**H ö h e r e T e c h n i s c h e B u n d e s l e h r a n s t a l t**

**S a l z b u r g**

**Abteilung für Elektronik**

**Übungen im**

**Laboratorium für Elektronik**

**Protokoll**

**für die Übung SreS 02**

**Gegenstand der Übung**

|  |
| --- |
| **Logic Analyzer** |
|  |

|  |  |
| --- | --- |
| **Name:** | **Markus Zundl** |
| **Jahrgang:** | **4AHEL** |
| **Gruppe Nr.:** | **C03** |
| **Übung am:** | **23.10.2019** |

|  |  |
| --- | --- |
| **Anwesende:** | Markus Zundl, Robert Seethaler |

***Inhaltsverzeichnis***

[1. Messanweisung 3](#_Toc22996314)

[2. Einleitung 4](#_Toc22996315)

[3. Inventarliste 6](#_Toc22996316)

[4. Übungsdurchführung 7](#_Toc22996317)

[4.1 Timer CTC 7](#_Toc22996318)

[4.1.1. Training an der Messumgebung 9](#_Toc22996319)

[4.1.2 Triggerung von Signalen 10](#_Toc22996320)

[4.1.3 Auflösung und Speichertiefe 13](#_Toc22996321)

[4.2 Protokollierung eines I2C Datentransfers 15](#_Toc22996322)

[1. Zusammenfassung 18](#_Toc22996323)

# Messanweisung

# Einleitung

Der Inhalt dieser Laborübung war der Logic Analyzer. Für das HTL CRUMTL wurde ein Programm geschrieben, mit dem der interne 8-Bit Timer im CTC-Mode aktiviert wurde. Die Register für den Timer wurden so berechnet, dass das LSB des Ausgangsignal mit einer Frequenz von 1kHz toggelte und am Port A des CRUMTL Werte von 0 bis 127 ausgegeben wurden. Die Ausgangssignale wurden mit dem Logic Analyzer aufgezeichnet. Dabei wurden die verschiedenen Funktionen der Software „Logic 1.2.18“ durchgetestet. Danach wurde eine I2C Übertragung zwischen dem CRUMTL und dem Temperatursensor LM75 aufgezeichnet und das gemessene Signal in der Messsoftware ausgewertet und so die gemessene Temperatur bestimmt.

**Logic Analyzer:**

Mit dem Logic Analyzer ist es im Gegensatz zu einem Oszilloskop möglich, digitale Signale zu messen und aufzuzeichnen. Die aufgezeichneten Signale können auch durch den Logic Analyzer analysiert werden, wobei hier die Möglichkeit besteht, das Protokoll von Busverbindungen gleich zu decodieren und die einzelnen Werte, wie das Acknowledge - Bit bei I2C, ausdrücklich zu kennzeichnen. Logic Analyzer haben im Gegensatz zum Oszilloskop deutlich mehr Eingänge. Sie werden unter anderem eingesetzt, um die Kommunikation auf Bussystemen zu überprüfen, für das Debugging von Firmware, bei Reverse Engineering und in der Chip-Entwicklung. Die Logic Analyzer werden als selbständige Geräte gebaut, in Kombination mit einem Oszilloskop als Mixed-Signal Oszilloskop und als PC-basierte Versionen für den Einsatz außerhalb von Industrie und spezialisierten Laboratorien. Die Kenndaten eines Logic Analyzers sind die Abtastrate in MHz, die bei dem verwendeten LA max. 24MHz beträgt, die Speichergröße, welche bei PC-basierten durch die Anbindung an den PC sehr groß ist und die Anzahl der Eingänge. Dazu kommen noch die verschiedenen Trigger-Arten (rising-edge, falling-edge…) und die verschiedenen Protokolle, die unterstützt und somit decodiert werden können. Der LA hat die Möglichkeit zwei Analyse-Arten durchzuführen, wobei bei der Zeit-Analyse die Eingänge innerhalb von festen Zeitabständen gemessen werden und wo eine hohe Abtastrate erreicht wird, sowie eine Zustands-Analyse, wo die Leitungen wegen externer Ereignisse gelesen werden und die Abtastrate um mehr als die Hälfte reduziert ist.

**Aufbau Logic Analyzer:**

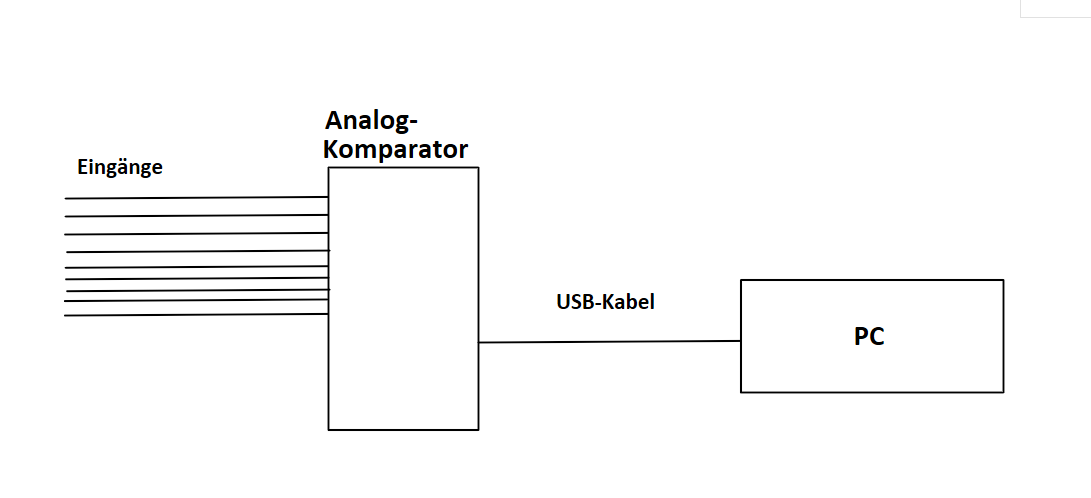


Abb. 1: Aufbau eines einfachen PC-basierten Logic Analyzers, wie er in der Übung verwendet wurde

Ein Logic-Analyzer besteht aus einem Analog- und einem Digitalteil. Der Analogteil hat die Aufgabe, das Eingangssignal auf den verwendeten Pegel des LAs anzupassen. Der Digitalteil besteht aus einem Komparator, einem Sampler und einem RAM-Speicher, der bei PC-basierten von dem PC zu Verfügung gestellt wird. Der Komparator erfasst die digitalen Signale und bestimmt auch die Messdauer. Der Sampler erfasst die Signale ebenfalls mittels Flankentriggerung und interpretiert sie als 0 oder 1 um sie dann im RAM-Speicher für die spätere Analyse und Darstellung abzulegen.

**Timer:**

Ein Timer ist ein spezielles Register im Mikrocontroller, der fortlaufend durch den CPU-Takt inkrementiert wird. Erreicht das Zählregister (TCNTn) im Timer dem maximalen Zählstand, so tritt beim nächsten Inkrement ein Überlauf auf und der Zählstand wird wieder auf 0 zurückgesetzt. Ist das entsprechende Bit im Timer Interrupt Mask Register (TIMSKn) und auch das Global Interrupt Flag im Status Register gesetzt, so wird beim Überlaufen des Timers ein Interrupt ausgelöst. Damit der Timer nicht mit jedem CPU-Taktsignal inkrementiert wird, kann der CPU-Takt durch den Vorteiler um die Faktoren 1/8/64/256/1024 geteilt werden, wenn die entsprechenden Clock-Select Bits (CSxx) im Timer Counter Controll Register B(TCCRnB) gesetzt sind. Allgemein kann bei dem verwendeten Mikrocontroller ATMega664P der Timer im Normal Mode, im CTC Mode, Fast PWM- und im Phase Correct PWM Mode verwendet werden. Da in der Übung nur der CTC Mode für den Timer 0 verwendet wurde, folgt für diesen noch eine ausführliche Beschreibung mit dem verwendeten Code.

**Clear Timer on Compare Match (CTC):**

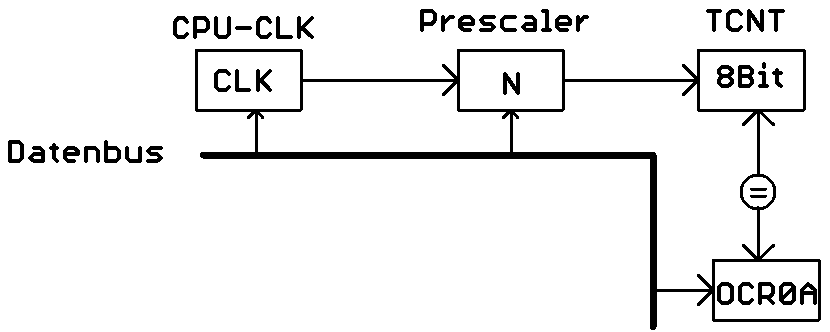


Abb. 2: Schematischer Aufbau eines Timers im CTC-Mode

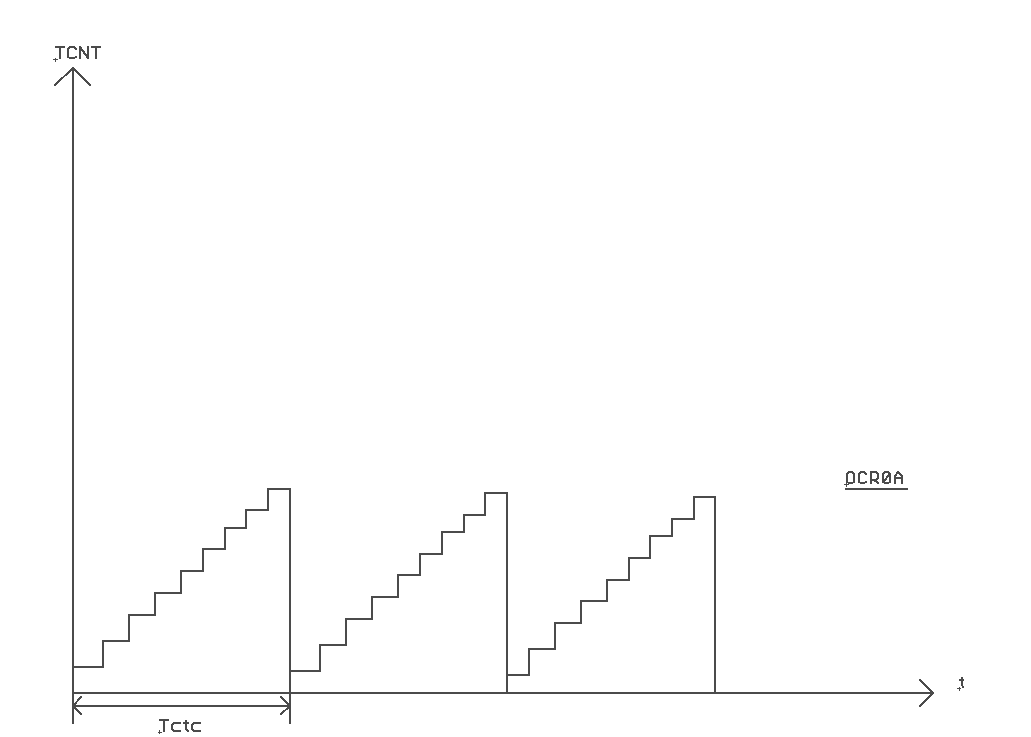


Abb. 3: Timing Diagramm CTC-Mode

Während im Normal Mode das TCNTn -Register (Timer Counter Register) bis zum maximal möglichen Wert (bei dem verwendeten 8Bit Timer 255) inkrementiert und bei Überlauf ein Interrupt ausgelöst wird, kann man im CTC-Mode einen Wert in das Output Compare Register A (OCR0A) speichern. Das TCNT-Register wird in diesem Modus laufend erhöht und mit dem Wert im OCR0A Register verglichen. Erreicht nun das TCNT-Register nun den Wert, der im OCR0A gespeichert ist, wird ein Interrupt ausgelöst (der Compare-Match-Interrupt) und das TCNT wird zurückgesetzt und von vorne laufend mit dem geteilten CPU-Takt inkrementiert. So hat ist es möglich, die Frequenz des beim Überlaufen ausgelösten Interrupts zu kontrollieren. Die Werte, die man in das OCR0A-Register speichern darf, gehen von 0 bis 0xFF und diese sollten möglichst groß sein, was durch die Verwendung eines möglichst kleinen Vorteilfaktors N erreicht wird.

**Berechnung des Vorteilfaktors und des Vergleichswertes:**

Um den geeigneten Vorteilfaktor und den Wert für das Vergleichsregister (OCRnA) zu berechnen, braucht man zuerst die Formel für die Periodendauer eines Zählzyklus. Diese lautet folgendermaßen:

Dabei ist

Eingesetzt in die Formel für ergibt das dann:

In diese Formel wird dann der gewünschte Wert für und der Wert für eingesetzt und der Vorteilfaktor N solange erhöht, bis die auf OCR0A umgeformte Formel einen Wert kleiner gleich 0xFF (im Dezimalsystem 255) ergibt.

# Inventarliste

|  |  |  |
| --- | --- | --- |
| **Stück** | **Gerätebezeichnung** | **Inventarnummer** |
| 1 | Laptop, Windows 10, 64-Bit |  |
| 1 | Saleae Logic Software; Version 1.2.18 |  |
| 1 | Atmel Studio; Version 7.0 |  |
| 1 | Saleae Logic Analyzer; 24MHz, 8CH |  |
| 1 | chip45boot2 GUI |  |
| 1 | Steckbrett | - |
| 1 | Spannungsversorgung 5V |  |
| 1 | HTL CRUMTL | - |
| 2 | Widerstand 10kΩ |  |
| 1 | Temperatur Sensor LM75 |  |

# Übungsdurchführung

Das Ziel dieser Übung war einerseits der Aufbau eines 8Bit Zählers mit dem CTC-Mode des Timer0 des CRUMTL und die I2C Ansteuerung des Temperatursensors LM75. Bei dem ersten Teil der Übung sollten verschiedene Messungen durchgeführt werden und dabei die Funktionen des Logic Analyzers und der Messsoftware kennengelernt werden und bei dem zweiten Teil sollte die I2C Kommunikation zwischen dem CRUMTL und dem LM75 aufgezeichnet werden und dabei die Temperatur aus dem I2C Protokoll abgelesen werden.

## 4.1 Timer CTC

Der erste Teil der Übung bestand darin, einen 8Bit Zähler mit dem CTC-Mode des Timer0 aufzubauen. Dieser Zähler sollte von 0 bis 127 zählen und den aktuellen Zählstand am Port A des CRUMTL ausgeben. Die Vorgabe war, dass das LSB des Zählers mit einer Frequenz von 1kHz toggelt. Mit dieser Vorgabe sollte ein Programm für den CTC-Mode in Atmel Studio geschrieben werden, wobei die Werte für den Vorteiler und für das Compare Register vorab noch berechnet werden sollten.

Die Berechnung lautet folgendermaßen:

**Geg:**

Da jedoch das LSB nur bei Überlauf des Timers den Zustand wechselt, muss nochmal halbiert werden, damit das LSB eine Frequenz von 1kHz hat. Diesen Wert kann man in die weiter oben angegebene Formel einsetzten, um den Wert für das OCR0A und für den Vorteiler zu erhalten. N wird hier mit 64 angenommen:

Umgeformt auf OCR0A kommt man dann auf folgendes Ergebnis:

Diese Werte können dann im Source Code festgelegt werden.

Ein Bild, das Elektronik enthält.

Automatisch generierte Beschreibung

Abb. 4: Aufbau des Zählers auf dem Steckbrett

/\*

\* Labor\_Timer\_CTC.c

\*

\* Created: 23.10.2019 09:09:09

\* Author : Markus Z

\*/

#include <avr/io.h>

#include <avr/interrupt.h> // Die definierten Interrupts werden importiert

#define *F\_CPU* 16000000 //CPU-Takt auf 16MHz festlegen

unsigned int counter = 0; // Zählervariable deklarieren

void TimerInit0(); //Prototyp für Funktion festlegen

int main(void)

{

DDRA = 0xFF; // DDRA auf Output setzen

InitTimer0(); // Timer im CTC-Mode Initialisieren

while (1)

{

PORTA = counter; // Inhalt der Zählervariable auf PORTA ausgeben

}

}

ISR(TIMER0\_COMPA\_vect) { // Interrupt Service Routine für Compare Match Interrupt definieren

if(counter < 128){ // Überprüfung ob Zählervariable kleiner 128 ist; wenn ja, wird diese inkrementiert

counter++;

}

else // Sollte Zählervariable 128 sein wird diese zurückgesetzt

{

counter = 0;

}

}

InitTimer0() {

TCCR0A |= (1<<WGM01); // CTC Mode mit WGM (Waveform Generation Mode) Bits im TCCR0A Register aktivieren

OCR0A = 124; // Den ausgerechneten Wert 124 für das Compare Register in OCR0A speichern

TCCR0B |= ((1<<CS00) | (1<<CS01)); // Den Prescaler auf 64 festlegen

TIMSK0 |= (1 << OCIE0A); // Den Compare Match Interrupt im Timer Mask Register freischalten

sei (); // Das Global Interrupt Bit im Status Register freischalten, damit der Interrupt ausgelöst werden kann

}

### Training an der Messumgebung

1. Als erstes war es unsere Aufgabe, die Signale am Port A des Mikrocontrollers mit dem internen Takt, also 16MHz, aufzuzeichnen und die Periodendauer und den Duty-Cycle des LSB sowie die Periodendauer zu bestimmen. Ein Vergleich mit der eingestellten Frequenz zeigt nur eine kleine Abweichung von 0.3 Hz.

Ein Bild, das drinnen, Monitor, Computer, Elektronik enthält.

Automatisch generierte Beschreibung

Abb. 5: Gemessene Periodendauer und Duty Cycle des LSB

1. Danach sollten wir die Benennung des ersten Einganges (Channel 0) ändern. Dies erreicht man dadurch, dass man einen Doppelklick mit der linken Maustaste auf den aktuellen Namen (in dem Fall „ZunM“) ausführt. Danach kann man den gewünschten Namen in das sich öffnende Textfeld hineinschreiben.

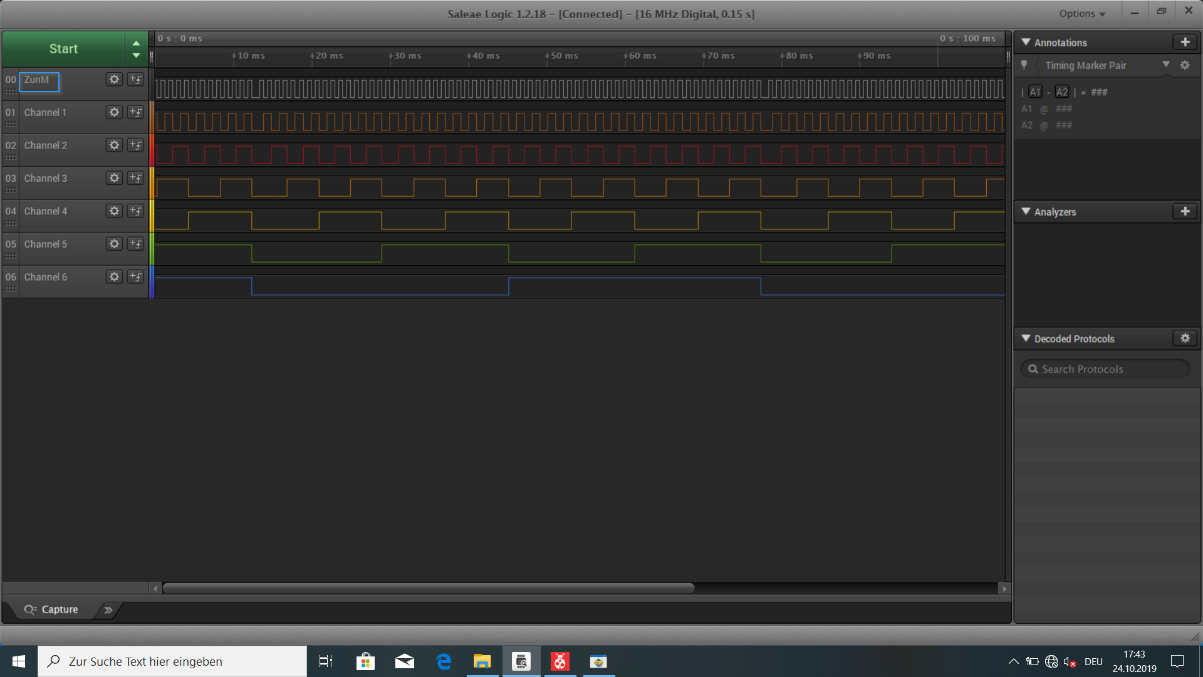


Abb. 6: Änderung der Namensgebung von Channels am Beispiel von Channel 0

1. Als nächstes sollten wir das obere und unter Nibble (obere vier Bits und untere vier Bits) innerhalb der Messumgebung vertauschen. Dazu nimmt man mit der linken Maustaste den Channel im Feld wo die Nummerierung steht und dann kann man ihn frei von unten nach oben schieben.

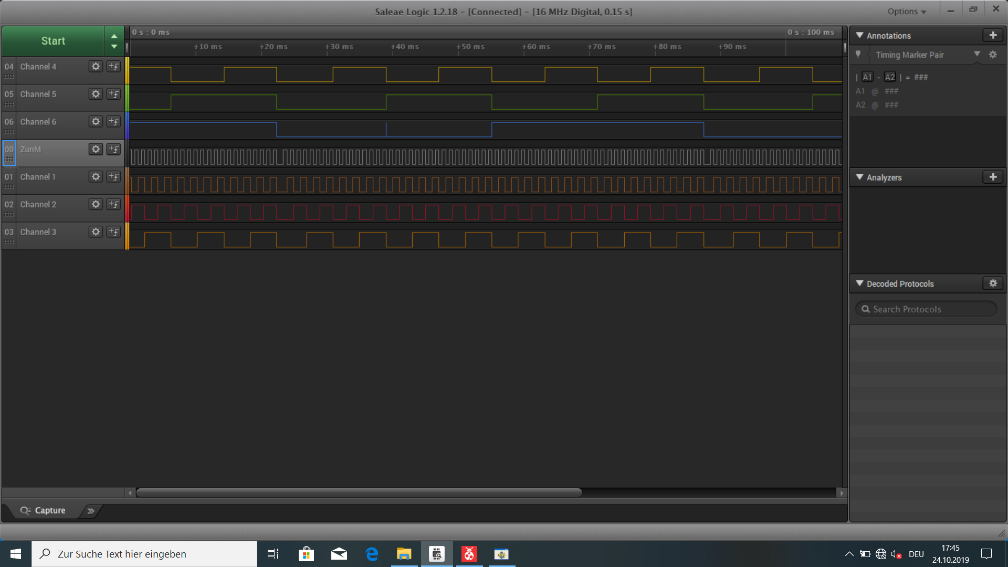


Abb. 7: Oberes und unteres Nibble des Eingangssignals vertauscht. Im blau markierten Feld kann man den Channel mit der Maus nehmen und dann vertauschen.

1. Die letzte Aufgabe dieses Punktes war, die Periodendauer des MSB zu messen. Dabei sollte die Marker-Funktion „snap to edge“ verwendet werden. Diese funktioniert so, dass wenn man mit dem Cursor zu einer Flanke fährt, die Software diese Flanke automatisch als Teil der Messung betrachtet, die man gerade vornehmen will.

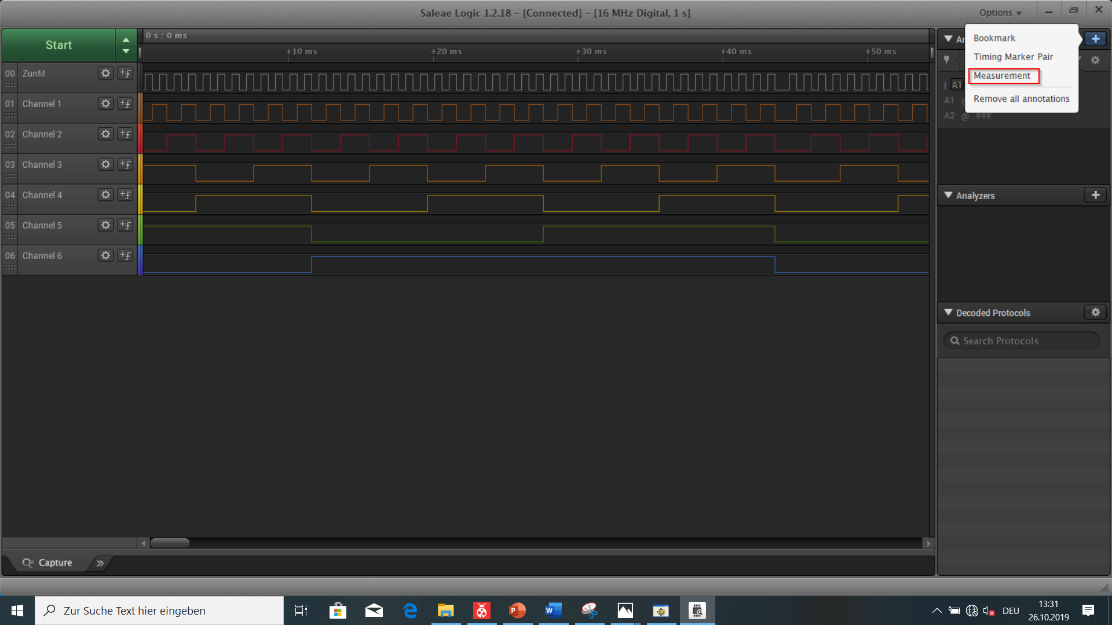


Abb.8: Measurement Funktion hinzufügen

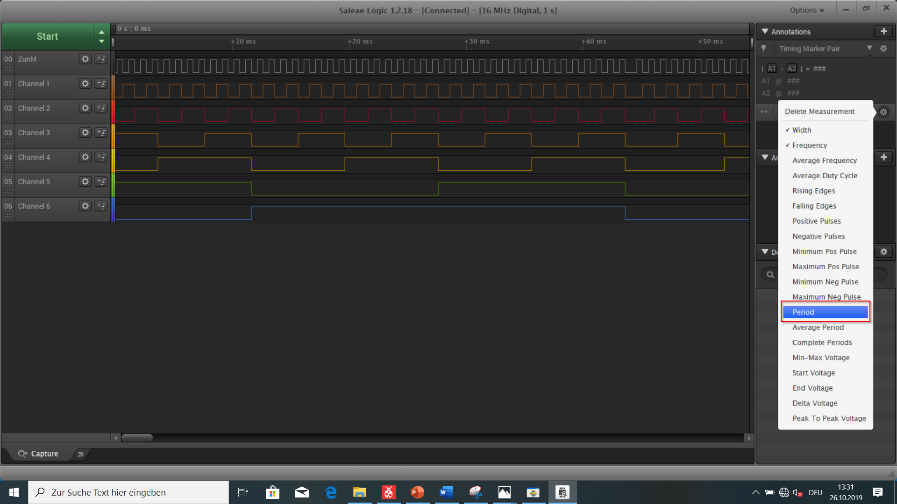


Abb. 9: Periodendauer als zu messende Größe auswählen

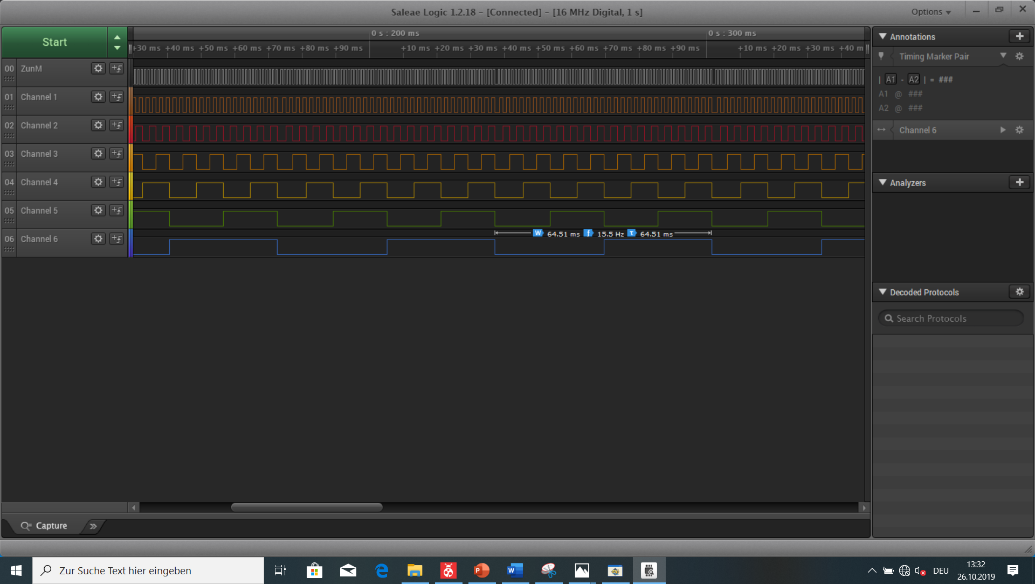


Abb. 10: Gemessene Periodendauer des MSB, die 64.51ms ist

### Triggerung von Signalen

1. Die erste Aufgabe dieses Unterpunkts war es, den Trigger des Logic Analyzers auf ein bestimmtes Event zu setzen. Um da zu erreichen muss man auf den Button „“ klicken und dann kann man den Trigger auswählen (rising edge, falling edge, High, Low).

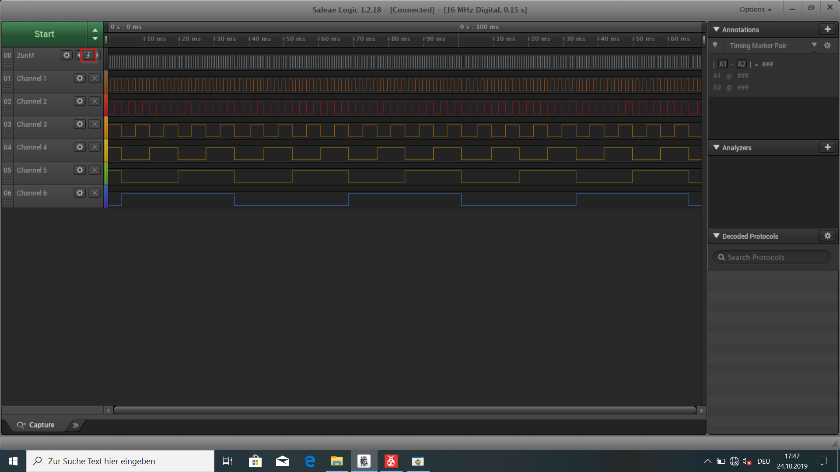


Abb. 11: Triggerung auf die steigende Flanke des LSB

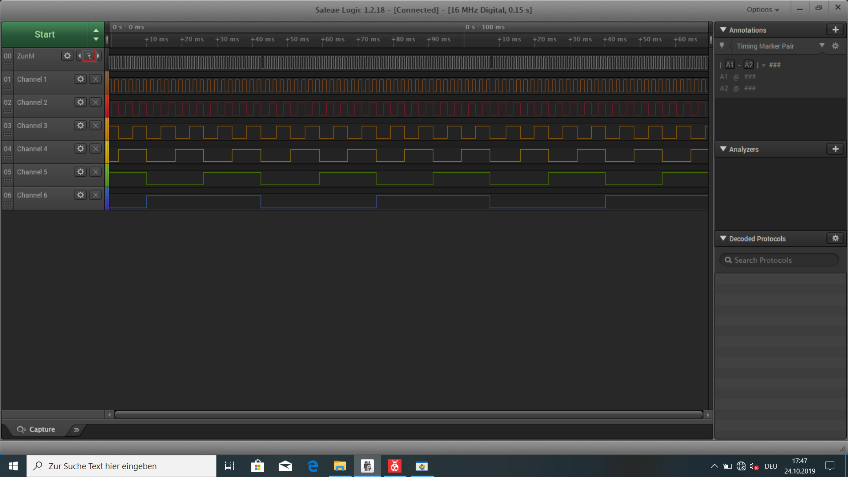


Abb.12: Triggerung auf die fallende Flanke des LSB

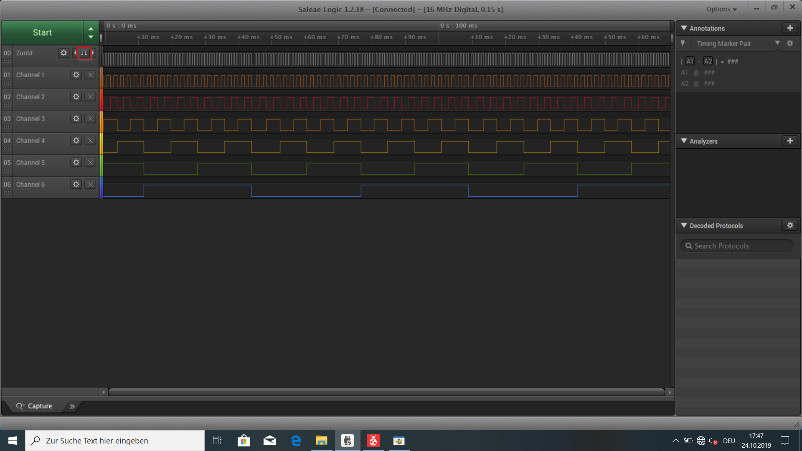


Abb.13: Triggerung auf den „HIGH“ Zustand des LSB

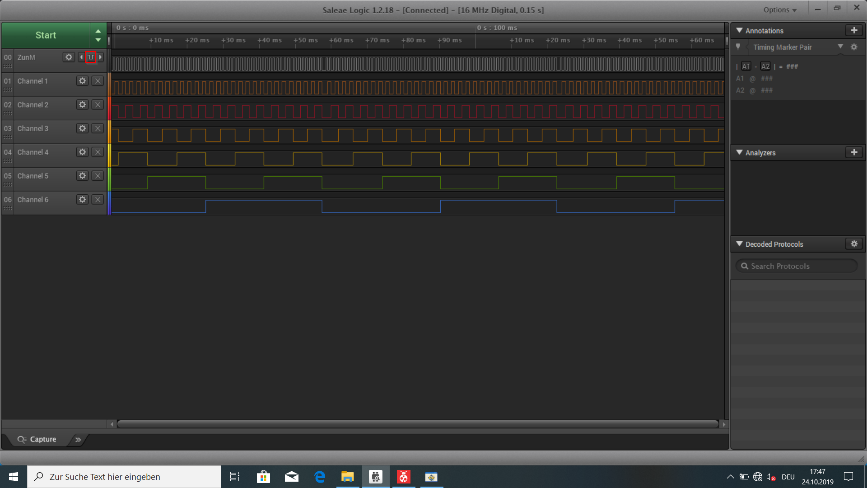


Abb.14: Triggerung auf den „LOW“ Zustand des LSB

1. Als nächste Übung sollte der Trigger auf einen bestimmten Buszustand gesetzt werden. Dabei sollte der Wechsel von 7F (dezimal 127) auf 0 und ein Gruppenspezifischer Wert ((Gruppennummer \* 16) – 1), also bei dieser Gruppe der Wert 47 dargestellt werden. Bei der ersten Messung ist der Trigger bei Channel 6 auf die fallende Flanke zu setzen und bei der zweiten Messung sollte der Trigger von Channel 1 auf die steigende Flanke, der von Channel 5 auf Low und der Trigger der anderen Channels auf High gesetzt werden.

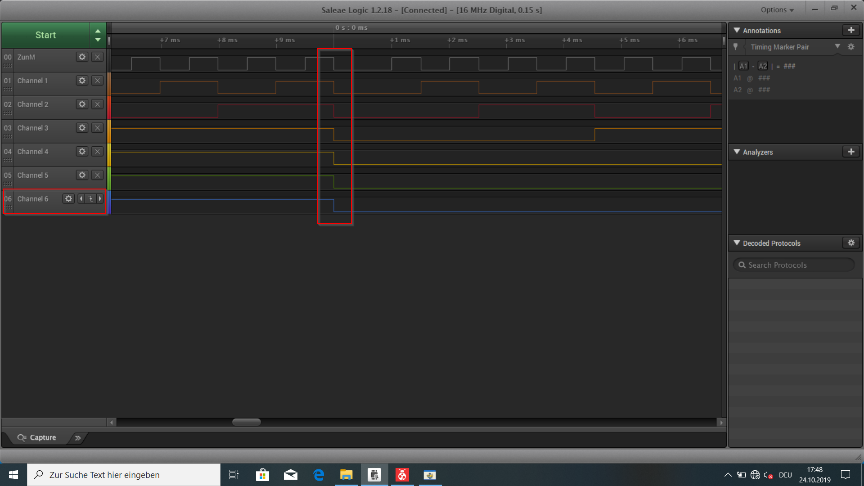


Abb. 15: Wechsel von 7F auf 0

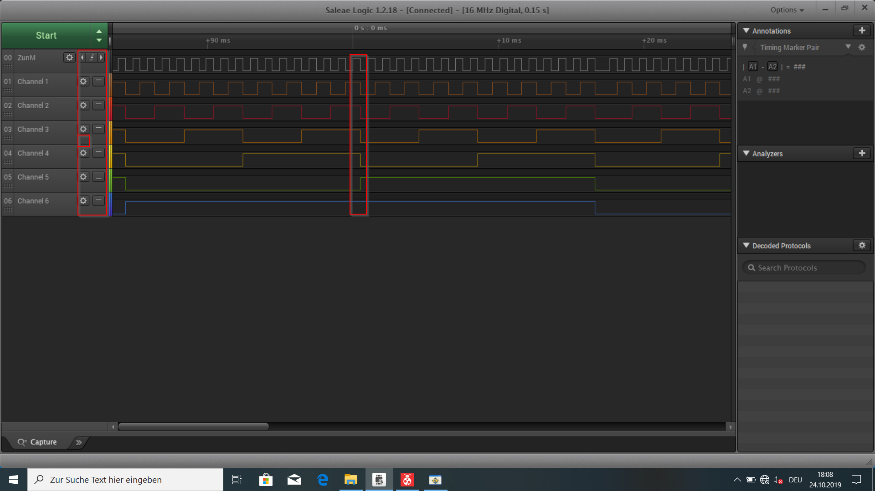


Abb.16: Buszustand „47“

1. Die letzte Aufgabe dieses Unterpunktes bestand darin, das LSB des Zählers mit 22nF oder 100nF durch das hinzuschalten von Kondensatoren zu belasten und die Laufzeitveränderung gegenüber dem unbelasteten Zustand festzuhalten.

Ein Bild, das Monitor, Elektronik, drinnen, Computer enthält.

Automatisch generierte Beschreibung

Abb. 17: Pulsweite des unbelasteten LSB

Ein Bild, das Monitor, Elektronik, drinnen, Computer enthält.

Automatisch generierte Beschreibung

Abb. 18: Pulsweite des mit 22nF belasteten LSB

Ein Bild, das Monitor, Elektronik, drinnen, Computer enthält.

Automatisch generierte Beschreibung

Abb. 19: Pulsweite des mit 100nF belasteten LSB

Was bei diesen Abbildungen auffällt, ist, dass mit steigender Kapazität der Belastung die Pulsweite und somit die Laufzeit des LSB zunimmt. Dies ist darauf zurückzuführen, dass sich die Kondensatoren während dem „HIGH“ Zustands des LSB aufladen und nachdem dieses Signal auf Low gegangen ist, halten sie die Leitung durch ihre Ladung noch kurz auf High, bevor sie sich entladen haben. Je größer die Kapazität der Belastung, um so länger wird die Leitung zusätzlich auf High gehalten.

**Messwerte im Detail:**

|  |  |
| --- | --- |
| **Pulsweite [ms]** | **Kapazitive Belastung [nF]** |
| **0.5003** | **0** |
| **0.5008** | **22** |
| **0.5021** | **100** |

### 4.1.3 Auflösung und Speichertiefe

Die Aufgabe bei diesem Unterpunkt war es, die Frequenz des Zählers auf 32kHz zu erhöhen durch die Veränderung der Abtastrate die Auswirkungen auf den gemessenen Duty-Cycle der einzelnen Bits sichtbar zu machen. Die Berechnung des OCR0A erfolgte wie weiter oben angegeben. Daher wird diese genau wie der veränderte Code hier nur abschnittsweise wiedergegeben. Der Vorteiler N war dieses Mal 1:

Im Source Code wurden folgende Veränderungen vorgenommen:

…

OCR0A = 249; //Wert für Compare Register auf 249 festlegen

TCCR0B |= (1<<CS00); //Prescaler auf 1 setzten, also CPU-Takt verwenden

..

1. Als erstes sollte bei diesem Unterpunkt eine Abtastrate von 50kHz gewählt und damit der Duty-Cycle der einzelnen Zählerbits dokumentiert werden.

Ein Bild, das Monitor, Elektronik, drinnen, Computer enthält.

Automatisch generierte Beschreibung

Abb. 20: Duty-Cycle des LSB und des LSB+1 bei einer Abtastrate von 50kHz und einer Zählfrequenz von 32kHz

1. Als nächstes wurde die Abtastrate auf 1MHz erhöht und wieder der Duty-Cycle (DC) gemessen.

Ein Bild, das Monitor, drinnen, Elektronik, Computer enthält.

Automatisch generierte Beschreibung

Abb. 21: Duty-Cycle bei einer Abtastrate von 1MHz

Durch die Erhöhung der Abtastrate wird die Messung des DC genauer, da der LA laut Abtasttheorem mindestens mit der doppelten Frequenz des Eingangssignals abtasten muss, um das Signal gut analysieren zu können. Mit einer Abtastrate von 1MHz, tastet der LA das Eingangssignal mehr als ausreichend oft ab, um es sehr genau darzustellen und zu analysieren.

1. Als Abschließende Übung zu diesem Unterpunkt wurde die Abtastrate ausgehend von 1MHz sukzessiv bis unter die Zählfrequenz vermindert und dabei die Auswirkung auf die Messung des DC untersucht. Die Abtastraten 50kHz und 1MHz wurden bei dieser Messung ausgelassen, da sie schon weiter oben dokumentiert sind.

Ein Bild, das Elektronik, Monitor, drinnen, Computer enthält.

Automatisch generierte Beschreibung

Abb.23: DC des LSB und des LSB+1 bei einer Abtastrate von 500kHz

Ein Bild, das Monitor, drinnen, Elektronik, Computer enthält.

Automatisch generierte Beschreibung

Abb. 24: DC des LSB und des LSB+1 bei einer Abtastrate von 250kHz

Ein Bild, das Monitor, Elektronik, drinnen, Computer enthält.

Automatisch generierte Beschreibung

Abb. 25: DC des LSB und des LSB+1 bei einer Abtastrate von 200kHz

Ein Bild, das Monitor, drinnen, Elektronik, Computer enthält.

Automatisch generierte Beschreibung

Abb. 26: DC des LSB und des LSB+1 bei einer Abtastrate von 100kHz

Ein Bild, das drinnen, Monitor, Elektronik, Computer enthält.

Automatisch generierte Beschreibung

Abb. 27: DC des LSB und des LSB+1 bei einer Abtastrate von 25kHz

Wie man aus den Messergebnissen ablesen kann, verändert sich bei sinkender Abtastrate sowohl die gemessene, durchschnittliche Frequenz, als auch der gemessene, durchschnittliche Duty-Cycle. Zumindest bei dem LSB kann man auch mit sinkender Abtastrate eine Veränderung der Form des gemessenen Signals feststellen. Der Grund für diese Veränderungen ist, dass der LA die Eingangssignale mindestens mit der doppelten Frequenz abtasten muss, um es exakt darzustellen. Ist dies nicht der Fall, so kann der LA das Signal nicht korrekt messen und somit sind die aus der Messung resultierende Darstellung und Berechnung ungenau.

Mit steigender Abtastrate werden hingegen die Messung und die Berechnungen immer genauer.

## Protokollierung eines I2C Datentransfers

Der zweite Teil dieser Laborübung bestand darin, den Temperatursensor LM75 über den I2C-Bus des CRUMTL anzusteuern. Diese Datenübertragung sollte mit dem LA aufgezeichnet und dargestellt werden. Dabei sollten die Entsprechenden Bits, wie etwa das ACK-Bit, in der Messumgebung gekennzeichnet werden und danach die gemessene Temperatur ausgewertet werden. Um dies zu bewerkstelligen, muss in der Software unter dem Menü „Analyzers“ der Eintrag I2C ausgewählt werden. Danach kennzeichnet die Software die speziellen Bits automatisch. Softwaretechnisch wurde vom Übungsleiter noch das fertige File „I2C\_basic.c“ zur Verfügung gestellt, welches nur noch in Atmel Studio kompiliert und auf den CRUMTL geladen werden musste. Dieses File ist bei den Messanweisungen auf dem Klassenlaufwerk verfügbar und wird daher nicht explizit dokumentiert. Beim Aufbau muss daher nur darauf geachtet werden, dass die herausgeführten Adress-Pins des LM75 auf Masse gelegt und damit auf 0 gesetzt werden, damit der Code funktioniert, sowie dass die SDA und die SCL mit zwei 10kΩ Pull-Up Widerständen auf 5V gelegt werden.

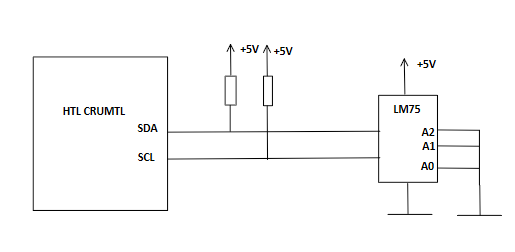


Abb. 28: Schematischer Aufbau der Messschaltung

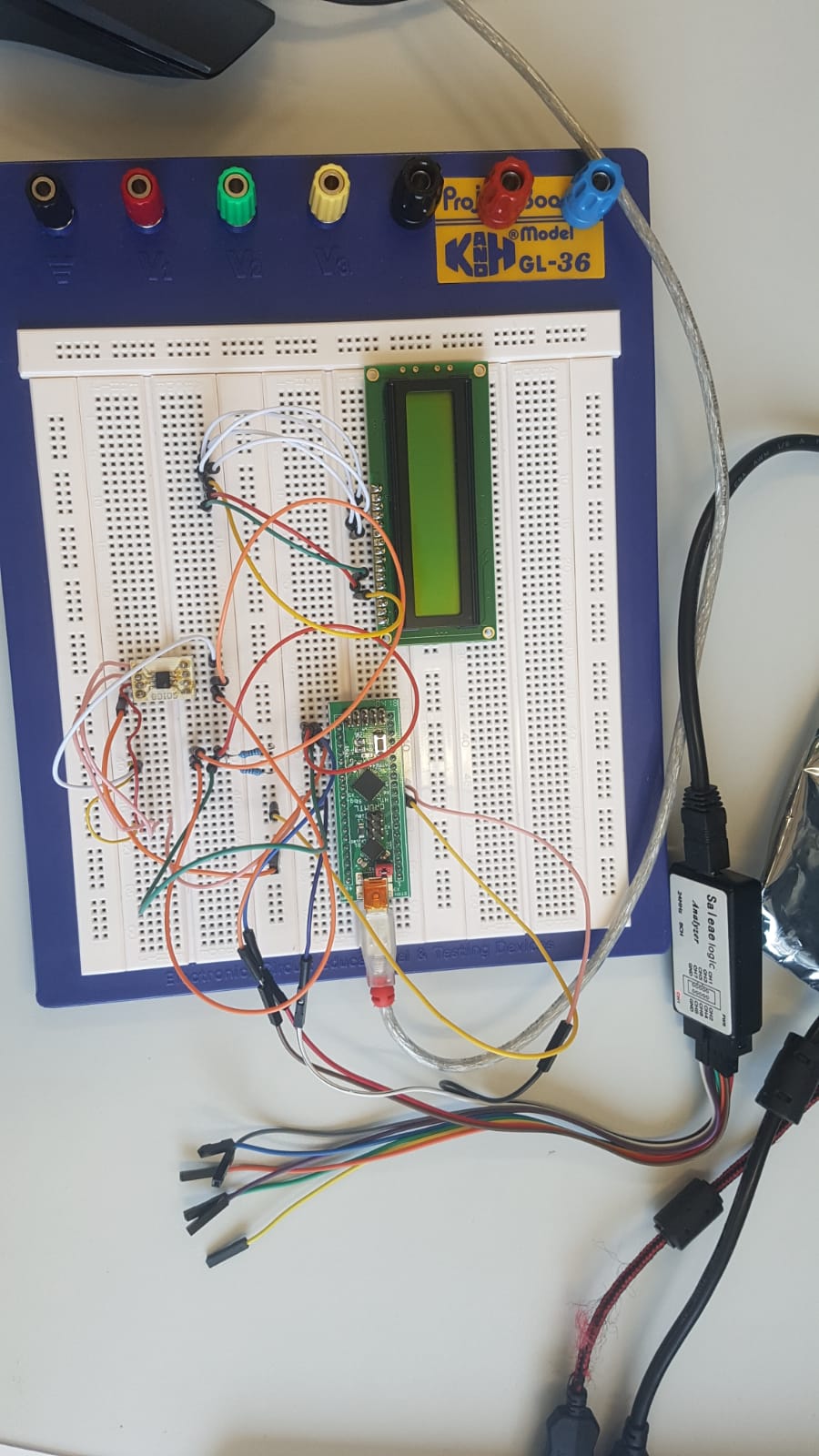


Abb. 29: Aufbau der Messschaltung auf dem Steckbrett

**Messergebnis:**

Die Analyse mit dem Logic Analyzer führte zu folgendem Ergebnis:



Abb. 30: Aufgezeichnete I2C Kommunikation zwischen CRUMTL und LM75

**Auswertung des Messergebnisses:**Gekennzeichnete Stellen:

1. Adresse
2. R/
3. Most Significant Data Byte
4. ACK
5. Least Significant Data Byte
6. NACK
7. ACK

Eine genaue Auswertung der oben angeführten Kommunikation wäre folgendermaßen:



Die in Blau markierten Bits im Most Significant Byte ergeben im Hexadezimale System 10 und stehen daher für eine Temperatur von 10℃ und das MSB im Least Significant Data Byte heißt, das zu den 10℃ noch 0,5℃ dazu addiert gehören. Der Temperatursensor misst somit eine Temperatur von 10,5℃. Die restlichen Bits im Least Significant Byte dürfen laut Datenblatt ignoriert werden.

# Zusammenfassung

* Zusammenfassung:

Dies war eine sehr interessante Übung, um zu sehen, dass es neben dem Oszilloskop eine weitere Möglichkeit gibt, Signale grafisch darzustellen und zu analysieren. Auch war es gut, die im letzten Jahr in DIC erlernte Theorie über die Timer eines Mikrocontrollers nochmal zu wiederholen und durch die Analyse mit dem Logic Analyzer auf ihre Richtigkeit hin zu überprüfen.

* Fragen:

1. Wie unterscheidet sich ein Logikanalysator von einem DSO?

Ein Logic Analyzer unterscheidet sich dahingehend von einem DSO, dass er viel mehr Eingänge besitzt und es mit ihm möglich ist, digitale Signale aufzuzeichnen, darzustellen und hinsichtlich der unterstützten Bus-Protokolle (I2C, SPI…) zu analysieren. Darüber hinaus besitzt ein LA einen sehr komplexen Trigger er verdeckt große Teile des Signales, während ein Oszilloskop einen einfacheren Trigger besitzt und ein Signal komplett, mit allen Schwankungen und Spannungsspitzen darstellt.

1. Welche Sample Rate ist zur Aufzeichnung digitaler Signale anzuwenden?

Dem Abtasttheorem zufolge muss die Sample Rate mindestens doppelt so groß sein wie die Frequenz der Eingangssignale, um diese exakt darzustellen. Jede darüber hinausgehende höhere Sample Rate sorgt für eine bessere Darstellung des Signals.

1. Beschreibe in einem Timingdiagramm den I2C Schreibzugriff auf die 7Bit Adresse 55h mit dem Datum 99h.

Diese Frage ist laut dem Übungsleiter nicht zu Beantworten.

Unterschrift:

|  |  |  |  |
| --- | --- | --- | --- |
| **Datum:** | **Note:** | **Punkte:** | **Unterschrift:** |