Hochschule für Angewandte Wissenschaften Hamburg
Fachbereich Elektrotechnik und Informatik
Labor für Informationstechnik - Mikroprozessortechnik
Praktikum Computertechnik

Aufgabe 5: Eingaben über eine Hexadezimal-Tastatur

Semester/Gruppe/Team:	Abgabedatum:	Protokollführer:
2/2/4		Anton Neike
Versuchstag: 03.04.19		**weitere Versuchsteilnehmer:** Emin Hasanov, Davud Kartoglu
Hochschullehrer: Prof. Dr.-Ing Leutelt		
Kommentar des Hochschullehrers:		

1 Ausarbeitung

In die schriftliche Ausarbeitung der Versuche gehört unter anderem:

- eine kurze Beschreibung der Aufgabenstellung,
- eine Beschreibung des Lösungsansatzes, des Algorithmus oder der Messmethode,
- bei komplexen Programmteilen ein kommentierter Programmablaufplan, oder ein Struktogramm.
- Listings der Programme
- bei Messungen eine Beschreibung des Messaufbaus und der zu berücksichtigenden Effekte einschließlich relevanter Geräteeinstellungen
- Erläuterungen der Betriebsmodi der Schnittstellen, Belegung der Eingabe- und Ausgabeports (ggf. mit Schaltbild)
- eine kurze Diskussion der Versuchs- und Messergebnisse.

Inhalt

Einleitung & Laborgeräte

Einleitung

Im ersten Laborversuch befassen wir uns hauptsächlich mit dem Mikrocontroller und der Hexadezimaltastatur (s. Abb. 1). Dabei soll das zeitliche Verhalten zwischen µ-Controller und Hexadezimaltastatur untersucht werden. Mithilfe einer C-Funktion ist dieses Verhalten für das Hauptprogramm unter 4.3 schließlich zu kompensieren. Dieses Hauptprogramm ist so zu entwerfen, dass alle gestellten Anforderungen erfüllt werden.

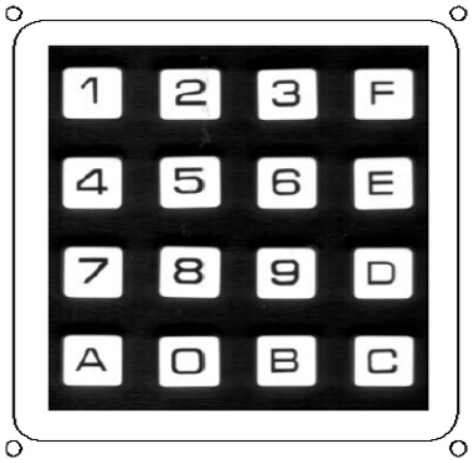

Abb. 1: Hexadezimaltastatur mit je vier Ein-/Ausgängen

Laborgeräte

Für die Laboraufgaben wurde das folgende Equipment verwendet:

- Mikrocomputer (Tiva TM4C1294)
- Hexadezimaltastatur
- Oszilloskop
- Software: CSS (von der Firma Texas Instruments)

Aufgabe 4.1: Ausführungszeitmessung mit dem Oszilloskop

In diesen Aufgabenteil wird ein periodischer Rechteckimpuls an einem Ausgangspin des Mikrocomputers erzeugt. Unter Verwendung einer Warteschleife (C -Funktion) konnte die Periodendauer des Rechtecksignals durch die variierende Zahl der Schleifendurchgänge bestimmt werden. Das Ziel dieser Aufgabe liegt darin, die Verzögerungszeit für eine bestimmte Anzahl von Durchläufen mithilfe des Oszilloskops zu messen und zu untersuchen, ob es ein lineares Verhältnis zwischen der gemessen Verzögerungszeit und der Anzahl der

Durchläufe vorliegt. Wir erwarten ein lineares Verhältnis von Schleifendurchläufen zur Ausführungszeit. Wir erwarten allerdings einen Offset, da der Funktionsaufruf auch Zeit benötigt.

Tabelle (Verzögerungszeiten)

Anzahl der Durchläufe (Schleifen)	**Verzögerungszeit [µs]**
10	7,3
100	62,2
1000	640
10000	7040
100000	76200

Abb. 2: Verhalten zwischen Schleifendurchläufe und der Verzögerungsdauer – Tabellarische Darstellung

Funktionsverlauf

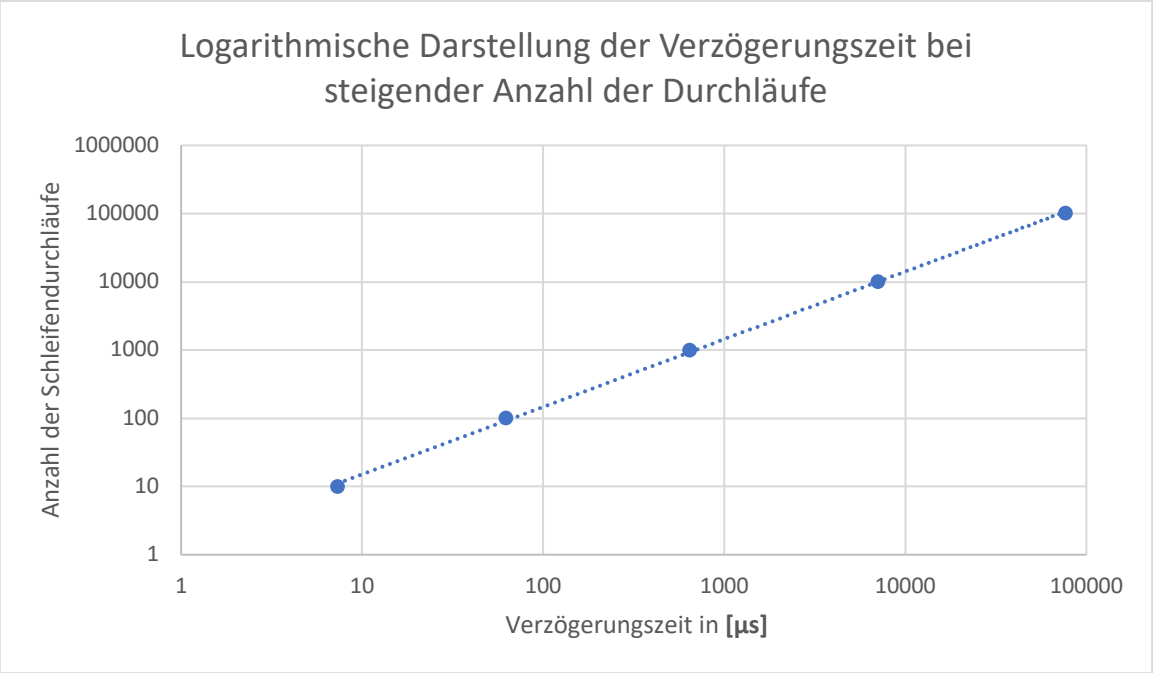

Abb. 3: Verhalten zwischen Schleifendurchläufe und der Verzögerungsdauer

Auswertung

Anhand des Funktionsverlaufs in Abbildung 3 ist klar festzustellen, dass sich Periodendauer und die Anzahl der Schleifendurchläufe linear Verhalten, wobei zu schließen ist, dass mit steigenden Koordinatenwerten sich vermutlich eine leichte Abweichung in der Linearität einstellen sollte. Die „Wait-Funktion“ wird benötigt, da die Zyklenzeit des Programmablaufes schneller ist, als das physikalische Signal vom Ausgang bis zum Eingang des Ports M benötigt. Wir können auch errechnen, dass die durschnitt Verzögerungszeit pro Schleifendurchlauf in etwa 0,7 µs beträgt. In der Realität bei nur einem gemessenen Durchlauf aber von dem Wert leicht abweichen wird.

Aufgabe 4.2: Signallaufzeitmessung

In diesem Aufgabenteil wird die Signallaufzeitmessung realisiert. Sie beschreibt die zeitliche Differenz, dass das Signal vom Ausgang des Mikrocomputers über die Hexadezimal-Tastatur bis zum Eingang des Microcomputers benötigt. Für die Durchführung wurde jeweils der Ein- und Ausgang einer bestimmten Taste der Hexadezimaltastatur mit dem Oszilloskop verbunden. Nach betätigen dieser Taste wurde die Verzögerung grafisch erfasst, was deutlich in den folgenden Abbildungen zu sehen ist.

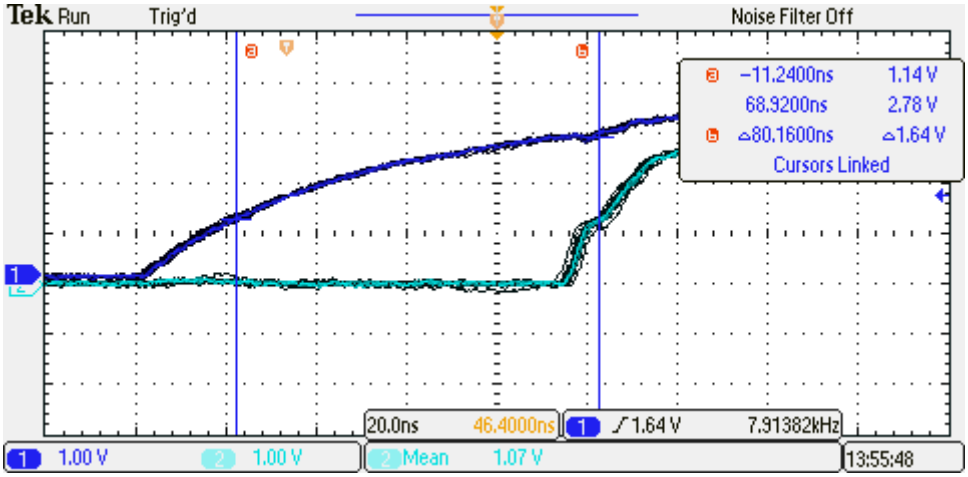

Abb. 4: Laufzeitverzögerung High-Pegel

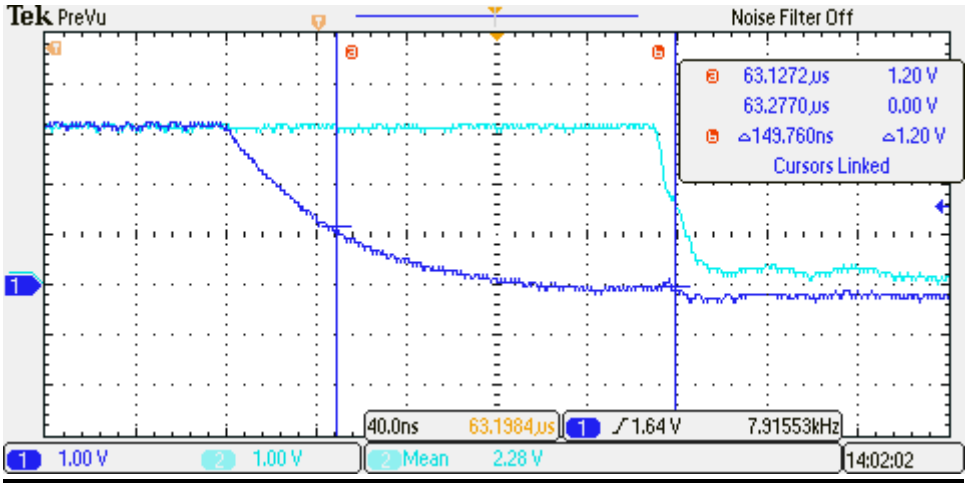

Abb. 5: Laufzeitverzögerung Low-Pegel

High-/Low-Pegel

Um die Laufzeitverzögerung der steigenden oder auch der fallenden Flanke genaustens bestimmen zu können ist ein Blick in das Datenblatt unabdingbar. Gesucht ist hierbei die min. High-Pegel- Spannung, die Grenze ab der Low zu High wechselt, und die max. Low-Pegel-Spannung, der den umgekehrten Fall beschreibt. Da wir mit einer Spannung von 3,3 V arbeiten, erhalten wir für den High-Pegel eine Spannung von 2,145 V und für den Low-Pegel dementsprechend 1,155 V.

Parameter	Parameter Name	Min	Nom	Max	Unit
V_{IH}	Fast GPIO high-level input voltage	0.65 * V_{DD}	-	4	V
I_{IH}	Fast GPIO high-level input current [a]	-	-	300	nA
V_{IL}	Fast GPIO low-level input voltage	0	-	0.35 * V_{DD}	V

Abb. 6: Datenblattwerte [1, p. 1820]

Auswertung

Aus den Abbildungen 4 & 5 kann man entnehmen, dass der Low-High-Wechsel nach 80,16ns am Eingang des Microcomputers registriert wird und somit den Wert von 2,145 V erreicht. Der High-Low-Wechsel beträgt dagegen 149,76ns und nimmt mehr Zeit in Anspruch. Schlussfolgernd kann man sagen, dass die Laufzeitverzögerung problemlos mit der Warteschleife gelöst werden kann, da wir wissen, dass diese um 0,7 µs pro Schleifendurchlauf verzögert.

Aufgabe 4.3: C-Programme zum Einlesen einer Taste der Hexadezimal-Tastatur und Ausgabe auf dem Terminal

In der letzten Teilaufgabe soll ein C-Programm geschrieben werden, dass die Bedienung mit der Hexadezimal-Tastatur ermöglicht. Dabei sollen bestimmte Anforderungen durch das Programm gewährleistet 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 wird diese
kontinuierlich hintereinander dargestellt. Die Periodendauer soll 1 s betragen.

Durchführung

Quellcode Aufgabe 4.3 -> siehe Anhang

Fazit

Aus dem ersten Laborversuch wurde deutlich, dass bei einem Mikrocomputer eine gewisse Signallaufzeit berücksichtigt werden muss, um eine fehlerfreie Auswertung von externen Eingabegeräten zu ermöglichen.
Um gewonnene Kenntnisse zu dokumentieren wird auch vorausgesetzt, dass sich die Kompetenzen zur Erstellung wissenschaftlicher Arbeiten angeeignet werden.
Außerdem konnte festgestellt werden, dass die Funktionsweise der verwendeten Hexadezimaltastatur, gewisse anfälligkeiten hervorruft. Zum Beispiel: wenn zwei Tasten aus der gleichen Spalte gleichzeitig betätigt werden, ist es nicht möglich dies zu erkennen und die entsprechende Fehlermeldung auszugeben.

Literaturverzeichnis

[1, p. 1820] Texas Instruments, “Tiva™ TM4C1294NCPDT Microcontroller” datasheet, June 2014 [Revised Nov. 2015].

[2] Prof. Dr.-Ing Lutz Leutelt. " Mikroprozessortechnik Vorlesungsskript," in MP Vorlesung, HAW Hamburg, Hamburg, 2019.

[3] Prof. Dr. Thomas Lehmann, " T1_Nr_5-Hex-Tastatur.pdf ", HAW Hamburg, Hamburg, July 2016.

Anhänge

Quellcode Aufgabe3.c siehe Anhang

```c
#include "tm4c1294ncpdt.h"
#include "stdio.h"

void wait(void) {
    int tmp;
    for (tmp = 0; tmp < 10; tmp++);
}

const unsigned char tasten[16] = {0xBE, 0x77, 0xB7, 0xD7, 0x7B, 0xBB, 0xDB,
                    0x7D, 0xBD, 0xDD, 0x7E, 0xDE, 0xEE, 0xED, 0xEB, 0xE7};   // Tastenkombinationen von Taste 0 bis F

int main(void) {                                        // Port  Clock Gating Control
    SYSCTL_RCGCGPIO_R |= 0x800;                         // Port M clock ini
    while ((SYSCTL_PRGPIO_R & 0x00000800) == 0);        // Port M ready ?

    // Set direction
    GPIO_PORTM_DIR_R = 0xF0;      // Pin 4-7 Ausgang ,0-3  Eingang

    // Digital enable
    GPIO_PORTM_DEN_R = 0xFF;      // Pin 0-7 aktivieren


    while (1) {
        unsigned char eingangMuster[4] = {0x70, 0xB0, 0xD0, 0xE0}; // Eingangsmuster in ein Array von Größe 4 dargestellt
        int i;
        for (i = 0; i < 4; i++) {                       // Eingangsmatrix wird generiert
            GPIO_PORTM_DATA_R = eingangMuster[i];
            wait();
            int k;
            for (k = 0; k < 16; k++) {
                if (GPIO_PORTM_DATA_R == tasten[k]) {           // Es wird überprüft ob der PORTM-Dataregister den Tasten Bitmuster 0-F entspricht
                    int tmp;
                    if (k <= 9) {                               // Falls die Tasten 0-9 gedrückt werden
                        printf("%d\n", k);                      // Taste 0-9 werden werden geprintet
                    }
                    if (k > 9 && k < 16) {                      // Die Tasten A-F werden im Case überprüft und Geprintet
                        switch (k) {
                            case 10:
                                printf("A\n");
                                break;
                            case 11:
                                printf("B\n");
                                break;
                            case 12:
                                printf("C\n");
                                break;
                            case 13:
                                printf("D\n");
                                break;
                            case 14:
                                printf("E\n");
                                break;
                            case 15:
                                printf("F\n");
                                break;
                        }
                    }
                    if ((GPIO_PORTM_DATA_R != tasten[k])) {             //
                        printf("error\n");
                    }

                    for (tmp = 0; tmp < 1000000; tmp++)                 // warten ungefähr für eine sekunde
                    {
                        if ((GPIO_PORTM_DATA_R & 0x0F) == 0x0F) {       // wenn keine Ausgang-> Schleife Beenden
                            break;
                        }
                    };
                }

            }

        }
    }
}
```