**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 05**

**Gegenstand der Übung**

|  |
| --- |
| **DAC SPI (Digital-to-analog converter)** |
|  |

|  |  |
| --- | --- |
| **Name:** | **Christian Kreidenhuber** |
| **Jahrgang:** | **4AHEL** |
| **Gruppe Nr.:** | **B01** |
| **Übung am:** | **18.12.2019** |

|  |  |
| --- | --- |
| **Anwesende:** | Christian Kreidenhuber, Clemens Hütter |

***Inhaltsverzeichnis***

[1. Einleitung 3](#_Toc29150336)

[1.1. LTC1661 3](#_Toc29150337)

[2. Inventarliste 3](#_Toc29150338)

[3. Übungsdurchführung 4](#_Toc29150339)

[3.1. Timing-Diagramm / Kenngrößen 4](#_Toc29150340)

[3.2. DAC (Konstante Spannung) 5](#_Toc29150341)

[3.2.1. Schaltung 5](#_Toc29150342)

[3.2.2. Berechnung 5](#_Toc29150343)

[3.2.3. Programm Code 6](#_Toc29150344)

[3.2.4. Messergebnisse 6](#_Toc29150345)

[3.3. DAC (Sägezahn) 7](#_Toc29150346)

[3.3.1. Schaltung 7](#_Toc29150347)

[3.3.2. Berechnung 7](#_Toc29150348)

[3.3.3. Programm Code 8](#_Toc29150349)

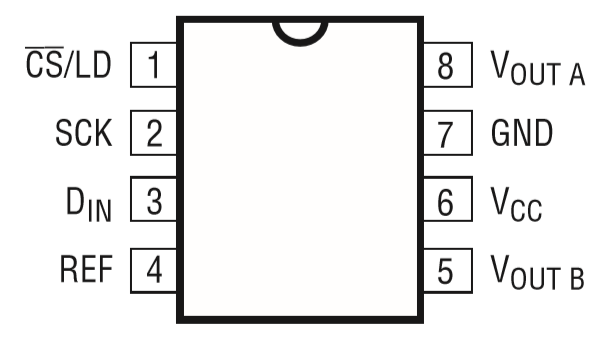
[3.3.4. Messergebnisse 9](#_Toc29150350)

# Einleitung

In dieser Übung wurde ein Digital-to-analog Converter aufgebaut. Als Eingang verwendet dieser eine Bitfolge und transformiert diese in ein analoges Signal.

## LTC1661

Der LTC1661 verfügt über zwei DACs. Diese können unabhängig voneinander angesteuert werden (Vout A, Vout B). Das verwendete Protokoll für die Datenübertragung ist hierbei SPI (CS, SCK, Din).



*Abbildung 1: Pin-Layout des LTC1661, Quelle: LTC1661* [*www.linear.com*](http://www.linear.com)

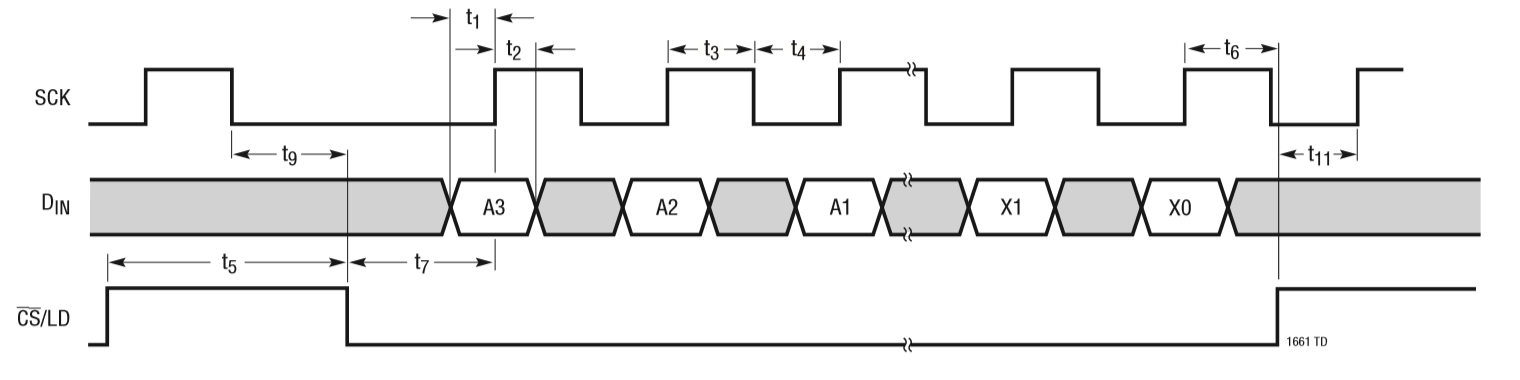
# Inventarliste

|  |  |  |
| --- | --- | --- |
| **Stück** | **Gerätebezeichnung** | **Inventarnummer** |
| 1 | Hera Spannungs/Stromquelle | Nicht vorhanden |
| 1 | UNI-T UT803 True RMS Multimeter | Nicht vorhanden |
| 1 | KEYSIGHT InfiniiVision DSOX2014A | 400000480074 0000 |

# Übungsdurchführung

## Timing-Diagramm / Kenngrößen

Um Daten korrekt an den DAC zu übertragen können, müssen bestimmte Zeiten aus dem Datenblatt eingehalten werden.



*Abbildung 2: Timing-Diagramm des LTC1661, Quelle: LTC1661* [*www.linear.com*](http://www.linear.com)

Zuerst wird die CS Leitung von High auf Low gezogen, um den Chip zu aktivieren. Als nächstes werden die einzelnen Bits angelegt (angefangen mit dem MSB). Damit jedes Bit vom Chip registriert wird, muss die Clock Leitung einen Flankenwechsel von Low auf High erreichen. Nach dem Senden der Daten (in diesem Fall 16 Bits), wird die CS Leitung wieder auf High gezogen, um die Übertragung zu beenden.

**Kenngrößen LTC1661:**

*Datenwortbreite:* Die Anzahl der Bits für einen vollständigen Befehl (16 Bits)

*Aufbau des Datenwortes:* 4 Control Bits, 10 Daten Bit, 2 Don’t Care Bits

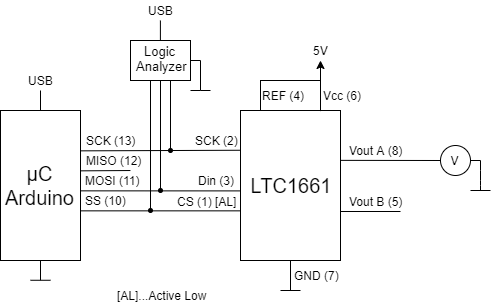
*Referenzspannung:* Zwischen 0V und Vcc

*Spannung am Ausgang:* Abhängig von den 10 Daten Bits. Sie liegt im Bereich von 0V bis Vref \* (1023/1024). Für eine beliebige Bitkombination kann sie folgendermaßen berechnet werden:

*Taktfrequenz:* Die Frequenz des Taktsignals darf 10MHz nicht überschreiten

## DAC (Konstante Spannung)

### Schaltung



*Abbildung 3: Messschaltung*

Für die Datenübertagung wird ein Mikrocontroller in Form eines Arduinos verwendet. Dabei werden die Daten mit einem Logic Analyzer aufgenommen, um anschließend auf dem Computer visualisiert werden zu können. Nachdem der DAC den Befehl ausgeführt hat, wird sich die Spannung am Ausgang ändern und mit einem Voltmeter aufgenommen.

### Berechnung

Am Ausgang soll eine Spannung von 2V realisiert werden. Mit der Formel aus 3.1. lässt sich das Datenwort berechnen:

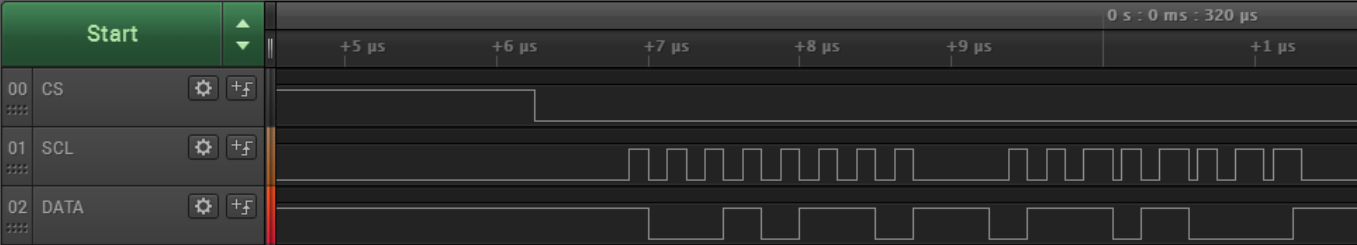
Dazu kommen noch die Steuer Bits und die Don’t Care Bits. Um den Ausgang A anzusteuern, muss laut dem Datenblatt für die Steuer Bits **1001b** verwendet werden. Dieser Befehl lädt die Daten in den DAC A und aktualisiert den Ausgang. Die Don’t Care Bits können beliebig gewählt werden, da diese keinen Einfluss auf den Befehl haben und lediglich als Platzhalter dienen.

Als Datenwort ergibt sich nun: 1001 0110 0110 1000b

### Programm Code



### Messergebnisse



*Abbildung 4: Timing Diagramm, aufgenommen mit einem Logic Analyzer*

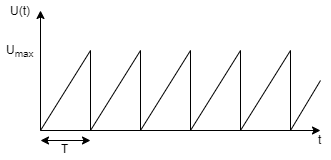
Man erkennt, dass alle Daten korrekt übertragen werden. Die standardmäßige Frequenz des SPI Interfaces des Arduinos beträgt 400kHz. Dies kann man sehr gut im Bereich von 7µs bis 8µs sehen, da dort genau vier Perioden vergehen. Am Voltmeter ergibt sich ebenso die vorgesehene Spannung, mit einem geringen Fehler: **2,021V**

## DAC (Sägezahn)

### Schaltung

Die Schaltung ist hierbei dieselbe wie in 3.2.1., nur dass statt einem Voltmeter ein Oszilloskop verwendet wurde.

### Berechnung



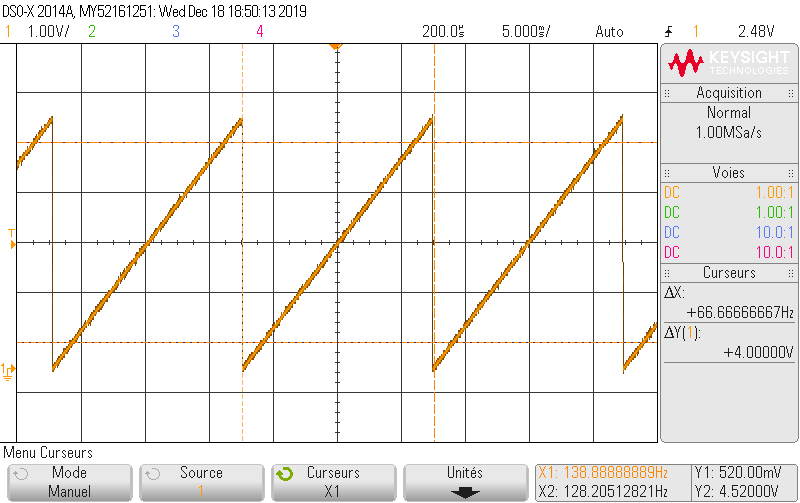
*Abbildung 5: Funktion eines Sägezahns*

Um einen Sägezahn mit einem DAC zu realisieren, muss lediglich der Faktor k inkrementiert werden und bei dem Maximalwert wieder zurückgesetzt werden. Die Periodendauer ergibt sich aus der Taktfrequenz des SPI Interfaces und der Auflösung.

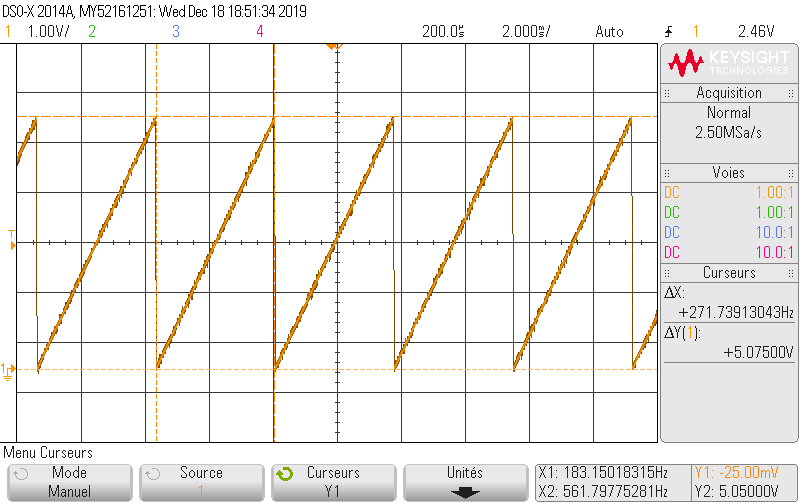
### Programm Code



### Messergebnisse



*Abbildung 6: Oszilloskopbild des Sägezahns (Frequenzunoptimiert)*



*Abbildung 7: Oszilloskopbild des Sägezahns (Frequenzoptimiert)*

In Abbildung 6 sieht man die Sägezahnfunktion mit einer Frequenz von 66Hz und aufgrund einer fehlerhaften Cursor-Platzierung eine Sp.-Sp. Spannung von 4V. Diese sollte wie in Abbildung 7 bei 5V liegen. In Abbildung 7 erkennt man, dass die Frequenz ungefähr 4 Mal so groß (271Hz) wie in Abbildung 6 ist, da die Schleife im Programm 4 Mal so schnell durchlaufen wird. Dadurch sinkt dementsprechend die Auflösung (Die Schleife wird 4 Mal so schnell durchlaufen, indem man den Indexer i bei jedem Schleifendurchgang um 4 erhöht).

Bilder des Logic Analyzer wurden aus zeittechnischen Gründen nicht mehr aufgenommen.

Unterschrift:

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