

Prof. Dr. Paweł Buczek

Rami Chaari

Felix Rehaag

23.10.2019

16.10.2019

4/1/6

4/1/6

Inhalt

[*Abbildungsverzeichnis* 2](#_Toc22395610)

[*Einleitung:* 2](#_Toc22395611)

[*Benutzte Laborgeräte:* 3](#_Toc22395612)

[*Laufzeitverzögerung in Mikrocontrollern* 4](#_Toc22395613)

[Ausführung 4](#_Toc22395614)

[Messwerte 4](#_Toc22395615)

[Auswertung 5](#_Toc22395616)

[*Signallaufzeitmessung* 5](#_Toc22395617)

[Ausführung 5](#_Toc22395618)

[Gemessene Laufzeiten 6](#_Toc22395619)

[Auswertung 7](#_Toc22395620)

[*C-Programm zum Einlesen einer Taste der Hexadezimaltastatur und Ausgabe auf dem Terminal* 7](#_Toc22395621)

[Ausführung 7](#_Toc22395622)

[Source Code 8](#_Toc22395623)

[*Fazit* 10](#_Toc22395624)

# Abbildungsverzeichnis

[Abbildung 1: Hexadezimal Tastatur 3](#_Toc22395555)

[Abbildung 2: C-Code für die Generierung eines Rechteckimpuls 4](#_Toc22395556)

[Abbildung 3: Verzögerungszeit in Abhängigkeit von ON 5](#_Toc22395557)

[Abbildung 4: Laufzeitverzögerung beim Sprung von 1 auf 0 6](#_Toc22395558)

[Abbildung 5: Laufzeitverzögerung beim Sprung von 0 auf 1 7](#_Toc22395559)

# Einleitung:

Im ersten Laborversuch geht es darum, ein C-Programm für die Bedienung einer Hexadezimaltastatur (Abb. 1) zu schreiben. Um dieses zu realisieren, soll erstmal die Laufzeitverzögerung in Mikrocomputerschaltungen gemessen werden. Anschließend soll die Verzögerungszeit einer wait()-Funktion mit unterschiedlichen Variablen gemessen werden. Zu guter Letzt sollen die gemessen Zeiten im C-Programm benutzt werden, um die Tastatur optimal benutzen zu können.



Abbildung 1: Hexadezimal Tastatur

# Benutzte Laborgeräte:

Für die Laboraufgaben wurden folgende Geräte benutzt:

* Mikrocontroller (Tiva TM4C1294)
* Hexadezimaltastatur
* Oszilloskop

# Laufzeitverzögerung in Mikrocontrollern

## Ausführung

Um die Laufzeitverzögerung im Mikrokontroller zu ermitteln, haben wir ein Rechteckimpuls am Ausgangsport PM0 mit Hilfe einer wait()-Funktion (warte(ON)) im C-Programm (Abbildung 2) generiert. Dabei hatten wir verschiedene Werte für ON benutzt, und jedes Mal die Dauer zwischen einer High- und Low-Flanke gemessen mittels Oszilloskops.



Abbildung 2: C-Code für die Generierung eines Rechteckimpuls

## Messwerte

|  |  |
| --- | --- |
| Wert für die Variabel ON | Gemessene Laufzeit in µs |
| 10 | 7,2 |
| 100 | 63,4 |
| 1000 | 630 |
| 10000 | 7000 |
| 100000 | 620100 |
| 1600000 | 980000 |

Tabelle 1: Laufzeit in Abhängigkeit von ON

Abbildung 3: Verzögerungszeit in Abhängigkeit von ON

## Auswertung

Der Wert von der Variabel ON ist letztendlich die Anzahl der Schleifendurchläufe. Anhand der Tabelle und des Diagrammes kann man feststellen, dass sich die Verzögerungszeit und die Anzahl der Schleifendurchläufe linear verhalten.

Um die gewünschte Wartezeit von 1s zu erreichen, muss man die Variabel ON auf ungefähr 1600000 setzen.

# Signallaufzeitmessung

## Ausführung

Die Signallaufzeit beschreibt die Laufzeit, die das Signal vom Ausgang des Mikrokontrollers über die Hexadezimaltastatur bis zum Eingang des Mikrokontrollers benötigt. Um diese zu messen, haben wir ein C-Programm geschrieben (Abbildung 5), dass einen periodischen Rechteckimpuls auf PM(0) ausgibt. Mittels Oszilloskops messen wir am Ausgang der Hexadezimaltastatur die benötigte Laufzeit um vom High- zum Low-pegel bzw. vom Low- zum High-pegel zu schalten.

Dafür werden aus dem Datasheet des Tiva TM4C1294 die Low- und High-Pegel (Tabelle 2) am Mikrocomputer benötigt.

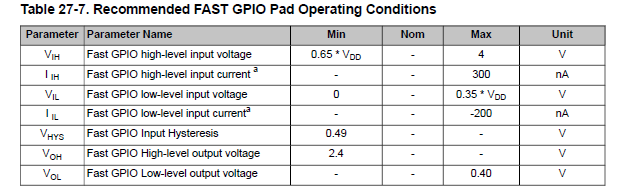


Tabelle 2: High- und Low-Pegel des Mikrocomputer Tiva TM4C1294

## Gemessene Laufzeiten

Am Oszilloskop wurden 2 Spannungen gemessen:

* Spannung 1 in Gelb: Ausgang am Port der Hexadezimaltastatur
* Spannung 2 in Hellblau: Ausgang am Port des Mikrocomputers

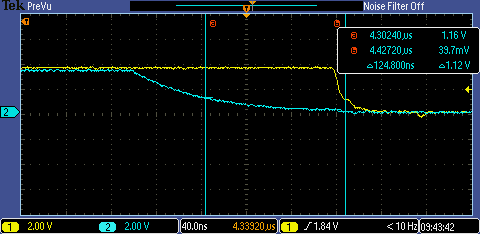


Abbildung 4: Laufzeitverzögerung beim Sprung von 1 auf 0

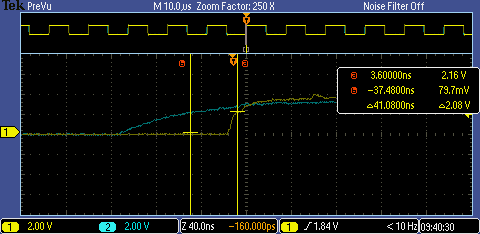


Abbildung 5: Laufzeitverzögerung beim Sprung von 0 auf 1

Bei der Auswertung der Oszilloskop-Graphen ergeben sich folgende Laufzeitverzögerungen:

* High- zu Low-Pegel: (Abbildung 4)
* Low- zu High-Pegel: (Abbildung 5)

## Auswertung

Die Signallaufzeiten zwischen Ausgang des Mikrocomputers und Eingang der Hexadezimaltastatur müssen berücksichtigt werden, um die Funktionalität beider Komponenten miteinander zu sichern.

# C-Programm zum Einlesen einer Taste der Hexadezimaltastatur und Ausgabe auf dem Terminal

## Ausführung

Zur Bedienung der Hexadezimaltastatur soll ein C-Programm geschrieben werden. Dabei sollen bestimmte Anforderungen durch erfüllt werden:

* **Anforderung 1**: Jede Taste soll nur einmal dargestellt werden, solange die Taste gedrückt ist.
* **Anforderung 2**: Werden mehrere Tasten gleichzeitig gedrückt, so soll eine Fehlermeldung ausgegeben werden.
* **Anforderung 3**: Ergänzend zu Anforderung 1, wird eine Taste länger als 1 s gedrückt, so soll diese in Abstand von einer Sekunde ausgegeben werden.

## Source Code

#include "tm4c1294ncpdt.h" //Einbinden der MC-Header

#include "stdio.h"

#define FOUR 4

#define TWENTY 20

#define DELAY 300000 // Um die Aufgabe ordnungsgemäß zu erfüllen muss DELAY mit WARTEN ersetzt werden. Wenn WARTEN verwendet wird,

#define WARTEN 1600000 // wird nach der Eingabe eine Zeit von etwa 1s bis zur nächsten Eingabe gewartet. Mit DELAY ist die

#define FEHLERT 500000 // Ansprechzeit (mit ~0,2s Verzögerung) der Tastatur angenehmer.

#define PORTBESCHR 1

**void** warte(**unsigned** **long**);

**void** Tastaturabfrage(**void**);

**int** main(){

SYSCTL\_RCGCGPIO\_R |= 0x800; // Port M clock ini

**while** ((SYSCTL\_PRGPIO\_R & 0x00000800) ==0); // Port M ready ?

GPIO\_PORTM\_DEN\_R = 0xFF;

GPIO\_PORTM\_DIR\_R = 0x0F;

GPIO\_PORTM\_DATA\_R = 0xFF;

**while**(1){

Tastaturabfrage();

}

}

**void** Tastaturabfrage(**void**){

**unsigned** **int** i = 0;

**for**(i= 0;i<FOUR;i++){

**switch** (i){

**case** 0:

GPIO\_PORTM\_DATA\_R = 0xFE;

**break**;

**case** 1:

GPIO\_PORTM\_DATA\_R = 0xFD;

**break**;

**case** 2:

GPIO\_PORTM\_DATA\_R = 0xFB;

**break**;

**case** 3:

GPIO\_PORTM\_DATA\_R = 0xF7;

**break**;

}

warte(PORTBESCHR);

**switch**(GPIO\_PORTM\_DATA\_R){

**case** 0xEE :

printf("1\n");

warte(DELAY);

**break**;

**case** 0xDE:

printf("4\n");

warte(DELAY);

**break**;

**case** 0xBE :

printf("7\n");

warte(DELAY);

**break**;

**case** 0x7E :

printf("A\n");

warte(DELAY);

**break**;

**case** 0xED :

printf("2\n");

warte(DELAY);

**break**;

**case** 0xDD :

printf("5\n");

warte(DELAY);

**break**;

**case** 0xBD :

printf("8\n");

warte(DELAY);

**break**;

**case** 0x7D :

printf("0\n");

warte(DELAY);

**break**;

**case** 0xEB :

printf("3\n");

warte(DELAY);

**break**;

**case** 0xDB :

printf("6\n");

warte(DELAY);

**break**;

**case** 0xBB :

printf("9\n");

warte(DELAY);

**break**;

**case** 0x7B :

printf("B\n");

warte(DELAY);

**break**;

**case** 0xE7 :

printf("F\n");

warte(DELAY);

**break**;

**case** 0xD7 :

printf("E\n");

warte(DELAY);

**break**;

**case** 0xB7 :

printf("D\n");

warte(DELAY);

**break**;

**case** 0x77 :

printf("C\n");

warte(DELAY);

**break**;

**case** 0xFE:

**break**;

**case** 0xFD:

**break**;

**case** 0xFB:

**break**;

**case** 0xF7 :

**break**;

**default** :

printf("Fehler\n");

warte(FEHLERT);

**break**;

}

}

}

**void** warte (**unsigned** **long** zeit){ //Wartefunktion

**unsigned** **long** j ;

**for** (j = 0 ; j < zeit ; j++ ) ;

}

# Fazit

Aus dem ersten Laborversuch wurde deutlich, dass die Signallaufzeitverzögerungen am Eingang und Ausgang des Mikrocomputers berücksichtigt werden müssen, um die Funktionalität mit der Außenwelt zu gewährleisten. Dies erfolgt durch die Benutzung von wait()-Funktionen mit unterschiedlichen Parametern, je nachdem wie lange die Signallaufzeitverzögerung dauert.

Außerdem konnte festgestellt werden, dass die Funktionsweise der verwendeten

Hexadezimaltastatur nicht einwandfrei ist. Wird eine Taste gedrückt und innerhalb einer Sekunde eine andere Taste gedrückt, so wird nur die erste Taste erkannt und auf der Konsole ausgegeben. Darüber hinaus wird kein Fehler erkannt, wenn zwei Tasten einer gleichen Spalte gedrückt werden, dies liegt an der elektronischen Bauweise der Hexadezimaltastatur.