**Protokoll zum Praktikum Mikrocomputertechnik**

Inhalt:

Aufgabe 5 am 13.11.2019

Aufgabe 8 am 04.12.2019

Aufgabe 10 am 11.12.2019

Durchgeführt von:

Max Malinowski Mat.-Nr. \_\_\_\_\_

Thomas Egen Mat.-Nr. 89555

Arbeitsplatz:

Nr.8

Studiengang:

FFI

Dozent:

Prof. Dr. Ulrich Margull

Praktikumsgruppe:

MCTP2 Mittwoch 14:00 – 17:20

# **Aufgabe 5: Tastaturmatrix**

## Aufgabenstellung:

Ziel der Aufgabe war die Programmierung einer Funktion, mithilfe der Zustand der Tasten einer 4x4 Tastaturmatrix ermittelt werden konnte. Hierzu müssen die Ports korrekt konfiguriert sein und die Ansteuerung der entsprechenden Ports muss über die entsprechenden Ansteuerungssignale erfolgen [Aufgabenstellung].

## Auswertung:

Eingang 1. PB4

Eingang 2. PB5

Eingang 3. PB6

Eingang 4. PB7

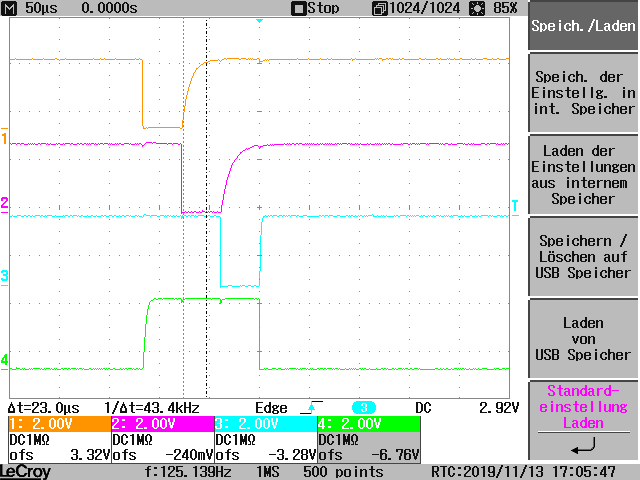


Abbildung 1: Oszillogramm, welches die Auslesesequenz der Tastaturmatrix zeigt.

Auf der Abbildung 1 sind die einzelnen Steuerleitungen der Tastaturmatrix zu sehen. Diese Leitungen werden nacheinander auf eine aktive Null gesetzt. Dies sieht man sehr im Oszillogramm. Nachdem eine Leitung auf eine aktive Null gesetzt wurde kann mithilfe der Messleitung, welche durch einen Pull-Up Wiederstand realisiert ist, überprüft werden, ob eine Taste gedrückt ist. Damit beim Lesen der Messleitung immer ein definierter Pegel herrscht, zieht der Pull-Up Wiederstand die Leitung immer auf eine Eins. Lediglich wen eine Taste gedrückt wird, so liegt eine Null an, was detektiert werden kann.

Am Oszillogramm kann ebenfalls erkannt werden, dass das Hochziehen auf eine Eins ca. 23 Mikrosekunden dauert. Dies muss in der Software berücksichtigt werden, da man die entsprechende Zeit abwarten muss bevor die nächste Zeile auf null gesetzt wird.

Die letzte Zeile bleibt nach einer Lesesequenz auf einer logischen Null, da dies so in der Software implementiert und vorgesehen ist.

# **Aufgabe 8: LIN-Bus**

## Aufgabenstellung:

Ziel der Aufgabe war die Programmierung und Konfiguration eines LIN Slave. Dieser sollte auf drei unterschiedliche Anfragen des Masters entsprechend mit der aktuellen Zeit (Millisekunden seit Programmstart), der aktuell anliegenden Frequenz oder dem aktuellen Zustand der Tasten, gefolgt von der korrekten Checksum antworten [Aufgabenstellung].

## Auswertung:

Eingang 1. LIN Bus

Eingang 2. RxD Pin

Eingang 3. TxD Pin

Eingang 1. PB4

Eingang 2. PB5

Eingang 3. PB6

Eingang 1. PB4

Eingang 2. PB5

Eingang 3. PB6

Eingang 1. PB4

Eingang 2. PB5

Eingang 3. PB6

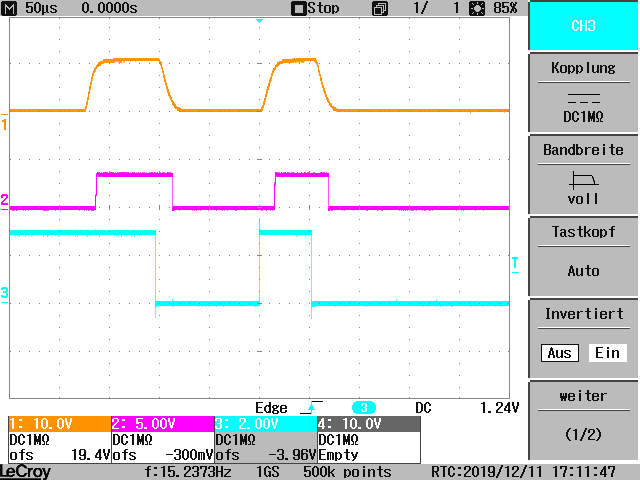


Abbildung 2: Oszillogramm, welches das Senden einer 0-1-0 Folge zeigt.

Abbildung 2 zeigt eine 0-1-0 Folge, jeweils an den verschiedenen Pins. Am TxD Pin kann erkannt werden, dass zuerst eine 0 gesendet wird, gefolgt von einer 1. Das senden der 0 wird durch das Verbinden des LIN Buses mit Ground realisiert. Die 1 entsteht durch die Grundspannung. Diese wird durch einen Pull-Up Wiederstand realisiert. Als Folge dieser Realisierung muss sich die Spannung erst aufbauen, was dazu führt, dass die Flanken flacher sind. Dies hat ebenfalls zur Folge, dass die EMV Abstrahlung dieser Flanken geringer ist und es zu weniger Störungen kommt. Der RxD Pin erkennt die Übertragung erst etwas zeitverzögert, da das Signal am Bus erst ein gewisses Potential erreichen muss um als High-Pegel erkannt zu werden.

Abbildung 3 zeigt eine gesamte LIN Nachricht an den verschiedenen Pins. Was auffällt ist, dass der der TxD Pin eine 1 sendet, diese aber scheinbar den LIN Bus und den RxD Pin, welcher den LIN Bus liest, nicht zu beeinflussen schein. Dies liegt daran, dass eine 1 rezessiv ist und von einer 0 überschrieben wird. Man erkennt deutlich, dass die LIN Nachricht zuerst mit einer Sequenz von 13x0 und einer Rahmenverletzung beginnt. Hierbei handelt es sich um den LIN Break, welcher vom Master gesendet wird, um den Slaves zu signalisieren, dass eine neue Nachricht übertragen wird. Da keine 1 eine 0 überschreiben kann, kann dieses Signal nicht gestört werden. Anschließend wird vom Master eine alternierende Folge von 0 und 1 gesendet. Das ist das sogenannte Sync-Field. Dieses erlaubt dem Takt des Slaves an den Takt des Masters anzupassen, damit die Nachricht fehlerfrei übertragen werden kann. Nachdem dies geschehen ist, sendet der Master den LIN-Identifier. Dieser identifiziert einen bestimmten Slave im LIN-Netzwerk. Anhand des Identifiers kann jeder Slave überprüfen, ob die Nachricht für Ihn bestimmt ist und er antworten soll oder nicht. In diesem Fall handelt es sich um den Identifier des Slaves 0x18 und er antwortet mit 2 Byte. Der aktuellen Zeit, gefolgt von einer 1 Byte Checksum. Anhand dieser kann der Master überprüfen, ob die Nachricht korrekt übertragen wurde. Abbildung 4 zeigt die analysierte LIN-Nachricht. Jedes Byte ist zudem von einem Start und einem Stopp Bit eingerahmt, da die Übertragung beim LIN Bus byteweise erfolgt.

Eingang 1. LIN Bus

Eingang 2. RxD Pin

Eingang 3. TxD Pin

Eingang 1. PB4

Eingang 2. PB5

Eingang 3. PB6

Eingang 1. PB4

Eingang 2. PB5

Eingang 3. PB6

Eingang 1. PB4

Eingang 2. PB5

Eingang 3. PB6

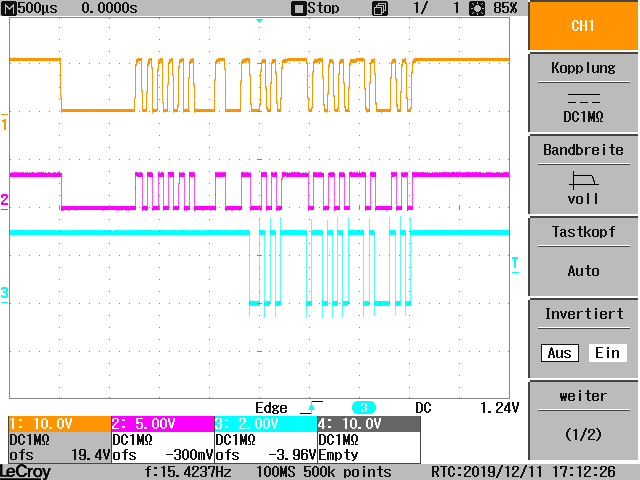


Abbildung 3: Oszillogramm, welches das Versednen einer gesamten Lin-Nachricht auf dem Bus zeigt.

Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

Response

Header

Checksum

Data

STOP

STOP

Data

LIN-Identifier

STOP

STOP

STOP

START

START

START

START

START

LIN-Break

Synch-Field

Abbildung 4: Analyse der LIN-Nachricht

# **Aufgabe 10: Drosselklappe**

## Aufgabenstellung:

Ziel der Aufgabe war die Ansteuerung einer Drosselklappe. Hierzu sollte der analoge Wert eines Potentiometers in einen digitalen Prozentwert gewandelt, und anschließend für die Erzeugung eines PWM-Signales genutzt werden [Aufgabenstellung].

## Auswertung:

Eingang 1. PB0

Eingang 2. PB1

Eingang 3. OUT1

Eingang 4. OUT2

Eingang 1. PB4

Eingang 2. PB5

Eingang 3. PB6

Eingang 4. PB7

Eingang 1. PB4

Eingang 2. PB5

Eingang 3. PB6

Eingang 4. PB7

Eingang 1. PB4

Eingang 2. PB5

Eingang 3. PB6

Eingang 4. PB7

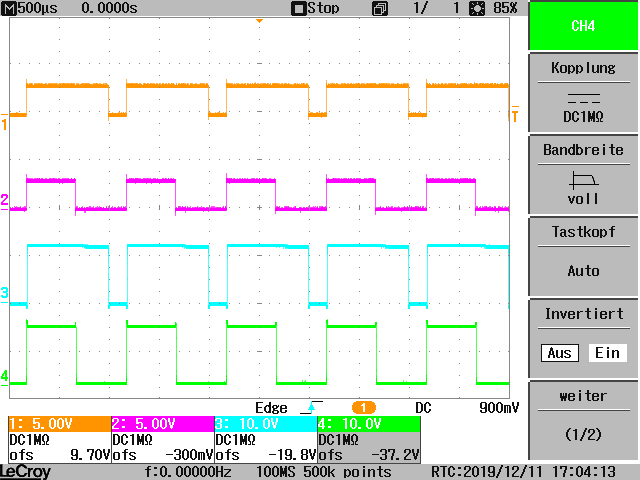


Abbildung 5: Oszillogramm, welches zwei generierte PWM-Signale zeigt. Einmal nicht verstärkt und einmal verstärkt.

Die Abbildung 5 zeigt zwei PWM-Signale an den Eingängen 1 und 2 mit welchen die Drosselklappe gesteuert wird. Das Signal an PB0 sorgt hierbei für eine Öffnung der Drosselklappe (Drehung im Uhrzeigersinn) und das Signal an PB1 sorgt für eine Schließung der Drosselklappe (Drehung gegen den Uhrzeigersinn). Sind beide Signale aktiv, so heben sie sich gegenseitig auf. Lediglich in der Zeit in der nur eins der beiden Signale aktiv ist wird eine Drehung der Drosselklappe in die ein oder andere Richtung bewirkt. An den Eingängen 3 und 4 sind die Signale OUT1 und OUT2 der H-Brücke zu sehen. Hierbei handelt es sich um dieselben PWM Signale wie die des Mikrocontrollers, welche jedoch verstärkt wurden. Da die Drosselklappe mit 12V arbeitet und nicht wie der Mikrocontroller mit 3V, muss das PWM Signal des Controllers verstärkt werden.

# **Abbildungen:**

[Abbildung 1: Oszillogramm, welches die Auslesesequenz der Tastaturmatrix zeigt. 2](file:////Users/maxmacbook/mct/microcontroller/doc/Report%20Microcontroller%20Class.docx#_Toc27469946)

[Abbildung 2: Oszillogramm, welches das Senden einer 0-1-0 Folge zeigt. 3](file:////Users/maxmacbook/mct/microcontroller/doc/Report%20Microcontroller%20Class.docx#_Toc27469947)

[Abbildung 3: Oszillogramm, welches das Versenden einer gesamten Lin-Nachricht auf dem Bus zeigt. 4](file:////Users/maxmacbook/mct/microcontroller/doc/Report%20Microcontroller%20Class.docx#_Toc27469948)

[Abbildung 4: Analyse der LIN-Nachricht 4](#_Toc27469949)

[Abbildung 5: Oszillogramm, welches zwei generierte PWM-Signale zeigt. Einmal nicht verstärkt und einmal verstärkt. 5](file:////Users/maxmacbook/mct/microcontroller/doc/Report%20Microcontroller%20Class.docx#_Toc27469950)

# **Literatur:**

[Aufgabenstellung] -

Prof. L. Gaul, Prof. U. Margull, Prof. G. Passig „*Aufgaben für den Studiengang FFI*“, 2019