

HÖHERE TECHNISCHE BUNDESLEHRANSTALT HOLLABRUNN

Höhere Abteilung für Elektronik – Technische Informatik

Klasse/ Jahrgang: 5BHEL	Gruppe: 1 / b	Übungsleiter: Prof. Reisinger
Übungsnummer: II / 4	Übungstitel: Digitale Regler	
Datum der Übung: 01.12.15	Teilnehmer: Dorner, Mötz, Oberhamberger, Prinz	
Datum der Abgabe: 22.12.2015	Schritfführer: 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: Digitale Regler

Übungsnummer: II / 4

Übungsplatz: T10

Datum der Übung: 01.12.2015

Klasse: 5BHEL

Schritfführer: Dorner Fabian

Übungsteilnehmer: Dorner Fabian

Mötz Florian

Oberhamberger Christoph

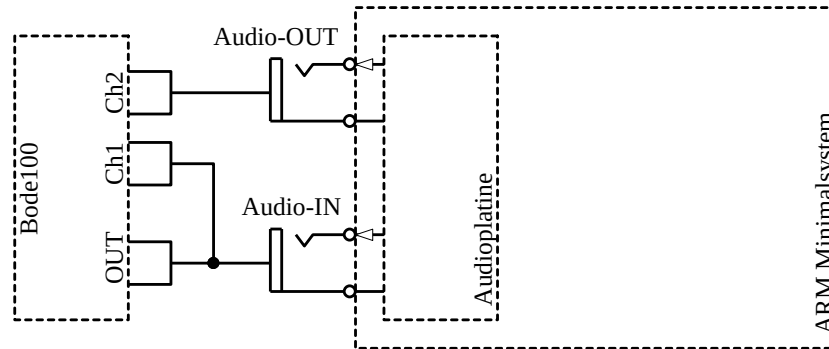
Prinz Bernhard

Inhaltsverzeichnis

1. Messaufbau	3
2. Testen der DSP	4
3. Aufgabenstellung	6
4. FIR-Filter	7
4.1. Design in Matlab	7
4.2. Filter implementieren	9
4.3. Übertragungsfunktion messen	10
4.4. ISR-Zeit-Messung	12
5. IIR-Filter	14
5.1. Design in Matlab	14
5.2. Filter implementieren	15
5.3. Übertragungsfunktion messen	17
5.4. ISR-Zeit-Messung	17
6. Erkenntnisse	18
Messprotokoll	19
Inventarliste	24

1. Messaufbau

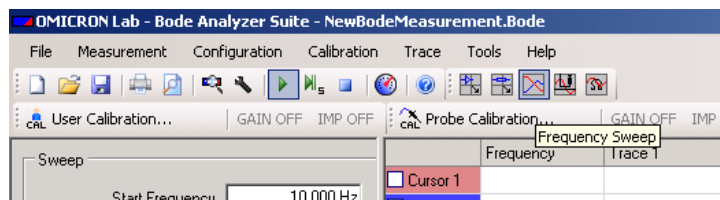
Der Messaufbau bestand aus einem ARM-Minimalsystem auf dem eine Audio-Platine mit Input und Output verbaut war. Die Ausgangssignale (siehe Punkt 4. und 5.) wurden mit einem Netzwerkanalysator (P1) gemessen.



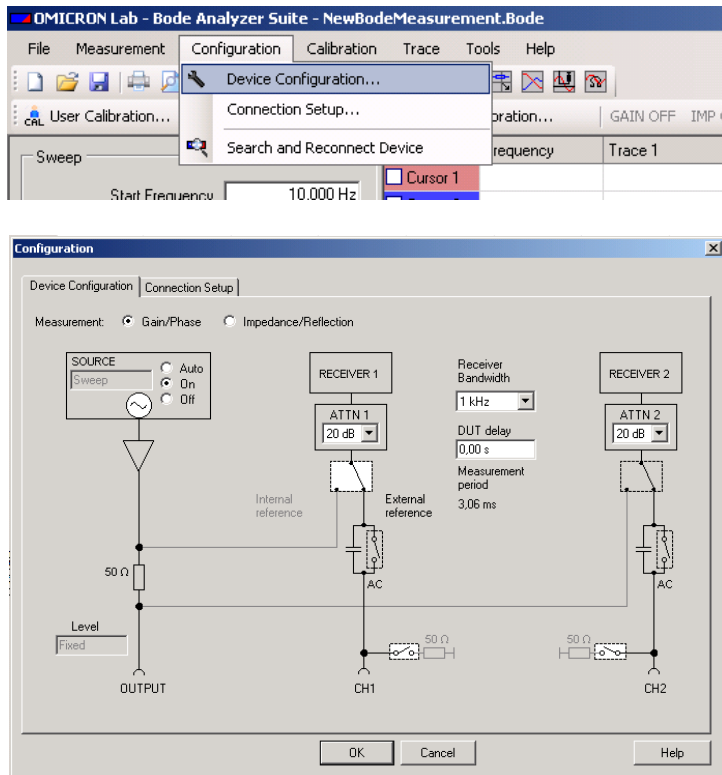
Die Messungen wurden, über eine serielle Verbindung mit dem PC, digital abgespeichert. Dazu wurde der „Bode Analyzer Suite“ von OMICRON Lab verwendet.

Bevor jedoch Messungen durchgeführt werden konnten wurden folgende Einstellungen vorgenommen:

- Frequency Sweep auswählen

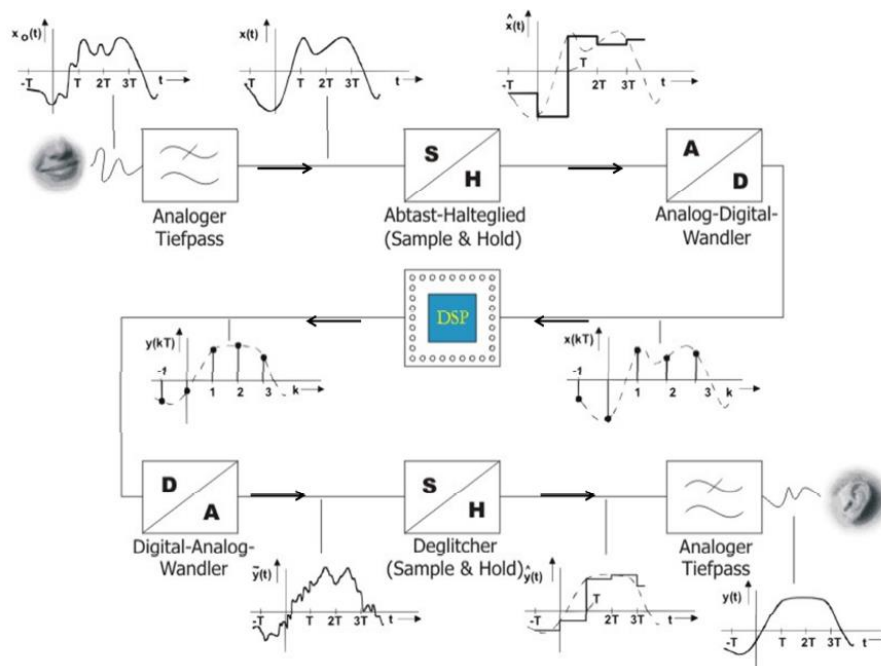


- Configuration → Device Configuration ... → Auswahl von „External reference“ für „RECEIVER 1“

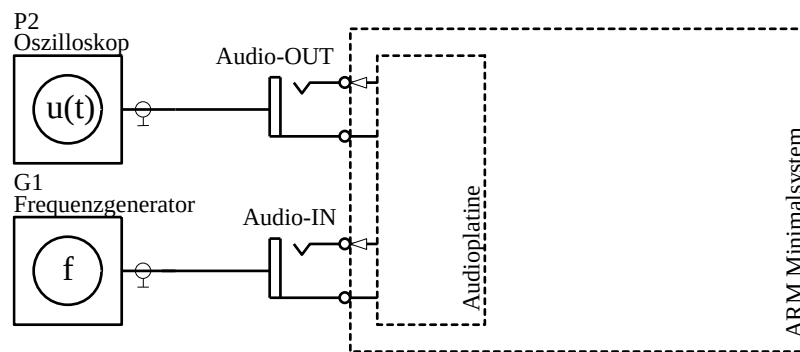


2. Testen des DSP

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 (Digital Signal Processor), ein DA-Wandler, ein Deglitcher und ein Rekonstruktions-TP.



Der Aufbau dazu sah wie folgt aus:



Getestet wurde diese Kette mittels folgendem Programm:

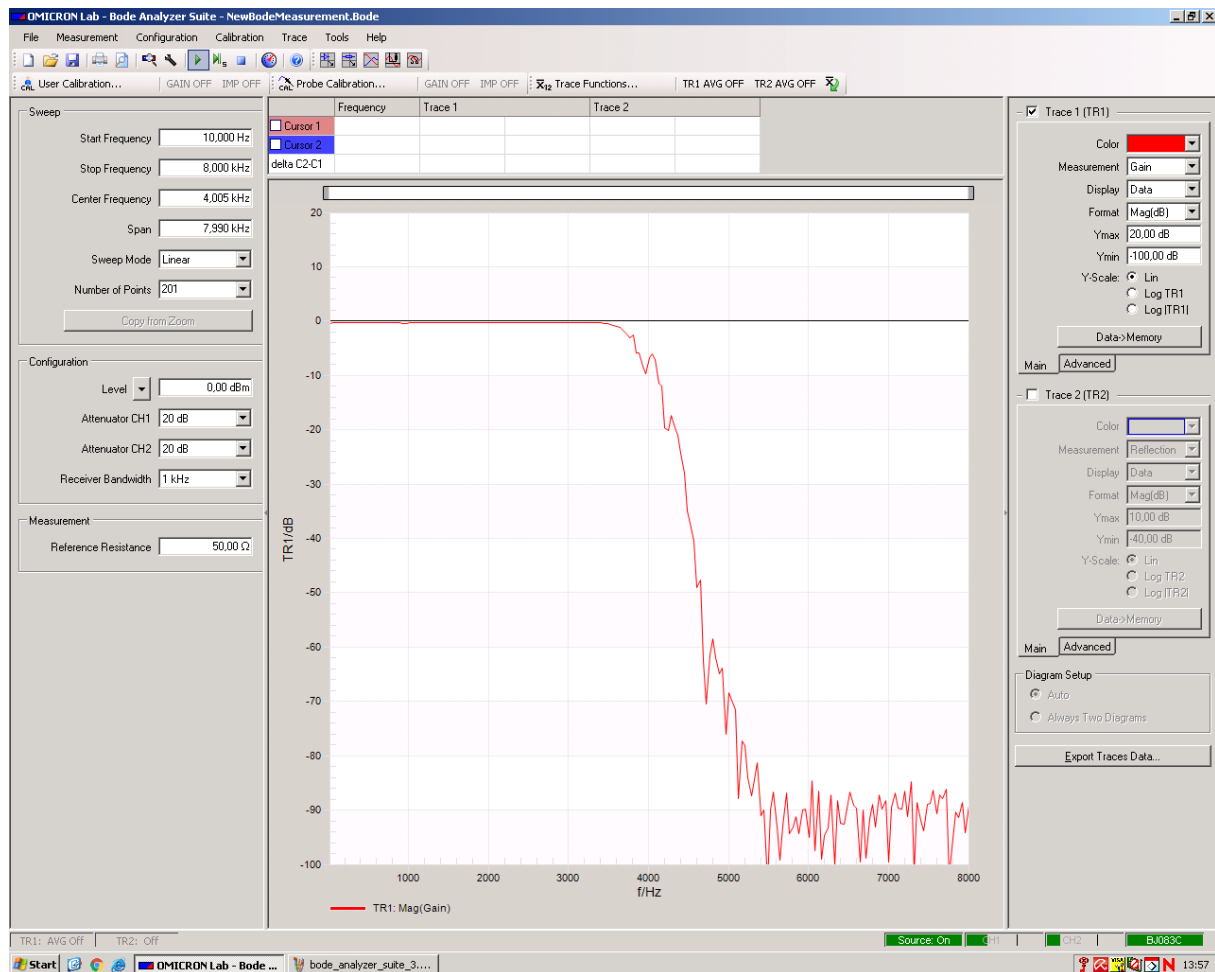
```
void process_data(void)
{
    //Audiowerte in eine hilfsvARIABLE speichern fuer filter-Upro
    in_1=Audio_IN[0];
    in_2=Audio_IN[1];

    //Ausgang=Eingang, Durchverbindung
    out_1 = in_1;
    out_2 = in_2;

    //bearbeiteten Werte auf die DMA2 Adresse spielen
    Audio_OUT[0] = out_1;
    Audio_OUT[1] = out_2;
}
```

Am Audio-Eingang wurde wie der Aufbau zeigt ein Signal eingespeist, welches mittels dem angeführten Programmcode 1:1 durch die DSP geschleust wurde und am Audio-Ausgang mit dem Oszilloskop gemessen und auf Richtigkeit überprüft wurde.

Weiters wurde mit dem Bode100 die Übertragungsfunktion der 1:1-Durchverbindung gemessen. Dazu wurde die Audioplatine mit dem Messgerät (P1), wie unter Punkt 1. angeführt, verbunden. Die Start-Frequenz wurde mit 10Hz und die Stop-Frequenz mit 8kHz gewählt.



Bei 4kHz ist zu erkennen, dass das Schenontheorem zu wirken beginnt. Alle Werte über 4kHz sind unterabgetastet.

3. Aufgabenstellung

Aufgabe dieser Übung war es ein FIR (finite impulse response) –Filter und ein IIR (infinite impulse response) –Filter mittels Matlab zu designen, die Filterkoeffizienten daraus in ein C-Header-File zu exportieren und in der DSP zu implementieren.

Weiters sollte die Übertragungsfunktion des jeweiligen Filters gemessen werden und mit dem Bode100 gemessen und mit jenen aus Matlab verglichen werden.

Die Filter sollten beide eine Grenzfrequenz von $f_g=910\text{Hz}$ haben und das FIR-Filter sollte ein Filter 4.Ordnung und das IIR-Filter eines 2.Ordnung sein.

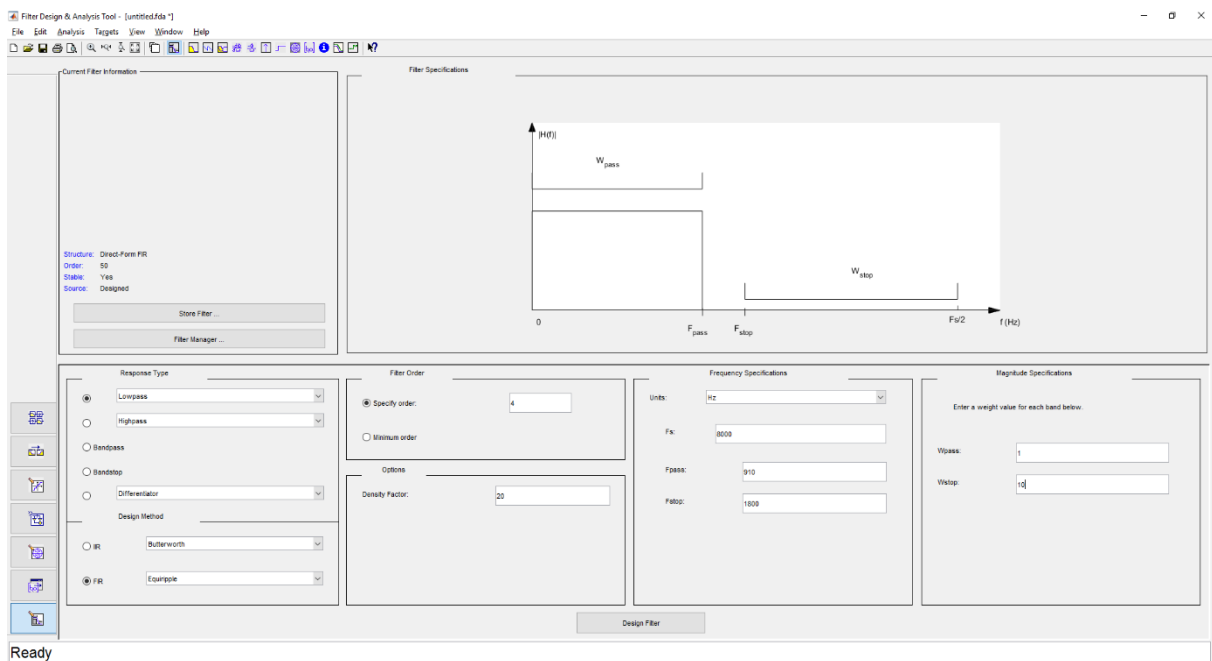
Zuletzt sollte noch die Dauer der jeweiligen ISR (Interrupt Service Routine) bestimmt werden.

4. FIR-Filter

4.1. Design in Matlab

Um ein Filter in Matlab designen zu können muss zuerst das FDA-Tool gestartet werden. Dies ist möglich durch die Auswahl von „Filter Design & Analysis“ unter „APPS“ oder durch die Eingabe des Befehls `fdatool`.

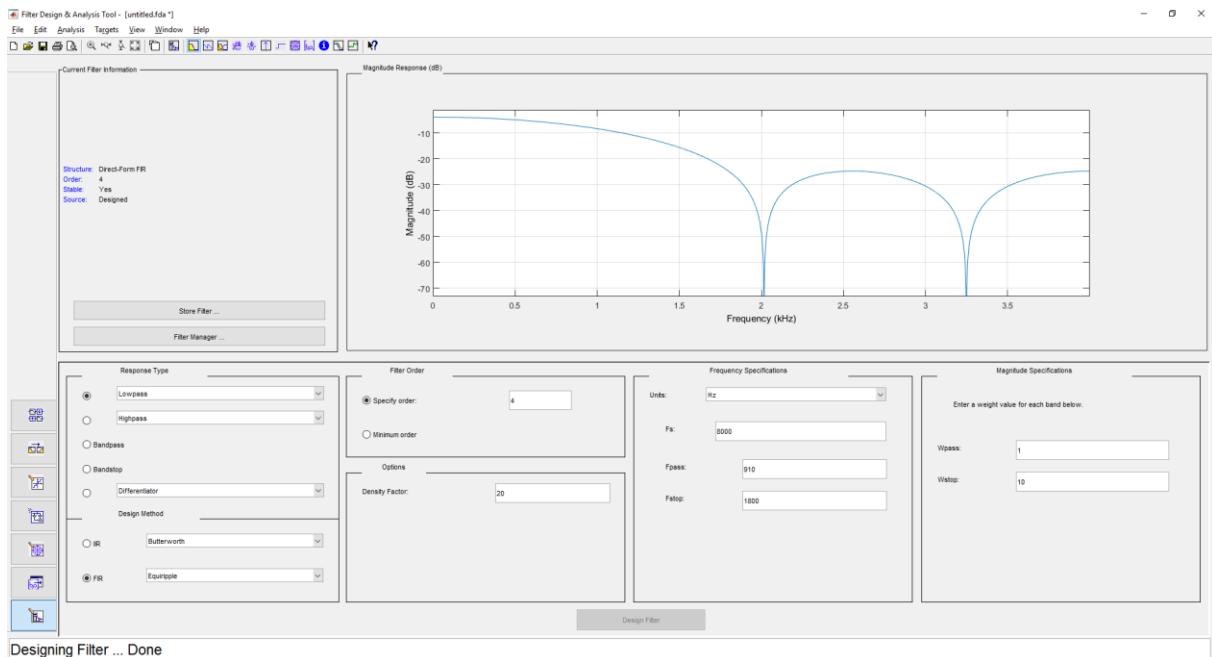
Nachdem das FDA-Tool gestartet wurde können die Spezifikationen des Filters eingetragen werden. Das folgende Bild und die Auflistung zeigen welche Eingaben vorgenommen wurden.



Änderungen:

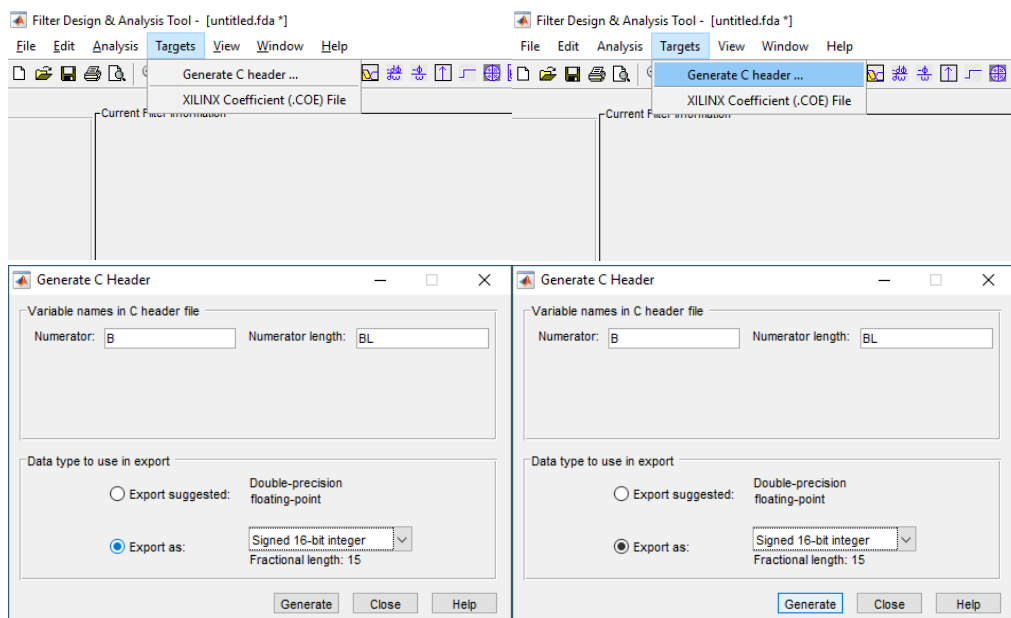
- Design Method:
 - FIR: Equiripple
- Filter Order:
 - Specify order: 4
- Frequency Specifications:
 - Units: Hz
 - Fs: 8000
 - Fpass: 910
 - Fstop: 1800
- Magnitude Specification:
 - Wstop: 10 (Verhältnis von Wstop:Wpass legt das Verhältnis von Sperr- zu Durchlassdämpfung fest)

Nachdem diese Änderungen vorgenommen wurden, wurde das Filter mit einem Klick auf „Design Filter“ berechnet.



Danach konnte der C-Header mit den Filterkoeffizienten generiert werden, dazu wurde wie folgt vorgegangen:

Targets → Generate C header ... → Export as: “Signed 16-bit integer” → Generate



4.2. Filter implementieren

Als nächster Schritt wurde das Implementieren des FIR-Filters in der DSP durchgeführt.

Um die Koeffizienten des generierten Header-Files nun auch nutzen zu können muss der Datentyp dieser von `int` auf `q15_t` geändert werden. Weiters wurde noch die Zeile auskommentiert in welcher der Header `tmwtypes.h` inkludiert wird.

Der Header sieht dann wie folgt aus:

```
/*
 * Discrete-Time FIR Filter (real)
 * -----
 * Filter Structure      : Direct-Form FIR
 * Filter Length        : 5
 * Stable                : Yes
 * Linear Phase          : Yes (Type 1)
 */

/* General type conversion for MATLAB generated C-code */
// #include "tmwtypes.h"
/*
 * Expected path to tmwtypes.h
 * C:\Program Files\MATLAB\R2015b\extern\include\tmwtypes.h
 */
/*
 * Warning - Filter coefficients were truncated to fit specified data
 * type.
 * The resulting response may not match generated theoretical
 * response.
 * Use the Filter Design & Analysis Tool to design accurate
 * int16 filter coefficients.
 */
const int BL = 5;
const q15_t B[5] = {
    2836,  4784,  5792,  4784,  2836
};
```

Nun kann der C-Header einfach mit der Pre-Prozessoranweisung `include` in das C-File eingebunden werden und die Koeffizienten zu Berechnung verwendet werden.

Das C-File sah dann folgendermaßen aus:

```
void process_data(void)
{
    static q15_t buf [BL] = {0};
    int i;
    q15_t obuf;
    q15_t eobuf = 0;

    // Shift the old values by 1 step
    // buf[4] = buf[3] etc.
    for (i=BL-1; i>0; i--){
        buf[i] = buf[i-1];
    }
}
```

```

// Set new input value
buf[0] = Audio_IN[0];

// out_1 = buf[0]*B[0] + buf[1]*B[1] ... + buf[BL]*B[BL];
for (i=0; i<BL; i++){
    arm_mult_q15(&buf[i] , &B[i], &obuf, 1);
    arm_add_q15(&obuf, &eobuf, &eobuf, 1);
}
out_1 = eobuf;

// bearbeiteten Werte auf die DMA2 Adresse spielen
Audio_OUT[0] = out_1;
Audio_OUT[1] = out_1;
}

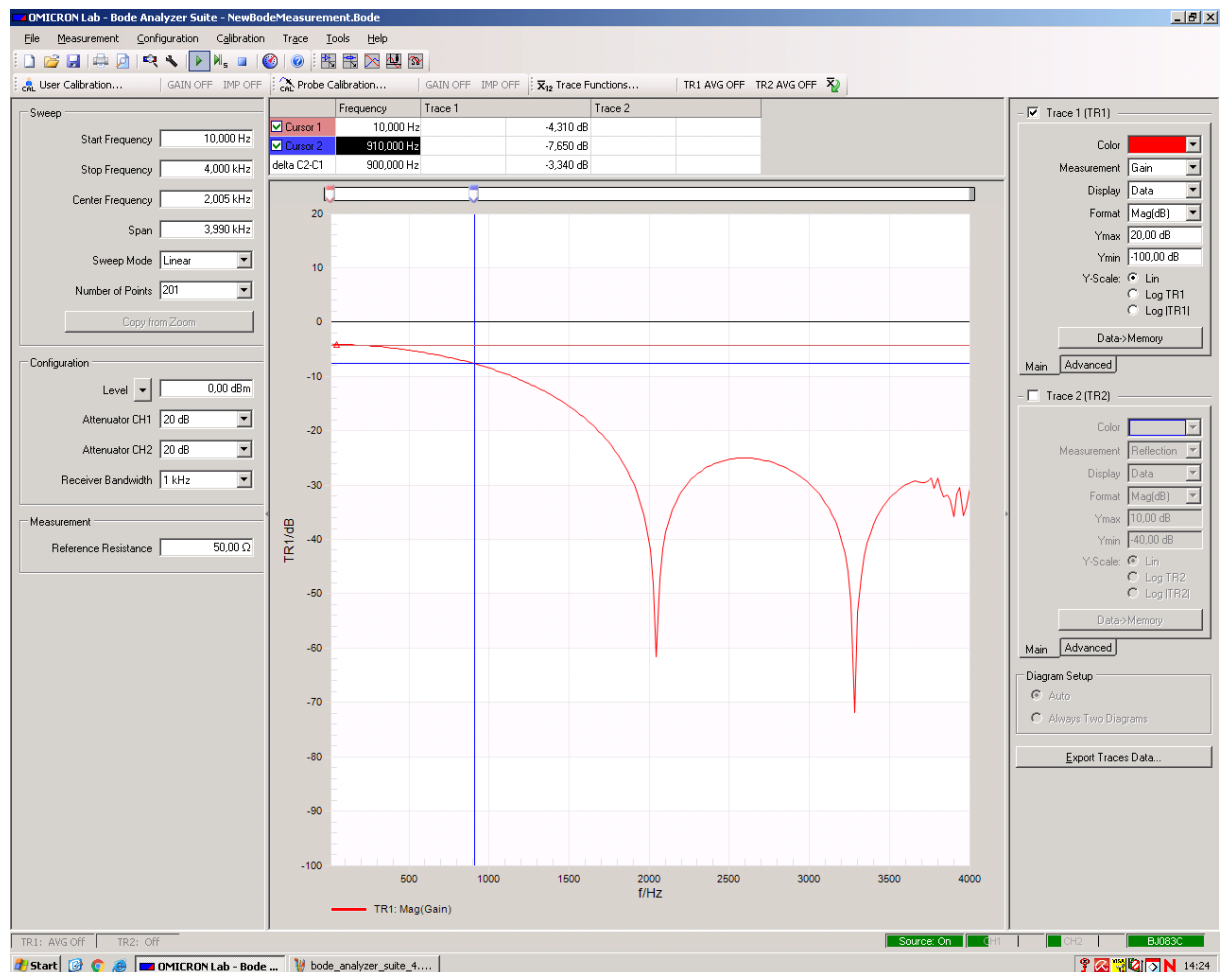
```

Dieser Code wurde kompiliert und auf das ARM-System geflasht, danach wurde mit Punkt 4.3. fortgesetzt.

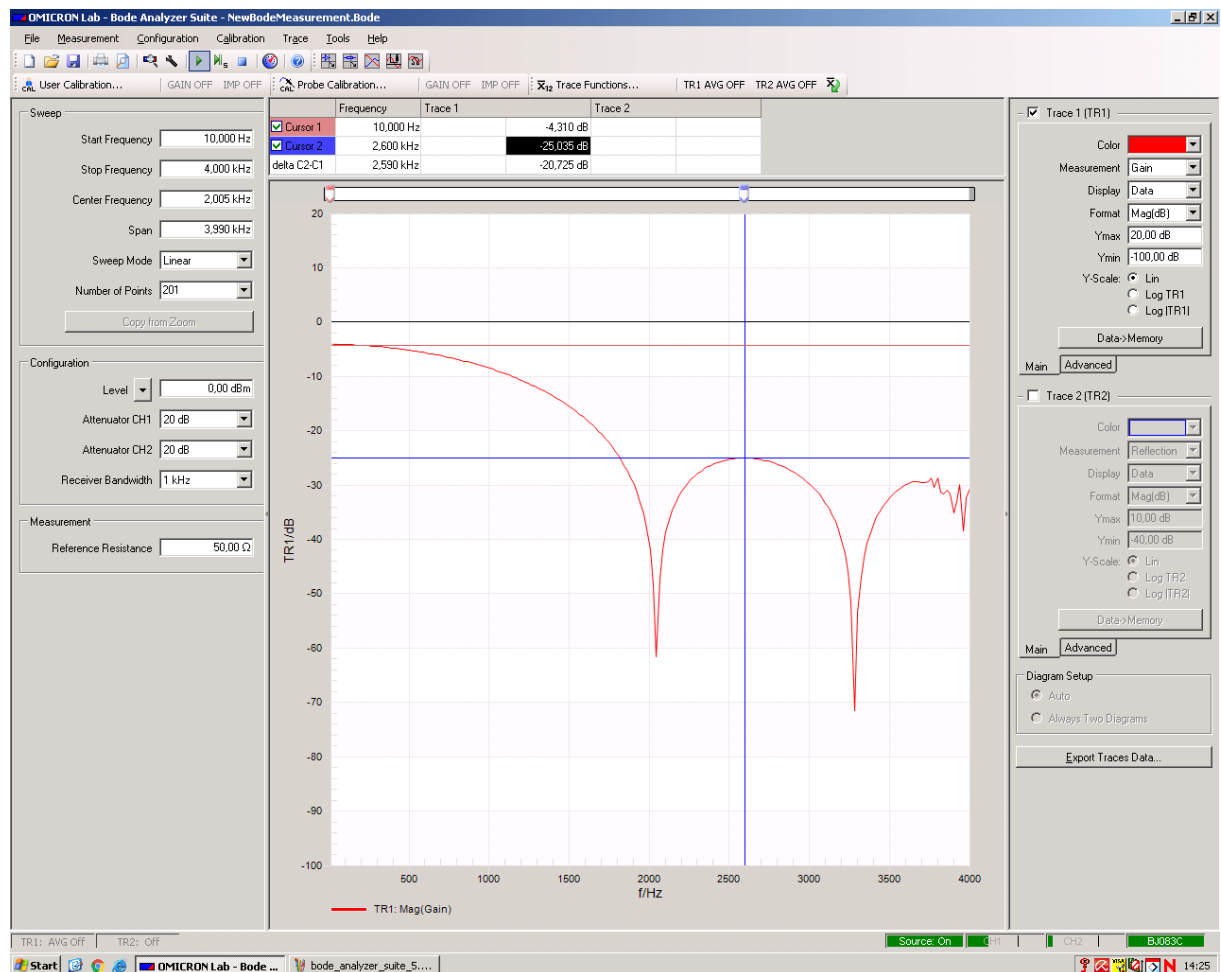
4.3. Übertragungsfunktion messen

Zu Messung der Übertragungsfunktion wurde die Start-Frequenz von 10Hz beibehalten und die Stop-Frequenz auf 4kHz heruntergesetzt.

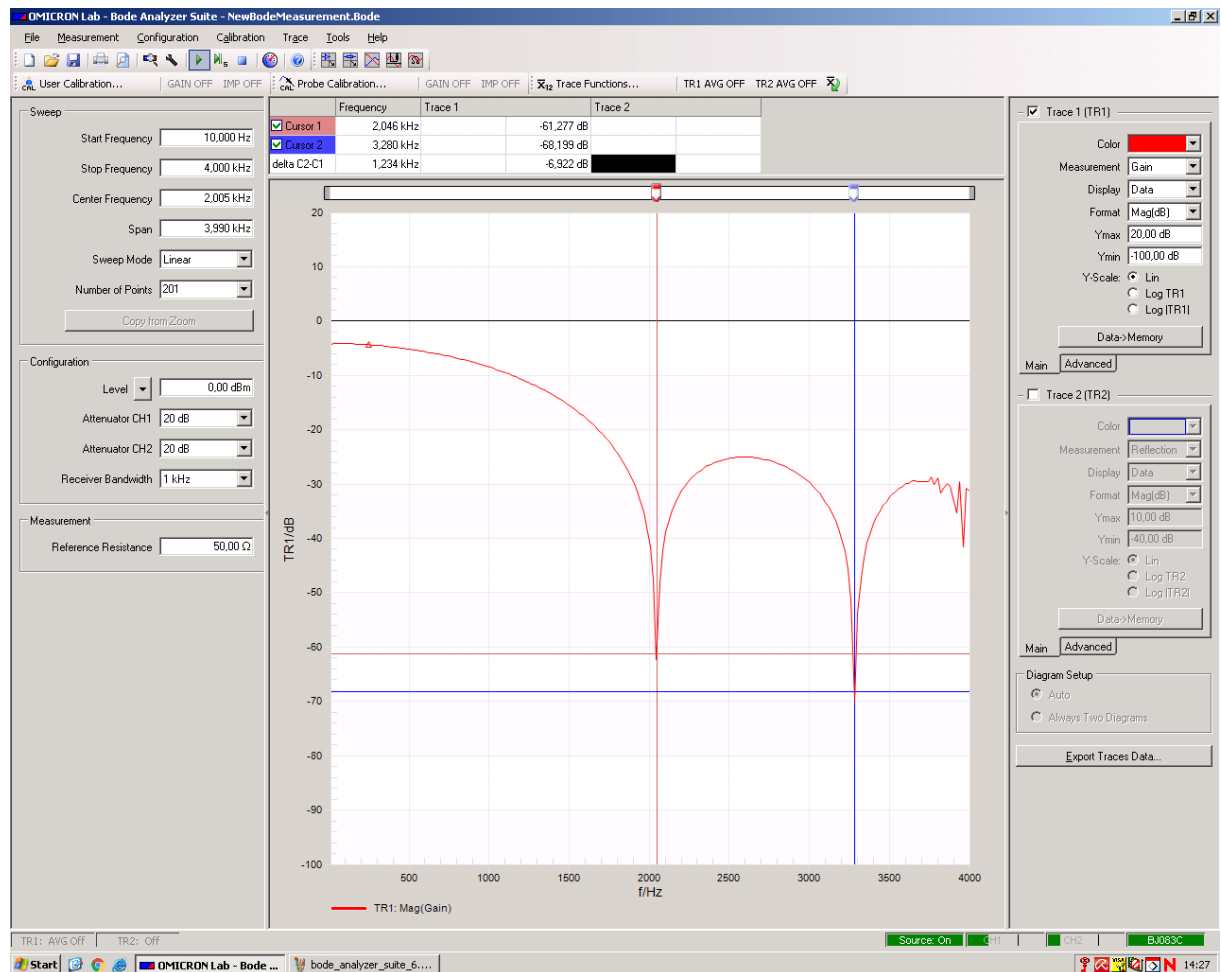
Es ergab sich folgende Übertragungsfunktion:



Wie der Messung mittels Cursors zu entnehmen ist, liegt die Dämpfung im Durchlassbereich bei -4,31dB und bei der Grenzfrequenz f_g bei -7,65dB.



Als nächstes wurde die Dämpfung im Sperrbereich gemessen welche -25,035dB betrug.



Zuletzt wurden noch die Polstellen bestimmt welche sich bei 2,046kHz und 3,28 kHz befanden.

Der Vergleich mit der Simulation in Matlab ergab keine größeren Unterschiede.

4.4. ISR-Zeit-Messung

Um die Dauer der ISR bestimmen zu können wurde im Programmcode am Beginn der ISR eine LED eingeschalten und am Ende wieder aus. Die Dauer während die LED leuchtet ist also die Dauer der ISR.

Das Programmlisting dazu sieht wie folgt aus:

```
void process_data(void)
{
    static q15_t buf [BL] = {0};
    int i;
    q15_t obuf;
    q15_t eobuf = 0;
    LED0 = 1;

    // Shift the old values by 1 step
    // buf[4] = buf[3] etc.
    for (i=BL-1; i>0; i--){
        buf[i] = buf[i-1];
    }
}
```

```

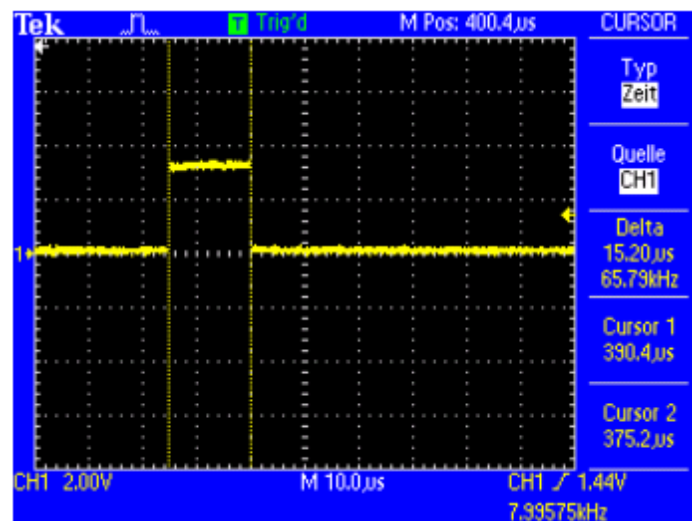
// Set new input value
buf[0] = Audio_IN[0];

// out_1 = buf[0]*B[0] + buf[1]*B[1] ... + buf[BL]*B[BL];
for (i=0; i<BL; i++){
    arm_mult_q15(&buf[i] , &B[i], &obuf, 1);
    arm_add_q15(&obuf, &eobuf, &eobuf, 1);
}
out_1 = eobuf;

//bearbeiteten Werte auf die DMA2 Adresse spielen
Audio_OUT[0] = out_1;
Audio_OUT[1] = out_1;
LED0 = 0;
}

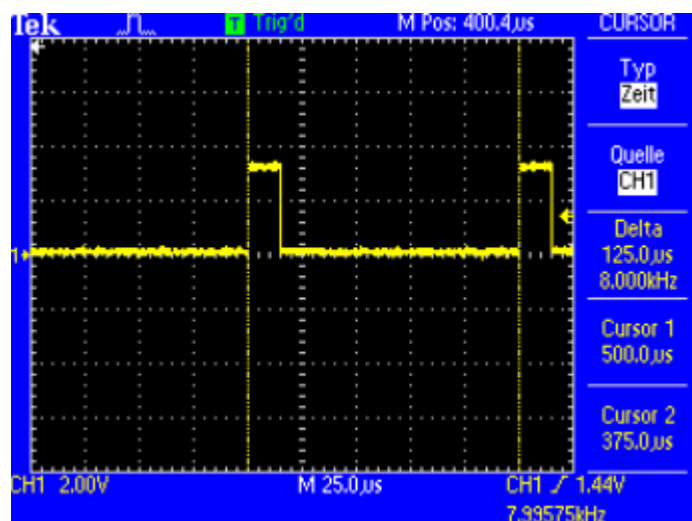
```

Gemessen wurde dies mittels Oszilloskop. Der Tastkopf des CH1 wurde an die LED gehängt und es wurde folgendes gemessen:



Die Messung zeigt, dass die Dauer der ISR 15,2 µs war.

Weiters wurde gleich die Zeitspanne zwischen zwei ISR-Aufrufen mit gemessen.



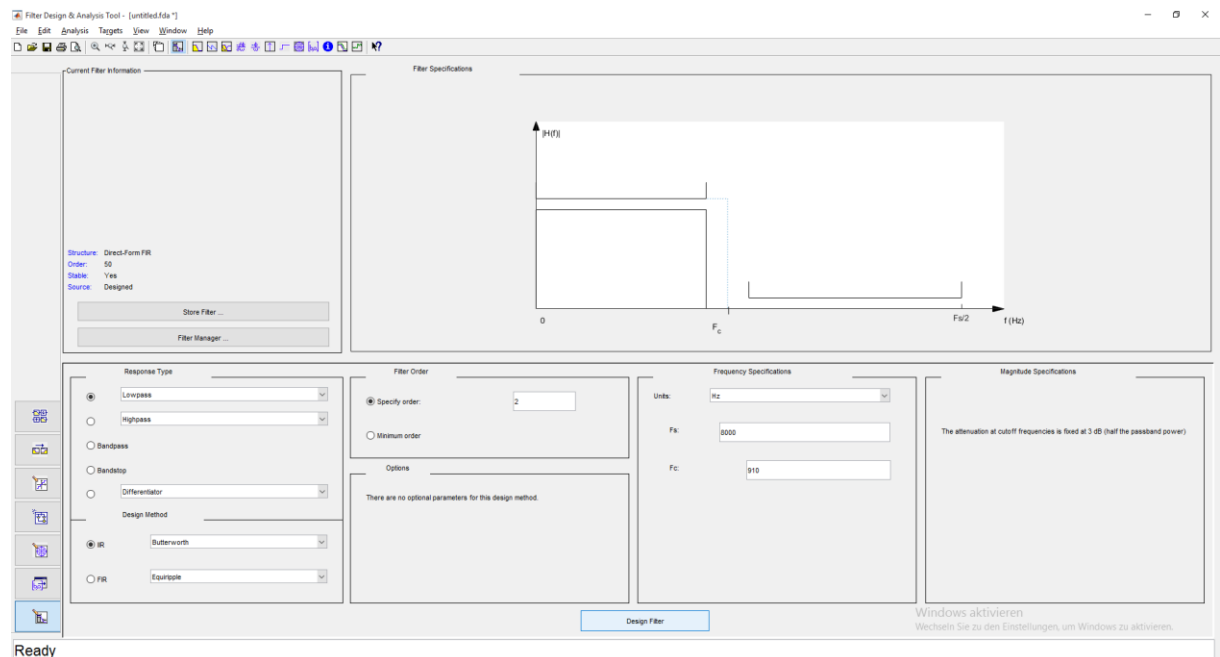
Wie die Messung zeigt ist die Zeitspanne genau $125\mu\text{s}$ lang. Dieses Ergebnis war zu erwarten da die Abtastfrequenz $f_A = 8\text{kHz}$ war.

5. IIR-Filter

5.1. Design in Matlab

Um das Filter zu designen wurde zuerst das FDA-Tool von Matlab gestartet.

Danach wurden die Einstellungen für das IIR-Filter vorgenommen, welche in dem nachfolgendem Bild und der Auflistung zu sehen sind.

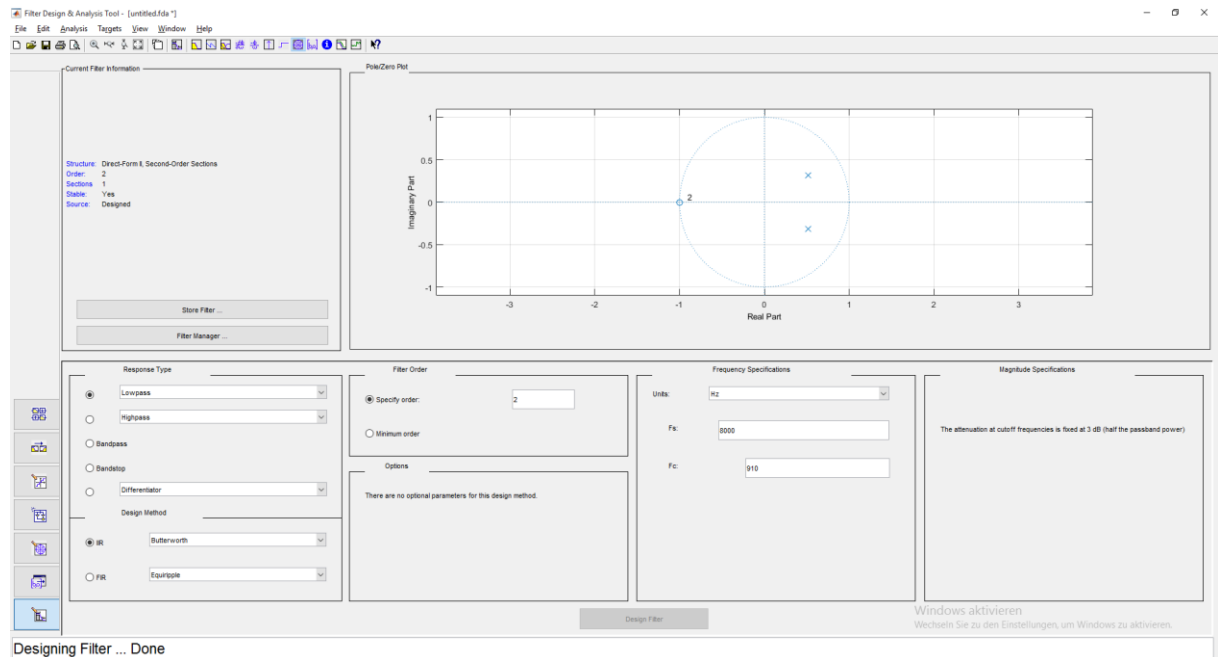


Änderungen:

- Design Method:
 - IIR: Butterworth
- Filter Order:
 - Specify order: 2
- Frequency Specifications:
 - Units: Hz
 - Fs: 8000
 - Fc: 910

Nachdem die Änderungen durchgeführt wurden wurde auf „Design-Filter“ geklickt, um die Berechnung zu starten.

Nach der Berechnung konnte man sich das Pol-Nullstellendiagramm ansehen, welches in diesem Fall eine Doppelnullstelle und 2 Polstellen zeigte.



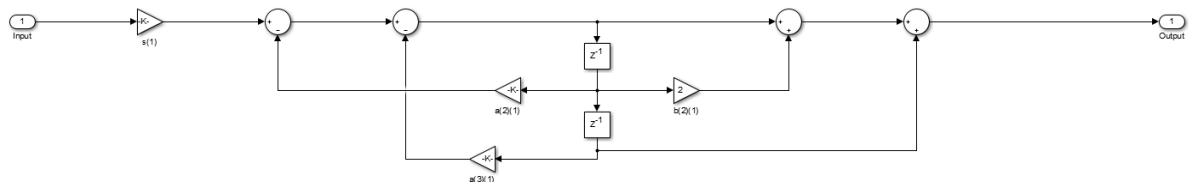
5.2. Filter implementieren

Als nächster Schritt wurde das Implementieren des IIR-Filters in der DSP durchgeführt.

Nachdem die Pol- und Nullstellen kontrolliert wurden, wurden wie bei Punkt 4.2. die Koeffizienten in ein C-Header-File exportiert.

Nun ist nur noch der Aufbau des Filters notwendig um es in die DSP zu implementieren. Dazu wurde in dem noch geöffneten FDA-Tool unter dem Punkt „Realize Model“ in dem Block mit der Überschrift „Model“ das Häkchen bei „Build model using basic elements“ gesetzt und auf den Button „Realize Model“ geklickt.

Daraufhin öffnete sich ein neues Fenster mit einem Block darin. Per Doppelklick auf den Block wird in den Block gesprungen und der Aufbau des IIR-Filters sichtbar.



Nun kann gemeinsam mit den Koeffizienten aus dem Header und dem Aufbau der Code für den DSP geschrieben werden.

Dieser sieht wie folgt aus:

```
void process_data(void)
{
    static q15_t ZERO = {0};
    static q15_t buf [2] = {0};
    q15_t in, subbuf, bufsave;

    // Calculate the factors (to q15 from float)
    if (0 == s){
        arm_float_to_q15( &af[0],    &a[0],    1);
        arm_float_to_q15( &af[1],    &a[1],    1);
        arm_float_to_q15( &b,        &b,        1);
        arm_float_to_q15( &s,        &s,        1);
    }

    // Read input
    in = Audio_IN[0];

    // Verstärken
    arm_mult_q15(&in,          &s,          &in,          1);

    arm_sub_q15( &ZERO,        &buf[0],    &subbuf,    1);
    arm_mult_q15(&subbuf,      &a[0],    &subbuf,    1);
    arm_add_q15( &in,          &subbuf,    &in,          1);

    arm_sub_q15( &ZERO,        &buf[1],    &subbuf,    1);
    arm_mult_q15(&subbuf,      &a[1],    &subbuf,    1);
    arm_add_q15( &in,          &subbuf,    &in,          1);

    // Save for shifting later
    bufsave = in;

    arm_mult_q15(&buf[0], &b, &subbuf, 1);
    arm_add_q15( &subbuf, &in, &in, 1);

    arm_add_q15(&in, &buf[1], &in, 1);

    // Multiply with 4 to correct gain losses with the equivalent
    factors
    in = in << 2;

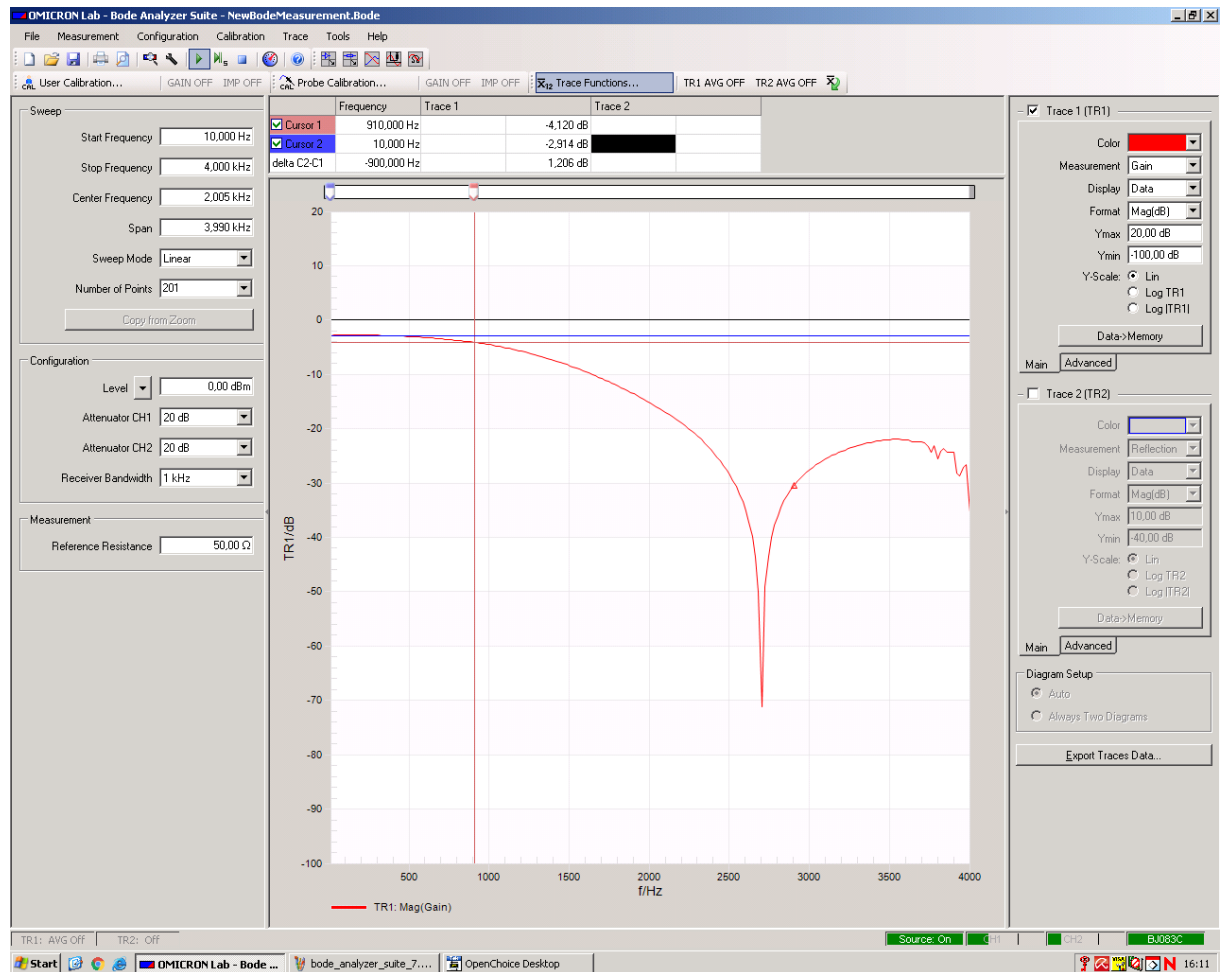
    Audio_OUT[0] = in;
    Audio_OUT[1] = in;

    // Shift and save
    buf[1] = buf[0];
    buf[0] = bufsave;
}
```

Der fertige Code wurde kompiliert und auf das ARM-System geflasht.

5.3. Übertragungsfunktion messen

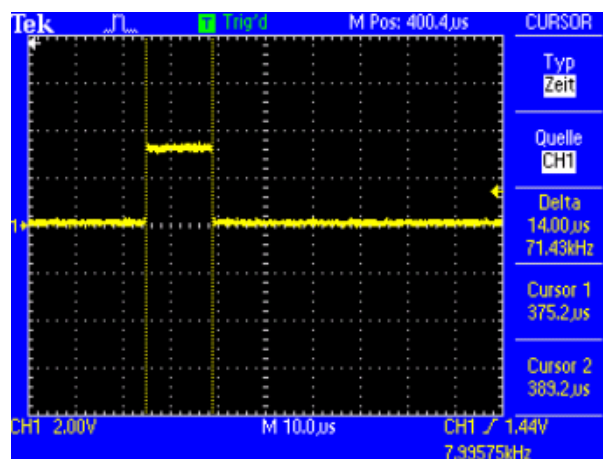
Weiters wurde die Übertragungsfunktion mit dem Bode100 gemessen.



Bei dieser Messung ergab sich die Dämpfung im Durchlassbereich zu -4,12dB und die Dämpfung bei der Grenzfrequenz f_g betrug -2,914dB.

5.4. ISR-Zeit-Messung

Wie bereits bei Punkt 4.4. erfolgt die Messung der Dauer der ISR auch über das Ein- und Ausschalten einer LED am Beginn und am Ende der ISR.



Die Messung ergab eine Dauer von $14\mu\text{s}$ für die ISR. Die Zeitpanne zwischen zwei ISR Aufrufen ist genauso wie bei Punkt 4.4. durch die Abtastfrequenz f_A vorgegeben und beträgt $125\mu\text{s}$.

6. Erkenntnisse

Während der Übung und in der Vorbereitung darauf wurden theoretische Kenntnisse über FIR- und IIR-Filter, sowie über das FDA-Tool von Matlab wiederholt, vertieft und praktisch angewandt.

Weiters wurden die Kette der digitalen Signalverarbeitung und das q15-Format wiederholt und angewandt.