

1 Zusammenfassung

In unserer Studienarbeit ging es um die Realisierung eines einstellbaren IIR-Filters. Auf dem PC läuft eine von uns entwickelte Windows Applikation, welche die IIR-Filterkoeffizienten berechnet. Es können 2 Filterapproximationen (Butterworth und Chebyshev 1) bis 10. Ordnung gewählt werden. Zudem ist wählbar, ob ein Hochpass oder ein Tiefpass realisiert werden soll. Die Grenzfrequenz des Filters ist frei einstellbar zwischen 0 und der halben Samplingfrequenz. Bei der Chebyshev 1 Filterapproximation ist zudem noch der Passband-Rippel einstellbar. Es besteht die Möglichkeit 4 verschiedene Samplingfrequenzen (48 kHz, 32 kHz, 16 kHz, 8 kHz) für den DA/AD-Wandler zu wählen.

Die Software auf dem PC kommuniziert über die RS 232 Schnittstelle mit dem DSP 56002 evaluation Module. Auf dem Modul ist das IIR-Filter realisiert. Es ist möglich während des Filtervorganges die neuen Filterkoeffizienten vom PC herunterzuladen, der DSP verwendet anschliessend automatisch die neuen Filterkoeffizienten. Es besteht auch die Möglichkeit, die Auslastung des DSP's auf dem PC zu betrachten.

2 Inhaltsverzeichnis

1	ZUSAMMENFASSUNG	1
2	INHALTSVERZEICHNIS	2
3	IIR-FILTERDESIGNER ANFORDERUNGS-SPEZIFIKATION	4
3.1	AUFGABENSTELLUNG:	4
3.2	SOFTWARE-SCHNITTSTELLEN:	4
3.3	HARDWARE-SCHNITTSTELLE:	4
3.4	ZIELE DER SEMESTERARBEIT:	5
3.5	TERMINE:	5
3.6	ZEITPLAN:	5
4	EINLEITUNG	7
4.1	WARUM DIGITALE FILTER	7
4.2	AUFBAU EINES DIGITALEN FILTER	7
5	THEORETISCHE GRUNDLAGEN	8
5.1	DIGITALE SIGNALVERARBEITUNG	8
5.1.1	<i>Die Differenzgleichung</i>	8
5.1.2	<i>Die Z-Transformation</i>	8
5.1.3	<i>Die bilineare Transformation</i>	9
5.2	GRUNDLEGENDE FILTERTHEORIE	10
5.3	IIR-FILTER	11
5.3.1	<i>Theorie</i>	11
5.3.2	<i>Designflow (Entwicklung eines IIR-Filters):</i>	13
6	LÖSUNG DES PROBLEMS	14
6.1	LÖSUNGSANSATZ	14
6.2	LÖSUNGSWEG	14
6.2.1	<i>Berechnung eines Polpaars für ein Chebyshev Filter Typ 1</i>	14
6.2.2	<i>Berechnung eines Polpaars für ein Butterworth Filter</i>	16
6.2.3	<i>Transformation eines Polpaares in die Z-Ebene</i>	16
6.2.4	<i>Transformation eines Tiefpasspolpaares in ein Hochpasspolpaar und anschliessende Transformation in die Z-Ebene</i>	17
7	ÜBERPRÜFUNG DER IMPLEMENTATION MIT MATHEMATICA	18
7.1	CHEBYSHEV TIEFPASS	18
7.2	CHEBYSHEV HOCHPASS	20
7.3	BUTTERWORTH TIEFPASS	21
7.4	BUTTERWORTH HOCHPASS	22
8	IMPLEMENTATION AUF DEM PC	24
8.1	ENTWICKLUNGSSYSTEM	24
8.2	GRUNDSTRUKTUR DER PC-SOFTWARE	24
8.3	ÜBERTRAGUNG	25
8.3.1	<i>Serielle Schnittstelle (COM)</i>	25
8.3.2	<i>Uebertragungsvorgang</i>	25
8.3.3	<i>Leistungsmonitor</i>	25
8.3.4	<i>Übertragungsprotokoll</i>	26
9	IMPLEMENTATION AUF DEM DSP	27
9.1	ENTWICKLUNGSSYSTEM	27
9.2	GRUNDSTRUKTUR DER DSP-SOFTWARE	27
9.2.1	<i>Grundfunktionen</i>	27
9.2.2	<i>Realisierungsart</i>	27
9.2.3	<i>Grobe Zusammenhänge der Teilprogramme</i>	28
9.3	DETAILSTRUKTUR DER DSP-SOFTWARE	29

9.3.1	Speicheraufteilung	29
9.3.2	Registeraufteilung	30
9.3.3	Hauptprogramm (main)	30
9.3.4	Initialisierungsfunktionen	31
9.3.5	Filterung (Interrupt Serviceroutine)	32
9.3.6	Leistungsmonitor(TimerISR)	32
9.3.7	ISR für das Senden über die asynchrone serielle Schnittstelle	32
9.3.8	Übertragung	33
10	MESSBERICHTE	36
10.1	MESSAUFBAU	36
10.2	AUSMESSUNG EINES IIR-FILTERS	36
10.2.1	Messresultate	37
10.2.2	Auswertung der Messung	41
10.2.3	Kommentar zur Messung	42
10.3	AUSMESSUNG DES LEISTUNGSMONITORS	43
10.3.1	Gemessenen Werte	43
10.3.2	Kommentar zur Messung, Auswertung	43
11	PROBLEME BEI DER REALISIERUNG	44
11.1	PROBLEME BEI DER KOEFFIZIENTENBERECHNUNG	44
11.2	PROBLEME UND SCHWIERIGKEITEN BEI DER IMPLEMENTATION DES GUI	44
11.3	PROBLEME UND SCHWIERIGKEITEN BEI DER IMPLEMENTATION DER ÜBERTRAGUNG	45
12	AUSBLICK UND VISIONEN FÜR DEN FILTERDESIGNER	46
12.1	MÖGLICHE ERWEITERUNGEN	46
12.2	GESAMT - KONZEPT	46
13	IIR-FILTERDESIGNER MANUAL	47
13.1	EINFÜHRUNG	47
13.2	BESCHREIBUNG DER EINZELNEN PARAMETER	47
14	RÜCKBLICK AUF DIE SEMESTERARBEIT	49
15	LITERATURLISTE	49
	C++ SOURCECODE	A
	ASSEMBLER SOURCECODE	B
	MATHEMATICA BERECHNUNGEN	C

3 IIR-FILTERDESIGNER Anforderungs-Spezifikation

3.1 Aufgabenstellung:

Es soll ein Zweikanal-IIR-Filter für den Audiofrequenzbereich mit dem digitalen Signalprozessor DSP56002 implementiert werden. Als Hardwareaufbau soll das Evaluationsmodul EVM56002 von Motorola verwendet werden.

Zusätzlich soll ein geeignetes User-Interface entwickelt werden, welches dem Benutzer erlaubt, folgende Parameter für jeden Kanal separat vorzugeben:

- Filtercharakteristik : Tiefpass oder Hochpass
- Filtertyp : Butterworth oder Chebyshev 1
- Abtastfrequenz (f_s) : Es kann zwischen den Samplingfrequenzen
 - 8000Hz
 - 16000Hz
 - 32000Hz
 - 48000Hz
 gewählt werden.
- Grenzfrequenz (f_g) : Die Grenzfrequenz ($1 \dots f_s/2$)
- Sperrfrequenz (f_{sperr}) : Die Sperrfrequenz ($1 \dots f_s/2$)
 (Beim Tiefpass-Filter, muss die Sperrfrequenz immer grösser sein als die Grenzfrequenz und beim Hochpass-Filter immer kleiner)
- Dämpfung : Die minimale Dämpfung im Sperrbereich (3...100dB) und beim Chebyshev 1-Filter zusätzlich die maximale Dämpfung im Durchlassbereich (1...10dB)
- Filterordnung : Anzahl kaskadierte Filter (1...10)

Aufgrund dieser vorgegebenen Parameter sollen die Filterkoeffizienten auf dem PC berechnet und auf den DSP heruntergeladen werden. Dort soll das IIR-Filter in Echtzeit betrieben werden.

3.2 Software-Schnittstellen:

Übergabe Parameter zwischen dem Graphischen-Userinterface und der Koeffizientenberechnung sind:

- f_s (8k,...48k)
- f_g ($0 \dots f_s/2$)
- Filterordnung (1..10)
- Bauweise des Filters (0 = butterworth, 1 = Chebyshev1)
- Filtercharakteristik (0 = Tiefpass, 1 = Hochpass)
- Dämpfung (Sperrbereich 3..100dB, Durchlassbereich 1..10dB)

3.3 Hardware-Schnittstelle:

Das Herunterladen der Koeffizienten geschieht entweder über das Programm EVM-Debugger oder durch direktes Ansprechen der seriellen Schnittstelle auf dem EVM-Board durch den DSP.

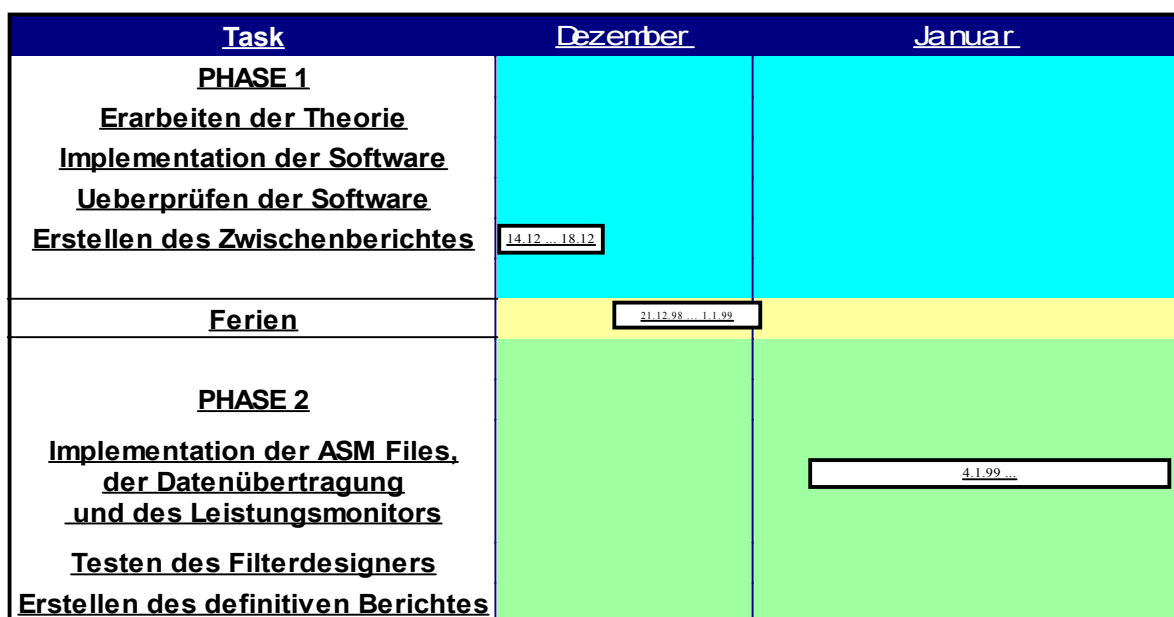
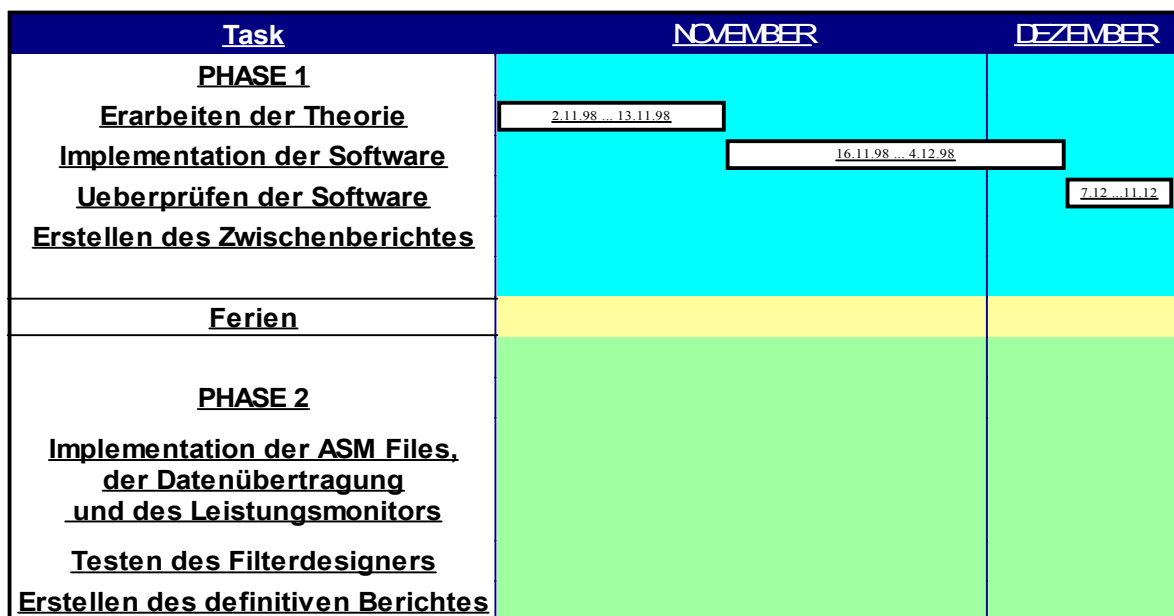
3.4 Ziele der Semesterarbeit:

Das Ziel dieser Semesterarbeit ist es, einerseits einen funktionierenden Prototypen zu implementieren und andererseits sollen Erfahrungen mit der Programmierung von digitalen Signalprozessoren gesammelt werden. Dazu sollen auch quantitative Aussagen (Benchmarks) gemacht, sämtliche Resultate diskutiert und die gesamte Arbeit umfassend dokumentiert werden.

3.5 Termine:

Ausgabe der Aufgabenstellung : 02.11.1998
 Einarbeitung in die Materie : 02.11. - 18.12.1998
 Abgabe Zwischenbericht : 18.12.1998
 Praktischer Teil : 04.01. - 26.02.1999

3.6 Zeitplan:



Task	FEBRUAR
PHASE 1 <u>Erarbeiten der Theorie</u> <u>Implementation der Software</u> <u>Ueberprüfen der Software</u> <u>Erstellen des Zwischenberichtes</u>	
<u>Ferien</u>	
PHASE 2 <u>Implementation der ASM Files,</u> <u>der Datenübertragung</u> <u>und des Leistungsmonitors</u> <u>Testen des Filterdesigners</u> <u>Erstellen des definitlven Berichtes</u>	<div style="border: 1px solid black; width: 100px; height: 15px; margin-bottom: 10px; position: relative;"> 12.2-9% </div> <div style="border: 1px solid black; width: 100px; height: 15px; margin-bottom: 10px; position: relative;"> 15.2-19.2 </div> <div style="border: 1px solid black; width: 100px; height: 15px; position: relative;"> 22.2-26.2 </div>

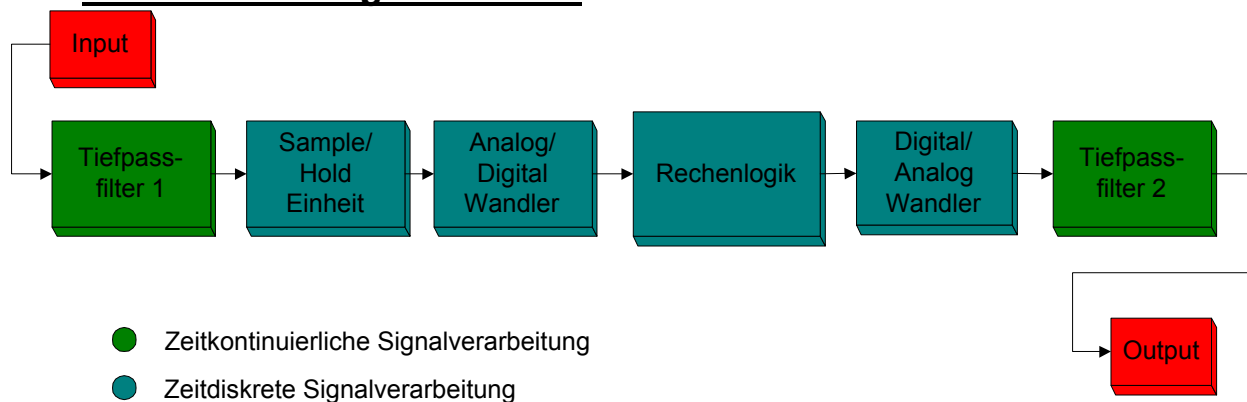
4 Einleitung

4.1 Warum digitale Filter

Digitale Filter sind heutzutage weit verbreitet und werden immer häufiger anstelle von analogen Filter verwendet. Es gibt verschiedene Gründe die für die Verwendung von digitalen Filtern sprechen. Einer der Hauptgründe ist, dass es heutzutage in den meisten Anwendungen billiger ist digitale Filter zu verwenden. Billiger vor allem deshalb, weil bei der Herstellung von digitalen Filtern keine aufwendigen Abgleicharbeiten mehr notwendig sind. Auch der Platzbedarf eines Filters kann ein Grund für die Verwendung digitaler Filter sein, es ist möglich ein digitales Filter samt DSP, AD/DA-Wandler etc. in einem IC zu integrieren. Ein weiterer Vorteil ist, dass allfällige Änderungen am Filterdesign meist ohne Veränderung der Hardware vorgenommen werden können. Zudem können mit einem digitalen Filter Funktionen realisiert werden, die analog nicht, oder nur schwer durchführbar sind z.B: Faltung (FIR-Filter), FFT, Filter sehr hoher Ordnung, adaptive Filter etc.

Digitale Filter haben aber nicht nur Vorteile gegenüber analogen Filter, ein Hauptproblem der digitalen Filter sind hohe Frequenzen. Bei der Abtastung von Signalen mit Frequenzkomponenten die grösser als die halbe Samplingfrequenz sind, entstehen Aliasingfehler. Zudem ist die Signalverarbeitung bei hohen Frequenzen (>100 MHz) problematisch bis unmöglich. Da es (noch) keine guten A/D-Wandler gibt die mit so hohen Samplingraten abtasten können.

4.2 Aufbau eines digitalen Filter



- **Tiefpassfilter 1:** Damit wird die Bandbreite des Signals auf die halbe Samplingfrequenz beschränkt. So werden Aliasingfehler verhindert.
- **Sample/Hold-Einheit:** Das zeitkontinuierliche Signal wird abgetastet und somit zeitdiskretisiert.
- **A/D-Wandler:** Die zeitkontinuierlichen Signalproben, werden in digitale Werte umgewandelt.
- **Rechenlogik:** In der Rechenlogik wird die eigentliche Filterung durchgeführt. Beim IIR-Filter werden anhand der Differenzgleichung des Filters die Ausgangswerte berechnet.
- **D/A-Wandler:** Er wandelt die digitalen Werte wieder in analoge Pegel um.
- **Tiefpassfilter 2:** Das Tiefpassfilter ($f_g = f_s/2$) am Ausgang entfernt die Oberwellen, die durch die D/A-Wandlung entstanden sind.

5 Theoretische Grundlagen

5.1 Digitale Signalverarbeitung

5.1.1 Die Differenzgleichung

Ein digitales kausales LTI System hängt nur von der Vergangenheit der Signale an Ein- und Ausgang ab. Deshalb gilt [os]:

$$y_n = \sum_{i=1}^{\infty} n_i \cdot y_{n-i} + \sum_{k=0}^{\infty} m_k \cdot u_{n-k} \quad (1)$$

Wobei y_n der n-te Wert am Ausgang und u_n der n-te Eingangswert ist. Die n_i und m_k sind konstant da das System zeitinvariant ist.

Gleich wie bei Zeitkontinuierlichen Signalen kann man nun ein Differential definieren:

Differential im zeitkontinuierlichen Fall: Differential in zeitdiskreten Fall:

$$y'(t) = \lim_{h \rightarrow 0} \left(\frac{y(t) - y(t-h)}{h} \right) \quad y'_n = y_n - y_{n-1} \quad (2)$$

Mit Hilfe von (2) kann man die Gleichung (1) nun wie folgt umformen:

$$\sum_{i=0}^{\infty} a_i \cdot y^{(i)}_n = \sum_{k=0}^{\infty} b_k \cdot u^{(k)}_n \quad (3)$$

5.1.2 Die Z-Transformation

Zeitkontinuierliche Systeme werden oft im Bildbereich beschrieben oder analysiert. Die Laplace-Transformation dient dann zur Transformation des Systems vom Zeit- in den Bildbereich.

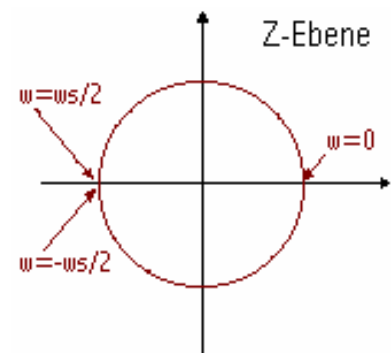
Für zeitdiskrete System kann ein Pendant zur Laplacetransformation definiert werden, die Z-Transformation [os]:

$$X(z) = \sum_{n=-\infty}^{\infty} x_n \cdot z^{-n} \quad (4)$$

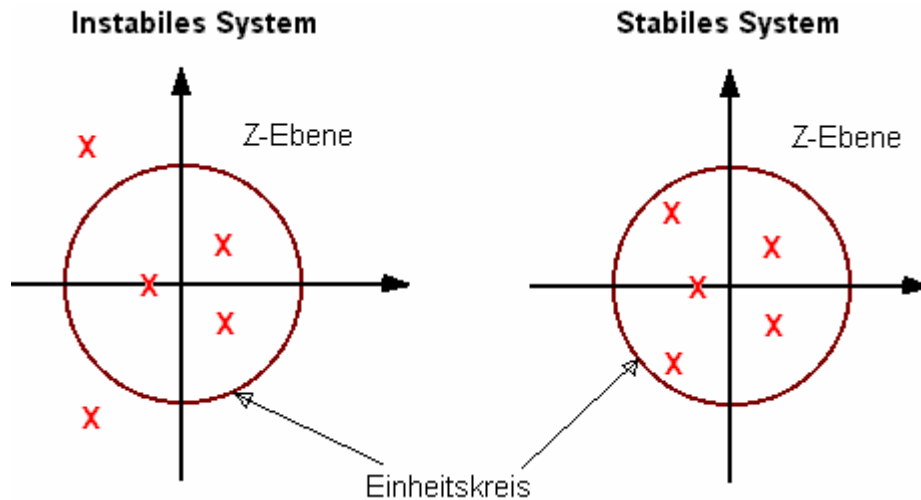
Wie bei der Laplace-Transformation ist auch die Fourier-Transformation eine Unter-
menge der Z-Transformation. Es handelt sich dabei um die diskrete Fouriertransformation:

$$X(\omega) = \sum_{n=-\infty}^{\infty} x_n \cdot e^{-j \cdot \omega \cdot n} \quad (5) \quad (\text{Herleitung siehe [OS]})$$

Ein Vergleich der beiden Transformationen liefert: $z = e^{j\omega}$
Z ist eine Variable in der komplexen Ebene und $e^{j\omega}$ ist auf dem Einheitskreis. Man erkennt somit sehr schön, dass das Frequenzspektrum eines Signals periodisch sein muss.



Die Z-Ebene ist ein gutes Hilfsmittel um Stabilitätsbetrachtungen eines Systems durchzuführen. Man kann die Pole und Nullstellen der Übertragungsfunktion in der Z-Ebene einzeichnen. Falls Pole ausserhalb oder auf dem Einheitskreis zu liegen kommen, ist das System instabil.



Es wäre nun schön, wenn wir einen Zusammenhang zwischen einem zeitkontinuierlichen System und dem entsprechenden zeitdiskreten System hätten, sodass das Eingangs- und Ausgangssignal des zeitdiskreten Systems als gesampeltes Signal des zeitkontinuierlichen Systems aufgefasst werden könnte.

Es gibt verschiedene Methoden, mit denen das bewerkstelligt werden kann. Zum Beispiel Transformation im Zeitbereich. Man betrachtet dabei die Stossantwort des zeitdiskreten Systems als gesampelte Stossantwort des zeitkontinuierlichen Systems. Bei dieser Methode entsteht aber ein Aliasingfehler wenn die Stossantwort nicht eine begrenzte Bandbreite hat.

Eine andere Methode ist eine Transformation zwischen der S- und der Z-Ebene. Sie wird im nachfolgendem Kapitel erläutert.

Weiterführende Literatur zur Z-Transformation und Tabellen siehe: [os], [mk].

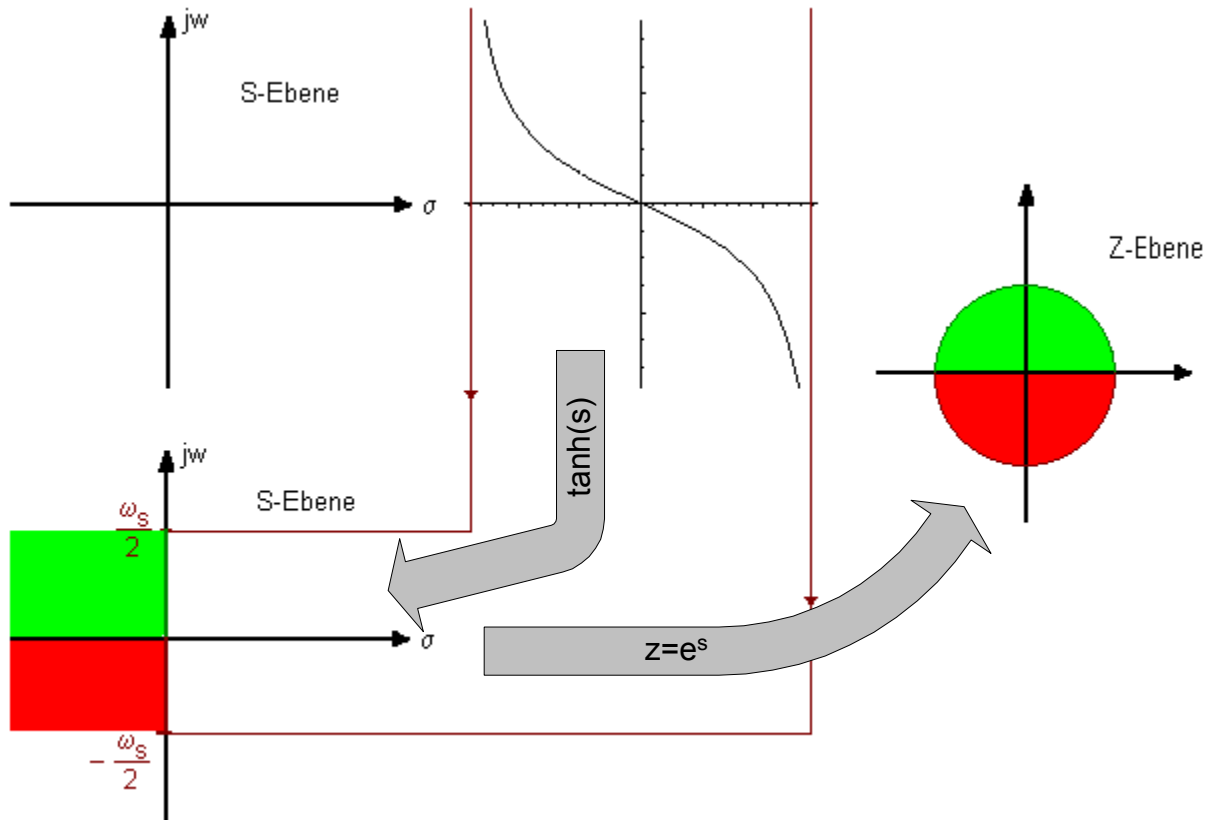
5.1.3 Die bilineare Transformation

Wie schon erwähnt kann mit der bilinearen Transformation ein System von der S-Ebene in die Z-Ebene transformiert werden. Genauer gesagt wird die imaginäre Achse der S-Ebene auf den Einheitskreis der Z-Ebene abgebildet. Dabei gibt es aber ein Problem, der Einheitskreis hat nur ein Umfang von $2 \cdot \pi$, die imaginäre Achse hat jedoch eine unendliche Ausdehnung. Die maximale Frequenz in der Z-Ebene auf dem Einheitskreis ist $\omega_s/2$, die minimale $-\omega_s/2$. Wir müssen also auch in der S-Ebene die imaginäre Achse auf $\pm\omega_s/2$ begrenzen. Am geeignetsten dazu ist eine bijektive Abbildung die bei kleinen Frequenzen möglichst linear ist, also zum Beispiel die Arkustangens-Funktion. Nun kann ganz einfach mit der Exponentialfunktion die beschränkte $j \cdot \omega$ -Achse auf den Einheitskreis abgebildet werden.

Nach ein wenig Umformen [bj] erhält man dann die Formel für die bilineare Transformation:

$$s = \frac{\omega_s}{\pi} \cdot \frac{1 - z^{-1}}{1 + z^{-1}} \quad (6)$$

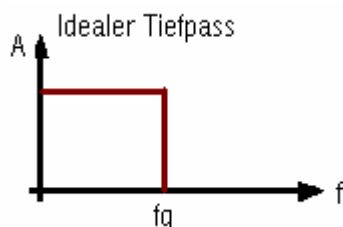
Bilineare Transformation



Durch die Abbildung mit der Arkustangens Funktion werden die Frequenzen relativ zu ω_s verschoben. Damit die Grenzfrequenz des digitalen Filters mit der des analogen Filters übereinstimmt, muss ein sogenanntes Prewarping vorgenommen werden [Ibj]:

$$\omega_{\text{prew}} = \frac{\omega_s}{\pi} \tan\left(\frac{\omega}{\omega_s} \cdot \pi\right) \quad (7)$$

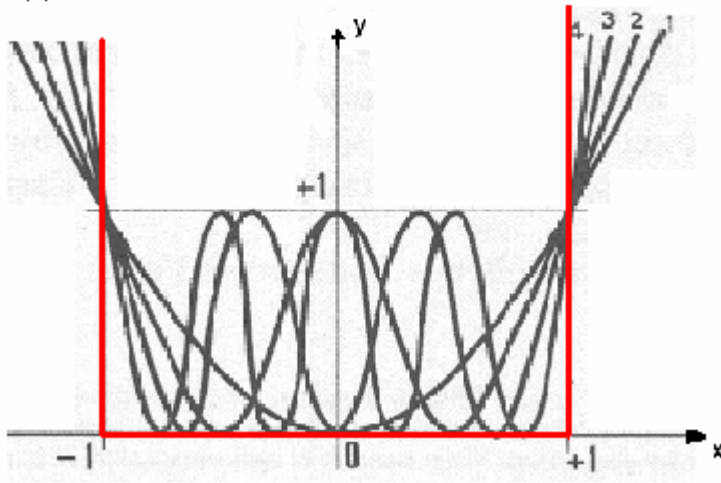
5.2 Grundlegende Filtertheorie



Ein ideales Tiefpassfilter lässt Frequenzen bis zu einer bestimmten Frequenz (f_g) ohne Dämpfung passieren, alle Frequenzen die darüberliegen, werden unendlich stark gedämpft. Ein ideales Tiefpassfilter besitzt zudem einen linearen Phasengang.

Ein Filter das diese Bedingungen erfüllt ist grundsätzlich nicht realisierbar, das Filter besäße eine Stossantwort die von $-\infty$ bis ∞ gehen würde ($\sin(x)/x$ -Funktion), das heisst das Filter wäre nicht kausal. Es gibt nun verschiedene Approximationen an diesen Filtertyp, Ziel ist immer mit einer Polynomfunktion ein Rechteck zu approximieren [rb].

Rechteckfunktion durch Chebyshev Polynome approximiert



Je nach dem auf welche Eigenschaft des idealen Filters man am ehesten verzichten kann, ist eine andere Approximation geeignet. Die zwei am häufigsten verwendeten Approximationen sind die Butterworth und die Chebyshev Polynome. Der Unterschied zwischen den beiden ist, dass beim Chebyshev Polynom die Dämpfung im Durchlassbereich nicht flach sondern gewellt ist. Das Butterworth Polynom hat hingegen den Nachteil, dass die Steilheit beim Übergang vom Durchlass- in den Sperrbereich nicht so gross ist wie beim Chebyshev Polynom.

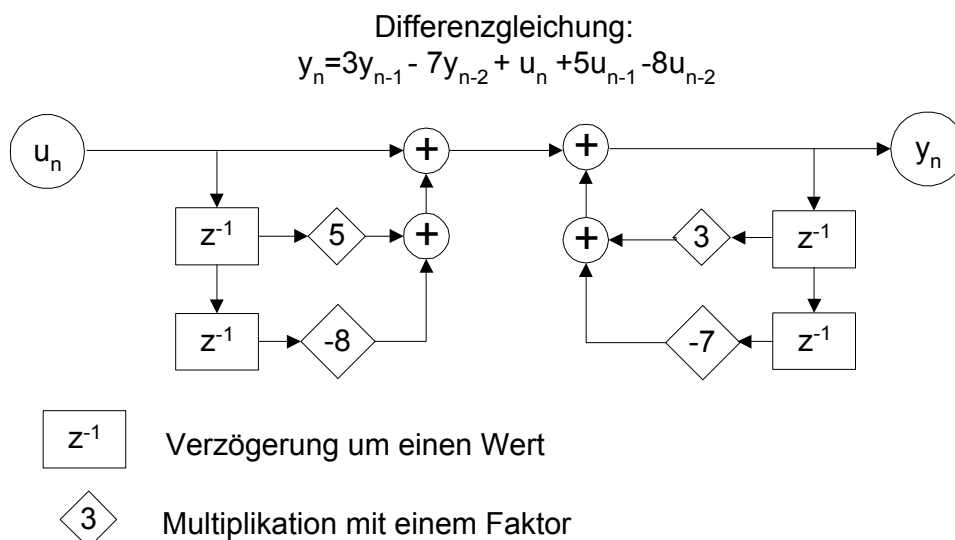
5.3 IIR-Filter

5.3.1 Theorie

Es gibt grundsätzlich zwei Arten von digitalen Filter, die FIR-Filter und die IIR-Filter. Auf die FIR-Filter [os,mk] möchte ich hier nicht eingehen, weil sie nichts mit unserer Semesterarbeit zu tun haben.

Bei IIR-Filtern wird grundsätzlich die Differenzgleichung (1) nachgebildet. Meist wird sie jedoch nicht direkt so implementiert wie sie bei (1) steht, sondern sie wird noch umgeformt. Der Übersichtlichkeit wegen wird sowieso meist zuerst die Differenzgleichung als Blockdiagramm dargestellt.

Blockdiagramm:

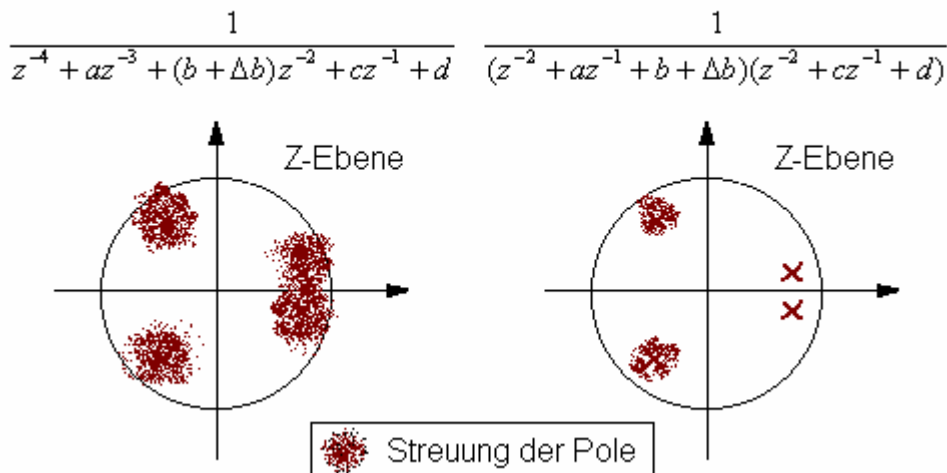


Eine sehr häufig verwendete Implementationsform der Differenzgleichung ist die kanonische Normalform. Dabei wird die Grösse des benötigten Speichers minimiert. Zusätzlich wird oft noch der Zähler und Nenner der Übertragungsfunktion in Polynome 2. Ordnung zerlegt. Anschliessend wird die Übertragungsfunktion in Differenzgleichungen 2. Ordnung umgewandelt. Man hat also schlussendlich lauter Systeme 2. Ordnung (Second-Order-Sections), bei denen der Eingang des einen, mit dem Ausgang des anderen verbunden ist. Der grosse Vorteil dieser Form ist die viel kleinere Anfälligkeit auf Quantisierungsfehler in Folge begrenzter Wortlänge der Koeffizienten.

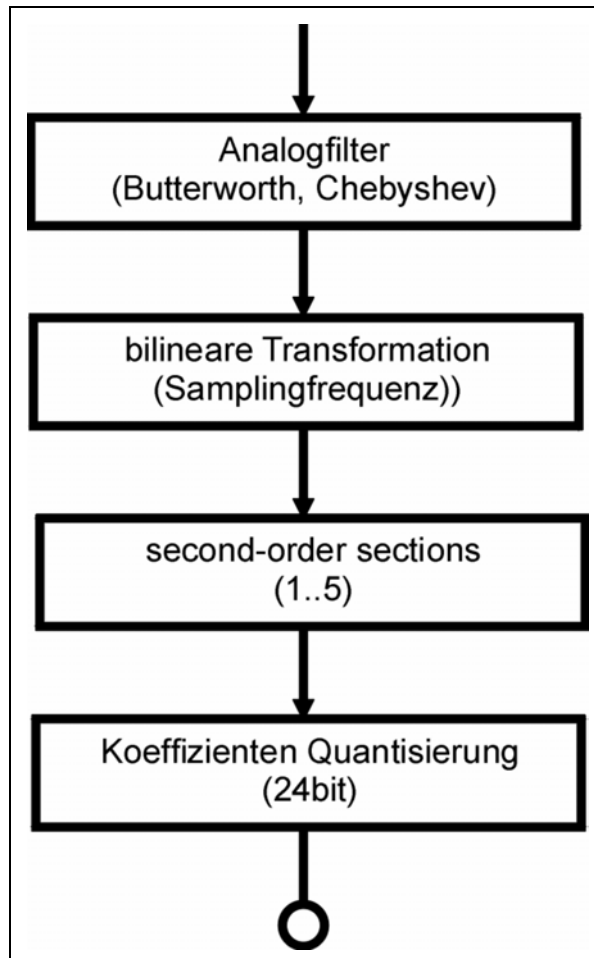
$$H_i(z) = \frac{\alpha \cdot (1 + \mu \cdot z^{-1} + \sigma \cdot z^{-2})}{\frac{1}{2} - \gamma \cdot z^{-1} + \beta \cdot z^{-2}} \quad (8)$$

Um das zu verstehen betrachtet man am besten die Lage der Pol und Nullstellen des Systems in der z-Ebene. Wenn sich nun beim Gesamtpolynom einer der Koeffizienten verändert, so verschieben sich alle Pole. Anders sieht es aus wenn einer der Koeffizienten bei den Polynomen 2. Ordnung ändert. Es sind dann nur die 2 Pole von dem Fehler betroffen, bei denen der Koeffizient steht.

Polstreuung auf Grund der Wertdiskretisierung



5.3.2 Designflow (Entwicklung eines IIR-Filters):



Zuerst wird ein analoges Filter mit der gewünschten Charakteristik berechnet. Dabei muss berücksichtigt werden, welchen Filtertyp (Butterworth, Chebyshev1/2, Cauer,...) man verwenden will. Je nach Filtertyp ergibt sich ein etwas anderes Design.

Danach werden mittels bilinearer Transformation und unter Einbezug der Abtastfrequenz die Filterkoeffizienten für das IIR-Filter berechnet.

Nun werden die Koeffizienten in sogenannte Second-Order-Sections umgewandelt. Dies reduziert die Gefahr von Quantisierungsfehlern erheblich. Das grösste Problem bei der Implementation dieser Umrechnung, stellt das Finden der Nullstellen und der Umgang mit komplexen Zahlen in C++ dar.

Zum Schluss müssen die Koeffizienten quantisiert werden. Das Risiko von Quantisierungsfehlern, wird durch den Umstand, dass wir für unsere 16-Bit Daten einen 24-Bit Datenbus auf dem DSP zur Verfügung haben und der Bildung von Second-Order-Sections minimalisiert.

6 Lösung des Problems

6.1 Lösungsansatz

Wir teilten die durchzuführenden Berechnungen zuerst einmal in 2 Teilbereiche auf. Zum einen haben wir die Berechnung der Filterkoeffizienten für die Second-Order-Sections. Diese Berechnungen müssen nicht in Echtzeit durchgeführt werden. Wichtig ist jedoch, dass die Berechnungen genau sind und dass die Eingabe der Daten benutzerfreundlich ist. Wir entschieden uns deshalb für eine Softwarelösung mit einem Windows GUI auf dem PC. Die Berechnung des Ausgangssignals mit Hilfe des Eingangssignals und der Differenzgleichung hat hingegen in Echtzeit zu erfolgen. Wir werden deshalb diesen Teil mit dem DSP 56002 Evaluation Board von Motorola lösen.

In diesem ersten Teil der Semesterarbeit haben wir uns vorwiegend mit der Entwicklung der Software beschäftigt. Wie schon erwähnt muss beim Design eines IIR-Filters zuerst ein zeitkontinuierliches Filter berechnet werden. Weil es kompliziert ist die komplexen Nullstellen eines Polynoms mit Hilfe von C++ zu berechnen, benötigen wir eine geschlossene Formel die uns die komplexe Polpaare für ein Tiefpassfilter vom Typ Chebyshev 1 oder Butterwoth liefert. Die Pole die wir so erhalten sind für einen Tiefpass. Wird ein Hochpass, Bandpass oder eine Bandsperre benötigt, müssen die Pole noch transformiert werden. Anschliessend kann das Filter mit der bilinearen Transformation in die Z-Ebene transformiert werden und weil die bilineare Transformation die Ordnung eines Filters nicht verändert, erhalten wir direkt Second-Order-Sections. Wir müssen also nur noch die einzelnen Filterkoeffizienten $(\alpha, \beta, \gamma, \mu, \sigma)$ bestimmen, welche dann über das serielle Kabel auf das DSP-Board transferiert werden können.

6.2 Lösungsweg

6.2.1 Berechnung eines Polpaars für ein Chebyshev Filter Typ 1

Das Ziel unserer Berechnung ist eine geschlossene Funktion zur Berechnung von a_i und b_i in der Folgenden Formel zu erhalten.

$$H_i(s) = \frac{1}{1 + a_i \frac{s}{\omega_g} + b_i \frac{s^2}{\omega_g^2}} \quad (9) \quad \omega_g = \text{Grenzfrequenz}$$

Glücklicherweise fanden wir eine solche Formel in der Literatur [ts], leider aber nirgends eine Herleitung:

Ordnung n gerade:

$$b_i = \frac{1}{\cosh^2(\gamma) - \cos^2\left(\frac{(2 \cdot i - 1) \cdot \pi}{2 \cdot n}\right)} \quad (10.1) \quad a_i = 2 \cdot b_i \cdot \sinh(\gamma) \cdot \cos\left(\frac{(2 \cdot i - 1) \cdot \pi}{2 \cdot n}\right) \quad (10.2)$$

$$i \in \left[1, \frac{n}{2}\right]$$

Ordnung n ungerade:

$$b_i = \frac{1}{\cosh^2(\gamma) - \cos^2\left(\frac{(i-1) \cdot \pi}{n}\right)} \quad (10.5)$$

$$a_i = 2 \cdot b_i \cdot \sinh(\gamma) \cdot \cos\left(\frac{(i-1) \cdot \pi}{n}\right) \quad (10.6)$$

Wobei aber: $b_1 = 0 \quad (10.3)$

$$a_1 = \frac{1}{\sinh(\gamma)} \quad (10.4)$$

Darin ist: $\gamma = \frac{1}{n} \operatorname{ArSinh} \frac{1}{\sqrt{\left(\frac{A_{\max}}{A_{\min}}\right)^2 - 1}} \quad (10.7), \quad i \in \left[2, \frac{n+1}{2}\right]$

A_{\max} = Maximale Verstärkung im Durchlassbereich

A_{\min} = Minimale Verstärkung im Durchlassbereich

6.2.2 Berechnung eines Polpaars für ein Butterworth Filter

Das Ziel ist auch hier eine geschlossene Funktion zur Berechnung von a und b in der Formel:

$$H_i(s) = \frac{1}{1 + a_i \frac{s}{\omega_g} + b_i \frac{s^2}{\omega_g^2}} \quad (9)$$

Beim Butterworth Filter liegen die Pole auf einem Halbkreis in der S-Ebene [rb]. Ein Pol auf diesem Halbkreis hat den Wert:

$$p_i = -\sin\left(i \cdot \frac{\pi}{n+1}\right) + j \cdot \cos\left(i \cdot \frac{\pi}{n+1}\right) \quad (12) \quad \text{Wobei der Index } i \text{ von } 1 \text{ bis } n \text{ geht.}$$

Man kann nun je die 2 konjugiert komplexen Pole zusammenfassen und erhält die Gleichungen für a_i und b_i [ts]:

Ordnung n gerade:

$$a_i = 2 \cdot \cos\left(\frac{(2 \cdot i - 1) \cdot \pi}{2 \cdot n}\right) \quad (11.1) \quad b_i = 1 \quad (11.2) \quad i \in \left[1, \frac{n}{2}\right]$$

Ordnung n ungerade:

$$a_1 = 1 \quad (11.3) \quad b_1 = 0 \quad (11.4) \quad a_i = 2 \cdot \cos\left(\frac{(i-1) \cdot \pi}{n}\right) \quad (11.5) \quad b_i = 1 \quad (11.6)$$

$$i \in \left[2, \frac{n+1}{2}\right]$$

6.2.3 Transformation eines Polpaares in die Z-Ebene

Für die Umwandlung wenden wir die Formeln (6) und (7) der bilinearen Transformation auf die Formel (9) an. Wir erhalten:

$$H_i(z) = \frac{\pi^2 \cdot \omega_g^2 \cdot (1 + z^{-1} + z^{-2})}{\pi^2 \omega_g^2 + a_i \pi \omega_g \omega_s + b_i \omega_s^2 + 2(\pi^2 \omega_g^2 - b_i \omega_s^2) z^{-1} + (\pi^2 \omega_g^2 - a_i \pi \omega_g \omega_s + b_i \omega_s^2) z^{-2}}$$

Ein Koeffizientenvergleich mit der kanonischen Normalform (8) liefert uns:

$$\alpha = \frac{\pi^2 \omega_g^2}{2 \cdot (\pi^2 \omega_g^2 + a_i \pi \omega_g \omega_s + b_i \omega_s^2)} \quad \gamma = \frac{\pi^2 \omega_g^2 - b_i \omega_s^2}{\pi^2 \omega_g^2 + a_i \pi \omega_g \omega_s + b_i \omega_s^2}$$

$$\beta = \frac{\pi^2 \omega_g^2 - a_i \pi \omega_g \omega_s + b_i \omega_s^2}{2 \cdot (\pi^2 \omega_g^2 + a_i \pi \omega_g \omega_s + b_i \omega_s^2)} \quad \sigma = 1 \quad \mu = 2$$

Wobei: $\omega_g = \frac{\omega_s}{\pi} \tan\left(\frac{\omega_g}{\omega_s} \cdot \pi\right)$

6.2.4 Transformation eines Tiefpasspolpaares in ein Hochpasspolpaar und anschließende Transformation in die Z-Ebene

Bei der Umwandlung von Tiefpass in Hochpass muss der Frequenzverlauf an einer, zur Ordinate parallelen Achse die durch ω_g geht, gespiegelt werden. Aus dem Ausdruck

$\frac{s}{\omega_g}$ wird $\frac{\omega_g}{s}$. Anschliessend wird wieder wie oben die bilineare Transformation durchgeführt. Wir erhalten:

$$H_i(z) = \frac{\omega_s^2 \cdot (1 - 2 \cdot z^{-1} + z^{-2})}{b_i \pi^2 \omega_g^2 + a_i \pi \omega_g \omega_s + \omega_s^2 + 2(b_i \pi^2 \omega_g^2 - \omega_s^2) z^{-1} + (b_i \pi^2 \omega_g^2 - a_i \pi \omega_g \omega_s + \omega_s^2) z^{-2}}$$

Ein Koeffizientenvergleich mit der kanonischen Normalform (8) liefert uns:

$$\alpha = \frac{\omega_s^2}{2 \cdot (b_i \pi^2 \omega_g^2 + a_i \pi \omega_g \omega_s + \omega_s^2)} \quad \gamma = \frac{b_i \pi^2 \omega_g^2 - \omega_s^2}{b_i \pi^2 \omega_g^2 + a_i \pi \omega_g \omega_s + \omega_s^2}$$

$$\beta = \frac{b_i \pi^2 \omega_g^2 - a_i \pi \omega_g \omega_s + \omega_s^2}{2 \cdot (b_i \pi^2 \omega_g^2 + a_i \pi \omega_g \omega_s + \omega_s^2)} \quad \sigma = 1 \quad \mu = -2$$

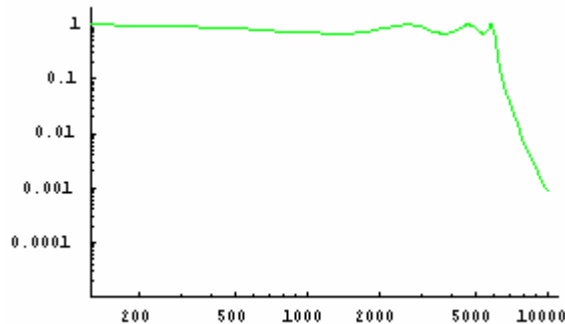
Wobei: $\omega_g = \frac{\omega_s}{\pi} \tan\left(\frac{\omega_g}{\omega_s} \cdot \pi\right)$

7 Überprüfung der Implementation mit Mathematica

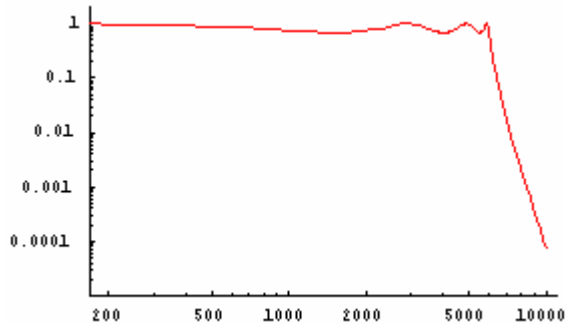
7.1 Chebyshev Tiefpass

Samplefrequenz[fs]=32000Hz, Grenzfrequenz[fg]=6000Hz, Sperrfrequenz[fr]=11000Hz, Dämpfung im Sperrbereich=80dB, Welligkeit im Durchlassbereich=2 dB, Filterordnung=7

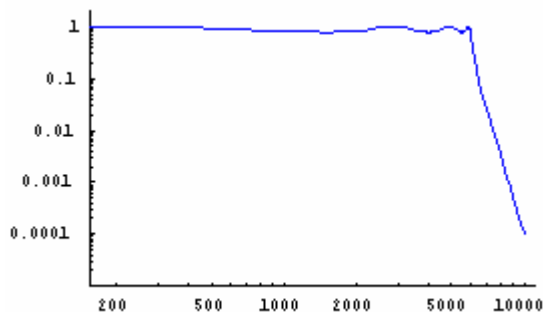
Frequenzgang des mit Mathematica berechneten analogen Filters:



Frequenzgang des mit Mathematica berechneten digitalen Filters:



Frequenzgang des mit der Software „Filterdesigner“ berechneten Filters:



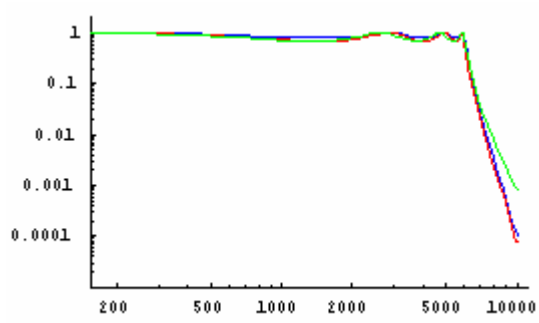
Koeffizienten (SOS1):
 $\alpha_1 = 0.047017$
 $\beta_1 = -0.405966$
 $\gamma_1 = -0.094034$
 $\sigma_1 = 1.000000$
 $\mu_1 = 2.000000$

Koeffizienten (SOS2):
 $\alpha_2 = 0.036986$
 $\beta_2 = 0.354093$
 $\gamma_2 = 0.706147$
 $\sigma_2 = 1.000000$
 $\mu_2 = 2.000000$

Koeffizienten (SOS3):
 $\alpha_3 = 0.100374$
 $\beta_3 = 0.408408$
 $\gamma_3 = 0.506912$
 $\sigma_3 = 1.000000$
 $\mu_3 = 2.000000$

Koeffizienten (SOS4):
 $\alpha_4 = 0.146872$
 $\beta_4 = 0.468816$
 $\gamma_4 = 0.381329$
 $\sigma_4 = 1.000000$
 $\mu_4 = 2.000000$

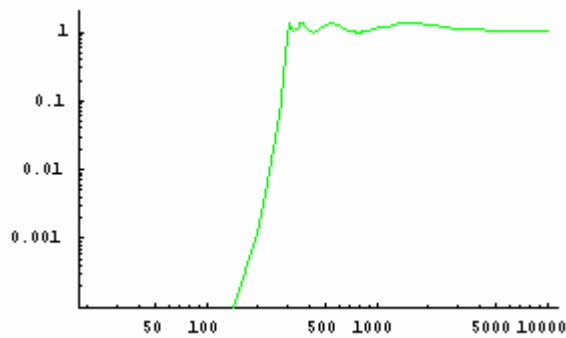
Vergleich der drei Frequenzgänge:



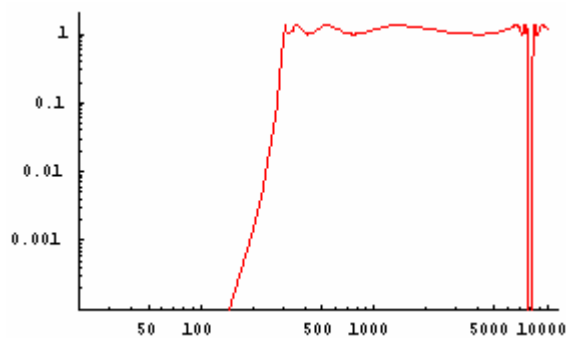
7.2 Chebyshev Hochpass

Samplefrequenz[fs]=8000Hz, Grenzfrequenz[fg]=300Hz, Sperrfrequenz[fr]=240Hz, Dämpfung im Sperrbereich=40dB, Welligkeit im Durchlassbereich=2 dB, Filterordnung=8

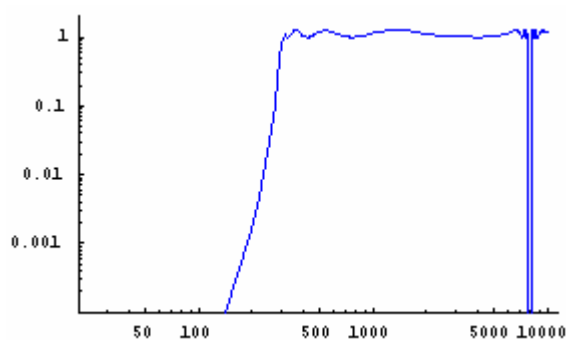
Frequenzgang des mit Mathematica berechneten analogen Filters:



Frequenzgang des mit Mathematica berechneten digitalen Filters:



Frequenzgang des mit der Software „Filterdesigner“ berechneten Filters:



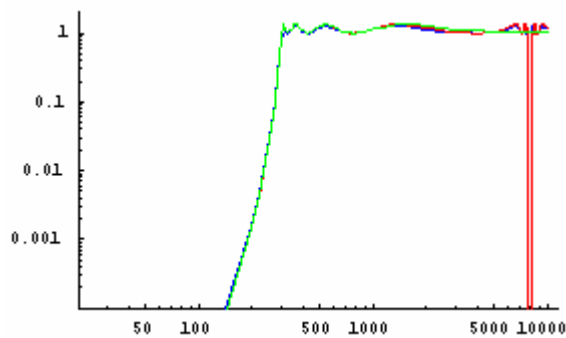
Koeffizienten (SOS1):
 $\alpha_1 = 0.276865$
 $\beta_1 = 0.191020$
 $\gamma_1 = 0.416440$
 $\sigma_1 = 1.000000$
 $\mu_1 = -2.000000$

Koeffizienten (SOS2):
 $\alpha_2 = 0.444627$
 $\beta_2 = 0.427338$
 $\gamma_2 = 0.851171$
 $\sigma_2 = 1.000000$
 $\mu_2 = -2.000000$

Koeffizienten (SOS3):
 $\alpha_3 = 0.478516$
 $\beta_3 = 0.475920$
 $\gamma_3 = 0.938143$
 $\sigma_3 = 1.000000$
 $\mu_3 = -2.000000$

Koeffizienten (SOS4):
 $\alpha_4 = 0.489867$
 $\beta_4 = 0.493733$
 $\gamma_4 = 0.965734$
 $\sigma_4 = 1.000000$
 $\mu_4 = -2.000000$

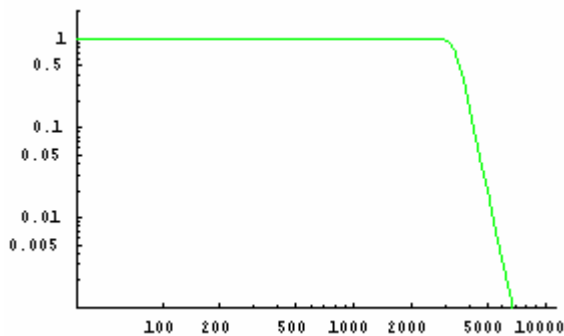
Vergleich der drei Frequenzgänge:



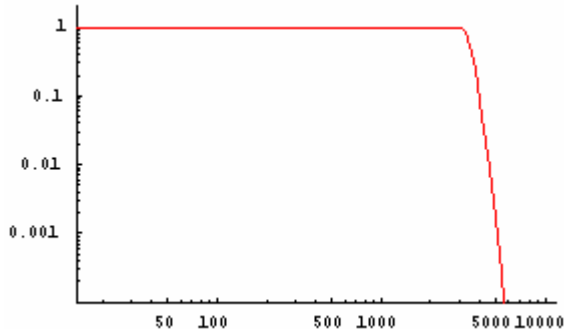
7.3 Butterworth Tiefpass

Samplefrequenz[fs]=16000Hz, Grenzfrequenz[fg]=3400Hz, Sperrfrequenz[fr]=4000Hz, Dämpfung im Sperrbereich=40dB, Filterordnung=10

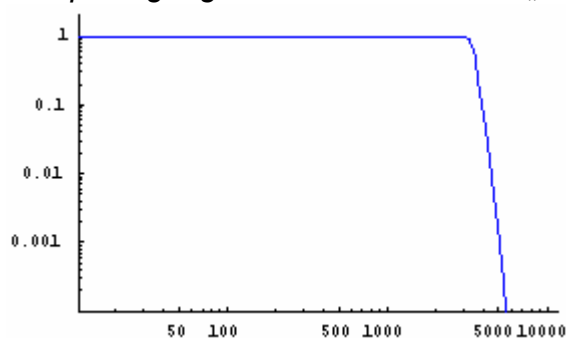
Frequenzgang des mit Mathematica berechneten analogen Filters:



Frequenzgang des mit Mathematica berechneten digitalen Filters:



Frequenzgang des mit der Software „Filterdesigner“ berechneten Filters:



Koeffizienten (SOS1):
 $\alpha_1 = 0.097755$
 $\beta_1 = 0.010100$
 $\gamma_1 = 0.119081$
 $\sigma_1 = 1.000000$
 $\mu_1 = 2.000000$

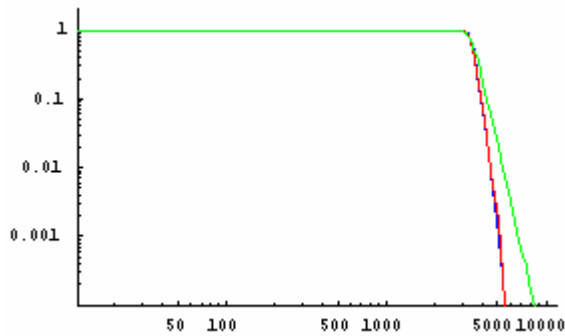
Koeffizienten (SOS2):
 $\alpha_2 = 0.102679$
 $\beta_2 = 0.035794$
 $\gamma_2 = 0.125079$
 $\sigma_2 = 1.000000$
 $\mu_2 = 2.000000$

Koeffizienten (SOS3):
 $\alpha_3 = 0.113559$
 $\beta_3 = 0.092568$
 $\gamma_3 = 0.138332$
 $\sigma_3 = 1.000000$
 $\mu_3 = 2.000000$

Koeffizienten (SOS4):
 $\alpha_4 = 0.132949$
 $\beta_4 = 0.193747$
 $\gamma_4 = 0.161952$
 $\sigma_4 = 1.000000$
 $\mu_4 = 2.000000$

Koeffizienten (SOS5):
 $\alpha_5 = 0.166337$
 $\beta_5 = 0.367971$
 $\gamma_5 = 0.202624$
 $\sigma_5 = 1.000000$
 $\mu_5 = 2.000000$

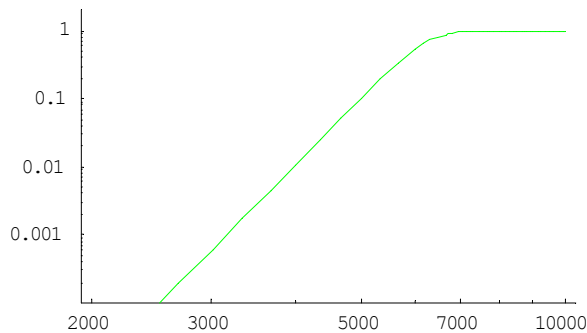
Vergleich der drei Frequenzgänge:



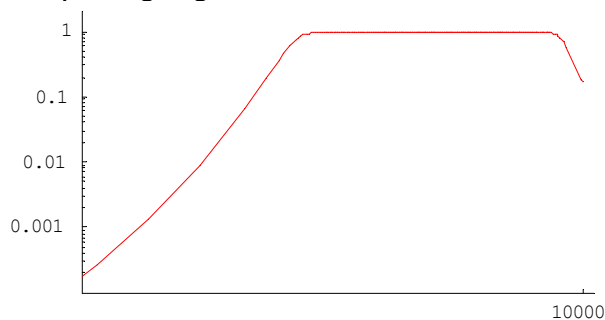
7.4 Butterworth Hochpass

Samplefrequenz[fs]=16000Hz, Grenzfrequenz[fg]=6302Hz, Sperrfrequenz[fr]=5192Hz, Dämpfung im Sperrbereich=40dB, Filterordnung=10

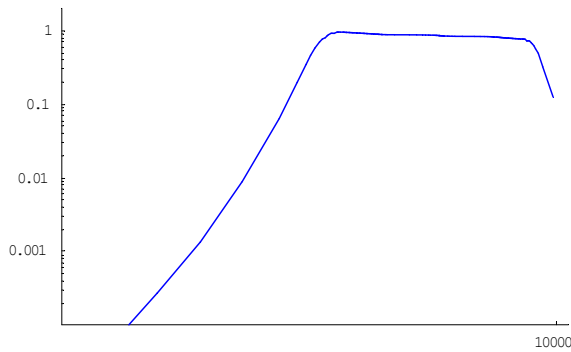
Frequenzgang des mit Mathematica berechneten analogen Filters:



Frequenzgang des mit Mathematica berechneten digitalen Filters:



Frequenzgang des mit der Software „Filterdesigner“ berechneten Filters:



Koeffizienten
(SOS1):
 $\alpha_1 = 0.033243$
 $\beta_1 = 0.120785$
 $\gamma_1 = -0.487815$
 $\sigma_1 = 1.000000$
 $\mu_1 = -2.000000$

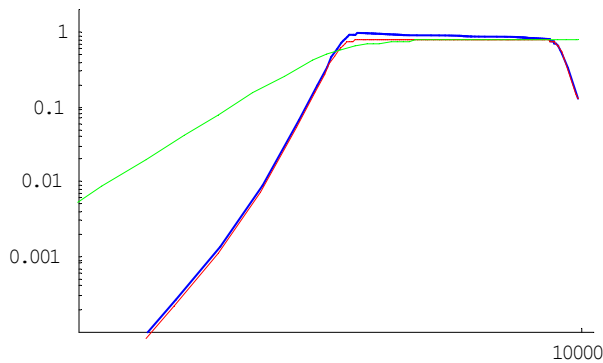
Koeffizienten
(SOS2):
 $\alpha_2 = 0.034524$
 $\beta_2 = 0.144717$
 $\gamma_2 = -0.506621$
 $\sigma_2 = 1.000000$
 $\mu_2 = -2.000000$

Koeffizienten
(SOS3):
 $\alpha_3 = 0.037256$
 $\beta_3 = 0.195735$
 $\gamma_3 = -0.546710$
 $\sigma_3 = 1.000000$
 $\mu_3 = -2.000000$

Koeffizienten
(SOS4):
 $\alpha_4 = 0.041810$
 $\beta_4 = 0.280773$
 $\gamma_4 = -0.613533$
 $\sigma_4 = 1.000000$
 $\mu_4 = -2.000000$

Koeffizienten
(SOS5):
 $\alpha_4 = 0.048825$
 $\beta_4 = 0.411784$
 $\gamma_4 = -0.716482$
 $\sigma_4 = 1.000000$
 $\mu_4 = -2.000000$

Vergleich der drei Frequenzgänge:



Fazit:

Die Überprüfung der Koeffizienten hat keine nennenswerten Abweichungen zwischen der Berechnung des digitalen Filters mit Mathematica und unserer Implementation ergeben. Weil durch die bilineare Transformation gewisse Frequenzbereiche ($\omega_s/4$ bis ∞) gestaucht werden, hat das digitale Filter in der Regel etwas steilere Flanken als das analoge.

8 Implementation auf dem PC

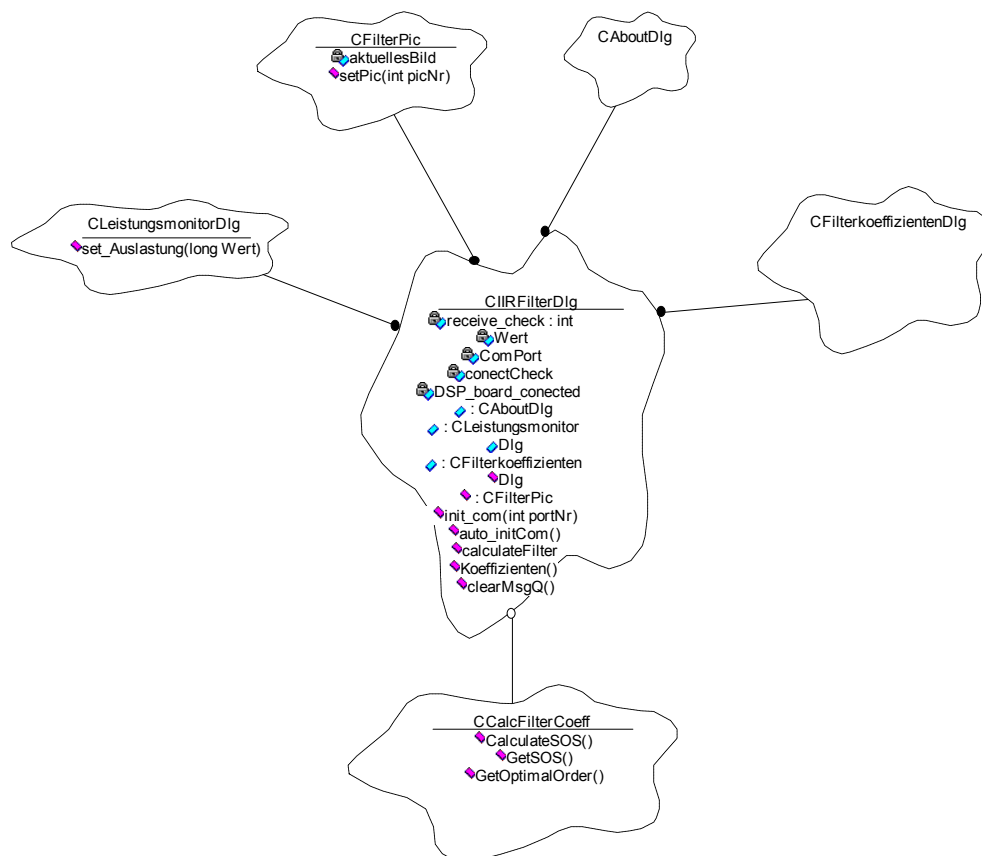
8.1 Entwicklungssystem

Zur Entwicklung des IIR-Filterdesigners wurde Visual C++ 6.0 von Microsoft verwendet. Der anfänglichen Euphorie über die gut implementierten neuen Funktionen in VC++6.0, folgte bald einmal der Frust über das neu Help-Programm MSDN.

Dies führte dazu, dass das alte Help-System von VC++5.0 genau so oft konsultiert wurde, wie die MSDN-Library von VC++6.0.

Die PC-Software wurde mit Hilfe der MFC-Bibliothek unter Windows98 entwickelt.

8.2 Grundstruktur der PC-Software



8.3 Übertragung

8.3.1 Serielle Schnittstelle (COM)

Die Ansteuerung der seriellen Schnittstelle, sowie ihre Initialisierung erfolgt durch die Klasse CSerialPort.

Diese Klasse wurde von Remon Spekrijse implementiert und unter <http://www.codeguru.com> frei zur Verfügung gestellt.

Mit dieser Klasse ist es möglich, die Schnittstelle zu initialisieren (Com-Nr, Baud-Rate, Stop-Bit, Parity-Bit), einzelne Zeichen (Bytes) über einen Windows-Event zu empfangen und einzelne Zeichen oder ganze Zeichenketten zu senden.

Beim Starten des IIR-Filterdesigners wird Com-Port 1 initialisiert. Kann dieses nicht gefunden werden, so versucht die Software Com-Port 2 zu aktivieren. Falls keine dieser Schnittstellen zur Verfügung steht, so wird eine Fehlermeldung ausgegeben und Com-Port1 als Defaultschnittstelle initialisiert.

Während der Laufzeit der Software, kann zwischen den zwei Schnittstellen umgeschaltet werden. Wird die gewählte Schnittstelle nicht gefunden, so erscheint eine Fehlermeldung und die alte Einstellung wird beibehalten.

8.3.2 Übertragungsvorgang

Falls der Benutzer auf den Knopf für die serielle Übertragung klickt (DSP Kanal links, DSP Kanal rechts), so werden zuerst die Filterkoeffizienten neu berechnet und in Double-Float Variablen auf dem PC gespeichert. Dieser 32-Bit Wert wird mit 2^{23} multipliziert (Umwandlung in 24-bit) und in eine Long-Integer Variabel (32-bit) geschrieben. Nun wird sukzessive die Übertragungszeichenkette aufgebaut:

Als erstes wird ein START -Zeichen eingefügt. Dieses signalisiert den Beginn der gesamten seriellen Übertragung. Danach kommt ein CH_L oder ein CH_R Zeichen, welches den Kanal festlegt. Nun kommt das FS-Zeichen, gefolgt von dem Datenbyte, welches die Samplingfrequenz festlegt. Jetzt erst beginnt die eigentliche Datenübertragung der Koeffizienten. Dazu wird als erstes die Nummer der Second-order Section eingefügt. Darauf folgt die Identifikation des Koeffizienten. Zur Übertragung des 24-bit Wertes des Koeffizienten wird das redundante vierte Byte der 32-bit Variabel auf dem PC weggelassen. Da nur 7 Datenbits pro übertragenem Byte mitgegeben werden können, werden die 24 zu übertragenden Bits in drei mal sieben und einmal drei Bits aufgeteilt und auf dem DSP-Board wieder zusammengesetzt. So werden Koeffizient um Koeffizient und Second-order Section um Second-order Section in die Zeichenkette eingefügt. Am Ende wird Ein END-Zeichen eingefügt. Dieses signalisiert das Ende der gesamten seriellen Übertragung.

Auf dem DSP-Board werden die Daten in ein Buffer im Y-Memory eingelesen. Von dort werden sie durch den im File „serInput.asm“ implementierte Assemblercode an die richtige Speicherstelle geschrieben und von dem eigentlichen Filtervorgang benutzt.

Der Ablauf dieser Verteilung ist aus dem Kapitel 9.3.8.2 ersichtlich.

8.3.3 Leistungsmonitor

Die Daten des Leistungsmonitors werden alle 0.5 Sekunden zum PC geschickt. Diese Daten bestehen aus einem 24-Bit Wert. Zur Identifikation der Leistungsmonitor-Daten wird zuerst dreimal das Steuerzeichen MONITOR_Daten (0xF0) gesendet. Die darauffolgenden drei Bytes ergeben den Auslastungswert des DSP. Dieser wird in einem separaten Dialog angezeigt.

9 Implementation auf dem DSP

9.1 Entwicklungssystem

Wir entschieden uns für die Implementation des IIR-Filters auf dem „DSP 56002 evaluation Module“ von Motorola. Der verwendete DSP 56002 ist ein fix-point Prozessor mit einer Wortlänge von 24 Bit und 2 Datenspeichern von 256 Worten. Für die DA/AD-Wandlung wird der Wandler CS4215 von Crystal verwendet, er ist über eine synchrone serielle Schnittstelle (SSI) mit dem DSP verbunden.

Die Software haben wir vollständig in Assembler entwickelt, was bei der geringen Komplexität unserer Software kein Problem darstellte. Die Initialisierung des DSP's sowie des DA/AD-Wandlers mussten wir nicht selbst erstellen, da bereits Initialisierungsroutinen von Motorola mitgeliefert wurden [mot].

9.2 Grundstruktur der DSP-Software

9.2.1 Grundfunktionen

Grundsätzlich mussten 3 Funktionen auf dem DSP realisiert werden:

1. Eigentlicher IIR-Filteralgorithmus
2. Empfangen der Filterkoeffizienten vom PC über das asynchrone serielle Schnittstelleninterface
3. Messung der DSP-Auslastung

Bei der Software war vor allem zu Beachten, dass gewisse Teile in Echtzeit ablaufen müssen. Es handelt sich dabei primär um den Filterungsvorgang, sowie das Einlesen vom AD-Wandler und das Schreiben zum DA-Wandler. Dieser Teil der Software musste auch auf minimalen Zeitbedarf optimiert werden.

Weiter muss auch der Empfang der Filterdaten über die asynchrone serielle Schnittstelle (SCI) schnell vonstatten gehen, da keine hardwaremässige Datenflusssteuerung (RTS/CTS-Leitungen) realisiert wurde.

Drittens muss noch die Auslastung des DSP's gemessen und der Messwert über die asynchrone serielle Schnittstelle (SCI) zum PC übertragen werden. Dieser Vorgang ist jedoch nicht sehr zeitkritisch vom DSP aus gesehen.

9.2.2 Realisierungsart

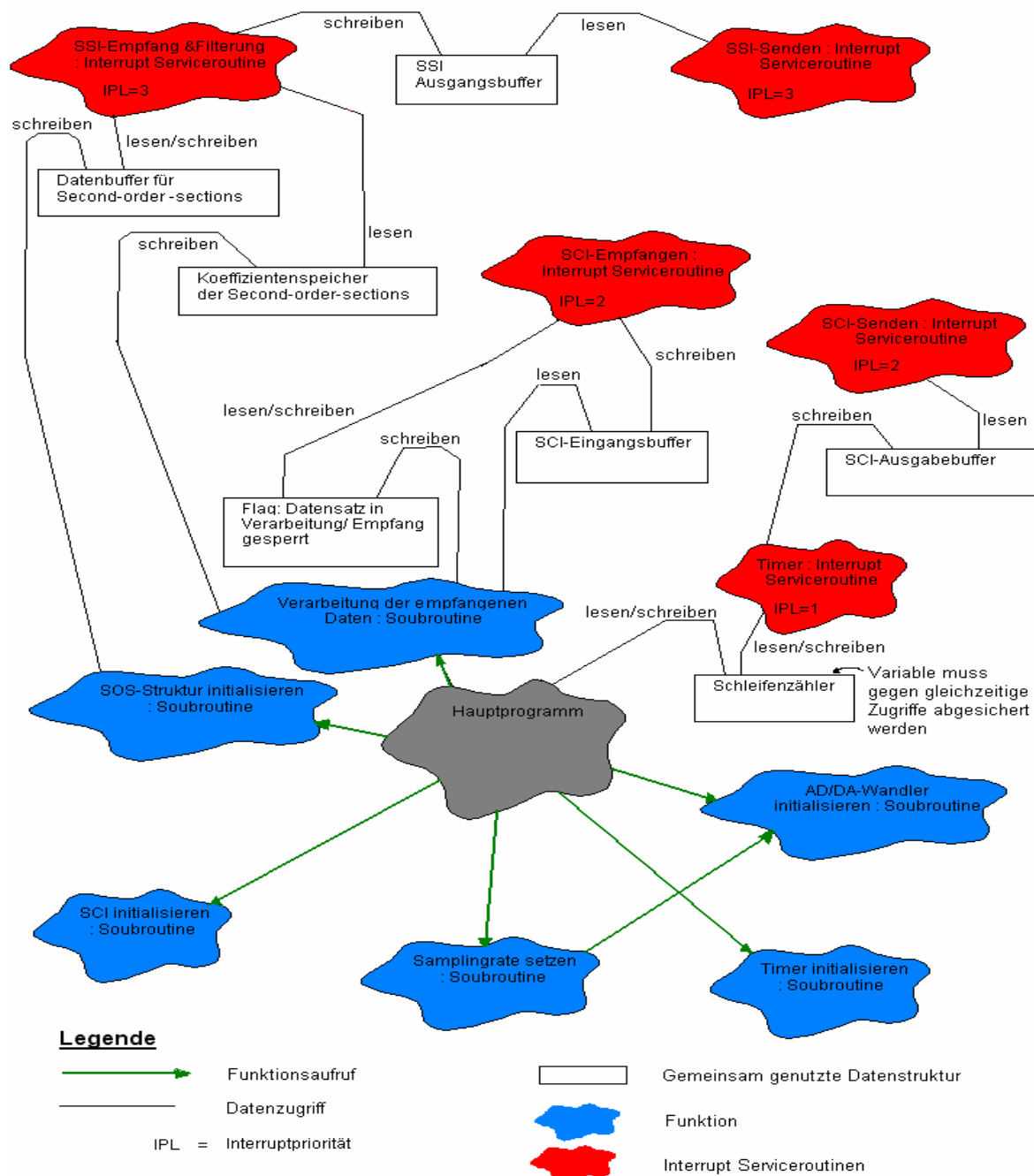
Wir entschieden uns die verschiedenen Funktionen soweit wie möglich mit Interrupt-handlern zu realisieren. Es ist dadurch möglich, den verschiedenen Programmteilen unterschiedliche Prioritäten zu geben, der Filteralgorithmus hat so die höchste Priorität, gefolgt vom Empfangen über die SCI und der Leistungsmessung. Die niedrigste Priorität hat automatisch das eigentliche Programm.

Die Leistungsmessung wird dadurch sehr simpel. Weil alle wichtigen Funktionen in Interrupt Serviceroutinen (ISR) realisiert werden, besteht das Hauptprogramm nur aus einem Initialisierungsteil sowie einer Endlosschleife, die durchlaufen wird, wenn gerade keine Interrupts anstehen. Man kann über eine bestimmte Gesamtzeitdauer hinweg

messen, wieviel Zeit der DSP mit dem Abarbeiten der Endlosschleife verbringt. Die Differenz ist dann die Zeit die der DSP wirklich gearbeitet hat. Setzt man diese Zeit in Relation zur Gesamtzeit, so hat man die prozentuale Auslastung des DSP's. Realisiert haben wir die Zeitmessungen so, dass ein Timer alle 0.5 Sekunden ein Interrupt generiert (Gesamtzeit). Innerhalb dieser Zeitdauer wird gezählt wieviele Schleifendurchläufe gemacht werden. Für die Leistung ergibt sich folgende Formel:

$$P_{\%} = \frac{t_{Gesamt} - AnzSchleifen \cdot t_{Schleife}}{t_{Gesamt}} \cdot 100\%$$

9.2.3 Grobe Zusammenhänge der Teilprogramme



9.3 Detailstruktur der DSP-Software

9.3.1 Speicheraufteilung

Erste Priorität bei der Speicherverteilung hat eindeutig die Filterroutine. Es muss beachtet werden, dass parallele „move“ Operationen bei der Filterung genutzt werden können und dass die verwendeten Bereiche des X und Y Memories im DSP internen Speicher liegen, so dass keine Bustransfers zum externen Memory nötig sind.

Programm-Memory Map des DSP's

P:\$0	Interrupt Vektoren
P:\$40	AD/DA-Wandler und SSI Initialisierungsroutine
	DSP Initialisierungsroutine
	Interrupt Serviceroutinen
	Initialisierungsroutinen für SOS Ringbuffer, SCI, Timer
	Hauptprogramm

Y-Memory Map des DSP's

Y:\$0	Koeffizientenringspeicher 1 für Filter links
Y:\$18	
X:\$20	Koeffizientenringspeicher 1 für Filter rechts
X:\$38	
X:\$40	Koeffizientenringspeicher 2 für Filter links
X:\$58	
X:\$60	Koeffizientenringspeicher 2 für Filter rechts
X:\$78	
X:\$79	Userstack
X:\$A0	
X:\$A1	Empfangsbuffer SCI
X:\$136	Sendebuffer SCI
.	
X:\$138	CW: Datenbyte Zähler SCI

X-Memory Map des DSP's

X:\$0	SSI In-Buffer
X:\$3	
X:\$4	SSI Out-Buffer
X:\$7	
X:\$10	Ringbuffer Datenspeicher SOS links
X:\$19	
X:\$20	Ringbuffer Datenspeicher SOS rechts
X:\$29	
X:\$30	CW: Anzahl SOS linker Kanal
.	CW: Anzahl SOS rechter Kanal
.	CW: Samplingrate
.	CW: Main-Schleifenzähler
	F: neue Daten im SCI Eingangsbuffer
	CW: Basisadresse Koeffizienten links
	CW: Basisadresse Koeffizienten rechts
	F: Empfang von Daten über SCI ein
	F: Daten werden über SCI empfangen
	CW: Zwischenspeicher für Register r4
	CW: Zwischenspeicher für Register m4
	CW: Ausrichtungsinfos für empfangenes Byte
	F: welchem Kanal die empf. Daten gelten
	CW: Kennung des aktuellen Filterkoeffizienten
	CW: Zielspeicherort für den Filterkoeffizienten
X:\$3F	F: nextes Datenbyte ist die Samplerate

CW = Kontrollwort

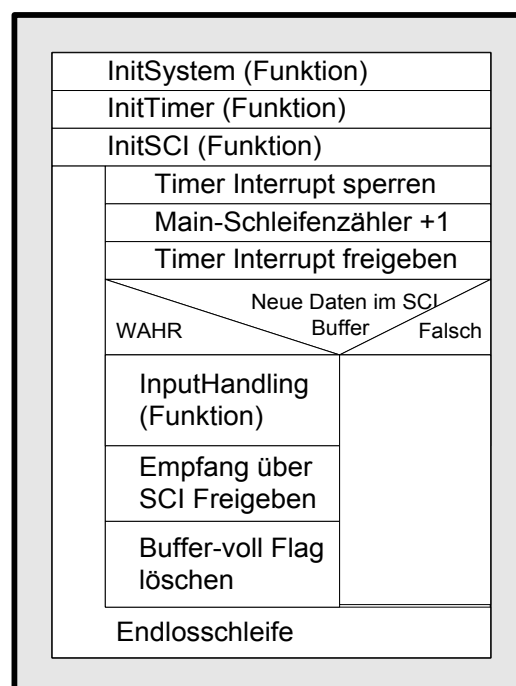
F = Flag

9.3.2 Registeraufteilung

Die Register des DSP's sind so aufgeteilt, dass in der Filter ISR möglichst wenige Register auf dem Userstack gesichert werden müssen. Die Register r0/m0/n0 bis r3/m3/n3 sowie r5/m5/n5 und der Akku a, stehen nach der Initialisierung ausschliesslich der Filter ISR zur Verfügung. R7 ist der Userstackpointer und r6 der Zeiger in den SCI-Eingangsbuffer.

Register	Verwendungszweck (Nach Initialisierung)
A	Akku für Filter ISR
B	allgemein verwendet
X	allgemein verwendet
Y	allgemein verwendet
R0	Zeiger in den Sendebuffer der SSI (Wird von der SSI TX-ISR verwendet)
R1	Zeiger in den Empfangsbuffer der SSI (Wird von der SSI RX-ISR, bzw. von der Filter ISR verwendet)
R2	Zeiger auf den Ringbuffer Datenspeicher der SOS für den linken Kanal (Wird von der Filter ISR verwendet)
R3	Zeiger auf den Ringbuffer Datenspeicher der SOS für den rechten Kanal (Wird von der Filter ISR verwendet)
R4	Frei zur Verfügung (Wird von der Filter ISR und dem Hauptprogramm verwendet)
R5	Zeiger auf den Ringspeicher der Filterkoeffizienten für die SOS des rechten Kanals (Wird von der Filter ISR verwendet)
R6	Zeiger in den Empfangsbuffer der SCI (Wird von der SCI RX-ISR verwendet)
R7	Userstackpointer

9.3.3 Hauptprogramm (main)



9.3.4 Initialisierungsfunktionen

9.3.4.1 Initialisierung des DSP's und DA/AD-Wandlers (InitSystem)

Interruptvektoren setzen
Taktfrequenz für DSP setzen
Interruptpriorität setzen SSI=2 SCI=1 Timer=0
SSI Input/Output-Buffer initialisieren
Stackpointer (r7) setzen
SetSOS (Funktion)
InitCodec (Funktion von Motorola)
Einstellungen für DA/AD-Wandler Ein/ Ausgänge vornehmen
SSI Synchronisieren und auf Filter ISR umschalten

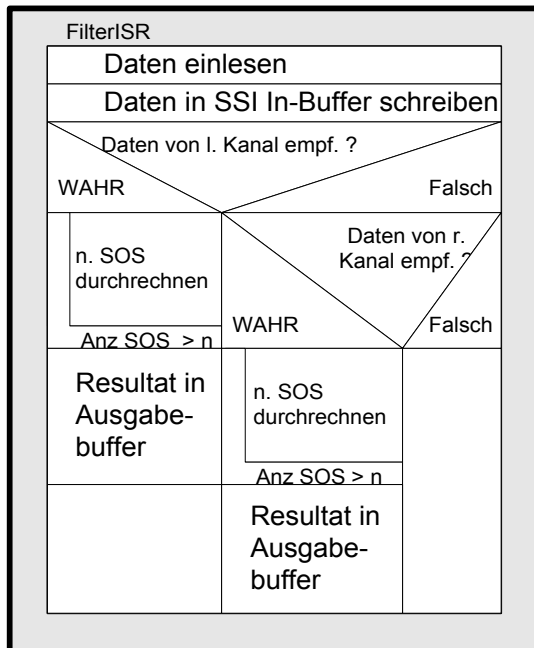
9.3.4.2 Setzen der Samplingrate (SetSampR)

Das setzen der Samplingrate ist leider nur möglich durch Neuinitialisierung der SSI und des DA/AD-Wandlers. Es ist deshalb nicht möglich, nach einer Übertragung der Filterkoeffizienten ein fließendes Umschalten von den alten zu den neuen Filterkoeffizienten durchzuführen.

9.3.4.3 Setzen des aktuellen Filterkoeffizienten Ringspeichers (SetSOS)

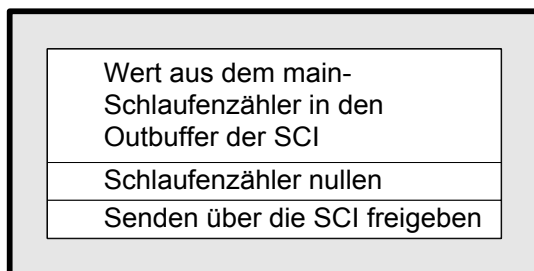
Datenringbuffer der SOS mit 0 initialisieren
Pointer auf den SOS Koeffizienten Ringspeicher setzen
Grösse des Koeffizienten Ringspeichers aktualisieren setzen
Pointer auf Datenringbuffer setzen
Datenringbuffer Grösse aktualisieren

9.3.5 Filterung (Interrupt Serviceroutine)



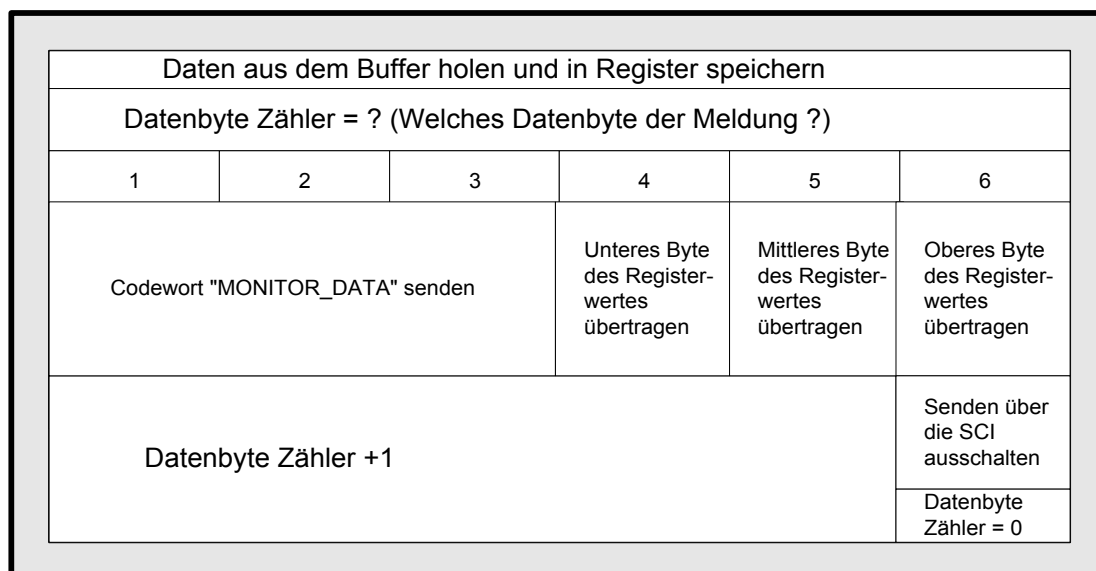
Die Filterung ist der einzige wirklich zeitkritische Teil unserer Software. Deshalb wurde die Berechnung des Ausgabewerts durch Verwendung von parallelen Move-Operationen optimiert.

9.3.6 Leistungsmonitor(TimerISR)



Wie bereits erwähnt wird die Leistung mit Hilfe eines Timers gemessen, der alle 0.5 Sekunden ein Interrupt erzeugt.

9.3.7 ISR für das Senden über die asynchrone serielle Schnittstelle

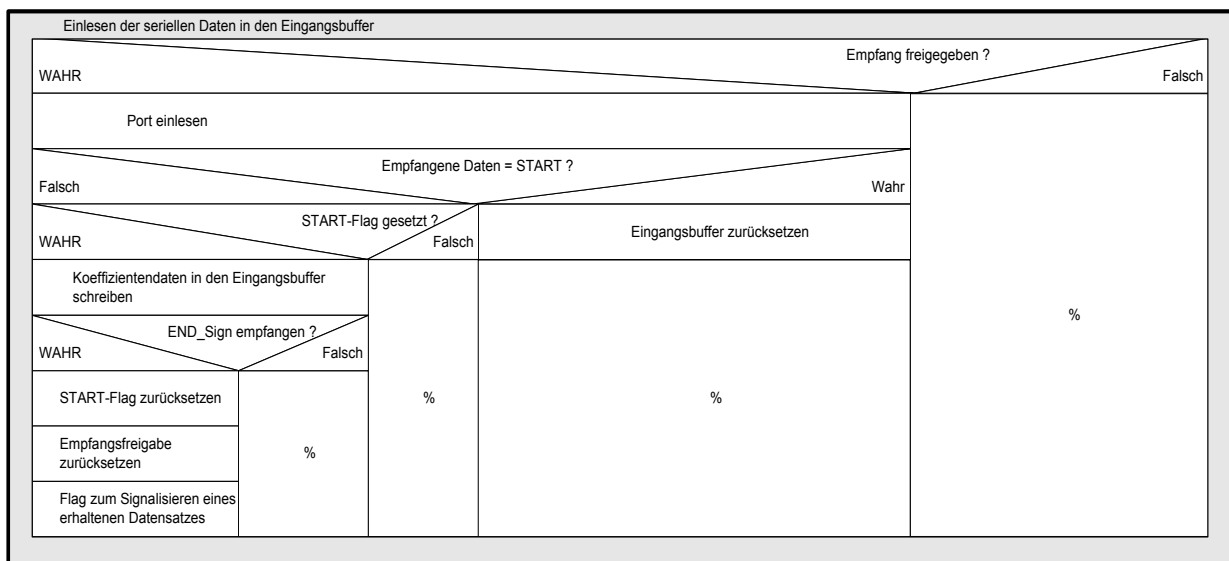


9.3.8 Übertragung

9.3.8.1 Einlesen der Daten vom seriellen Port in den Eingangsbuffer

Neue Daten können nur empfangen werden, wenn die Filtersoftware auf dem PC die Übertragung freigibt. Dies verhindert das Senden von Datenfragmenten. Beim Empfangen des START_Signs wird der Buffer im Y-Memory zurückgesetzt und das START-Flag gesetzt. Ist das START-Flag gesetzt, so können die Koeffizientendaten in die Variable „serInBuf“ im Y-Memory eingelesen werden. Am Ende der Übertragung wird das END_Sign empfangen. Dies bewirkt ein Rücksetzen des START-Flags, der Empfangsfreigabe und ein Setzen eines Flags, welches einen erhaltenen Datensatz signalisiert.

Struktogramm:



9.3.8.2 Schreiben der Koeffizienten in den entsprechenden Koeffizientenbuffer

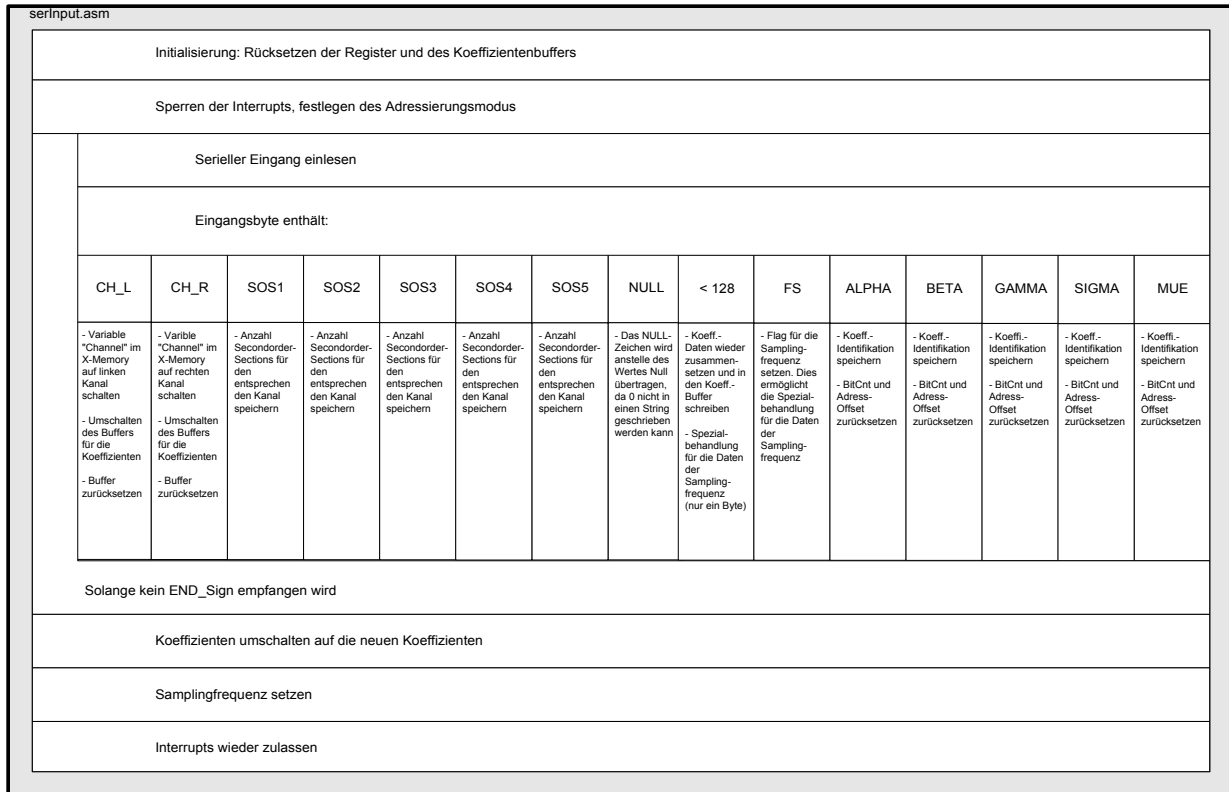
Die Daten im Eingangsbuffer werden aufgrund des Assemblercodes in dem File „serInput.asm“ behandelt. Zu Beginn des Files erfolgt die Initialisierung der Register, des Koeffizientenbuffers im Y-Memory, sowie die Festlegung des Adressierungsmodus (lineare Adressierung) und das Sperren der Interrupts. Danach erfolgt die Schleife mit der gesamten Datenbehandlung:

- CH_L : Gibt an, das die folgenden Koeffizienten für die Filterung auf dem linken Kanal bestimmt sind. Dies wird durch setzen eines Flags in der Variable „Channel“ im X-Memory angezeigt.
- CH_R : Gibt an, das die folgenden Koeffizienten für die Filterung auf dem rechten Kanal bestimmt sind. Dies wird durch setzen eines Flags in der Variable „Channel“ im X-Memory angezeigt.
- SOSX : Legt die Anzahl der zu übertragenden Second-Order-Sections fest und speichert diese in der entsprechenden Variable „Anz_SoS_L“ oder „Anz_SoS_R“ im X-Memory des DSP.
- NULL : Da die Übertragung des Wertes 0 Schwierigkeiten bereitet (z.B kann 0 nicht in einen String geschrieben werden), wird dieser Wert als ein Sonderzeichen gesendet.
- < 128 : Alle Steuerzeichen sind grösser oder gleich 128. Ein Wert kleiner als 128 entspricht folglich entweder einem Teil der Koeffizientendaten oder den Daten der Samplingfrequenz. Die Übertragung der Koeffizientendaten erfolgt 7-Bitweise. Der endgültige 24-Bit Wert eines Koeffizienten muss also zuerst noch zusammengesetzt werden. Dies erfolgt durch mehrmaliges Schieben und durch eine logische ODER Verknüpfung der übertragenen Bits.
Die Daten der Samplingfrequenz bestehen nur aus einem Byte und bewirken das Setzen des entsprechenden Flags SAMP_RATE_8, SAMP_RATE_16, SAMP_RATE_32, oder SAMP_RATE_48.
- FS : Das FS-Zeichen gibt an, das die folgenden Daten zur Samplingfrequenz gehören.
- ALPHA, BETA, GAMMA, SIGMA, MUE : Speichert die aktuelle Koeffizientenidentifikation und setzt den BitCnt (wird für das Zusammensetzen der Koeffizientendaten gebraucht) und den AdressOffset (zeigt auf die Speicherstelle der verschiedenen Second-Order-Sections) auf 0.

Die Schleife wird beendet, wenn ein END-Zeichen empfangen wird. Danach wird die Filterung umgeschaltet auf den Buffer mit den neuen Koeffizienten und die Samplingfrequenz gesetzt. Zuletzt werden alle Interrupts wieder zugelassen.

(Struktogramm siehe nächste Seite)

Struktogramm:



10 Messberichte

10.1 Messaufbau

Auszumessendes Objekt: IIR-Filter, realisiert mit DSP-Board DSP56002 EVM Rev. 2.2
Auf dem DSP läuft die von uns entwickelte Software

Daten des IIR-Filters: Realisiert wurde ein Chebyshev 1 Hochpass; Rippel im Durchlassbereich 1 dB; Grenzfrequenz 500Hz; Ordnung 10; Samplingrate 32KHz.

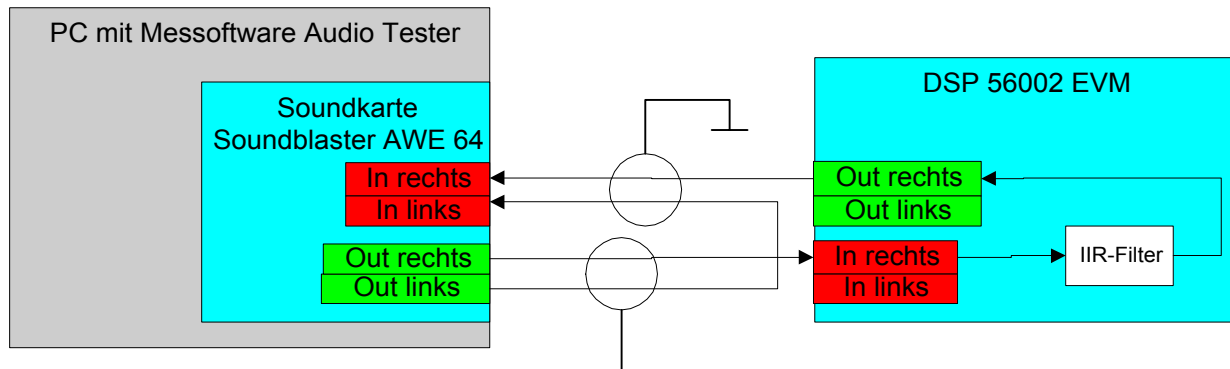
Messeinrichtung: PC mit Soundblaster AWE64: 16 Bit Auflösung; Samplingrate 44,1 KHz; Zum Ausmessen wurde die Software: „Audio Tester“ V:1.3b verwendet.

Einstellung der Software: Signal Outputlevel 60% Inputlevel 40% (Es treten gerade keine Verzerrungen auf); rechter Kanal = Messkanal, linker Kanal = Referenzkanal

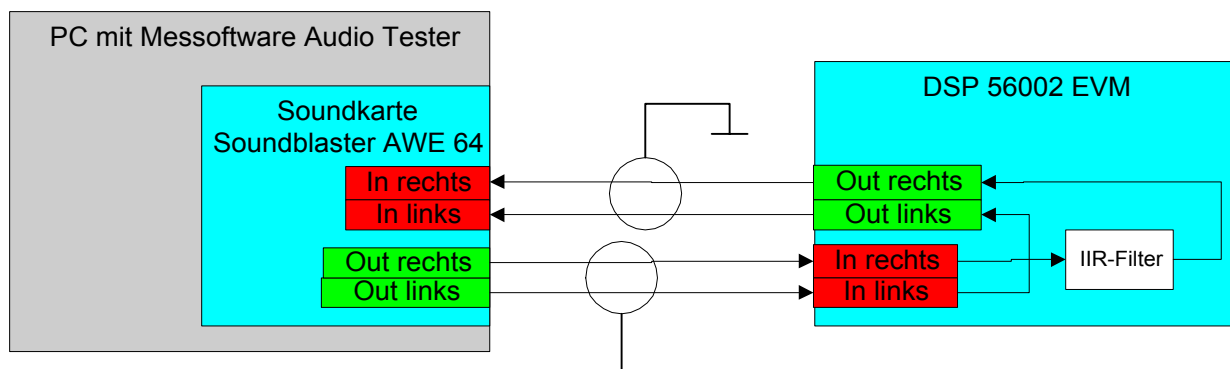
Datum der Messung: Zürich, 21.2.99

10.2 Ausmessung eines IIR-Filters

Schema des Aufbaus für Messung 1.5

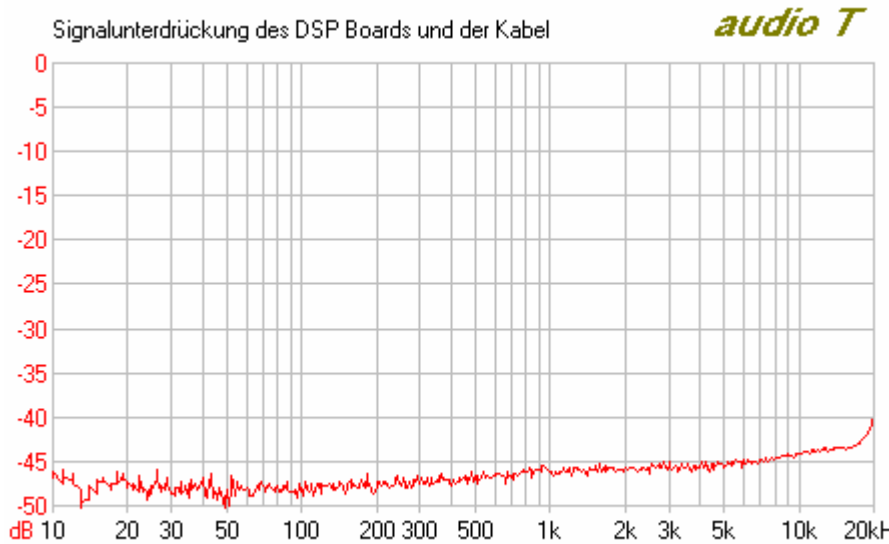


Schema des Aufbaus für Messung 6

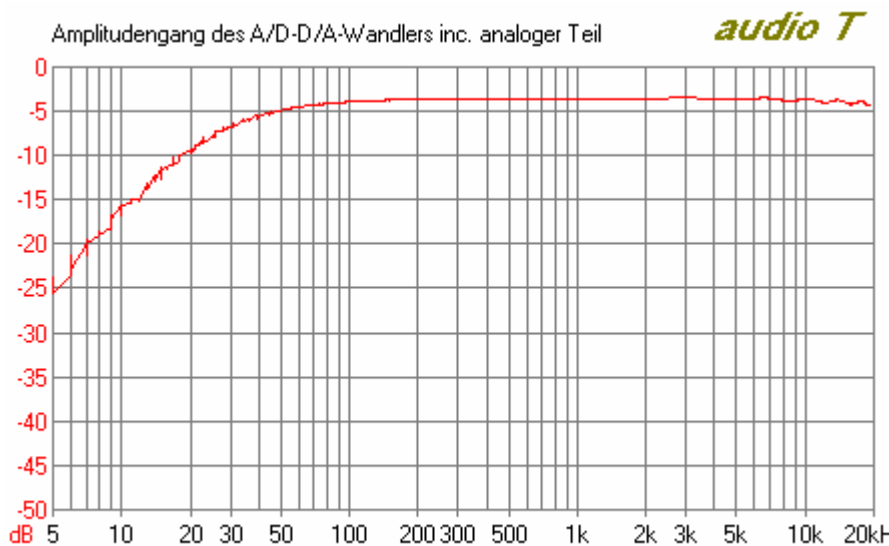


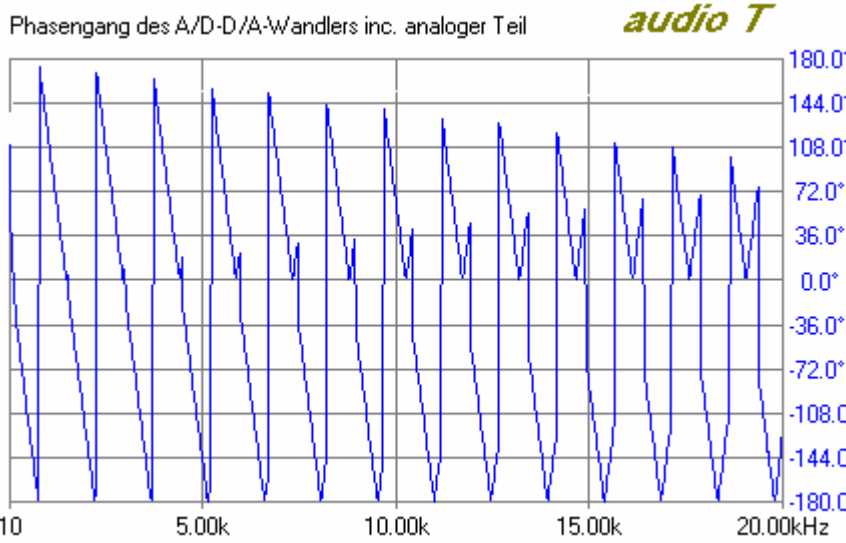
10.2.1 Messresultate

Messung 1: Maximale Signaldämpfung des Messaufbaues

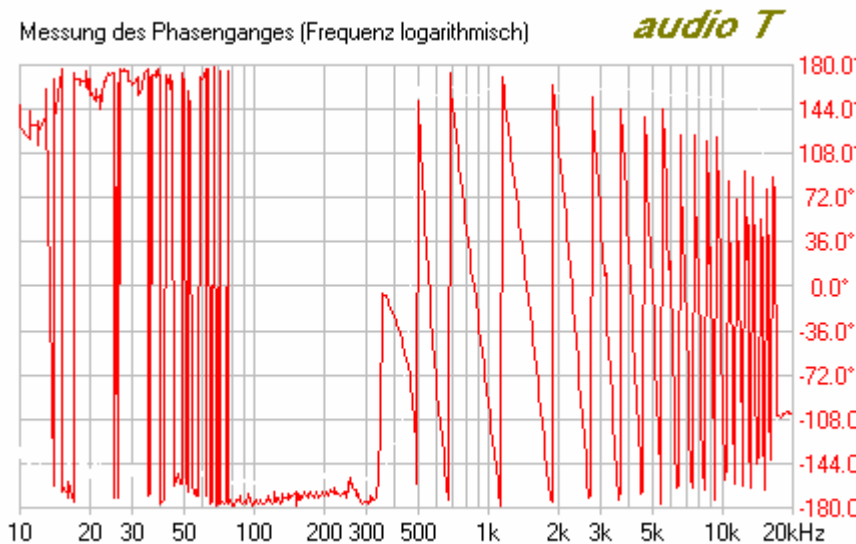
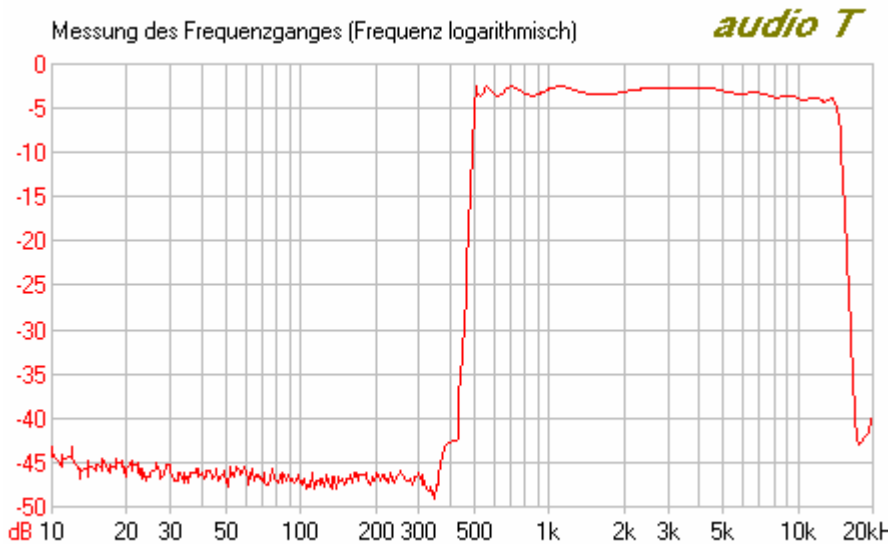


Messung 2: Übertragungsfunktion des Analogteils incl. Wandler

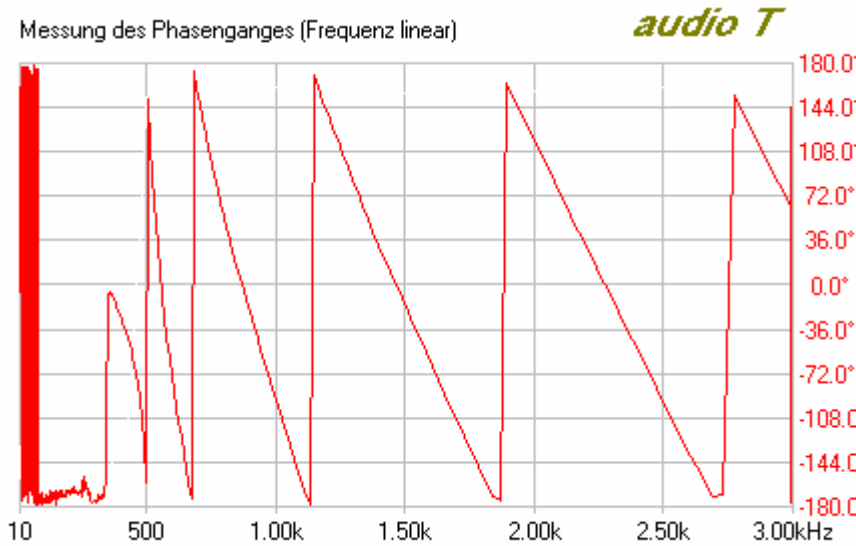
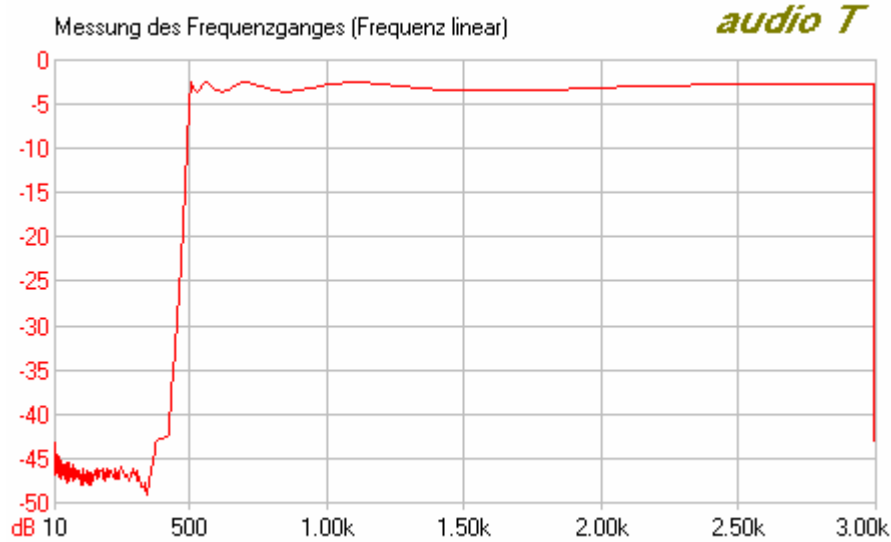




Messung 3: Gemessene Frequenz- und Phasengänge (Bodediagramm)

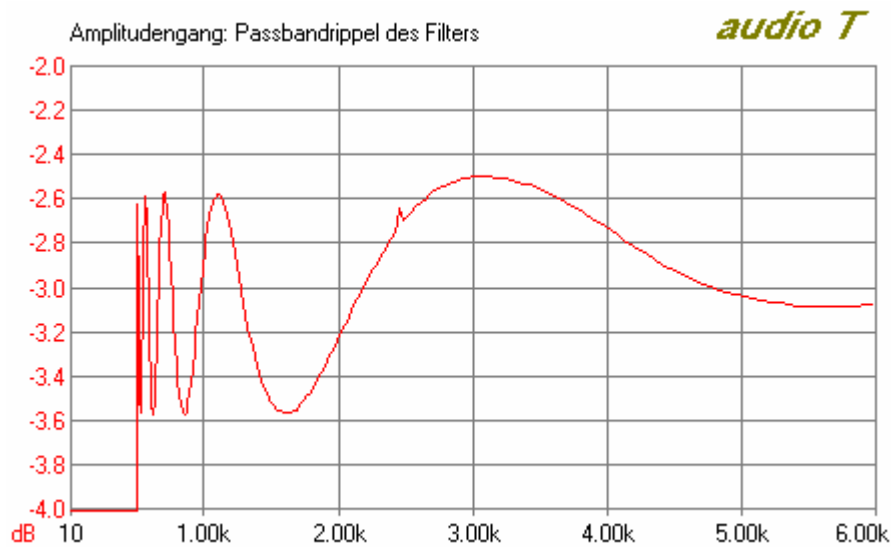


Messung 4: Gemessene Frequenz- und Phasengänge (Frequenz linear)



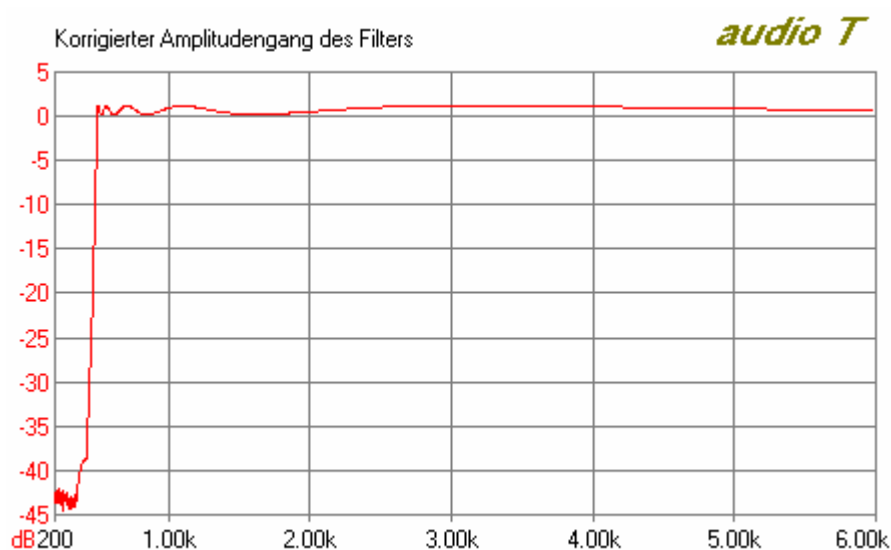
Messung 5: Passbandrippel

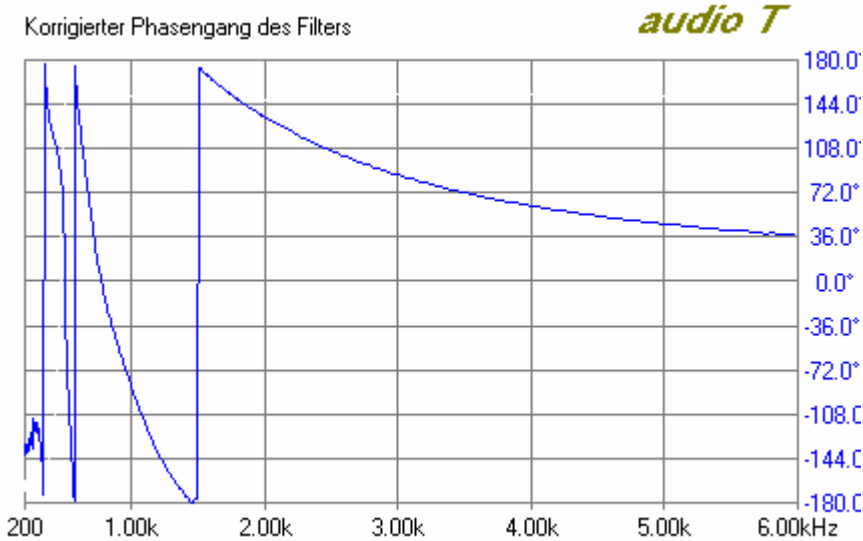
Bei dieser Messung wird die Gesamtübertragung des DSP-Boards gemessen, also inklusive Aliasingfilter, Vorverstärker und Sample/Hold-Glied etc.



Messung 6: Amplituden/Phasengang des Filters mit Korrektur

Bei dieser Messung wird der Einfluss des Analogteils sowie der Verzögerungen kompensiert, indem der Referenzkanal durch das DSP-Board geschleift wird.





10.2.2 Auswertung der Messung

Signaldämpfung des Aufbaus

Der Messaufbau hat eine Signaldämpfung von 40dB bis 50dB. Das schränkt den für die Messung zur Verfügung stehenden Dynamikbereich auf 40dB (2 Dekaden) ein. Signale die schwächer als 1/100 sind, können nicht mehr ausgewertet werden.

Übertragungsfunktion des Analogteils incl. Wandler

Der Amplitudengang ist relativ linear, Man erkennt jedoch, dass tiefe Frequenzen durch ein Hochpassfilter 1. Ordnung, welches ungefähr eine Grenzfrequenz von 30Hz aufweist, abgeschwächt werden. Dieses Hochpassfilter ist auf die Koppelkondensatoren zwischen Vorverstärkerstufe und A/D-Wandler auf dem DSP Board zurückzuführen. Zudem ist noch eine konstante Dämpfung von c.a. 3dB vorhanden. Sie entspricht der Gesamtverstärkung des Systems, die durch die Kopplungsfaktoren (Aufgrund der Ein- und Ausgangswiderstände) sowie durch den Vorverstärker bestimmt ist.

Der Phasengang des Systems ist linear. Dies deutet auf eine konstante Signalverzögerung im System hin. Die Signalverzögerung entsteht durch die Sample/Hold-Glieder im AD/DA-Wandler, durch die Übertragung von und zum DSP, sowie durch die benötigte Arbeitszeit im DSP. Zusätzlich erscheint bei höheren Frequenzen noch stückweise eine negative Zeitverzögerung. Dieses Phänomen ist uns unerklärlich, vermutlich handelt es sich um einen Fehler in der Messsoftware. Der Phasengang müsste deshalb zur Kontrolle noch mit einem anderen Messgerät gemessen werden.

Amplitudengang des Filters

Der Amplitudengang entspricht den Erwartungen. Unterhalb von 400 Hz verschwindet das Signal in der Messgenauigkeit bzw. im Rauschen. Darauf folgt die Flanke des Filters und bei 500Hz ist das Signal im 1 dB breiten Durchlassbereich. Nachfolgend sind noch die, für die Chebyshev Approximation charakteristischen Maxima und Minima der Übertragungsfunktion zu erkennen. Bei c.a. 13 kHz wird das Signal durch die antialiasing Filter des AD/DA-Wandlers unterdrückt. Der Einfluss des antialiasing Filters ist

auch schon früher zu erkennen. In der Detailmessung des Passbandrippels ist ein scheinbar linearer Anstieg des Amplitudenganges zu sehen. Er wird durch ein überlagertes Passbandrippel des Antialiasing-Filters hervorgerufen.

Phasengang des Filters

Der Phasengang des eigentlichen IIR-Filters ist erst in der korrigierten Messung (Messung 6) zu erkennen. Wie erwartet ist der Phasengang des Filters nichtlinear. Die Gesamtphasendrehung beträgt c.a. $5 \text{ Pi} = 10 * \text{Pi}/2$ was einem minimalphasigen Filter 10. Ordnung entspricht.

10.2.3 Kommentar zur Messung

Die Messung zeigt, dass sich das IIR-Filter wie erwartet verhält. Die Messung stimmt relativ genau mit der Theorie überein. Man erkennt jedoch auch den Unterschied zwischen Messung und Theorie auf Grund von zusätzlichen Bandbreitenbegrenzungen im Analogteil sowie durch Verzögerungen im Digitalteil.

Die Hauptschwierigkeit der Messung war der relativ kleine Dynamikbereich. Dies ist vor allem darauf zurückzuführen, dass uns keine geeignete Messapparatur zur Verfügung stand. Wir probierten verschiedenen Möglichkeiten aus. Zuerst verwendeten wir einen digitalen Spektrumanalyzer und einen Rauschgenerator. Leider wies auch der Spektrumanalyzer keinen sehr hohen Dynamikbereich auf, zudem war die Auflösung und Darstellung auf dem Display des Spektrumanalyzers inakzeptabel. Anschliessend versuchten wir den Frequenzgang mit KO und Sinusgenerator auszumessen. Die Messung erwies sich aber als zu aufwendig und das Signal war bei kleinen Amplituden zu stark mit Rauschen überlagert. Deshalb entschlossen wir uns für die Messung mit Hilfe von PC und Soundkarte, wie sich herausstellte war dies die beste der drei Lösungen. Aber auch bei dieser Messung kämpften wir mit eingestreutem Rauschen sowie Kopplung zwischen linkem und rechtem Kanal. Es wäre vielleicht von Vorteil, wenn sich das DS-Labor ein geeigneteres Messgerät zur Frequenzgangmessung anschaffen würde.

10.3 Ausmessung des Leistungsmonitors

10.3.1 Gemessenen Werte

Samplingrate	1 SOS	2 SOS	3 SOS	4 SOS	5 SOS
48 kHz	38 %	41 %	43 %	47 %	51 %
32 kHz	22 %	27 %	28 %	29 %	32 %
16 kHz	9 %	10 %	11 %	12 %	13 %
8 kHz	2 %	2 %	3 %	3 %	3 %

10.3.2 Kommentar zur Messung, Auswertung

Bei 5 Second-Order-Sections (SOS) pro Kanal und einer Samplingrate von 48kHz ist das System zu ca. 50 % ausgelastet. Auf den ersten Blick scheint so noch genügend Rechenleistung zur Verfügung zu stehen, um Filter mit 10 SOS oder einer Ordnung von 20 zu realisieren. Das Problem ist, dass der DA/AD-Wandler immer 4 Datenwörter pro Übertragungszyklus überträgt, wobei ein Übertragungszyklus $1/48\text{kHz} = 20,8 \mu\text{s}$ beträgt. Das erste Datenwort ist das Sample des linken Kanals, das 2. Datenwort das Sample des rechten Kanals und anschliessend folgen noch 2 Statuswörter, bis sich das Ganze wiederholt. Somit wird die Interruptroutine für den SSI 4 mal pro $20,8 \mu\text{s}$ (alle $5,2 \mu\text{s}$) aufgerufen. Dabei können nur die Hälfte der Zeit Daten gefiltert werden. Wird mehr als $5,2 \mu\text{s}$ für die Filterung pro Kanal benötigt, so wird der folgende Datenwert im SSI Eingangsregister überschrieben und die Synchronisation geht verloren. Der DSP ist praktisch somit schon bei 50 % ausgelastet.

Wird die Samplingrate jedoch reduziert, so ist die zur Filterung pro Kanal zur Verfügung stehende Zeit grösser (32 kHz => $7,8 \mu\text{s}$; 16 kHz => $15,6 \mu\text{s}$; 8 kHz => $31,2 \mu\text{s}$). Folglich ist bei gleicher Anzahl SOS die prozentuale Auslastung kleiner.

11 Probleme bei der Realisierung

11.1 Probleme bei der Koeffizientenberechnung

Es gab zwei Hauptprobleme bei der Berechnung der Koeffizienten. Beide hatten mit der Berechnung des zeitkontinuierlichen Filters zu tun. Da wir uns noch nie mit dieser Materie beschäftigt haben, mussten wir uns zuerst die Theorie der zeitkontinuierlichen Filter aneignen. Das Problem war genaue Beschreibungen der Filterapproximationen (Chebyshev, Cauer und Butterworth) zu finden. In den meisten Büchern wird dieses Thema nur sehr oberflächlich und unvollständig behandelt. Meist wird nur kurz die zugrunde liegende Formel erwähnt und manchmal ist auch eine Tabelle mit den Koeffizienten für Filter verschiedener Ordnungen vorhanden. Eine Ausnahme war hier das Buch von Tietze und Schenk [ts]. Wir fanden dort Formeln für Chebyshev 1 und Butterworthfilter. Leider fanden wir nirgends eine Herleitung oder Begründung für die Chebyshev Polynome.

Das zweite Problem war die Entwicklung eines Bandpasses bzw. einer Bandsperre. Bei der Umwandlung vom Tiefpassdesign zu einem Bandpass- bzw. Bandsperredesign wird nämlich die Ordnung des Filters verdoppelt. Es entstehen somit im Nenner der Übertragungsfunktion Produkte von Polynomen 4. Ordnung. Die Zerlegung dieser Polynome ist in C++ problematisch, da die Nullstellen dieser Polynome ausschliesslich komplex sind. Es gibt zwei Möglichkeiten dieses Problem zu lösen, entweder man versucht die komplexen Berechnungen in C++ zu implementieren oder man führt die Berechnung extern mit Hilfe einer Mathematiksoftware durch. Die Kommunikation zwischen der eigenen Software und Mathematiksoftware findet dann über eine spezielle Softwareschnittstelle statt. Wir werden Versuchen die Berechnung des Bandpasses direkt in unsere Software zu implementieren, da wir keine Erfahrung mit der Programmierung der Softwareschnittstelle zu Mathematica oder Matlab haben.

11.2 Probleme und Schwierigkeiten bei der Implementation des GUI

Das grösste Problem bei der Implementation des graphischen User-Interfaces stellte die mangelnde Erfahrung in der Windows-Programmierung dar. So gestaltete sich die Entwicklung des GUI's zu einem „learning by doing“- Prozess.

Der Beginn war relativ harzig, da der Programmier-Stil mittels eines main-files noch sehr tief in uns verankert war. Mit der Zeit gewöhnten wir uns aber an die Windows-Programmierung mittels Messages.

Gegen Ende der Arbeit stellte sich die Frage nach dem Bedienungskomfort. Wir wollten ein Bild in die Applikation einfügen, welches die Eingabe-Parameter anschaulich darstellt. Das Finden des Optimums von Speicherplatz, Bitmapgrösse und Lesbarkeit des Bildes war äusserst schwierig. Auch mussten diverse Begrenzungen und Umrechnungen eingeführt werden, um Eingaben, bei welchen die Koeffizientenberechnung zu Fehlern führen könnte, zu verhindern. (z.B.: $f_g > f_{sperr}$ beim Tiefpass oder $f_g < f_{sperr}$ beim Hochpass).

Der einzige uns bekannte Schönheitsfehler ist die Eingabemöglichkeit eines idealen Filters ($f_g = f_{sperr}$). Die Berechnung der optimalen Filterordnung für dieses Filter wird ermöglicht, indem wir eine Abweichung von 0.0000001 zwischen den beiden Frequenzen einführen. Diese Abweichung spielt insofern keine Rolle, da unsere maximale Filterordnung auf 10 beschränkt ist.

11.3 Probleme und Schwierigkeiten bei der Implementation der Übertragung

Da Visual C++ nur sehr grundlegende Funktionen zur Ansteuerung der seriellen Schnittstelle zur Verfügung stellt und zusätzlich deren Beschreibung in MSDN von VC++ 6.0 ins Uferlose führt, sahen wir uns aufgrund des Zeitmangels gezwungen, eine fremde Klasse zur Hilfe zu nehmen. Herr R. Spekreijse stellte uns diese Klasse zur Verfügung und so konnten wir nach einigen Tagen der Frustration endlich unsere Arbeit fortsetzen.

Bei der Initialisierung der seriellen Schnittstelle auf dem DSP-Board mussten wir nach längerem Suchen feststellen, dass Motorola ein fehlerhaftes Beispielprogramm publiziert hat. Die Schnittstelle wurde aufgrund eines falschen Wertes im Teilerregister mit einer falschen Baudrate initialisiert.

12 Ausblick und Visionen für den Filterdesigner

12.1 Mögliche Erweiterungen

- Einstellen der Verstärkung am Eingang und Ausgang (Mikrophon/line-in...Kopfhörer/line-out)
- Bandsperre/Bandpass als Auswahlmöglichkeit bei der Filtercharakteristik, wobei eine Bandsperre oder ein Bandpass relativ einfach realisiert werden könnte. Man müsste den rechten und den linken Kanal hintereinander schlafen. So könnte durch einstellen von Hoch- und Tiefpässen Bandsperren oder Bandpässe realisiert werden, welche allerdings dann nur in Monoqualität zur Verfügung stehen würden.
- Chebyshev 2 / Cauer (elliptisch) als Auswahlmöglichkeit beim Filtertyp
- Kaskadierung von Second-Order-Sections verschiedener Filtercharakteristiken

12.2 Gesamt - Konzept

Die Filter-Software könnte in ein EPROM gebrannt und hardware-mässig auf dem EVM-Board verankert werden. Dies würde ein Herunterladen der Software beim Starten des Programmes und das Umstecken des RS-232 Kabels erübrigen.

13 IIR-FILTERDESIGNER MANUAL

13.1 Einführung

Der Filterdesigner ist eine Applikation, die es ermöglicht, Koeffizienten von Infinite Impulse Response – Filtern auf Grund von diversen Parametern zu berechnen. Die berechneten Koeffizienten können auf das EVM-Board (bestückt mit dem Motorola DSP 56002) heruntergeladen werden. Auf dem Board wird das Eingangssignal durch die Second-Order Sections des IIR-Filters in Echtzeit umgeformt.

13.2 Beschreibung der einzelnen Parameter

Menu-Auswahl Com1/Com2

Es kann die serielle Schnittstelle definiert werden, über welche die Daten auf das DSP_Board geladen werden sollen. Default-Schnittstelle ist Com1.

Menu-Auswahl Leistungsmonitor

Der Leistungsmonitor für den DSP56002 wird angezeigt.

Grenz- und Sperrfrequenz

Die Grenzfrequenz gibt an, bei welcher Frequenz der Durchlassbereich auf -3dB abgefallen ist. Die Sperrfrequenz markiert den Beginn des Sperrbereiches.

Beim Tiefpass-Filter muss folglich die Grenzfrequenz immer *kleiner* sein als die Sperrfrequenz und beim Hochpass-Filter immer *grösser*.

Zur Verhinderung von Aliasing – Fehlern sind die Grenzfrequenz und die Sperrfrequenz auf die halbe Samplingfrequenz begrenzt.

Samplingfrequenz

Mit der Samplingfrequenz kann die Abtastrate des Eingangssignales eingestellt werden.

8 kHz entspricht dem Sprachstandard,

44.1kHz wäre CD-Qualität,

48 kHz entspricht dem Standard eines Audio-Studios oder dem DAT.

Filter-Art

Die Filter-Art beschreibt, ob alle Frequenzen unter der Grenzfrequenz (Tiefpass), oder alle Frequenzen über der Grenzfrequenz durchgelassen werden sollen (Hochpass).

Beim Umschalten von Tief- auf Hochpass, und umgekehrt, wird die Grenzfrequenz beibehalten und die Sperrfrequenz um das gleiche Delta auf die entsprechende Seite angepasst.

Filter-Bauweise

Die Filter-Bauweise beschreibt, nach welcher Approximationsmethode die Filter designed werden.

Dämpfung

Bei der Bauweise Butterworth, kann nur die minimale Dämpfung (in dB) im Sperrbereich angegeben werden.

Bei der Bauweise Chebyshev 1 hingegen, kann auch die maximale Dämpfung im Durchlassbereich definiert werden.

Filterordnung

Mit der Filterordnung, kann definiert werden, wie viele Filter hintereinander kaskadiert werden sollen.

Die Kaskadierung erfolgt in sogenannten Second-Order-Sections. D.h. es werden immer zwei Filter zusammengefasst. Dies verhindert gröbere Ungenauigkeiten auf Grund der Quantisierung.

Mit dem Knopf „Optimum“ kann die optimale Filterordnung berechnet werden. Zu beachten ist, dass die maximale Ordnung bei 10 liegt.

Koeffizienten-Knopf

Alle Koeffizienten werden berechnet, und in einem separaten Dialogfenster angezeigt.

DSP Kanal links-Knopf

Alle Koeffizienten werden berechnet, und auf den linken Kanal des EVM-Boards heruntergeladen.

DSP Kanal rechts-Knopf

Alle Koeffizienten werden berechnet, und auf den rechten Kanal des EVM-Boards heruntergeladen.

14 Rückblick auf die Semesterarbeit

Die Semesterarbeit war vielseitig und lehrreich. Wir konnten viele Erfahrungen in den Bereichen Filtertheorie, Filterdesign (analog/digital), diskrete Signalverarbeitung, Windows- Software Entwicklung, DSP – Programmierung und serielle Kommunikation sammeln.

Für Frustration sorgten MSDN 6.0 (VC++ Help), fehlerhafte DSP-Dokumentation von Motorola und unzulängliche Messinstrumente um das IIR-Filter detailliert auszumessen. Mit Herrn Ehrensperger hatten wir aber jederzeit einen kompetenten Ansprechspartner zur Verfügung. Er konnte uns meist die nötigen Informationen oder Hinweise geben, um weiter zu kommen.

Da wir über keine Kenntnisse der Filtertheorie (analog wie digital) verfügten, mussten wir uns somit erst in diesen Themenkreis einarbeiten. Zusätzlich erstellten wir ein relativ umfangreiches GUI und implementierten die gesamte serielle Übertragung zwischen PC und DSP selber. Dies führte dazu, dass wir einiges mehr als 8 Stunden pro Woche in diese Semesterarbeit investieren mussten. Auch konnten einige zusätzlich geplante Features wie Bandsperre und Bandpass nicht mehr realisiert werden.

Trotzdem empfanden wir die Semesterarbeit als sinnvoll und interessant.

15 Literaturliste

[os]: „Discrete-Time Signal Processing“ Alain V. Oppenheim, Ronald W. Schaffer
 ISBN: 0-13-216771-9

[lbj]: „Digital Filters and Signal Processing SECOND EDITION“ Leland B. Jackson
 ISBN: 0-89838-276-9

[ts]: „Halbleiter-Schaltungstechnik“ U. Tietze, Ch. Schenk
 ISBN: 3-540-56184-6

[mk]: „Handbook for Digital Signal Processing“ Sanjit K. Mitra, James F. Kaiser
 ISBN: 0-471-61995-7

[rb]: „Handbuch der analogen und digitalen Filterungstechnik“ Roland Best
 ISBN: 3-85502-148-1

[mot]: MOTOROLA:

- „Implementing IIR/FIR Filters with Motorolas DSP56000/DSP56001“
- „DSP56000 Digital Signal Processor Family Manual“
- „DSP56002 Digital Signal Processor User Manual“
- „DSP Applications with Motorolas DSP56002“ Mohamed El-Sharkawy, Ph.D.