MP 1
Praktikum 1

2021

Erste Schritte: TM4C1294NCPDT

3. November 2021

Florian Tietjen 2519584
Emily Antosch 2519935

Inhaltsverzeichnis

Abbildungsverzeichnis

Listings

1 Einführung in den TM4C1294NCPDT

Im ersten Praktikum wollen wir uns mit dem Evaluation Board TM4C1294NCPDT vertraut machen, mit dem wir uns auch in den nächsten Praktika beschäftigen wollen. Dabei richten wir über den Projektbrowser und den Projekteigenschaften alle wichtigen Eigenschaften ein, testen erste Konsolenausgaben und lassen LEDs in bestimmten Blinkmustern aufleuchten.

2 Print to Console

Aufgabe 2.0

In der ersten Aufgabe wollen wir die Ausgabe in der Konsole des Code Composer Studios betrachten.

Für das Bearbeiten dieser Aufgabe nutzen wir den folgenden Code:

```c
/*
Mikroprozessortechnik - Praktikum 1 Aufgabe 1
Print to Console
Autoren: Emily Antosch und Florian Tietjen
Beschreibung: Dieses Programm gibt eine geordnete Ausgabe eines Zaehlers, eines
    Namens und einer Matrikelnummer in der Konsole aus.
*/


#include "inc/tm4c1294ncpdt.h"
#include <stdio.h>
#include <stdint.h>

int main(void)
{
    int i, cnt; // Initalisierung der Varibalen

    cnt = 0; // cnt beginnt bei 0

    // Dauerschleife fuer den Zaehler
     while (1)
     {
         printf("%05d\n", cnt); // Ausgabe des Zaehlers
         printf("Name: Emily Antosch \n"); // Ausgabe des Namens
         printf("Matr.No: 2519935 \n"); // Ausgabe der Matrikelnummer
         cnt++; // Zaehler erhoehen um 1
         for (i = 0; i < 500000; i++) // Warteschleife
             ;
     }
}
```

Listing 1: printToConsole.c

Dabei schließen wir das Evaluation Board an und führen das Programm dann im Debugmodus aus. Auf der Konsole erscheint eine schnell wachsende Liste von dem eingegebnen Namen, der Matrikelnummer und der steigenden Zählervariable cnt.

2.1 Kurzfazit

Das zeigt uns zum Einen, wie wir die printf-Funktion in Code Composer Studio zu nutzen haben, um in größeren Programmen, Fehler finden zu können, zum Anderen haben wir das Setup des Boards richtig gemacht und das Kompilieren und Ausführen von Programmen funktioniert.

3 Blinking LED

Aufgabe 3.0

In der zweiten Aufgabe wollen wir üben, mit der Hardware auf dem Board zu arbeiten und bestimmte Register anzusprechen.

Dazu verwenden wir den folgenden Code:

```
/*
Mikroprozessortechnik - Praktikum 1 Aufgabe 2
Blinking LED
Autoren: Emily Antosch und Florian Tietjen
Beschreibung: Dieses Programm greift auf die Ports N, F zu, um ein
Blinkmuster zu realisieren
*/


#include <stdint.h>
#include "inc/tm4c1294ncpdt.h"

int main(int argc, char const *argv[])
{
    int i = 0;                      // => Deklaration und Initalisierung der Warte-
    Variable i
    SYSCTL_RCGCGPIO_R = 0x00001020; // => Aktivieren der Ports N und F
    i++;                           // => Erhoehen der Variable i um 1, warten auf
    Clock
    GPIO_PORTN_DEN_R = 0x03;      // => Enablen der Pins 0 und 1
    GPIO_PORTN_DIR_R = 0x03;      // => Richtung der Pins N 0 und 1 auf Ausgang
    GPIO_PORTF_AHB_DEN_R = 0x11; // => Enablen der Pins 0 und 4
    GPIO_PORTF_AHB_DIR_R = 0x11; // => Richtung der Pins F 0 und 4 auf Ausgang
    // => Dauerschleife zum Ausfuehren des Blinkmusters
    while (1)
    {
        GPIO_PORTN_DATA_R = 0x02;    // => Setzen der LED 1 auf ON
        for (i = 0; i < 500000; i++) // => Warten
            ;
        GPIO_PORTN_DATA_R = 0x01;    // => Setzen der LED 2 auf ON
        for (i = 0; i < 500000; i++) // => Warten
            ;
        GPIO_PORTN_DATA_R = 0x00;     // => Reset beider Outputs auf OFF
        GPIO_PORTF_AHB_DATA_R = 0x10; // => Setzen der LED 3 auf ON
        for (i = 0; i < 500000; i++)  // => Warten
            ;
        GPIO_PORTF_AHB_DATA_R = 0x01; // => Setzen der LED 4 auf ON
        for (i = 0; i < 500000; i++)  // => Warten
            ;
        GPIO_PORTF_AHB_DATA_R = 0x00; // Reset beider Outputs auf OFF
    }
}
```

Listing 2: Code für das Blinkmuster

Dabei verwenden wir die beiden Register F und N, in welchen die Pins für die LEDs auf dem Board sind. Wir setzen alle Pins auf Ausgang und Enablen diese. Im Nachgang setzen wir die jeweiligen Pins im entsprechenden Blinkmuster auf OFF oder ON mit einer Warteschleife dazwischen.

3.1 Ausgabe des Programms

Wir erkennen einen durchgängigen Verlauf der LEDs, bei dem die LEDs der Reihe nach aufleuchten und dann wieder ausgehen, sodass immer nur eine LED zur Zeit eingeschaltet ist. Es entsteht der Effekt eines "laufenden"Lichts, welches sich immer von links nach rechts bewegt.

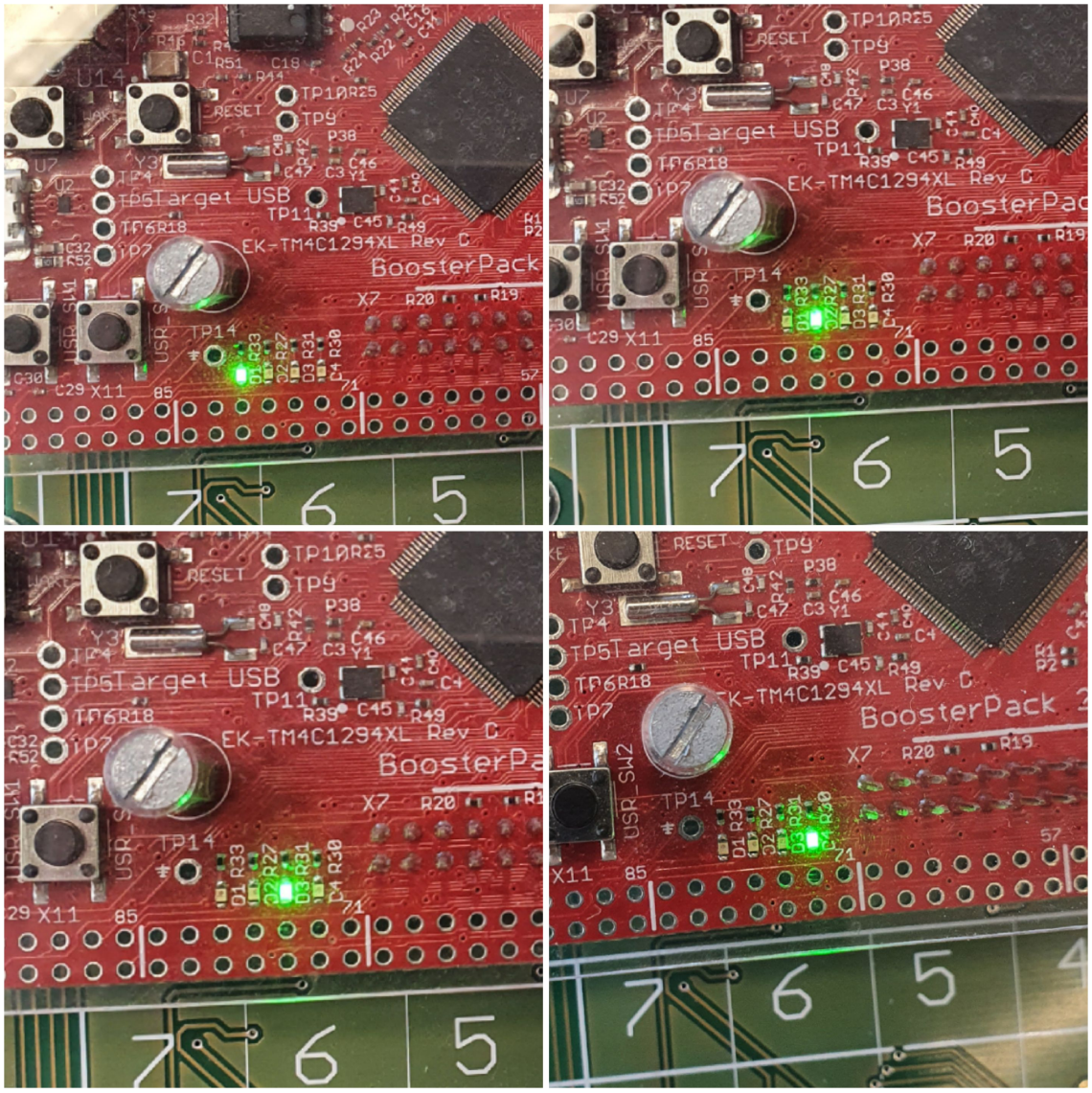

Abbildung 1: LED-Blinkmuster auf dem Evaluation Board

3.2 Gleichzeitige Schaltung beider Eingänge in Port N

Um beide LEDs an Port N gleichzeitig zu schalten müssen wir lediglich folgende Programmzeilen verwenden:

```
1  GPIO_PORTN_DATA_R = 0x03;
2  GPIO_PORTN_DATA_R = 0x00;
```

Listing 3: Port N Pins gleichzeitig schalten

Dabei ist die erste Zeile das Schalten beider Pins auf ON, die zweite Zeile schaltet beide auf OFF. Dabei gilt $0x03_{16} = 0011_2$ und $0x00_{16} = 0000_2$, wobei eine 1 auf einem bestimmten Port ein ON-Signal und eine 0 ein OFF-Signal darstellt.

3.3 Kurzfazit

An dieser Aufgabe konnten wir erkennen, wie wir auf einfache Weise die Pins des Boards über die Ports ansprechen können. Für zukünftige Aufgaben haben wir die Chance, diese LEDs einzubauen.

4 Blinking LED mit Tasten

Aufgabe 4.0

Im letzten Aufgabenblock wollen wir nun ein Blinkmuster mit dem integrieren einer

```c
/*
Mikroprozessortechnik - Praktikum 1 Aufgabe 3
Blinking LED mit Taster
Autoren: Emily Antosch und Florian Tietjen
Beschreibung: Dieses Programm greift auf die Ports N, F und J zu, um ein
Blinkmuster mit Tastern zu realisieren
*/

#include <stdint.h>
#include <stdio.h>
#include "inc/tm4c1294ncpdt.h"

int main(int argc, char const *argv[])
{
    int i = 0;                      // => Deklaration und Initalisierung der Warte-
    Variable i
    unsigned char state;            // => Deklaration der State-Variable zum
    Abfragen des Schalterstandes
    SYSCTL_RCGCGPIO_R = 0x00001120; // => Aktivieren der Ports N, J und F
    while ((SYSCTL_PRGPIO_R & 0x00001120) == 0) // => Warten bis Clocksignal anliegt
        ;
    i++;                           // => Erhoehen von i um 1, warten auf Clock
    GPIO_PORTN_DEN_R = 0x03;       // => Enablen der Pins N 0 und 1
    GPIO_PORTN_DIR_R = 0x03;       // => Richtung der Pins N 0 und 1 auf Ausgang
    GPIO_PORTJ_AHB_DEN_R = 0x03; // => Enablen der Pins J 0 und 1
    GPIO_PORTJ_AHB_PUR_R = 0x03; // => Weak-Pull-Up aktivieren fuer die Pins J 0 und
     1
    GPIO_PORTJ_AHB_DIR_R = 0x00; // => Richtung der Pins J 0 und 1 auf Eingang
    GPIO_PORTF_AHB_DEN_R = 0x11; // => Enablen der Pins F 0 und 4
    GPIO_PORTF_AHB_DIR_R = 0x11; // => Richtung der Pins F 0 und 4 auf Ausgang
    // => Dauerschleife zum Ausgeben der Blinkmuster
    while (1)
    {
        state = GPIO_PORTJ_AHB_DATA_R; // Abfragen des Schalterstandes

        if (state == 0x02) // Wenn Schalter 1 gedrueckt
        {
            GPIO_PORTN_DATA_R = 0x02;    // LED 1 leuchtet
            for (i = 0; i < 500000; i++) // Warten
                ;
            GPIO_PORTN_DATA_R = 0x01;    // LED 2 leuchtet
            for (i = 0; i < 500000; i++) // Warten
                ;

        }
        else if (state == 0x01) //Wenn Schalter 2 gedrueckt
        {
            GPIO_PORTF_AHB_DATA_R = 0x10; // LED 3 leuchtet
            for (i = 0; i < 500000; i++)  // Warten
                ;
            GPIO_PORTF_AHB_DATA_R = 0x01; // LED 4 leuchtet
            for (i = 0; i < 500000; i++)  // Warten
                ;

        }
        else if (state == 0x00) // Wenn beide Schalter gedrueckt
        {
            GPIO_PORTN_DATA_R = 0x02;     // LED 1 leuchtet
            GPIO_PORTF_AHB_DATA_R = 0x10; // LED 3 leuchtet
            for (i = 0; i < 500000; i++)  // Warten
```

```
58                  ;
59              GPIO_PORTN_DATA_R = 0x01;      // LED 2 leuchtet
60              GPIO_PORTF_AHB_DATA_R = 0x01; // LED 4 leuchtet
61              for (i = 0; i < 500000; i++)  // Warten
62                  ;
63
64          }
65          GPIO_PORTN_DATA_R = 0x00;      // Reset der LEDs 1 und 2
66          GPIO_PORTF_AHB_DATA_R = 0x00; // Reset der LEDs 3 und 4
67      }
68  }
```

Listing 4: Code für das Blinkmuster mit Taster

Der Programmcode lässt uns nun die folgenden Eigenschaften realisieren:

- Beim Drücken des Taster 1 blinken die LEDs 1 und 2 abwechselnd
- Beim Drücken des Taster 2 blinken die LEDs 3 und 4 abwechselnd
- Beim Drücken beider Taster blinken jeweils LEDs 1 und 2 und LEDs 3 und 4 paarweise abwechselnd

4.1 Weak-Pull-Up-Widerstand

737ms Wichtig für die korrekte Funktion der Taster ist ein Weak-Pull-Up-Widerstand. Dieser zieht im Fall, dass der Taster nicht gedrückt ist, den Pin auf ein ON-Signal. Ohne Pull-Up-Widerstand würde der Eingang bei nicht gedrücktem Taster in der Luft hängen. Er wäre also weder klar definiert logisch 1 oder 0 und wäre zudem stark beeinflussbar von äußeren Einflüssen (Induktivitäten). Ist der Taster jedoch gedrückt, so ist der PIN mit GND verbunden und ist in jedem Fall logisch 0.

4.2 Oszilloskop: Messung der Taktfrequenz

Um einen Einblick in die Verarbeitung der verwendeten Befehle zu bekommen, wollen wir nun uns einmal anschauen, mit welcher Periodendauer das paarweise abwechselnde Aufblicken der LED erfolgt.

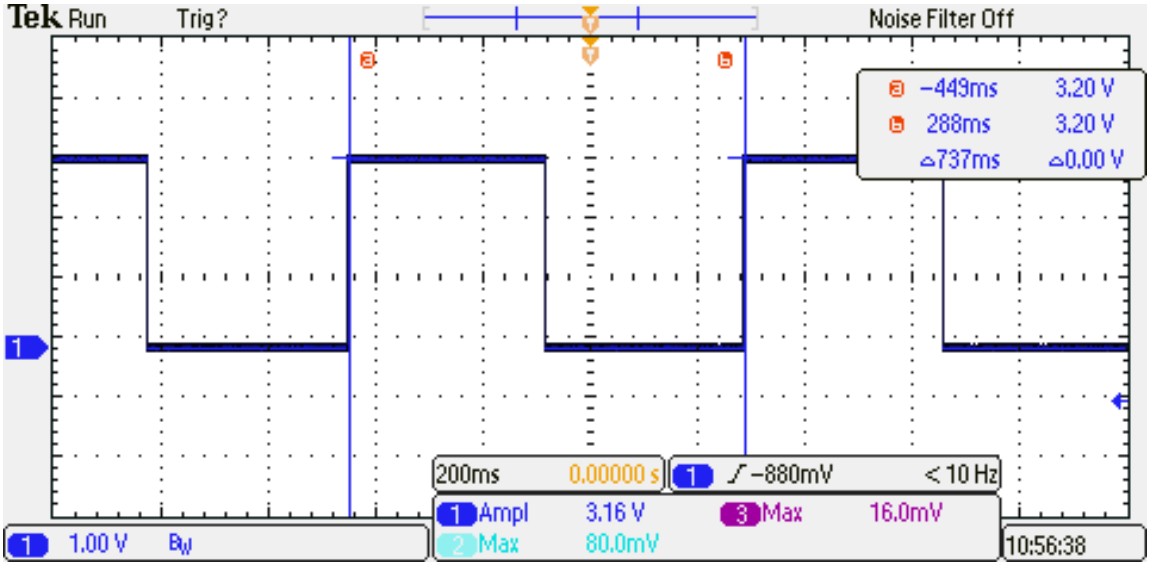

Abbildung 2: Verlauf der Spannung U_F an einer der LEDs mit Verzögerung $i < 500000$

Im oberen Beispiel haben wir den Verlauf der Spannung an einer LEDs beim Betätigen des entsprechenden Tasters aufgezeichnet. Die Verzögerung, die wir über die for-Schleife eingebaut haben, beträgt wie im Code-Beispiel $i < 500000$.

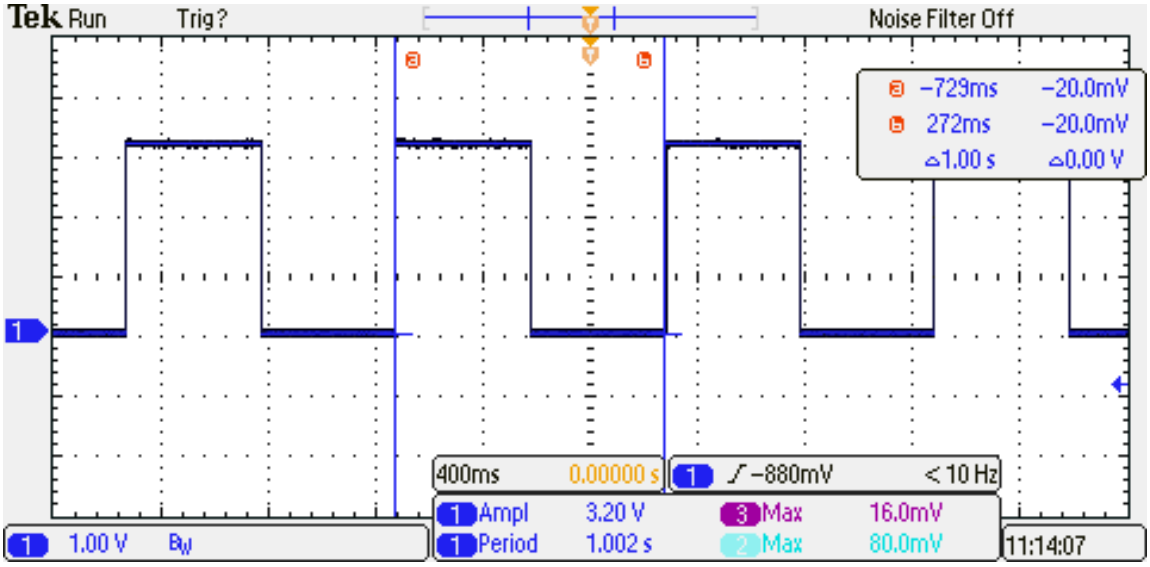

Abbildung 3: Verlauf der Spannung U_F an einer der LEDs mit Verzögerung $i < 678426$

Im nächsten Beispiel haben wir die Verzögerung auf $\Delta t = 1s$ angepasst, indem wir $f_{1s} = \frac{f_{737ms}}{737ms} = \frac{500000}{737ms} \approx 678426Hz$ gerechnet haben. Wir erkennen eine kleine Abweichung der Messung von $2ms$.

4.3 Kurzfazit

Wir haben nun unser erstes eigenes Programm auf der Grundlage von der zweiten Aufgabe erstellen können. Auch den Umgang mit dem Oszilloskop konnte erneut geübt und vertieft werden, sodass wir für die nächsten Praktika besser vorbereitet sind. Diese kann aus sowohl Ungenauigkeiten der Messstrecken, sowie Fehler beim Ablesen der ursprünglichen Werte entstanden sein.

5 Konklusion

Uns ist es gelungen, die Bedienung von CCS zu lernen und ein erstes Projekt richtig aufzusetzen. Die erste Arbeit mit Ports und Registern war erfolgreich und wir haben es geschafft, Zustände auf Ausgänge zu schreiben, sowie Taster als Eingangssignale einzulesen. Das erste eigene Programm haben wir ebenfalls zu Laufen bekommen und den erwünschten Effekt als Blinkmuster auf dem Board zu erhalten.