

HÖHERE TECHNISCHE BUNDESLEHRANSTALT

HOLLABRUNN

Höhere Abteilung für Elektronik – Technische Informatik

Klasse/ Jahrgang: 5BHEL	Gruppe: 1 / b	Übungsleiter: Prof. Reisinger
Übungsnummer: I / 4	Übungstitel: DSP Oszillatoren	
Datum der Übung: 22.09.15	Teilnehmer: Dorner, Mötz, Oberhamberger, Prinz	
Datum der Abgabe: 06.10.2015	Schrittführer: Dorner Fabian	Unterschrift:

	Beurteilung
Aufgabenstellung	
Dokumentation	
Messschaltungen	
Messtabellen	
Berechnungen	
Programmlistings	
Auswertung	
Diagramme	
Berechnungen	
Simulationen	
Schlussfolgerungen	
Kommentare	
Inventarliste	
Messprotokoll	
Form	
Summe	

Allgemeiner Teil

Titel der Übung: DSP Oszillatoren

Übungsnummer: I / 4

Übungsplatz: T10

Datum der Übung: 22.09.2015

Klasse: 5BHEL

Schritfführer: Dorner Fabian

Übungsteilnehmer: Dorner Fabian

Mötz Florian

Oberhamberger Christoph

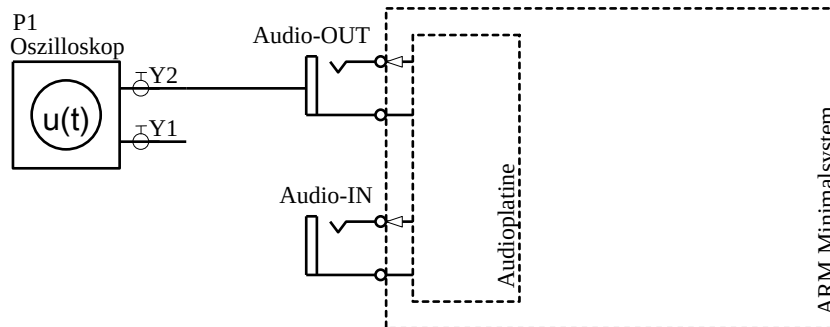
Prinz Bernhard

Inhaltsverzeichnis

1. Messaufbau	3
2. Testen der DSP	3
3. Aufgabenstellung	4
3.1. Rampe generieren	4
3.2. High- und Low-Pegel messen.....	6
3.3. Dreiecksignal generieren	7
3.4. Rechtecksignal generieren	8
3.5. Sinus generieren.....	8
3.6. Sinus um $\pi/4$ verschoben.....	9
4. Erkenntnisse	10
Messprotokoll.....	11
Inventarliste	13

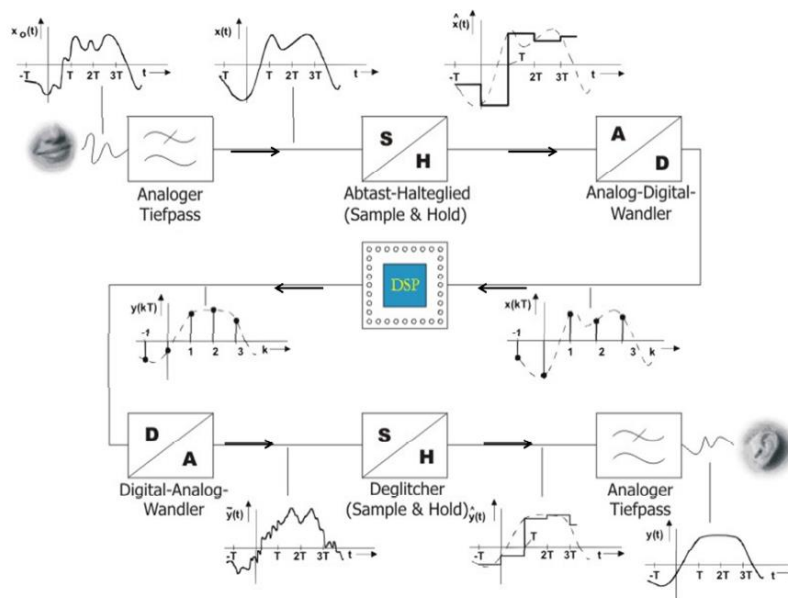
1. Messaufbau

Der Messaufbau bestand aus einem ARM-Minimalsystem auf dem eine Audio-Platine mit Input und Output verbaut war. Die generierten Signale (siehe Punkt 2. ff) wurden mit einem Oszilloskop gemessen und die Messungen wurden, über eine serielle Verbindung mit dem PC, digital abgespeichert.



2. Testen des DSP

Bevor mit den eigentlichen Übungsaufgaben begonnen wurde, wurde die gesamte Kette der digitalen Signalverarbeitung getestet. Dazu gehören der Antialiasing-Tiefpass, ein Abtast- und Halteglied, ein AD-Wandler, der DSP (**D**igital **S**ignal **P**rocessor), ein DA-Wandler, ein Deglitcher und ein Rekonstruktions-TP.



Getestet wurde diese Kette mittels folgendem Programm:

```
void process_data(void)
{
    in_1 = Audio_IN[0];      //Audio-IN
    in_2 = Audio_IN[1];

    out_1 = in_1;
    out_2 = in_2;

    Audio_OUT[0] = out_1;    //Audio-OUT setzen
    Audio_OUT[1] = out_2;
}
```

Das Audio-Eingangssignal stammt von der Audio-Ausgangsbuchse eines Laptops eines Schülers über die Musik ausgesendet wurde.

3. Aufgabenstellung

Die Aufgabe dieser Übung war es verschiedene Signale digital zu generieren. Bei den Signalen handelt es sich um eine Rampe, Dreieck, Rechteck, Sinus und einen um 45° verschobenen Sinus. Die generierten Signale sollten eine Frequenz von 750Hz haben. Weiters wurde das High- und Low-Level des Audio-Ausgangs gemessen.

3.1. Rampe generieren

Es sollte eine Rampe mit 750Hz generiert werden. Dafür wurde die Zählvariable „ramp“ angelegt welche bei jedem Aufruf des Unterprogramms um einen gewissen Wert erhöht wird. Diese Variable wird bei jedem Durchlauf am Ausgang ausgegeben. Die Frequenz mit der das UP aufgerufen wird ist die Abtastfrequenz.

Der Programmcode dafür sieht wie folgt aus:

```
void process_data(void)
{
    ramp += 1024;            //erzeugen der Rampe

    out_1 = ramp;

    Audio_OUT[0] = out_1;    //Audio-OUT setzen
}
```

Der Wert um den die Zählvariable erhöht werden muss, wurde so berechnet:

$$f_{step1} = \frac{f_{abtast}}{2^{bits}} = \frac{48.000 \text{ Hz}}{2^{16}} = 0,7324 \text{ Hz/step}$$

f_{step1}Frequenz wenn der Output pro Schritt um 1 erhöht wird

f_{abtast}Abtastfrequenz des DAC

2^{bits} Anzahl der Schritte

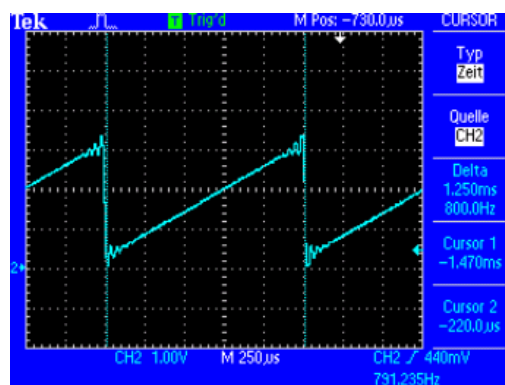
$bits$ Anzahl der Bits des Datentyps

$$step_{750} = \frac{f_{ramp}}{f_{step1}} = \frac{750 \text{ Hz}}{0,7324 \text{ Hz}} = 1024$$

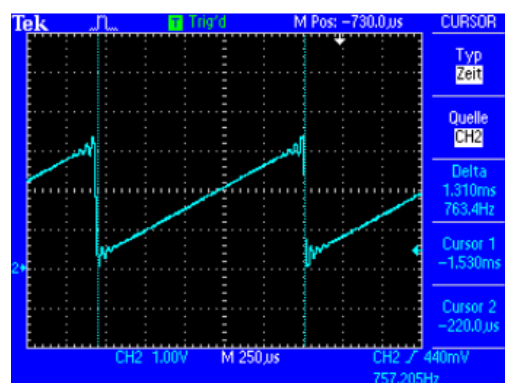
$step_{750}$ Erhöhung des Outputs pro Schritt für 750 Hz

f_{ramp} Frequenz der Rampe

Das berechnete Ergebnis erhöht dort die Output-Variable bei jedem Schritt der Zählvariable um 1024. Beim messen des Audio-Ausgangs stellte sich jedoch heraus, dass dieser Wert nicht passte um eine Frequenz von 750Hz zu erreichen. Wie das nachfolgende Bild zeigt waren die Schritte zu groß da die gemessene Frequenz bei 800Hz lag.



Durch ausprobieren wurde eine Schrittweite von 980 gefunden die eine Rampe mit einer Frequenz von 763Hz erzeugte.



Somit kommt man auch über die obere Formel zum Rückschluss, dass die Abtastfrequenz eine andere ist als die angenommenen 48kHz. Wenn man zurückrechnet ist diese nämlich ca. 51kHz.

$$f_{step1} = \frac{f_{ramp}}{step_{763}} = \frac{763 \text{ Hz}}{980} = 0,7786... \text{ Hz/step}$$

$$f_{abtast} = f_{step1} * 2^{bits} = 0,7786 \frac{\text{Hz}}{\text{step}} * 2^{16} = 51.024,46... \text{ Hz}$$

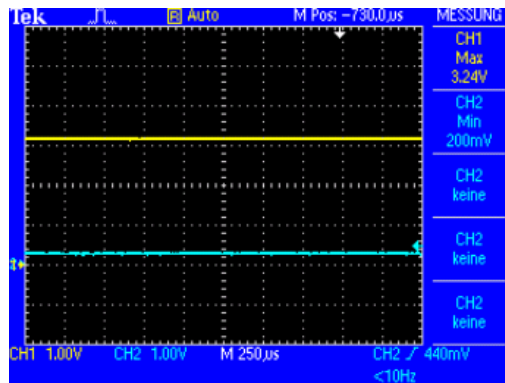
3.2. High- und Low-Pegel messen

Durch das q15-Format ergibt sich für den höchsten Wert und somit den höchsten Pegel die hexadezimale Zahl 0x7FFF und für den niedrigsten Wert und somit den niedrigsten Pegel die Zahl 0x8000.

Programmcode:

```
void process_data(void)
{
    out_1 = 0x7FFF;           //größter Wert --> High-Level
    out_2 = 0x8000;           //kleinster Wert --> Low-Level

    Audio_OUT[0] = out_1;     //Audio-OUT setzen
    Audio_OUT[1] = out_2;
}
```



Das Oszilloskop zeigt, dass wenn der Ausgang auf einen High-Pegel (0x7FFF) gesetzt wird die Ausgangsspannung 3,24V beträgt, bei einem Low-Pegel (0x8000) ist die Spannung 0,2V groß.

3.3. Dreiecksignal generieren

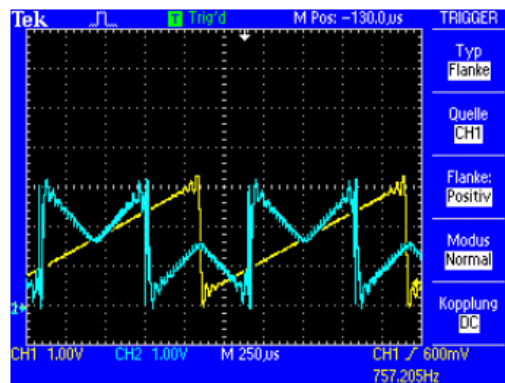
Um ein Dreiecksignal zu erzeugen wurde der Betrag des Rampensignals

Programmlisting:

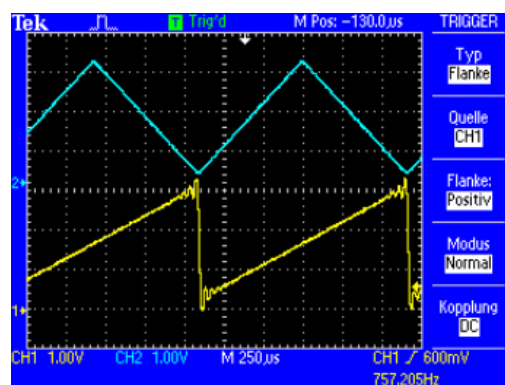
```
void process_data(void)
{
    //Dreieck generieren
    arm_abs_q15(&ramp, &output, 1); //Betrag von ramp

    output = output - 0x7FFF; //Betrag um Hälfte nach unten verschoben
    output = 0x7FFF - ((output)*2); //Betrag verdoppeln, um Hälfte
                                    nach oben schieben
    Audio_OUT[0] = output; //Audio-OUT setzen
}
```

Bei unserem ersten Versuch kam das folgende Signal am Ausgang heraus. Dabei wurde auf die letzte Verschiebung um die Hälfte nach „oben“ vergessen.



Nachdem das Signal nach „oben“ Verschieben wurde war das Signal ein perfektes Dreiecksignal.



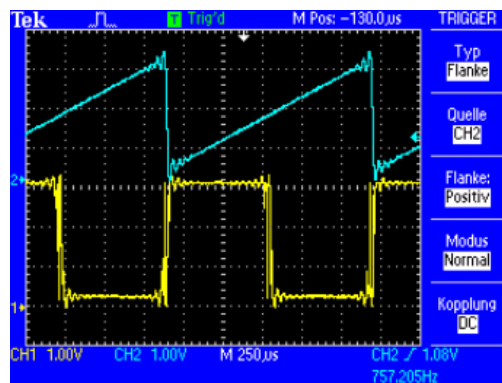
3.4. Rechtecksignal generieren

Hier wurde ein Rechtecksignal mit einem Tastverhältnis von 50:50 generiert. Solange das interne Rampensignal „ramp“ kleiner als 0x0000 ist, wird ein High-Pegel ausgegeben. Überschreitet die „ramp“-Variable die „0x0000“-Grenze wird der Output auf ein Low-Level gesetzt.

Programmlisting:

```
void process_data(void)
{
    //Rechteck generieren
    if(ramp < 0)
    {
        output = 0x7FFF;
    }
    else
    {
        output = 0x8000;
    }

    Audio_OUT[0] = output;    //Audio-OUT setzen
}
```



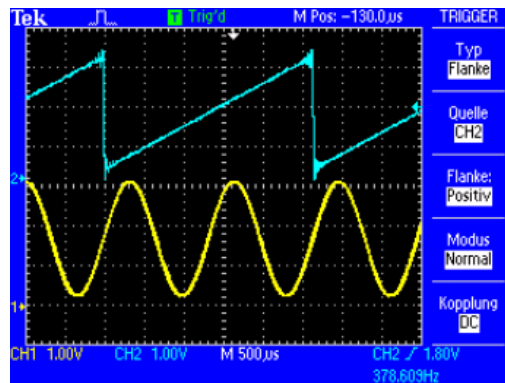
3.5. Sinus generieren

Mit Hilfe der Funktion „arm_sin_q15“ konnten leicht die Sinuswerte im q15-Format ermittelt werden. Diese Funktion benötigt als Übergabeparameter einen positiven x-Wert.

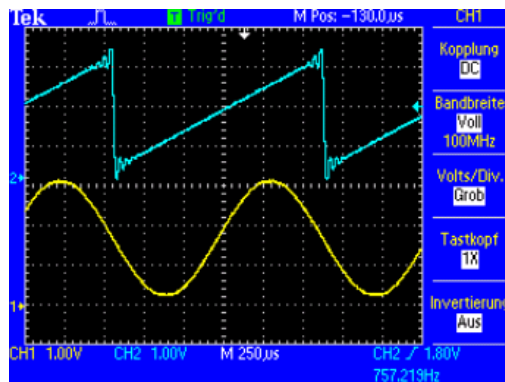
Programmlisting:

```
void process_data(void)
{
    //Sinus generieren
    help = ramp/2;    //
    output = arm_sin_q15(help & (~0x8000));    //Sinuswert als Look-
                                                Up-Table holen
    Audio_OUT[0] = output;    //Audio-OUT setzen
}
```

Beim ersten Versuch wurde ein Sinus mit der doppelten Frequenz ausgegeben.



Nachdem dieser kleine Fehler behoben wurde sah das Sinussignal wie folgt aus.

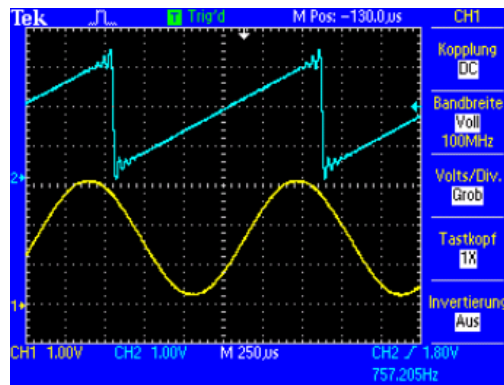


3.6. Sinus um $\pi/4$ verschoben

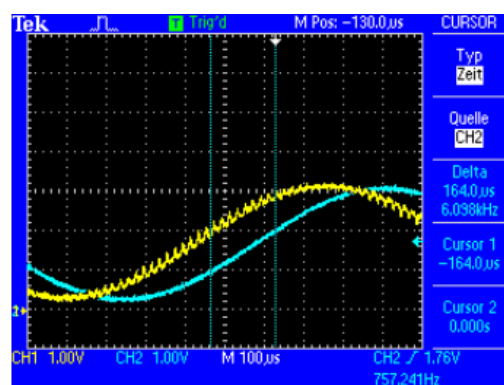
Als Letztes sollte noch ein Sinus erzeugt werden der um $\pi/4$ verschoben ist. Dieser wurde generiert durch eine Verschiebung der Rampe um 1/8 des gesamten Zahlenbereichs und wiederum mit der Funktion „arm_sin_q15“.

Programmlisting:

```
void process_data(void)
{
    //Sinus generieren
    help = (ramp - 0x1FFF)/2;           //Rampe um  $\pi/4$  verschoben
    output = arm_sin_q15(help & (~0x8000)); //Sinuswert als Look-
                                           Up-Table holen
    Audio_OUT[0] = output;             //Audio-OUT setzen
}
```



Um die Phasenverschiebung des generierten Sinus zu veranschaulichen wurden beide Sinussignale am Oszilloskop dargestellt.



Auf dem Bild ist eine Phasenverschiebung von $164\mu\text{s}$ abzulesen. Die Berechnete Verschiebung von $\pi/4$ bei einer Frequenz von ca. 760Hz ist genau so groß wie die gemessene $164\mu\text{s}$.

4. Erkenntnisse

Während der Übung und in der Vorbereitung darauf wurden theoretische Kenntnisse über ein DSPs und die gesamte Kette der digitalen Signalverarbeitung erworben und praktisch umgesetzt.

Weiters wurden die verschiedenen Zahlenformate, im Besonderen das q15-Format wiederholt und angewandt.