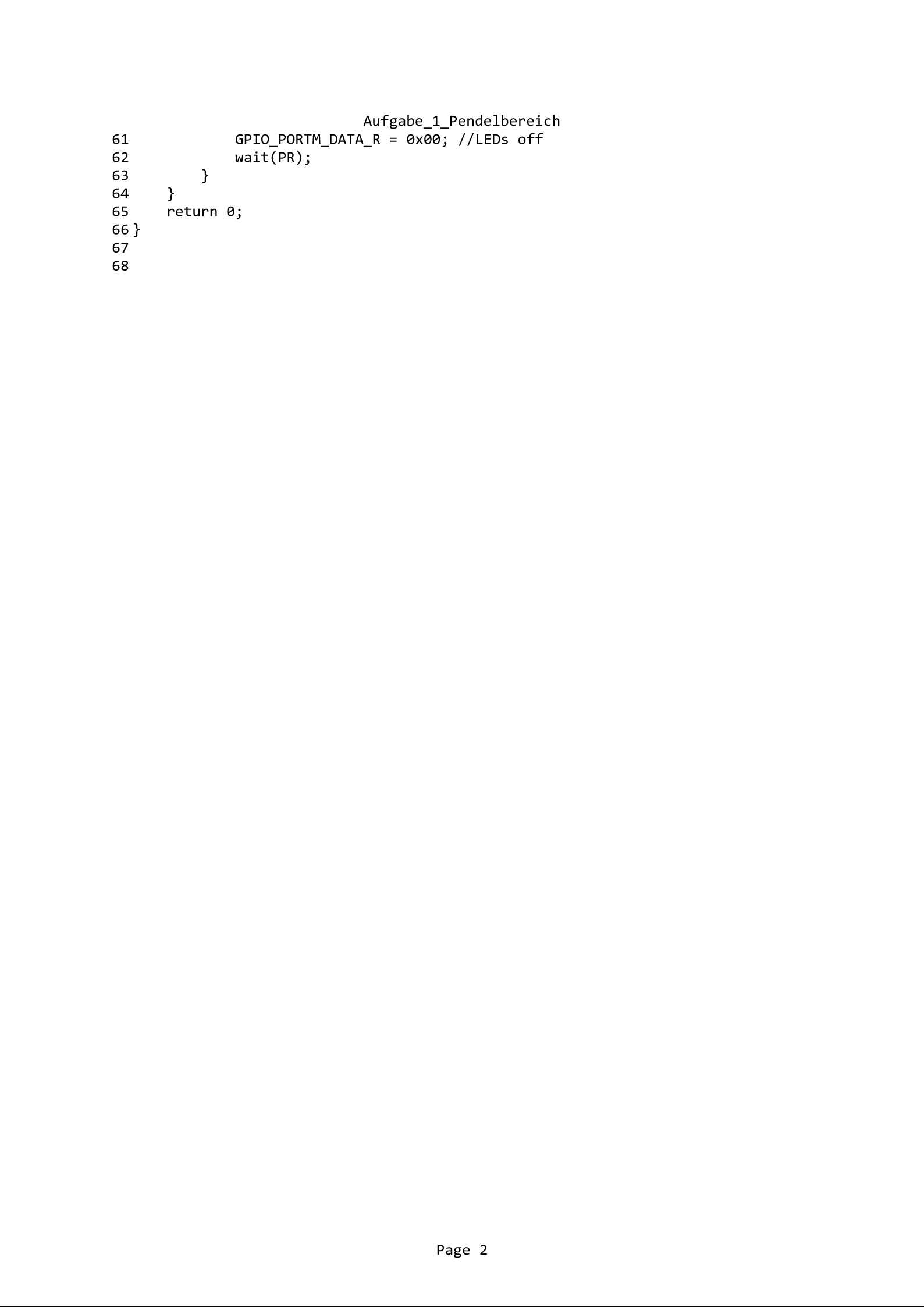
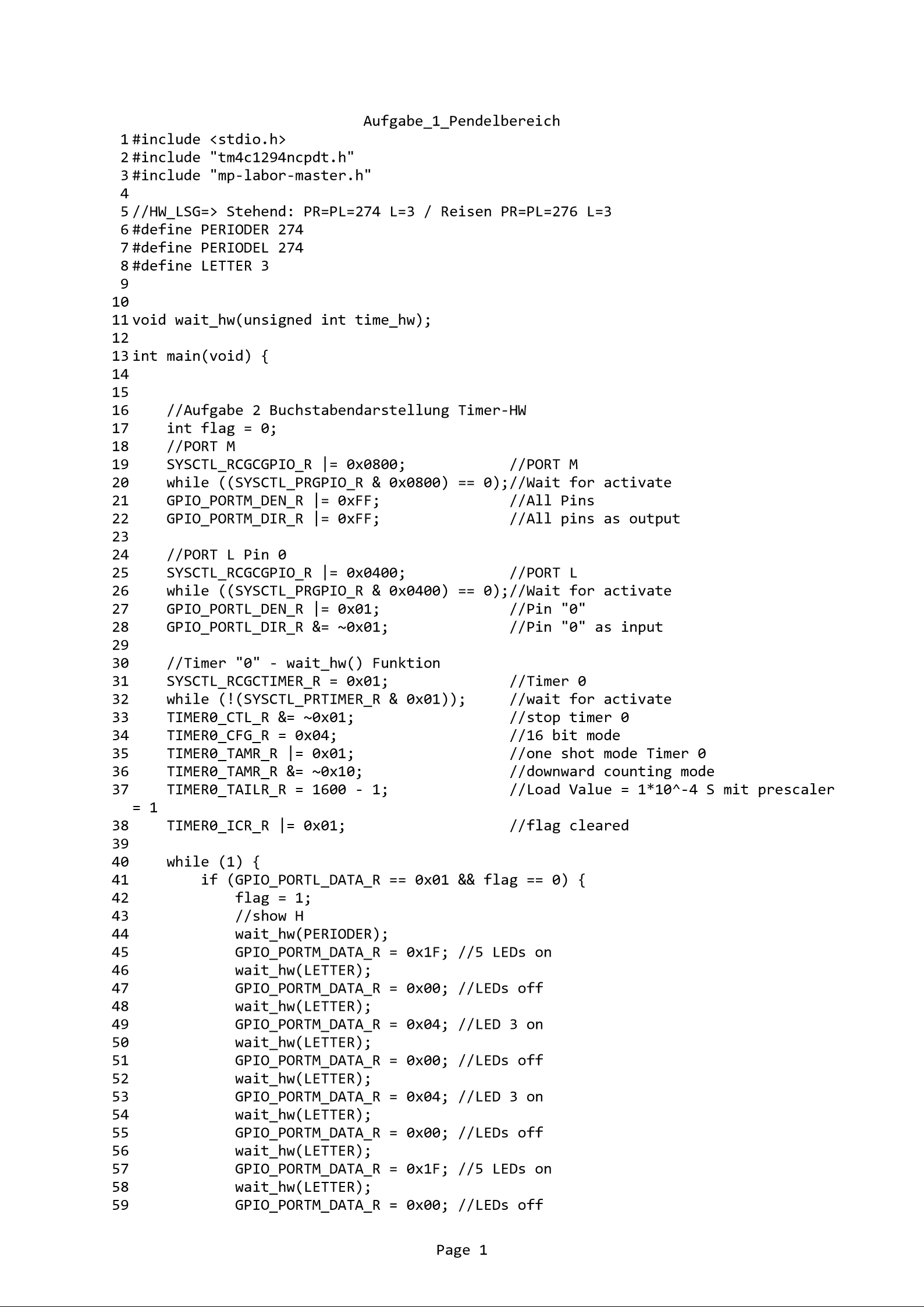


Abbildung Code Composer Studio - Main Aufgabe 2.1 mit Softwareverzögerung

### Aufgabe

Zur Umsetzung folgender Aufgabe war es gefordert einen General Purpose Timer (GPTM) zu verwenden. Hierfür wurde ein Compare Timer (Zeitgeber) in One-Shot-Mode verwendet. Der programmierte Quellcode unterscheidet sich von der vorherigen Aufgabe darin, dass zum einen im Hauptprogramm der Timer initialisiert wird und zum anderen in der warte-Funktion die for-Schleife mit dem Aufruf des Timers ersetzt wird.

#### Code



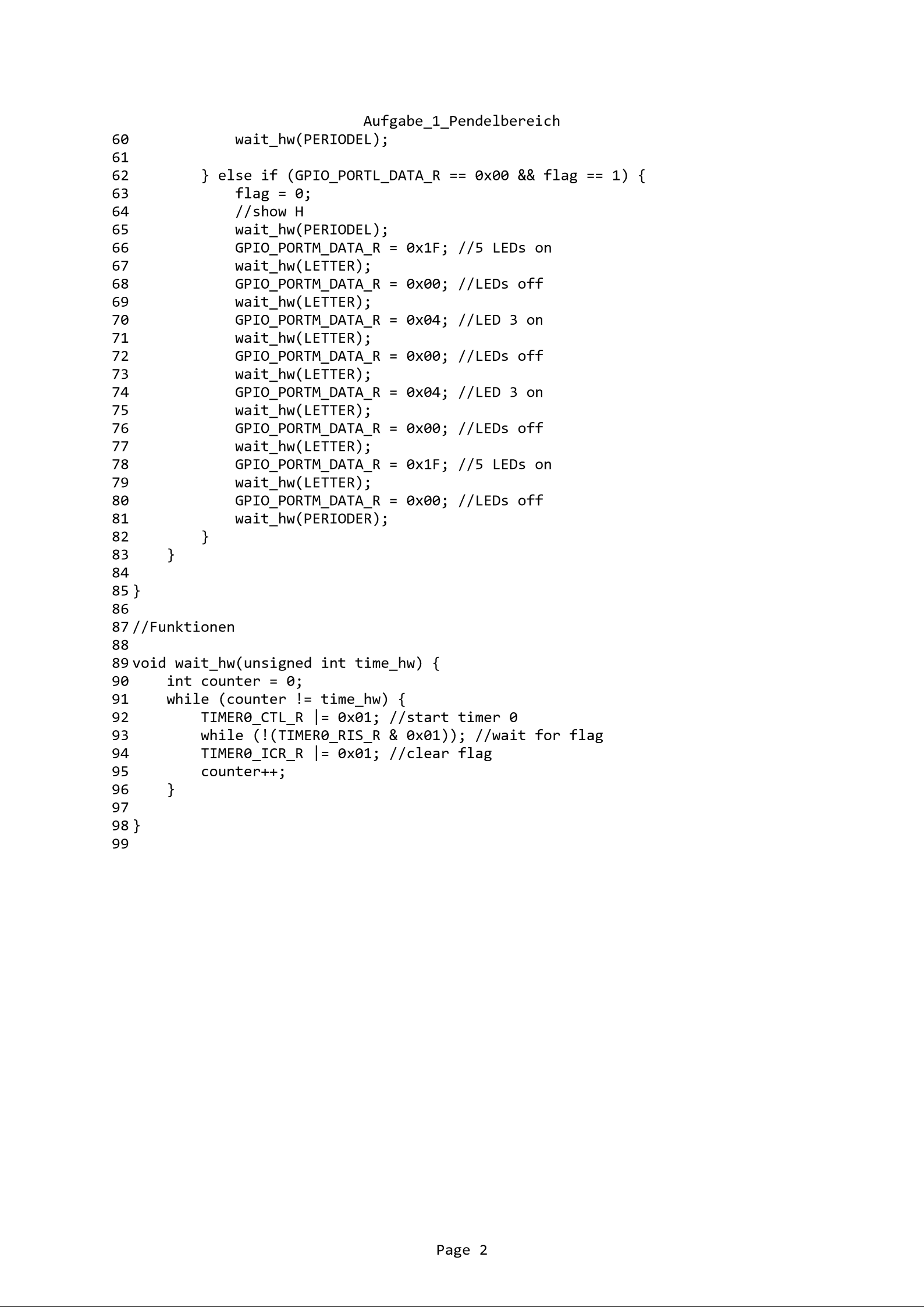


Abbildung Code Composer Studio - Main Aufgabe 2.2 mit Timer

### Code Auswertung

Im Folgenden gehe ich auf den Quellcode der Aufgabe 2 ein. Zur Optimierung des Codes, wäre es von Vorteil, ein Array für die einzelnen LED Adressen zu erstellen. Mithilfe von zwei for-Schleifen wäre es dann möglich, ein Code zu entwickeln, welcher mittels übergebenden Array (in der Main) die einzelnen Aufgaben ausführt. Der Code wäre dann wiederverwendbar, erweiterbar und kürzer. Die Verzögerung, die hier u.a. durch ein Timer realisiert wird, ist ebenfalls unabhängig vom vorherigen initialisierten Wert und setzt sich bei jedem Aufruf erneut auf sein start value.

Der Timer arbeitet dann mit der Anzahl von übergebenden Zyklen. Der nachfolgende Code wurde von mir geschrieben und ist nicht im Labor überprüft worden. Dieser dient nur zur Visualisierung.

#### Code





Abbildung Code Composer Studio - Optimierter Code für Aufgabe 2

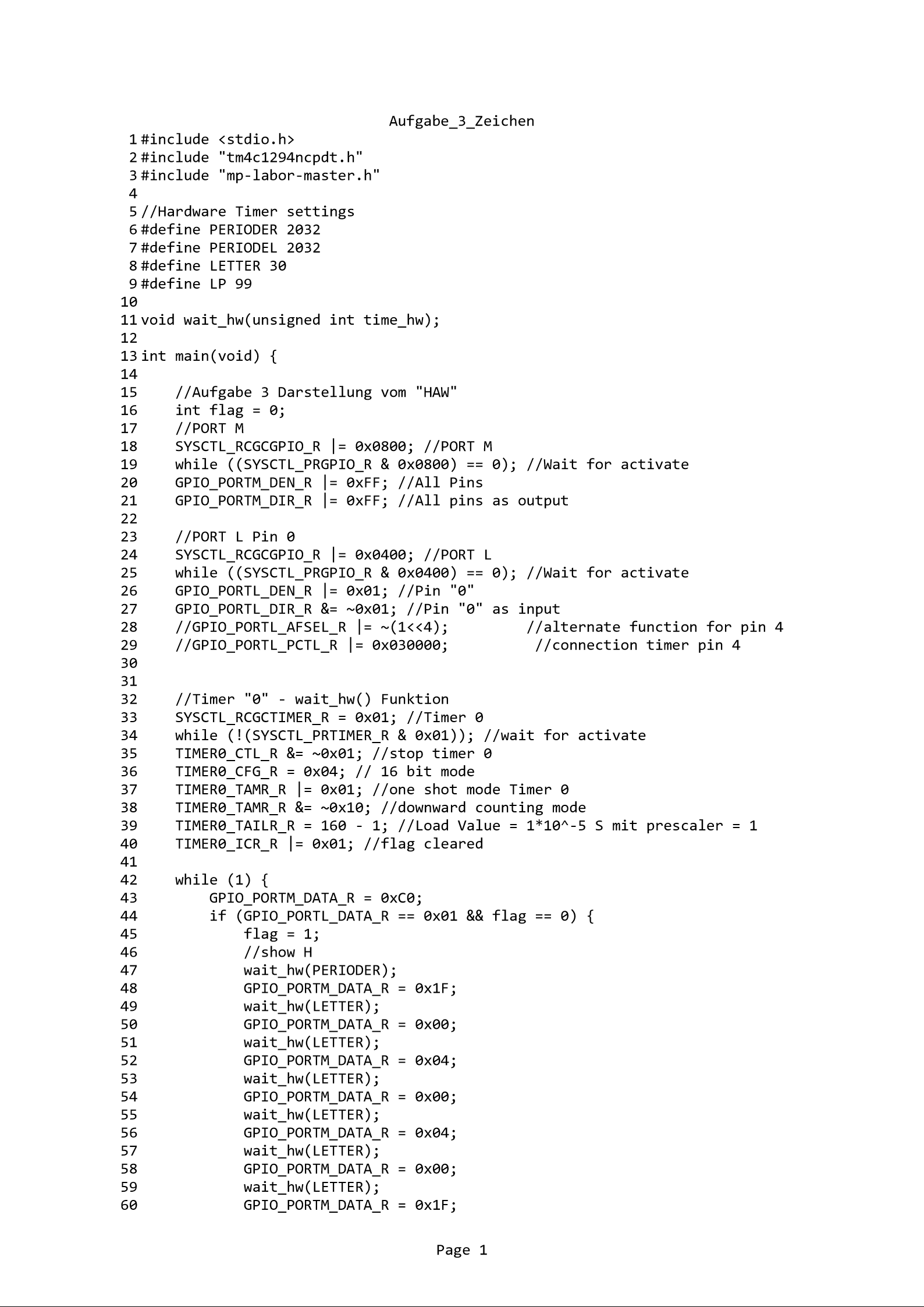
## Aufgabe 3

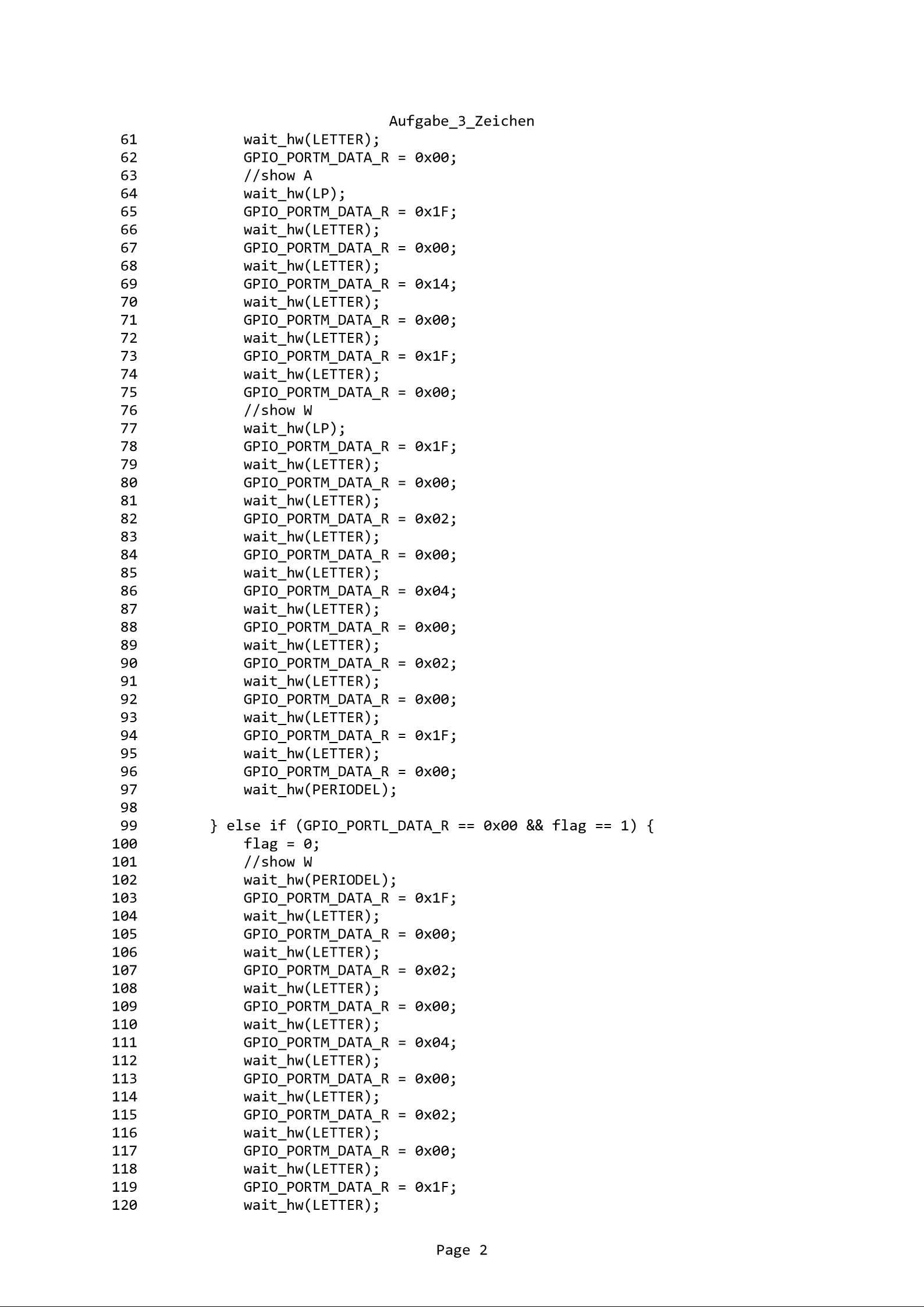
Schreiben Sie ein Programm, welches eine sinnvolle Zeichenfolge darstellt (Riemschneider, 2016).

### Lösungsansatz

Es wurde mit dem Programmcode aus der Aufgabe 2.2.1 weitergearbeitet. Der Programmcode wurde vervielfältigt, sodass die gewünschte Zeichenkette dargestellt werden kann.

### Code





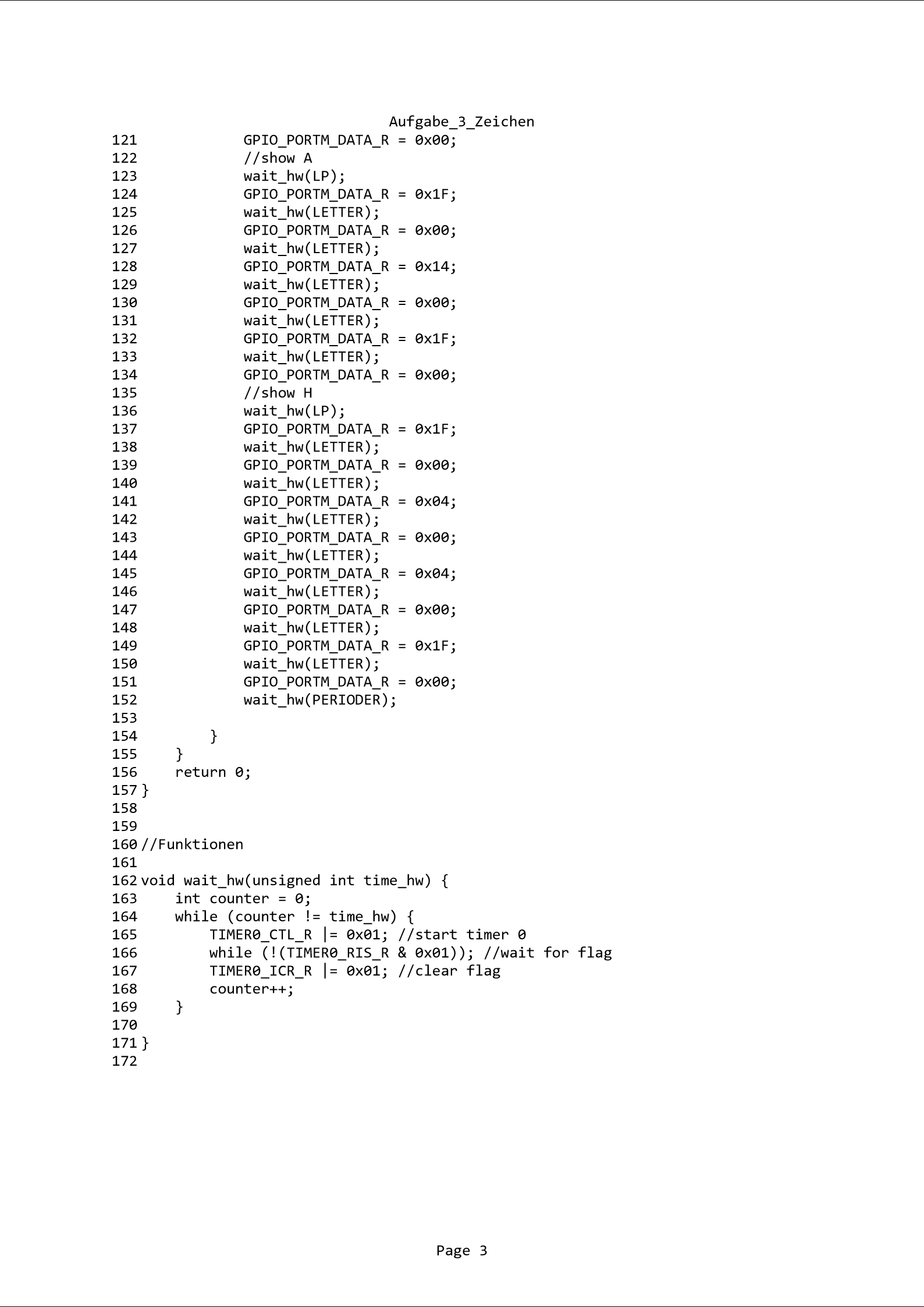


Abbildung Code Composer Studio - Main Aufgabe 3

### Code Auswertung

Wie in Aufgabe 2.

Zur Optimierung des Codes, wäre es von Vorteil, ein Array für die einzelnen LED Adressen zu erstellen. Mithilfe zwei ausgelagerten Funktionen mit jeweils zwei for-Schleifen wäre es dann möglich, ein Code zu entwickeln, welcher mittels übergebendem Array (in der Main) die einzelnen Buchstaben hintereinanderschreibt. Der nachfolgende Code wurde von mir geschrieben und ist nicht im Labor überprüft worden. Dieser dient nur zur Visualisierung.

#### Code



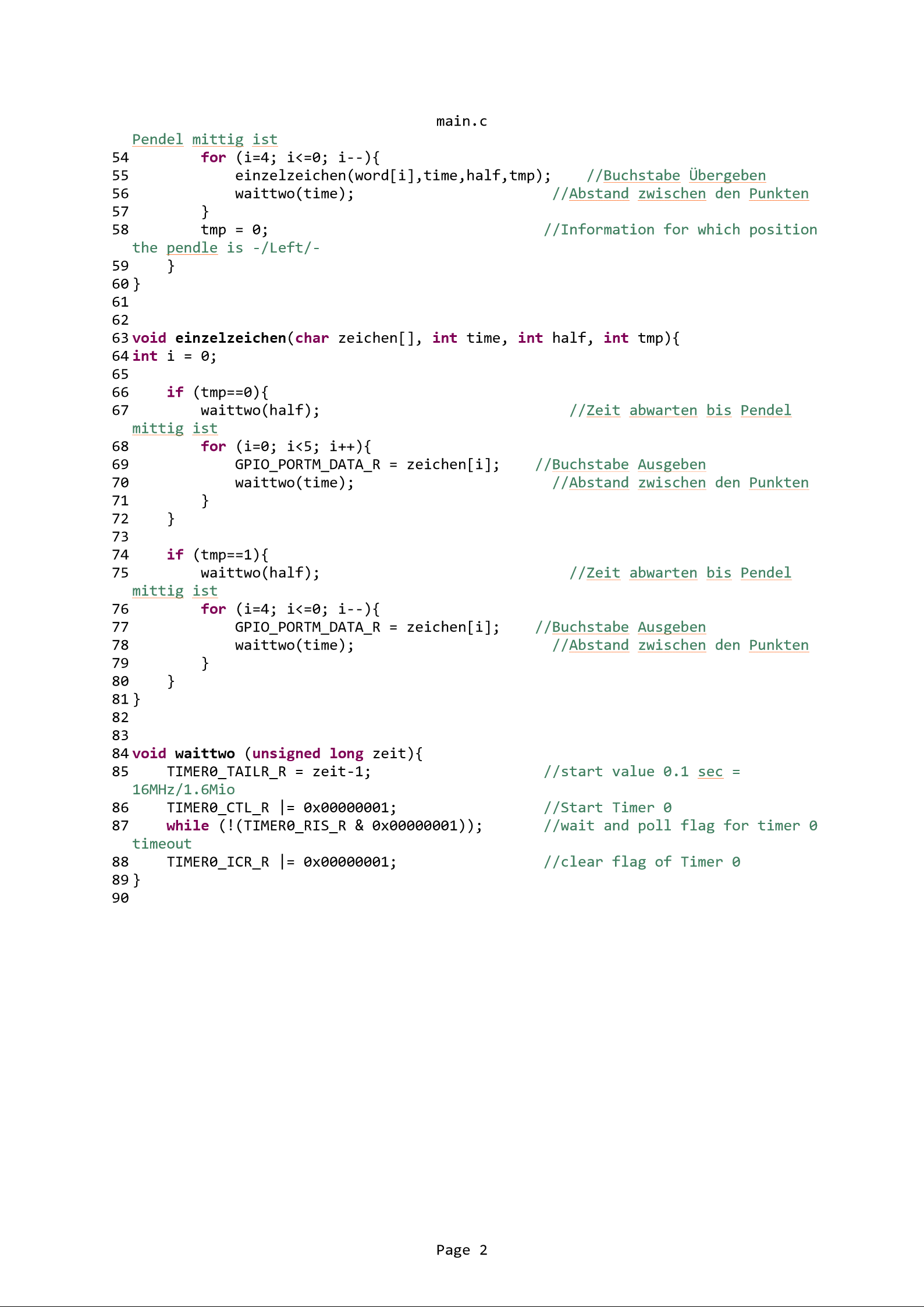


Abbildung Code Composer Studio - Optimierter Code für Aufgabe 3

## Aufgabe 4

Schreiben Sie ein Programm, welches vier verschiedene sinnvolle Zeichenfolgen darstellt. Die Auswahl der jeweiligen Zeichenfolgen erfolgt über die an z.B. Port P angeschlossenen Schalter. Benutzen Sie für die Herstellung der Verzögerungszeiten einen Timer (Riemschneider, 2016).

### Lösungsansatz

Mithilfe der Abfrage des Ports L, ist es möglich eine Hilfsvariable als Value zu nutzen um in das jeweilige Switch Case zu gelangen. In den Switch Cases befindet sich dann die eigentliche Zeichenkette. Nach viermaligem Drücken des Knopfes am Port L lädt sich die Hilfsvariable mit eins und fängt von vorne an. Wegen der Komplexität, befindet sich der Code im Anhang.

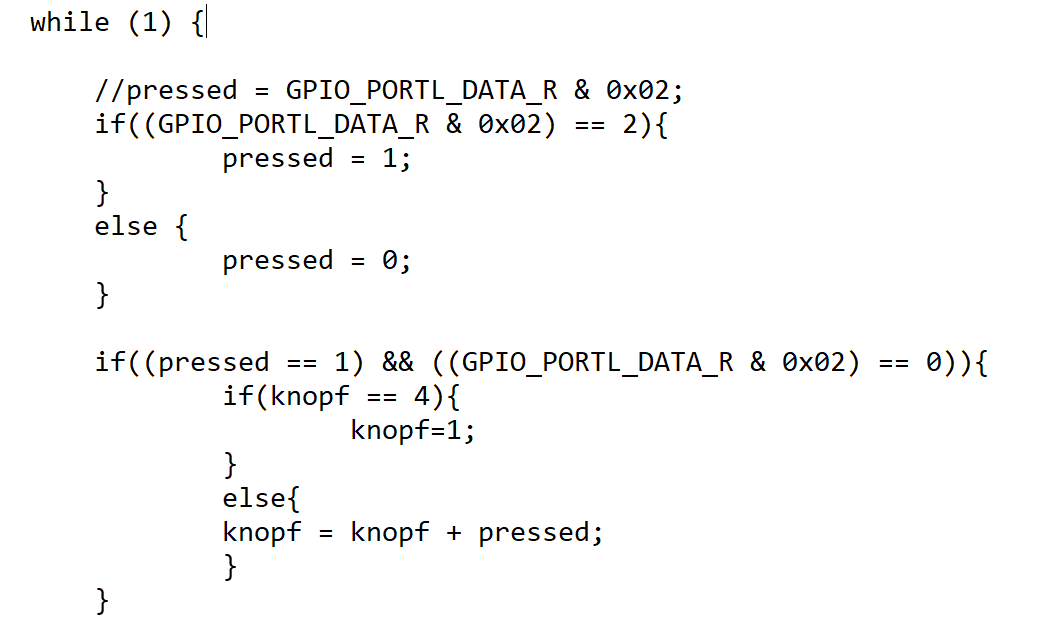


Abbildung Code Composer Studio - Abfrage vom Port L

### Code Auswertung

Auch hier ist es, wie in den anderen Aufgaben wichtig, die einzelnen Teilaufgaben zu teilen. Wie in der Vorlesung am 08.05.18 von Herrn Buczek erklärt, sollen Funktionen wiederverwendet und unabhängig von der eigentlichen Initialisierung sein. Des Weiteren wäre es in dieser Hinsicht besser, mit Pointern zu Arbeiten. Ein optimierter Code ist in diesem Fall nicht vorgesehen. Um eine kurze Erläuterung beizufügen, wäre der optimierte Code nach einer Abfrage des Knopfdruckes ein Switch Case bei dem man einen Pointer und die jeweiligen Zeiten übergibt. Die Ausgabe zum Pendel ist wie in Aufgabe 3 gleichgeblieben.

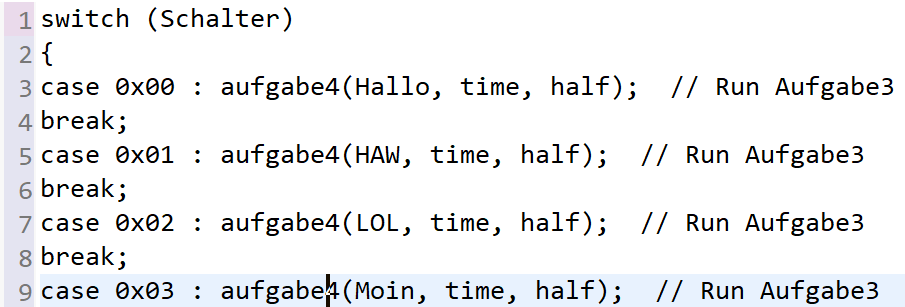


Abbildung Code Composer Studio - Optimierte Abfrage vom Port L

## Fazit

Da das Labor auch außerhalb der Praktikumszeiten zur Verfügung stand, war es uns möglich, den vorher geschriebenen Programmcode, praktisch zu testen. Das Labor war eine gute Gelegenheit, sich auch mit dem Timer auseinander setzen zu können. Die Softwareverzögerung war an sich eine reine programmtechnische Angelegenheit und war im Grunde ein Stück C Kenntnis. Uns war nicht bewusst, dass zur Vorbereitung die Aufgaben 1,2.1 und 3 reichen würden um zu bestehen. Leider haben wir dann sehr viel Zeit investiert um alle Aufgaben zu bewältigen.

## Abbildungsverzeichnis

[Abbildung 1 Systemübersicht mit einer möglichen Anschlussweise der LED-Pendel Hardware (Riemschneider, 2016) 3](#_Toc514058604)

[Abbildung 2 Code Composer Studio - Main Aufgabe 1 4](#_Toc514058605)

[Abbildung 3 Code Composer Studio - Ausgelagerte Softwareverzögerung 4](#_Toc514058606)

[Abbildung 4 Code Composer Studio - Main Aufgabe 2.1 mit Softwareverzögerung 7](#_Toc514058607)

[Abbildung 5 Code Composer Studio - Main Aufgabe 2.2 mit Timer 9](#_Toc514058608)

[Abbildung 6 Code Composer Studio - Optimierter Code für Aufgabe 2 11](#_Toc514058609)

[Abbildung 7 Code Composer Studio - Main Aufgabe 3 14](#_Toc514058610)

[Abbildung 8 Code Composer Studio - Optimierter Code für Aufgabe 3 16](#_Toc514058611)

[Abbildung 9 Code Composer Studio - Abfrage vom Port L 17](#_Toc514058612)

[Abbildung 10 Code Composer Studio - Optimierte Abfrage vom Port L 17](#_Toc514058613)

## Literaturverzeichnis

Riemschneider. (2016). *Laboraufgabe 2 LED Pendel (Swinging LED).*

## Anhang



