Ernst-Abbe-Fachhochschule Jena
Hochschule für angewandte Wissenschaften

Fachbereich Elektrotechnik und Informationstechnik

Bachelorarbeit

zur Erlangung des akademischen Grades

Bachelor of Engineering (B.Eng.):

Implementierung einer Klangregelung durch Shelving-Filter auf einem Signalprozessor mit Steuerung der Parameter über einen Mikrocontroller

eingereicht von:	Jürgen Döffinger
geboren am:	14.09.1976 in Leipzig
Matrikel-Nummer:	63 15 51
Studiengang:	Kommunikations- und Medientechnik
Hochschulbetreuer:	Prof. Dr.-Ing. Frank Giesecke
Datum der Themenausgabe:	02.10.2013
Datum der Abgabe:	06.01.2014

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ADC	Analog-Digital-Wandler (Analog-Digital-Converter)
AJAX	Asynchronous JavaScript and XML
API	Programmierschnittstelle (Application Programming Interface)
ASCII	American Standard Code for Information Interchange
ASPP	Audiosignalverarbeitungsprogramm (Audio Signal Processing Program)
AT	Ansprechzeit (Attack time)
AU	Audio Unit
BUS	Binary Unit System
CPU	Central Processing Unit
CU	Control Unit
CUIP	Control and User Interface Program
DAC	Digital-Analog-Wandler (Digital-Analog-Converter)
DAI	Digital Audio Interface
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DSK	Developer Starter Kit
DSP	digitaler Signalprozessor (Digital Signal Processor)
DU	Display Unit
FIR	Filter mit endlicher Impulsantwort (Finite Impulse Response)
GPIO	General Purpose Input/Output
GUI	grafische Benutzerschnittstelle (Graphical User Interface)
HPI	Host Port Interface
HTTP	Hypertext-Übertragungsprotokoll (Hypertext Transfer Protocol)
I^2C	Inter-Integrated-Circuit
IDE	integrierte Entwicklungsumgebung (Integrated Development Environment)
IIR	Filter mit unendlicher Impulsantwort (Infinite Impules Response)
IP	Internet Protocol
ISO	Internationale Organisation für Normung (International Organization for Standardization)
ISR	Interrupt Service Routine
LCD	Flüssigkristallanzeige (Liquid Crystal Display)
LT	Begrenzer (Limiter)
LUT	Nachschlagetabelle (LookUp Table)
MATLAB	MATrix LABoratory

McBSP	Multichannel Buffered Serial Port
MCU	Microcontroler Unit
PAP	Programmablaufplan
PC	Personal Computer
RT	Rücklaufzeit (Release time)
SPI	Serial Peripheral Interface
TC	Klangregelung (Tone Control)
TCGUIP	Tone Control Graphical User Interface Program
TCP	Übertragungssteuerungsprotokoll (Transmission Control Protocol)
TCWIP	Tone Control Webinterface Program
UDP	Benutzerdatensegmentprotokoll (User Datagram Protocol)
URI	einheitlicher Bezeichner für Ressourcen (Uniform Resource Identifier)
UI	Benutzerschnittstelle (User Interface)
WI	Webinterface

Formelzeichenverzeichnis

Formelzeichen

a_k	k-ter Koeffizient des vorwärtsgerichteten FIR-Filters
AT	Ansprechzeit (Attack time)
b_k	k-ter Koeffizient des rückwärtsgerichteten FIR-Filters
B	Balancewert
c	Zustands- oder Hilfsvariablen
f	Frequenz
f_c	3-dB-Grenzfrequenz (Cuttoff frequency)
f_g	Grenzfrequenz
f_{go}	obere Grenzfrequenz
f_{gu}	untere Grenzfrequenz
f_{lt}	Steuerwert innerhalb des Limiters
f_m	Mittenfrequenz (Center frequency)
f_{max}	maximal abzutastende Frequenz
f_N	Nyquist-Frequenz
f_s	Abtastfrequenz (Sample frequency)
g	linearer Verstärkungsfaktor
G	logarithmischer Verstärkungsfaktor in dB
$G(f)$	logarithmischer Verstärkungsfaktor in Abhängigkeit der Frequenz f in dB
$\underline{H}(f)$	komplexe Übertragungsfunktion
lt	Limiterschwelle (Threshold) in Prozent
LT	Limiterschwelle (Threshold)
M	Ordnung des rückwärtsgerichteten FIR-Filters eines IIR-Filters
n	Zähler
N	Ordnung des vorwärtsgerichteten FIR-Filters
N_Q	Anzahl der Quantisierungsstufen
N_W	Wortbreite des diskretisierten Audiosignals
ΔQ_f	Wertebereich des Quantisierungsfehlers
RT	Rücklaufzeit (Release time)
SNR	Signal-Rausch-Abstand (Signal-to-Noise-Ratio)
t_a	Ansprechzeit (Attack Time) in μs
t_F	Ein- und Ausblendzeit
t_r	Rücklaufzeit (Release Time) in ms

t_{se} Einstellzeit (Settling time)
T_s Periodendauer der Abtastfrequenz f_s
ΔU_{ADC} Aussteuerungsbereich des Analog-Digital-Wandlers
$U_{ADC_{max}}$ Maximalwert der positiven Eingangsspannung des *ADC*
$U_{ADC_{min}}$ Minimalwert der Eingangsspannung des *ADC*
ΔU_Q Quantisierungsintervall
W_o obere Grenze des anwendbaren Zahlenbereiches
W_Q Fehlerleistung des ADC
W_S Leistung eines sinusförmigen Signals
W_u untere Grenze des anwendbaren Zahlenbereiches
$x(n)$ zeitdiskretes Eingangssignal
x_p Spitzenwert
$y(n)$ zeitdiskretes Ausgangssignal
τ_G Gruppenlaufzeit
$\tau_G(f)$ Gruppenlaufzeit in Abhängigkeit der Frequenz f
$\tau_G(\omega)$ Gruppenlaufzeit in Abhängigkeit der Kreisfrequenz ω
$\varphi(f)$ Phasengang in Abhängigkeit der Frequenz f
$\varphi(\omega)$ Phasengang in Abhängigkeit der Kreisfrequenz ω
ω Kreisfrequenz $\omega = 2\pi f$
ω_c 3-dB-Kreisgrenzfrequenz

Indizes

ADC Analog-Digital-Wandler (Analog-Digital-Converter)
b Umgehung (Bypass)
B Balance
d Deakzentuierung (De-Emphasis)
DAC Digital-Analog-Wandler (Digital-Analog-Converter)
en aktivieren/deaktivieren (enable/disable)
F Ein-/Ausblenden (Fade in/out)
hp Höhen-Shelving-Filter (Highpass)
l linker Kanal (left channel)
lp Tiefen-Shelving-Filter (Lowpass)

lt	Begrenzer (Limiter)
m	Stummschaltung (Mute)
r	rechter Kanal (right channel)
V	Lautstärke (Volume)

1. Einleitung

Bereits während der Zeit des Studiums beschäftigte sich der Verfasser der Arbeit mit der Thematik der Audiosignalverarbeitung, aber auch das Interesse an der Signal- und Systemtheorie waren Grundlage für diese Arbeit. Als Grundlage seien hier Projektarbeiten im Fach „Programmierbare Logik“ und „Signalprozessoren“ zu nennen, bei denen bereits auf die Verfahren der digitalen Signalverarbeitung zurückgegriffen wurde um Audiodaten zu bearbeiten. Hinzu kommt der ständige Ärger über Aufnahmen in denen Frequenzbereiche zu stark oder nur kaum zu hören sind. Daher war der Entschluss naheliegend eine Klangregelung zu bauen.

Die Klangregelung soll durch Anheben oder Absenken der Amplituden die typischen Frequenzbereiche so verändern, dass eine Verbesserung des Klangeindrucks beim Hörer erreicht wird. Dazu kommen Shelving-Filter zum Anheben oder Absenken der tiefen- und hohen Frequenzen zur Anwendung. Die Grenzfrequenzen sind frei wählbar und nicht, wie in den meisten Systemen feststehend. So kann der Klangregler noch genauer an die Bedürfnisse des Hörers angepasst werden. Eine Alternative dazu wäre ein Equalizer.

Auf die Problematik der binären Rechenoperationen bei Festkommazahlen wird durch Wandlung in Fließkommazahl und Begrenzung durch einen Limiter Rechnung getragen. Der Limiter passt die Amplituden bei Überschreitung des Wertebereiches des Audio-Codecs so an, dass es beim Audio-Codec zu keinem Überlauf kommt, was sonst zu Verzerrungen führen würde.

Für eine mögliche zukünftige Erweiterung des Klangreglers mit weiteren Filtern und/oder Effekten wurde die Steuerung aus dem Signalprozessor in einen Mikrocontroller ausgelagert und die Programmierung dahingehend gestaltet, dass Filter oder andere Effekte leicht integrierbar sind. Der Mikrocontroller verbindet die Benutzerschnittstellen mit dem Signalprozessor in dem die Parameter der Audiosignalverarbeitung vom Mikrocontroller zum Signalprozessor über eine I^2C-Schnittstelle übertragen werden.

Zur Einstellung des Klangreglers stehen mehrere Benutzerschnittstellen zur Verfügung. Es ist möglich Einstellungen über Inkrementalgeber und Taster am Gerät vorzunehmen. Weiter wird ein Programm für die Betriebssysteme Windows und Linux bereitgestellt, welches über eine Ethernet-Verbindung (UDP) Anweisungen an die Steuerung sendet und von ihr den Zustand der Klangregelung abfragt, um diese anzeigen zu können. Genauso funktioniert auch das Webinterface, welches in einem Webbrowser ausgeführt wird. Der Datenaustausch erfolgt beim Webinterface über das HTTP-Protokoll.

Im ersten Teil der Arbeit werden die Grundlagen für die Audiosignalverarbeitung und die Steuerung zusammengetragen, um diese in einer Simulation zu testen, Der Test soll zeigen, dass die in den Grundlagen festgelegten Verfahren praxistauglich sind. Im zweiten Teil der Arbeit wird erläutert, wie der Klangregler praktisch aufgebaut ist und die Software arbeitet. Der letzte Teil gibt die Erkenntnisse aus den Messungen und Tests wieder.

2. Grundlagen

Eine Klangregelung dient der Beeinflussung eines Audiosignals zur subjektiven Verbesserung des Höreindrucks. Dazu kann der Klangregler die Amplituden der im Ton enthaltenen Frequenzen, die Phasen der Frequenzen zueinander oder die Zusammensetzung des Tones aus verschiedenen Frequenzen verändern. Der hier zur Anwendung kommende Klangregler beeinflusst die Amplituden der im Ton enthaltenen Frequenzen. Durch die Art der gewählten Filter, wird aber auch die Phase beeinflusst.

Die Grundlage der Klangregelung ergibt sich aus der Psychoakustik. Hier sind die Grenzen der Hörbarkeit des menschlichen Ohres zu berücksichtigen. Das menschliche Ohr kann Töne in einem Frequenzbereich von etwa 20 Hz bis 20000 Hz wahrnehmen. Wie laut ein Ton mit einer bestimmten Frequenz wahrgenommen wird, hängt von der Hörschwelle des Ohres für die jeweilige Frequenz ab. Diese Hörschwelle ist allerdings von Mensch zu Mensch unterschiedlich. Bild 1 zeigt verschiedene Kurven gleicher Lautstärke und die durchschnittliche Hörschwelle des menschlichen Ohres.

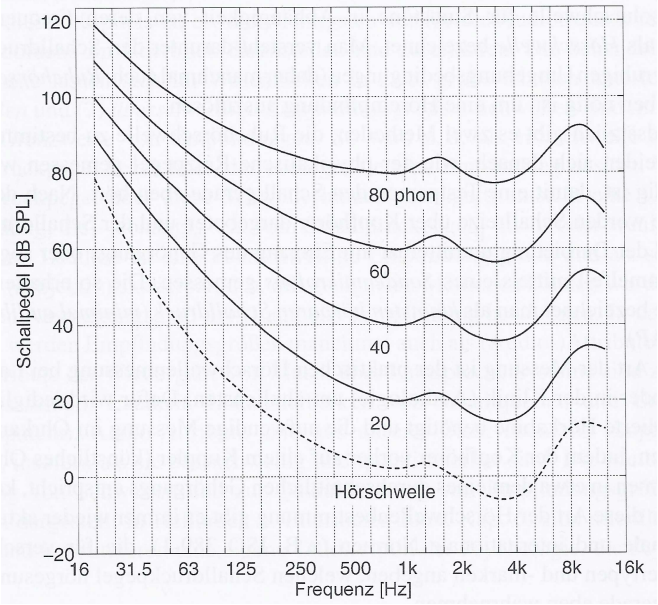


Bild 1: Hörschwelle und Kurven gleicher Lautstärke-Isophone[1]

Aus Bild 1 geht hervor, dass das menschliche Ohr für Frequenzen zwischen 2 kHz und 7 kHz die größte Empfindlichkeit hat. Für alle anderen Frequenzen muss der Schalldruckpegel höher sein, damit diese in gleicher Weise wahrgenommen werden. Um z. B. eine Tuba mit den ty-

[1] [1], S. 54, Abb. 2.5

pischen tiefen Frequenzen genauso stark wahrzunehmen wie eine Blockflöte mit den höheren Frequenzen, muss entweder die Tuba lauter spielen als die Blockflöte oder die tiefen Frequenzen werden durch einen Klangregler, z. B. in der Nachbearbeitung einer entsprechenden Aufnahme, in der Amplitude angehoben.

Ein weiterer Aspekt für die Notwendigkeit einer Klangregelung können entsprechende Unzulänglichkeiten in der Übertragungskennlinie von Aufnahme- und Wiedergabegeräten (z. B. Mikrofon, Verstärker, Lautsprecher usw.) sein. Hier kann eine Klangregelung dazu verwendet werden, diese Unzulänglichkeiten auszugleichen.

2.1. Die Klangregelung

Der Klangregler soll die Amplituden der tiefen und hohen Frequenzen anheben oder absenken. Dazu wird ein Tief- und ein Hochpass benötigt. Der Tiefpass besteht aus einem Tiefen-Shelvin-Filter und der Hochpass aus einem Höhen-Shelvin-Filter. Im Abschnitt 2.1.2 werden diese näher beschrieben. Das in Bild 2 gezeigte Blockschaltbild zeigt den grundsätzlichen Aufbau des Klangreglers.

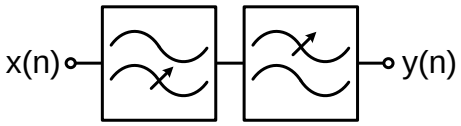


Bild 2: grundlegendes Blockschaltbild des Klangreglers

Die Verarbeitung soll digital erfolgen. Dazu ist das analoge Eingangssignal zu digitalisieren. Dies erfolgt über einen Analog-Digital-Wandler (*ADC*). Genauso muss das digitale Ausgangssignal in ein analoges Audiosignal gewandelt werden. Dies erfolgt durch einen Digital-Analog-Wandler (*DAC*). Somit erweitert sich das im Bild 2 gezeigte Blockschaltbild zu dem im Bild 3.

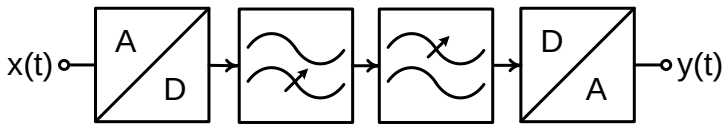


Bild 3: Blockschaltbild des Klangreglers erweitert um *ADC* und *DAC*

Der *ADC* und der *DAC* sind in einem Audio-Codec integriert. Dieser wandelt die Spannungswerte in einen digitalen Wert bzw. wandelt den digitalen Wert in eine Spannung um. Diese Werte sind durch die Wortbreite auf einen Wertebereich beschränkt. Bei der Verarbeitung kann es dazu kommen, dass dieser Wertebereich unter- oder überschritten wird. Dies führt im Audio-Codec zu einem Überlauf, dessen Auswirkung auf das Audiosignal im Abschnitt 2.1.4 näher beschrieben wird. Um das Unter- oder Überschreiten des Wertebereiches zu verhindern, wird ein Limiter vor die Übergabe des digitalen Wertes an den *DAC* eingefügt. Womit sich das Blockschaltbild um den Limiter zu dem im Bild 4 gezeigten erweitert.

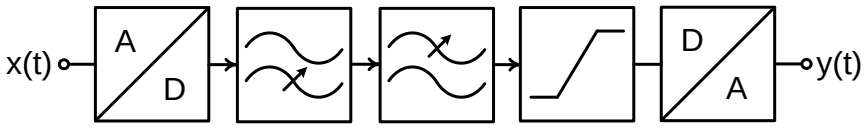


Bild 4: Blockschaltbild des Klangreglers erweitert um den Limiter

Für eine gewisse Flexibilität wird eine Lautstärkeeinstellung hinzugefügt. Mit Flexibilität ist gemeint, dass nicht erst vom Klangregler zum Wiedergabegerät gewechselt werden muss, um die Lautstärke zu verändern. Das um die Lautstärkeeinstellung erweiterte Blockschaltbild zeigt Bild 5. Die nähere Beschreibung der Lautstärkeeinstellung erfolgt im Abschnitt 2.1.3. Da die Lautstärkeeinstellung ebenfalls zu einem Unter- oder Überschreiten des Wertebereiches führen kann, ist diese noch vor dem Limiter einzufügen.

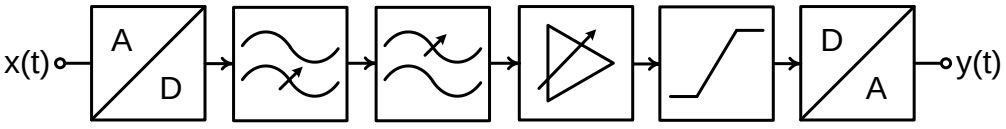


Bild 5: Blockschaltbild des Klangreglers erweitert um die Lautstärkeeinstellung

Der in Bild 5 dargestellte Klangregler wird auf einem Signalprozessor implementiert. Die Parametereinstellungen der Komponenten des Klangreglers erfolgen über einen Mikrocontroller. Die Parameterwerte sind über Steuersignale an den Signalprozessor zu übermitteln. Die Eingabe der Parameter kann über eine Benutzerschnittstelle direkt am Gerät, einem PC-Programm oder einem Webinterface vorgenommen werden. Die Verbindung zwischen PC-Programm/Webinterface zum Mikrocontroller erfolgt über eine Ethernet-Verbindung. Das Bild 6 zeigt das vollständige Blockschaltbild der Klangregelung.

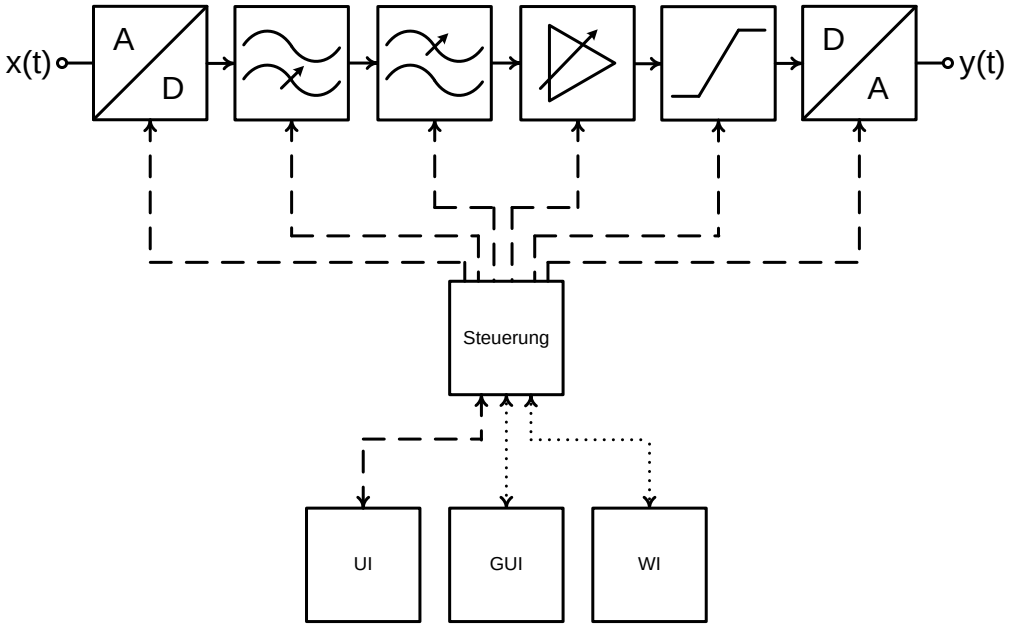


Bild 6: Blockschaltbild der Klangregelung

2.1.1. Die Analog-Digital-Wandlung

Um eine gewisse Kompatibilität zu anderen Systemen für den Klangregler zu gewährleisten, wird das Audiosignal in analoger Form zugeführt und abgegeben. Da die Signalverarbeitung in digitaler Form erfolgt, ist eine Wandlung des zeitkontinuierlichen Signals in ein zeitdiskretes Signal notwendig. Bei der Wandlung wird das zeitkontinuierliche Signal mit der Abtastfrequenz f_s abgetastet und in eine Folge von Abtastwerten (samples) gewandelt.

Bei der Abtastung ist das Nyquist-Shannon-Abtasttheorem zu berücksichtigen. Hiernach gilt:

$$f_s > 2f_{max} \tag{1}$$

Je höher die Abtastfrequenz gewählt wird, desto geringer wird sich die Abtastfolge von der zeitkontinuierlichen Signalform unterscheiden. Mit Nyquist-Frequenz f_N wird die Frequenz bezeichnet, welche sich als Grenze der rekonstruierbaren Frequenzen darstellt und mit

$$f_N = \frac{f_s}{2} \tag{2}$$

definiert ist. Alle Frequenzen unterhalb f_N lassen sich dann ohne Fehler wieder rekonstruieren. Daraus ergibt sich die Bedingung:

$$f_{max} < f_N \tag{3}$$

Für die Wahl der Abtastfrequenz f_s wurde die maximal abzutastende Frequenz mit $f_{max} = 20$ kHz festgelegt. Dies entspricht der maximal vom menschlichen Ohr hörbaren Frequenz. Für den Klangregler wurde somit die Abtastfrequenz $f_s = 48$ kHz gewählt. Womit die Nyquist-Frequenz f_N nach Gleichung (2) $f_N = 24$ kHz ist. Somit wird Gleichung (3) mit $f_{max} = 20$ kHz $< f_N = 24$ kHz eingehalten.

Da bei einem Audiosignal auch Frequenzen enthalten sein können, welche größer oder gleich der Nyquist-Frequenz f_N sind, kann es zum sogenannten Aliasing-Effekt kommen. Dabei überlappen sich die periodisch fortgesetzten Spektren, wodurch innerhalb der Bandbreite des Originalsignals Spiegelfrequenzen entstehen. Diese hinzugekommenen Frequenzen lassen sich nach der Wandlung nicht mehr entfernen, da nicht erkannt werden kann, dass es sich hier um einen Fehler in der Abtastung handelt bzw. das Signal einer Unterabtastung unterlag. Dem Aliasing-Effekt kann mittels Tiefpass entgegengewirkt werden. Der Tiefpass ist so zu dimensionieren, dass dieser Frequenzen die größer oder gleich der Nyquist-Frequenz f_N sind, herausfiltert. Solch ein Tiefpass muss dem Klangregler nicht hinzugefügt werden, da dieser im verwendeten Audio-Codec dem *ADC* bereits vorgeschaltet ist. Daher sollen die Probleme, welche bei der Realisierung des Tiefpasses zu berücksichtigen sind, hier nicht weiter betrachtet werden.

Nach der Abtastung der zeitlichen Abfolge des Signals müssen diese einem Quantisierer zugeführt werden. Dieser diskretisiert den Amplitudenverlauf des abgetasteten Signals. Dabei werden die Spannungswerte der Abtastfolge in Binärzahlen $x(n)$ der Wortbreite N_W gewandelt. Die Wortbreite N_W bestimmt die Auflösung des Quantisierers und damit auch die Quantisie-

rungskennlinie. Die Wortbreite N_W gibt die Anzahl Bits pro Zahlenwert an und somit auch die Anzahl der Quantisierungsstufen N_Q entsprechend Gleichung (4).

$$N_Q = 2^{N_W} \tag{4}$$

Mit der Vorgabe des Aussteuerungsbereiches ΔU_{ADC} kann das Quantisierungsintervall ΔU_Q nach Gleichung (5) berechnet werden.

$$\Delta U_Q = \frac{\Delta U_{ADC}}{N_Q} \tag{5}$$

Der Aussteuerungsbereich ΔU_{ADC} ergibt sich aus der Differenz der maximal ($U_{ADC_{max}}$) und minimal ($U_{ADC_{min}}$) zulässigen Spannung am Eingang des *ADC*. Damit wird Gleichung (5) zu Gleichung (6) bzw. (7).

$$\Delta U_Q = \frac{U_{ADC_{max}} - U_{ADC_{min}}}{N_Q} \tag{6}$$

$$\Delta U_Q = \frac{U_{ADC_{max}} - U_{ADC_{min}}}{2^{N_W}} \tag{7}$$

Nach [1] S. 790 ergibt sich der Wertebereich des Quantisierungfehlers zu

$$\Delta Q_f = \pm\frac{\Delta U_Q}{2}. \tag{8}$$

Weiter wird, nach [1] S.793 Gleichung (14.2), die Fehlerleistung wie folgt berechnet:

$$W_Q = \int_{-\infty}^{+\infty} q^2 p_Q(q)\,dq \tag{9}$$

$$W_Q = \frac{1}{\Delta U_Q} \int_{-\Delta U_Q/2}^{\Delta U_Q/2} q^2 dq$$

$$W_Q = \frac{1}{\Delta U_Q} \left[\frac{q^3}{3}\right]_{-\frac{\Delta U_Q}{2}}^{\frac{\Delta U_Q}{2}}$$

$$W_Q = \frac{1}{\Delta U_Q} \left[\left(\frac{\Delta U_Q^3}{24}\right) - \left(-\frac{\Delta U_Q^3}{24}\right)\right]$$

$$W_Q = \frac{\Delta U_Q^2}{12} \tag{10}$$

Nach [1] S. 793 Gleichung (14.3) wird die Leistung eines sinusförmigen Signals nach Gleichung (11) berechnet. Die Gleichung ist nur bei Vollausschlag (maximale Amplitude) gültig.

$$W_S = \frac{(\Delta U_Q \cdot 2^{N_W-1})^2}{2} \tag{11}$$

Der Signal-Rausch-Abstand *SNR* lässt sich mit

$$SNR = 10\log_{10}\left(\frac{W_S}{W_Q}\right) \tag{12}$$

berechnen.

Durch das Einsetzen der Gleichungen (10) und (11) in Gleichung (12) ergibt sich der *SNR* zu

$$SNR = 10\log_{10}\left(\frac{\Delta U_Q^2 \cdot 2^{2N_W-2}}{2} \cdot \frac{12}{\Delta U_Q^2}\right) \tag{13}$$

$$SNR = 10\log_{10}\left(2^{2N_W} \cdot \frac{12}{2 \cdot 2^2}\right)$$

$$SNR = 20\log_{10}(2)\,N_W + 10\log_{10}\left(\frac{3}{2}\right)$$

$$SNR = 6{,}02 \cdot N_W + 1{,}76 \quad \textit{in dB}. \tag{14}$$

Mit der Wortbreite N_W = 16 bit ergibt sich ein theoretischer Signal-Rausch-Abstand $SNR \approx 98$ dB. Gleichung (14) ist nur auf sinusförmige Signale anwendbar. Da Audiosignale in der Regel nicht sinusförmig sind, erklärt sich auch, dass der Hersteller des Audio-Codecs für eine Abtastfrequenz f_s = 48 kHz einen typischen *SNR* von 90 dB angibt (vergl. [2], S. 2-2). Weiterführende Grundlagen zur Analog-Digital-Wandlung finden sich in [1] ab Seite 785 und [3] ab S. 67.

2.1.2. Die Filter

Für die Klangregelung sind Filter zu finden, welche die Amplituden der Frequenzen von 0 Hz bis zur Grenzfrequenz f_g der tiefen Frequenzen und von der Grenzfrequenz f_g der hohen Frequenzen bis $f = +\infty$, mit dem jeweils eingestellten Verstärkungsfaktor g konstant anhebt oder absenkt. Zur Anhebung oder Absenkung der Amplituden stehen die vier grundsätzlichen Filterarten Tiefpass, Hochpass, Bandpass und Bandsperre (oder auch Bandstop) zur Verfügung. Bild 7 zeigt den prinzipiellen Verlauf des Amplitudengangs der vier Filterarten.

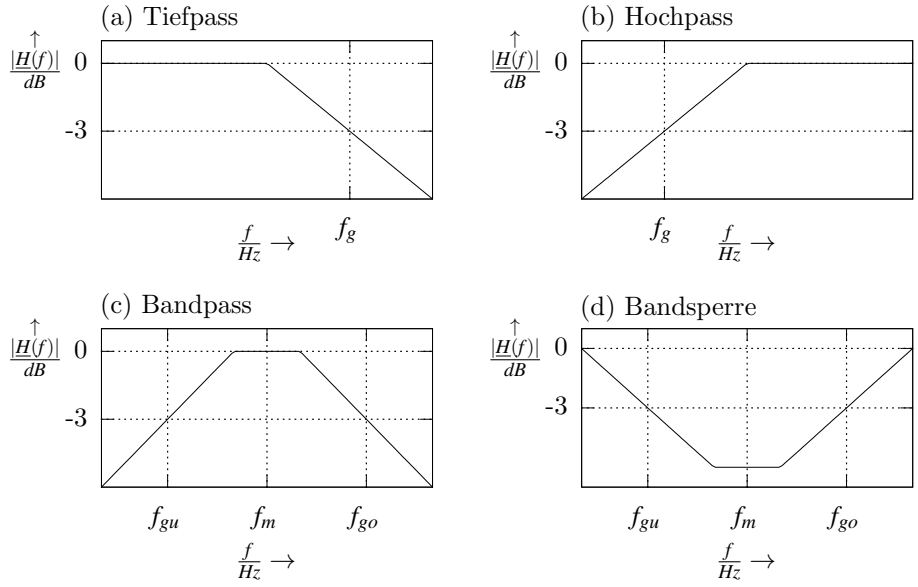
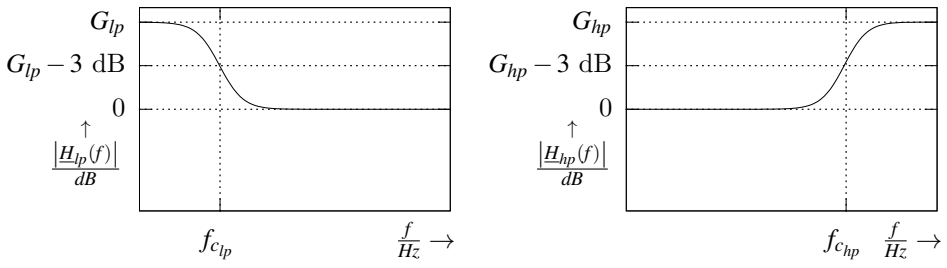


Bild 7: Filterarten im Vergleich

In Bild 7c und 7d ist zu sehen, dass die Frequenzen unterhalb und oberhalb der Grenzfrequenzen f_{gu} und f_{go} ebenfalls beeinflusst werden. Allerdings wird für die tiefen Frequenzen eine konstante Anhebung oder Absenkung der Amplituden von f = 0 Hz bis $f = f_g$ und für die hohen Frequenzen von $f = f_g$ bis $f = +\infty$ gefordert. Somit sind die Filterarten Bandpass und Bandsperre für die Klangregelung nicht geeignet. Bei den Filterarten Tiefpass und Hochpass sind die zuvor beschriebenen Bedingungen nahezu erfüllt. Lediglich in einem Übergangsbereich von der Grenzfrequenz f_g bis zur Frequenz f, bei der der gewünschte Verstärkungsfaktor G sich eingestellt hat, werden die Amplituden nicht konstant abgesenkt oder angehoben. Dabei liegt der maximale Unterschied zwischen den Verstärkungsfaktoren bei 3 dB, wie es für die Definition der Grenzfrequenz f_g festgelegt ist. Ein Nachteil bei dem klassischen Tief- und Hochpass ist, dass beim Tiefpass oberhalb und beim Hochpass unterhalb der Grenzfrequenz die Amplituden ebenfalls einer Verstärkung bzw. Dämpfung unterliegen. Dies lässt sich durch Verwendung der sogenannten Shelving-Filter vermeiden.

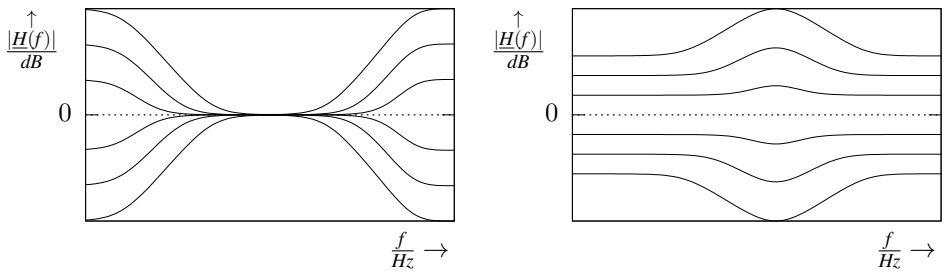
Das Shelving-Filter ist ein spezieller Fall des Tief- und Hochpasses. Das Shelving-Filter teilt sich in Tiefen-Shelving- und Höhen-Shelving-Filter. Bild 8 zeigt den typischen Verlauf des Amplitudengangs eines Tiefen- und Höhen-Shelving-Filters 2. Ordnung. Es ist zu sehen, dass beim Tiefen-Shelving-Filter beginnend bei 0 Hz bis zur Grenzfrequenz $f_{c_{lp}}$ der Verstärkungsfaktor $G_{lp}(f)$ konstant bleibt und bis $f_{c_{lp}}$ auf $G_{lp}(f = 0) - 3$ dB abfällt. So ist auch die Grenzfrequenz $f_{c_{lp}}$ und $f_{c_{hp}}$ definiert. Dies ist die Frequenz bei der der logarithmische Verstärkungsfaktor $G(f)$ um 3 dB kleiner ist gegenüber dem Verstärkungsfaktor $G(f = 0)$ für das Tiefen-Shelving-Filter und $G(f = +\infty)$ beim Höhen-Shelving-Filter. Beim Höhen-Shelving-Filter ist zu sehen, dass der Verstärkungsfaktor $G_{hp}(f)$ zunächst ansteigt, und ab der Grenzfrequenz $f_{c_{hp}}$ auf einen Verstärkungsfaktor $G_{hp}(f_{c_{hp}}) + 3$ dB weiter ansteigt und auf diesem bis $f = +\infty$ verbleibt. Damit ist gewährleistet, dass die zu beeinflussenden tiefen und hohen Frequenzen einer nahezu konstanten Verstärkung unterzogen werden, aber die anderen Frequenzen nicht bzw. nur minimal beeinflusst werden.



(a) Tiefen-Shelving-Filter (b) Höhen-Shelving-Filter

Bild 8: Amplitudengang von Höhen- und Tiefen-Shelving-Filter

Bild 9 zeigt die Kurvenschar aus der Kaskade von Tiefen- und Höhen-Shelving-Filter mit verschiedenen Verstärkungsfaktoren g_{lp} und g_{hp}, bei gleichen Grenzfrequenzen $f_{c_{lp}}$ und $f_{c_{hp}}$. Es ist zu sehen, dass mit zunehmendem bzw. abnehmendem Verstärkungsfaktor ausgehend von der 0-dB-Achse der Bereich der Frequenzen welche nicht verändert werden sollen, sich vergrößert. Dies kann dazu führen, dass im Verbund der beiden Filter auch der Bereich der mittleren Frequenzen verändert wird. Daher ist ein Anheben oder Absenken der Frequenzen nur bis zu einem bestimmten Verstärkungsfaktor g_{lp} und g_{hp} sinnvoll. Ab wann das weitere Anheben oder Absenken der Verstärkungsfaktoren g_{lp} und g_{hp} der einzelnen Filter nicht mehr sinnvoll ist, hängt davon ab, wie die Grenzfrequenzen $f_{c_{lp}}$ und $f_{c_{hp}}$ gewählt sind und welchen Effekt man mit dem Klangregler erzielen möchte. Bild 9b zeigt die gleiche Kurvenschar wie Bild 9a mit näher zueinander liegenden Grenzfrequenzen $f_{c_{lp}}$ und $f_{c_{hp}}$, wodurch es zu einer Beeinflussung in Form der Anhebung bzw. Absenkung der Amplituden der mittleren Frequenzen kommt.



(a) ohne Beeinflussung (b) mit Beeinflussung der mittleren Frequenzen

Bild 9: Kurvenschar der Kaskade aus Tiefen- und Höhen-Shelving-Filter

Die Klangregelung der Audiosignale erfolgt durch einen Signalprozessor. Daher sind die Tiefen- und Höhen-Shelving-Filter als digitale Filter auszulegen. Bei den digitalen Filterstrukturen wird zwischen rekursiven und nicht rekursiven Filtern unterschieden. Nicht rekursive Filter werden durch Filterstrukturen mit endlicher Impulsantwort (Finite Impulse Response - FIR) und rekursive Filter durch Filterstrukturen mit unendlicher Impulsantwort (Infinite Impulse Response - IIR) gebildet. Bild 10 und 11 zeigen die Signalflussgraphen der Filterstrukturen des FIR- und IIR-Filters.

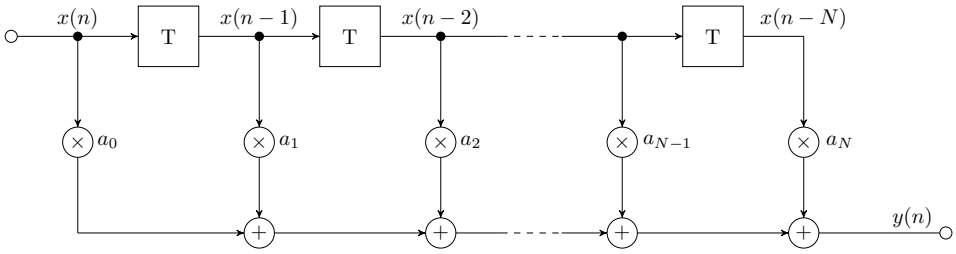


Bild 10: Signalflussgraph des FIR-Filters

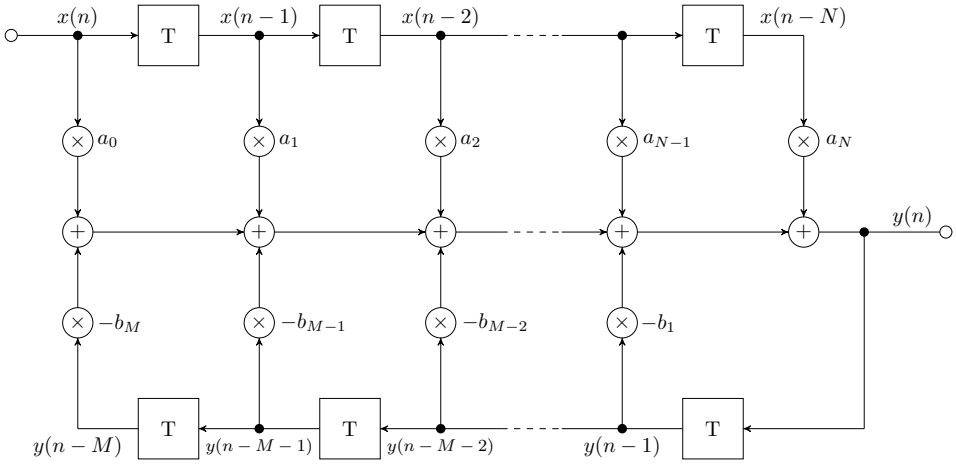


Bild 11: Signalflussgraph des IIR-Filters

Aus den Signalflussgraphen lassen sich die Differenzengleichungen für den FIR-Filter

$$y(n) = \sum_{i=0}^{N} a_i x(n-i) \tag{15}$$

und den IIR-Filter

$$y(n) = \sum_{i=0}^{N} a_i x(n-i) - \sum_{k=1}^{M} b_k y(n-k) \tag{16}$$

bilden.

Aus Gleichung (16) ist zu erkennen, dass die unendliche zeitdiskrete Faltungssumme durch die gewichtete Summe der Abtastwerte des Eingangs- und Ausgangssignals berechnet wird.[2] Das Eingangssignal $x(n)$ wird mit dem ersten Term aus (16) durch ein vorwärtsgerichtetes FIR-Filter und der zweite Term aus (16) koppelt das Ausgangssignal über ein FIR-Filter zurück. Die Kombination eines vorwärts- und rückwärtsgerichteten FIR-Filters bezeichnet man als rekursives Filter, welches auf Grund der Rückkopplung eine unendlich lange Impulsantwort besitzt.[3] Die Anzahl der Verzögerungselemente gibt die sogenannte Ordnung an.

Wie aus [1] (S. 829) entnommen werden kann, ist der Aufwand an Koeffizienten bei einem FIR-Filter wesentlich höher als bei einem IIR-Filter. Mit jedem weiteren Koeffizienten erhöht sich die Ordnung des Filters und damit auch die Verzögerung, welche das Audiosignal bei der Filterung erfährt. Dies kann zu Problemen führen, wenn eine zeitliche Abhängigkeit des Audiosignals mit einem anderen Signal besteht. Wendet man den Klangregler beispielsweise bei einem Konzert an, so könnte es hier zu einen Versatz zwischen der visuellen und auditiven Wahrnehmung kommen. Daher sind, um die Laufzeitverzögerungen so kurz wie möglich zu halten, für den Klangregler die IIR-Filter geeigneter als die FIR-Filter.

Allerdings hat der IIR-Filter gegenüber dem FIR-Filter den Nachteil eines nicht linearen Phasengangs und der daraus folgenden nicht konstanten Gruppenlaufzeit. Die Gruppenlaufzeit gibt an, wie der Laufzeitunterschied der Frequenzen zueinander ist und somit wann eine Frequenz gegenüber einer Anderen wahrgenommen wird. Da mit den Einstellmöglichkeiten der hier verwendeten Filter die Gruppenlaufzeiten in der Regel unterhalb von einer Millisekunde bleiben, kann davon ausgegangen werden, dass dieser Laufzeitunterschied nicht wahrgenommen wird und somit für den Klangregler nicht weiter relevant ist.

Ein weiterer Nachteil ist die Stabilität, da es durch die Rückkopplung beim IIR-Filter und ungünstiger Wahl der Koeffizienten zu Instabilitäten kommen kann. Es muss somit bei der Bildung der Koeffizientengleichungen auf die Stabilität geachtet werden. Da hier, wie im Folgenden noch gezeigt wird, auf bereits erarbeitete Koeffizientengleichungen aus [3] (S. 136, Tab. 5.3 und 5.4) zurückgegriffen wird und diese einen stabilen Zustand garantieren, soll die Problematik hier nur erwähnt sein und auf die entsprechende Literatur verwiesen werden.

[2] [1], S. 827
[3] [1], S. 827 f.

Die im Klangregler zur Anwendung kommenden Shelving-Filter sind Filter 2. Ordnung. Daraus ergibt sich aus Gleichung (16) mit $N = M = 2$ Gleichung (17).

$$y(n) = \sum_{i=0}^{2} a_i x(n-i) - \sum_{k=1}^{2} b_k y(n-k) \tag{17}$$

Aus Gleichung (17) lässt sich die im Bild 12 gezeigte Filterstruktur entwickeln.

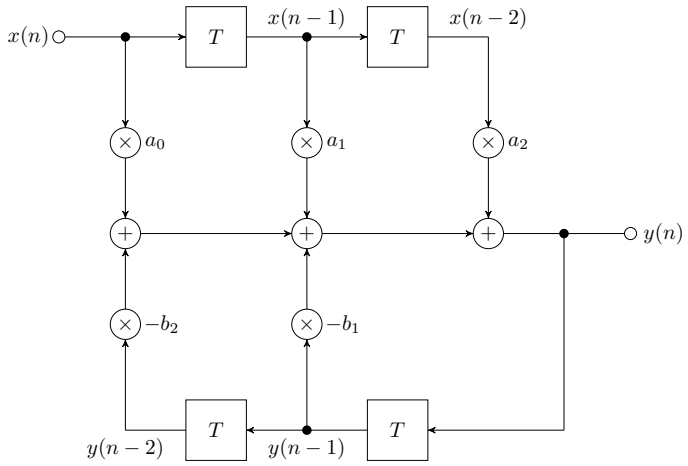


Bild 12: Signalflussgraph der Filterstruktur eines IIR-Filters 2.Ordnung

Zur Berechnung der Koeffizienten wird die Hilfsgröße K ([3], S. 135 und [4], S. 64 Tab. 2.3) eingeführt.

$$K = \tan\left(\frac{\omega_c T_s}{2}\right) \tag{18}$$

$$K = \tan\left(\frac{\pi f_c}{f_s}\right) \tag{19}$$

Mit der Hilfsgröße K und den Koeffizientengleichungen der Tabelle 1 lassen sich die Koeffizienten berechnen. Eine genaue Herleitung der Koeffizientengleichungen für die Shelving-Filter findet sich in [3] (S. 162 ff.).

Tiefen-Shelving $\left(\textit{Anhebung } g_{lp} = 10^{\frac{G_{lp}}{20}}\right)$				
a_0	a_1	a_2	b_1	b_2
$\frac{1+\sqrt{2g_{lp}}K+g_{lp}K^2}{1+\sqrt{2}K+K^2}$	$\frac{2\left(g_{lp}K^2-1\right)}{1+\sqrt{2}K+K^2}$	$\frac{1-\sqrt{2g_{lp}}K+g_{lp}K^2}{1+\sqrt{2}K+K^2}$	$\frac{2\left(K^2-1\right)}{1+\sqrt{2}K+K^2}$	$\frac{1-\sqrt{2}K+K^2}{1+\sqrt{2}K+K^2}$
Tiefen-Shelving $\left(\textit{Absenkung } g_{lp} = 10^{-\frac{G_{lp}}{20}}\right)$				
a_0	a_1	a_2	b_1	b_2
$\frac{1+\sqrt{2}K+K^2}{1+\sqrt{2g_{lp}}K+g_{lp}K^2}$	$\frac{2\left(K^2-1\right)}{1+\sqrt{2g_{lp}}K+g_{lp}K^2}$	$\frac{1-\sqrt{2}K+K^2}{1+\sqrt{2g_{lp}}K+g_{lp}K^2}$	$\frac{2\left(g_{lp}K^2-1\right)}{1+\sqrt{2g_{lp}}K+g_{lp}K^2}$	$\frac{1-\sqrt{2g_{lp}}K+g_{lp}K^2}{1+\sqrt{2g_{lp}}K+g_{lp}K^2}$
Höhen-Shelving $\left(\textit{Anhebung } g_{hp} = 10^{\frac{G_{hp}}{20}}\right)$				
a_0	a_1	a_2	b_1	b_2
$\frac{g_{hp}+\sqrt{2g_{hp}}K+K^2}{1+\sqrt{2}K+K^2}$	$\frac{2\left(K^2-g_{hp}\right)}{1+\sqrt{2}K+K^2}$	$\frac{g_{hp}-\sqrt{2g_{hp}}K+K^2}{1+\sqrt{2}K+K^2}$	$\frac{2\left(K^2-1\right)}{1+\sqrt{2}K+K^2}$	$\frac{1-\sqrt{2}K+K^2}{1+\sqrt{2}K+K^2}$
Höhen-Shelving $\left(\textit{Absenkung } g_{hp} = 10^{-\frac{G_{hp}}{20}}\right)$				
a_0	a_1	a_2	b_1	b_2
$\frac{1+\sqrt{2}K+K^2}{g_{hp}+\sqrt{2g_{hp}}K+K^2}$	$\frac{2\left(K^2-1\right)}{g_{hp}+\sqrt{2g_{hp}}K+K^2}$	$\frac{1-\sqrt{2}K+K^2}{g_{hp}+\sqrt{2g_{hp}}K+K^2}$	$\frac{2\left(K^2/g_{hp}-1\right)}{1+\sqrt{2/g_{hp}}K+K^2/g_{hp}}$	$\frac{1-\sqrt{2/g_{hp}}K+K^2/g_{hp}}{1+\sqrt{2/g_{hp}}K+K^2/g_{hp}}$

Tabelle 1: Koeffizientengleichungen der Shelving-Filter[4]

In der grafischen Benutzeroberfläche und dem Webinterface sollen die Diagramme für den Amplituden- und Phasengang sowie der Gruppenlaufzeit dargestellt werden. Dazu wird die komplexe Übertragungsfunktion $\underline{H}(f)$ benötigt. Diese ist mit

$$\underline{H}(f) = \frac{\sum_{i=0}^{N} a_i e^{-\frac{j2\pi fi}{f_s}}}{1+\sum_{k=1}^{M} b_k e^{-\frac{j2\pi fk}{f_s}}} \tag{20}$$

definiert und mit M=N=2 für die Ordnung ergibt sich Gleichung (20) zu Gleichung (21).

$$\underline{H}(f) = \frac{\sum_{i=0}^{2} a_i e^{-\frac{j2\pi fi}{f_s}}}{1+\sum_{k=1}^{2} b_k e^{-\frac{j2\pi fk}{f_s}}} \tag{21}$$

[4] [3], S. 136, Tab. 5.3 und 5.4

Mit der komplexen Übertragungsfunktion $\underline{H}(f)$ lässt sich der Amplitudengang aus der Betragsfunktion $|\underline{H}(f)|$ gemäß Gleichung (22) und der Phasengang $\varphi(f)$ aus dem Argument, wie es Gleichung (23) zeigt, berechnen.

$$|\underline{H}(f)| = \sqrt{Re\{\underline{H}(f)\}^2 + Im\{\underline{H}(f)\}^2} \tag{22}$$

$$\varphi(f) = \left(\frac{Im\{\underline{H}(f)\}}{Re\{\underline{H}(f)\}}\right) \tag{23}$$

Die Gruppenlaufzeit $\tau_G(\omega)$ ergibt sich aus der Steigungsfunktion

$$\tau_G(\omega) = -\frac{d\varphi(\omega)}{d\omega} \tag{24}$$

des Phasengangs $\varphi(\omega)$. Mit $\omega = 2\pi f$ lässt sich Gleichung (24) zu

$$\tau_G(f) = -\frac{1}{2\pi}\frac{d\varphi(f)}{df} \tag{25}$$

umschreiben.

Der Amplitudengang $|\underline{H}(f)|$ entspricht dem linearen Verstärkungsfaktor $g(f)$ für die jeweiligen Frequenzen, welcher auf das Eingangssignal $x(n)$ angewendet wird. Aus dieser Verarbeitung des Audiosignals ergibt sich die Phasenverschiebung $\varphi(f)$. Der Gruppenlaufzeit $\tau_G(f)$ kann der Laufzeitunterschied der Frequenzanteile des Audiosignals zueinander entnommen werden. Die Umrechnung des linearen Verstärkungsfaktors $g(f)$ erfolgt mit

$$G(f) = 20\log_{10}[g(f)] \tag{26}$$

zum logarithmischen Verstärkungsfaktor $G(f)$ in dB. Mit

$$g(f) = 10^{\frac{G(f)}{20}} \tag{27}$$

lässt sich der logarithmische Verstärkungsfaktor $G(f)$ in dB zum linearen Verstärkungsfaktor $g(f)$ umrechnen.

2.1.3. Die Lautstärkeeinstellung

Wie in Abschnitt 2.1 bereits erwähnt wurde, soll die Lautstärkeeinstellung dazu dienen die Bedienung etwas komfortabler zu gestalten. Die Lautstärkeeinstellung teilt sich in drei Blöcke. Der erste Block ist die vom Benutzer gewählte Lautstärke in Form eines einfachen Verstärkungsfaktors g_V und wird hier weiter als Volume bezeichnet. Der zweite Block ist eine Verhältniseinstellung zwischen dem rechten und linken Kanal des Stereosignals und wird im Folgenden mit Balance oder Balanceeinstellung bezeichnet. Die Balanceeinstellung wird durch den Balancewert B repräsentiert. Aus diesem werden die Verstärkungsfaktoren $g_{B_r}(B)$ für den rechten und $g_{B_l}(B)$ für den linken Kanal berechnet, welche mit den Signalwerten multipliziert werden.

Der dritte Block ist ein Ein- oder Ausblenden des Audiosignals bei entsprechender Einstellung für das Stummschalten (Mute) des Audioausgangs durch den Benutzer. Im Folgenden wird dieser Block mit *Fade* bezeichnet und durch den Verstärkungsfaktor $g_F(n)$ repräsentiert. So ergibt sich das Blockschaltbild der Lautstärkeeinstellung zu dem im Bild 13 gezeigten.

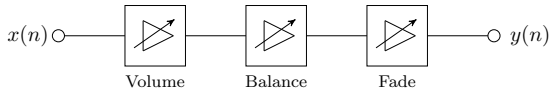


Bild 13: Blockschaltbild der Lautstärkeeinstellung

Der Block *Volume* dient der Lautstärkeeinstellung durch den Benutzer. Über die Benutzerschnittstellen gibt dieser einen Verstärkungsfaktor G_V in dB vor. Dieser liegt im Bereich von -30 dB bis 30 dB. Der lineare Verstärkungsfaktor g_V wird mit

$$g_V = 10^{\frac{G_V}{20}} \tag{28}$$

berechnet und nach

$$y(n) = x(n) \cdot g_V \tag{29}$$

auf das Eingangssignal multipliziert.

Der Signalflussgraph ist somit sehr einfach und im Bild 14 dargestellt.

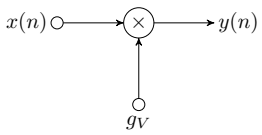


Bild 14: Signalflussgraph des Blocks *Volume* der Lautstärkeeinstellung

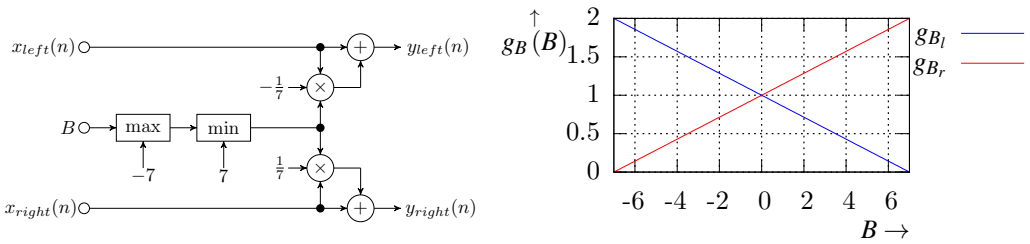
Der zweite Block *Balance* dient der Einstellung des Lautstärkeverhältnisses zwischen linken und rechtem Kanal des Stereosignals. Dazu verändert der Benutzer den Balancewert B, über welchem der Verstärkungsfaktor $g_{B_l}(B)$ für den linken Kanal mit

$$g_{B_l}(B) = \begin{cases} 2 & \text{für} \quad B < -7 \\ -\frac{1}{7}B + 1 & \text{für} \quad -7 \leq B \leq 7 \\ 0 & \text{für} \quad 7 < B \end{cases} \tag{30}$$

und der Verstärkungsfaktor $g_{B_r}(B)$ für den rechten Kanal mit

$$g_{B_r}(B) = \begin{cases} 0 & \text{für} \quad B < -7 \\ \frac{1}{7}B + 1 & \text{für} \quad -7 \leq B \leq 7 \\ 2 & \text{für} \quad 7 > B \end{cases} \tag{31}$$

berechnet wird. Die Verstärkungsfaktoren $g_{B_r}(B)$ und $g_{B_l}(B)$ werden entsprechend auf die Audiosignale des rechten und linken Kanals multipliziert. Der Signalflussgraph für die Balanceeinstellung und die Kurvenverläufe der Funktionen $g_{B_r}(B)$ und $g_{B_l}(B)$ zeigt Bild 15.



(a) Signalflussgraph (b) Kurvenverlauf $g_{B_r}(B)$ und $g_{B_l}(B)$

Bild 15: Block *Balance* der Lautstärkeeinstellung

Der dritte Block ist das Ein- oder Ausblenden entsprechend der Einstellung *Mute*, welche durch den Koeffizienten c_m repräsentiert wird. Diese Einstellung gibt der Benutzer vor, um die Stummschaltung des Audioausgangs ein- oder auszuschalten. Für den Koeffizienten c_m gilt:

$$c_m = \begin{cases} 0 & \text{für Stummschaltung (}\textit{Mute}\text{) aus} \\ 1 & \text{für Stummschaltung (}\textit{Mute}\text{) ein} \end{cases} \tag{32}$$

Der Benutzer kann über die Einstellung der Ein- bzw. Ausblendzeit t_F, im Intervall von 0 s bis 10 s vorgeben, wie lange das Ein- oder Ausblenden dauern soll. Durch Einbeziehen der Abtastfrequenz f_s ergibt sich der maximale Zählwert $n_{F_{max}}$ zu:

$$n_{F_{max}} = t_F \cdot f_s \tag{33}$$

Zur Bestimmung, ob der Benutzer eine Änderung in der Vorgabe der Stummschaltung vorgenommen hat, wird der Koeffizient c_{m2} mit

$$c_{m2} = [c_m(n) \Leftrightarrow c_m(n-1)] \tag{34}$$

bestimmt. Es wird die Äquivalenz der momentanen Einstellung *Mute* und der Vorgängereinstellung ermittelt. Tab. 2 zeigt die Wertetabelle zu Gleichung (34).

$c_m(n)$	$c_m(n-1)$	$c_{m2} = c_m(n) \Leftrightarrow c_m(n-1)$
0	0	1
0	1	0
1	0	0
1	1	1

Tabelle 2: Wertetabelle zur Berechnung des Koeffizienten c_{m2}

Somit kann der Zählwert durch eine einfache Multiplikation mit c_{m2} auf Null zurückgestellt werden, wenn der Benutzer die Vorgabe ändert. Dies ist aber nur für den Fall geeignet, wenn das Ein- oder Ausblenden bereits abgeschlossen ist, denn sonst würde es zu einem Sprung des Verstärkungsfaktors $g_F(n)$ kommen. Beim Einblenden würde dieser auf Null und beim Ausblenden auf Eins springen. Daher muss dafür gesorgt werden, dass der neue Zählwert, wenn das

Ein- oder Ausblenden noch nicht abgeschlossen ist, auf einen entsprechenden Wert eingestellt wird, der die neue Vorgabe bei dem Verstärkungsfaktor $g_F(n)$ auf $g_F(n-1)$ weiter arbeiten lässt, so dass es nicht zu einem Sprung kommt. Dies kann durch die Subtraktion $n_{F_{max}} - n_F(n-1)$ erfolgen. Somit ergibt sich der Zählwert $n_F(n)$ während einer Ein- oder Ausblendphase zu:

$$n_F(n) = [n_F(n-1)+1]\,c_{m2} + [n_{F_{max}} - n_F(n-1)]\,(\neg c_{m2}) \tag{35}$$

Der linke Term der Addition sorgt dafür, dass der Zählwert bei unveränderter Vorgabe für die Stummschaltung inkrementiert wird. Dabei hat c_{m2} durch die Multiplikation einer Null oder Eins mit den Zählerwerten die Funktion den Zähler zu aktivieren oder zu deaktivieren. Der rechte Term der Addition kommt zum Zug, wenn eine Änderung der Vorgabe der Stummschaltung erfolgt ist. Dann wird der Zählwert auf die weiter oben bereits besprochene Differenz eingestellt. Für den Fall, dass eine Vorgabeänderung erfolgt während das Ein- oder Ausblenden bereits abgeschlossen ist, wird der Zählwert $n_F(n)$ wie folgt berechnet:

$$n_F(n) = n_{F_{max}} \cdot c_{m2} \tag{36}$$

Dadurch wird der Zählwert $n_F(n)$ auf dem maximalen Zählwert gehalten, damit es nicht zu einem Überlauf kommt. Erfolgt eine Vorgabeänderung in der Stummschaltung, so wird der Zählwert $n_F(n)$ auf Null gesetzt.

Gleichungen (35) und (36) lassen sich zu

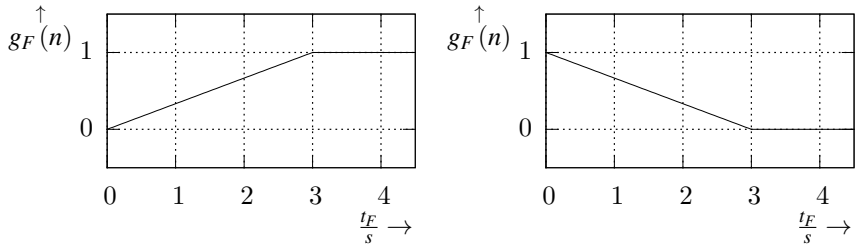
$$n_F(n) = \begin{cases} [n_F(n-1)+1]\,c_{m2} + [n_{F_{max}} - n_F(n-1)]\,(\neg c_{m2}) & \text{für } n_F(n-1) < n_{F_{max}} \\ n_{F_{max}} \cdot c_{m2} & \text{für } n_F(n-1) \geq n_{F_{max}} \end{cases} \tag{37}$$

zusammenfassen. Der Verstärkungsfaktor $g_F(n)$ lässt sich mit

$$g_F(n) = \begin{cases} \left| c_m(n) - \frac{n_F(n)}{n_{F_{max}}} \right| & \text{für } n_{F_{max}} \neq 0 \\ \neg c_m & \text{für } n_{F_{max}} = 0 \end{cases} \tag{38}$$

berechnen.

Bild 16 zeigt den Kurvenverlauf für das Ein- und Ausblenden mit einer Ein-bzw. Ausblendzeit t_F = 3 s. Zu sehen ist, wie nach dem Ein- bzw. Ausblenden der Verstärkungsfaktor $g_F(n)$ beim Einblenden konstant bei 1 und beim Ausblenden konstant bei 0 verbleibt. So wird der vom Benutzer gewählte Zustand für die Stummschaltung (*Mute*) sichergestellt.

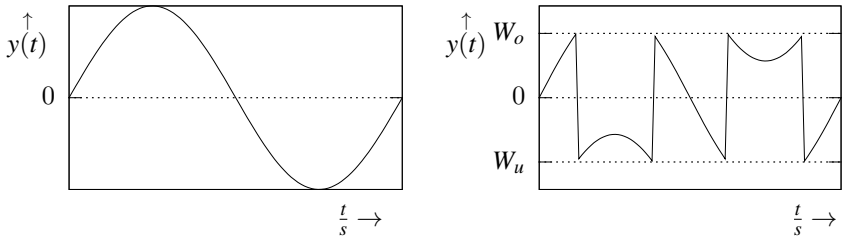


(a) Einblenden (b) Ausblenden

Bild 16: Kurvenverlauf für das Ein- und Ausblenden mit t_F = 3 s

2.1.4. Der Limiter

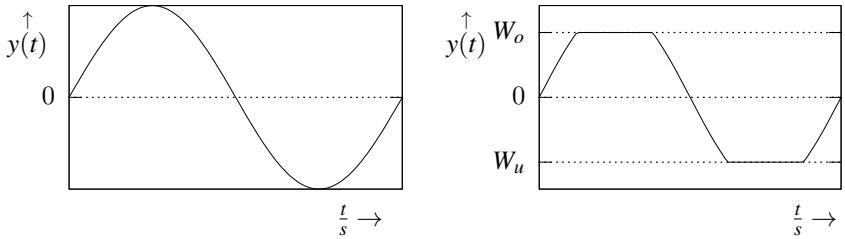
Bei der Verarbeitung der Audiosignale im Signalprozessor kann das Problem entstehen, dass das berechnete Ausgangssignal den durch den Audio-Codec vorgegebenen Wertebereich übersteigt bzw. unterschreitet. Dies führt zu einem Überlauf (Overflow), welcher sich durch einen Sprung in den negativen bzw. positiven Bereich bemerkbar macht. Bild 17 zeigt anhand eines Sinussignals, wie sich ein Überlauf auf dieses auswirkt. Dabei markieren W_o die obere und W_u die untere Grenze des anwendbaren Zahlenbereiches. Zu hören sind Verzerrungen, welche sich im einfachsten Fall als Knackgeräusche und im schlimmsten Fall als starkes Rauschen bemerkbar machen. Je stärker die Verzerrungen werden, desto weniger ist vom Original zu hören, bis es zu einem einzigen Rauschen kommt.



(a) Original (b) Verzerrungen nach Überlauf

Bild 17: Verzerrung im Zeitbereich durch Überlauf

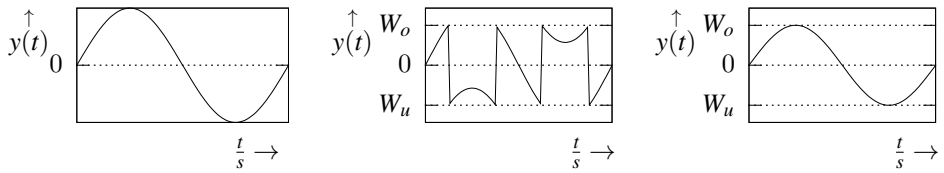
Um das Überschreiten des Wertebereiches zu verhindern, sind nach der Bearbeitung des Audiosignals entsprechende Maßnahmen zu treffen. Dies könnte das Begrenzen auf den maximalen bzw. minimalen Wert des Wertebereiches sein. Allerdings führt dies, wie es Bild 18 zeigt, ebenfalls zu Verzerrungen (Clipping).



(a) Original (b) Begrenzung der Amplituden

Bild 18: Verzerrungen durch Abschneiden der Werte ober- und unterhalb des Wertebereiches

Eine andere Möglichkeit ist die Verwendung eines Begrenzers (Limiter), wie er hier zur Anwendung kommt. Dieser berechnet im Voraus durch Verzögerung des Audiosignals einen Verstärkungsfaktor, der auf das Audiosignal multipliziert wird und so einen Überlauf verhindert. Bild 19 zeigt, wie sich der Limiter auf einen Überlauf auswirkt. Aus Bild 19c geht hervor, dass es nicht mehr zu Verzerrungen kommt. Dafür wird aber die Dynamik beeinflusst.



(a) Original (b) ohne Limiter (c) mit Limiter

Bild 19: Wirkungsweise des Limiters

Zunächst wird ermittelt, welcher Kanal des Stereosignals den größeren Wert des Betrages der Amplitude aufweist.

$$c_a = max\left(\left|x_{left}(n)\right|, \left|x_{right}(n)\right|\right) \tag{39}$$

Der so ermittelte größere Betragswert c_a wird nun mit dem vorherigen Spitzenwert $x_p(n-1)$ verglichen. Ist der Betragswert größer als der vorherige Spitzenwert, so wird mit der Ansprechzeit

$$AT = 1 - e^{\frac{-2.2 \cdot 10^6}{t_a \cdot f_s}} \tag{40}$$

reagiert. Ist dies aber nicht der Fall so wird mit der Rücklaufzeit

$$RT = 1 - e^{\frac{-2.2 \cdot 10^3}{t_r \cdot f_s}} \tag{41}$$

reagiert.

Wodurch sich der Spitzenwert zu

$$x_p(n) = \begin{cases} (1-AT)\,x_p(n-1) + AT \cdot c_a & \text{für } c_a > x_p(n-1) \\ (1-RT)\,x_p(n-1) + RT \cdot c_a & \text{für } c_a \leq x_p(n-1) \end{cases} \tag{42}$$

ergibt.

Die Ansprechzeit t_a ist in Mikrosekunden und die Rücklaufzeit t_r in Millisekunden anzugeben. Für den Limiter wurde für die Ansprechzeit t_a ein Wertebereich von 0,02 ms bis 10 ms und für die Rücklaufzeit t_r ein Wertebereich von 1 ms bis 5000 ms festgelegt. Aus dem Quotienten der Limiterschwelle

$$LT = lt\frac{2^{N_W-1}-1}{100} \tag{43}$$

und dem Spitzenwert $x_p(n)$, kann der Steuerwert

$$f_{lt} = min\left(1, \frac{LT}{x_p(n)}\right) \tag{44}$$

bestimmt werden.

Der Benutzer gibt die Limiterschwelle *lt* in Prozent von an. Eine Limiterschwelle *lt* = 100 % entspricht der Vollaussteuerung (0 dB). Der Quotient aus Limiterschwelle *LT* und Spitzenwert $x_p(n)$ wird mit dem Wert 1 verglichen und der kleinere Wert wird für den Steuerwert f_{lt} verwendet. Der Steuerwert f_{lt} wird nun mit dem vorherigen Verstärkungsfaktor $g_{lt}(n-1)$ verglichen. Ist der Steuerwert f_{lt} kleiner als der Verstärkungsfaktor $g_{lt}(n-1)$, wird mit der Ansprechzeit *AT* reagiert.

Ist dies aber nicht der Fall, so wird mit der Rücklaufzeit *RT* reagiert und der Verstärkungsfaktor $g_{lt}(n)$ mit

$$g_{lt}(n) = \begin{cases} (1-AT)\, g_{lt}(n-1) + AT \cdot f_{lt} & \text{für } f_{lt} < g_{lt}(n-1) \\ (1-RT)\, g_{lt}(n-1) + RT \cdot f_{lt} & \text{für } f_{lt} \geq g_{lt}(n-1) \end{cases} \tag{45}$$

berechnet. Der Verstärkungsfaktor $g_{lt}(n)$ wird nun mit dem verzögerten Audiosignal $x(n-D_{lt})$ multipliziert.

$$y(n) = g_{lt}(n) \cdot x(n-D_{lt}) \tag{46}$$

Beim hier verwendeten Limiter wurde sich am Beispiel *M-file 4.1 (limiter.m)* aus [4] S. 109 f. orientiert und diese auf die Gegebenheiten der Klangregelung angepasst.

2.1.5. Die Digital-Analog-Wandlung

Um das digital verarbeitete Signal wieder hörbar zu machen, muss es einem Digital-Analog-Wandler (DAC) zugeführt werden. Dieser setzt das zeitdiskrete Signal in ein zeitkontinuierliches Signal um. Dazu wird entsprechend der Abtastfrequenz f_s der zum Amplitudenwert des zeitdiskreten Signals gehörende Spannungswert für die Zeit zwischen den Abtastungen gehalten. Während der Einstellzeit t_{se} (Settling time) wird das nächste umzusetzende binäre Wort an den DAC übergeben und die entsprechende Ausgangsspannung U_{DAC} aufgebaut.

Da die Auflösung durch die verwendete Wortbreite N_w begrenzt ist, entstehen zwischen den einzelnen Werten Stufen. Die Anzahl der Stufen ergeben sich aus der Wortbreite N_w zu 2^{N_w}. Da die Wortbreite mit N_w = 16 bit festgelegt ist, ergeben sich somit 65536 Stufen bzw. Spannungswerte.

Diese Anzahl muss durch zwei geteilt werden, da sowohl positive als auch negative Spannungen dargestellt werden müssen. Somit ergibt sich für den positiven und negative Bereich jeweils 2^{15} = 32768 Stufen. Die Null wurde dem positiven Bereich zugeordnet. Das Ausgangssignal $U_{DAC}(t)$ wird mit

$$U_{DAC}(t) = y(n) \cdot \frac{U_{DAC_{max}}}{2^{N_w-1}-1} \tag{47}$$

$$U_{DAC}(t) = y(n) \cdot \frac{U_{DAC_{max}}}{2^{16-1}-1} \tag{48}$$

$$U_{DAC}(t) = y(n) \cdot \frac{U_{DAC_{max}}}{32767} \tag{49}$$

berechnet. Weiterführendes zum Thema der Digital-Analog-Wandlung findet sich in [3] auf Seite 92 und den folgenden Seiten.

2.2. Die Steuerung

Die Steuerung ist von der Audiosignalverarbeitung entkoppelt und erfolgt über einen Mikrocontroller, um den Signalprozessor zu entlasten und die Audioverarbeitung im vorgegebenen zeitlichen Rahmen der Abtastfrequenz f_s sicherzustellen. Es werden lediglich die Parameter der Audiosignalverarbeitung von der Steuerung an diese, über Steuersignale, weitergeleitet. Die Übertragung der Steuerparameter erfolgt über einen Inter-Integrated-Circuit-Bus (I^2C-Bus). Die Vorgaben des Benutzers an die Steuerung kann über drei Benutzerschnittstellen erfolgen.

Die erste Benutzerschnittstelle ist die am Gerät selber und wird im weiteren Verlauf als User Interface (UI) bezeichnet. Dazu zählen die Inkrementalgeber, die Taster und das Liquid Crystal Display (LCD). Diese werden über die GPIOs (General Purpose Input/Output) des Mikrocontrollers an diesen angebunden. Die zweite Benutzschnittstelle ist das PC-Programm, welches im

folgenden als Graphical User Interface (GUI) bezeichnet wird. Dieses übermittelt die Vorgaben des Benutzers über eine Ethernet-Verbindung. Auch die Vorgaben des Benutzers über die dritte Benutzerschnittstelle werden über eine Ethernet-Verbindung übermittelt. Dabei handelt es sich um ein Webinterface (WI), also eine Anwendung, welche im Webbrowser ausgeführt wird.

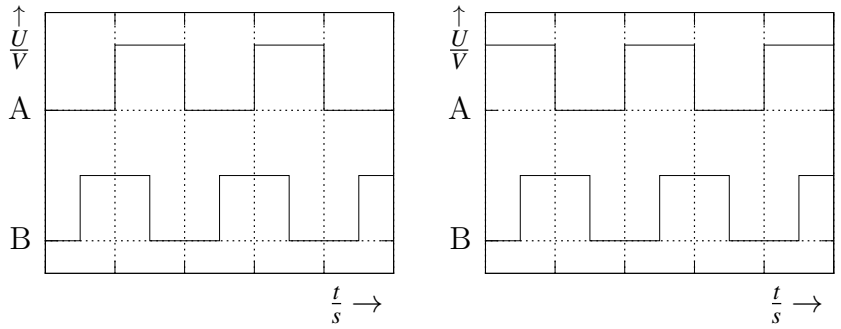
Die Übermittlung der Steuerparameter über den I^2C-Bus erfolgt in zwei Phasen. Zunächst wird ein 8-bit-Datenwort gesendet, welches die Adresse des Registers für den jeweiligen Parameter beinhaltet. In der zweiten Phase wird der Steuerwert in Form eines 8-bit-Datenwortes übertragen. Bei Steuerparametern, welche aus einem 16-bit-Wort bestehen, wird zunächst das untere Byte und anschließend das obere Byte gesendet. Dies macht es notwendig, dass 16-bit-Worte zwei Registeradressen erhalten. Die Steuersignale und deren Eigenschaften kann dem Anhang C entnommen werden.

2.2.1. Das User Interface (UI)

Mit der UI sind die Ein- und Ausgabemöglichkeiten am Gerät selber gemeint. Für die Eingabe werden Inkrementalgeber und Taster und für die Ausgabe ein LCD verwendet. Die Inkrementalgeber und Taster werden direkt an die GPIOs des Mikrocontrollers angeschlossen. Die Steuersignale an das LCD werden zunächst an das *I^2C-Bus LC-Display Module* übertragen. Dieses enthält den *Remote 8-bit I/O expander for I^2C-bus* PCF8574 der die seriellen Signale in das parallele Datenformat des LCD wandelt. Das Modul wird im folgenden mit Expander-Modul bezeichnet.

Inkrementalgeber sind Eingabeelemente bei denen der Benutzer durch Drehung des Gebers eine Richtungs- und Lageänderung vorgibt. Die Lageänderung ergibt sich aus der Anzahl der Impulse welche von den beiden Ausgängen A und B gesendet werden. Die Lage der Impulse zueinander (Phasenverschiebung), gibt die Drehrichtung an. Dabei kann festgehalten werden, dass bei den im Klangregler verwendeten Inkrementalgebern eine Phasenverschiebung von 90° einer Drehung entgegen dem Uhrzeigersinn und von -90° einer Drehung im Uhrzeigersinn entspricht. Es kann aber auch einfacher betrachtet werden, so wie es im Algorithmus des CUIP umgesetzt wurde. Mit jedem Flankenwechsel am Kontakt A wird eine Lageänderung um eine Längeneinheit registriert. Gleichzeitig wird geprüft, ob der logische Pegel am Kontakt B sich von dem des Kontakts A unterscheidet. Ist dies der Fall, wurde im Uhrzeigersinn gedreht. Sind

die logischen Pegel gleich, so wurde entgegen des Uhrzeigersinns gedreht. Die Inkrementalgeber werden für diverse Einstellungen innerhalb des Menüs, aber auch des Hauptbildschirmes verwendet. Bild 20 zeigt die Folge der Impulse an den Kontakten A und B entsprechend der Drehrichtung.



(a) Drehrichtung links (b) Drehrichtung rechts

Bild 20: Impulsdiagramme der Inkrementalgeber entsprechend der Drehrichtung

Die Kontakte A und B werden über jeweils einen Pull-Up-Widerstand an die GPIOs angeschlossen. Der Kontakt C wird mit Masse verbunden. Bild 21 zeigt das Prinzip des Anschlusses eines Inkrementalgebers an die GPIOs des Mikrocontrollers. Die Komponenten innerhalb des gestrichelten Rechtecks sind im Inkrementalgeber integriert.

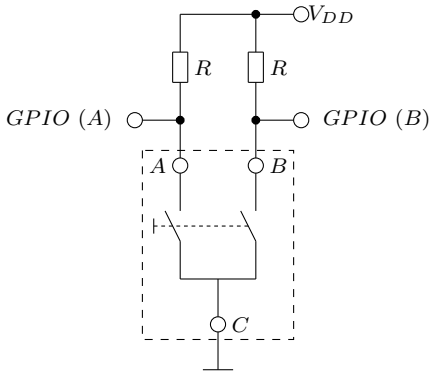
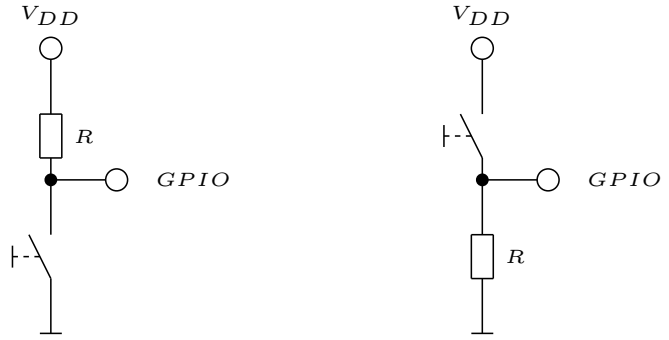


Bild 21: Prinzipschaltbild zum Anschluss der Inkrementalgeber an den Mikrocontroller

Die verwendeten Inkrementalgeber besitzen zusätzlich einen Taster mit den Kontakten D und E. Diese werden, wie die anderen Taster der Benutzerschnittstelle, ebenfalls über Pull-Up-Widerstände an die GPIOs des Mikrocontrollers angeschlossen. Dazu wird der Kontakt D mit

dem Pull-Up-Widerstand und dem GPIO und der Kontakt E mit Masse verbunden. Die Taster dienen der Auswahl des Menüs, des Untermenüs, aber auch dem Ein- oder Ausschalten der Filter, des Limiters sowie der Stummschaltung (*Mute*).

Prinzipiell können Taster über zwei Methoden an die GPIOs angeschlossen werden. Bild 22 zeigt die beiden Schaltungsarten. Zur Anwendung kommt der Anschluss über den Pull-Up-Widerstand, da diese im Mikrocontroller bereits vorhanden sind. Es ist darauf zu achten, dass die Pull-Up-Widerstände über die Programmierung des Mikrocontrollers zugeschaltet werden, da sonst die Gefahr besteht, dass die Taster ohne Pull-Up-Widerstände betrieben werden, was zu einer Beschädigung des Mikrocontrollers führen kann.



(a) über Pull-Up-Widerstand (b) über Pull-Down-Widerstand

Bild 22: Prinzipschaltbild zum Anschluss der Taster an einen Mikrocontroller

Die drei zusätzlichen Taster des UI sind den Funktionen *Bypass*, *Backlight* und *Reset* zugeordnet. Bei der Funktion *Bypass* kann die Umgehung des Klangreglers ein- oder ausgeschaltet werden. Mit der Funktion *Backlight* lässt sich die Hintergrundbeleuchtung des LCDs zuschalten, wenn diese auf Grund der Zeitschaltung abgeschaltet wurde. Über die Funktion *Reset* lässt sich der Mikrocontroller und der Signalprozessor zurücksetzen.

Im Klangregler wird ein LCD mit 2 Zeilen und 16 Zeichen pro Zeile verwendet. Die einzelnen Zeichen werden über eine 5x7 Punktmatrix dargestellt. Standardmäßig ist der ASCII-Zeichensatz vorinstalliert. Es ist möglich diesen Zeichensatz zu verändern. Dies kommt beim Klangregler aber nicht zur Anwendung, da hier der ASCII-Zeichensatz völlig ausreichend ist. Angesteuert wird das LCD über das Expander-Modul , welches die Steuersignale über einen I^2C-Bus vom Mikrocontroller erhält und diese in das parallele Datenformat des LCD umwandelt. Das LCD ist mit seinen Datenleitungen direkt an das Modul angeschlossen.

2.2.2. Das Graphical User Interface (GUI)

Das GUI ist eine Software für die Betriebssysteme Windows und Linux. Das Programm wurde in der Programmmiersprache C++ mit dem Framework Qt geschrieben und mit *Tone Control Graphical User Interface Program* (TCGUIP) bezeichnet. Dazu wurde die integrierte Entwicklungsumgebung (IDE) *Qt Creator* verwendet. Die GUI wird über eine Ethernet-Verbindung an die Klangregelung angebunden. Dazu ist zum Mikrocontroller eine entsprechende Schnittstelle zu integrieren und das User Datagram Protocol (UDP) zu implementieren.

Der Benutzer soll sich nicht mit den Einstellungen auseinandersetzen müssen. Dazu soll die GUI einen Broadcast starten und durch die Antwort der Klangregelung automatisch die IP-Adresse einstellen. Der Broadcast-Request ist in bestimmten Abständen zu wiederholen, um Änderungen (z. B. Wechsel der IP-Adresse) erfassen zu können. Auf Anwendungsebene wird kein bestimmtes Protokoll verwendet. Die Befehle und Werte zu den Parametern werden als Zeichenkette gesendet. Das Verfahren auf Anwendungsebene wird im Abschnitt 5 näher beschrieben. Es sind alle Parameter steuerbar, welche auch über die UI einstellbar sind. Einzige Ausnahme bilden die Netzwerkeinstellungen. Diese können nur über die UI vorgenommen werden.

2.2.3. Das Webinterface (WI)

Das WI bietet alle Möglichkeiten der Steuerung des Klangreglers, welche die GUI auch hat. Es wurde für die Browser *Chrome*, *Firefox* und *Opera* erstellt.

Das WI benutzt das Transmission Control Protocol (TCP) auf Port 80. Auf Anwendungsebene wird das Hypertext Transfer Protocol (HTTP) verwendet. Dieses stellt verschiedene Request-Methoden zur Verfügung. Von diesen wird die GET-Methode verwendet. Die GET-Methode verlangt, dass übertragene Daten in den Uniform Resource Identifier (URI) geschrieben werden. Der URI enthält die Adresse zum Server und wie schon erwähnt, die Daten welche an diesen übermittelt werden sollen. Im Falle des Klangreglers sind die Daten die Parameter, welche geändert werden sollen oder eine Abfrage, welche Werte die Parameter im Moment der Abfrage haben.

Die Adresse zum Klangregler besteht in der Regel aus der IP-Adresse, welche durch den DHCP-Server (Dynamic Host Control Protocol) zugewiesen wurde oder aus einer statischen IP-Adresse, wenn DHCP nicht verwendet wird. Wird ein DNS-Server (Domain Name System) verwendet mit dem eine Namensauflösung möglich ist, so kann die Adresse auch aus einem Namen bestehen. Dazu ist beim DNS-Server ein entsprechender Eintrag vorzunehmen.

Soll das WI abgerufen werden, so ist der Adresse kein weiterer Zusatz hinzuzufügen. Mit dem Zusatz *get_config.html* kann die gesamte Konfiguration des Klangreglers abgefragt werden. Dies geschieht im WI über einen HTTPRequest in Form des Konzeptes der asynchronen Datenübertragung per *Asynchronous JavaScript and XML* (AJAX). Mit dem Zusatz *set.html* und dem Anhängen der Kombination aus Parameterbezeichner und Parameterwert, wird dem Klangregler mitgeteilt, welcher Parameter auf welchen Wert einzustellen ist. Wird der Adresse ein anderer beliebiger Zusatz hinzugefügt, erfolgt immer als Rückgabe das WI selber. Die Beschreibung der Datenübertragung erfolgt in Abschnitt 5. Die Software zum WI wurde mit *Tone Control Webinterface Program* (TCWIP) bezeichnet.

3. Die Simulation

Vor der praktischen Umsetzung des Klangreglers ist durch eine Simulation die Funktion der Audiosignalverarbeitung nachzuweisen. Für die Simulation wird die Software MATLAB/Simulink verwendet. MATLAB (MATrix LABoratory) ist ein Softwarepaket für nummerische Mathematik und Simulink ist eine Erweiterung zur grafischen Modellierung und Simulation dynamischer Systeme. Über Toolboxen können weitere Softwareelemente hinzugefügt werden. So kommt hier die *DSP System Toolbox* zur Anwendung.

3.1. Die Konstruktion des Modells

Zur Erstellung der Simulation wurde nach dem Top-Down-Verfahren vorgegangen. Zunächst wurde das Modell zur Audiosignalverarbeitung erstellt, wie es Bild 23 zeigt.

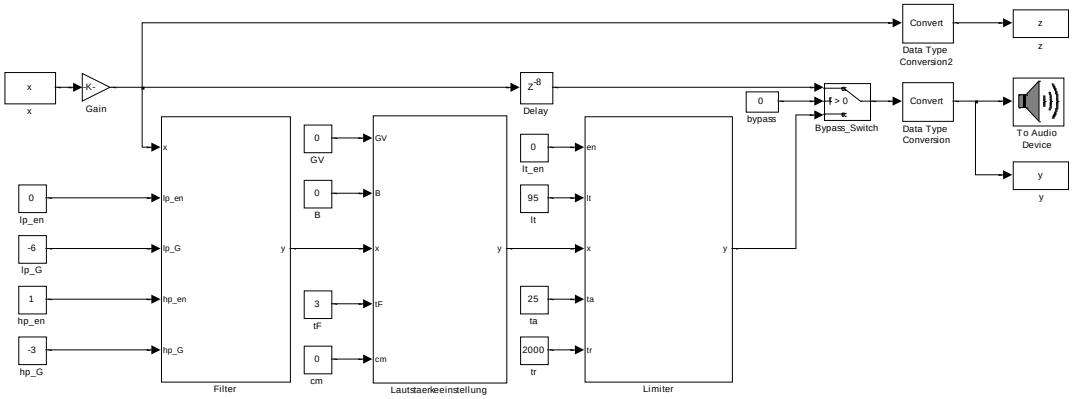


Bild 23: Simulink-Modell der Audiosignalverarbeitung

Für die Modelle Filter, Lautstärkeeinstellung und Limiter wurden Subsystem-Blöcke angelegt. Zur Übergabe der Parameter werden Constant-Blöcke verwendet. Die Realisierung des Bypass erfolgt über einen Switch-Block, bei dem das unbearbeitete oder das bearbeitete Signal auf den Ausgang geschaltet werden kann. Um die Übergabe der Daten in Form von vorzeichenbehafteten 16-bit-Festkommawerten zu simulieren, wird das Ausgangssignal entsprechend konvertiert. Das Ausgangssignal wird auf die Soundkarte des PCs gegeben und über die Variable y auf

dem Workspace von MATLAB abgelegt. Über die Variable z wird das Eingangssignal auf dem Workspace abgelegt. Dies ist notwendig, damit die Zeitstempel und die Größe der Variablen z und y übereinstimmen, da diese für die Darstellung des Zeit- bzw. Frequenzbereiches im Hauptbildschirm der Benutzerschnittstelle der Simulation verwendet werden.

3.1.1. Das Modell des Filters

Das Modell des Filters wurde als Subsystem-Block angelegt und beinhaltet die Nachschlagetabellen (Lookup Table - LUT) für die Koeffizienten des Tiefen- und Höhen-Shelving-Filters, sowie die ebenfalls als Subsystem-Blöcke angelegten IIR-Filter-Modelle. Die Übergabe der Parameter, des Ein- und Ausgangssignals erfolgt über In- und Outport-Blöcke. Bild 24 zeigt den Aufbau des Filter-Modells.

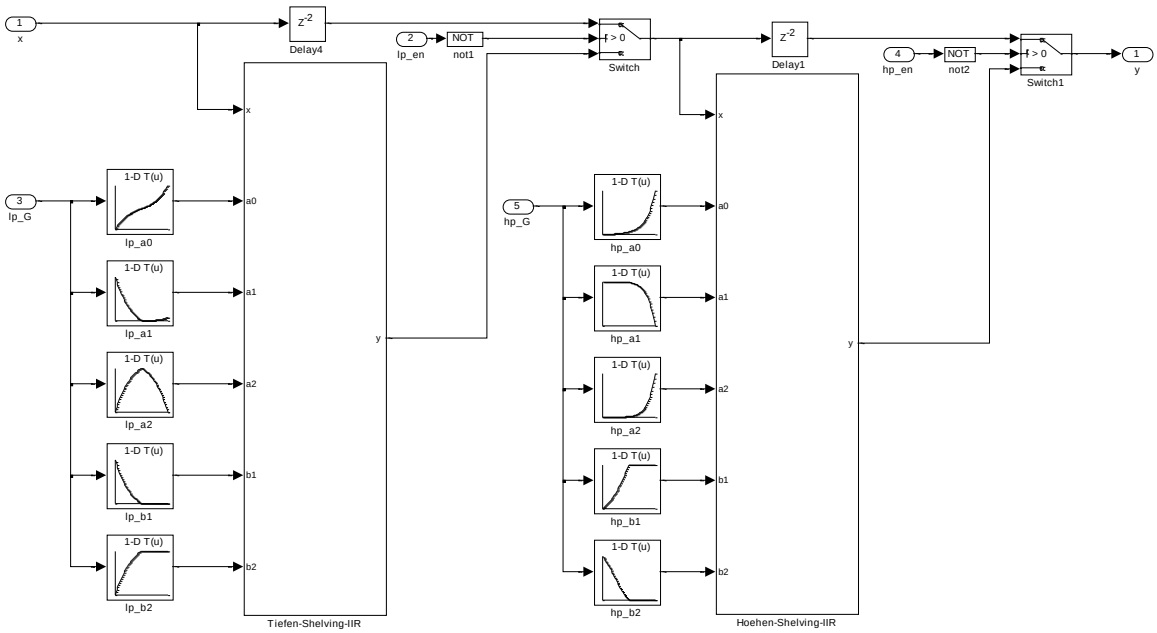


Bild 24: Simulink-Modell des Filters

Die LUTs bestehen aus *1-D Lookup Table* Blöcken. Die Änderung des Inhalts der Koeffizienten-LUTs erfolgt über die Benutzerschnittstelle der Simulation. Bei Änderung der Grenzfrequenz f_c wird eine Funktion aufgerufen, welche die jeweils 61 Koeffizientenwerte pro LUT anhand der Abtastfrequenz f_s und der neu eingestellten Grenzfrequenz f_c berechnet und der jeweiligen LUT übergibt.

Die 61 Koeffizientenwerte ergeben sich aus der Möglichkeit die Verstärkungsfaktoren G_{lp} und G_{hp} im Bereich von -30 dB bis 30 dB in 1-dB-Schritten wählen zu können. Je nach eingestelltem Verstärkungsfaktor geben die LUTs die entsprechenden Koeffizientenwerte an die IIR-Modelle weiter.

Die IIR-Modelle für den Tiefen- und Höhen-Shelving-Filter sind identisch aufgebaut. Es wurde nicht die Struktur der Direktform I umgesetzt wie diese im Abschnitt 2.1.2 im Bild 12 vorgestellt wurde, sondern die transponierte Direktform II wie es Bild 25 zeigt. Der Vorteil liegt in der Verwendung von zwei statt der vier Verzögerungsglieder, womit Speicher eingespart wird. Nimmt man die Struktur und wandelt diese in eine Gleichung um, so erhält man die Gleichung eines IIR-Filters 2. Ordnung entsprechend Gleichung (17).

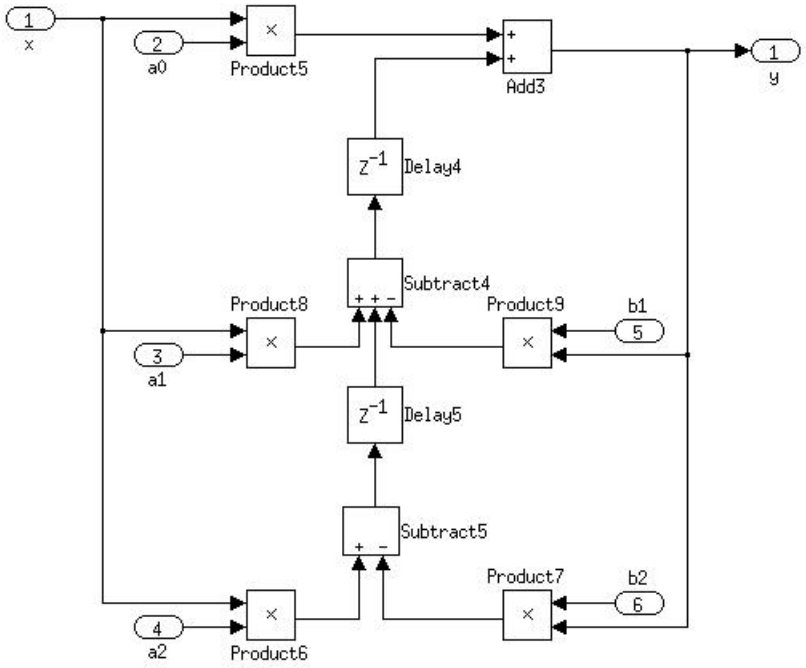


Bild 25: Simulink-Modell der IIR-Filterstruktur

Für das Zu- und Abschalten der einzelnen Filter werden Switch-Blöcke verwendet. Damit die Verzögerung des Signals auch bei Abschalten der Filter beibehalten wird, sind in den Signalpfad des unbearbeiteten Signals zwei Verzögerungsglieder eingearbeitet. Damit sollen eventuelle Knackgeräusche, beim Umschalten, unterdrückt werden. Für die Übergabe der Parameter und des Eingangssignals werden Inport-Blöcke und für das Ausgangssignal ein Outport-Block verwendet.

3.1.2. Das Modell der Lautstärkeeinstellung

Das Modell der Lautstärkeeinstellung enthält als Subsystem-Blöcke die Modelle *Volume*, *Balance* und *Fade*, wie dies aus Bild 26 hervorgeht. Als Übergabepunkte für Parameter, Ein- und Ausgangssignal dienen wieder die Inport- und Outport-Blöcke.

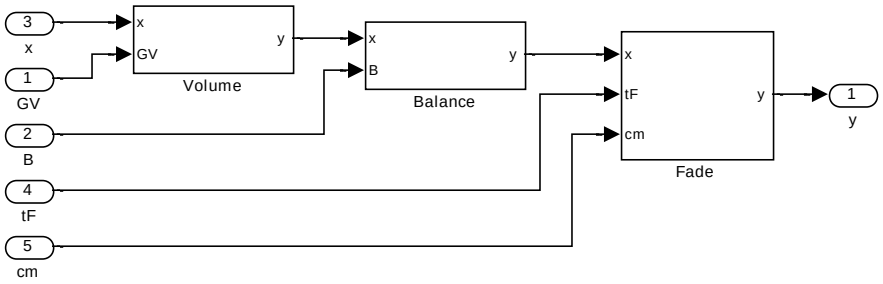


Bild 26: Simulink-Modell der Lautstärkeeinstellung

Das Modell des Blocks *Volume* besteht aus einer 1-D Lookup Table. Diese setzt den logarithmischen Verstärkungsfaktor G_V in den linearen Verstärkungsfaktor g_V um, welcher auf das Eingangssignal x multipliziert wird. Bild 27 zeigt das Modell des Blocks *Volume*.

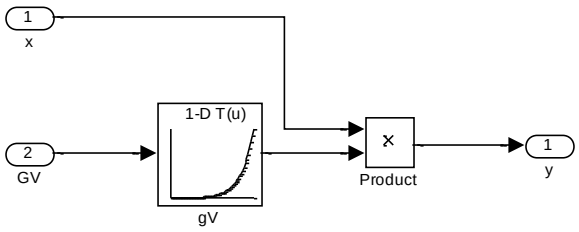


Bild 27: Simulink-Modell des Blocks *Volume*

Das Modell des Blocks *Balance* besteht hauptsächlich aus mathematischen Blöcken, welche so zusammengeschaltet sind, dass diese die Gleichungen (30) und (31) abbilden, wie es Bild 28 zu entnehmen ist. Um die berechneten Verstärkungsfaktoren für den linken und rechten Kanal auf die entsprechenden Signale multiplizieren zu können, wird das Stereosignal mithilfe eines Demux-Blocks in die einzelnen Signale des linken und rechten Kanals aufgesplittet und nach der Multiplikation mithilfe des Concatenate-Blocks wieder zu einem Stereosignal zusammengefasst.

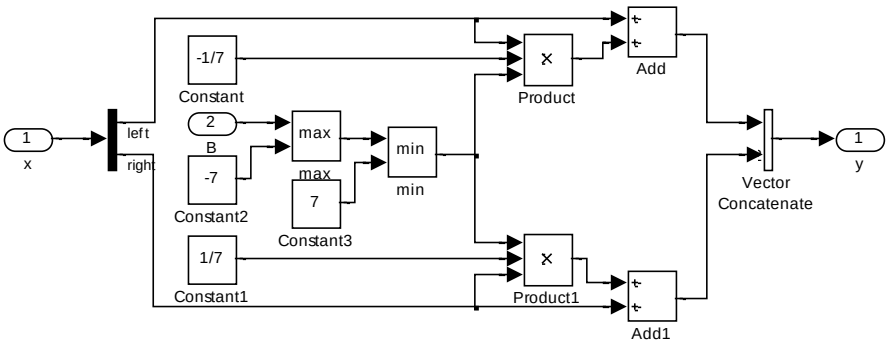


Bild 28: Simulink-Modell des Blocks *Balance*

Das Modell für den Block *Fade* enthält einen Subsystem-Block für den Zähler n_F aus dem der Verstärkungsfaktor g_F für das Ein- und Ausblenden berechnet wird. Der Verstärkungsfaktor g_F wird auf das Eingangssignal multipliziert, wodurch sich das Ein- bzw. Ausblenden des Audiosignals ergibt. Bild 29 zeigt das Modell des Blocks *Fade*.

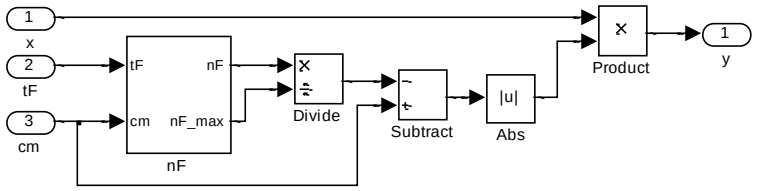


Bild 29: Simulink-Modell des Blocks *Fade*

Das Modell des Zählers n_F besteht aus mathematischen Blöcken, Verzögerungsgliedern und einem Switch-Block. Das Modell bildet damit die Gleichung (38) ab, wie dies dem Bild 30 entnommen werden kann.

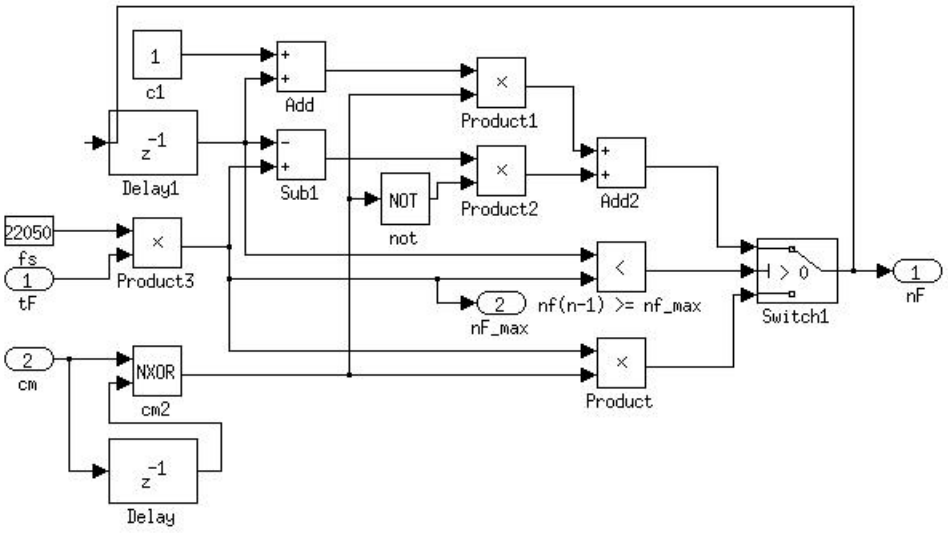


Bild 30: Simulink-Modell des Zählers n_F im Block *Fade*

3.1.3. Das Modell des Limiters

Das Modell des Limiters unterteilt sich in zwei Subsystem-Blöcke. Der Erste ermittelt den Spitzenwert x_p und mit dem Zweiten wird der Verstärkungsfaktor g_{lt} ermittelt, wie es im Bild 31 zu sehen ist.

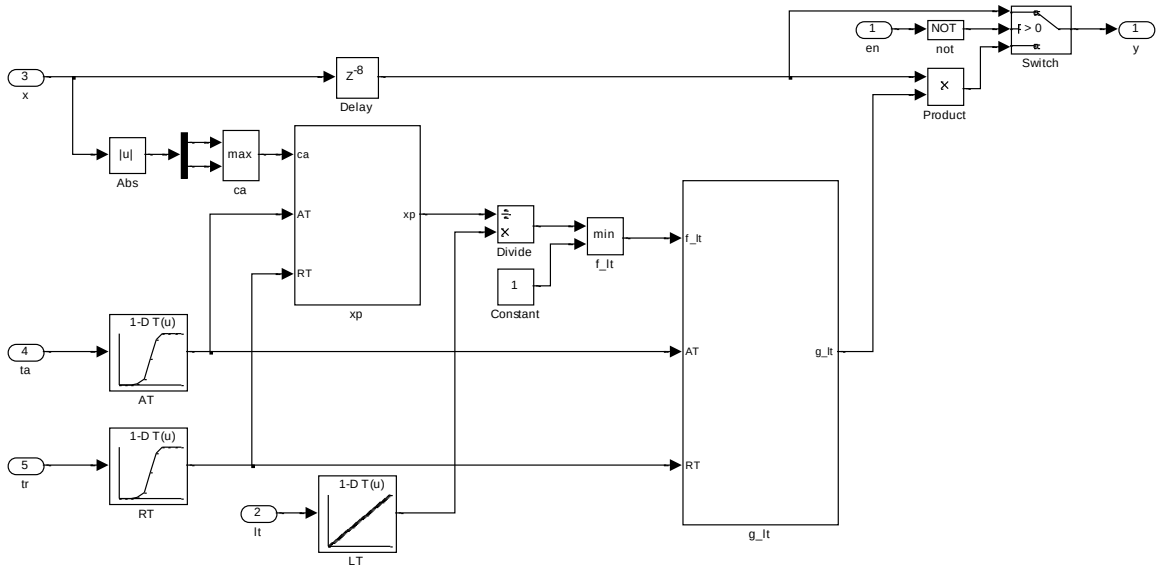


Bild 31: Simulink-Modell des Limiters

Die LUTs AT und RT setzen entsprechend der Abtastfrequenz f_s die Ansprechzeit t_a und Rücklaufzeit t_r in die entsprechende Werte AT und RT gemäß Gleichungen (40) und (41) um. Mit der LUT LT wird die Limiterschwelle *lt*, welche in Prozent angegeben wird, in den Wert LT umgesetzt, entsprechend Gleichung (43).

Das Verzögerungsglied verzögert das Signal um 8 Abtastungen, um der Ansprech- bzw. Rücklaufzeit auch Wirkung zu geben. Um auch hier eventuell auftretende Knackgeräusche beim Zu- oder Abschalten des Limiters zu verhindern, wird auch das unbearbeitete Signal der Verzögerung unterzogen. Das Zu- oder Abschalten des Limiters erfolgt über einen Switch-Block.

Über den MinMax-Block wird der maximale Wert der Beträge des linken und rechten Kanals ermittelt, welcher als Vergleichswert c_a für die Ermittlung des Spitzenwertes x_p dient. Das Modell zur Spitzenwertermittlung zeigt Bild 32, worin Gleichung (42) umgesetzt wurde.

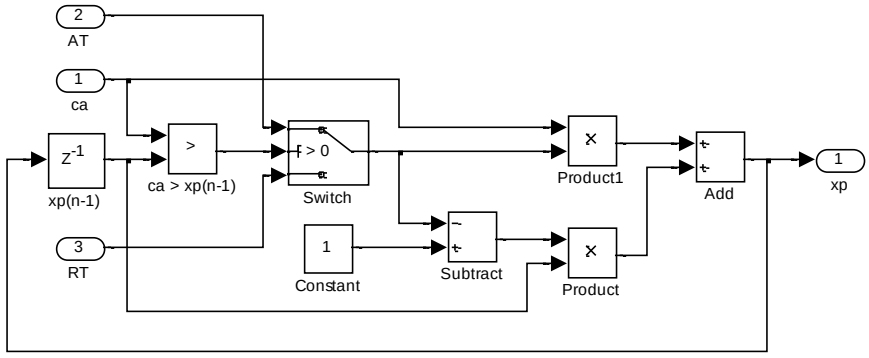


Bild 32: Simulink-Modell der Spitzenwertermittlung des Limiters

Nach dem Subsystem *xt* erfolgt die Ermittlung des Steuerwertes f_{lt} nach Gleichung (44). Daran schließt sich das Subsystem *g_lt* an, welches den Verstärkungsfaktor g_{lt} entsprechend Gleichung (45) ermittelt und in Bild 33 abgebildet ist.

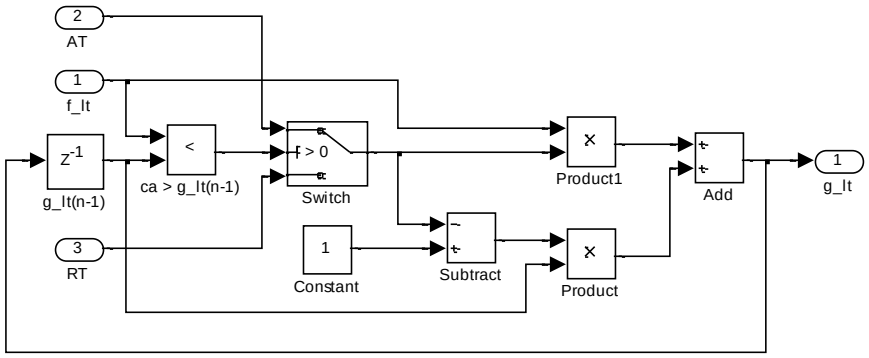


Bild 33: Simulink-Modell der Verstärkungsfaktorermittlung des Limiters

3.2. Die Benutzerschnittstelle zur Simulation

Die Benutzerschnittstelle der Simulation besteht aus einer grafische Benutzeroberfläche und MATLAB-Funktionen.

3.2.1. Die Benutzeroberfläche der Simulation

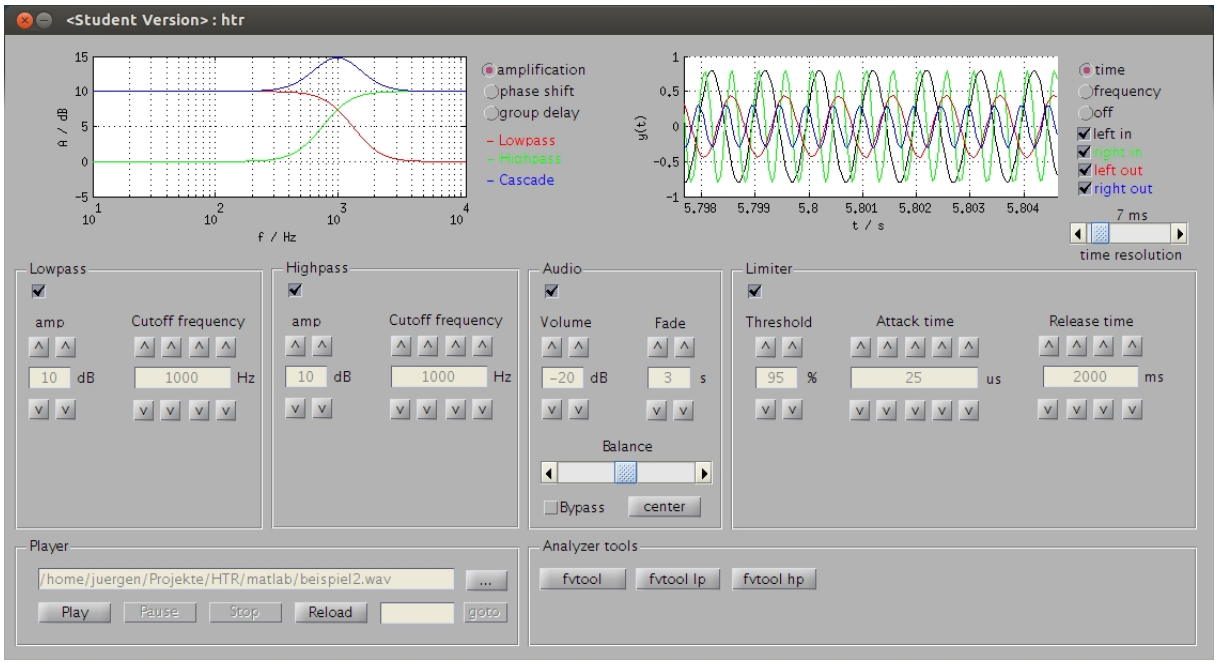


Bild 34: grafische Benutzeroberfläche der Simulation

Im Bild 34 wird die Benutzeroberfläche gezeigt, welche im oberen Bereich zwei Diagrammfelder besitzt. Beim linken Diagrammfeld kann zwischen Amplitudengang, Phasengang und Gruppenlaufzeit gewählt werden. Das rechte Diagrammfeld zeigt das aktuell abgespielte Signal entweder im Zeitbereich oder im Frequenzbereich. Im linken Diagrammfeld werden für den Amplituden- und Phasengang sowie der Gruppenlaufzeit zur Darstellung der Kurven des Tiefen-Shelving-Filters die Farbe Rot, für das Höhen-Shelving-Filter Grün und für die Kaskade aus beiden Filtern Blau verwendet.

Beim rechten Diagrammfeld werden für den Kurvenverlauf des linken Kanals des Eingangssignals die Farbe Schwarz, für den rechten Kanal des Eingangssignals die Farbe Grün, für den linken Kanal des Ausgangssignals die Farbe Rot und für den rechten Kanal des Ausgangssignals die Farbe Blau verwendet. Über Checkboxen rechts neben dem rechten Diagrammfeld lässt sich einstellen welche der einzelnen Kanäle angezeigt werden sollen. Bei der Anzeige im Zeitbereich lässt sich über ein Slider-Element die Auflösung von 0,1 ms bis 90 ms in 0,1-ms-Schritten einstellen. Die Auflösung der Anzeige des Frequenzbereiches richtet sich nach der Abtastfrequenz f_s. Dabei wird immer eine Auflösung im Bereich 10 Hz bis zur Nyquist-Frequenz F_N eingestellt. Mit der Einstellung *off* kann die Anzeige abgeschaltet bzw. eingefroren werden.

Die Einstellungen für Lowpass, Highpass, Limiter usw. sind in Gruppen eingeteilt. Die Gruppen *Lowpass* und *Highpass* enthalten die Einstellmöglichkeiten zu den Verstärkungsfaktoren G_{lp} und G_{hp} sowie zu den Grenzfrequenzen $f_{c_{lp}}$ und $f_{c_{hp}}$ des Tiefen- und Höhen-Shelving-Filters.

Für die Verstärkungsfaktoren sind Werte im Bereich von -30 dB bis 30 dB in 1-dB-Schritten wählbar. Für die Grenzfrequenz $f_{c_{lp}}$ ist ein Bereich von 20 Hz bis 1000 Hz in 1-Hz-Schritten wählbar. Der Bereich der Grenzfrequenz $f_{c_{hp}}$ kann im Bereich 1000 Hz bis $0,8 \cdot f_N$ in 1-Hz-Schritten gewählt werden.

Die Gruppe *Audio* beinhaltet die Einstellmöglichkeiten für das Modell der Lautstärkeeinstellung. Dazu gehört der Verstärkungsfaktor G_V für das Modell *Volume*, welcher im Bereich von -30 dB bis 30 dB in 1-dB-Schritten wählbar ist. Weiter gehört dazu die Ein- und Ausblendzeit t_F des Modells *Fade*. Diese kann im Bereich von 0 s bis 10 s in 1-Sekunden-Schritten gewählt werden. Eine weitere Einstellmöglichkeit der Gruppe *Audio* ist der Balancewert B für das Modell *Balance*. Dieser kann mit dem Slider in 1er-Schritten im Bereich -7 bis 7 gewählt werden. Links unterhalb des Sliders zur Balanceeinstellung befindet sich die Schaltfläche *center*. Klickt man diese an, so wird der Balancewert direkt auf 0 und somit auf die Mitte eingestellt. Neben der Schaltfläche *center* befindet sich die Checkbox *Bypass*. Mit dieser kann der Bypass ein- oder ausgeschaltet werden.

Die Gruppe *Limiter* beinhaltet die Einstellmöglichkeiten Threshold, Attack time und Release time. Über Threshold wird die Limiterschwelle *lt* im Bereich von 0 % bis 100 % in 1-%-Schritten gewählt. Die Attack time ist die Ansprechzeit t_a, welche im Bereich von 20 μs bis 10000 μs gewählt werden kann. Die Release time (Rückstellzeit) t_r kann im Bereich 2 ms bis 5000 ms gewählt werden.

Bei allen bisher vorgestellten Gruppen gibt es die Möglichkeit durch eine Checkbox im oberen linken Bereich der Gruppe, die jeweilige Gruppe aus dem Signalweg zu entfernen bzw. diese wieder in den Signalweg einzubinden. Bei der Gruppe *Audio* ist die Checkbox mit dem Signal *Mute* c_m verbunden. So dass beim Anhaken der Checkbox die Stummschaltung aktiviert und beim Entfernen des Hakens diese wieder deaktiviert wird.

Über die Gruppe *Player* kann eine WAVE-Datei geladen und abgespielt werden. Mit der Schaltfläche *Play* wird das Abspielen gestartet, mit *Pause* kann es pausiert werden und mit *Stop* beendet werden. Mit *Play* und *Stop* ist auch das Starten und Stoppen der Simulation verbunden. Über die Schaltfläche *Reload* lässt sich die WAVE-Datei erneut laden, falls es hier mal zum Verlust der Daten im MATLAB-Workspace kommen sollte. Beim Laden werden die Daten als

Variable *data* auf dem Workspace von MATLAB abgelegt. An das Modell der Simulation werden die Daten über die Variable x übergeben. Dazu werden entsprechend der Startzeit bis zur Ende der Abspielzeit die Daten aus der Variable *data* extrahiert und in der Variablen x abgelegt. So konnte eine Vor- und Rückspulfunktion realisiert werden. Dazu ist in das Eingabefeld links neben der Schaltfläche *goto* die Zeit in Sekunden anzugeben zu der gesprungen werden soll. Anschließend wird das Vor- bzw. Rückspulen zur angegebenen Abspielzeit durch Betätigen der Schaltfläche *goto* ausgeführt. Dazu wird die Simulation zunächst gestoppt, die Daten von der angegebenen Abspielzeit bis zum Ende der Abspielzeit aus der Variable *data* extrahiert und in der Variable *x* abgelegt. Anschließend wird die Startzeit der Simulation auf die neue Abspielzeit eingestellt und die Simulation gestartet. Nach Ablauf oder Stoppen der Simulation werden alle Daten aus der Variable *data* wieder in die Variable *x* geschrieben und die Simulationszeit auf die gesamte Abspielzeit eingestellt und der Startzeitpunkt auf 0 zurückgestellt.

Über die Gruppe *Analyzer Tools* kann das Tool *fvtool* zum Auswerten des eingestellten Filters aufgerufen werden. Dabei wird bei Aufruf über die Schaltfläche *fvtool lp* nur der Tiefen-Shelving-Filter und bei Aufruf über die Schaltfläche *fvtool hp* nur der Höhen-Shelving-Filter an das Tool übergeben. Bei Aufruf über die Schaltfläche *fvtool* wird die Kaskade aus Tiefen- und Höhen-Shelving-Filter an das Tool *fvtool* übergeben. Eine Anleitung zu *fvtool* kann in MATLAB über den Befehl *doc fvtool* abgerufen werden.

3.2.2. Die Funktionen der Benutzerschnittstelle

Beim Start über den Befehl *tc* werden zunächst, über den Befehl *bdclose('all')*, alle anderen System-Windows geschlossen. Anschließend wird geprüft, ob als Betriebssystem ein Unix-basiertes vorliegt. Ist dies der Fall so wird der Zeichensatz auf *ISO-8859-1* eingestellt. Danach wird das Simulationsmodell *tc_sim* geladen. Der nächste Schritt ist das Ausführen der Funktion *tc_init*, welche in der Datei *tc_init.m* abgelegt ist. Zum Schluss wird das *Reload* über die Funktion *tc_player* aufgerufen. Die Funktion *tc_player* ist in der Datei *tc_player.m* hinterlegt.

Die Funktion *tc_init* initialisiert die Anzeigen und Auswahlmöglichkeiten auf der Benutzeroberfläche und die Konstanten im Simulationsmodell mit den Standardwerten. Die Möglichkeit, wie beim späteren Gerät, zum Speichern und Laden bestimmter Zustände ist für die Simulation nicht vorgesehen. Weiterhin werden die nötigen Variablen auf dem MATLAB-Workspace initia-

lisiert. Zum Schluss werden zwei Timer gestartet. Der erste Timer (plottimer) ruft sekündlich die Funktion *tc_plot_signal* auf, welche den Kurvenverlauf für das aktuell abgespielte Signal im rechten Diagrammfeld aktualisiert. Der zweite Timer (enabletimer) ruft die Funktion *tc_set_enable* alle hundert Millisekunden auf. Die Funktion *tc_set_enable* aktiviert oder deaktiviert die Einstellmöglichkeiten auf der Benutzeroberfläche entsprechend des gewählten Zustandes.

Wird in der Benutzeroberfläche eine Einstellung verändert, so wird eine Funktion aufgerufen. Die Funktionen sind in ihrer Struktur gleich aufgebaut. In einer Kette aus if-ifelse-Anweisungen wird nach dem Algorithmus für das übergebene Kommando (cmd) gesucht. Wird eine Übereinstimmung gefunden, so wird entsprechend der neue Einstellwert ermittelt, dieser validiert und dem Simulationsmodell übergeben. Eine vollständige Übersicht über die Funktionen, Kommandos und deren Bedeutung findet sich im Anhang D.

4. Der praktische Aufbau

Im folgenden Kapitel soll der Aufbau des Gerätes erläutert werden. Das Prinzip des Aufbaus besteht aus der Teilung in Audiosignalverarbeitung, Steuerung und Benutzerschnittstelle. Die Audiosignalverarbeitung wird durch die Audio Unit (AU) übernommen. Die Steuerung erfolgt über die Control Unit (CU), welche auch Teile der Benutzerschnittstelle beinhaltet. Die Benutzerschnittstelle besteht aus drei Teilen, der geräteseitigen Benutzerschnittstelle (User Interface - UI), der graphischen Benutzerschnittstelle (Graphical User Interface - GUI) und dem Webinterface (WI). Die GUI ist in die CU integriert. Die UI besteht aus den Eingabe- und Ausgabeelementen am Gerät selber. Diese sind Taster, Inkrementalgeber und LCD. Das LCD ist in die Display Unit (DU) integriert. Zur DU gehört das *I^2C-Bus LC-Display Modul* der Firma Conrad Electronic SE (Expander-Modul), welches dem LCD vorgeschaltet ist. Das Konzept des praktischen Aufbaus der Klangregelung zeigt Bild 35.

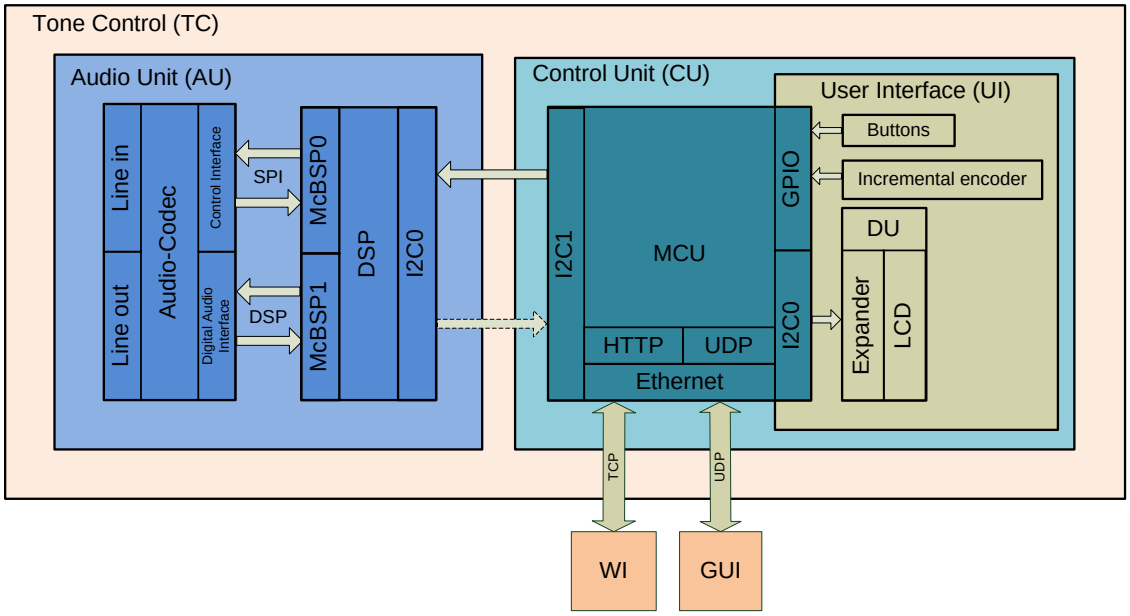


Bild 35: Konzept des praktischen Aufbaus der Klangregelung

4.1. Die Audio Unit (AU)

Die AU besteht aus dem TMS320C6713 Developer Starter Kit (DSK) der Firma Spectrum Digital, Incorporated und verwendet den Audio-Codec TLV320AIC23, den DSP C6713, den Flash-Speicher AM29LV400B, das Host Port Interface (HPI) und die Klinkenbuchsen für den Line in und Line out. Das Bild 36 zeigt das DSK mit den von der Audio Unit verwendeten Bestandteilen.

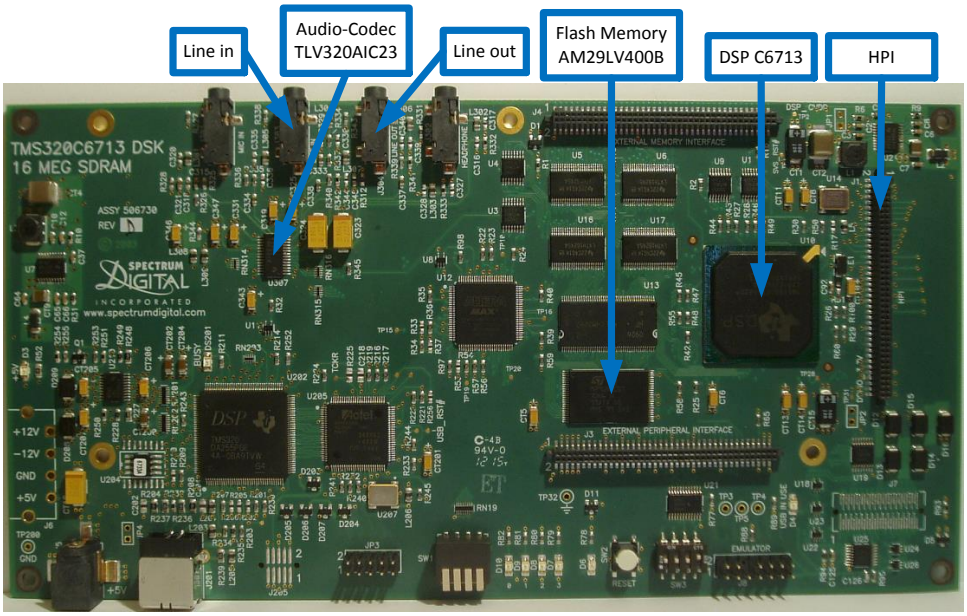


Bild 36: DSK mit den von der AU verwendeten Bestandteilen

Der Flash-Speicher wird dazu genutzt, um das Programm zur Audiosignalverarbeitung dauerhaft auf dem Board zu speichern. Beim Einschalten oder Rücksetzen des DSK wird das Programm aus dem Flash-Speicher in den L2-Speicher des DSP geladen und gestartet. Das Audio Signal Processing Program (ASPP) wird im Abschnitt 5.1 näher beschrieben.

Die analogen Audiosignale werden der AU über die Klinkenbuchsen *Line in* und *Line out* zugeführt bzw. entnommen. Bild 37 zeigt dazu die Rückseite des Klangreglers mit den zuvor erwähnten Klinkenbuchsen.

Bild 37: Rückseite des Gehäuses

Die Klinkenbuchsen sind über Kabelverbindungen, die an die Buchsen angelötet sind, mit den Klinkenbuchsen des DSK verbunden, wie dies Bild 38 zeigt.

Bild 38: Audioverbindung zwischen DSK und den Klinken-Buchsen

Diese Klinkenbuchsen sind mit dem Audio-Codec verbunden. Der Anschluss *Line in* ist so mit den Pins RLINEIN (rechter Kanal) und LLINEIN (linker Kanal) des Audio-Codecs verbunden. Die Pins ROUT (rechter Kanal) und LOUT (linker Kanal) des Audio-Codecs sind mit der Buchse *Line Out* verbunden.

Der Audio-Codec führt mit den über den Anschluss *Line in* zugeführten Audiosignalen, eine Analog-Digital-Wandlung durch und übergibt die digitalen Audiosignale über das Digital Audio Interface (DAI) an den McBSP0 des DSP. In umgekehrter Reihenfolge gelangen die vom DSP bearbeiteten Audiosginale zum Audio-Codec. Dieser führt eine Digital-Analog-Wandlung durch und gibt die analogen Audiosignale über den Anschluss *Line out* aus.

Der Austausch der Audiosignale zwischen Audio-Codec und DSP erfolgt im DSP-Format. Der Audio-Codec ist als Master für die Taktung konfiguriert. Er gibt damit den Takt von 12 MHz auf dem Signal *BCLK* aus. Im DSP-Format werden der rechte und linke Kanal in zwei Datenworten direkt nacheinander übertragen. Der Start der Übertragung wird mit dem Signal *LRCIN*

bzw. *LRCOUT* vom Audio-Codec durch einen Impuls angezeigt. Dabei erfolgt der Start der Datenübertragung mit der fallenden Flanke. Die beiden Signale müssen mit den Frame Sync Signalen *FSX2* bzw. *FSR2* des McBSP1 verbunden sein. Die Übertragung der Daten erfolgt über die Signalleitungen *DIN* bzw. *DOUT*, welche mit *DX2* bzw. *DR2* des McBSP1 verbunden sind. Bild 39 zeigt den zeitlichen Signalverlauf zum Austausch der Audiodaten im DSP-Format.

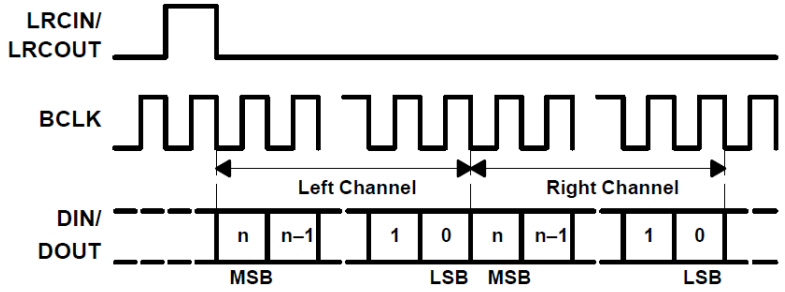


Bild 39: DSP Mode Timing[5]

Die Konfiguration des Audio-Codecs wird über den McBSP0 vorgenommen. Dieser ist mit dem Control Interface des Audio-Codecs verbunden. Die Datenübertragung erfolgt über das SPI-Format. Um dies dem Audio-Codec mitzuteilen, ist der Mode pin des Audio-Codecs auf High-Pegel gelegt. Den zeitlichen Signalverlauf des SPI-Formats kann Bild 40 entnommen werden.

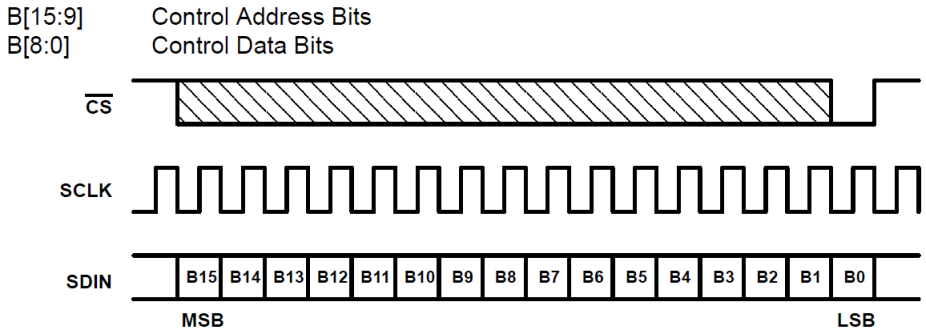


Bild 40: SPI Timing[6]

Das Chip-Select-Signal CS ist mit dem Frame-Sync-Signal FSX0 verbunden. Mit einer steigenden Flanke des CS-Signals wird ein 16-bit-Datenwort in den Audio-Codec übernommen. Der Austausch der Datenworte erfolgt über den Pin *SDIN* des Audio-Codecs und den Pin *DX0* des McBSP0 des DSP. Das 16-bit-Datenwort besteht aus der Adresse des Registers und dem Datum. Das Datum findet sich in den Bits 0 bis 8 und ist somit 9 bit groß. Die Adresse des Registers ist

[5] [2], S. 3-8, Figure 3-8
[6] [2], S. 3-1, Figure 3-1

in den Bits 9 bis 15 zu finden und ist 7 bit groß. Die Taktung erfolgt über die Pins *SCLK* des Audio-Codecs und *CLKX0* des McBSP0 des DSP. Bild 40 zeigt den zeitlichen Signalverlauf des SPI-Formats und Bild 41 zeigt die Signalwege der Audio-Unit-Komponenten. Weitere Informationen zum TMS320C6713 DSK finden sich in [5] und zum Audio-Codec TLV320AIC23 in [2].

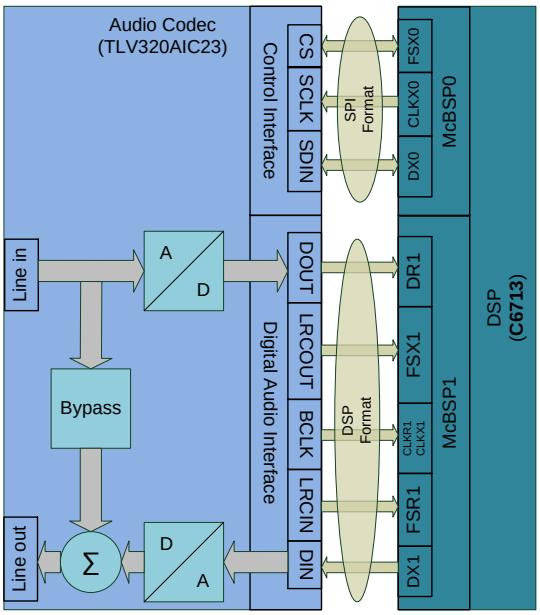


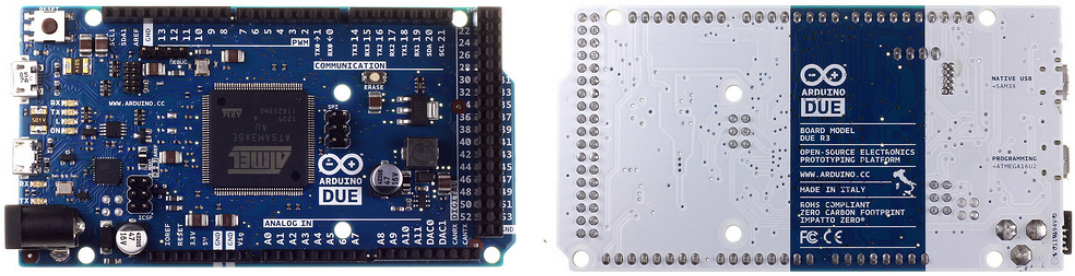
Bild 41: Signalwege der Audio-Unit-Komponenten

4.2. Die Control Unit (CU)

Die CU besteht aus der Mikrocontroller Unit (MCU) und dem User Interface (UI). Es hat die Aufgabe die Parameter der Audiosignalverarbeitung zu verwalten und damit die Eingaben aus den verschiedenen Benutzerschnittstellen zu koordinieren. Des Weiteren hat es die Ein- und Ausgabeelemente der UI an die Klangregelung anzubinden und für die Ethernet-Verbindungen entsprechende Server und Schnittstellen bereitzustellen.

4.2.1. Die Mikrocontroller Unit (MCU)

Die MCU ist die zentrale Schnittstelle zwischen der AU und den Benutzerschnittstellen. Die MCU besteht aus dem Mikrocontroller-Board *Arduino Due*. Es basiert auf der Atmel SAM3X8E ARM Cortex-M3 CPU. Die CPU ist mit 84 MHz getaktet und kann mit vielen Schnittstellen aufwarten. Genutzt werden für die MCU die GPIOs, die beiden I^2C-Schnittstellen und die Stromversorgungsanschlüsse am Power header. Mehr Informationen zum Arduino Due finden sich auf der Internetseite *http://arduino.cc/en/Main/ArduinoBoardDue*. Im Bild 42 ist das Arduino Due in Front- und Rückansicht zu sehen.

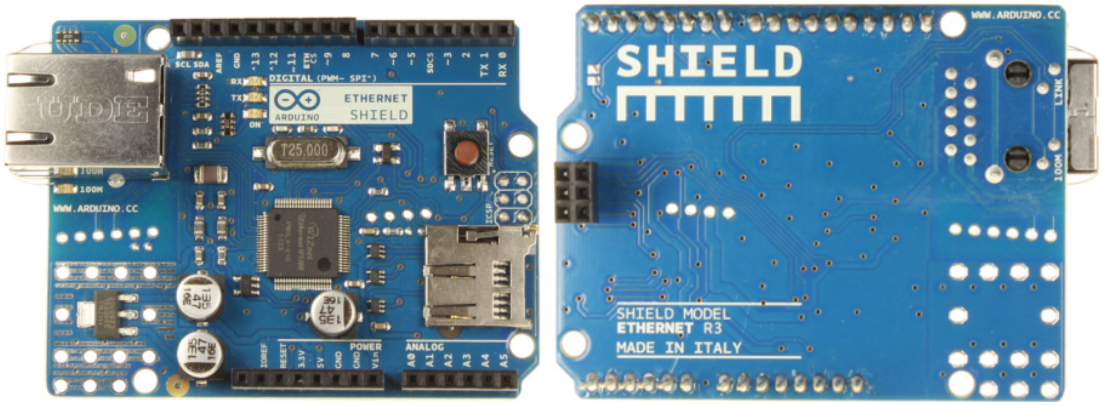


(a) Frontansicht[7] (b) Rückansicht[8]

Bild 42: Arduino Due

Für die Anbindung der GUI und des WI über eine Ethernet-Verbindung wird das Arduino Due ergänzt durch das Arduino Ethernet Shield, welches im Bild 43 in Front- und Rückansicht zu sehen ist. Das Ethernet Shield ist mit dem Ethernet Controler W5100 bestückt und kann mit 10 Mbit/s oder 100 Mbit/s betrieben werden. Der W5100 ist zu den Normen IEEE 802.3 10BASE-T und 802.3u 100BASE-TX konform. Weiterhin ist das Ethernet Shield mit einem micro-SD card reader bestückt, welcher von der MCU dazu verwendet wird um die Konfigurationen abzuspeichern, damit diese zu einem späteren Zeitpunkt wieder geladen werden können. Auf der micro-SD card ist auch das Webinterface hinterlegt.

[7] http://arduino.cc/en/uploads/Main/ArduinoDue_Front_450px.jpg
[8] http://arduino.cc/en/uploads/Main/ArduinoDue_Back_450px.jpg



(a) Frontansicht[9] (b) Rückansicht[10]

Bild 43: Arduino Ethernet Shield

Das Ethernet Shield wird auf das Arduino Due aufgesteckt, wie es Bild 44 zeigt. Für die Kommunikation zwischen Arduino Due und dem Ethernet Shield wird der SPI-Bus über den ICSP header verwendet. Dieser ist im Bild 43b mittig am linken Rand zu erkennen. Beim Arduino Due wurde der ICSP header mit SPI header bezeichnet. Die äußeren Pins des Ethernet Shields werden zum größten Teil nicht verwendet und nur durchgeschleift. Die Pins 4 und 10 werden als Chip-Select-Signale für die Kommunikation zwischen Arduino Due und dem W5100 bzw. dem micro-SD card reader verwendet. Dabei ist Pin 4 dem micro-SD card reader und Pin 10 dem W5100 zugeordnet.

Des Weiteren werden die Anschlüsse des Power headers für die Stromversorgung und das Reset-Signal verwendet. Der Pin 2 ist mit dem W5100 Interrupt (INT) verbunden. Pin 2 findet sich beim Arduino Due am PWML header und beim Ethernet Shield am JLOW header. Die Verbindung des Ethernet-Anschlusses (RJ45-Buchse) des Arduino Ethernet Shields mit der Außenwelt erfolgt über eine kurze Kabelverbindung zur RJ45-Buchse an der Rückwand des Gehäuses (siehe Bild 37).

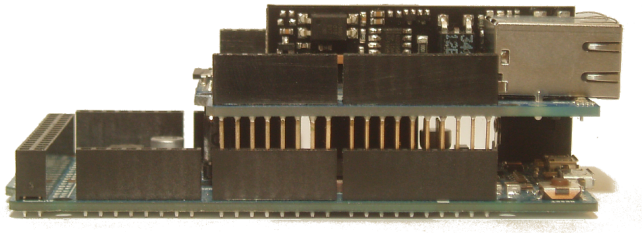


Bild 44: Kombination aus Arduino Due und Arduino Ethernet Shield

[9] http://arduino.cc/en/uploads/Main/ArduinoEthernetShield_R3_Front_450px.jpg
[10] http://arduino.cc/en/uploads/Main/ArduinoEthernetShield_R3_Back_450px.jpg

Mit den Bibliotheken *Ethernet* und *SD* werden Programmierschnittstellen zur Kommunikation zwischen Arduino Due und dem Ethernet Shield zur Verfügung gestellt. Daher wird diese Kommunikation hier nicht weiter behandelt und die Anwendung der Bibliotheken im Abschnitt 5.2 zum Control and User Interface Program (CUIP) näher beschrieben.

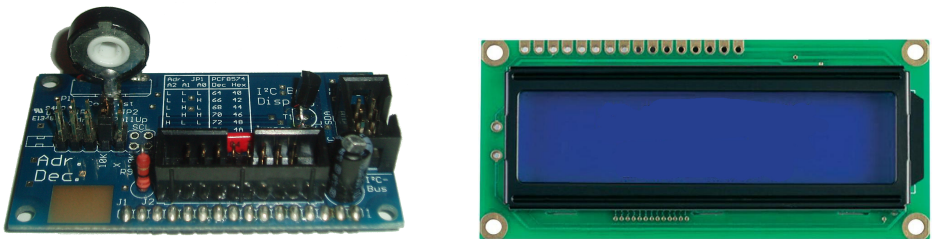
4.2.2. Das User Interface (UI)

Das UI ist die geräteseitige Benutzerschnittstelle. Das UI hat somit die Aufgabe, die Eingaben des Anwenders an die MCU weiterzugeben und die Ausgaben der MCU an das LCD weiterzuleiten. Das UI besteht aus den Ein- und Ausgabeelementen Taster, Inkrementalgeber und Display Unit (DU). Die Ein- und Ausgabeelemente sind am Gehäuse des Klangreglers angebracht, wie dies Bild 45 zeigt.

Bild 45: Ein- und Ausgabeelemente der Benutzerschnittstelle

Die mit F gekennzeichneten Elemente sind die Inkrementalgeber. Diesen werden je nach Zustand verschiedene Funktionen zugewiesen. Die Taster mit den Bezeichnungen *Bypass*, *Backlight* und *Reset* sind immer nur einer Funktion zugewiesen. Mit dem Taster *Bypass* lässt sich die Stummschaltung aktivieren oder deaktivieren. Über den Taster *Backlight* kann die Hintergrundbeleuchtung des LCD eingeschaltet werden. Durch betätigen des Tasters *Reset* wird die gesamte Klangregelung zurückgesetzt und die Standardkonfiguration geladen, welche auf dem Speicherplatz 0 bzw. default gespeichert ist.

Die DU besteht aus dem Expander-Modul und dem LCD GE-C1602B-TMI-JT/R der Firma Gleichmann & Co. Electronics GmbH. Bild 46 zeigt die Komponenten der DU. Das Modul stellt eine Datenverbindung zum Arduino Due I2C0 über den J3 header her. Das Arduino Due ist als Master und das Modul als Slave konfiguriert. Die Daten von der I^2C-Schnittstelle werden durch den *Remote 8-bit I/O expander for I2C-bus* in das entsprechende parallele Datenwort für das LCD gewandelt. Dieses wird über den J2 header an das LCD übertragen. Die Verbindung erfolgt über ein Flachbandkabel.



(a) I^2C-Bus LC-Display Modul (b) LCD GE-C1602B-TMI-JT/R

Bild 46: Display Unit (DU)

Um die einzelnen Komponenten miteinander zu verbinden, wurde eine Platine entwickelt. Diese Platine wird mit Connector Board bezeichnet und wird auf das Arduino Due gesteckt, wie es im Bild 47 zu sehen ist.

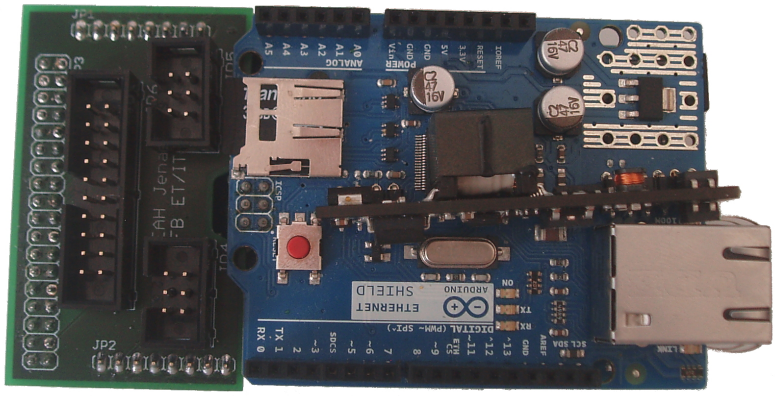


Bild 47: MCU vollständig mit Connector Board

Durch das Aufstecken der Platine werden die GPIOs des XIO headers auf dem Arduino Due mit dem JP3 header des Connector Boards verbunden. Diese sind direkt mit dem JP6 header verbunden an dem die Eingabeelemente über eine Flachbandkabelverbindung angeschlossen und somit direkt mit den GPIOs des Arduino Due verbunden sind.

Den Pins 8 und 7 des Communication headers auf dem Arduino Due sind den I^2C-Signalen der I2C0-Schnittstelle zugeordnet. Dem Pin 8 ist das Taktsignal SCL0-3 und dem Pin 7 das Datensignal SDA0-3 zugeordnet. Der Communication header ist direkt mit dem JP2 header des Connector Boards verbunden. Dieses gibt die Signale an den JP4 header weiter. Dort kommt über den JP5 header noch die 5 V Stromversorgung hinzu. Über eine weitere Flachbandkabelverbindung ist der JP4 header mit dem J3 header des Expander-Moduls verbunden. Somit sind die 5V-Stromversorgung und der I2C0 des Arduino Due mit dem Expander-Modul verbunden.

Die 5V-Stromversorgung kommt von der Verbindung zwischen dem Power header des Arduino Ethernet Shields und dem JP5 header des Connector Boards. Bei dieser Verbindung wird nicht nur die 5V-Stromversorgung an das Connector Board angeschlossen, sondern auch das Reset-Signal des Arduino Ethernet Shileds, welches diese direkt an den Arduino Due und das TMS320C6713 DSK weitergibt. Die Verbindung zum TMS320C6713 DSK erfolgt direkt vom Arduino Ethernet Shield zum HPI Pin 4 (RESETn-Signal). Beim Auslösen des Resets, wird so auch die Rücksetzanweisung an das DSP-Board weitergegeben und dieses ebenfalls zu einem Rücksetzen veranlasst. Weiterhin wird das Reset-Signal vom JP5 header an den JP6 header weitergegeben, womit eine Verbindung zum Reset-Taster der UI hergestellt wird. Dann wird das Reset-Signal vom JP5 header an den JP3 header und somit an den GPIO Pin 52 weitergegeben, damit auch über die Software ein Reset ausgelöst werden kann.

Der JP1 header ist durch das Aufstecken des Connector Boards auf das Arduino Due direkt mit dem ADCH header des Arduino Due verbunden. Hier werden keine Signale weitergegeben. Diese Steckverbindung dient nur der mechanischen Stabilität der Verbindung von Arduino Due und Connector Board.

Den Schaltplan für das Conector Board kann dem Anhang E.1.5 entnommen werden. Die Verbindungen zwischen den Komponenten des Klangreglers können dem Schaltplan im Anhang E.1.6 entnommen werden.

Eine Verbindung der MCU wurde hier noch nicht erwähnt. Es ist die Verbindung des I2C1 des Arduino Due zum I2C0 des TMS320C6713 DSK. Die Pins 10 und 9 des PWMH headers des Arduino Due sind direkt mit dem JHIGH header des Arduino Ethernet Shields verbunden und werden über eine Flachbandkabelverbindung an die Pins 68 und 72 des HPI des DSP-Boards weitergegeben. Der Takt SCL wird über die Verbindung der Pins 68 des HPI und des Pins 10 des Arduino Due übertragen. Das Arduino Due ist als Master und das TMS320C6713 DSK als Slave konfiguriert. Der Datenaustausch über das Signal SDA erfolgt über den Pin 72 des HPI auf dem DSP-Board und dem Pin 9 des Arduino Due.

4.3. Die Stromversorgung

Die Stromversorgung erfolgt über die Power-Buchse an der Rückwand des Gehäuses (siehe Bild 37). Innerhalb des Gehäuses sind an die Buchse zwei Kabel angelötet, welche am anderen Ende mit einem Stecker verbunden sind. Diese Stecker werden mit den Buchsen *Power Input* des TMS320C6713 DSK und *Power Supply* des Arduino Due verbunden. Der Klangregler ist mit 6 V Gleichspannung zu versorgen. Die durchschnittliche Stromaufnahme liegt bei 500 mA. Um eine gewisse Reserve sicherzustellen, sollte ein Netzteil verwendet werden, welches mindestens 750 mA zur Verfügung stellen kann.

5. Die Software

Die Software besteht aus den Teilen: Audiosignalverarbeitung, Benutzerschnittstelle, graphische Benutzerschnittstelle und Webinterface. Die Audiosignalverarbeitung beinhaltet die Software für das DSP-Board, die Benutzerschnittstelle die Steuerung und die Schnittstelle zur Benutzeroberfläche am Gerät sowie den HTTP-Server und UDP-Server. Die grafische Benutzeroberfläche ist ein Programm das auf einem PC unter den Betriebssystemen Windows und Linux ausgeführt werden kann und stellt zur Benutzerschnittstelle eine UDP-Verbindung her. Sie arbeitet somit als UDP-Client. Das Webinterface ist ein Programm das in einem Webbrowser ausgeführt wird und über HTTP eine Verbindung zur Steuerung bzw. dem HTTP-Server herstellt. In den folgenden Abschnitten sollen die einzelnen Bestandteile der Software näher erläutert werden.

5.1. Audio Signal Processing Program (ASPP)

Das Audio Signal Proccesing Program (ASPP) ist für die Audiosignalverarbeitung auf dem DSP-Board (TMS320C6713 DSK) zuständig und als Firmware ausgelegt. Des Weiteren verbindet es als I^2C-Slave das Board mit der Steuerung und stellt die Parameter entsprechend der Steuerbefehle ein, welche vom Mikrocontroller kommen, der als I^2C-Master fungiert. Das Programm wurde mittels der IDE *Code Composer Studio* in der Version 3.1.0 programmiert, welche dem Board beigelegt war. Programmiert wurde das Programm in den Programmiersprachen C und Assembler.

Beim Einschalten oder einem Rücksetzen (Reset) des DSP wird zunächst der Bootloader aufgerufen, welcher das Programm vom Flash-Speicher in den L2-Speicher des DSP überträgt. Anschließend werden die Interrupt Service Routinen (ISR) den Interrupts zugewiesen und zum Schluss wird das Hauptprogramm *main* aufgerufen. Bild 48 zeigt den Startvorgang als Programmablaufplan. Der Startvorgang wurde in Assembler programmiert und findet sich in der Datei *boot.asm*. Die Definitionen der Sektionen und Speicherbereiche ist in der Datei *lnk.cmd* hinterlegt.

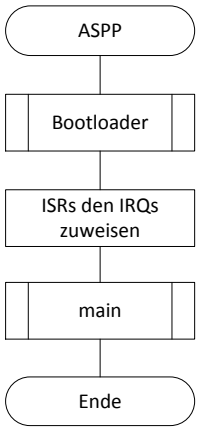


Bild 48: Programmablaufplan Audiosignalverarbeitung Startvorgang

5.1.1. Das Hauptprogramm *main*

Das Hauptprogramm *main* ist in der Datei *main.c* hinterlegt und ruft zunächst die Funktion *Init* auf. Anschließend wird in einer Endlosschleife immer wieder die Funktion *i2c_check* aufgerufen, wie es Bild 49 im abgebildeten Programmablaufplan zeigt.

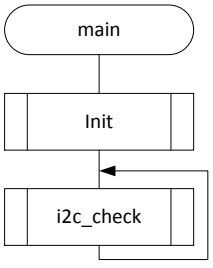


Bild 49: Programmablaufplan Audiosignalverarbeitung Hauptprogramm *main*

Mithilfe der Funktion *Init* wird die Klangregelung initialisiert. Es werden zunächst alle Interrupts deaktiviert. Anschließend wird die LUT zur Umrechnung vom logarithmischen zum linearen Verstärkungsfaktor initialisiert. Danach werden das Filter, die Audioverarbeitung, der Limiter und die I^2C-Schnittstelle initialisiert. Zum Schluss wird der globale Interrupt wieder aktiviert. Bild 50 zeigt den Programmablaufplan zur Funktion *Init*.

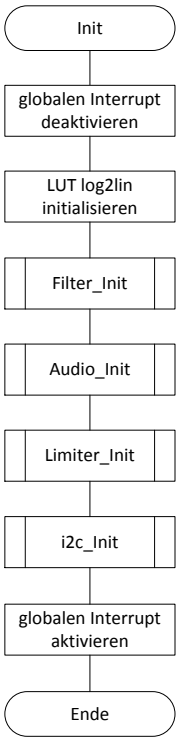
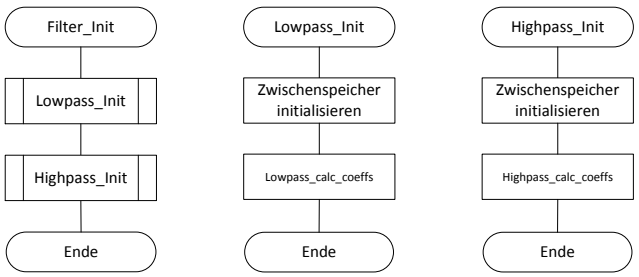


Bild 50: Programmablaufplan Audiosignalverarbeitung Funktion *Init*

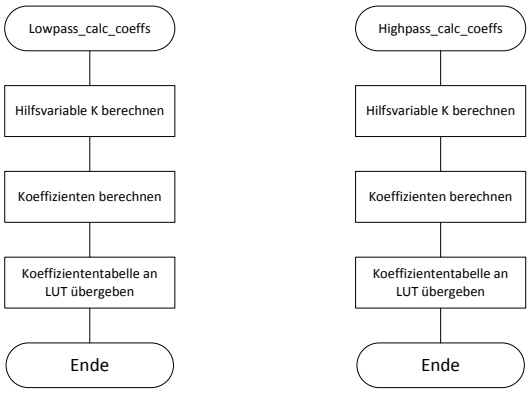
Die Funktion *Filter_Init* ruft die Funktionen *Lowpass_Init* und *Highpass_Init* zur Initialisierung des Tief- und Hochpasses auf. Mit der Funktion *Lowpass_Init* wird der Tiefpass initialisiert. Dazu werden, über die jeweils eingestellte Grenzfrequenz, die Koeffizienten für die LUT berechnet und die Zwischenspeicher gelöscht. Das Berechnen der Koeffizienten erfolgt über die Funktionen *Lowpass_calc_coeffs* und *Highpass_calc_coeffs*. Bild 51 zeigt den Programmablaufplan zur Funktion *Filter_Init*.



(a) Filter_Init (b) Lowpass_Init (c) Highpass_Init

Bild 51: Programmablaufpläne zur Initialisierung des Filters

Mit den Funktionen *Lowpass_calc_coeffs* und *Highpass_calc_coeffs* werden entsprechend der eingestellten Abtastfrequenz f_s und der Grenzfrequenz $f_{c_{lp}}$ bzw. $f_{c_{hp}}$ für den Tiefen- und Höhen-Shelving-Filter die Koeffizienten für die 61 wählbaren Verstärkungsfaktoren berechnet und in der LUT für die Tiefpass- bzw. Hochpass-Filterkoeffizienten hinterlegt. Dazu wird zunächst eine temporäre LUT angelegt und erst nach der Berechnung wird diese in die eigentliche LUT kopiert. Bild 52 zeigt die dazugehörigen Programmablaufpläne.



(a) Lowpass_calc_coeffs (b) Higpass_calc_coeffs

Bild 52: Programmablaufpläne zur Berechnung der Filterkoeffizienten

Die Funktion *Audio_Init* initialisiert die LUTs für den linken und rechten Kanal zur Umrechnung des Balancewertes in den entsprechenden Verstärkungsfaktor. Des Weiteren werden die Einstellungen zu *Volume*, *Balance*, *Line in Volume*, *De-Emphasis*, *DAC select* und *Bypass* initialisiert. Darüber hinaus wird der Audio-Codec konfiguriert. Bild 53 zeigt dazu den Programmablaufplan.

Im Programmablaufplan ist zu sehen, dass mehrere Funktionen aufgerufen werden, welche mit *set* beginnen. Diesen Funktionen ist ein bestimmter Wert zu übergeben, welcher validiert und anschließend genutzt wird um einen Parameter der Audiosignalverarbeitung einzustellen. Da diese Funktionen alle gleich sind und einfach aufgebaut sind, soll auf die Quellcodes und die Kommentaren zu den Funktionen verwiesen werden.

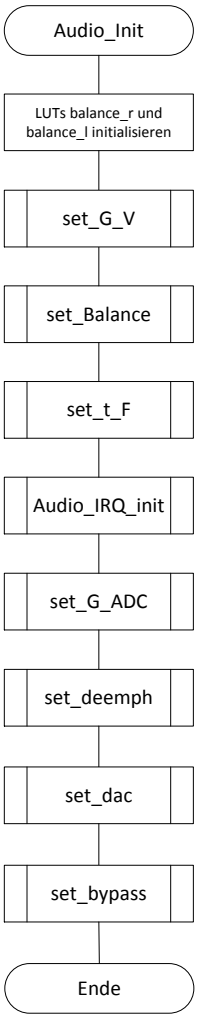
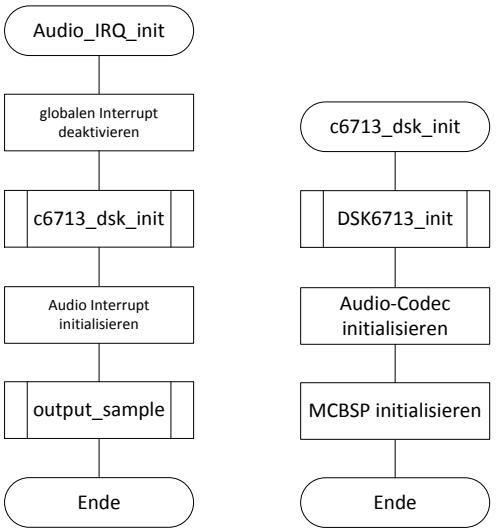


Bild 53: Programmablaufplan Audiosignalverarbeitung Funktion *Audio_Init*

Die Funktion *Audio_IRQ_init* sticht hier natürlich hervor. Diese initialisiert den Audio-Codec und den Interrupt für die Audioverarbeitung. Zunächst wird der globale Interrupt abgeschaltet. Danach wird die Funktion *c6713_dsk_init* aufgerufen, welche das DSP-Board initialisiert. Anschließend wird die *EventId* ermittelt und diese dem Interrupt 11 zugewiesen. Anschließend wird der Event und Interrupt aktviert. Zum Schluss wird der Austausch der Audiodaten zwischen Audio-Codec und DSP mithilfe der Funktion *output_sample* angestoßen.

Die Funktion *c6713_dsk_init* ruft die Funktion *DSK6713_init* auf. Diese Initialisiert die API (Application Programming Interface) des DSP-Boards. Danach wird der Audio-Codec initialisiert und auf die Standardkonfiguration eingestellt, die McBSP-Schnittstelle initialisiert und gestartet. Bild 54 zeigt die Programmablaufpläne der beiden Funktionen.



(a) Audio_IRQ_init (b) c6713_dsk_init

Bild 54: PAPs der Funktionen zur Initialisierung des Audio-Codecs und des DSP-Boards

Nach der Initialisierung der Audioverarbeitung erfolgt die Initialisierung des Limiters durch Aufruf der Funktion *Limiter_Init*. Als erstes wird die Periodendauer T_s der Abtastfrequenz f_s berechnet. Danach wird der Spitzenwert und der Verstärkungsfaktor des Limiters initialisiert. Anschließend werden die Funktionen *calc_threshold*, *calc_attack_time* und *calc_release_time* aufgerufen. Damit werden die Werte für die Limiterschwelle *LT*, die Attack time *AT* und die Release time *RT* berechnet. Zum Schluss wird der Puffer zur Verzögerung des Audiosignals mit Nullen initialisiert. Bild 55 zeigt den Programmablaufplan der Funktion *Limiter_Init*.

Nach der Funktion *Limiter_Init* wird als letzte Funktion die Funktion *i2c_Init* aufgerufen. Diese deaktiviert zunächst den globalen Interrupt und ruft dann die Funktion *I2C_open* auf. Dieser wird mit der Konstanten *I2C_DEV0* mitgeteilt, dass von den beiden I2C-Devices das Erste verwendet werden soll. Weiter wird der Funktion das Signal zum Rücksetzen des I2C-Devices übermittelt. Anschließend wird die Funktion *I2C_config* aufgerufen, womit die Konfiguration an das Device übergeben wird. Es wird hier als Slave verwendet und wartet somit auf den Master. Der nächste Schritt ist die Variable *addr* mit 255 zu initialisieren. Die Variable *addr* repräsentiert die Adresse zum Register in das ein Wert von der I^2C-Schnittstelle geschrieben werden soll. Die 255 ist allerdings nicht vergeben, womit die Funktion *i2c_check* weiß, dass kein Kommando von der Steuerung gesendet wurde. Der letzte Schritt ist das Aktivieren des globalen Interrupts. Bild 56 zeigt den Programmablaufplan zur Funktion *i2c_init*.

Nach Abschluss der Initialisierung geht das Hauptprogramm *main* in eine Endlosschleife über, in der immer wieder die Funktion *i2c_check* aufgerufen wird.

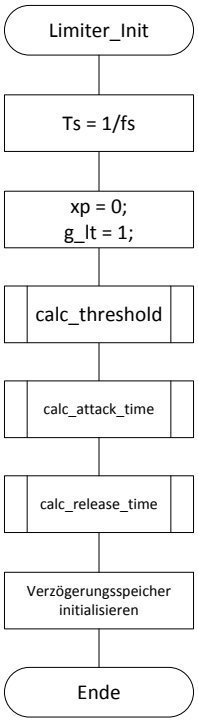


Bild 55: Programmablaufplan Audiosignalverarbeitung Funktion *Limiter_Init*

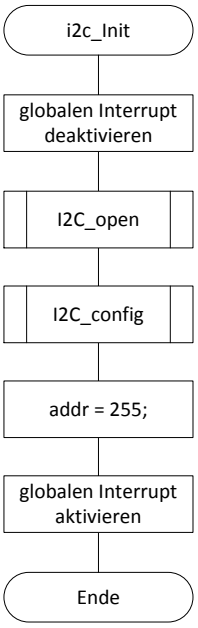


Bild 56: Programmablaufplan Audiosignalverarbeitung Funktion *i2c_init*

5.1.2. Das Unterprogramm *i2c_check*

Die Funktion *i2c_check* prüft zunächst, mithilfe der Funktion *I2C_rrdy*, das *Receive Data Ready Interrupt Flag*. Liegen Daten, beim I2C-Device, zum Abruf bereit, dann ist das Flag gesetzt. Ist dies der Fall wird ein Datum vom I2C-Device abgerufen. Dies erfolgt mit der Funktion *I2C_readByte*. Das Datum wird in der Variablen *code* hinterlegt. Anschließend wird die Variable *addr* auf den Wert 255 geprüft. Hat die Variable den Wert 255 wird der Wert der Variablen *addr*

auf den der Variablen *code* gesetzt. Mit dem Wert 255 wird erkannt, dass dies in der Kette der Übertragung von der Steuerung das erste Datum ist und somit es sich um die Adresse des Registers handelt, hinter dem der entsprechende Parameter steckt, welcher verändert werden soll.

Hat die Variable *addr* nicht den Wert 255, so wurde die Adresse bereits gesendet und das Datum beinhaltet den Wert zur Einstellung des entsprechende Parameters. Es ist aber auch möglich, dass die Steuerung einen Wert abfragen möchte, daher wird der Wert der Variablen *addr* mit 0x80 UND-Verknüpft und in der Variablen *rw* hinterlegt. Ob gelesen oder geschrieben werden soll, hängt also vom achten Bit des Datums ab. Ist das Bit gesetzt, so handelt es sich beim Datum um einen Wert zum Einstellen eines Parameters. Ist das Bit nicht gesetzt, so ist das Datum irrelevant, da in der Variablen *addr* hinterlegt ist, welcher Parameterwert an die Steuerung übertragen werden soll. Das Übertragen eines Parameterwertes an die Steuerung ist hier zwar schon vorgesehen, aber noch nicht umgesetzt und in der Steuerung auch nicht vorgesehen.

Im weiteren Verlauf wird die Variable *addr* mit dem Wert 0x7F UND-Verknüpft, um damit das Schreib/Lese-Bit zu löschen und so den Wert zur Einstellung eines Parameters zu extrahieren. Somit ergibt sich, dass für 8-Bit-Parameter und für das obere Byte eines 16-Bit-Parameters nur 7 bit für den Wert zur Verfügung stehen. Soll ein Parameter geändert werden, so wird dieser anhand der Adresse des entsprechenden Parameter identifiziert und anhand des übertragenen Steuerparameters geändert. Sollte es sich um einen 16-Bit-Parameter handeln und es sich um das erste Datum des Parameters handeln, so ist dies das obere Byte. Dieses wird in der Variablen *high* hinterlegt. Das nächste übertragene Datenwort ist das untere Byte. Dieses wird in der Variablen *low* gespeichert. Anschließend wird die Variable *high* in die Variable *valU16* kopiert. Die Variable *valU16* ist ein unsigned 16-Bit-Integer. Nach dem Kopieren wird die Variable *valU16* einer Rechtsschiebung um 8 bit unterzogen, damit es als oberes Byte abgelegt ist. Danach wird die Variable *valU16* mit der Variablen *low* ODER-Verknüpft. Dies hat zur Folge, dass das Byte der Variablen *low* als unteres Byte in der Variablen *valU16* hinterlegt ist. Damit ist das 16-Bit-Wort vollständig in der Variablen *valU16* hinterlegt und kann zur Einstellung des entsprechenden Parameters verwendet werden. In der Regel werden dazu Funktionen mit dem Vorsatz *set* aufgerufen. Ist der Vorgang der Parametereinstellung abgeschlossen wird die Variable *addr* wieder auf den Wert 255 eingestellt. Weitere Informationen zum I^2C-Modul des DSP findet sich in [6]. Eine Übersicht der Parameter findet sich im Anhang C. Bild 57 zeigt den Programmablaufplan der Funktion *i2c_check*.

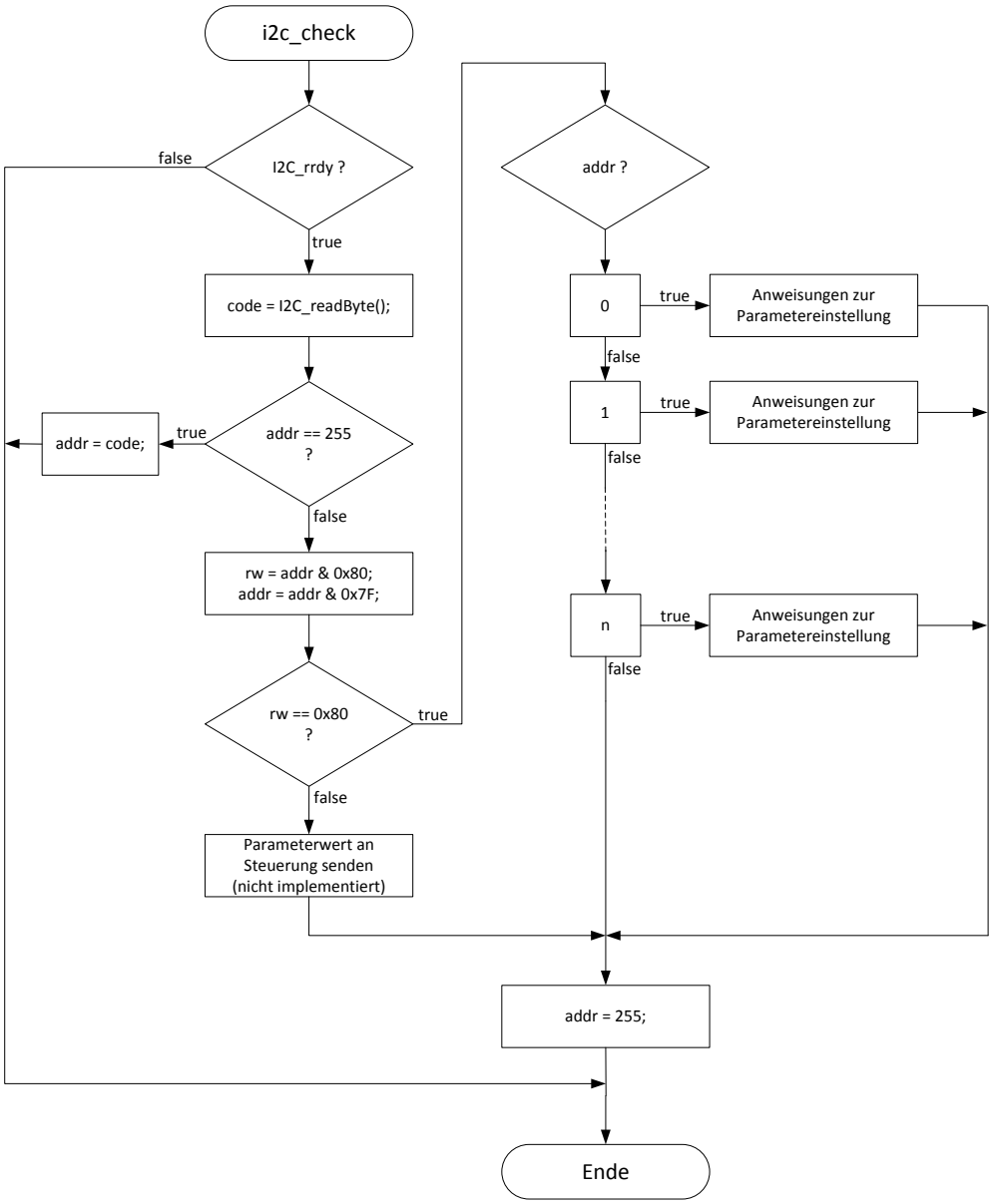


Bild 57: Programmablaufplan Audiosignalverarbeitung Funktion *i2c_check*

5.1.3. Die Interrupt Service Routine (ISR) *audio_isr*

Die Funktion *audio_isr* ist die Interrupt Service Routine (ISR), welche auf die Anforderung des Audio-Codecs zum Lesen oder Schreiben von Audiodaten reagiert. Zunächst werden die Audiodaten vom Audio-Codec abgeholt und in den rechten und linken Kanal aufgesplittet und anschließend in 32-Bit-Fließkommazahlen (float) gewandelt. Danach wird das Durchlaufen der Audiodaten durch die Blöcke *Filter*, *Audio* und Limiter veranlasst. Zum Schluss erfolgt die Umwandlung in eine Festkommazahl und die Zusammenführung des linken und rechten Kanals zu einem unsigned 32-Bit-Integer sowie der Übergabe der Daten an den Audio-Codec. Bild 58 zeigt den Programmablaufplan der ISR *audio_isr*.

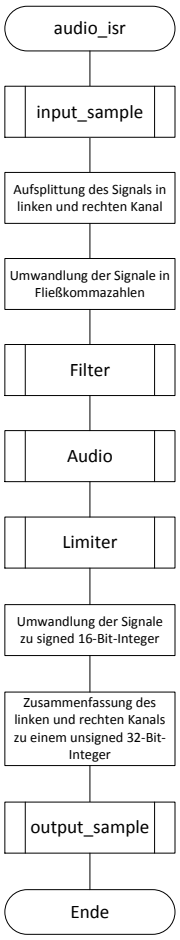


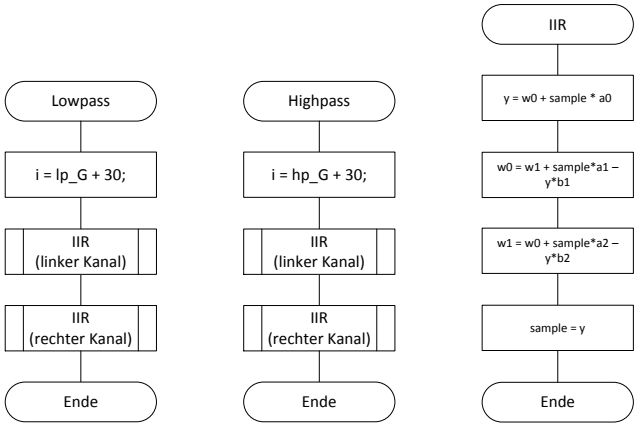
Bild 58: PAP der Funktion *audio_isr*

Für das Abholen der Daten vom Audio-Codec wird die Funktion *input_sample* aufgerufen. Dieses ruft wiederum die Funktion *MCBSP_read* auf, welches das über den *MCBSP* gesendete Audiosignal zurückgibt. Die Daten sind in einem unsigned 32-Bit-Integer hinterlegt. Die oberen 16 bit beinhalten den Wert des linken und die unteren 16 bit den Wert des rechten Kanals.

Zur Übergabe der verarbeiteten Audiosignale an den Audio-Codec wird die Funktion *output_sample* aufgerufen. Dieser Funktion wird ein unsigned 32-Bit-Integer übergeben, welcher in den oberen 16 bit den Wert des linken und in den unteren 16 bit den Wert des rechten Kanals beinhaltet. Die Funktion ruft ihrerseits die Funktion *MCBSP_write* auf, um über den *McBSP1* das Datum an den Audio-Codec zu übertragen.

Um das Audiosignal der Filterung mit dem Tiefen- und Höhen-Shelving-Filter zu unterziehen wird die Funktion *Filter* aufgerufen. Mit der Funktion werden die Audiosignale entsprechend der Konfiguration der Filterung durch *Lowpass* und *Highpass* unterzogen. Ist die Filterung eines der beiden Filter deaktiviert oder für einen bestimmten Kanal, so wird entsprechend nur das Signal verzögert und nicht gefiltert zurückgegeben. Die Funktion *Lowpass* extrahiert aus der LUT die entsprechenden Koeffizienten anhand des eingestellten Verstärkungsfaktors G_{lp} und übergibt diese zusammen mit dem Audiosignal der Funktion *IIR*, welche die Berechnungen vornimmt. Für die Funktion *Highpass* gilt gleiches, außer dass hier

Koeffizienten des Höhen-Shelving-Filters entsprechend des Verstärkungsfaktors G_{hp} extrahiert werden. Bild 59 und 60 zeigen die Programmablaufpläne für die Funktionen *Filter*, *Lowpass*, *Highpass* und *IIR*.



(a) Lowpass (b) Higpass (c) IIR

Bild 59: Programmablaufpläne der Unterfunktionen zur Filterung des Audiosignals

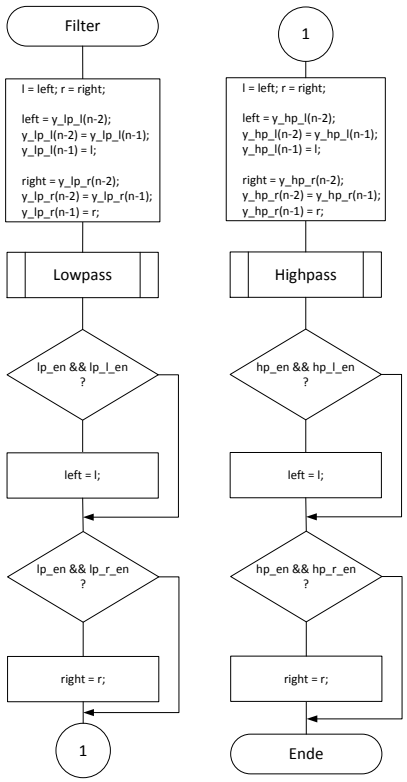


Bild 60: Programmablaufplan Audiosignalverarbeitung Funktion *Filter*

5.2. Control and User Interface Program (CUIP)

Das CUIP ist die zentrale Schnittstelle zwischen der Audio Unit und den Benutzerschnittstellen. Es registriert die Eingaben des Benutzers und verarbeitet diese zu Steuersignalen die es an die AU sendet. Des Weiteren werden die Bildschirminhalte des LCD erstellt und an die DU weitergeleitet. Weiterhin wird ein HTTP- und UDP-Server bereitgestellt. Über diese können die externen Benutzerschnittstellen mit der CU kommunizieren. Eine weitere Aufgabe ist das Speichern und Laden von Konfigurationen auf die micro SD-Card.

Das CUIP ist objektorientiert in der Programmiersprache C++ programmiert und ist im internen Flash-Speicher der Atmel SAM3X8E ARM Cortex-M3 CPU des Arduino Due abgelegt. Zur Erstellung des Programms wurde die von Arduino eigens erstellte Entwicklungsumgebung verwendet. Da das Arduino Due benutzt wird, ist die Version 1.5 zu verwenden.[11]

Wie schon erwähnt wurde ist das Programm objektorientiert programmiert. Daher sollen zunächst die Klassen mit ihren Attributen und Methoden vorgestellt werden. Im Anschluss folgen die Erläuterungen zum Startvorgang und dem allgemeinen Programmablauf.

5.2.1. Die Klasse CUIP_DSP

Die Klasse CUIP_DSP stellt die Schnittstelle zur Audio Unit bereit. Weiterhin hält es die Konfigurationsdaten zur Änderung bzw. zum Abrufen bereit. Wurde ein Parameter der Konfiguration geändert, so veranlasst die Klasse das Ändern der entsprechenden Parameter in der AU. Eine Rückmeldung der AU an das CUIP war vorgesehen, wurde aber nicht umgesetzt.

Die Kommunikation mit der AU erfolgt über den I2C1 des Arduino Due und dem I2C0 des DSP C6713 auf dem TMS320C6713 DSK. Der I2C1 ist als Master konfiguriert und I2C0 als Slave. Für die Kommunikation wird die Bibliothek *Wire*[12] verwendet. Diese stellt Schreib- und Lesemethoden sowie Methoden zur Abfrage des Zustandes der I^2C-Schnittstelle zur Verfügung.

[11] Die Arduino IDE kann über die Website *http://arduino.cc/en/Main/Software* heruntergeladen werden. Die Version 1.5.4 für Windows und Linux ist auf der beigelegten CD enthalten.

[12] http://arduino.cc/en/Reference/Wire

Auf der Hardwareebene erfolgt die Kommunikation über die Signalleitung SDA. Die Übertragung ist getaktet über die Signalleitung SCL. Zunächst muss die Adresse des ICs gesendet werden an den die Nachricht gehen soll. Die Adresse besteht aus 7 Bits. Das 8. Bit dient als Kennzeichnung ob geschrieben oder gelesen werden soll. Nach der Adresse folgt das Datum. Bild 61 zeigt den zeitlichen Ablauf der Kommunikation über die I^2C-Schnittstelle.

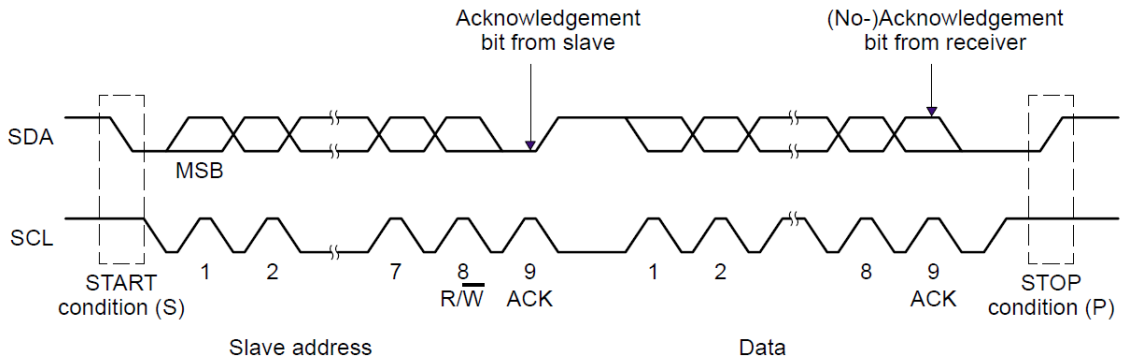


Bild 61: zeitlicher Verlauf einer Kommunikation über eine I^2C-Schnittstelle[13]

Auf der Anwendungsebene wird für die Kommunikation zwischen ASPP und CUIP kein bestimmtes Protokoll verwendet. Es ist zunächst die Adresse des Parameters und anschließend ein 8-Bit-Datenwort zu senden. Sollte der Parameter ein 16-Bit-Datenwort benötigen, so ist nacheinander zunächst die Adresse und Datum für das obere Byte und anschließend für das untere Byte zu senden (siehe Bild 62).

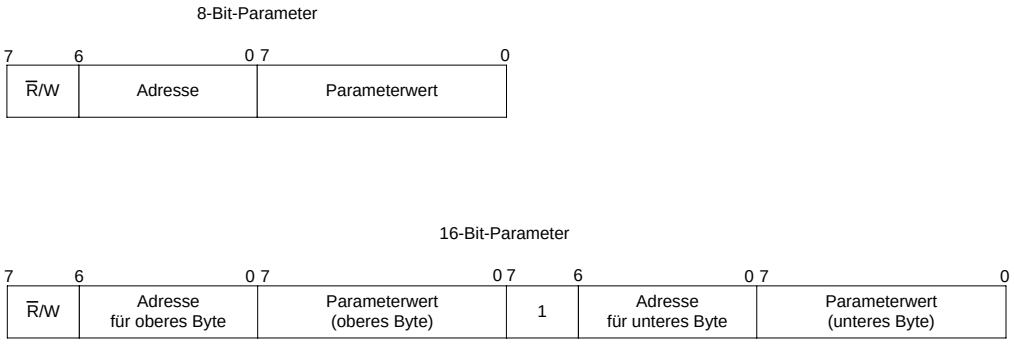


Bild 62: zeitlicher Verlauf der Kommunikation zwischen ASPP und CUIP

[13] [6], S. 8, Figure 5

Das Bit 7 der Adresse gibt jeweils an, ob geschrieben oder gelesen werden soll. Das Auslesen eines Parameterwertes ist zwar schon in Teilen vorhanden, aber nicht vollständig umgesetzt. Zum Überschreiben eines Parameterwertes ist somit das Bit 7 einer Adresse auf 1 und zum Auslesen auf 0 zu setzen. Somit begründet es sich auch, warum das Bit 7 der Adresse für das untere Byte auf 1 gesetzt ist, denn wollte man einen Parameter auslesen, so erfolgt dies über den zeitlichen Verlauf für ein 8-Bit-Parameter unabhängig davon ob es sich um einen 16-Bit-Parameter handelt, da nach dem Senden der 1. Kombination aus Adresse und Datum, der angeforderte Parameterwert zurück gesendet werden würde. Eine Übersicht der Parameter findet sich im Anhang C.

Die Konfiguration der Audiosignalverarbeitung wird im Attribut *config* gespeichert. Dieses Attribut basiert auf der Datenstruktur *dsp_config*, welche folgende Auflistung zeigt.

Listing 1: cuip_dsp.h

```
// DSP Konfigurationsstruktur definieren
106 struct dsp_config {
    int s_G_ADC;     //  Code des AIC23 zur Einstellung der Eingangslautstärke
108 int s_d;         //  De-Emphasis enabled/disabled
    int s_DAC_en;    //  DAC select
110 int s_b;         //  Bypass
    int s_lp_en;     //  Ein-/Ausschalten Tiefpass
112 int s_lp_l_en;   //  Ein-/Ausschalten Tiefpass linker Kanal
    int s_lp_r_en;   //  Ein-/Ausschalten Tiefpass rechter Kanal
114 int s_lp_fc;     //  Tiefpass - Grenzfrequenz in Hz
    int s_lp_G;      //  logarithmischer Verstärkungsfaktor des Tiefpasses in dB
116 int s_hp_en;     //  Ein-/Ausschalten Hochpass
    int s_hp_l_en;   //  Ein-/Ausschalten Hochpass linker Kanal
118 int s_hp_r_en;   //  Ein-/Ausschalten Hochpass rechter Kanal
    int s_hp_fc;     //  Hochpass - Grenzfrequenz
120 int s_hp_G;      //  logarithmischer Verstärkungsfaktor des Hochpasses in dB
    int s_G_V;       //  logarithmischer Verstärkungsfaktor fürs Volume
122 int s_t_F;       //  Mute Ein-/Ausblendzeit
    int s_B;         //  Balance (rechter - linker - Kanal)
124 int s_m;         //  Ein-/Ausschalten Mute
    int s_lt_en;     //  Ein-/Ausschalten Limiter
126 int s_lt;        //  Threshold in %
    int s_ta;        //  Attack Time in µs
128 int s_tr;        //  Release Time in ms
};
```

Soll ein Parameter geändert werden, so ist die Methode *set* aufzurufen. Soll auf einen Parameterwert ein bestimmter Wert addiert oder subtrahiert werden, so sind die Methoden *add* bzw. *sub* aufzurufen. Den Methoden ist das Schlüsselwort, welches den Parameter bestimmt der geändert werden soll und der Wert auf den der Parameter geändert werden soll bzw. den Wert der auf den Parameter addiert oder subtrahiert werden soll zu übergeben. Als erstes validieren die Methoden den übergebenen Wert. Je nachdem, in welche Richtung der Wert den Parameter aus seinem Wertebereich bringen würde, wird der Parameter auf einen Minimal- oder Maximalwert gesetzt. Nach Änderung des Parameterwertes wird dieser über die Methode *write* an das ASPP übertragen. Der Methode *write* ist dazu das Kommando des Parameters

und der neue Parameterwert zu übergeben. Über das Kommando wird bestimmt ob es sich um einen 8- oder 16-Bit-Parameter handelt und welche Adressen zum Parameter gehören. Das Kommando ist ebenfalls ein struct-Element, wie es folgende Auflistung zeigt.

Listing 2: cuip_dsp.h

```
// Strukturdefinitionen für Kommandos die an den DSP gesendet werden,
// um dort die Konfiguration zu ändern.

union dsp_cmd {
  byte cmd;
  byte cmds[2];
};

struct dsp_command {
  dsp_cmd cmd;      // Kommando
  byte bytes;       // Anzahl zu übertragender Bytes
};

struct dsp_commands {
  dsp_command s_G_ADC;
  dsp_command s_d;
  dsp_command s_DAC_en;
  dsp_command s_b;
  dsp_command s_lp_en;
  dsp_command s_lp_l_en;
  dsp_command s_lp_r_en;
  dsp_command s_lp_fc;
  dsp_command s_lp_G;
  dsp_command s_hp_en;
  dsp_command s_hp_l_en;
  dsp_command s_hp_r_en;
  dsp_command s_hp_fc;
  dsp_command s_hp_G;
  dsp_command s_G_V;
  dsp_command s_t_F;
  dsp_command s_B;
  dsp_command s_m;
  dsp_command s_lt_en;
  dsp_command s_lt;
  dsp_command s_ta;
  dsp_command s_tr;
};
```

Während der Initialisierungsphase werden diese Kommandos wie folgt initialisiert:

Listing 3: cuip_dsp.cpp

```
/*
  Die Methode init_commands initialisiert die Kommandos mit den Adressen. Die Adressen dienen dem DSP zur Unterscheidung der
      Kommandos.
  Des Weiteren werden den Kommandos Angaben zugewiesen, welche Aussagen, ob ein Byte oder zwei Bytes gesendet werden müssen.
      Dies benötigt
  die Methode write. Sind zwei Bytes zu senden, so muss die Adresse für das Low- und High-Byte übergeben werden.
*/
void CUIP_DSP::init_commands(void)
{
  cmd.s_G_ADC.cmd.cmd      =  0;                                  cmd.s_G_ADC.bytes    = 1;
  cmd.s_d.cmd.cmd          =  1;                                  cmd.s_d.bytes        = 1;
  cmd.s_DAC_en.cmd.cmd     =  2;                                  cmd.s_DAC_en.bytes   = 1;
  cmd.s_b.cmd.cmd          =  3;                                  cmd.s_b.bytes        = 1;
  cmd.s_lp_en.cmd.cmd      =  4;                                  cmd.s_lp_en.bytes    = 1;
  cmd.s_lp_l_en.cmd.cmd    =  5;                                  cmd.s_lp_l_en.bytes  = 1;
  cmd.s_lp_r_en.cmd.cmd    =  6;                                  cmd.s_lp_r_en.bytes  = 1;
  cmd.s_lp_fc.cmd.cmds[0]  =  7; cmd.s_lp_fc.cmd.cmds[1]  =  8; cmd.s_lp_fc.bytes    = 2;
  cmd.s_lp_G.cmd.cmd       =  9;                                  cmd.s_lp_G.bytes     = 1;
```

```
328   cmd.s_hp_en.cmd.cmd      = 10;                                  cmd.s_hp_en.bytes    = 1;
      cmd.s_hp_l_en.cmd.cmd    = 11;                                  cmd.s_hp_l_en.bytes  = 1;
330   cmd.s_hp_r_en.cmd.cmd    = 12;                                  cmd.s_hp_r_en.bytes  = 1;
      cmd.s_hp_fc.cmd.cmds[0] = 13; cmd.s_hp_fc.cmd.cmds[1] = 14; cmd.s_hp_fc.bytes    = 2;
332   cmd.s_hp_G.cmd.cmd       = 15;                                  cmd.s_hp_G.bytes     = 1;
      cmd.s_G_V.cmd.cmd        = 16;                                  cmd.s_G_V.bytes      = 1;
334   cmd.s_t_F.cmd.cmd        = 17;                                  cmd.s_t_F.bytes      = 1;
      cmd.s_B.cmd.cmd          = 18;                                  cmd.s_B.bytes        = 1;
336   cmd.s_m.cmd.cmd          = 19;                                  cmd.s_m.bytes        = 1;
      cmd.s_lt_en.cmd.cmd      = 20;                                  cmd.s_lt_en.bytes    = 1;
338   cmd.s_lt.cmd.cmd         = 21;                                  cmd.s_lt.bytes       = 1;
      cmd.s_ta.cmd.cmds[0]     = 22; cmd.s_ta.cmd.cmds[1]     = 23; cmd.s_ta.bytes       = 2;
340   cmd.s_tr.cmd.cmds[0]     = 24; cmd.s_tr.cmd.cmds[1]     = 25; cmd.s_tr.bytes       = 2;
}
```

Auch die Schlüsselwörter sind in einer Struktur hinterlegt, welches die folgende Auflistung zeigt.

Listing 4: cuip_dsp.h

```
// Schlüsselworte – wird in diversen Methoden zur Erkennung des Parameters benötigt.
174 struct dsp_Keys {
      byte s_G_ADC;
176   byte s_d;
      byte s_DAC_en;
178   byte s_b;
      byte s_lp_en;
180   byte s_lp_l_en;
      byte s_lp_r_en;
182   byte s_lp_fc;
      byte s_lp_G;
184   byte s_hp_en;
      byte s_hp_l_en;
186   byte s_hp_r_en;
      byte s_hp_fc;
188   byte s_hp_G;
      byte s_G_V;
190   byte s_t_F;
      byte s_B;
192   byte s_m;
      byte s_lt_en;
194   byte s_lt;
      byte s_ta;
196   byte s_tr;
};
```

Initialisiert werden die Schlüsselwörter wie folgt:

Listing 5: cuip_dsp.cpp

```
/*
346   Die Methode init_keys initialisiert die Keys über die das Schreiben oder Lesen von Konfigurationswerten gesteuert wird.
      Ein Key ist immer anzugeben, damit die Klasse CUIP_DSP weiß, welcher Konfigurationswert gemeint ist. Das Attribute
348   keys entspricht der Struktur dsp_keys und enthält für jeden Key einen Bytewert.
*/
350 void CUIP_DSP::init_keys(void)
{
352   keys = {
        0,  //  Eingangslautstärke (Line In)
354     1,  //  De-Emphasis
        2,  //  DAC select
356     3,  //  Bypass
        4,  //  Enable/Disable Tiefpass
358     5,  //  Enable/Disable Tiefpass linker Kanal
```

```
      6,  //  Enable/Disable Tiefpass rechter Kanal
360   7,  //  Grenzfrequenz Tiefpass
      8,  //  logarithmischer Verstärkungsfaktor Tiefpass
362   9,  //  Enable/Disable Hochpass
     10,  //  Enable/Disable Hochpass linker Kanal
364  11,  //  Enable/Disable Hochpass rechter Kanal
     12,  //  Grenzfrequenz Hochpass
366  13,  //  logarithmischer Verstärkungsfaktor Hochpass
     14,  //  Volume
368  15,  //  Ein-/Ausblendzeit beim Ein-/Ausschalten des Mute
     16,  //  Balance
370  17,  //  Mute
     18,  //  Enable/Disable Limiter
372  19,  //  Threshold Limiter
     20,  //  Attack Time Limiter
374  21,  //  Release Time Limiter
  };
376 }
```

Mit der Methode *set_config* kann die komplette Konfiguration geändert werden. Dazu ist die neue Konfiguration in Form der Struktur *dsp_config* der Methode zu übergeben. Diese validiert die neue Konfiguration und überschreibt die alte Konfiguration mit der Neuen. Anschließend wird die Methode *init_dsp* aufgerufen. Diese überträgt nacheinander alle Parameter an das ASPP. So wird Stück für Stück die neue Konfiguration an die AU übertragen.

Soll ein bestimmter Parameter ausgelesen werden, so kann dies mit der Methode *get* erfolgen. Der Methode ist das Schlüsselwort des Parameters zu übergeben und diese gibt den Wert des Parameters zurück. Möchte man die gesamte Konfiguration abrufen, so ist die Methode *get_config* zu verwenden. Dieser ist kein Parameter zu übergeben. Die Methode gibt die gesamte Konfiguration in der Struktur *dsp_config* zurück.

Eine weitere öffentliche Methode der Klasse CUIP_DSP ist die Methode *set_to_factory_settings*. Ruft man diese Methode auf, werden alle Parameter der Konfiguration auf die Standardwerte gesetzt und auf der micro-SD card auf dem Speicherplatz 0 (default) gespeichert. Anschließend wird der Klangregler zurückgesetzt (Reset). Beim Start des Klangreglers wird diese Konfiguration ausgelesen und die AU auf diese eingestellt.

Die Klasse CUIP_DSP hat weitere private Methoden, welche zur Initialisierung und Validierung aufgerufen werden.

5.2.2. Die Klasse CUIP_GPIO

Die Klasse CUIP_GPIO übernimmt die Aufgabe, die GPIOs der MCU zu initialisieren und Eingaben vom Anwender über die Eingabeelemente, welche bei den GPIOs, auflaufen zu verarbeiten. Dies können sein Änderungen an den Parametern der Audiosignalverarbeitung, Menü-Aufrufe, Speichern/Laden der Konfiguration, Einschalten der Hintergrundbeleuchtung und das Zurücksetzen des Klangreglers.

Der folgenden Auflistung kann die Zuordnung der GPIOs zu den Eingabeelementen entnommen werden.

Listing 6: cuip_gpio.h

```
// Pinzuweisung für Inkrementalgeber F1
#define  Key_1_Pin_A    42
#define  Key_1_Pin_B    45
#define  Key_1_Pin_D    44

// Pinzuweisung für Inkrementalgeber F2
#define  Key_2_Pin_A    41
#define  Key_2_Pin_B    40
#define  Key_2_Pin_D    43

// Pinzuweisung für Inkrementalgeber F3
#define  Key_3_Pin_A    36
#define  Key_3_Pin_B    39
#define  Key_3_Pin_D    38

// Pinzuweisung für Inkrementalgeber F4
#define  Key_4_Pin_A    35
#define  Key_4_Pin_B    34
#define  Key_4_Pin_D    37

// Pinzuweisung für Inkrementalgeber F5
#define  Key_5_Pin_A    30
#define  Key_5_Pin_B    33
#define  Key_5_Pin_D    32

// Pinzuweisung für Taster S1 (Reset)
#define  Reset_Pin      52

// Pinzuweisung für Taster S2 (Backlight)
#define  Backlight_Pin 46

// Pinzuweisung für Taster S3 (Bypass)
#define  Bypass_Pin     47
```

Bei der Initialisierung der GPIOs ist darauf zu achten, dass die internen Pull-Up-Widerstände der MCU zugeschaltet werden. Die Zuweisung erfolgt über die Methode *init*, wie es folgende Auflistung zeigt. Die Methode *init* wird während des Startvorgangs aufgerufen, welcher im Abschnitt 5.2.8 näher beschrieben wird.

Listing 7: cuip_gpio.cpp

```
62 /*
     Initialisierungsmethode, welche von der externen Methode aufzurufen ist, wenn
64   eine Instanz der Klasse CUIP_GPIO erzeugt werden soll. Die Initialisierungsmethode
     stellt die Richtung der verwendeten GPIOs ein und initialisiert die Statuswerte.
66 */
   void CUIP_GPIO::init(void)
68 {
     // Reset_Taster initialisieren
70   pinMode(Reset_Pin, OUTPUT);
     digitalWrite(Reset_Pin, HIGH);
72
     // Bypass Taster initialisieren
74   pinMode(Bypass_Pin, INPUT_PULLUP);

76   // Backlight Taster initialisieren
     pinMode(Backlight_Pin, INPUT_PULLUP);
78
     // Pinrichtung für Drehimpulsgeber 1 festlegen
80   pinMode(Key_1_Pin_A, INPUT_PULLUP);
     pinMode(Key_1_Pin_B, INPUT_PULLUP);
82   pinMode(Key_1_Pin_D, INPUT_PULLUP);

84   // Pinrichtung für Drehimpulsgeber 2 festlegen
     pinMode(Key_2_Pin_A, INPUT_PULLUP);
86   pinMode(Key_2_Pin_B, INPUT_PULLUP);
     pinMode(Key_2_Pin_D, INPUT_PULLUP);
88
     // Pinrichtung für Drehimpulsgeber 3 festlegen
90   pinMode(Key_3_Pin_A, INPUT_PULLUP);
     pinMode(Key_3_Pin_B, INPUT_PULLUP);
92   pinMode(Key_3_Pin_D, INPUT_PULLUP);

94   // Pinrichtung für Drehimpulsgeber 4 festlegen
     pinMode(Key_4_Pin_A, INPUT_PULLUP);
96   pinMode(Key_4_Pin_B, INPUT_PULLUP);
     pinMode(Key_4_Pin_D, INPUT_PULLUP);
98
     // Pinrichtung für Drehimpulsgeber 5 festlegen
100  pinMode(Key_5_Pin_A, INPUT_PULLUP);
     pinMode(Key_5_Pin_B, INPUT_PULLUP);
102  pinMode(Key_5_Pin_D, INPUT_PULLUP);

104  // States initialisieren
     Key_1_Pin_A_oldState = digitalRead(Key_1_Pin_A);
106  Key_2_Pin_A_oldState = digitalRead(Key_2_Pin_A);
     Key_3_Pin_A_oldState = digitalRead(Key_3_Pin_A);
108  Key_4_Pin_A_oldState = digitalRead(Key_4_Pin_A);
     Key_5_Pin_A_oldState = digitalRead(Key_5_Pin_A);
110  Key_1_Pin_A_State = Key_1_Pin_A_oldState;
     Key_2_Pin_A_State = Key_2_Pin_A_oldState;
112  Key_3_Pin_A_State = Key_3_Pin_A_oldState;
     Key_4_Pin_A_State = Key_4_Pin_A_oldState;
114  Key_5_Pin_A_State = Key_5_Pin_A_oldState;
     Key_1_Pin_D_State = HIGH;
116  Key_2_Pin_D_State = HIGH;
     Key_3_Pin_D_State = HIGH;
118  Key_4_Pin_D_State = HIGH;
     Key_5_Pin_D_State = HIGH;
120  Bypass_State = bypass_off;
     Bypass_State_Change = false;
122  state_change = false;
   }
```

Ab Zeile 103 ist zu sehen wie die Statusvariablen zu den Tastern und Inkrementalgebern intiailisiert werden. Diese Statusvariablen werden von den ISRs entsprechend einer Eingabe über Taster und Inkretementalgeber gesetzt. Die ISRs sind bestimmten GPIOs zugeordnet, welche dies sind zeigt folgende Auflistung.

Listing 8: cuip.ino

```
144   attachInterrupt(Key_1_Pin_A,    Key_1_Pin_A_ISR,    CHANGE);
      attachInterrupt(Key_1_Pin_D,    Key_1_Pin_D_ISR,    CHANGE);
146   attachInterrupt(Key_2_Pin_A,    Key_2_Pin_A_ISR,    CHANGE);
      attachInterrupt(Key_2_Pin_D,    Key_2_Pin_D_ISR,    CHANGE);
148   attachInterrupt(Key_3_Pin_A,    Key_3_Pin_A_ISR,    CHANGE);
      attachInterrupt(Key_3_Pin_D,    Key_3_Pin_D_ISR,    CHANGE);
150   attachInterrupt(Key_4_Pin_A,    Key_4_Pin_A_ISR,    CHANGE);
      attachInterrupt(Key_4_Pin_D,    Key_4_Pin_D_ISR,    CHANGE);
152   attachInterrupt(Key_5_Pin_A,    Key_5_Pin_A_ISR,    CHANGE);
      attachInterrupt(Key_5_Pin_D,    Key_5_Pin_D_ISR,    CHANGE);
154   attachInterrupt(Bypass_Pin,     Bypass_Pin_ISR,     FALLING);
      attachInterrupt(Backlight_Pin,  Backlight_Pin_ISR,  FALLING);
```

Zu sehen ist, dass bei den Inkrementalgebern nur auf Änderungen am Pin A reagiert wird. Der Zustand des Pin B wird von den entsprechenden Methoden abgerufen und mit dem Pin A verglichen, wie es im Abschnitt 2.2.1 beschrieben wurde. Beim Pin A der Inkrementalgeber wird die ISR sowohl bei fallender als auch bei steigender Flanke aufgerufen. Dies wird mit dem Parameter *CHANGE* gekennzeichnet. Bei den Tastern wird auf eine fallende Flanke reagiert. Das heißt, dass bereits beim herunterdrücken des Tasters die entsprechende Funktion ausgeführt wird und dass die ISR beim loslassen des Tasters nicht noch einmal aufgerufen wird. Dem aufmerksamen Leser dürfte aufgefallen sein, dass ein Taster fehlt. Es ist der Taster mit der Funktion *Reset*. Dieser ist direkt mit dem Reset-Pin am Arduino Due verbunden. So kann in der Auflistung der Pin-Zuweisungen erkannt werden, dass der GPIO für die Funktion *Reset* als Output-Pin initialisiert wurde. Dieser Pin ist ebenfalls mit dem Reset-Pin des Arduino Due verbunden und wird aktiv, wenn im Menü der Menüpunkt Reset ausgewählt wird. Bei der Initialisierung ist der GPIO auf High-Pegel zu setzen, da dass Reset-Signal Low-aktiv ist.

Wird nun einer der Inkrementalgeber oder Taster betätigt, wird die entsprechende Statusvariable mit dem alten Status verglichen. Dies erfolgt über die Methode *checkStates*, welche immer wieder aus der Endlosschleife des Hauptprogramms aufgerufen wird, wie dies im Abschnitt 5.2.8 noch beschrieben wird. Um eine schnelle Prüfung zu ermöglichen, ob ein Status sich geändert hat, setzen die ISRs das Attribut *state_change* auf TRUE und die Methode *checkStates* prüft zuerst dieses Attribut. Ist das Attribut auf TRUE gesetzt so wird das Attribut auf FALSE zurückgesetzt und die Statusvariablen geprüft. Bei den Tastern wird auf Low-Pegel und bei den Inkrementalgebern auf einen Unterschied zwischen aktuellem Status und Vorgängerstatus geprüft.

Ergibt sich ein Treffer bei der Prüfung wird die entsprechende Methode aufgerufen. Welche dies ist, kann der folgenden Auflistung der Methode *checkStates* entnommen werden.

Listing 9: cuip_gpio.cpp

```
/*
  Mit der Methode checkStates kann eine externe Methode
  das Prüfen der Status der GPIOs veranlassen. Wird hier
  ein Unterschied erkannt, so wird die entsprechende Methode
  aufgerufen.
*/
void CUIP_GPIO::checkStates(void)
{
  if (state_change)
  {
    // Statusänderung zurücksetzen
    state_change  = false;

    // Status auswerten
    if (Key_1_Pin_D_State != HIGH) Key_1_Pin_D_Falling();
    if (Key_2_Pin_D_State != HIGH) Key_2_Pin_D_Falling();
    if (Key_3_Pin_D_State != HIGH) Key_3_Pin_D_Falling();
    if (Key_4_Pin_D_State != HIGH) Key_4_Pin_D_Falling();
    if (Key_5_Pin_D_State != HIGH) Key_5_Pin_D_Falling();
    if (Key_1_Pin_A_State != Key_1_Pin_A_oldState) Key_1_Pin_A_Change();
    if (Key_2_Pin_A_State != Key_2_Pin_A_oldState) Key_2_Pin_A_Change();
    if (Key_3_Pin_A_State != Key_3_Pin_A_oldState) Key_3_Pin_A_Change();
    if (Key_4_Pin_A_State != Key_4_Pin_A_oldState) Key_4_Pin_A_Change();
    if (Key_5_Pin_A_State != Key_5_Pin_A_oldState) Key_5_Pin_A_Change();
    if (Bypass_State_Change) set_bypass();
  }
}
```

Eine weitere Aufgabe der Klasse CUIP_GPIO ist die Verwaltung des Menüs der UI. Die Menüstruktur der UI kann der Tabelle 12 im Anhang G.1 entnommen werden. Für das Menü sind diverse Attribute deklariert. Diese sollen im Folgenden näher erläutert werden.

Der Wert des Attributs *menu_level* steht für die Ebene in der man sich im Menü befindet. Das Menü hat 2 Ebenen. So kann das Attribut *menu_level* folgende Werte annehmen:

-1 Menü nicht aufgerufen (menu_off)
0 Hauptmenü aufgerufen
1 Untermenü aufgerufen

Welche Menüpunkte in den jeweiligen Ebenen ausgewählt sind, ist im Attribut *menu_item* hinterlegt. Das Attribut ist ein Datenfeld mit zwei Feldern (menu_item[2];). Das erste Feld steht für die Ebene 0 (Hauptmenü) und das zweite Feld für die Ebene 1 (Untermenü). Wurde beispielsweise im Untermenü *Highpass* der Menüpunkt *HP Cuttoff freq.* ausgewählt, so hat das Attribut *menu_item* die Werte $\{1,3\}$.

Im Attribut *menu_max_items* ist die maximale Anzahl der Menüpunkte der Untermenüs hinterlegt. Definiert ist es wie folgt:

Listing 10: cuip_gpio.cpp

```
42  // Maximale Anzahl Untermenüs der jeeiligen Ebenen
    menu_max_items = {
44    menu_lowpass_items      - 1,  // Lowpass
      menu_highpass_items     - 1,  // Highpass
46    menu_limiter_items      - 1,  // Limiter
      menu_audio_items        - 1,  // Audio
48    menu_audiocodec_items   - 1,  // Audio-Codec
      menu_network_items      - 1,  // Network
50    menu_options_items      - 1,  // Options
      menu_exit_items         - 1   // Exit
52  };
```

Die Konstanten für die maximale Anzahl an Unterpunkten zu den Untermenüs sind wie folgt definiert:

Listing 11: cuip_gpio.h

```
66  // Anzahl der Menüpunkte in einem Untermenü
    #define menu_lowpass_items      5  // Lowpass
68  #define menu_highpass_items     5  // Highpass
    #define menu_limiter_items      4  // Limiter
70  #define menu_audio_items        4  // Audio
    #define menu_audiocodec_items   2  // Audio-Codec
72  #define menu_network_items      2  // Network
    #define menu_options_items      5  // Options
74  #define menu_exit_items         1  // Exit
```

Das Menü und die Untermenüs werden durch betätigen des Tasters des Inkrementalgebers F1 aufgerufen. Soll ein Untermenü verlassen werden und in das Hauptmenü zurückgesprungen werden so erfolgt dies ebenfalls durch das Betätigen des Tasters F1. Wird der Taster F1 betätigt, wird die Methode *Key_1_Pin_D_Falling* aufgerufen, dessen Struktogramm Bild 63 zeigt.

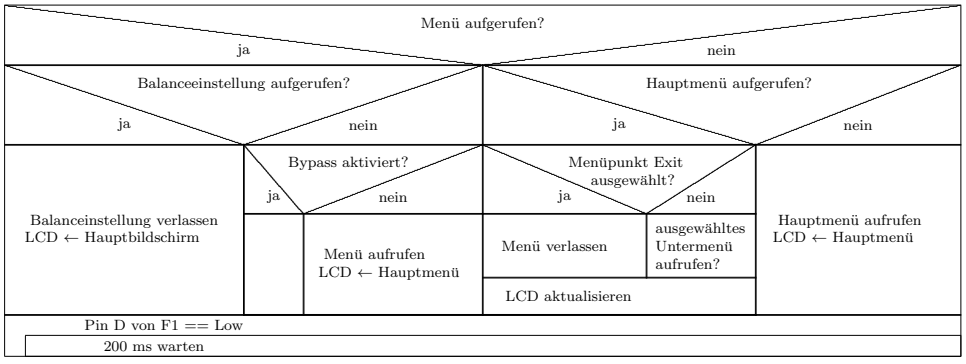


Bild 63: Struktogramm der Methode Key_1_Pin_D_Falling

Die Methode prüft zunächst ob das Menü aufgerufen ist. Ist dies nicht der Fall, so wird weiter geprüft, ob die Balanceeinstellung aufgerufen ist. Ist dies der Fall, so wird der Hauptbildschirm aufgerufen. Ist die Balanceeinstellung nicht aufgerufen, prüft die Methode weiter ob der Bypass aktiviert ist. Ist dies der Fall, so erfolgt keine Maßnahme, da im Bypass-Modus nicht vorgesehen ist, dass Änderungen an der Konfiguration vorgenommen werden. Ist der Bypass nicht aktiviert, wird das Hauptmenü aufgerufen und die Methode beendet.

Wurde bei der ersten Prüfung festgestellt, dass das Menü bereits aufgerufen ist, so wird nun geprüft, ob das Hauptmenü oder ein Untermenü aufgerufen ist. Ist das Hauptmenü aufgerufen, so wird das ausgewählte Untermenü aufgerufen oder wenn der Menüpunkt *Exit* ausgewählt ist, wird das Menü verlassen, der Hauptbildschirm aufgerufen und die Methode beendet. Ist aber ein Untermenü aufgerufen, so wird ins Hauptmenü zurückgesprungen und die Methode beendet.

Beim Beenden der Methode wird geprüft, ob der Anwender den Taster losgelassen hat. Diese Prüfung wird alle 200 ms wiederholt. Hat der Anwender den Taster losgelassen, wird die Methode endgültig beendet.

Soll im Menü ein Untermenü ausgewählt werden oder in einem Untermenü ein Menüpunkt, so ist dazu der Inkrementalgeber F1 zu verwenden. Durch drehen des Inkrementalgebers F1 im Uhrzeigersinn wird das nächste Untemenü oder der nächste Unterpunkt ausgewählt. Dreht man den Inkrementalgeber F1 entgegen dem Uhrzeigersinn, so wird das vorherige Untermenü oder der vorherige Unterpunkt ausgewählt. Beim Drehen des Inkrementalgebers F1 wird die Methode *Key_1_Pin_A_Change* aufgerufen. Die Methode prüft in einer switch-case-Anweisung ob das Menü aufgerufen wurde und ob man sich im Hauptmenü oder einem Untermenü befindet. Dem Folgenden Struktogramm kann entnommen werden, welche Methode bei welchem Status aufgerufen wird.

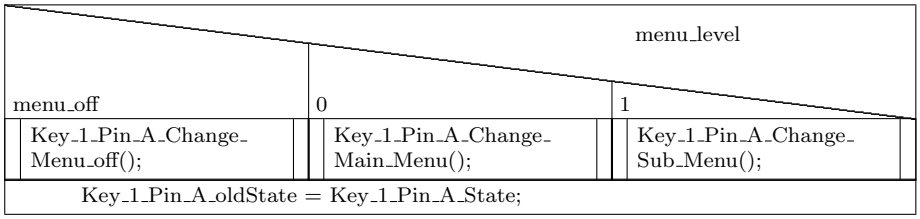


Bild 64: Struktogramm der Methode Key_1_Pin_A_Change

Wurde das Menü nicht aufgerufen, so wird die Methode *Key_1_Pin_A_Change_Menu_off* aufgerufen. Diese prüft zunächst ob der Bypaass aktiviert ist. Ist dies der Fall, so wird die Eingabe ignoriert, da während der Bypass aktiviert ist keine Änderungen an der Konfiguration des Klangreglers vorgenommen werden dürfen. Ist der Bypass deaktiviert, wird die Drehrichtung des Inkrementalgebers F1 ermittelt. Wurde im Uhrzeigersinn gedreht wird der Balancewert inkrementiert, was zur Folge hat, dass das Lautstärkeverhältnis zwischen linkem und rechtem Kanal des Stereosignals nach rechts verschoben wird. Wurde jedoch entgegen des Uhrzeigersinns gedreht, dann verschiebt sich das Lautstärkeverhältnis nach links. Anschließend wird die Anzeige im LCD auf die Balanceeinstellung eingestellt. War die Balanceeinstellung bereits aufgerufen, so wird die Anzeige im LCD nur aktualisiert. Im Folgenden ist das Struktogramm der Methode *Key_1_Pin_A_Change_Menu_off* zu sehen.

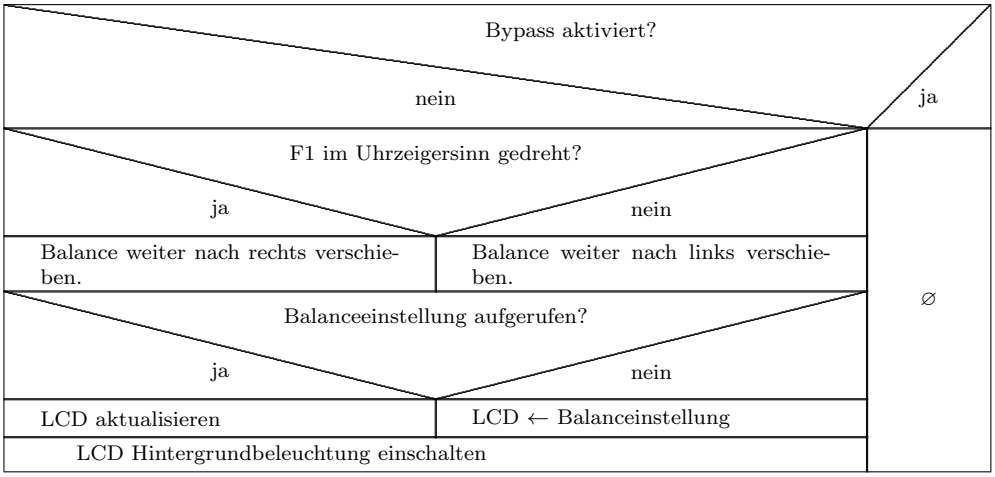


Bild 65: Struktogramm der Methode Key_1_Pin_A_Change_Menu_off

Wird die Methode *Key_1_Pin_A_Change_Main_Menu* aufgerufen befindet man sich im Hauptmenü und durch das Drehen des Inkrementalgebers F1 wird je nach Drehrichtung der nächste oder vorherige Menüpunkt des Hauptmenüs ausgewählt. Anschließend wird das LCD aktualisiert.

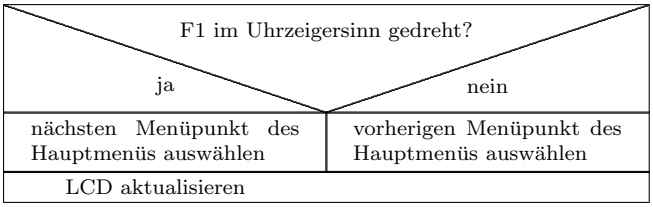


Bild 66: Struktogramm der Methode Key_1_Pin_A_Change_Main_Menu

Wenn die Methode *Key_1_Pin_A_Change_Sub_Menu* aufgerufen wird, befindet man sich in einem Untermenü und wählt den nächsten oder vorherigen Menüpunkt. Ob der vorherige oder nächste Menüpunkt ausgewählt wird, hängt von der Drehrichtung ab. Nach der Auswahl wird das LCD aktualisiert.

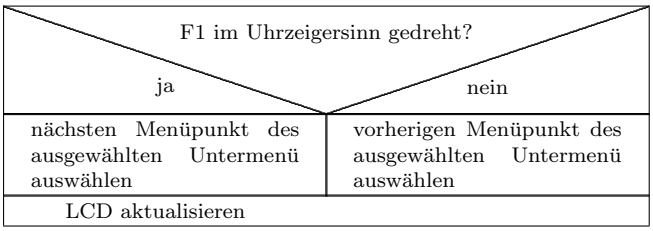


Bild 67: Struktogramm der Methode Key_1_Pin_A_Change_Sub_Menu

In dieser Art weist die Klasse CUIP_GPIO weitere Methoden für die Inkrementalgeber F2 bis F5 auf. Diese sind in ähnlicher Weise aufgebaut. Über den Zustand der UI entscheiden die Methoden welche Funktionen auszuführen sind. Die jeweiligen Funktionen der Inkrementalgeber in Bezug auf den Zustand der UI kann der Tabelle 13 im Anhang G.1 entnommen werden.

Im Zuge der Verwaltung des Menüs ist die Klasse CUIP_GPIO auch zuständig für die Zusammenstellung der Bildschirminhalte, welche auf dem LCD ausgegeben werden. Dazu wird die Methode *lcd_menu_change* aufgerufen. Die folgende Auflistung zeigt den Quellcode zur Methode *lcd_menu_change*.

Listing 12: cuip_gpio.cpp

```
1402 /*
       Mit der Methode lcd_menu_change wird die Anzeige des LCDs
1404   entsprechend des Zustandes des User Interfaces eingestellt.
     */
1406 void CUIP_GPIO::lcd_menu_change(void)
     {
1408   String line1;  // obere Zeile des LCD
       String line2;  // untere Zeile des LCD
1410   float  ans;

1412   // Je na Menu Level und ausgewähltem Menüpunkt Anzeige auf dem LCD einstellen
       switch(menu_level)
1414   {
         case menu_off : (*lcd).print_main((*dsp).get_config()); break;
1416
         case 0        : switch(menu_item[0])
1418                     {
                           case menu_lowpass     : line1 = ">> Lowpass";     line2 = "   Highpass";     break;
1420                       case menu_highpass    : line1 = ">> Highpass";    line2 = "   Limiter";      break;
                           case menu_limiter     : line1 = ">> Limiter";     line2 = "   Audio";        break;
1422                       case menu_audio       : line1 = ">> Audio";       line2 = "   Audio-Codec";  break;
                           case menu_audiocodec : line1 = ">> Audio-Codec"; line2 = "   Network";      break;
1424                       case menu_network     : line1 = ">> Network";     line2 = "   Options";      break;
                           case menu_options     : line1 = ">> Options";     line2 = "   Exit";         break;
1426                       case menu_exit        : line1 = ">> Exit";        line2 = "                "; break;
```

```
                    }
1428                    break;
      case 1        : lcd_menu_change_level_1(&line1, &line2);
1430                    break;
    }
1432
    // Menu aufgerufen?
1434 if (menu_level != menu_off)
    {
1436   // Wenn sich am Menüpunkt bzw. der ausgewählten Option nichts geändert hat,
      // dann nur den geänderten Wert der Option auf dem LCD überschreiben.
1438   if ((menu_level_pre != menu_level) || (menu_item[0] != menu_item_pre[0]) || (menu_item[1] != menu_item_pre[1])) (*lcd).
        println(0,line1);
      (*lcd).println(1,line2);
1440   if ((menu_item[0] == menu_network) && (menu_item[1] == menu_network_ip) && ((*srv).getMode() == network_mode_static))
      {
1442     (*lcd).cursor();
        (*lcd).setCursor(ip_cursor,1);
1444   } else {
        (*lcd).noCursor();
1446   }

1448 }
    // aktuelle Werte als Vorgängerwerte übernehmen
1450 menu_item_pre[0] = menu_item[0];
    menu_item_pre[1] = menu_item[1];
1452 menu_level_pre   = menu_level;
}
```

Zunächst werden zwei Zeichenkettenvariablen angelegt, welche die obere und untere Zeile des LCD repräsentieren. Anschließend wird über eine Switch-Case-Anweisung geprüft, welche Menüebene aufgerufen ist. Ist das Menü nicht aufgerufen, so wird das Ausgeben des Hauptbildschirmes veranlasst (siehe Zeile 1415). Ist das Hauptmenü aufgerufen, so wird über eine weitere Switch-Case-Anweisung das ausgewählte Untermenü ermittelt und den beiden Zeichenkettenvariablen *line1* und *line2* die entsprechenden Untermenübezeichnungen zugewiesen (siehe Zeilen 1417 bis 1428). Ist jedoch ein Untermenü ausgewählt so wird die Methode *ld_menu_change_level_1* aufgerufen. Diese prüft, welches Untermenü aufgerufen ist und ruft die entsprechende Methode zum Untermenü auf. Welche Methode für welches Untermenü zuständig ist kann der Tabelle 3 entnommen werden.

Untermenü	Methode
Lowpass	lcd_menu_change_lowpass
Highpass	lcd_menu_change_highpass
Limiter	lcd_menu_change_limiter
Audio	lcd_menu_change_audio
Audio-Codec	lcd_menu_change_audiocodec

Untermenü	Methode
Network	lcd_menu_change_network
Options	lcd_menu_change_option

Tabelle 3: Übersicht der Untermenüs und der zugehörigen Methoden

In diesen Methoden wird weiter geprüft, welcher Menüpunkt ausgewählt ist und der Bildschirminhalt zu diesem Menüpunkt zusammengestellt und in den Zeichenkettenvariablen *line1* und *line2*, welche über Call-by-Reference zu übergeben sind, hinterlegt.

Nach der Switch-Case-Anweisung der Methode *lcd_menu_change* geht es in der Methode mit der Abfrage, ob das Menü aufgerufen ist, weiter. Das hat den Hintergrund, wenn das Menü nicht aufgerufen ist, in der Switch-Case-Anweisung eine eigene Methode aufgerufen wird, welche den Bildschirminhalt des LCDs eigenständig verwaltet bzw. ändert. Diese Methode ist *print_main* der Klasse CUIP_LCD, welche im Abschnitt 5.2.3 noch näher beschrieben wird. Diese Methode ist für die Anzeige des Hauptbildschirmes und der Balanceeinstellung zuständig. Wurde festgestellt, dass das Menü aufgerufen ist, wird das LCD aktualisiert (siehe Zeilen 1433 bis 1453).

Zu erwähnen sind noch die Methoden *set_lcd_main_lp*, *set_lcd_main_hp*, *set_lcd_main_lt* und *set_lcd_main_vol*. Diese Methoden werden aufgerufen, wenn die entsprechenden Inkrementalgeber gedreht werden und man sich im Hauptbildschirm befindet. Diese Methoden ändern nur den entsprechenden Bereich des LCDs. Wurde z. B. der Inkrementalgeber F4 im Uhrzeigersinn gedreht, so wurde der Verstärkungsfaktor für den Hochpass vergrößert. Es wird die Methode *set_lcd_main_hp* aufgerufen die nur den Bereich im LCD ändert, welcher den Wert des Verstärkungsfaktors zeigt.

Für den Taster S3 (Bypass) ist die Methode *set_bypass* zuständig, dessen Quellcode im Folgenden zu sehen ist.

Listing 13: cuip_gpio.cpp

```
1792 /*
       Mit der Methode set_bypass wird der Tiefpass und der Hochpass abgeschaltet,
1794   wenn einer oder beide Filter noch nicht abgeschaltet sind. Sind beide Filter
       bereits abgeschaltet, dann werden diese wieder zugeschaltet. Somit wird ein
```

```
1796   Bypass-Effekt realisiert. Dieser wird von der Methode checkState aufgerufen,
       wenn eine steigende Flanke des Bypass-Tasters detektiert wurde.
1798 */
     void CUIP_GPIO::set_bypass(void)
1800 {
       int t = 1;
1802   int f = 0;

1804   if (Bypass_State == bypass_on)
       {
1806     (*dsp).set(&(*dsp).keys.s_DAC_en, &t);
       (*dsp).set(&(*dsp).keys.s_b,      &f);
1808   Bypass_State = bypass_off;
       } else {
1810   (*dsp).set(&(*dsp).keys.s_b,      &t);
       (*dsp).set(&(*dsp).keys.s_DAC_en, &f);
1812   Bypass_State = bypass_on;
       }
1814   menu_level = menu_off;
       menu_item = {0, 0};
1816   isBalance == false;
       Bypass_State_Change = false;
1818   lcd_menu_change();
       while(digitalRead(Bypass_Pin) == LOW) { delay(200); }
1820 }
```

Je nachdem ob der Bypass aktiviert oder deaktiviert ist, werden zunächst die Parameterwerte entsprechende geändert (Zeilen 1804 bis 1813). Anschließend wird der Hauptbildschirm aufgerufen und über die Methode *lcd_menu_change* veranlasst, dass statt den Werten, die normalerweise im Hauptbildschirm zu sehen sind, der Schriftzug „Bypass“ in der 2. Zeile des LCD erscheint.

5.2.3. Die Klasse CUIP_LCD

Die Klasse CUIP_LCD ist für die Ausgabe der Bildschirminhalte auf das LCD zuständig. Dazu wird auf die Bibliothek *LiquidCrystal_I2C*[14] in der Version 2.0 zurückgegriffen. Die Klasse *LiquidCrystal_I2C* ist die Elternklasse der Klasse *CUIP_LCD* und übernimmt die Kommunikation zwischen MCU und DU über I^2C. Die Tabelle 4 gibt eine Übersicht über die Methoden der Klasse und deren Aufgabe.

[14] http://hmario.home.xs4all.nl/arduino/LiquidCrystal_I2C/

Methode	Beschreibung
clearline(unit8_t row)	Löscht die Zeile *row* des LCD durch überschreiben mit Leerzeichen (0 für Zeile 1 und 1 für Zeile 2).
println(unit8_t row, String s)	Überschreibt die Zeile *row* mit der Zeichenkette *s*. Sind in *s* mehr Zeichen angegeben als Spalten vorhanden werden diese ignoriert.
println_center(uint_t row, String s)	Diese Methode hat die gleiche Funktion wie die Methode *println*. Es wird allerdings der Text in *s* mittig angezeigt.
print_cur(uint8_t row, uint8_t col, String s)	Schreibt den Text in *s* ab der Postion (*col*,*row*).
print_main	Lässt den Hauptbildschirm auf dem LCD anzeigen.
lcd_time_inc	Erhöht die Abschaltzeit der Hintergrundbeleuchtung des LCD um eine Sekunde.
lcd_time_dec	Verringert die Abschaltzeit der Hintergrundbeleuchtung des LCD um eine Sekunde.
get_interval	Gibt die Abschaltzeit in Millisekunden der aufrufenden Methode zurück.
set_interval(int *value)	Kann zum festlegen der Abschaltzeit der Hintergrundbeleuchtung des LCD verwendet werden. Mit *value* ist die Abschaltzeit in Millisekunden zu übergeben.
get_backlight_state	Gibt den Status zurück, ob die Hintergrundbeleuchtung des LCD ein- oder ausgeschaltet ist.
backlight_on	Schaltet die Hintergrundbelechtung des LCD ein und startet die Abschaltverzögerung.

Methode	Beschreibung
backlight_off	Schaltet die Hintergrundbeleuchtung des LCD ab und beendet die Abschaltverzögerung.
check_backlight_off	Prüft ob der Abschaltzeitpunkt, zum Abschalten der Hintergrundbeleuchtung des LCD, erreicht ist. Ist dies der Fall, so wird die Hintergrundbelechtung abgeschaltet und die Abschaltverzögerung beendet.
print_error(int errno)	Wird aufgerufen, um eine Fehlermeldung zum Fehler *errno* auf dem LCD auszugeben.

Tabelle 4: Übersicht über die Methoden der Klasse CUIP_LCD

5.2.4. Die Klasse CUIP_SD

Die Klasse CUIP_SD ist für die Lade- und Speichervorgänge auf der micro-SD card zuständig. Dazu stellt sie Methoden bereit, deren Funktion der Tabelle 5 entnommen werden kann.

Methode	Beschreibung
byte get_storage()	Gibt den ausgewählten Speicherort (0 bis 9) zurück.
set_storage(byte new_storage)	Stellt die Auswahl des Speicherorts auf *new_storage* ein.
load(CUIP_DSP *dsp, CUIP_LCD *lcd, CUI_ETHERNET *srv, boolean show_ip)	Lädt von der micro-SD card die Konfiguration vom ausgewählten Speicherort und stellt die Konfiguration ein.
save(dsp_config *cfg, CUIP_LCD *lcd, CUIP_ETHERNET *srv)	Speichert die aktuelle Konfiguration auf der micro-SD card am ausgewählten Speicherort.

Tabelle 5: Übersicht über die Methoden der Klasse CUIP_SD

Die Speicherung der Konfiguration der Klangregelung erfolgt über die Methode *save* und das Laden über die Methode *load*. Beide kommunizieren mit dem micro-SD card reader, wofür die Bilbiothek *SD* eingebunden und auf deren Funktionen zurückgegriffen wird. Gespeichert wird direkt im Hauptverzeichnis der micro-SD card. Die Dateien haben als Dateinamen den Speicherort und als Erweiterung tc (für Tone Control). Es stehen die Speicherorte 0,1,2 ... 9 zur Verfügung. Der Speicherort 0 nimmt eine besondere Rolle ein. Ist eine micro-SD Card vorhanden und kann eine Konfiguration geladen werden, dient dieser Speicherplatz als der mit der Standardkonfiguration, welche beim Einschalten oder Rücksetzen des Klangreglers geladen wird. Daher wird dieser Speicherplatz in den Anzeigen auch mit *default* bezeichnet.

Die Speicherung erfolgt binär. Alle Parameterwerte liegen als reine Integer-Zahlen vor. Beim Arduino Due sind Integer 32-bit breit. Da die Speicherung über den Befehl *write* erfolgt und mit dem nur ein Byte geschrieben werden kann, erfolgt das Speichern in der Form, dass die 32-Bits in ihre Bytes aufgespalten werden. Dies wird durch Rechtsschieben, wie man dem Ausschnitt aus der Methode *save* für das Speichern des Wertes *s_lp_fc* (Grenzfrequenz $f_{c_{lp}}$) entnehmen kann.

Listing 14: cuip_sd.cpp

```
    f.write((*cfg).s_lp_fc >> 24);
276 f.write((*cfg).s_lp_fc >> 16);
    f.write((*cfg).s_lp_fc >> 8);
278 f.write((*cfg).s_lp_fc);
```

In umgekehrter Weise erfolgt das Auslesen der Werte in der Methode *load*. Dazu nutzt diese die Funktion *read* aus der Bibliothek *SD*. Die Funktion kann nur ein Byte auslesen. Die folgende Auflistung zeigt den Ausschnitt aus der Methode *load* zum Auslesen des Wertes *s_lp_fc*. Es ist zu sehen, dass die einzelnen Bytes nach dem jeweiligen Linksschieben durch eine ODER-Verknüpfung miteinander verbunden werden.

Listing 15: cuip_sd.cpp

```
120     cfg.s_lp_fc   = f.read();
        cfg.s_lp_fc   = (cfg.s_lp_fc << 8)   | f.read();
122     cfg.s_lp_fc   = (cfg.s_lp_fc << 8)   | f.read();
        cfg.s_lp_fc   = (cfg.s_lp_fc << 8)   | f.read();
```

Nimmt man an, dass die Grenzfrequenz $f_{c_{hp}}$ für den Höhen-Shelving-Filter mit 8000 Hz festgelegt wurde, dann findet man in der jeweiligen Datei *.tc den Wert in Hexadezimaler Form wie folgt vor:

1. Byte	2. Byte	3. Byte	4. Byte
00	00	1F	40

Das vollständige Dateiformat zur Speicherung der Konfiguration des Klangreglers kann der Tabelle 14 im Anhang G.1 entnommen werden.

5.2.5. Die Klasse CUIP_Ethernet

Die Klasse CUIP_Ethernet hat die Aufgabe die Kommunikation über Ethernet zu initialisieren. Dazu stellt es Methoden bereit, welche je eingestellten Netzwerkmodus eine Verbindung mit dem DHCP herstellen oder eine statische IP Adresse einrichten. Weiter wird die UDP-Verbindung initialisiert und die Ports eingestellt. Dazu stehen verschiedene Methoden zur Verfügung. Es gibt diverse get-Methoden mit denen die IP Adresse, die Netzwerkmaske, das Gateway usw. abgefragt werden können. Genauso gibt es mehrere set-Methoden mit denen bestimmte Werte gesetzt werden können.

Die wichtigste Methode ist die Methode *init*. Diese initialisiert das Arduino Ethernet Shield für die Kommunikation mit einer Ethernet-Verbindung. Dazu wird zunächst der Netzwerkmodus ermittelt. Es stehen folgende Netzwerkmodi zur Verfügung, welche der Anwender wählen kann:

Netzwerkmodus	Beschreibung
off	Netzwerkverbindung wird nicht verwendet.
dhcp	Netzwerkverbindungsdaten werden über einen DHCP-Server ermittelt.
static	Netzwerkverbindungsdaten werden manuell durch den Anwender vorgegeben.

Tabelle 6: Übersicht der Netzwerkmodi

Im Modus *off* wird die Methode direkt wieder beendet. Ist der Modus *dhcp* ausgewählt wird eine Verbindung mit dem DHCP-Server aufgebaut und eine IP-Adresse eingestellt. Diese Kommunikation findet über Methoden der Elternklasse *EthernetServer*[15] statt. Ist der Modus static eingestellt, so wird die IP-Adresse die in den Konfigurationsdaten des Klangreglers hinterlegt ist eingestellt. Über das Untermenü *Network* kann diese abgefragt und geändert werden.

[15] http://arduino.cc/en/Reference/Ethernet

5.2.6. Die Klasse CUIP_UDP

Die Klasse CUIP_UDP repräsentiert mit ihren Methoden den UDP-Server. Als UDP-Ports kommen für das Senden 56789 und für das Empfangen 56788 zur Anwendung. Diese Ports sind in der Version, der dieser Arbeit zugrunde liegt, feststehend. Es ist aber denkbar, die Ports während des Betriebes änderbar zu machen. Für die Kommunikation über UDP wird auf die Methoden der Bibliothek *EthernetUDP* zurückgegriffen.

Mit der Methdoe *init* wird der UDP-Server intialisiert und für den Empfang auf dem Port 56789 eingerichtet. Mit der Methode *checkUDP* wird geprüft, ob ein Request vorliegt. Ist dies der Fall, so wird der Request abgerufen und ausgewertet. Dazu hat der Client, der den Request gesendet hat, eine Kennung an den Anfang des Request zu stellen, welcher aus der Zeile *HTRUI#1.0.0* zu bestehen hat, als Kennung, dass der Klangregler gemeint ist. Es wäre denkbar das auch andere Server auf dem Port 56789 hören. Mal abgesehen davon, dass das TCGUIP zunächst einen Broadcast sendet, um die IP-Adresse des Klangreglers zu ermitteln, wie es noch im Abschnitt 5.3 näher beschrieben wird. Als Trennung zwischen der Kennung und dem Inhalt des Request dient ein Fragezeichen. An das Fragezeichen ist der Request anzuhängen. Die möglichen Requests können der Tabelle 15 im Anhang G.1 entnommen werden. Bei den Requests, welche einen Parameter ändern sollen, hat dem Request-Bezeichner der Wert zu folgen. Als Trenner dient hier das Gleichheitszeichen. Möchte man beispielsweise den Verstärkungsfaktor des Tiefen-Shelving-Filters auf 6 dB ändern, so ist die Zeichenkette *HTRUI#1.0.0?s_lp_G=6* als Inhalt des Requests zu senden. Die Methode *checkUDP* zerlegt diese Zeichenkette in seine Bestandteile und wertet diese aus. Bei den Abfragen zur IP-Adresse und der Gesamtkonfiguration werden diese als Response dem Client zurückgegeben. Bei den Requests bei denen Parameterwerte geändert werden sollen, werden diese entsprechend geändert und dem Client als Response auf den Request zurückgegeben.

Die Response ist ähnlich aufgebaut, wie der Request. Zunächst wird wieder die Kennung *HTRUI#1.0.0* der eigentlichen Nachricht vorangestellt. Der Trenner ist das Fragezeichen. Anschließend ist der Parameterbezeichner anzuhängen gefolgt vom Gleichheitszeichen als Trennzeichen und dem Wert des Parameters. Bei der Abfrage der Gesamtkonfiguration erfolgt die Trennung der einzelnen Parameter durch das kaufmännische Und.

5.2.7. Die Klasse CUIP_WI

Die Klasse CUIP_WI stellt den HTTP-Server bereit. Über die Klasse *CUIP_Ethernet* wurde zum Abhören bereits der Port 80 eingerichtet. Alle Requests die über den Port 80 an die CU gerichtet werden, werden an die Methoden der Klasse CUIP_WI weitergeleitet.

Mit der Methode *checkRequest* wird geprüft, ob ein Request von einem Client anliegt. Ist dies der Fall, wird der HTTP-Header in seine Bestandteile zerlegt. Die Tabelle 7 zeigt welche Aktionen bei welchem Zusatz zur Adresse ausgeführt werden.

Adresszusatz	Beschreibung
set_param	Parameterwert ändern
get_param	Wert eines einzelnen Parameters abrufen
get_cfg	gesamte Konfiguration abrufen
beliebiger Zusatz	index.html (Startseite des Webinterfaces) abrufen

Tabelle 7: Übersicht über die Adresszusätze beim Request über das WI

Beim Ändern eines Parameterwertes ist dem Adresszusatz der Bezeichner und der Wert anzuhängen. Soll beispielsweise der Wert für die Attack time des Limiters auf 120 μs gesetzt werden, so muss das Webinterface einen Request mit der Adresse *http://xxx.xxx.xxx.xxx/set_param?s_ta=120* senden. Die Stellen mit x sind durch die IP-Adresse des Klangreglers oder einer Domain, welche über einen DNS auf die IP-Adresse zum Klangregler zeigt, zu ersetzen.

Die Methode *checkRequest* prüft, nach dem Zerlegen des HTTP-Headers in seine Bestandteile, ob einer der Adresszusätze vorliegt und ruft die Methode *response* auf. Diese sendet eine entsprechende Response an den Client zurück. Sollte ein Parameterwert geändert werden, so ruft die Methode *response* die Methode *set* der Klasse CUIP_DSP auf und übergibt den Wert. Der Wert wird validiert und der Parameter entsprechend geändert. Zur Bestätigung wird als Response der neu eingestellte Parameterwert dem Client zurückgegeben.

In der Tabelle 7 findet sich der Adresszusatz *get_param* mit dem ein einzelner Wert abgerufen werden kann. Diese Funktion ist zwar schon vorbereitet, aber noch nicht umgesetzt und kann daher noch nicht angewendet werden.

Wurde der Request mit dem Adresszusatz *get_cfg* gesendet, so wird als Response die gesamte Konfiguration an den Client zurückgegeben. Dies erfolgt in der Form:

Parameterbezeichner1=Parameterwert1;Parameterbezeichner2=Parameterwert2; ...

Wie man sieht, werden die Parameter durch Semikolon getrennt und der Parameterbezeichner und der Parameterwert durch ein Gleichheitszeichen getrennt.

Wurde ein Anderer als die Adresszusätze *set_param*, *get_param* und *get_cfg* angegeben oder gar keiner, wird immer die Startseite des Webinterfaces dem Client als Response zurückgegeben. Die Startseite ist auf der micro-SD card im Verzeichnis *wi* als *index.html* hinterlegt. Soll die Startseite als Response gesendet werden, so wird die Methode *index_response* aufgerufen. Diese lädt von der miceo-SD card zeilenweise die index.html und sendet die Startseite zeilenweise an den Cleint.

5.2.8. Der Programmablauf

Im Hauptprogramm eines Arduino-Sketches sind zwei Funktionen zu implementieren. Zum einen ist das die Funktion *setup* und zum anderen die Funktion *loop*. Die Funktion *setup* ist für die Initialisierung der Arduino-Komponenten vorgesehen und die Funktion *loop* ist eine Endlosschleife. Arduino beschreibt die Funktion *setup* wie folgt:

„The setup() function is called when a sketch starts. Use it to initialize variables, pin modes, start using libraries, etc. The setup function will only run once, after each powerup or reset of the Arduino board.“[16]

[16]http://arduino.cc/en/Reference/Setup

Zur Funktion *loop* schreibt Arduino:

„After creating a setup() function, which initializes and sets the initial values, the loop() function does precisely what its name suggests, and loops consecutively, allowing your program to change and respond. Use it to actively control the Arduino board.“[17]

Im Folgenden ist die Auflistung des Quellcodes zur Funktion *setup* zu sehen.

Listing 16: cuip.ino

```
108 void setup()
{
110   // LCD initialisieren
  lcd.init();
112   lcd.println_center(0,"Tone Control");
  lcd.println_center(1,"Version 1.0.0");
114   delay(500);  // Wartezeit, zur Sicherstellung, dass der DSP gestartet ist.

116   // SD Karte initialisieren
  if (show) lcd.println_center(1,"initialize SD");
118   if (sd.init())
  {
120     sd.set_storage(0);
    sd.load(&dsp, &lcd, &srv, false);
122     if (show) delay(500);
  } else {
124     srv.setConfig(&srv_cfg, false);
    lcd.print_error(200);
126   }

128   // DSP initialisieren
  if (show)
130   {
    lcd.println_center(0,"Tone Control");
132     lcd.println_center(1,"initialize DSP");
  }
134   dsp.init();
  if (show) delay(1000);
136
  // GPIOs initialisieren
138   if (show) lcd.println(1,"initialize GPIOs");
  gpio.init();
140   if (show) delay(1000);

142   // Interrupts initialisieren
  if (show) lcd.println_center(1,"initialize IRQs");
144   attachInterrupt(Key_1_Pin_A,    Key_1_Pin_A_ISR,    CHANGE);
  attachInterrupt(Key_1_Pin_D,    Key_1_Pin_D_ISR,    CHANGE);
146   attachInterrupt(Key_2_Pin_A,    Key_2_Pin_A_ISR,    CHANGE);
  attachInterrupt(Key_2_Pin_D,    Key_2_Pin_D_ISR,    CHANGE);
148   attachInterrupt(Key_3_Pin_A,    Key_3_Pin_A_ISR,    CHANGE);
  attachInterrupt(Key_3_Pin_D,    Key_3_Pin_D_ISR,    CHANGE);
150   attachInterrupt(Key_4_Pin_A,    Key_4_Pin_A_ISR,    CHANGE);
  attachInterrupt(Key_4_Pin_D,    Key_4_Pin_D_ISR,    CHANGE);
152   attachInterrupt(Key_5_Pin_A,    Key_5_Pin_A_ISR,    CHANGE);
  attachInterrupt(Key_5_Pin_D,    Key_5_Pin_D_ISR,    CHANGE);
154   attachInterrupt(Bypass_Pin,     Bypass_Pin_ISR,     FALLING);
  attachInterrupt(Backlight_Pin, Backlight_Pin_ISR,  FALLING);
156   if (show) delay(1000);

158   // Hauptbildschirm anzeigen
  lcd.print_main(dsp.get_config());
160 }
```

[17] http://arduino.cc/en/Reference/Loop

In der Funktion *setup* werden alle Methoden der zuvor vorgestellten Klassen aufgerufen, welche die entsprechenden Objekte initialisieren. So wird zunächst die Methode *init* der Klasse CUIP_LCD aufgerufen. Diese Initialisiert das LCD gemäß der dem Konstruktor übergebenen Parameter. Diese Parameter sind die I^2C-Adresse, die Anzahl Zeilen und Spalten des LCDs. Diese Parameter gibt der Konstruktor direkt an die Elternklasse *LiquidCrystal_I2C* weiter. Die Parameter werden in Form der Struktur *lcd_config* übergeben.

Der nächste Punkt in der Funktion *setup* ist die Initialisierung des micro-SD card readers, um anschließend die Konfiguration des Klangreglers auslesen zu können. Zur Initialisierung wird die Methode *init* der Klasse CUIP_SD aufgerufen. Diese initialisiert die Chip Select Signale. Diese liegen auf den GPIOs 4 und 10. Die GPIOs sind als Ausgänge zu initialisieren. Bei der Kommunkation zwischen Arduino Due und dem micro-SD card reader bzw. dem W5100 werden diese entsprechend gesetzt. Weiter wird die Kommunikation mit der Funktion *begin* gestartet. Der Funktion wird als Parameter der Chip Select Pin übergeben. In diesem Fall ist es für den micro-SD card reader die 4. Konnte eine Kommunikation mit dem micro-SD card reader nicht hergestellt werden, wird eine Fehlermeldung ausgegeben, aber die Initialisierung nicht abgebrochen, da der Klangregler auch ohne eine Verbindung zum micro-SD card reader funktionsfähig ist.

Der nächste Schritt ist die Initialisierung der AU. Dies erfolgt über die Methode *init* der Klasse CUIP_DSP. Diese veranlasst über die Funktion *init* der Bibliothek *Wire* die Initialisierung der I^2C-Schnittstellen. Anschließend ruft die Methode die Methode *init_dsp* auf. Mit dieser Methode wird aus der Konfiguration der Klangregelung die Konfiguration der AU extrahiert und diese an die AU übertragen.

Zum Schluss der Funktion *setup* werden den Interrupts die ISRs zugewiesen und der Hauptbildschirm aufgerufen und auf dem LCD ausgegeben. Mit dem Ende der Funktion *setup* wird die Funktion *loop* aufgerufen.

Die folgende Auflistung zeigt den Quellcode der Funktion *loop*.

Listing 17: cuip.ino

```
170 void loop()
    {
172   // Prüfen ob einer der Taster oder Inkrementalgeber gedrückt bzw. gedreht
      // wurde, also entsprechende Interrupts ausgelöst wurden.
174   if (gpio.state_change) gpio.checkStates();

176   // Prüfen ob die Zeit für das Leuchten der Hintergrundbeleuchtung
      // abgelaufen ist. Wenn ja, dann Hintergrundbeleuchtung des LCD abschalten.
178   if(lcd.check_backlight_off()) gpio.backlight_off();
```

```
180   // Prüfen ob eine Anfrage an den Server vorhanden ist.
      // Wenn ja, dann Server anweisen darauf zu reagieren.
182   if (srv.getMode() != network_mode_off)
      {
184     EthernetClient client = srv.available();
        if (client) wi.checkRequest(&client);
186     udp.checkUDP();
      }
188 }
```

Bei jedem Schleifendurchgang werden die Methoden *checkStates*, *backlight_off* aufgerufen. Danach wird geprüft, ob der Netzwerkmodus nicht *off* ist. Ist dies der Fall, wird zusätzlich zu den zuvor genannten Methoden geprüft, ob ein Request für den HTTP-Server oder UDP-Server vorliegt. Liegt ein Request für den HTTP-Server vor, wird die Methode *checkRequest* aufgerufen und liegt eine Request für den UDP-Server vor, wird die Methode *checkUDP* aufgerufen.

5.3. Tone Control Graphical User Interface Program (TCGUIP)

Das TCGUIP ist ein Programm, welches auf einem PC unter den Betriebssystemen Linux und Windows ausgeführt werden kann. Es wurde in der Programmiersprache C++ objektorientiert programmiert. Genutzt wurde das Framework Qt in der Version 5 und die IDE *QtCreator* in der Version 2.7.1. Mithilfe dessen wurden die Elemente der Benutzeroberfläche, welche im Bild 68 zu sehen ist, mit dem graphischen Editor positioniert. Diese über den graphischen Editor erstellte Benutzeroberfläche ist in der Datei *mainwindow.ui* hinterlegt.

Bild 68: Benutzeroberfläche des TCGUIP

Die Benutzeroberfläche ist in Gruppen (Lowpass, Highpass usw.) unterteilt und die Elemente nach den Funktionen logisch angeordnet. Im oberen Bereich sind die drei Diagramme zum Amplitudengang, Phasengang und der Gruppenlaufzeit der Filter angeordnet. Darunter folgen die gruppierten Eingabe- und Anzeigeelemente.

Die Benutzerschnittstelle besteht aus fünf Klassen. Die Klasse *mainwindow* ist für die Darstellung der Eingabe- und Anzeigeelemente zuständig. Werden Eingabeelemente betätigt, so werden entsprechende Events ausgelöst, welche wiederum Methoden der Klasse aufrufen. Nach dem Betätigen eines Eingabeelementes werden der jeweilige Parameter entsprechend geändert und per UDP an den Klangregler übermittelt. Dieser stellt den Parameter entsprechend ein und gibt ihn an die AU weiter. Als Response sendet der Klangregler den neu eingestellten Wert zurück und die Methoden der Klasse *mainwindow* aktualisieren die Anzeigeelemente.

Für das Berechnen der Koeffizienten und Übertragungsfunktionen der Filter sowie das Zeichnen der Graphen ist die Klasse *TCGUIP_GRAPH* zuständig. Um die Übertragungsfunktionen berechnen zu können, muss die Klasse in der Lage sein komplexe Zahlen zu verarbeiten. Dazu wird auf die Methoden der Klasse *QComplex* zurückgegriffen. Die Klasse *QComplex* repräsentiert den Datentyp für eine komplexe Zahl. Dazu stellt es Attribute bereit, welche den Real- und Imaginärteil speichern. Weiterhin stellt es mathematische Operatoren zur Addition, Subtraktion, Multiplikation und Division von komplexen Zahlen bereit.

Mit der Klasse *TCGUIP_ADDR_FINDER* werden Methoden bereitgestellt, um über einen Broadcast-Request die IP-Adresse des Klangreglers zu ermitteln. Die Klasse ist abgeleitet von der Klasse *QThread* und arbeitet als eigenständiger Prozess. Durch den Broadcast erhalten alle am Netzwerk angeschlossen Geräte den Request. Durch die Kennung *HTRUI#1.0.0* in der Nachricht erkennt der Klangregler, dass diese für ihn gedacht ist und extrahiert die IP-Adresse des Clients und sendet eine Response mit der gleichen Kennung zurück. Womit aus der empfangenen Nachricht das TCGUIP die IP-Adresse des Klangreglers extrahieren kann und die weiteren Request an diese sendet. Dieser Vorgang wird jede Minute wiederholt, so dass bei Änderungen am Klangregler, was die Netzwerkeinstellungen betrifft, das TCGUIP selbstständig diesen wiederfindet. Somit muss der Anwender sich nicht um die Einstellungen der Netzwerkparameter kümmern. Der Klangregler und der Client müssen im gleichen Netzwerk sein. Sollte dies nicht so sein, dann muss über die Router die Weiterleitung der UDP-Ports 56789 und 56788 sichergestellt sein.

Die Klasse *TCGUIP_UDP_TRANSCEIVER* ist für die Kommunikation mit dem UDP-Server zuständig. Dazu werden zwei Objekte der Klasse *QUdpSocket* angelegt. Zum Empfangen wird

auf UDP-Port 56789 gehört. Kommt eine Nachricht vom Server wird ein Event ausgelöst, welcher eine Methode zum auswerten der Nachricht aufruft. Wird ein Parameter geändert so veranlasst die Klasse *TCGUIP_UDP_TRANCEIVER* das Zusammenstellen und Senden der Nachricht. Bei der Auswertung werden die Daten in die einzelnen Parameter zerlegt und diese in der Konfiguration des TCGUIP abgelegt und die Klasse *mainwindow* dazu veranlasst die Anzeigen und Diagramme zu aktualisieren.

5.4. Tone Control Webinterface Program (TCWIP)

Das TCWIP ist ein Programm, welches in einem Webbrowser ausgeführt wird. Es werden die Scriptsprachen HTML, CSS und Javascript angewandt. Als IDE wurde das Aptana Studio 3[18] verwendet. Das Webinterface wurde für die Browser Mozilla Firefox[19] (ab Version 25.0) und Google Chrome[20] (ab Version 26.0) geschrieben und auch mit diesen getestet. Es sei hier ausdrücklich erwähnt, dass eine Optimierung für den Microsoft Internet Explorer nicht stattgefunden hat.

Wie bereits einleitend erwähnt wurde, wurde die Programmiersprache Javascript verwendet. Dies hat den Hintergrund, dass das Webinterface auf dem Client-Rechner alle Berechnungen durchführt und so den Mikrocontroller entlastet. Dieser liefert über eine Ethernet-Verbindung lediglich die Daten bzw. den Programmcode für das Webinterface. Die Daten werden per HTTPRequest[21] abgerufen.

Das Webinterface ist auf der micro-SD card im Verzeichnis *wi* als Datei *index.html* hinterlegt. Es wird aus dem Browser heraus durch die Eingabe der IP-Adresse des Klangreglers aufgerufen. Sollte ein DNS-Server im Netz vorhanden sein, so kann dieser natürlich genutzt werden, um der IP-Adresse des Klangreglers eine bestimmte Domain zuzuweisen. Die Kommunikation findet über TCP/IP statt und auf Anwendungsebene kommt das HTTP zur Anwendung. Es wird der TCP-Port 80 verwendet.

[18] http://www.aptana.com/
[19] https://www.mozilla.org/de/firefox/new/
[20] https://www.google.com/intl/de/chrome/browser/
[21] http://www.w3schools.com/dom/dom_http.asp

Die *index.html* ist in drei Abschnitte eingeteilt. Im head-Bereich des HTML-Dokumentes finden sich die Javascript-Funktionen und die CSS-Anweisungen. Die Javascript-Funktionen dienen dem Abrufen der Daten und dem Senden von Kommandos an den Klangregler sowie der dynamischen Änderung der HTML-Elemente. Die CSS-Anweisungen teilen dem Browser mit, wie bestimmte HTML-Elemente anzuzeigen sind. Dies können Eigenschaften wie Farbe, Größe, Schriftart usw. sein. Der dritte Teil findet sich im body-Bereich des HTML-Dokumentes. Dort ist ein HTML-Grundgerüst aufgebaut, welches über die Javascript-Funktionen dynamisch angepasst wird bzw. mit Inhalt gefüllt wird. Die Aufgaben der jeweiligen Javascript-Funktionen können der Tabelle 16 im Anhang G.2 entnommen werden.

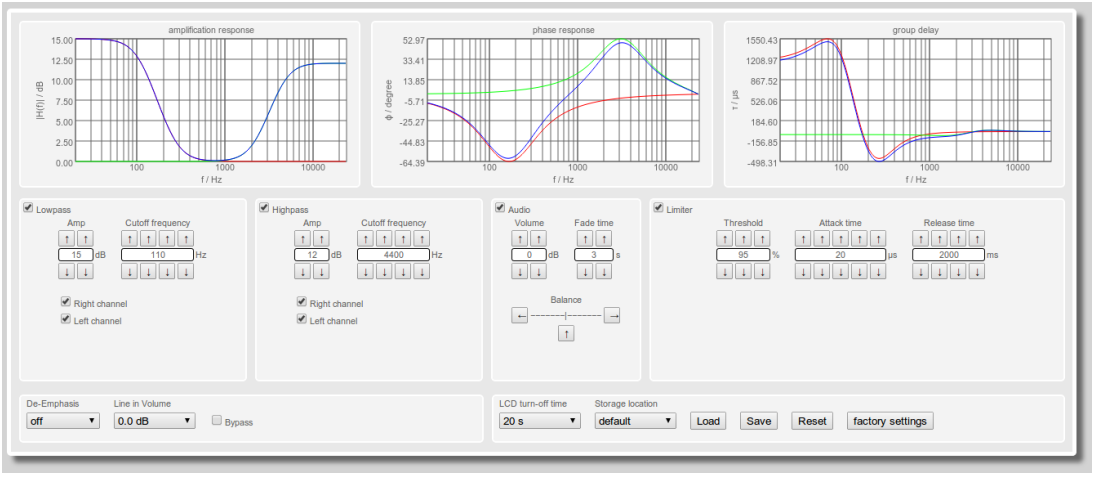


Bild 69: Benutzeroberfläche des TCWIP

Die Benutzeroberfläche, welche Bild 69 zeigt, ist nach logischen Blöcken sortiert. Die Eingabe- und Anzeigeelemente für Lowpass, Highpass, Limiter usw. sind in Gruppen angeordnet. Oberhalb der Anzeige- und Eingabeelemente sind die Diagramme für den Amplitudengang, Phasengang und der Gruppenlaufzeit zu sehen.

Wird eines der Eingabeelemente betätigt, so wird ein HTTPRequest an den Klangregler gesendet. Dieser wertet das übertragene Kommando aus und stellt den Parameter auf den übertragenen Wert ein. Anschließend sendet der HTTP-Server als Response den eingestellten Wert zurück und das Webinterface stellt die Anzeige auf den empfangenen Wert ein. So kann nach betätigen eines Eingabelementes überprüft werden, ob die Eingabe bei der CU angekommen ist. Die CU gibt die Parameteränderung an die AU weiter.

Alle 500 ms wird vom Webinterface ein Request mit dem Kommando *get_cfg* gesendet. Dieser weist den HTTP-Server an, die gesamte Konfiguration des Klangreglers als Response zurückzusenden. Empfängt das Webinterface diese Response, stellt es alle Anzeigen entsprechend der empfangenen Werte ein. Gleichzeitig werden bei allen empfangenen Parameterwerte, welche Einfluss auf den Graphen der jeweiligen Diagramme haben die Graphen neu berechnet und gezeichnet. So ist gewährleistet, dass Änderungen vom Webinterface erfasst werden, welche nicht vom Webinterface aus vorgenommen wurden. Dies könnte z.B. eine Änderung über die Eingabeelemente am Gerät sein.

Wird versucht ein Request zu senden obwohl der davor noch nicht abgeschlossen ist oder kommt nach einer Zeit von 5 Sekunden keine Response zu einem Request, wird dieser abgebrochen. Wurde ein Timeout ermittelt, wird nach dem Abbruch des Request ein Neuer mit dem Kommando *get_cfg* gesendet.

6. Die Messergebnisse

6.1. Die Filter

Bei den Filtern wurden zunächst Messungen am Tiefen- und Höhen-Shelving-Filter einzeln durchgeführt und anschließend wurde die Kaskade aus beiden Filtern gemessen. Gemessen wurde der Amplituden- und Phasengang.

Zunächst wurde der Amplitudengang des Tiefen-Shelving-Filters mit einer Grenzfrequenz $f_{c_{lp}}$ = 100 Hz und verschiedenen Verstärkungsfaktoren G_{lp} gemessen. Begonnen wurde mit dem Verstärkungsfaktor G_{lp} = -30 dB, welcher von Messung zu Messung um 5 dB erhöht wurde. Die letzte Messung erfolgte mit dem Verstärkungsfaktor G_{lp} = 30 dB. Das Ergebnis der Messung ist in Bild 70 dargestellt. Es ist gut zu sehen, dass die vorgegebene Verstärkung erreicht wird und mit der Grenzfrequenz $f_{c_{lp}}$ = 100 Hz der Übergang zu einem annähernd linearen Abfall bzw. Anstieg erfolgt und die Verstärkung bei 0 dB verbleibt.

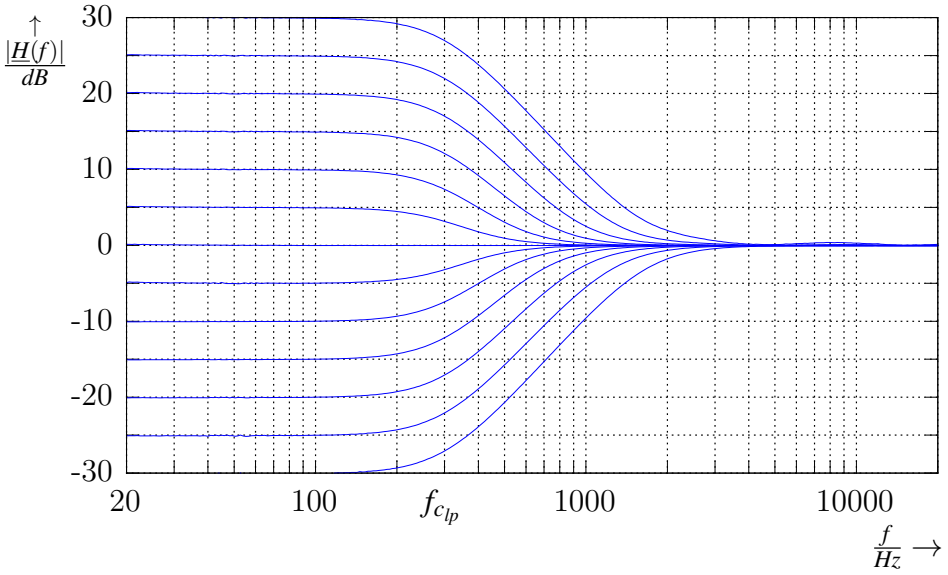
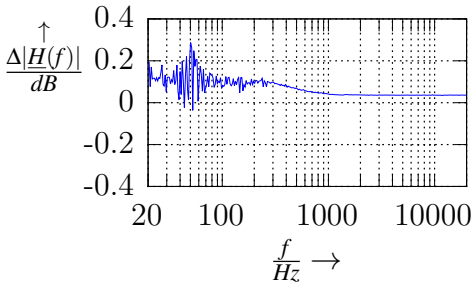
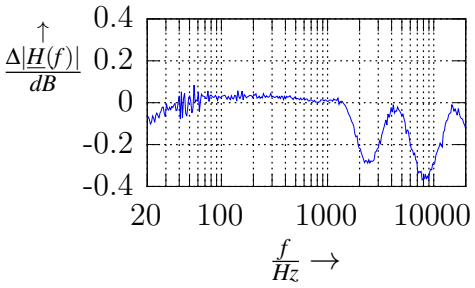


Bild 70: Amplitudengang Tiefen-Shelving-Filter ($f_{c_{lp}}$ = 300 Hz)

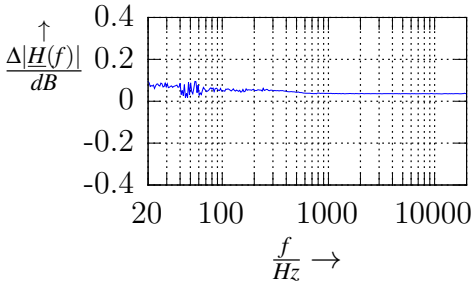
Um eine Abweichung zu den berechneten Werten erkennen zu können, wurde die Differenz des berechneten Amplitudengangs vom Gemessenen gebildet. Diese sind dem Bild 71 zu entnehmen.



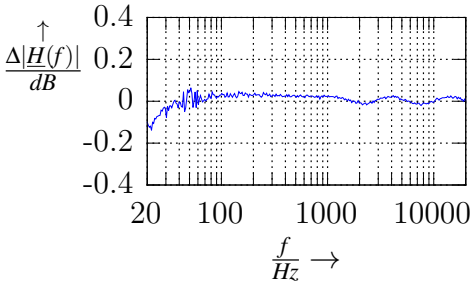
(a) G_{lp} = -30 dB



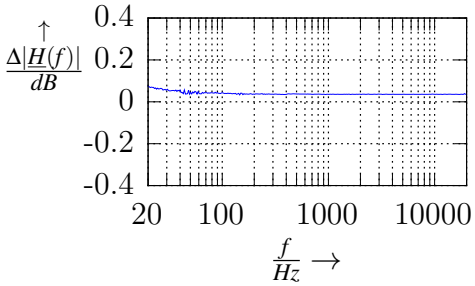
(b) G_{lp} = 30 dB



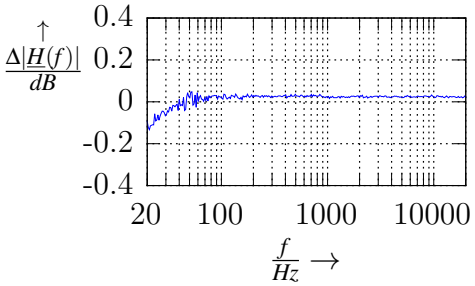
(c) G_{lp} = -20 dB



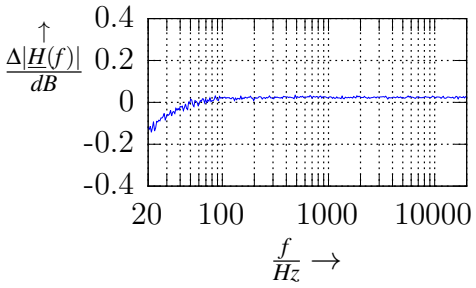
(d) G_{lp} = 20 dB



(e) G_{lp} = -10 dB



(f) G_{lp} = 10 dB



(g) G_{lp} = 0 dB

Bild 71: Differenzen der berechneten zu den gemessenen Amplitudengängen

Es ist zu sehen, dass die Abweichungen nicht sehr ausgeprägt sind. Die Abweichungen können auf Messfehler bzw. Rechenungenauigkeiten zurückgeführt werden.

Für den Höhen-Shelving-Filter wurde die gleiche Messung durchgeführt. Der Filter wurde auf die Grenzfrequenz $f_{c_{hp}}$ = 3000 Hz eingestellt. Das Ergebnis der Messung zeigt Bild 72.

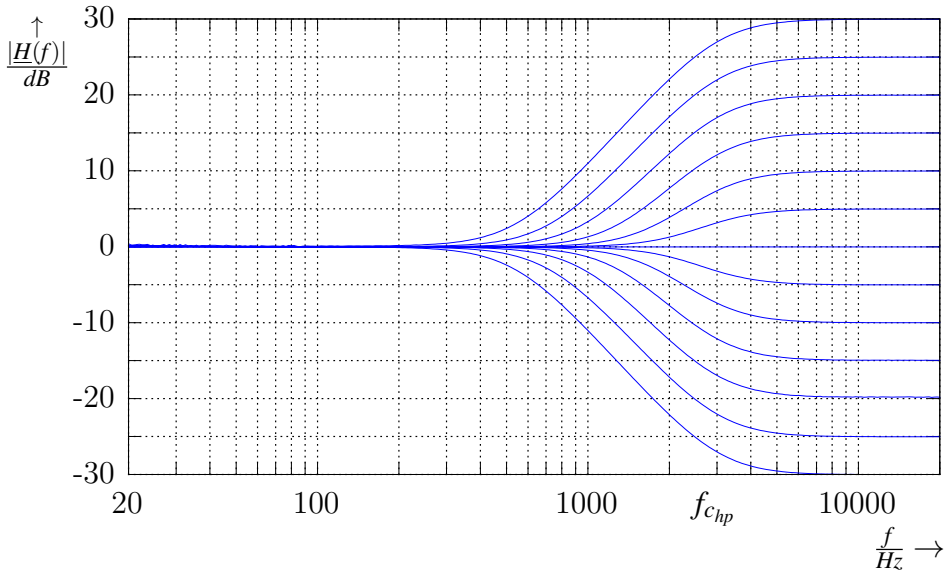
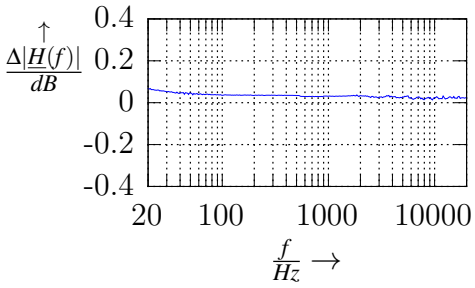
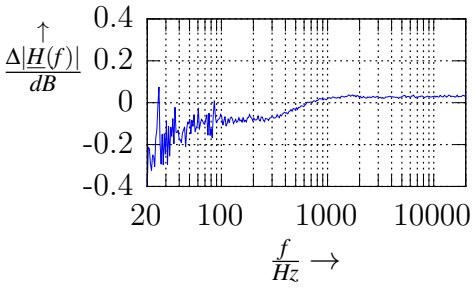


Bild 72: Amplitudengang Höhen-Shelving-Filter ($f_{c_{hp}}$ = 3000 Hz)

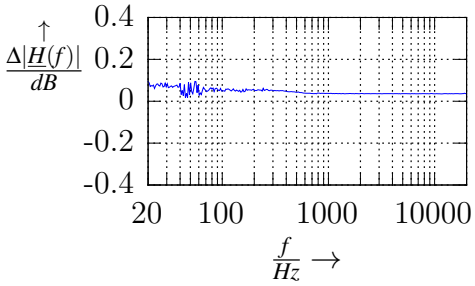
Auch beim Höhen-Shelving-Filter kann gesagt werden, dass dieser die Verstärkungsfaktoren einnimmt, auf die er eingestellt wurde. Auch die Anstiege und Abfälle sind korrekt. Dies zeigen auch die Differenzdiagramme der Differenzen zwischen den berechneten und gemessenenen Amplitudengängen, welche Bild 73 zeigt.



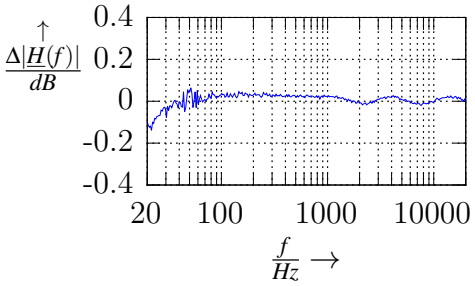
(a) G_{hp} = -30 dB



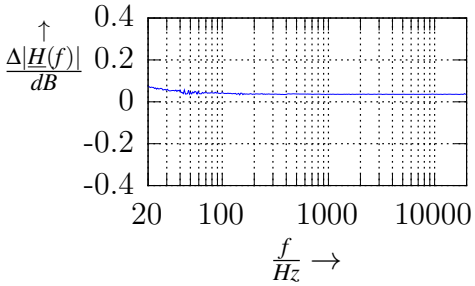
(b) G_{hp} = 30 dB



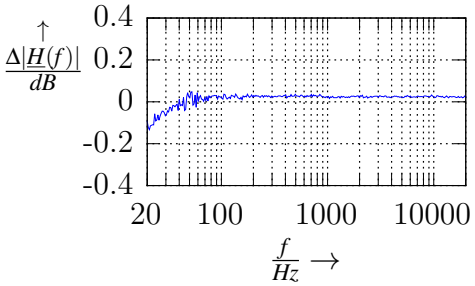
(c) G_{hp} = -20 dB



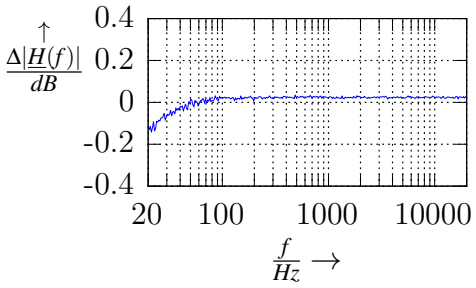
(d) G_{hp} = 20 dB



(e) G_{hp} = -10 dB



(f) G_{hp} = 10 dB



(g) G_{hp} = 0 dB

Bild 73: Differenzen der berechneten zu den gemessenen Amplitudengängen des Höhen-Shelving-Filters

Auch bei den Differenzen der berechneten zu den gemessenen Amplitudengängen des Höhen-Shelving-Filters ist zu sehen, dass die Abweichungen minimal sind und durch Messfehler bzw. Messungenauigkeiten zurückzuführen sind, aber auch auf Rechenungenauigkeiten beruhen.

Als nächstes wurden Amplitudengänge der Kaskade aus Tiefen- und Höhen-Shelving-Filter aufgenommen. Dabei wurde mit einer Verstärkung von G_{lp} = G_{hp} = -30 dB begonnen und in 5-dB-Schritten erhöht, bis zu einer Verstärkung von G_{lp} = G_{hp} = 30 dB. So wurden drei Messungen durchgeführt und die Grenzfrequenzen $f_{c_{lp}}$ und $f_{c_{hp}}$ variiert. Bei der ersten Messung lagen die Grenzfrequenzen bei $f_{c_{lp}}$ = 100 Hz und $f_{c_{hp}}$ = 6000 Hz. Das Ergebnis der Messung kann Bild 74 entnommen werden.

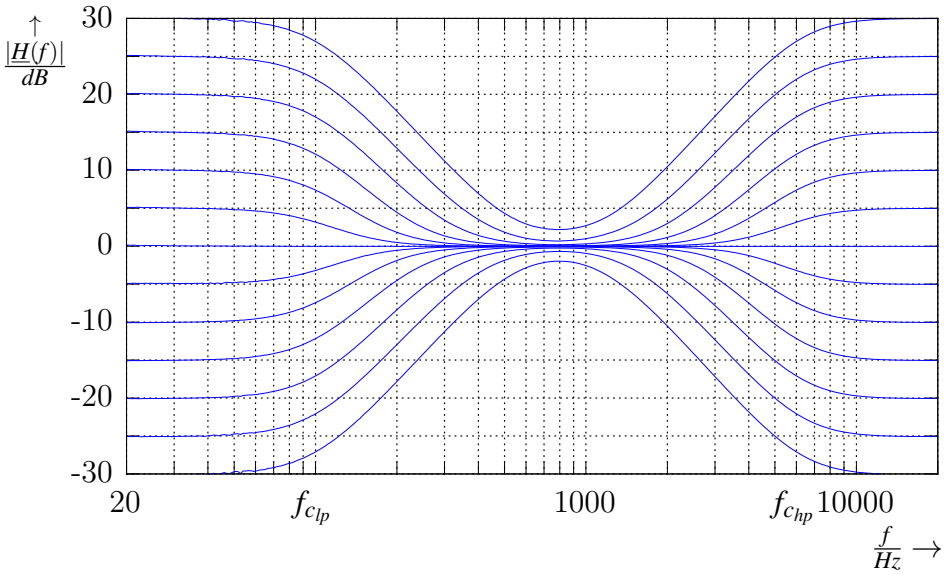


Bild 74: Amplitudengang ($f_{c_{lp}}$ = 100 Hz ; $f_{c_{hp}}$ = 6000 Hz)

Wie schon in den Einzelmessungen nimmt auch hier die Kaskade die eingestellten Verstärkungsfaktoren ein. Gut zu sehen ist auch, dass der mittlere Bereich der Frequenzen, welcher eigentlich keine Verstärkung erfahren sollte, dies aber trotzdem, wenn auch gering, erfährt. Je näher die Grenzfrequenzen zueinander liegen und je höher die Verstärkungen werden, desto größer ist der Einfluss auf die mittleren Frequenzen. Bei den nächsten beiden Messungen lagen die Grenzfrequenzen bei 300 Hz und 1000 Hz für den Tiefen-Shelving-Filter und bei 3000 Hz und 1000 Hz für den Höhen-Shelving-Filter.

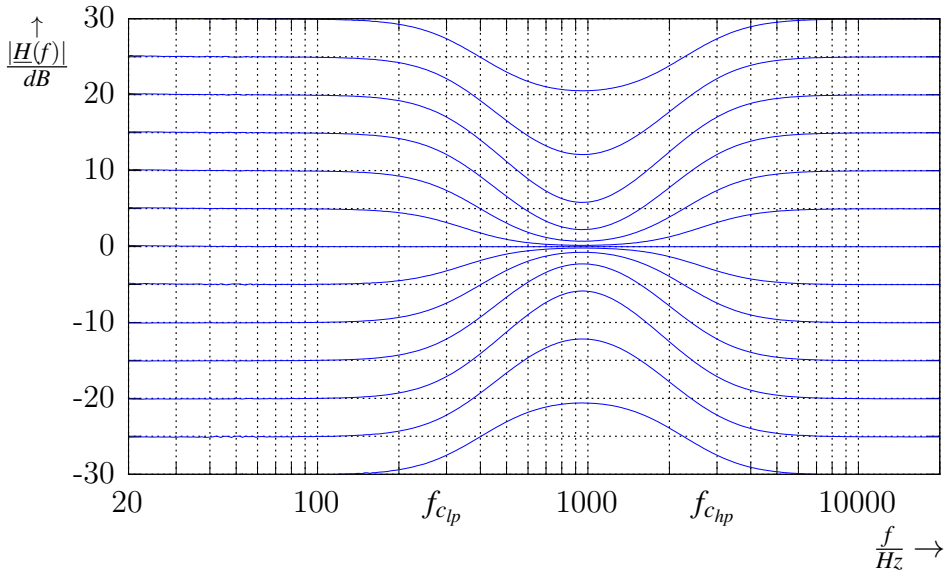


Bild 75: Amplitudengang ($f_{c_{lp}}$ = 300 Hz ; $f_{c_{hp}}$ = 3000 Hz)

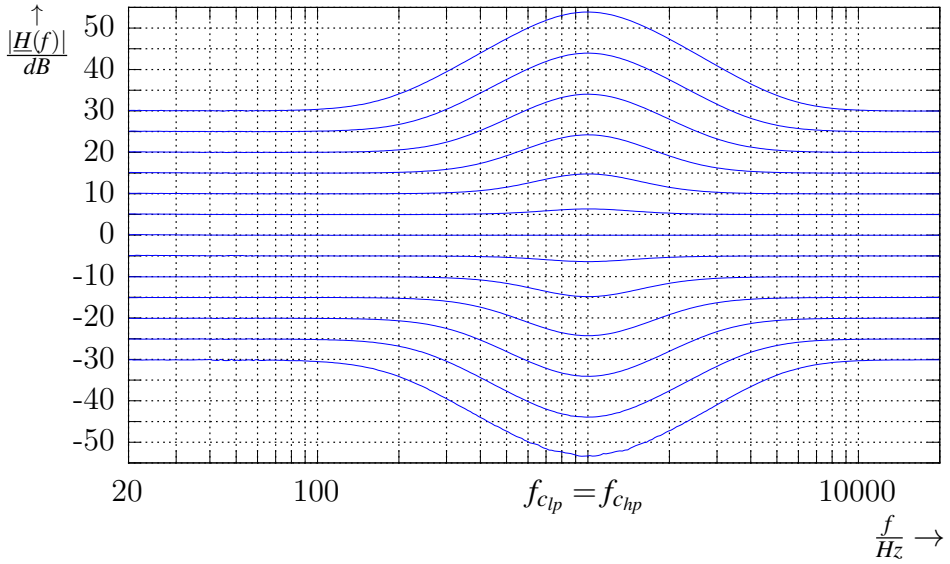


Bild 76: Amplitudengang ($f_{c_{lp}} = f_{c_{hp}}$ = 1000 Hz)

Die Bilder 75 und 76 zeigen deutlich die Problematik, wenn die Grenzfrequenzen sich weiter annähern und die Verstärkungen größer werden, dass die mittleren Frequenzen stärker beeinflusst werden. Besonders ausgeprägt ist dies, wenn wie bei der letzten Messung, die Grenzfrequenzen gleich sind ($f_{c_{lp}} = f_{c_{hp}}$).

Hier kommt es sogar für die mittleren Frequenzen durch die Addition beider Filter, zu einer größeren Anhebung bzw. Absenkung als für die tiefen bzw. hohen Frequenzen vorgesehen ist.

Bei den nächsten Messungen wurden die Phasengänge gemessen. Zunächst wurde der Phasengang des Tiefen-Shelving-Filters ermittelt. Dazu wurde die Grenzfrequenz $f_{c_{lp}}$ = 300 Hz eingestellt. Es wurde mit einer Verstärkung von G_{lp} = -30 dB begonnen und diese in 10-dB-Schritten bis auf G_{lp} = 30 dB erhöht. Bild 77 zeigt das Messergebnis.

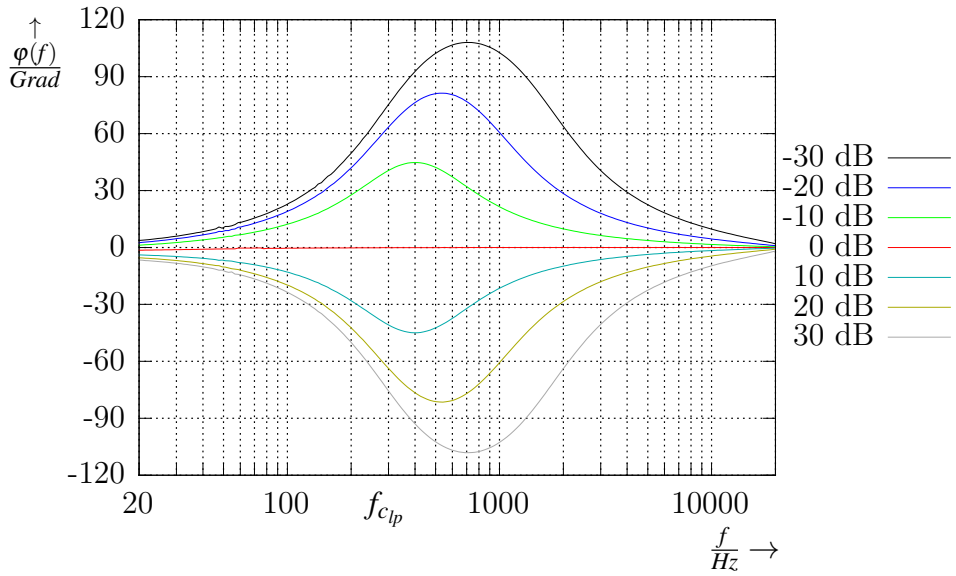


Bild 77: Phasengang Tiefen-Shelving-Filter ($f_{c_{lp}}$ = 300 Hz)

Die gleiche Messung wurde mit dem Höhen-Shelving-Filter durchgeführt. Die Grenzfrequenz wurde mit $f_{c_{hp}}$ = 3000 Hz festgelegt. Auch hier entsprach das Ergebnis, welches im Bild 78 zu sehen ist, dem Erwarteten.

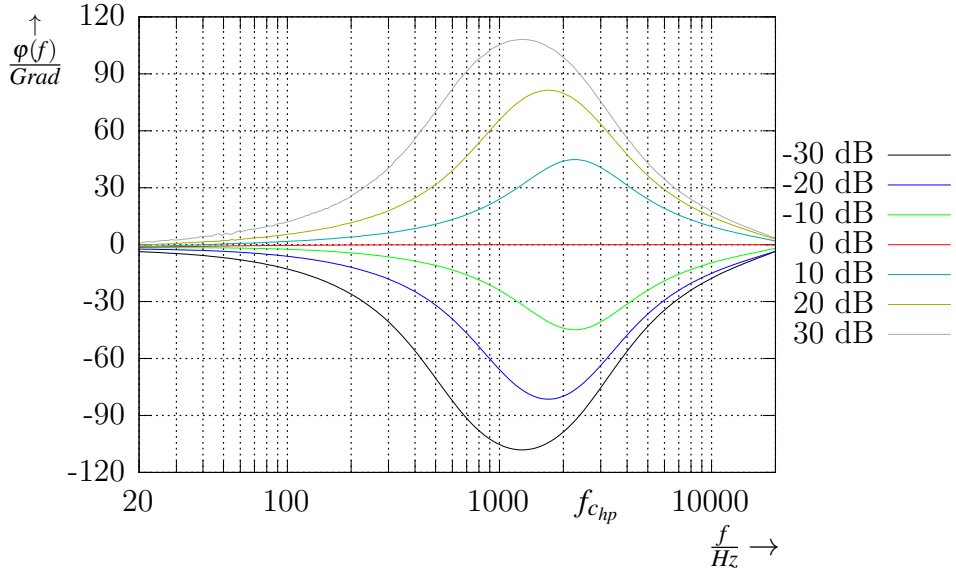


Bild 78: Phasengang Höhen-Shelving-Filter ($f_{c_{hp}}$ = 3000 Hz)

Der Phasengang der Kaskade aus Tiefen- und Höhen-Shelving-Filter wurde mit den Grenzfrequenzen $f_{c_{lp}}$ = 100 Hz und $f_{c_{hp}}$ = 6000 Hz gemessen. Die Verstärkungsfaktoren wurden wieder verändert und begonnen wurde erneut mit -30 dB. Die Verstärkungsfaktoren wurden in 10-dB-Schritten bis auf 30 dB erhöht. Das Ergebnis der Messung entsprach dem Erwarteten und ist in Bild 79 zu sehen.

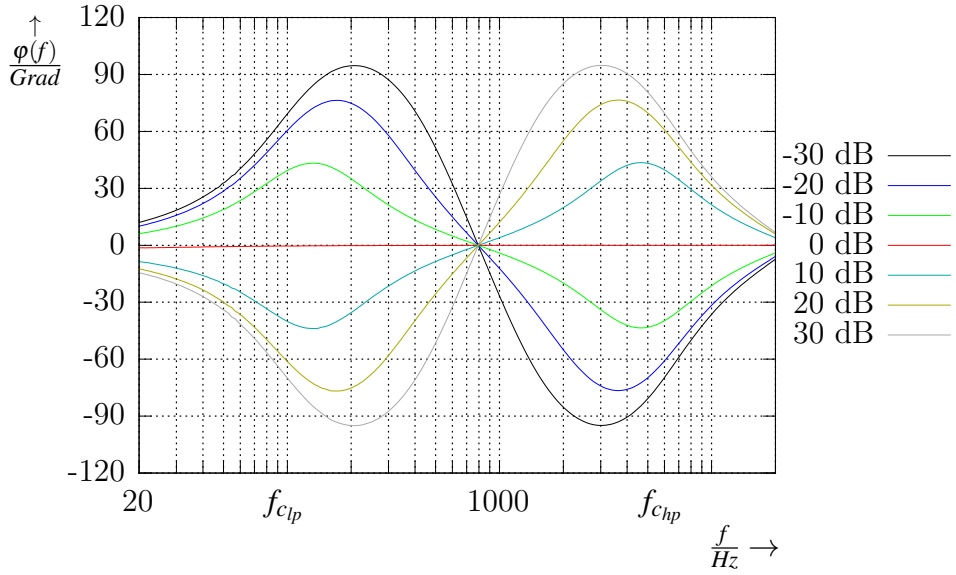


Bild 79: Phasengang Kaskade aus Tiefen- und Höhen-Shelving-Filter ($f_{c_{lp}}$ = 100 Hz; $f_{c_{hp}}$ = 6000 Hz)

Auch bei den folgenden Messungen entsprach das Ergebnis dem Erwarteten. Hier wurde wieder mit denselben Verstärkungsfaktoren gemessen. Die Grenzfrequenzen wurden auf 300 Hz und 1000 Hz für den Tiefen-Shelving-Filter und auf 3000 Hz und 1000 Hz für den Höhen-Shelving-Filter eingestellt.

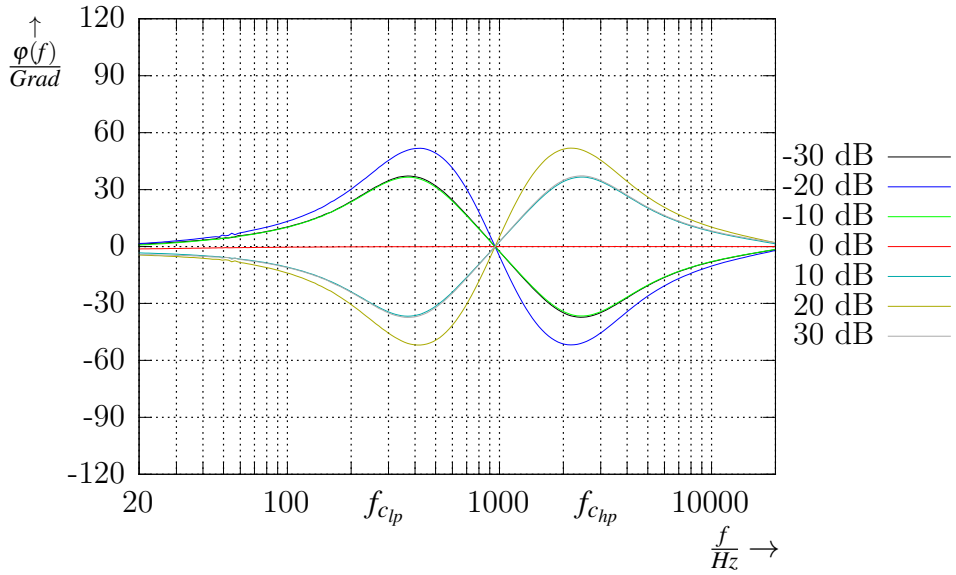


Bild 80: Phasengang Kaskade aus Tiefen- und Höhen-Shelving-Filter ($f_{c_{lp}}$ = 300 Hz; $f_{c_{hp}}$ = 3000 Hz)

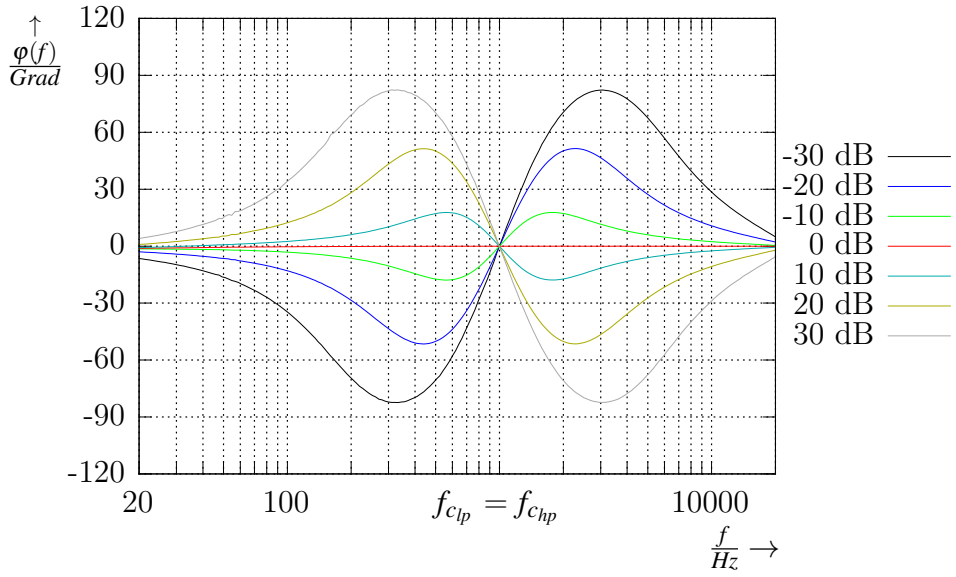


Bild 81: Phasengang Kaskade aus Tiefen- und Höhen-Shelving-Filter ($f_{c_{lp}} = f_{c_{hp}}$ = 1000 Hz)

6.2. Der Limiter

Im Bild 82 ist in der linken Hälfte des Diagramms ein Sinus zu sehen, welcher so stark verstärkt wurde, dass es zu Überläufen kam. Um die Verzerrung zu beseitigen und den Überlauf zu verhindern, wurde der Limiter zugeschaltet. Die Wirkung ist in der rechten Hälfte des Diagramms zu sehen. Somit kann ausgesagt werden, dass der Limiter erfolgreich einen Überlauf verhindert.

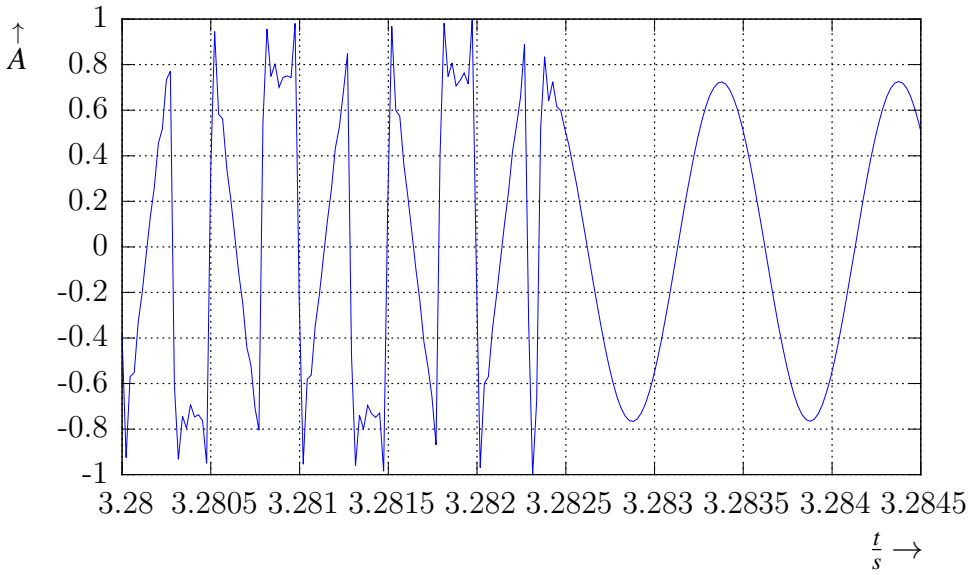


Bild 82: Begrenzung nach Zuschalten des Limiters

Bei der nächsten Messung wurde der Limiter aus dem Signalweg des linken Kanals herausgenommen und nur der rechte Kanal dem Limiter zugeführt. Das Eingangssignal wurde von 0 bis auf Maximum in der Amplitude erhöht. Die Limiterschwelle war auf 50 % eingestellt. Das Ergebnis zeigt Bild 83. Der linke Kanal ist im Bild 83 rot und der rechte Kanal blau dargestellt. Es ist zu sehen, dass der rechte Kanal der Erhöhung der Amplitude folgt. Der linke Kanal hingegen, wird bei Überschreiten von 50 % auf diese begrenzt. Somit erfüllt der Limiter auch bei dieser Messung seine Aufgabe.

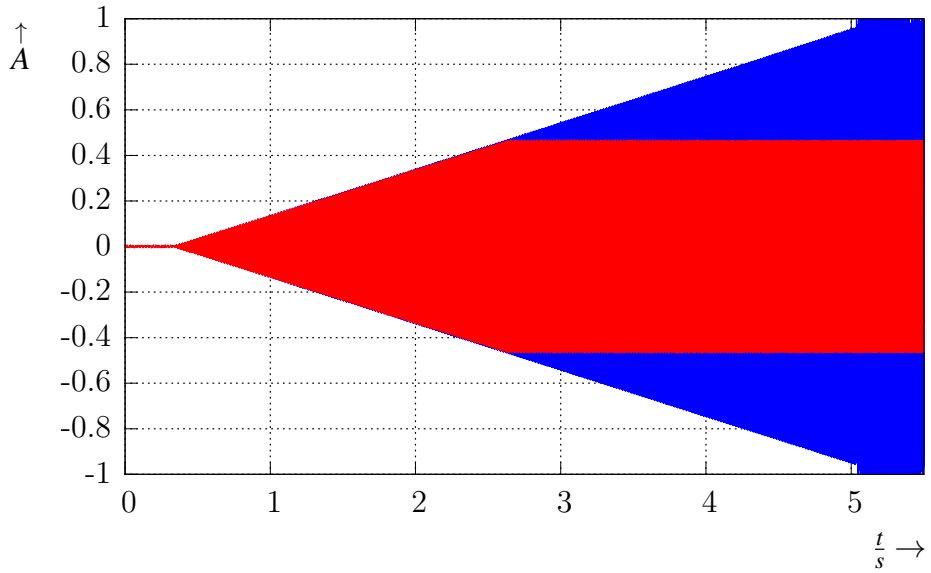


Bild 83: Begrenzung des linken Kanals auf 50 % und keine Begrenzung des rechten Kanals.

6.3. Die Balanceeinstellung

Mit der folgenden Messung wurde die Funktion der Balanceeinstellung geprüft. Dazu wurde die Balance Schrittweise bis zum Maximum nach links verschoben. Anschließend wurde die Balance wieder schrittweise auf die Mittelstellung zurückgeführt. Das Ergebnis zeigt Bild 84. Der linke Kanal ist rot und der rechte Kanal ist blau dargestellt. Es ist gut zu sehen, wie der linke Kanal in der Amplitude schrittweise angehoben und der rechte Kanal abgesenkt wird und beim zurückführen auf Mittelstellung beide wieder die gleiche Amplitude annehmen. Das Gleiche zeigt Bild 85 nur dass hier die Balance nach rechts verschoben und anschließend wieder auf Mittelstellung zurückgeführt wurde.

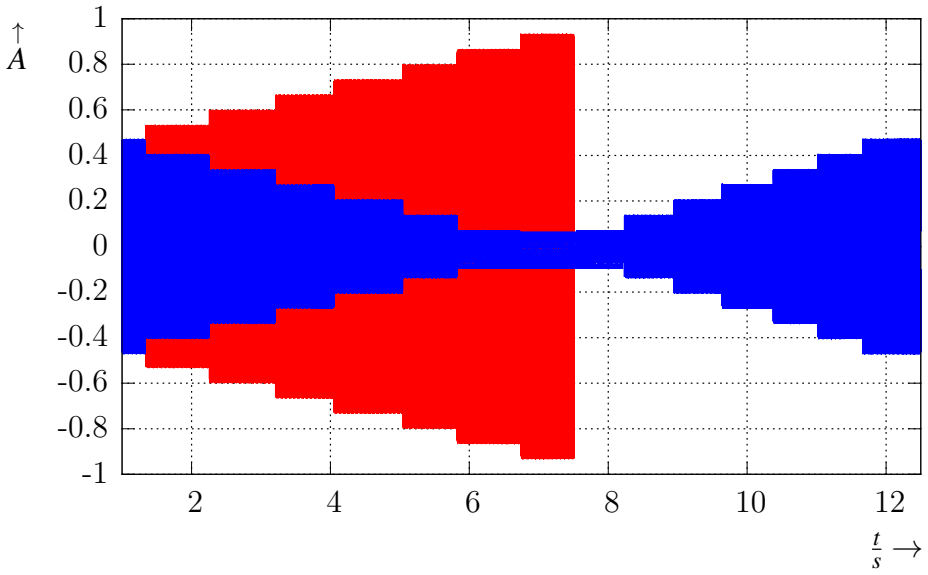


Bild 84: Balanceverschiebung nach Links und Anschließend wieder zurück zur Mittelstellung.

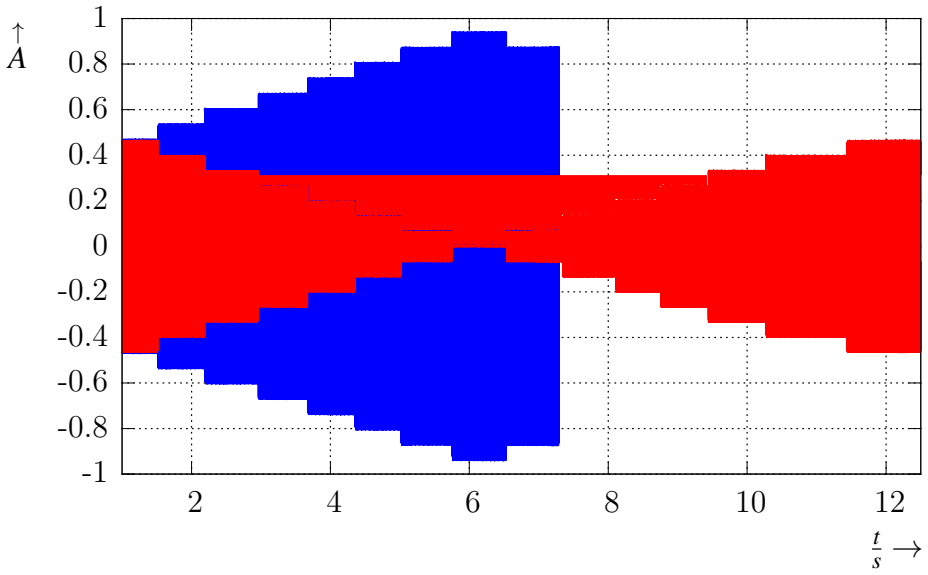


Bild 85: Balanceverschiebung nach Rechts und Anschließend wieder zurück zur Mittelstellung.

6.4. Die Stummschaltung

In der letzten Messung wird das Ein- und Ausblenden der Stummschaltung geprüft. Zunächst wird die Stummschaltung abgeschaltet und die Einblendzeit ermittelt. Diese wurde auf 5 Sekunden eingestellt. Anschließend wird die Stummschaltung wieder aktiviert und die Ausblendzeit gemessen. Im Bild 86 ist zu sehen, dass das Ein- und Ausblenden die eingestellten 5 Sekunden gedauert hat.

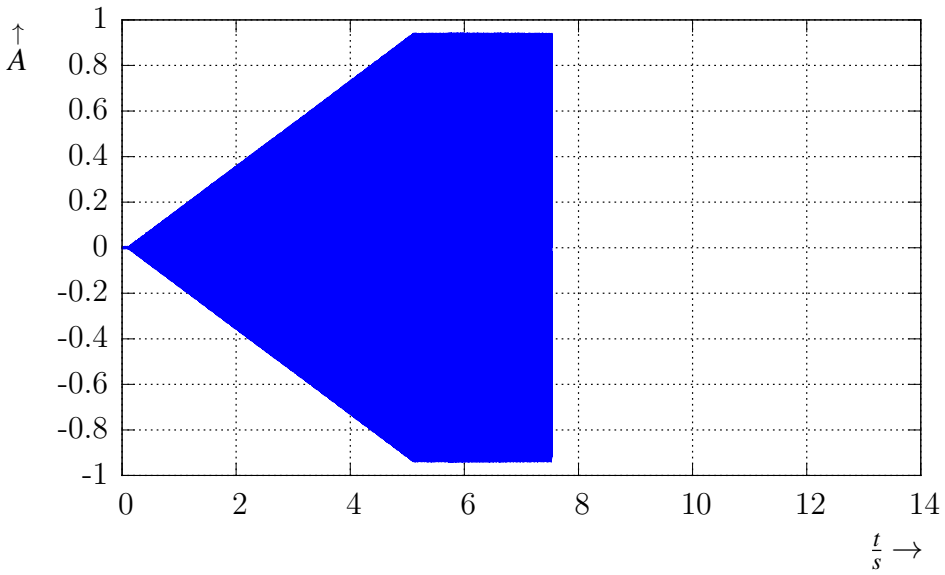


Bild 86: Einblenden und Ausblenden bei aktivieren bzw. deaktivieren der Stummschaltung.

7. Tests und erkannte Fehler

Getestet wurde nicht die Audiosignalverarbeitung, da diese durch die Messungen überprüft wurde. Es wurden die Benutzerschnittstellen und die Kommunikation zwischen diesen und der CU sowie die Kommunikation zwischen CU und AU getestet.

Die Kommunkation zwischen AU und CU wurde in der Art getestet, dass die Parameterwerte, welche von der CU gesendet wurden, mittels Konsolenausgabe in der IDE *Code Composer* überprüft wurden. Gleichzeitig konnte so die Eingabe über die UI geprüft werden. Wurde eine Eingabe an der UI vorgenommen, so muss auf der Konsole der Wert des Parameters erscheinen. War dieser richtig, so hat auch die Übertragung funktioniert und somit auch die Eingabe an der UI. Im Folgenden soll dies näher am Beispiel der Änderung der Grenzfrequenz $f_{c_{lp}}$ des Tiefen-Shelving-Filters erläutert werden.

Im ASPP in der Datei *filter.c* wurde im Kopfbereich die Bibliothek für Standard-Ein-/Ausgabe-Operationen eingebunden:

```
#include <stdio.h>
```

Des Weiteren wurde eine Konstante *test* definiert. Wird diese auskommentiert, werden alle Tests nicht mehr durchgeführt.

```
#define test
```

Die Funktion *set_lp_fc* dient dazu die Grenzfrequenz $f_{c_{lp}}$ einzustellen und wird bei der Übertragung des Parameterwertes von der CU zur AU aufgerufen. Am Ende der Funktion wurden folgende Zeilen eingefügt:

```
#ifdef test
   printf("f_c_lp : \t%f \t", lp_fc);
#endif
```

Damit ist gewährleistet, dass die printf-Anweisung angewandt wird, wenn die Konstante *test* definiert wurde. Die printf-Anweisung sorgt dafür, dass auf der Konsole der Wert der globalen Variable *lp_fc* angezeigt wird. Dies geschieht in folgender Form:

```
f_c_lp:   130.000000
f_c_lp:   140.000000
f_c_lp:   150.000000
```

Es ist zu sehen, dass zunächst die Grenzfrequenz $f_{c_{lp}}$ auf 130 Hz eingestellt wurde und im weiteren Verlauf jeweils um 10 Hz erhöht wurde. So wurden alle Parameter, welche im Anhang C aufgeführt sind überprüft.

Bei der Prüfung wurde ein Fehler ermittelt. Dieser trat in der Stummschaltung auf. Wurde die Ein- und Ausblendzeit t_F auf 0 s gesetzt, so ergab sich in der Gleichung zur Berechnung des Verstärkungsfaktors g_F, welche zunächst mit

$$g_F(n) = \left| c_m(n) - \frac{n_F(n)}{n_{F_{max}}} \right| \tag{50}$$

definiert war, eine Division durch 0. Setzt man in Gleichung (33) $t_F = 0$ s so wird $n_{F_{max}} = 0$, was in Gleichung (50) zu einer Division durch 0 führt. Die Auswirkung auf die Stummschaltung ist, dass diese ständig aktiv ist. Der Fehler wurde dadurch behoben, dass die Gleichung eine Fallunterscheidung erhielt und zu Gleichung (38) formuliert wurde.

In dieser Weise wurden auch die Eingaben über die GUI und das WI geprüft. Werden in der AU die Werte geändert, so sind die Eingaben an den oben genannten Benutzerschnittstelle angekommen und die Kommunikation funktioniert in diese Richtung. Wurden auch die Werte in den Anzeigen der GUI und des WIs geändert, so funktioniert auch die Kommunikation von der CU zu den Benutzerschnittstellen. Bei dieser Prüfung konnten keine Fehler festgestellt werden.

Noch nicht getestet wurden die Funktionen der Netzwerkeinstellung, das Speichern und Laden der Konfiguration, das Rücksetzen des Klangreglers, das Rücksetzen auf die „Werkseinstellungen“ und die Abschaltverzögerung der Hintergrundbeleuchtung des LCD. Die Tests zu den zuvor genannten Funktionen, sollen im Folgenden näher beschrieben werden.

Das Speichern und Laden der Konfiguration wurde dadurch getestet, dass eine bestimmte Konfiguration eingestellt wurde und auf einem Speicherort abgelegt wurde. Anschließend wurden alle Parameter der Konfiguration verändert und die zuvor gespeicherte Konfiguration geladen. Es wurde dann geprüft, dass alle Parameter auf die Werte eingestellt wurden, welche dem des gespeicherten Zustandes entsprachen. Dies wurde mit allen Speicherorten und Variationen (z.B. Netzwerkmodus) wiederholt. Bei diesem Test konnten keine Fehler festgestellt werden.

Des Weiteren wurde geprüft, dass beim Einschalten oder Rücksetzen des Klangreglers die Konfiguration, welche am Speichert 0 (default) hinterlegt ist, auch eingestellt wurde. Dazu wurde eine Konfiguration am Speicherort 0 gespeichert und die aktuelle Konfiguration auf andere Werte eingestellt und ein Rücksetzen ausgelöst. Nach dem Einschaltvorgang wurden die Parameter darauf überprüft, dass diese die entsprechenden Werte eingenommen haben. Auch bei dieser Prüfung traten keine Fehler auf. Mit dieser Prüfung wurde auch gleich das Rücksetzen geprüft. Sowohl das TMS320C6713 DSK, das Arduino Due und das Arduino Ethernet Shield erhielten das Rücksetzsignal und wurden zurückgesetzt.

Bei dem Test, ob beim Ausführen des Menüpunkts *set to factory settings* die Konfiguration am Speicherort 0 (default) auf die Werkseinstellungen gesetzt wurde und das Rücksetzen ausgelöst wurde, wurden zunächst die Parameterwerte der Konfiguration auf andere als die der „Werkseinstellung“gesetzt. Anschließend wurde der Menüpunkt *set to factory settings* aufgerufen und nach beenden des Einschaltvorgangs geprüft, dass die Parameterwerte denen der „Werkseinstellung“ entsprachen. Auch bei diesem Test wurden keine Fehler festgestellt.

Die Abschaltverzögerung der Hintergrundbeleuchtung des LCD wurde mit einer Stoppuhr getestet. Dazu wurde bei abgeschalteter Hintergrundbeleuchtung der Taster S2 betätigt und gleichzeitig die Stoppuhr gestartet. Bei Abschalten der Hintergrundbeleuchtung wurde die Stoppuhr gestoppt und die Zeit abgelesen. Diese Messung wurde mit den Abschaltzeiten 10 s, 20 s, 30 s, 40 s, 50 s und 60 s durchgeführt. Es kann davon ausgegangen werden, wenn die Tests bei diesen Werten zu keinem Fehler führen auch die anderen Zeiten stimmen. Eine Abweichung konnte nicht festgestellt werden bzw. lag unterhalb einer Sekunde, wobei hier die Reaktionszeit beim Stoppen mit der Stoppuhr zu berücksichtigen ist.

Die nächsten Tests beschäftigten sich mit der Netzwerkeinstellung. Hier wurden zunächst die Modi getestet. Dazu wurde zunächst der Modus *off* eingestellt und es wurde versucht eine Verbindung mit dem Klangregler über das WI und die GUI herzustellen. Dies blieb ohne Erfolg, womit der Test erfolgreich war. War allerdings zuvor einer der anderen Modi eingestellt und eine IP-Adresse vergeben, so konnte man das Gerät auf dieser noch erreichen. Dies stellt insofern

einen Fehler dar, als dass bei Vergabe dieser IP-Adresse an ein anderes Gerät ein Adressenkonflikt entsteht. Leider lässt es die Bibliothek *Ethernet* von Arduino nicht zu, einen gestarteten Server zu beenden. Hier würde nur ein Rücksetzen des Arduino Ethernet Shields helfen, bei dem beim Startvorgang der Server nicht erneut gestartet wird. Da dies aber auch einen Neustart des gesamten Klangreglers bedeuten würde, kam dies nicht zur Anwendung. Abhilfe wurde dadurch geschaffen, dass bei Wechsel in den Modus *off* die IP-Adresse des Klangreglers auf 127.0.0.1 (localhost) gesetzt wurde. Bei der Rückkehr in den Modus *static* wird die Standardadresse 192.168.0.100 eingestellt.

8. Zusammenfassung und Ausblick

In der hier vorliegenden Arbeit wurden zunächst die Grundlagen zum Thema der digitalen Audiosignalverarbeitung in Hinsicht auf die Themen der Klangregelung zusammengetragen. Weiterhin wurden grobe Grundlagen zur Steuerung gelegt. Daran schloss sich eine Simulation mit MATLAB/Simulink an. In dieser wurden die Verfahren der Audiosignalverarbeitung simuliert. Es wurde sich an das in den Grundlagen erstellte Blockschaltbild für den Signalfluss gehalten. Mit der Simulation konnte nachgewiesen werden, dass die Verfahren aus den Grundlagen für die Praxis anwendbar sind.

Mit Abschluss der Simulationsphase wurde mit dem praktischen Aufbau begonnen. Dazu wurde das DSP-Board festgelegt und anschließend wurden verschiedene Mikrocontroller-Boards in einem kleinen Rahmen getestet. Es wurde sich schließlich für das Arduino Due zusammen mit dem Arduino Ethernet Shield entschieden. Weiter wurde nach einer Kombination aus Schaltern und Inkrementalgebern für die UI gesucht und nach Festlegung dieser das Gehäuse ausgewählt. Parallel zum Erarbeiten des praktischen Aufbaus wurde mit der Erstellung der Software begonnen. Zunächst wurde das ASPP erstellt. Diesem folgte das CUIP. Erst nach den Tests, dass die Kommunikation zwischen CU und AU funktioniert, wurde auch mit der Erstellung des TCGUIP und TCWIP begonnen.

Die ersten Messungen wurden bereits nach Abschluss der Erstellung des ASPP durchgeführt. Mit diesen konnte die Funktion der Audiosignalverarbeitung nachgewiesen werden. Das ASPP wurde so angelegt, dass es mit weiteren Filtern zu einem Equalizer (EQ) ausgebaut werden kann. Dazu braucht die Kaskade nur um Peak-Filter 2. Ordnung, ausgelegt als IIR-Filter, erweitert werden. Weiter ist auch durch die Steuerung außerhalb des DSP genügend Raum, um Effekte wie Echo, Hall, Flanger, Chorus usw. hinzuzufügen. Auch denkbar wären Dolby Pro Logic oder andere Raumklangverfahren. Für einen Ausbau auf 5.1-Systeme o. ä. könnte das DSP-Board mit den diversen Daughter Cards[22] ausgestattet werden. So liese sich die Anzahl an Eingängen und Ausgängen erweitern, um entsprechende Raumklangverfahren abbilden zu können. Für den Anfang könnte es auch lediglich ein Austausch des Limiters durch einen Kompressor sein, welcher nicht ganz so stark die Dynamik beeinflusst.

Eine weitere Verbesserung kann darin liegen, das LCD durch ein Grafisches zu ersetzen, so dass die Filterkurven oder das Spektrum des Audiosignals auch auf dem LCD angezeigt werden kann. Der Arduino Due besitzt eine USB-Schnittstelle. Diese könnte für die Steuerung,

[22] z.B. http://www.educationaldsp.com/stockproduct_dsk_audio16_base.htm

statt der Ethernet-Schnittstelle, verwendet werden. Mit Nutzung und Erweiterung des SDRAM lässt sich die Klangregelung zu einem Mischpult, Equalizer und Effektgerät in einem einzigen Gerät erweitern. Es gibt für die Zukunft dieser Kombination aus DSP und MCU noch viele Möglichkeiten des Einsatzes ohne dabei die grundsätzliche Programmierung und den Aufbau der Hardware zu verändern.

Literatur

[1] WEINZIERL, Stefan: *Handbuch der Audiotechnik*. Springer-Verlag Berlin Heidelberg, 2008. – ISBN 978–3–540–34300–4

[2] INCORPORATED, Texas I.: *TLV320AIC23 Stereo Audio CODEC, 8-to 96-kHz, With Integrated Headphone Amplifier - Data Manual*. SLWS106C. Texas Instruments, Post Office Box 655303, Dallas, Texas 75265, July 2001

[3] ZÖLZER, Udo: *Digitale Audiosignalverarbeitung*. 3. überarbeitete und erweiterte Auflage 2005. Vieweg+Teubner GWV Fachverlage GmbH, Wiesbaden, 2008. – ISBN 978–3–519–26180–3

[4] ZÖLZER, Udo: *DAFX: Digital Audio Effects*. 2. Auflage. John Wiley and Sons, Ltd., Publication, 2011. – ISBN 978–0–470–66599–2

[5] INCORPORATED, Texas I.: *TMS320C6713 DSK Technical Reference*. 506735-0001 Rev. B. Texas Instruments, Post Office Box 655303, Dallas, Texas 75265, November 2003

[6] INCORPORATED, Texas I.: *TMS320C6000 DSP Inter-Integrated-Circuit (I2C) Module Reference Guide*. SPRU175A. Texas Instruments, Post Office Box 655303, Dallas, Texas 75265, October 2002

[7] HOFFMANN, Josef ; QUINT, Franz: *Signalverarbeitung mit MATLAB und Simulink*. Oldenbourg Wissenschaftsverlag GmbH, 2007. – ISBN 978–3–486–58427–1

[8] ANGERMANN, Anne ; BEUSCHEL, Michael ; RAU, Martin ; WOHLFARTH, Ulrich: *MATLAB - Simulink - Stateflow Grundlagen, Toolboxen, Beispiele*. 7., aktualisierte Auflage. Oldenbourg Wissenschaftsverlag GmbH, 2011. – ISBN 978–3–486–70585–0

[9] STOTZ, Dieter: *Computergestützte Audio- und Videotechnik - Multimediatechnik in der Anwendung*. Springer-Verlag Berlin Heidelberg, 2011. – ISBN 978–3–642–23253–4

[10] RULPH CHASSAING, Donald R.: *Digital Signal Processing and Applications with TMS320C6713 and TMS320C6416 DSK*. Second Edition. John Wiley and Sons, Ltd., Publication, 2008. – ISBN 978–0–470–13866–3

[11] INCORPORATED, Texas I.: *TMS320C6000 DSP Multichannel Buffered Serial Port (McBSP) Reference Guide*. Texas Instruments, Post Office Box 655303, Dallas, Texas 75265, March 2004

[12] INCORPORATED, Texas I.: *TMS320C6000 DSP Host Port Interface (HPI) Reference Guide*. Texas Instruments, Post Office Box 655303, Dallas, Texas 75265, September 2003

[13] INCORPORATED, Texas I.: *TMS320C6000 DSP Multichannel Audio Serial Port (McASP) Reference Guide*. Texas Instruments, Post Office Box 655303, Dallas, Texas 75265, August 2003

[14] ARDUINO: *Arduino Language Reference*. `http://arduino.cc/en/Reference/HomePage`. Version: November 2013

[15] ARDUINO: *Arduino Ethernet Library*. `http://arduino.cc/en/Reference/Ethernet`. Version: November 2013

[16] ARDUINO: *Arduino SD Library*. `http://arduino.cc/en/Reference/SD`. Version: November 2013

[17] ARDUINO: *Arduino Wire Library*. `http://arduino.cc/en/Reference/Wire`. Version: November 2013

[18] ARDUINO: *Arduino Due*. `http://arduino.cc/en/Main/ArduinoBoardDue`. Version: November 2013

[19] ARDUINO: *Arduino Ethernet Shield*. `http://arduino.cc/en/Main/ArduinoEthernetShield`. Version: November 2013

[20] H., Mario: *Arduino LiquidCrystal_I2C library*. `http://hmario.home.xs4all.nl/arduino/LiquidCrystal_I2C/`. Version: November 2013

Anhang

Anhangsverzeichnis

A. Datenblatt des Gerätes

Stromversorgung:	6 V DC; 750 mA
Tiefpass:	Grenzfrequenz 20 Hz bis 1000 Hz in 1-Hz-Schritten einstellbar Verstärkung -30 dB bis 30 dB in 1-dB-Schritten einstellbar
Hochpass:	Grenzfrequenz 1000 Hz bis 16000 Hz in 1-Hz-Schritten einstellbar Verstärkung -30 dB bis 30 dB in 1-dB-Schritten einstellbar
Eingangslautstärke:	Einstellbar von -34,5 dB bis 12,0 dB in 1,5-dB-Schritten
De-Emphasis-Filter:	deaktiviert; 32,0 kHz; 44,1 kHz; 48 kHz wählbar
Lautstärke:	wählbar zwischen -30 dB und 30 dB in 1-dB-Schritten
Stummschaltung:	mit Ein- und Ausblendfunktion; Ein-/Ausblendzeit wählbar von 0 s bis 10 s in 1-Sekunden-Schritten
Balance:	Lautstärkeverhältnis zwischen linkem und rechtem Kanal einstellbar in jeweils 7 Schritten.
Limiter:	Limiterschwelle wählbar, bezogen auf die maximale Amplitude, von 0 % bis 100 % in 1-Prozent-Schritten. Ansprechzeit ist einstellbar von 20 μs bis 10 ms in 1-μs-Schritten. Rücklaufzeit ist einstellbar von 2 ms bis 5 s in 1-ms-Schritten.
LCD:	2 Zeilen; 16 Zeichen; 5x8 Punkt-Matrix; Abschaltzeit wählbar im Bereich 0 s bis 60 s in 1-Sekunden-Schritten.
Speicher:	10 Speicherplätze zum Speichern der gesamten Konfiguration des Klangreglers.

B. Materialliste

Im folgenden ist das verwendete Material in Stückzahl und mit der Bezugsquelle aufgeführt. Für die Bezugsquellen werden folgende Abkürzungen verwendet:

Bezugsquelle	Abkürzung
Conrad Electronic SE Klaus-Conrad-Str. 1 92240 Hirschau http://www.conrad.de	conrad
Farnell GmbH Keltenring 14 82041 Oberhaching http://de.farnell.com/	farnell
Link Research 131 Fairview Street Providence, RI 02908 http://www.link-research.com/	link research
Reichelt Elektronik GmbH & Co. KG Elektronikring 1 26452 Sande Germany	reichelt

http://www.reichelt.de	
Watterott electronic GmbH Breitenhölzer Str. 6 37327 Leinefelde Deutschland http://www.watterott.com/	watterott

Abkürzungen zu den Bezugsquellen

Bezeichnung	Anzahl	Bezugsquelle	Artikelnr.
80 pin Samtec Breakout Panel	1 Stk.	link research	LR-BP-80PIN
Abstandsbolzen M 3,0 Innengewinde 10 mm Länge sechskant Schlüsselweite 5,5 mm	9 Stk.	conrad	526517
Arduino Due	1 Stk.	watterott	A000062
Arduino Ethernet Shield R3	1 Stk.	watterott	A000072
Audio/NF-Kabel Klinkenstecker 3,5 mm gewinkelt	2 Stk.	conrad	739621
CAT-5e-Stecker 8p8c (RJ45)	2 Stk.	conrad	397266
CCA Netzwerkkabel CAT 5e, abgepackt	1 Stk.	conrad	608427
Connector Board	1 Stk.		
DC-Buchse 2,5 mm FC681446	1 Stk.	conrad	735627
Distanzrolle ohne Gewinde Polystyrol M 3,0 10 mm Länge	4 Stk.	conrad	526363
Drehimpulsg., 15/30, vert., MT	5 Stk.	reichelt	STEC11B03
Drehknopf für Rundachse 6 mm	5 Stk.	reichelt	KNOPF 13-164B
Drucktaster	3 Stk.	conrad	701051
Flachbandkabel 20 Pole AWG 28	2 Meter	conrad	601978
Gehäuse ATPH 1865-300 + ATD Deckel	1 Stk.	reichelt	ATPH 1865-300

Bezeichnung	Anzahl	Bezugsquelle	Artikelnr.
I^2C-LC-Display-Modul	1 Stk.	conrad	198330
Klinken-Steckverbinder 3.5 mm Buchse	2 Stk.	conrad	718489
LCD-Modul Gleichmann GE-C1602B-TMI-JT/R	1 Stk.	conrad	183045
Litze, 2 x 0,14 mm^2	1 Stk.	reichelt	ZL 214SWW-5
Lochraster-Platine	1 Stk.	conrad	531341
Metallschicht-Widerstand 2.2 kΩ 0.25 W	1 Stk.	conrad	408204
microSDHC-Karte 4 GB	1 Stk.	conrad	417479
Niedervolt-Steckverbinder Stecker, gewinkelt 5.5 mm 2.5 mm	3 Stk.	conrad	393468
Pfostenverbinder RM 2,54 Rastermaß: 2.54 mm Pole: 2 x 8	1 Stk.	conrad	742145
Pfostenverbinder RM 2,54 Rastermaß: 2.54 mm Pole: 2 x 10	1 Stk.	conrad	742166
RJ45-Einbaubuchse	1 Stk.	conrad	732291
Schraube M 2,5 x 16 DIN 84	4 Stk.	reichelt	SZK M2,5X16-200
Schraube M 3,0 x 6,0 DIN 7985	18 Stk.	conrad	815322
Sechskantmutter M 2,5	4 Stk.	reichelt	SK M2,5-100

Bezeichnung	Anzahl	Bezugsquelle	Artikelnr.
Stecker-Netzteil VOLTCRAFT USPS-1000	1 Stk.	conrad	518371
Stiftleiste RM 2,54, gerade Pole: 1 x 10	1 Stk.	conrad	393481
TMS320C6713 DSK	1 Stk.	farnell	1831919

Materialliste der Klangregelung

Die Stiftleiste mit 1x10 Polen ist in 2 Stiftleisten mit 1x8 und 1x2 Pole zu teilen. Diese beiden Stiftleisten werden für die Steckverbinder X1 und X2 verwendet. Die Lochrasterplatine wird ebenfalls für die Steckverbinder X1 und X2 verwendet. Dazu ist diese für X1 in die Größe 4x10 Löscher und für X2 in die Größe 4x4 Löscher zu zerlegen.

Zusätzlich wird für das Connector-Board folgendes Material benötigt:

Bezeichnung	Anzahl	Bezugsquelle	Artikelnr.
Doppelreihige Stiftleiste RM 2,54, gerade Pole: 2 x 20 (Achtung!!! Muss auf 2x18 gekürzt werden!)	1 Stk.	conrad	393508
Pfostenverbinder RM 2,54 Rastermaß: 2.54 mm Pole: 2 x 3	3 Stk.	conrad	742063
Platine	1 Stk.	selbstständig herzustellen	
Stiftleiste Print-Montage gerade 2x3 Pole RM 2,54	2 Stk.	conrad	741435
Stiftleiste Print-Montage gerade 2x10 Pole RM 2,54	2 Stk.	conrad	741766

Bezeichnung	Anzahl	Bezugsquelle	Artikelnr.
Stiftleiste RM 2,54, gerade Pole: 1 x 8	2 Stk.	conrad	393480

Materialliste zum Connector Board

C. Steuersignale

Im folgenden sind die Steuersignale mit ihren Eigenschaften aufgelistet, welche zum Einstellen der Parameter vom Mikroprozessor zum Signalprozessor, über den I^2C-Bus übermittelt werden.

Bezeichnung:	Eingangslautstärke (Line in Volume)
Symbol:	G_{ADC}
Einstellbereich:	-34,5 dB bis 12,0 dB in 1,5-dB-Schritten
Steuerparameter:	$s_{G_{ADC}}$
Registeradresse:	0
Wortbreite:	8 bit
Steuerwerte:	$s_{G_{ADC}} = \frac{G_{ADC}+34,5}{1,5}$
Beschreibung:	Die Eingangslautstärke (Line in Volume) G_{ADC} ist die Lautstärkeeinstellung, welche im Audio-Codec für das Eingangssignal vorgegeben werden kann.

Bezeichnung:	Deakzentuierung (De-Emphasis)
Einstellungen:	off, 32 kHz, 44,1 kHz, 48 kHz
Steuerparameter:	s_d
Registeradresse:	1
Wortbreite:	8 bit
Steuerwerte:	0 - off 1 - 32 kHz 2 - 44,1 kHz 3 - 48 kHz
Beschreibung:	Das De-Emphasis ist ein Deakzentuierungsfilter, welches eine Akzentuierung bei der Aufnahme oder Übertragung aufhebt.

Bezeichnung:	DAC select
Einstellungen:	enable, disable
Steuerparameter:	$s_{DAC_{en}}$
Registeradresse:	2
Wortbreite:	8 bit
Steuerwerte:	0 - disable 1 - enable
Beschreibung:	Der Steuerparameter *DAC select* gibt an, ob das DAC zugeschaltet sein soll oder nicht.

Bezeichnung:	Bypass
Einstellungen:	enable, disable
Steuerparameter:	s_b
Registeradresse:	3
Wortbreite:	8 bit
Steuerwerte:	0 - disable
	1 - enable
Beschreibung:	Der Bypass ist im Audio-Codec verankert und schleift das Eingangssignal direkt auf den Ausgang. Ist der DAC noch zugeschaltet, dann werden beide Signale auf den Ausgang gelegt und überlagern sich. Somit muss für das verwenden des Bypass nicht nur der Steuerparameter s_b geschaltet werden, sondern auch der Steuerparameter $s_{DAC_{en}}$.

Bezeichnung:	Lowpass Bypass
Einstellungen:	enable, disable
Steuerparameter:	$s_{lp_{en}}$
Registeradresse:	4
Wortbreite:	8 bit
Steuerwerte:	0 - disable
	1 - enable
Beschreibung:	Über den Steuerparameter *Lowpass Bypass* kann der Tiefen-Shelving-Filter in den Audiosignalweg zugeschaltet werden oder aus diesem herausgenommen werden.

Bezeichnung:	Lowpass left channel bypass
Einstellungen:	enable, disable
Steuerparameter:	$s_{lp_{l_{en}}}$
Registeradresse:	5
Wortbreite:	8 bit
Steuerwerte:	0 - disable
	1 - enable
Beschreibung:	Über den Steuerparameter *Lowpass left channel bypass* kann das Tiefen-Shelving-Filter in den Signalweg des linken Kanals des Stereosignal zugeschaltet werden oder aus diesem herausgenommen werden.

Bezeichnung:	Lowpass right channel bypass
Einstellungen:	enable, disable
Steuerparameter:	$s_{lp_{ren}}$
Registeradresse:	6
Wortbreite:	8 bit
Steuerwerte:	0 - disable
	1 - enable
Beschreibung:	Über den Steuerparameter *Lowpass right channel bypass* kann das Tiefen-Shelving-Filter in den Signalweg des rechten Kanals des Stereosignal zugeschaltet oder aus diesem herausgenommen werden.

Bezeichnung:	Lowpass cutoff frequency
Einstellbereich:	20 Hz bis 1000 Hz in 1-Hz-Schritten
Symbol:	$f_{c_{lp}}$
Steuerparameter:	$s_{f_{c_{lp}}}$
Registeradresse:	7 (unteres Byte)
	8 (oberes Byte)
Wortbreite:	16 bit
Steuerwerte:	$s_{f_{c_{lp}}} = f_{c_{lp}}$
Beschreibung:	Über den Steuerparameter $s_{fc_{lp}}$ werden die Werte der Grenzfrequenz des Tiefen-Shelving-Filters übermittelt.

Bezeichnung:	Lowpass amplification factor
Einstellbereich:	-30 dB bis 30 dB in 1-dB-Schritten
Symbol:	G_{lp}
Steuerparameter:	$s_{G_{lp}}$
Registeradresse:	9
Wortbreite:	8 bit
Steuerwerte:	$s_{G_{lp}} = 30 + G_{lp}$
Beschreibung:	Über den Steuerparameter $s_{G_{lp}}$ wird der logarithmische Verstärkungsfaktor für den Tiefen-Shelving-Filter übertragen.

Bezeichnung:	Highpass Bypass
Einstellungen:	enable, disable
Steuerparameter:	$s_{hp_{en}}$
Registeradresse:	10
Wortbreite:	8 bit
Steuerwerte:	0 - disable
	1 - enable
Beschreibung:	Über den Steuerparameter *Highpass Bypass* kann der Höhen-Shelving-Filter in den Audiosignalweg zugeschaltet werden oder aus diesem herausgenommen werden.

Bezeichnung:	Highpass left channel bypass
Einstellungen:	enable, disable
Steuerparameter:	$s_{hp_{len}}$
Registeradresse:	11
Wortbreite:	8 bit
Steuerwerte:	0 - disable
	1 - enable
Beschreibung:	Über den Steuerparameter *Highpass left channel bypass* kann das Höhen-Shelving-Filter in den Signalweg des linken Kanals des Stereosignals zugeschaltet oder aus diesem herausgenommen werden.

Bezeichnung:	Highpass right channel bypass
Einstellungen:	enable, disable
Steuerparameter:	$s_{hp_{ren}}$
Registeradresse:	12
Wortbreite:	8 bit
Steuerwerte:	0 - disable
	1 - enable
Beschreibung:	Über den Steuerparameter *Highpass right channel bypass* kann das Höhen-Shelving-Filter in den Signalweg des rechten Kanals des Stereosignal zugeschaltet oder aus diesem herausgenommen werden.

Bezeichnung:	Highpass cutoff frequency
Einstellbereich:	1000 Hz bis 16000 Hz in 1-Hz-Schritten
Symbol:	$f_{c_{hp}}$
Steuerparameter:	$s_{f_{c_{hp}}}$
Registeradresse:	13 (unteres Byte)
	14 (oberes Byte)
Wortbreite:	16 bit
Steuerwerte:	$s_{f_{c_{hp}}} = f_{c_{hp}}$
Beschreibung:	Über den Steuerparameter $s_{fc_{hp}}$ werden die Werte der Grenzfrequenz des Höhen-Shelving-Filters übermittelt.

Bezeichnung:	Highpass amplification factor
Einstellbereich:	-30 dB bis 30 dB in 1-dB-Schritten
Symbol:	G_{hp}
Steuerparameter:	$s_{G_{hp}}$
Registeradresse:	15
Wortbreite:	8 bit
Steuerwerte:	$s_{G_{hp}} = 30 + G_{hp}$
Beschreibung:	Über den Steuerparameter $s_{G_{hp}}$ wird der logarithmische Verstärkungsfaktor für den Höhen-Shelving-Filter übertragen.

Bezeichnung:	Volume
Einstellbereich:	-30 dB bis 30 dB in 1-dB-Schritten
Symbol:	G_V
Steuerparameter:	s_{G_V}
Registeradresse:	16
Wortbreite:	8 bit
Steuerwerte:	$s_{G_V} = 30 + G_V$
Beschreibung:	Mit dem Steuerparameter Volume wird der logarithmische Verstärkungsfaktor G_V für den Block *Volume* der Lautstärkeeinstellung übertragen.

Bezeichnung:	Fade time
Einstellbereich:	0 s bis 10 s in 1-s-Schritten
Symbol:	t_F
Steuerparameter:	s_{t_F}
Registeradresse:	17
Wortbreite:	8 bit
Steuerwerte:	$s_{t_F} = t_F$
Beschreibung:	Die Ein- und Ausblendzeit t_F gibt an, wie lange das Ein- und Ausblenden beim Zu- oder Abschalten der Stummschaltung (Mute) dauern soll.

Bezeichnung:	Balance
Einstellbereich:	-7 bis 7 in 1er-Schritten
Symbol:	B
Steuerparameter:	s_B
Registeradresse:	18
Wortbreite:	8 bit
Steuerwerte:	$s_B = B$
Beschreibung:	Der Balancewert der im Bereich von -7 bis 7 als ganze natürliche Zahl angegeben wird, repräsentiert das Lautstärkeverhältnis zwischen rechtem und linken Kanal des Stereosignals.

Bezeichnung:	Mute
Einstellungen:	enable, disable
Symbol:	c_m
Steuerparameter:	s_{c_m}
Registeradresse:	19
Wortbreite:	8 bit
Steuerwerte:	$s_{c_m} = c_m$
Beschreibung:	Der Wert c_m gibt mit 0 an, dass die Stummschaltung abgeschaltet und mit 1, dass diese zugeschaltet ist. Dieser Wert wird direkt über den Steuerparameter s_{c_m} übertragen.

Bezeichnung:	Limiter bypass
Einstellungen:	enable, disable
Symbol:	lt_{en}
Steuerparameter:	$s_{lt_{en}}$
Registeradresse:	20
Wortbreite:	8 bit
Steuerwerte:	$s_{lt_{en}} = lt_{en}$
Beschreibung:	Mit dem Steuerparameter $s_{lt_{en}}$ wird an den Signalprozessor übertragen, ob der Limiter in den Signalweg zugeschaltet oder aus diesem herausgenommen werden soll.

Bezeichnung:	Threshold
Einstellbereich:	0 % bis 100 % in 1-%-Schritten
Symbol:	lt
Steuerparameter:	s_{lt}
Registeradresse:	21
Wortbreite:	8 bit
Steuerwerte:	$s_{lt} = lt$
Beschreibung:	Der Schwellwert (Threshold) des Limiters lt gibt die Schwelle in Prozent vom Maximalwert des Aussteuerungsbereiches des DAC an, bei der der Limiter begrenzen soll.

Bezeichnung:	Attack time
Einstellbereich:	20 μs bis 10000 μ in 1-μs-Schritten
Symbol:	t_a
Steuerparameter:	s_{t_a}
Registeradresse:	22 (unteres Byte) 23 (oberes Byte)
Wortbreite:	16 bit
Steuerwerte:	$s_{t_a} = t_a$
Beschreibung:	Die Ansprechzeit gibt an, wie schnell der Limiter auf ein überschreiten der Schwelle reagieren soll.

Bezeichnung:	Release time
Einstellbereich:	2 ms bis 5000 ms in 1-ms-Schritten
Symbol:	t_r
Steuerparameter:	s_{t_r}
Registeradresse:	24 (unteres Byte) 25 (oberes Byte)
Wortbreite:	16 bit
Steuerwerte:	$s_{t_r} = t_r$
Beschreibung:	Die Rücklaufzeit gibt an, wie schnell der Limiter die Begrenzung nach überschreiten der Schwelle zurücknehmen soll.

Die folgenden Steuerparameter werden nicht an den Signalprozessor übermittelt, sondern von den Benutzerschnittstellen an den Mikrocontroller übertragen, um diesen in seinem Verhalten zu steuern.

Bezeichnung:	LCD turn-off time
Einstellbereich:	0 s bis 10 s in 1-Sekunden-Schritten
Symbol:	t_{LCD}
Steuerparameter:	$s_{t_{LCD}}$
Registeradresse:	26
Wortbreite:	8 bit
Steuerwerte:	$s_{t_{LCD}} = t_{LCD}$
Beschreibung:	Die *LCD turn-off time* t_{LCD} gibt an, wie lange die Hintergrundbeleuchtung des LCD leuchten soll, wenn keine weitere Eingaben vom Benutzer gemacht werden. Nach Ablauf dieser Zeit, wird die Hintergrundbeleuchtung abgeschaltet.

Bezeichnung:	Storage location
Einstellbereich:	0 bis 9
Symbol:	*storage*
Steuerparameter:	$s_{storage}$
Registeradresse:	27
Wortbreite:	8 bit
Steuerwerte:	$s_{storage} = storage$
Beschreibung:	Die Angabe des Speicherortes für das Speichern der Konfiguration erfolgt über das Steuersignal $s_{storage}$.

Bezeichnung:	Load
Einstellbereich:	beliebig
Symbol:	*load*
Steuerparameter:	s_{load}
Registeradresse:	28
Wortbreite:	8 bit
Steuerwerte:	$s_{load} = load$
Beschreibung:	Wird der Steuerparameter s_{load} mit einem beliebigen Wert von einer Benutzerschnittstelle an den Mikrocontroller gesendet, so veranlasst dieser das Laden der am eingestellten Speicherort abgelegten Konfiguration.

Bezeichnung:	Save
Einstellbereich:	beliebig
Symbol:	*save*
Steuerparameter:	s_{save}
Registeradresse:	29
Wortbreite:	8 bit
Steuerwerte:	$s_{save} = save$
Beschreibung:	Wird der Steuerparameter s_{save} mit einem beliebigen Wert von einer Benutzerschnittstelle an den Mikrocontroller gesendet, so veranlasst dieser das Speichern der aktuellen Konfiguration im angegebenen Speicherort.

Bezeichnung:	Reset
Einstellbereich:	beliebig
Symbol:	*reset*
Steuerparameter:	s_{reset}
Registeradresse:	30
Wortbreite:	8 bit
Steuerwerte:	$s_{reset} = reset$
Beschreibung:	Wird der Steuerparameter s_{reset} mit einem beliebigen Wert von einer Benutzerschnittstelle an den Mikrocontroller gesendet, so veranlasst dieser das Zurücksetzen des Signalprozessors und des Mikrocontrollers.

Bezeichnung:	Factory settings
Einstellbereich:	beliebig
Symbol:	*factory*
Steuerparameter:	$s_{factory}$
Registeradresse:	31
Wortbreite:	8 bit
Steuerwerte:	$s_{factory} = factory$
Beschreibung:	Wird der Steuerparameter $s_{factory}$ mit einem beliebigen Wert von einer Benutzerschnittstelle an den Mikrocontroller gesendet, so veranlasst dieser die Speicherung der Standardwerte auf den Speicherort 0 und anschließend das Zurücksetzen des Signalprozessors und des Microcontrollers.

D. Funktionen der Simulation und deren Kommandos

Die folgende Tabelle listet die MATLAB-Funktionen und die darin verwendeten Kommandos des TCSP auf.

Funktion	Kommando	Beschreibung
tc		Befehl unter MATLAB um die Simulation mit der Benutzeroberfläche zu starten.
tc_audio	set_mute	Aktiviert oder Deaktiviert die Stummschaltung entsprechend der Auswahl an der Checkbox *cb_audio*.
	set_bypass	Aktiviert oder Deaktiviert den Bypass entsprechend der Auswahl über die Checkbox *cb_bypass*.
	set_balance	Ändert den Balancewert B entsprechend der Stellung des Sliders *sl_balance*.
	center	Veranlasst das Setzen des Balancewertes B auf 0.
	set_volume	Setzt den Verstärkungsfaktor G_V auf den Wert des im Eingabefeld *edit_volume* eingetragenen.
	set_volume_p1	Erhöht den Verstärkungsfaktor G_V um 1 dB.
	set_volume_p10	Erhöht den Verstärkungsfaktor G_V um 10 dB.
	set_volume_m1	Verringert den Verstärkungsfaktor G_V um 1 dB.
	set_volume_m10	Verringert den Verstärkungsfaktor G_V um 10 dB.
	set_fade	Stellt die Ein- und Ausblendzeit t_F auf den Wert ein, der im Eingabefeld *edit_fade* eingetragen ist.
	set_fade_p1	Erhöht die Ein- und Ausblendzeit t_F um eine Sekunde.
	set_fade_p10	Erhöht die Ein- und Ausblendzeit t_F um zehn Sekunde.
	set_fade_m1	Verringert die Ein- und Ausblendzeit t_F um eine Sekunde.
	set_fade_m10	Verringert die Ein- und Ausblendzeit t_F um zehn Sekunde.
	set_Fade_fs	Stellt die Konstante *fs* im Zählermodell *nF* des Modells *Fade* auf die Abtastfrequenz f_s ein.

Funktion	Kommando	Beschreibung
tc_close		Beendet die Simulation und schließt die Benutzeroberfläche. Dabei werden auch die Timer *plottimer* und *enabletimer* gestoppt.
tc_fvtool		Ruft das Tool *fvtool* auf und übergibt diesem die Kaskade aus Tiefen- und Höhen-Shelving-Filter im aktuellen Zustand.
	fvtool_lowpass	Ruft das Tool *fvtool* auf und übergibt diesem den Tiefen-Shelving-Filter im aktuellen Zustand.
	fvtool_highpass	Ruft das Tool *fvtool* auf und übergibt diesem den Höhen-Shelving-Filter im aktuellen Zustand.
tc_highpass	set_hp_en	Aktiviert oder Deaktiviert den Höhen-Shelving-Filter entsprechend der Auswahl an der Checkbox *cb_hp_en*.
	set_hp_fc	Berechnet die Koeffizienten des Tiefen-Shelving-Filters neu und übergibt diese an die entsprechenden LUTs des Simulationsmodells.
	set_hp_fc_p1	Inkrementiert die Grenzfrequenz $f_{c_{hp}}$.
	set_hp_fc_p10	Erhöht die Grenzfrequenz $f_{c_{hp}}$ um 10 Hz.
	set_hp_fc_p100	Erhöht die Grenzfrequenz $f_{c_{hp}}$ um 100 Hz.
	set_hp_fc_p1000	Erhöht die Grenzfrequenz $f_{c_{hp}}$ um 1000 Hz.
	set_hp_fc_m1	Dekrementiert die Grenzfrequenz $f_{c_{hp}}$.
	set_hp_fc_m10	Verringert die Grenzfrequenz $f_{c_{hp}}$ um 10 Hz.
	set_hp_fc_m100	Verringert die Grenzfrequenz $f_{c_{hp}}$ um 100 Hz.
	set_hp_fc_m1000	Verringert die Grenzfrequenz $f_{c_{hp}}$ um 1000 Hz.
	set_hp_G_p1	Inkrementiert den Verstärkungsfaktor G_{hp}.
	set_hp_G_p10	Erhöht den Verstärkungsfaktor G_{hp} um 10 dB.
	set_hp_G_m1	Dekrementiert den Verstärkungsfaktor G_{hp}.
	set_hp_G_m10	Verringert den Verstärkungsfaktor G_{hp} um 10 dB.
tc_init		Initialisiert die Anzeigen der Benutzeroberfläche und die Konstanten und LUTs im Simulationsmodell
tc_limiter	set_lt_en	Aktiviert oder Deaktiviert den Limiter entprechend der Auswahl in der Checkbox *cb_lt_en*
	set_lt_th	Setzt die Limiterschwelle *lt* im Simulationsmodell auf die im Eingabefeld *edit_lt_th* Eingetragene.
	set_lt_th_p1	Erhöht die Limiterschwelle *lt* um 1 %.

Funktion	Kommando	Beschreibung
	set_lt_th_p10	Erhöht die Limiterschwelle *lt* um 10 %.
	set_lt_th_m1	Verringert die Limiterschwelle *lt* um 1 %.
	set_lt_th_m10	Verringert die Limiterschwelle *lt* um 10 %.
	set_lt_at	Setzt die Ansprechzeit t_a auf den Wert der im Eingabefeld *edit_lt_ta* angegeben ist.
	set_lt_at_p1	Erhöht die Ansprechzeit t_a um 1 μs.
	set_lt_at_p10	Erhöht die Ansprechzeit t_a um 10 μs.
	set_lt_at_p100	Erhöht die Ansprechzeit t_a um 100 μs.
	set_lt_at_p1000	Erhöht die Ansprechzeit t_a um 1000 μs.
	set_lt_at_p10000	Erhöht die Ansprechzeit t_a um 10000 μs.
	set_lt_at_m1	Verringert die Ansprechzeit t_a um 1 μs.
	set_lt_at_m10	Verringert die Ansprechzeit t_a um 10 μs.
	set_lt_at_m100	Verringert die Ansprechzeit t_a um 100 μs.
	set_lt_at_m1000	Verringert die Ansprechzeit t_a um 1000 μs.
	set_lt_at_m10000	Verringert die Ansprechzeit t_a um 10000 μs.
	set_lt_rt	Setzt die Rücklaufzeit t_r auf den Wert der im Eingabefeld *edit_lt_tr* angegeben ist.
	set_lt_rt_p1	Erhöht die Rücklaufzeit t_r um 1 μs.
	set_lt_rt_p10	Erhöht die Rücklaufzeit t_r um 10 μs.
	set_lt_rt_p100	Erhöht die Rücklaufzeit t_r um 100 μs.
	set_lt_rt_p1000	Erhöht die Rücklaufzeit t_r um 1000 μs.
	set_lt_rt_m1	Verringert die Rücklaufzeit t_r um 1 μs.
	set_lt_rt_m10	Verringert die Rücklaufzeit t_r um 10 μs.
	set_lt_rt_m100	Verringert die Rücklaufzeit t_r um 100 μs.
	set_lt_rt_m1000	Verringert die Rücklaufzeit t_r um 1000 μs.
	set_LUTs	Lässt die Tabellen der LUTs AT und RT neu berechnen.
	set_AT_LUT	Berechnet die Tabelle der LUT AT und übergibt diese an die LUT.
	set_RT_LUT	Berechnet die Tabelle der LUT RT und übergibt diese an die LUT.
tc_lowpass	set_lp_en	Aktiviert oder Deaktiviert den Tiefen-Shelving-Filter entsprechend der Auswahl an der Checkbox *cb_lp_en*.

Funktion	Kommando	Beschreibung
	set_lp_fc	Berechnet die Koeffizienten des Tiefen-Shelving-Filters neu und übergibt diese an die entsprechenden LUTs des Simulationsmodells.
	set_lp_fc_p1	Inkrementiert die Grenzfrequenz $f_{c_{lp}}$.
	set_lp_fc_p10	Erhöht die Grenzfrequenz $f_{c_{lp}}$ um 10 Hz.
	set_lp_fc_p100	Erhöht die Grenzfrequenz $f_{c_{lp}}$ um 100 Hz.
	set_lp_fc_p1000	Erhöht die Grenzfrequenz $f_{c_{lp}}$ um 1000 Hz.
	set_lp_fc_m1	Dekrementiert die Grenzfrequenz $f_{c_{lp}}$.
	set_lp_fc_m10	Verringert die Grenzfrequenz $f_{c_{lp}}$ um 10 Hz.
	set_lp_fc_m100	Verringert die Grenzfrequenz $f_{c_{lp}}$ um 100 Hz.
	set_lp_fc_m1000	Verringert die Grenzfrequenz $f_{c_{lp}}$ um 1000 Hz.
	set_lp_G_p1	Inkrementiert den Verstärkungsfaktor G_{lp}.
	set_lp_G_p10	Erhöht den Verstärkungsfaktor G_{lp} um 10 dB.
	set_lp_G_m1	Dekrementiert den Verstärkungsfaktor G_{lp}.
	set_lp_G_m10	Verringert den Verstärkungsfaktor G_{lp} um 10 dB.
tc_player	play	Mit *play* wird das Abspielen einer geladenen WAVE-Datei gestartet. Ebenfalls wird die Simulation gestartet.
	pause	Lässt das Abspielen der Audiodaten und die Simulation pausieren.
	stop	Stoppt das Abspielen einer geladenen WAVE-Datei und die Simulation. Gleichzeitig wird die Simulation zurückgesetzt.
	reload	Liest die Daten einer WAVE-Datei erneut ein und legt diese in die entsprechenden Variablen auf dem MATLAB-Workspace ab.
	browse	Öffnet einen Dialog mit der eine WAVE-Datei ausgewählt werden kann und veranlasst das Laden der Daten dieser Datei.
	goto	Springt an die Stelle der Audiodaten, welche im Eingabefeld *edit_goto* angegeben ist. Die Angabe muss in der Einheit Sekunden erfolgen.

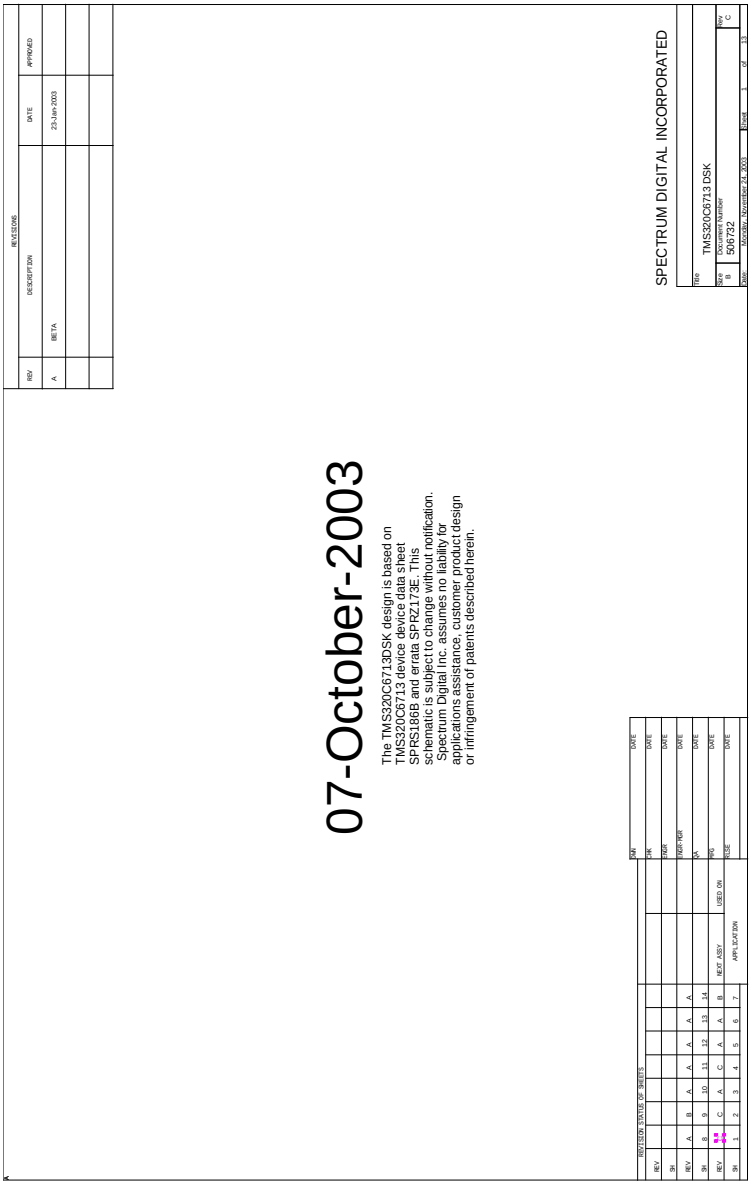
Funktion	Kommando	Beschreibung
tc_plot	plot_filter	Veranlasst das Löschen des derzeit angezeigten Kurvenverlaufs und zeichnet anschließend entsprechend der Auswahl der Radiobuttons *rb_amp* für den Amplitudengang, *rb_phase* für den Phasengang und *rb_delay* für die Gruppenlaufzeit den entsprechenden Kurvenverlauf neu.
tc_plot_signal		Je nach den Einstellungen der Radiobuttons *rb_freq* und *rb_time* wird der Kurvenverlauf des linken und rechten Kanals des Ein- und Ausgangssignals neu gezeichnet. Welches Signal und welcher Kanal davon gezeichnet wird hängt von der Auswahl der Checkboxen *cb_left_in*, *cb_right_in*, *cb_left_out* und *cb_right_out* ab. Mit welcher Auflösung, für die Zeitachse im Zeitbereich, gezeichnet wird hängt vom eingestellten Wert des Sliders *slider_time_signal_resolution* ab.
tc_set_enable		Aktiviert oder Deaktiviert bestimmte Eingabeelemente je nach Zustand der Simulation. Das heißt, wird beispielsweise der Lowpass deaktiviert, so veranlasst die Funktion das Deaktivieren der Eingabeelemente für die Grenzfrequenz $f_{c_{lp}}$ und den Verstärkungsfaktor G_{lp}.
tc_sim_stop		Bei Ende der Simulation wird von dieser die Funktion *tc_sim_stop* aufgerufen. Diese veranlasst das Stoppen der Simulation, das Setzen der Simulation auf den Anfang und das Laden aller Daten aus der Variablen *data* in die Variable x.
rad2deg		Mit der Funktion wird ein übergebener Wert, der im Bogenmaß anzugeben ist, in das Gradmaß umgerechnet und an die aufrufende Funktion zurückgegeben.

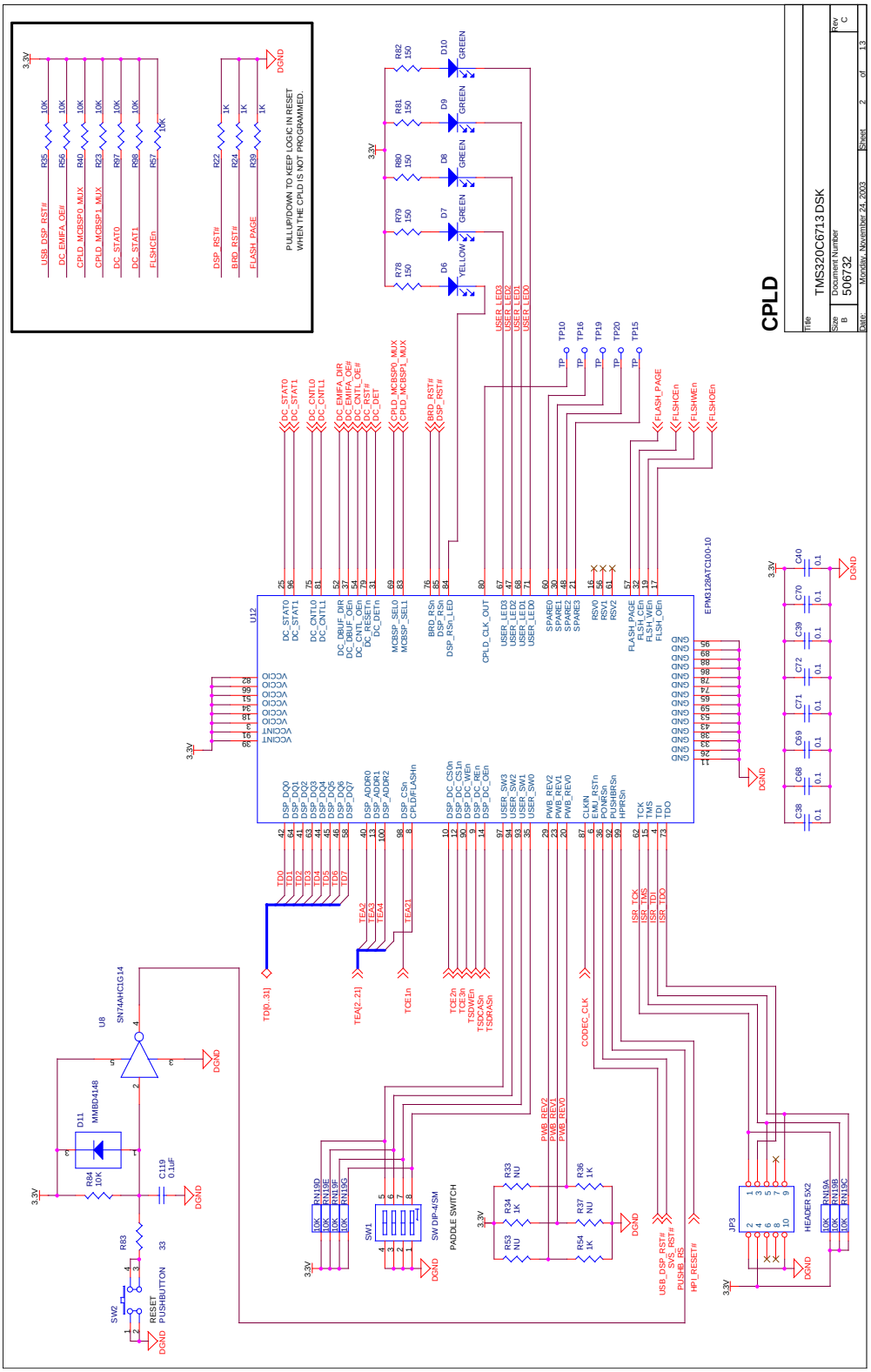
Funktionen und Kommandos der Simulation

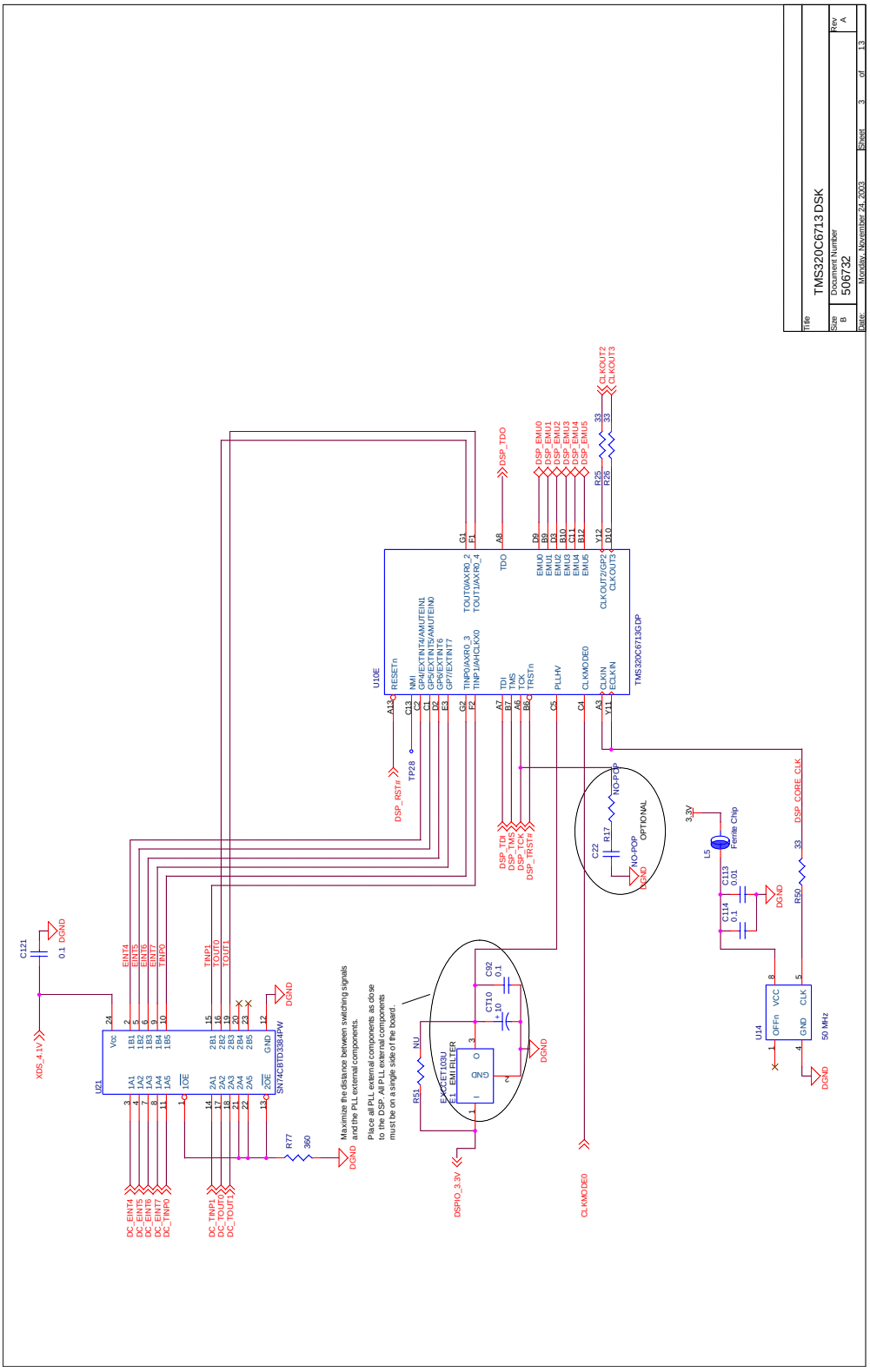
E. Herstellungsunterlagen

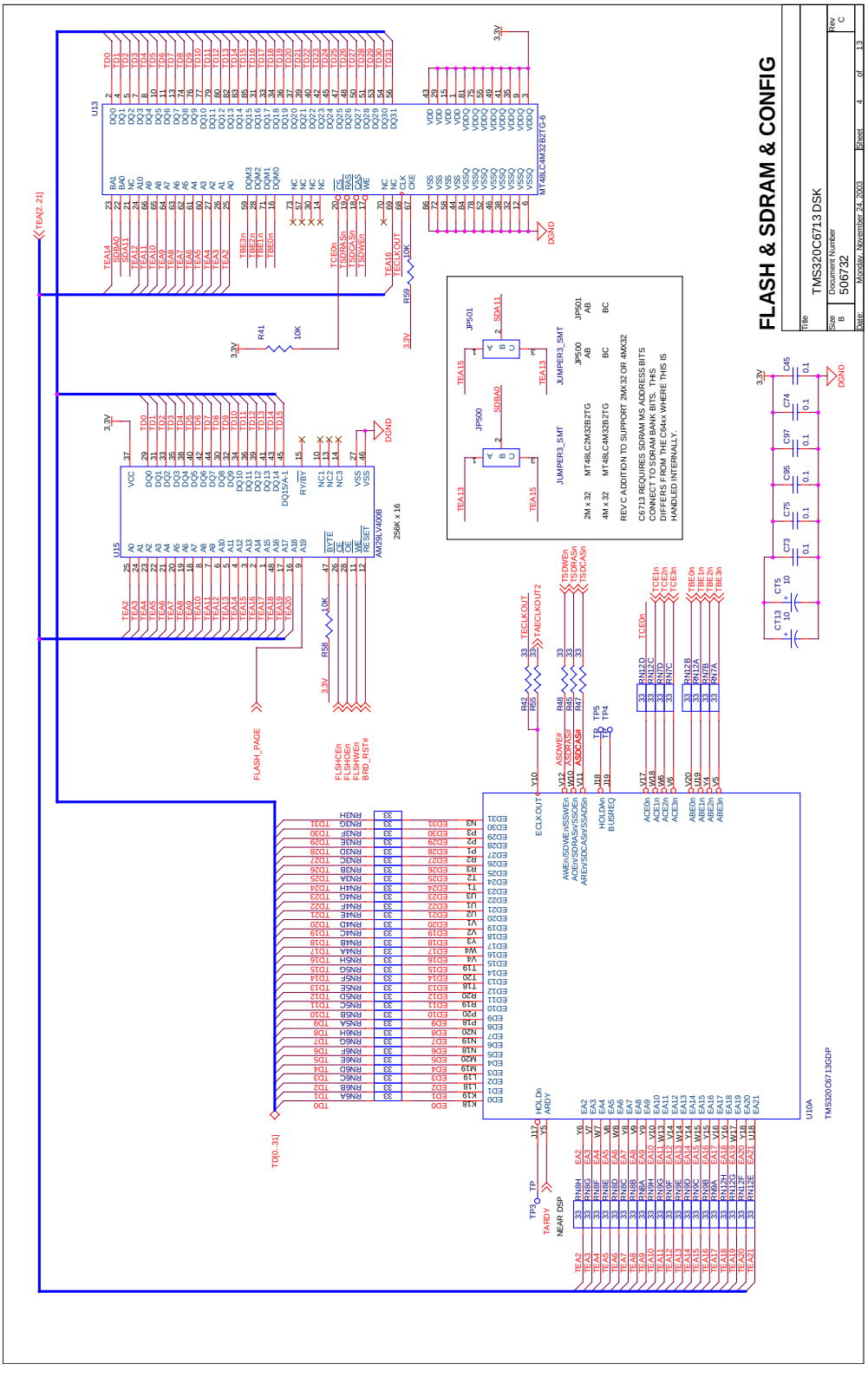
E.1. Schaltpläne

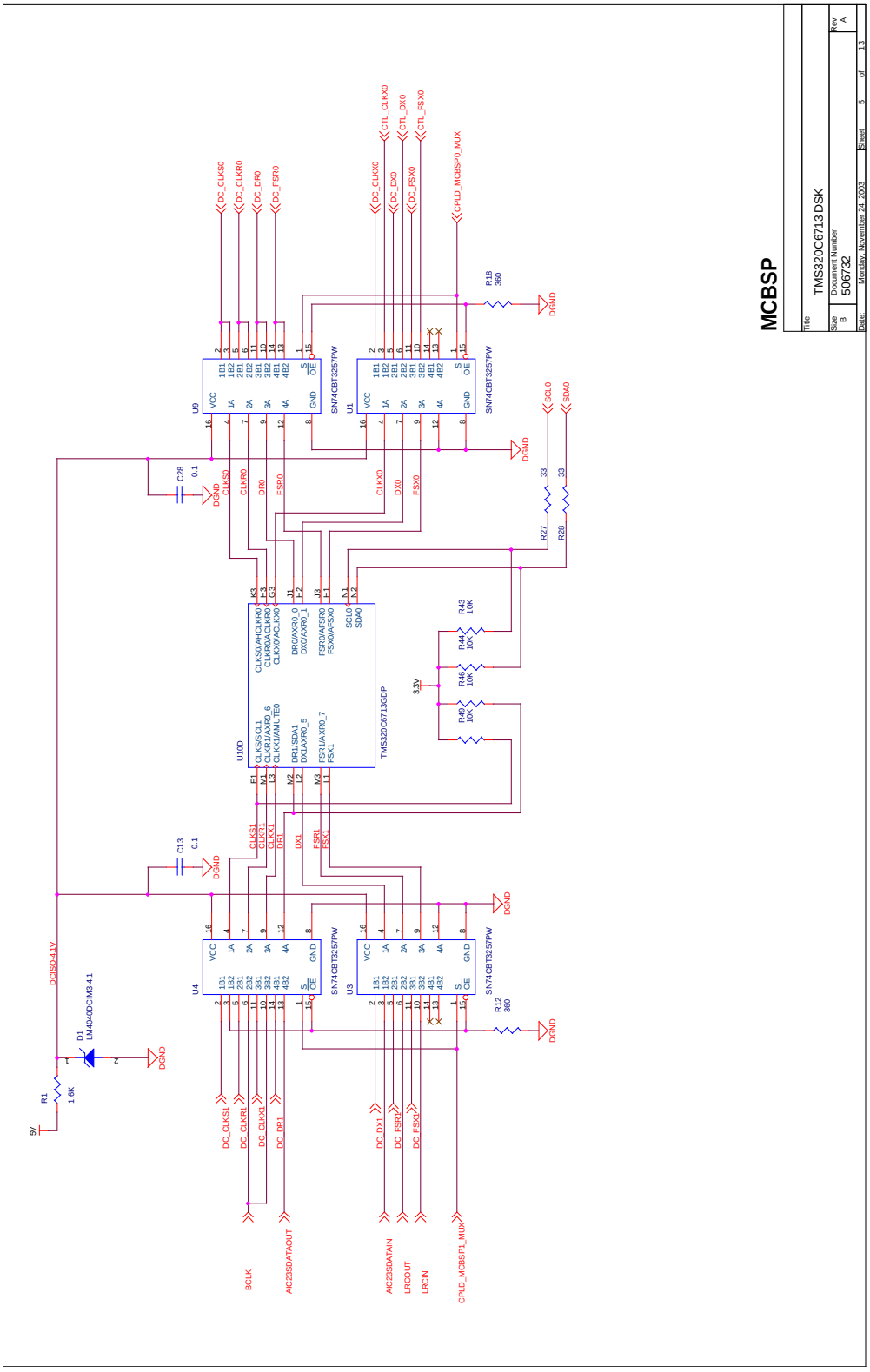
E.1.1. TMS320C6713 DSK

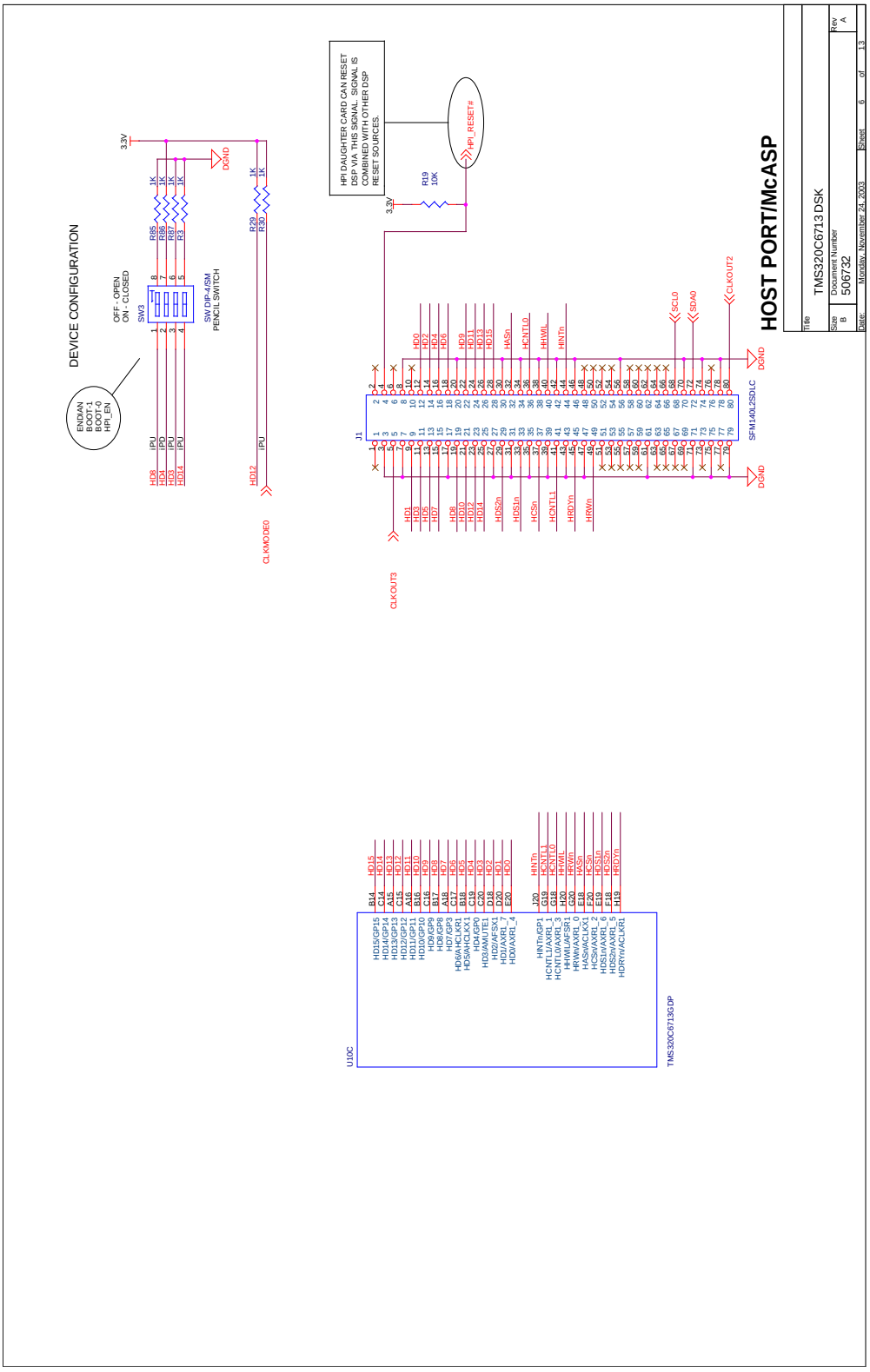


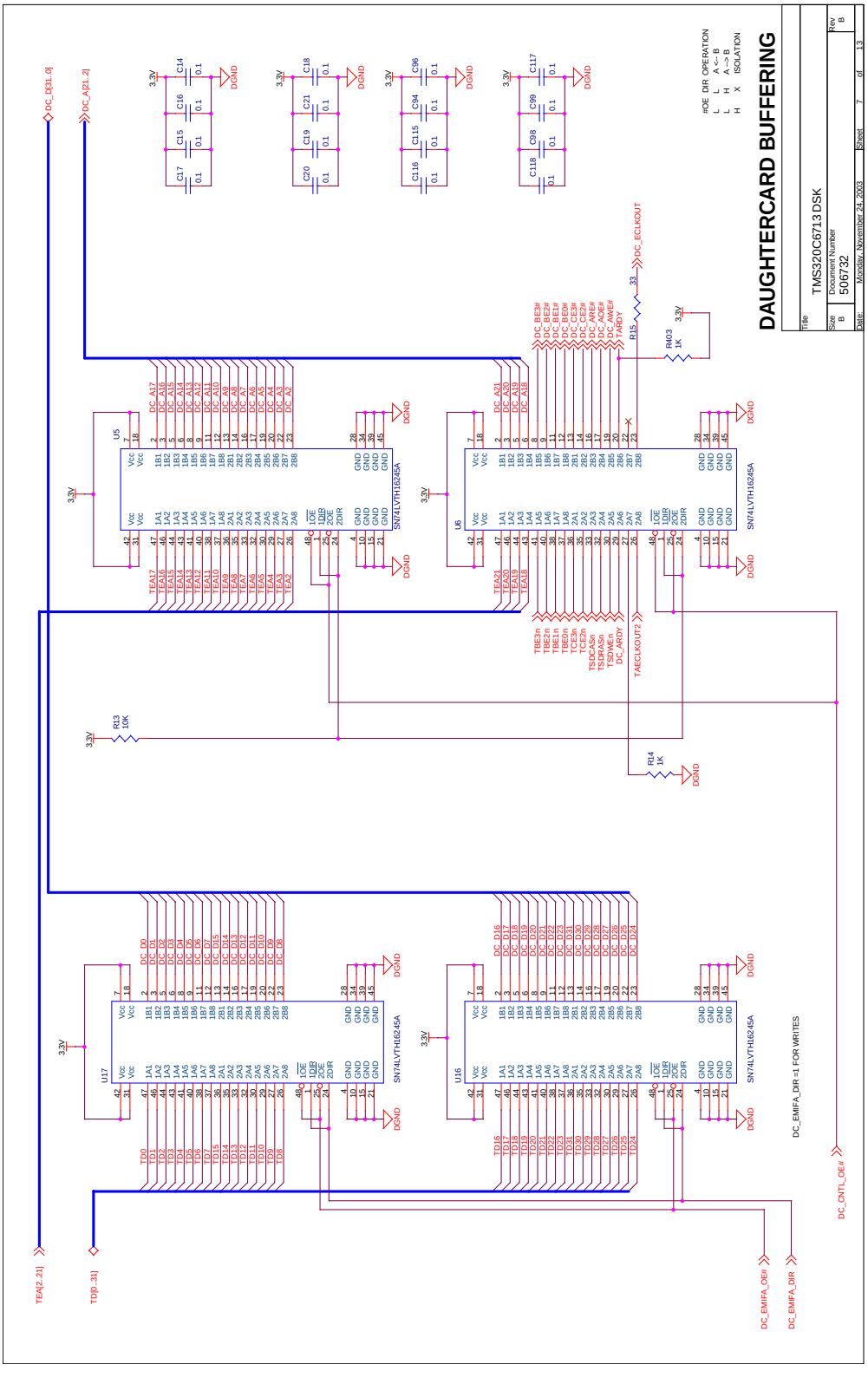

CPLD
TMS320C6713 DSK
Document Number
506732
Monday, November 24, 2003
PULL UP/DOWN TO KEEP LOGIC IN RESET
WHEN THE CPLD IS NOT PROGRAMMED
EPM3128ATC100-10
SN74AHC1G14
SW DIP-4/SM
PADDLE SWITCH
HEADER 5X2
PUSHBUTTON
MMBD4148
YELLOW
GREEN

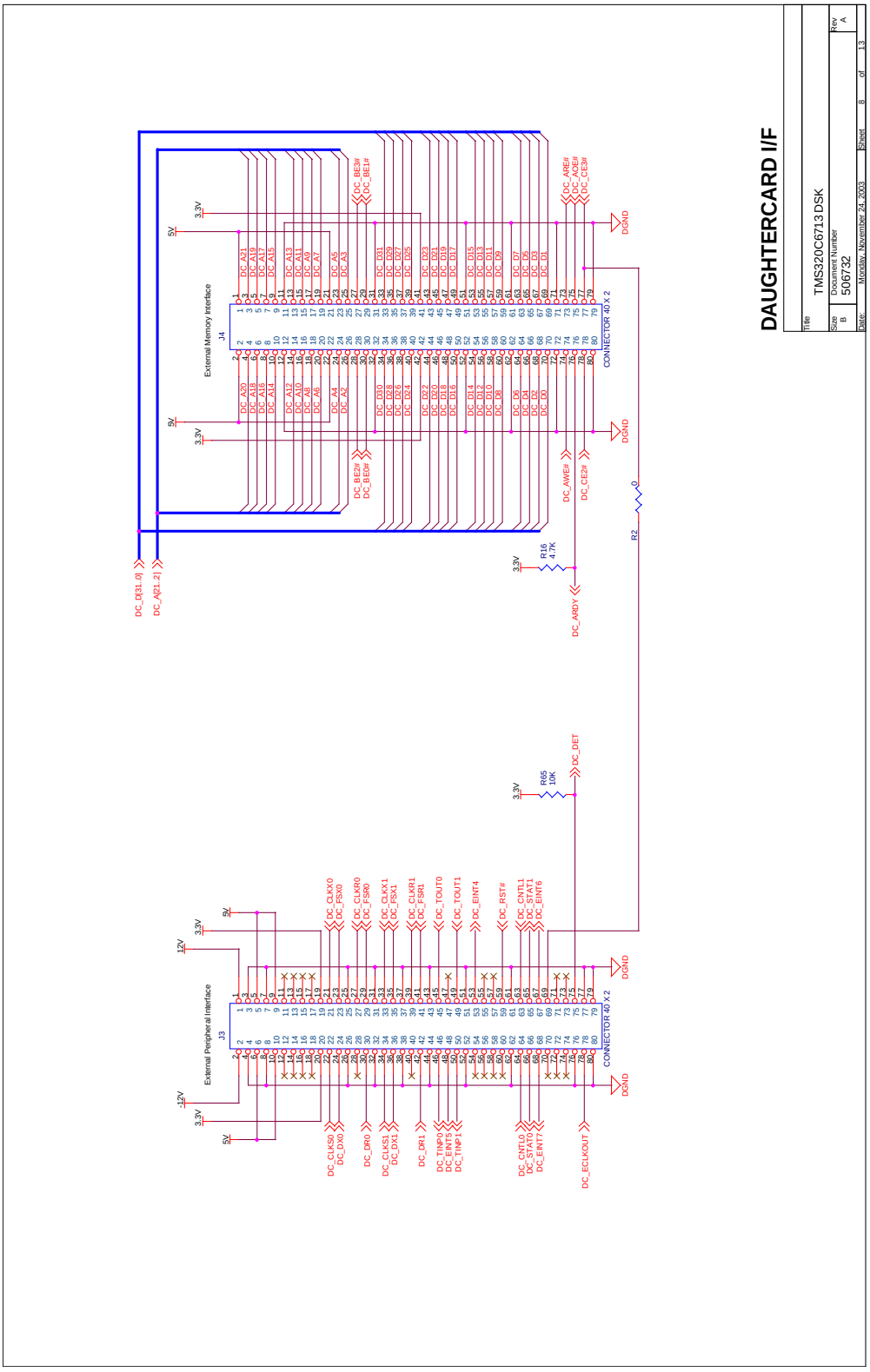

TMS320C6713 DSK
Document Number
506732
Monday, November 24, 2003
Sheet 3 of 13
Rev A
TMS320C6713DSP
SN74CBTD3384PW
EXCELSTJ03U
EMI FILTER
Fertite Chip
50 MHz
XDS_4.1V
DGND
DSPIO_3.3V
CLKMODE0
DSP_CORE_CLK
DSP_RST#
DSP_TDO
DSP_TDI
DSP_TMS
DSP_TCK
DSP_TRST#
DSP_EMU0
DSP_EMU1
DSP_EMU2
DSP_EMU3
DSP_EMU4
DSP_EMU5
CLKOUT2
CLKOUT3
Maximize the distance between switching signals and the PLL external components.
Place all PLL external components as close to the DSP_A/PLL as possible. All components must be on a single side of the board.

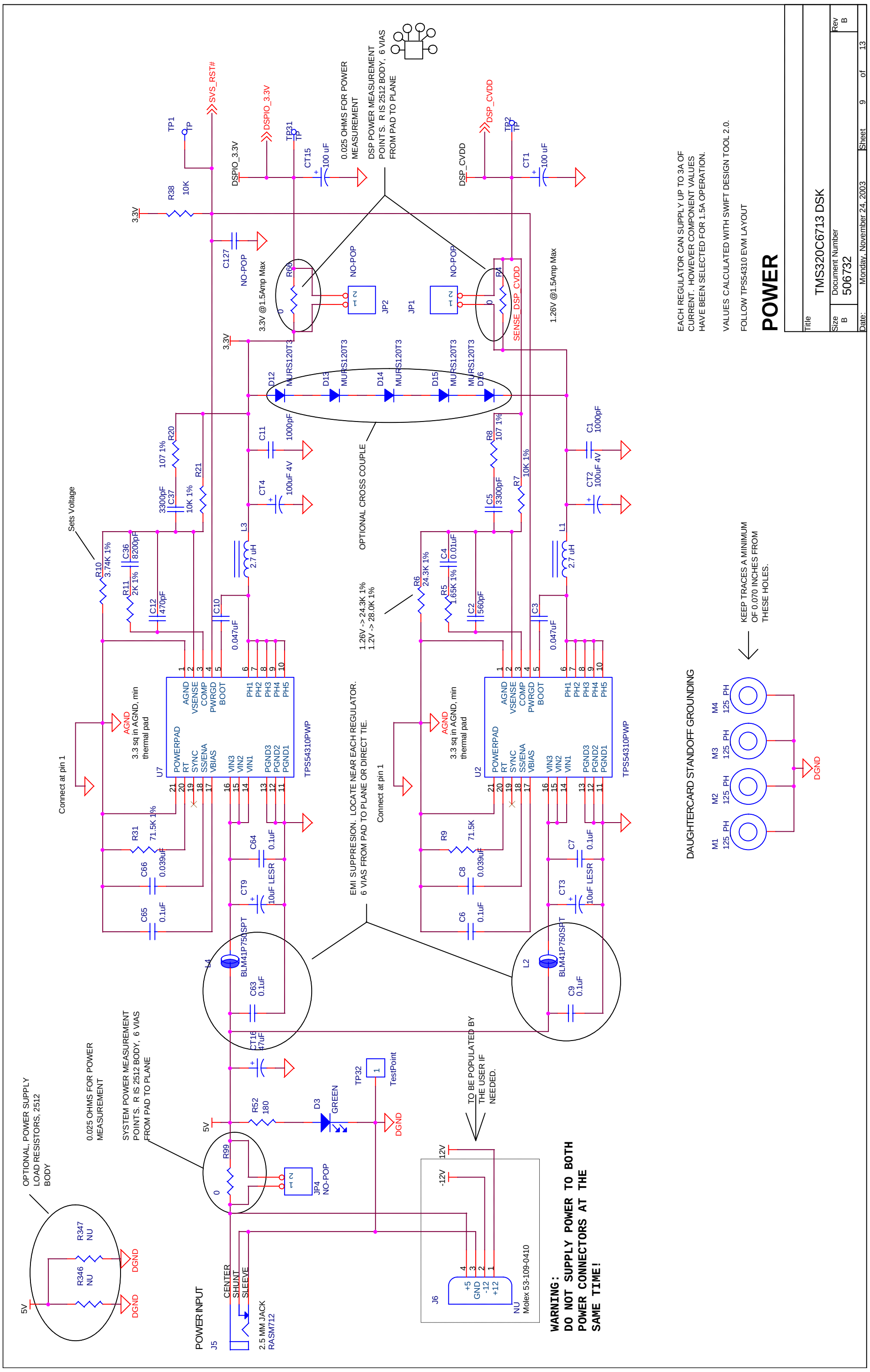

FLASH & SDRAM & CONFIG
TMS320C6713 DSK
Document Number
506732
Monday, November 24, 2003
U13
MT48LC4M32B2TG-6
U15
AM29LV400B
256K x 16
U10A
TMS320C6713GDP
JP501
JP500
JUMPER3_SMT
MT48LC2M32B2TG
MT48LC4M32B2TG
REV C ADDITION TO SUPPORT 2MX32 OR 4MX32
C6713 REQUIRES SDRAM MS ADDRESS BITS
CONNECT TO SDRAM BANK BITS. THIS
DIFFERS FROM THE C64xx WHERE THIS IS
HANDLED INTERNALLY.
FLASH_PAGE
NEAR DSP
TARDY
3.3V
DGND

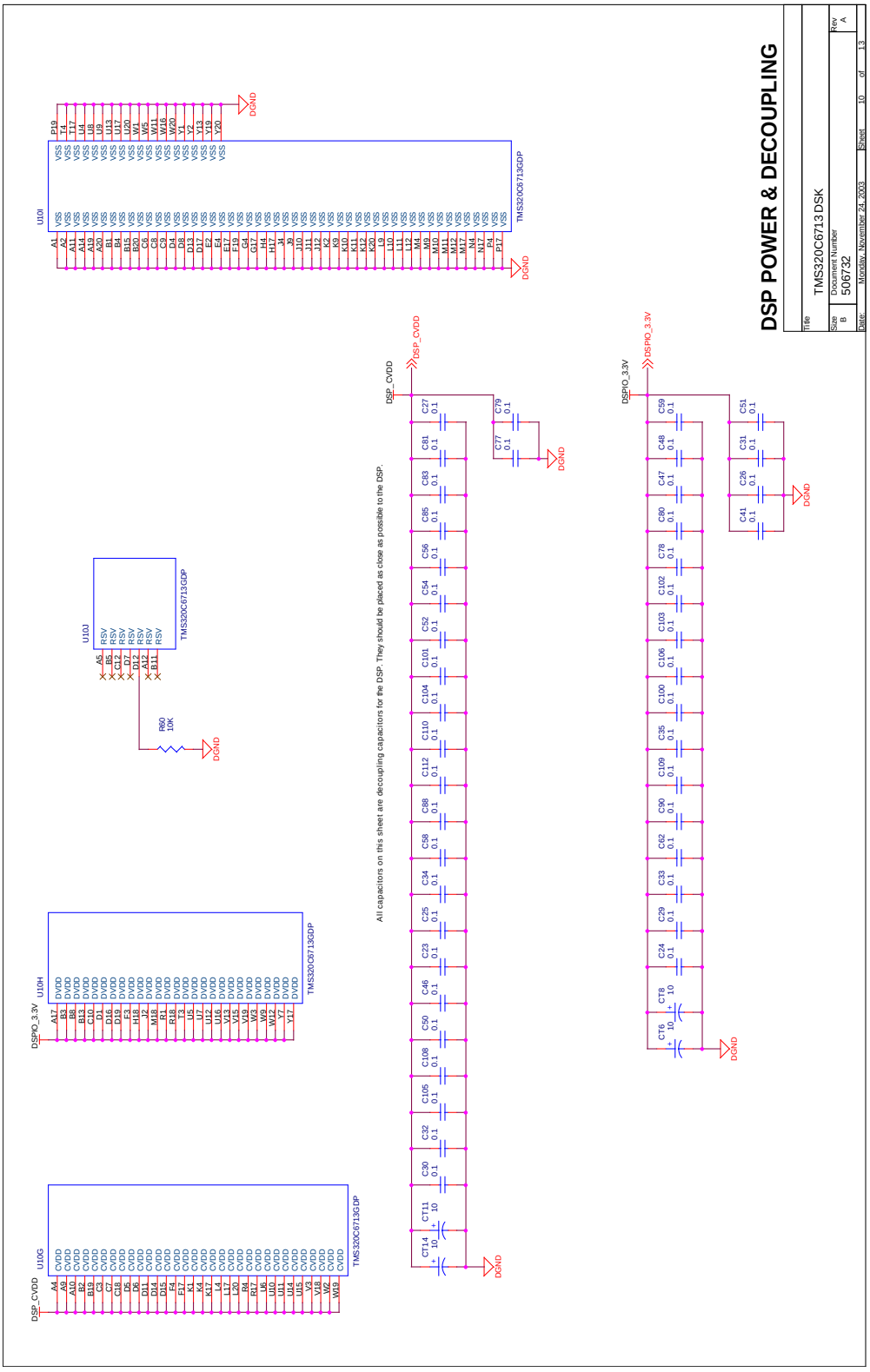

MCBSP
TMS320C6713 DSK
Document Number
506732
Monday, November 24, 2003
SN74CBT3257PW
TMS320C6713GDP
DGND
3.3V
5V
DC_CLKS0
DC_CLKR0
DC_DR0
DC_FSR0
CTL_CLKX0
CTL_DX0
CTL_FSX0
CPLD_MCBSP0_MUX
SCL0
SDA0
DC_CLKS1
DC_CLKR1
DC_CLKX1
DC_DR1
DC_DX1
DC_FSR1
DC_FSX1
CPLD_MCBSP1_MUX
BCLK
AIC23SDATAOUT
AIC23SDATAIN
LRCOUT
LRCIN

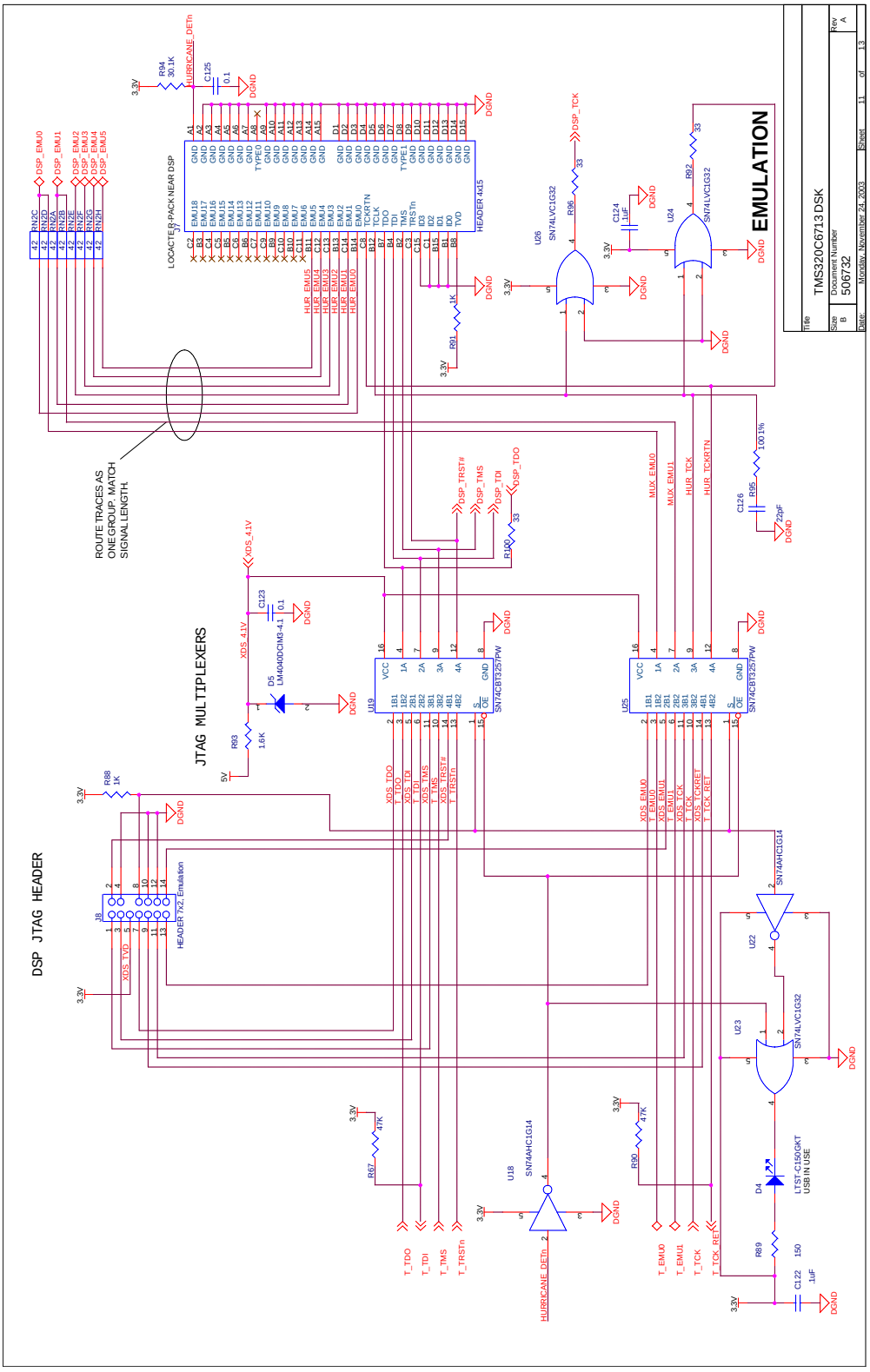

DEVICE CONFIGURATION
ENDIAN
BOOT 1
BOOT 0
HPI_EN
OFF: OPEN
ON: CLOSED
SW3
SW DIP-4/SM
PENCIL SWITCH
3.3V
DGND
1K
R85
R86
R87
R3
R29
R30
IPU
HD8
HD4
HD3
HD14
HD12
CLKMODE0
HPI DAUGHTER CARD CAN RESET
DSP VIA THIS SIGNAL. SIGNAL IS
COMBINED WITH OTHER DSP
RESET SOURCES.
R19
10K
HPI_RESET#
J1
SFM140L2SDLC
HD0
HD1
HD2
HD3
HD4
HD5
HD6
HD7
HD8
HD9
HD10
HD11
HD12
HD13
HD14
HD15
HDS2n
HDS1n
HCSn
HCNTL1
HCNTL0
HRDYn
HRWn
HASn
HHWIL
HHINTn
SCL0
SDA0
CLKOUT2
CLKOUT3
U10C
TMS320C6713GDP
HOST PORT/McASP
TMS320C6713 DSK
Rev
A
Size
B
Document Number
506732
Date:
Monday, November 24, 2003
Sheet
6
of
13

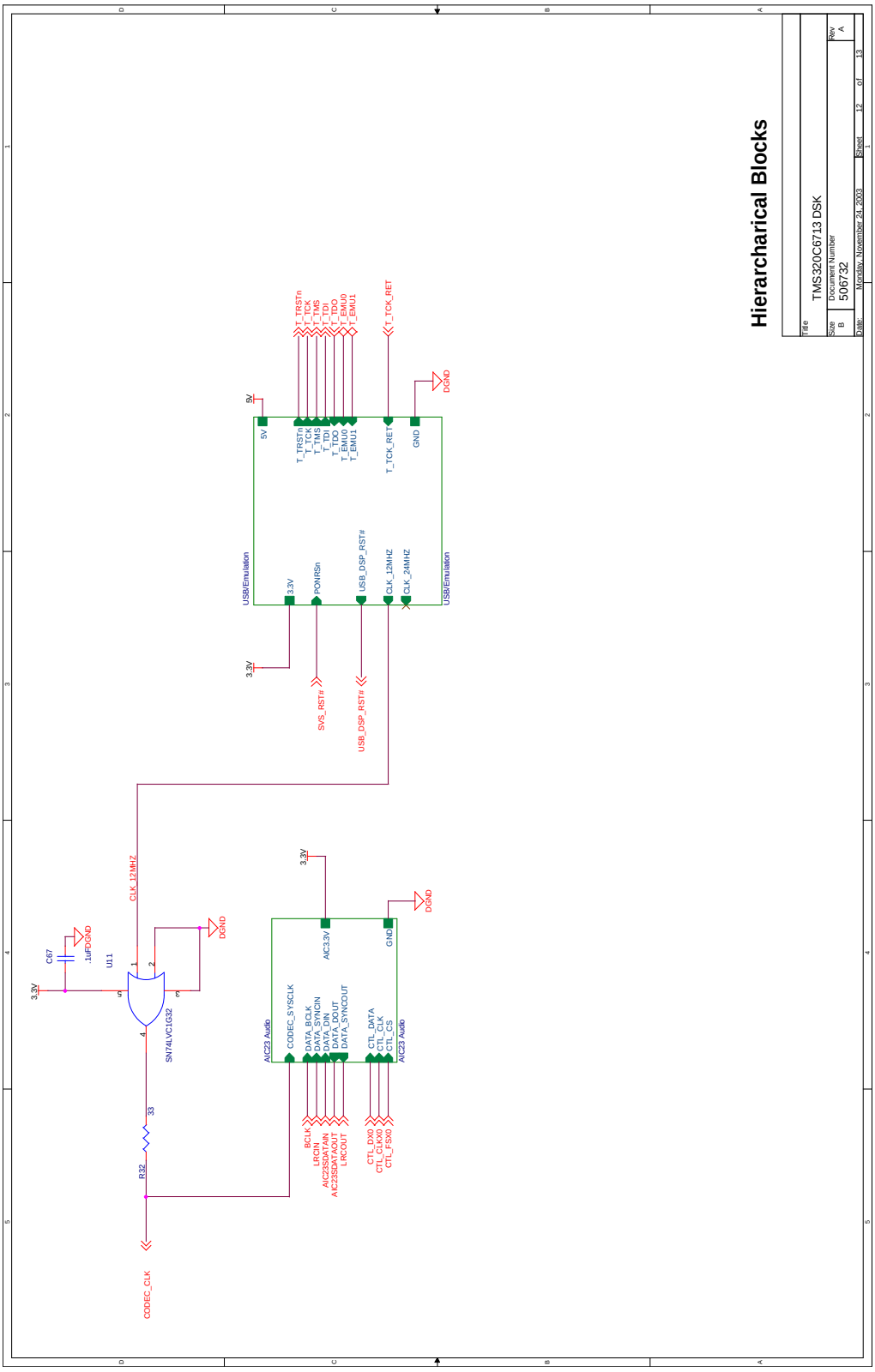

DAUGHTERCARD BUFFERING
TMS320C6713 DSK
Document Number
506732
Monday, November 24, 2003
SN74LVTH16245A
DC_EMIFA_DIR =1 FOR WRITES


DAUGHTERCARD I/F
TMS320C6713 DSK
Document Number
506732
Monday, November 24, 2003
External Memory Interface
CONNECTOR 40 X 2
External Peripheral Interface
CONNECTOR 40 X 2
R2
R16 4.7K
R66 10K
DGND
3.3V
5V
9V
12V

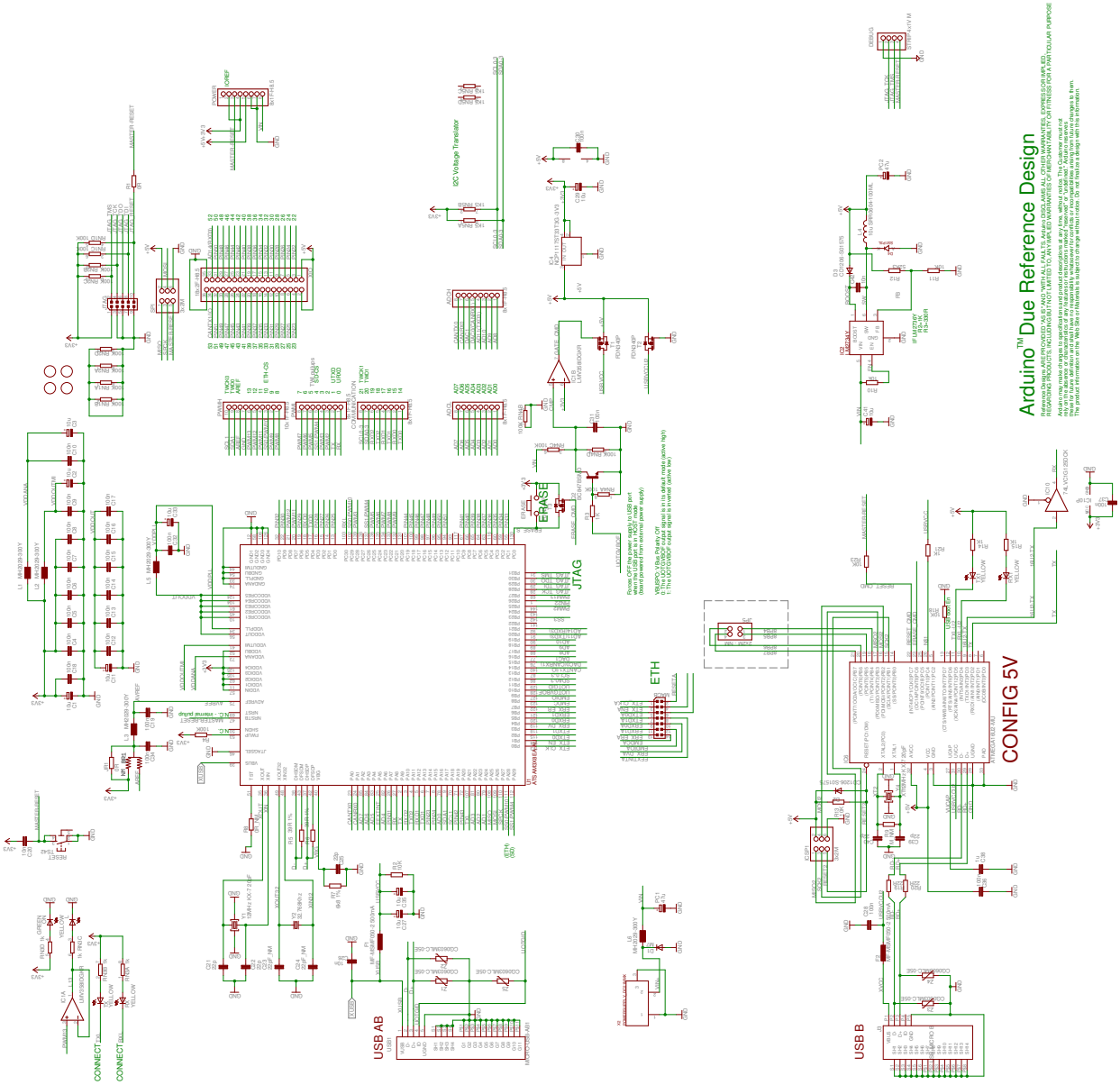

POWER
TMS320C6713 DSK
506732
Monday, November 24, 2003
EACH REGULATOR CAN SUPPLY UP TO 3A OF CURRENT. HOWEVER COMPONENT VALUES HAVE BEEN SELECTED FOR 1.5A OPERATION.
VALUES CALCULATED WITH SWIFT DESIGN TOOL 2.0.
FOLLOW TPS54310 EVM LAYOUT
DAUGHTERCARD STANDOFF GROUNDING
KEEP TRACES A MINIMUM OF 0.070 INCHES FROM THESE HOLES.
WARNING: DO NOT SUPPLY POWER TO BOTH POWER CONNECTORS AT THE SAME TIME!
EMI SUPPRESION. LOCATE NEAR EACH REGULATOR.
6 VIAS FROM PAD TO PLANE OR DIRECT TIE.
OPTIONAL CROSS COUPLE
0.025 OHMS FOR POWER MEASUREMENT
DSP POWER MEASUREMENT POINTS. R IS 2512 BODY, 6 VIAS FROM PAD TO PLANE
SYSTEM POWER MEASUREMENT POINTS. R IS 2512 BODY, 6 VIAS FROM PAD TO PLANE
OPTIONAL POWER SUPPLY LOAD RESISTORS, 2512 BODY
TO BE POPULATED BY THE USER IF NEEDED.
POWER INPUT
TPS54310PWP
Sets Voltage
Connect at pin 1
1.26V -> 24.3K 1%
1.2V -> 28.0K 1%
3.3V @1.5Amp Max
1.26V @1.5Amp Max
Molex 53-109-0410
2.5 MM JACK RASM712


DSP POWER & DECOUPLING
TMS320C6713 DSK
Document Number
506732
Monday, November 24, 2003
All capacitors on this sheet are decoupling capacitors for the DSP. They should be placed as close as possible to the DSP.
TMS320C6713GDP
DSP_CVDD
DSPIO_3.3V
DGND

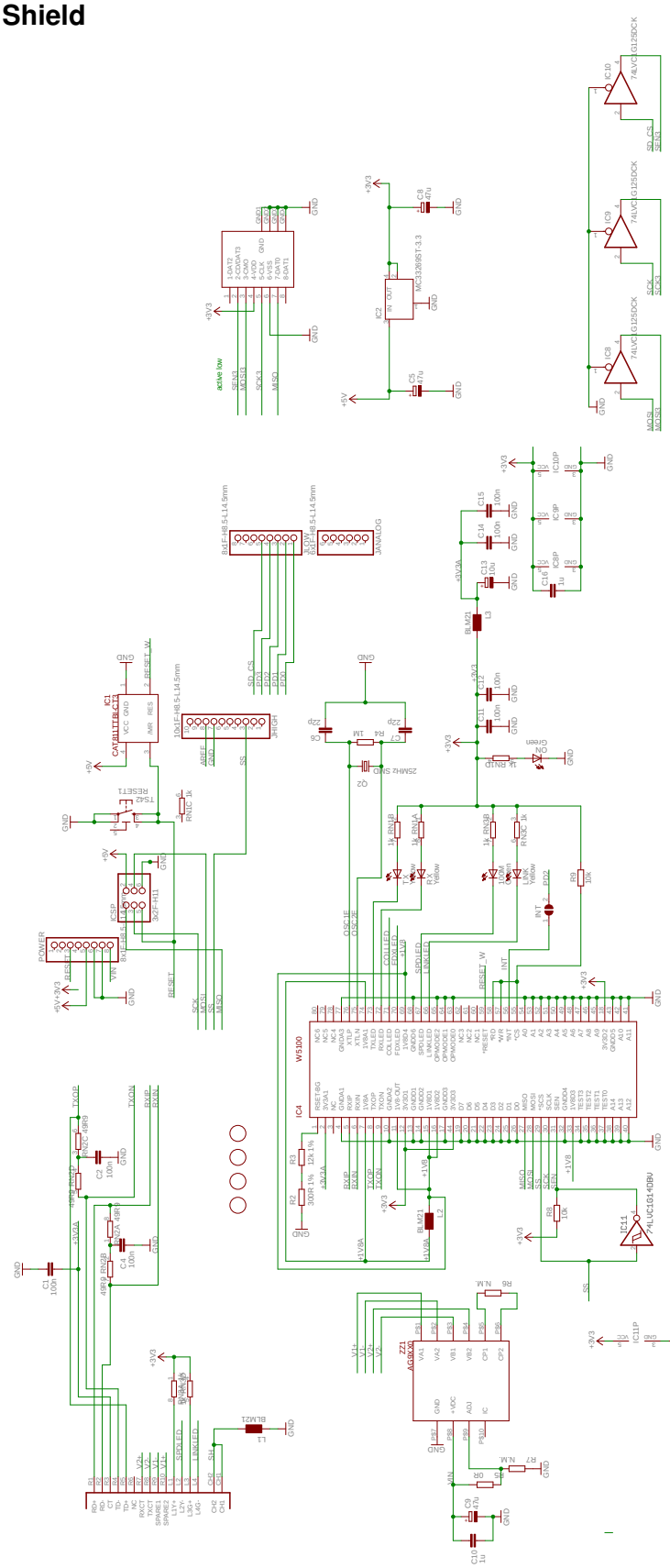

EMULATION
TMS320C6713 DSK
Document Number
506732
Monday, November 24, 2003
JTAG MULTIPLEXERS
DSP JTAG HEADER
ROUTE TRACES AS
ONE GROUP, MATCH
SIGNAL LENGTH.
LOCACTER-PACK NEAR DSP
HEADER 4x15
HEADER 7x2, Emulation
SN74LVC1G32
SN74AHC1G14
SN74CBT3257PW
LM4040DCIM3-4.1
LTST-C150GKT
USB IN USE
HURRICANE_DETn
DSP_EMU0
DSP_EMU1
DSP_EMU2
DSP_EMU3
DSP_EMU4
DSP_EMU5
DSP_TCK
DSP_TRST#
DSP_TMS
DSP_TDI
DSP_TDO
XDS_4.1V
MUX_EMU0
MUX_EMU1
HUR_TCK
HUR_TCKRTN
T_TDO
T_TDI
T_TMS
T_TRSTn
T_EMU0
T_EMU1
T_TCK
T_TCK_RET
3.3V
5V
DGND


Hierarcharical Blocks
TMS320C6713 DSK
Document Number
506732
Rev
A
Size
B
Date: Monday, November 24, 2003
Sheet 12 of 13
USBEmulation
T_TRSTn
T_TCK
T_TMS
T_TDI
T_TDO
T_EMU0
T_EMU1
T_TCK_RET
5V
GND
DGND
3.3V
PONRSn
USB_DSP_RST#
CLK_12MHZ
CLK_24MHZ
SYS_RST#
AIC23 Audio
CODEC_SYSCLK
DATA_BCLK
DATA_SYNCIN
DATA_DIN
DATA_DOUT
DATA_SYNCOUT
AIC3.3V
CTL_DATA
CTL_CLK
CTL_CS
BCLK
LRCIN
AIC23SDATAIN
AIC23SDATAOUT
LRCOUT
CTL_DIN
CTL_CLKIN
CTL_SIN
U1.1
SN74LVC1G32
C67
.1uF
R32
33
CODEC_CLK

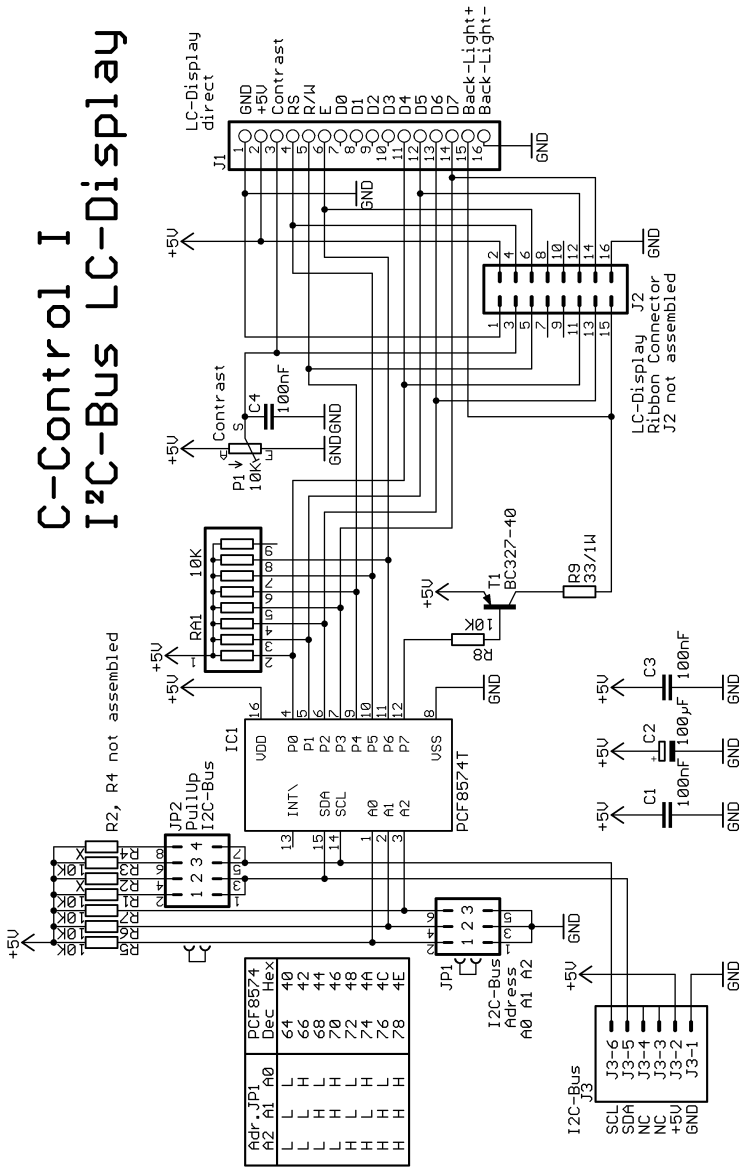
E.1.2. Arduino Due



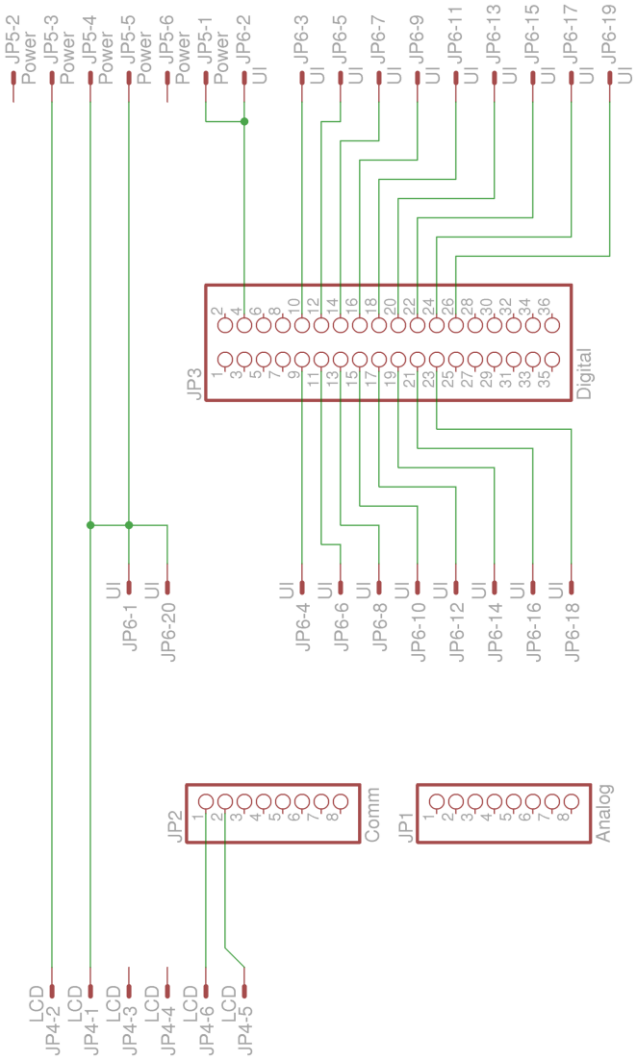
E.1.3. Arduino Ethernet Shield



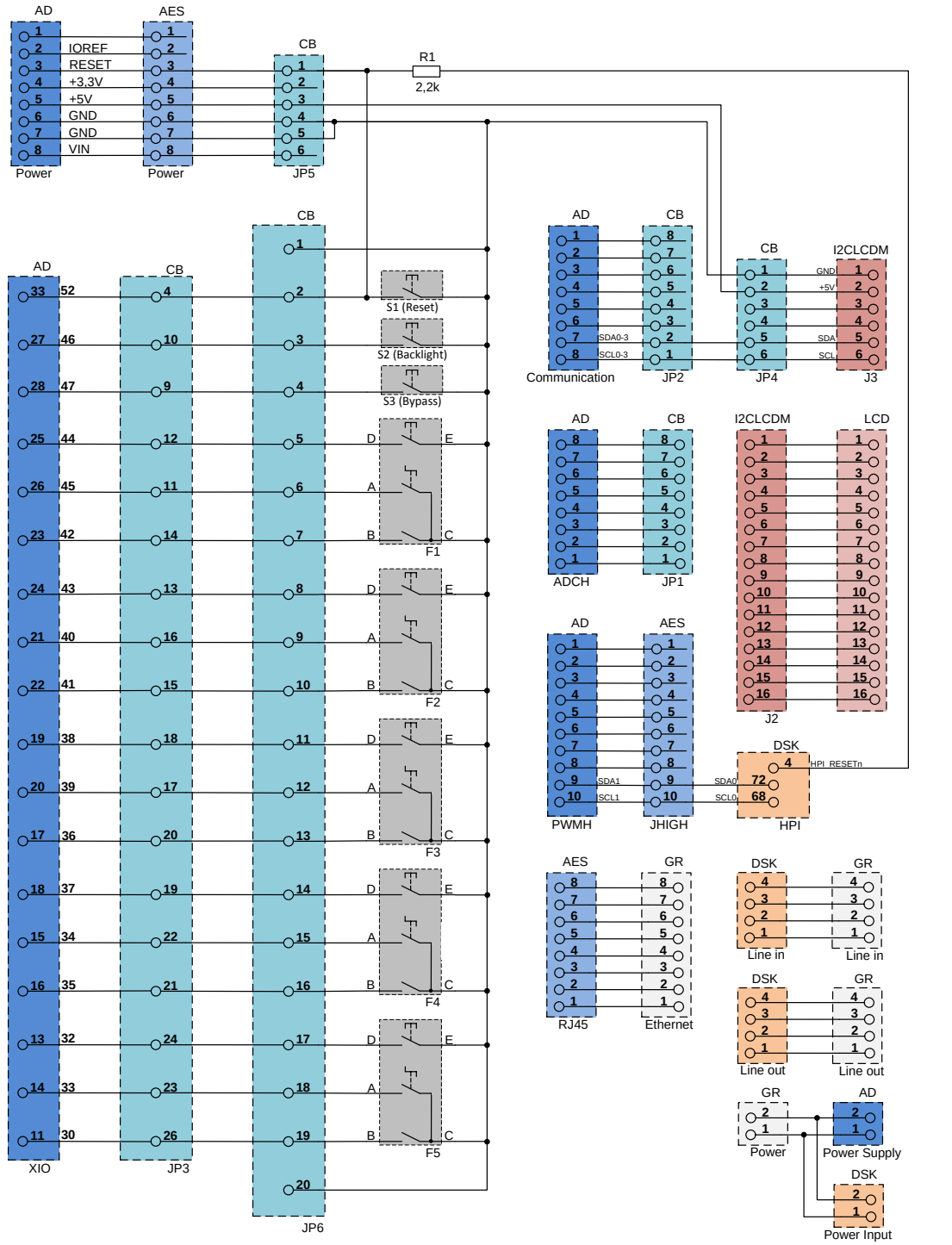
E.1.4. I²C LC-Display Modul



E.1.5. Connector Board

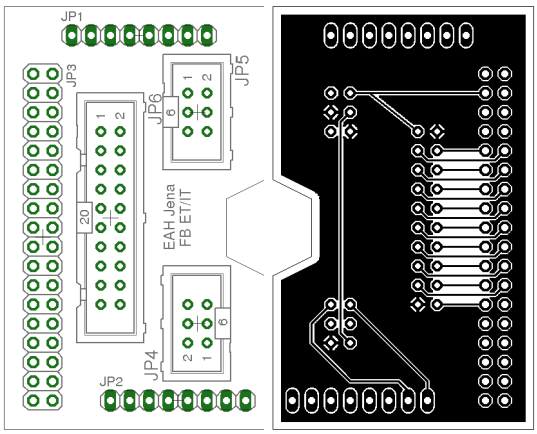


E.1.6. Verbindungsplan der Hardware-Komponenten



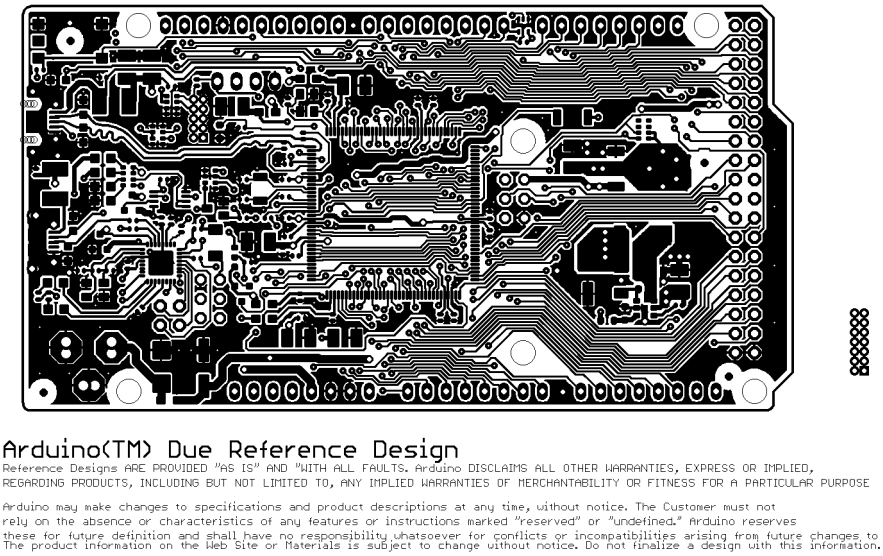
E.2. Platinenlayouts

Bitte beachten Sie das die folgenden Bilder nicht maßstabsgetreu sind. Die Dateien mit den Original-Layouts zum Herstellen weiterer Platinen sind der beigelegten DVD zu entnehmen.

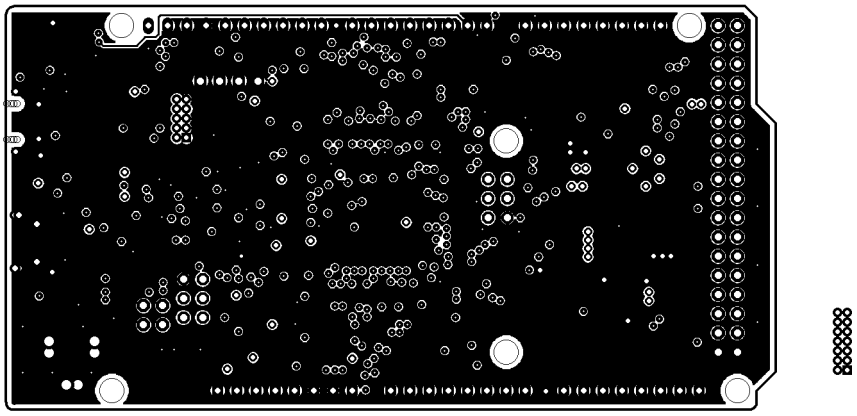


(a) Top (b) Bottom

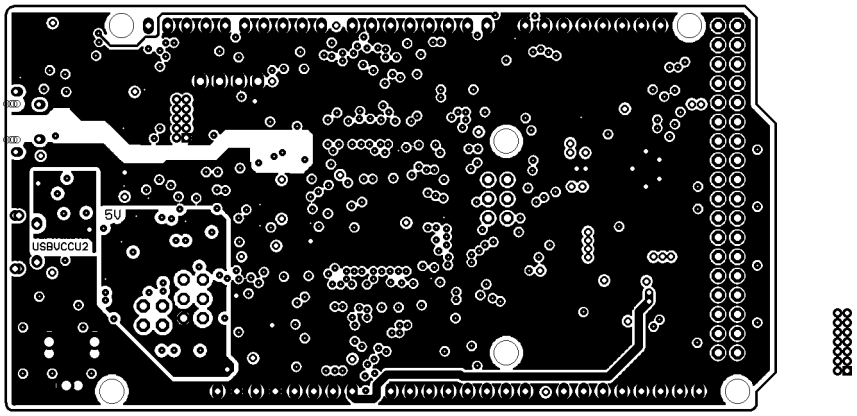
Connector Board



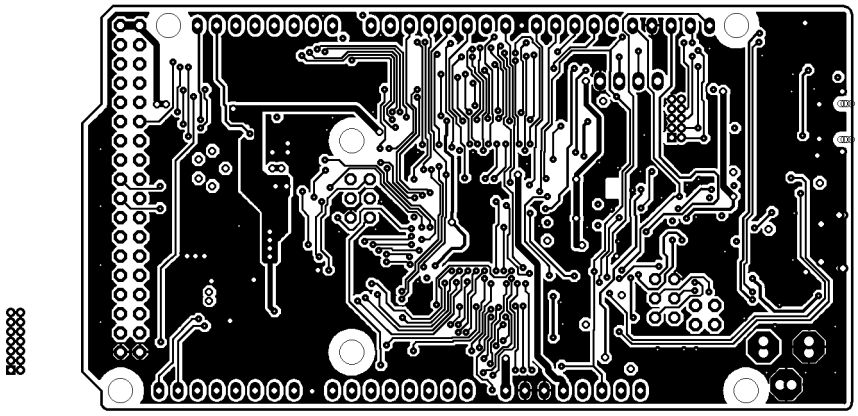
Arduino Due - Top



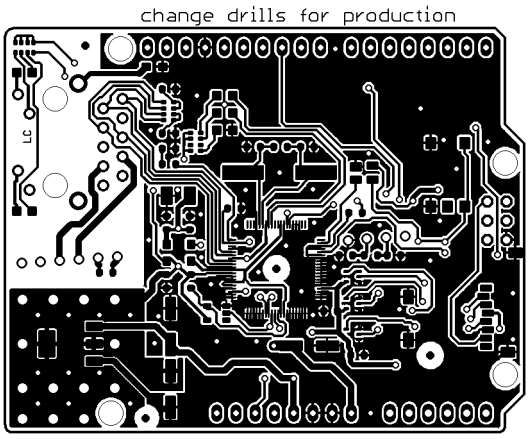
Arduino Due - Layer 2



Arduino Due - Layer 3



Arduino Due - Bottom



ARDUINO ETH SHIELD 06 - Rev 3

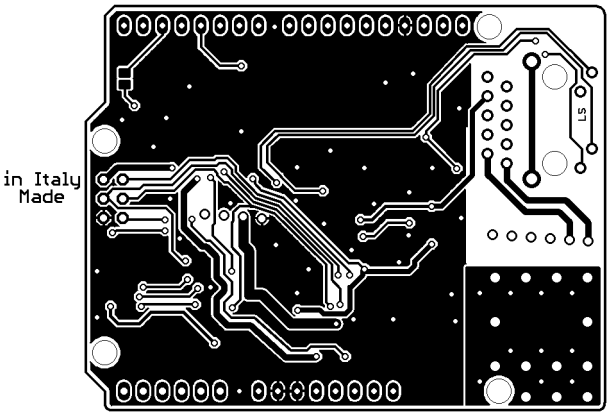
Reference Designs ARE PROVIDED "AS IS" AND "WITH ALL FAULTS. Arduino DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, REGARDING PRODUCTS, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE

Arduino may make changes to specifications and product descriptions at any time, without notice. The Customer must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Arduino reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them.

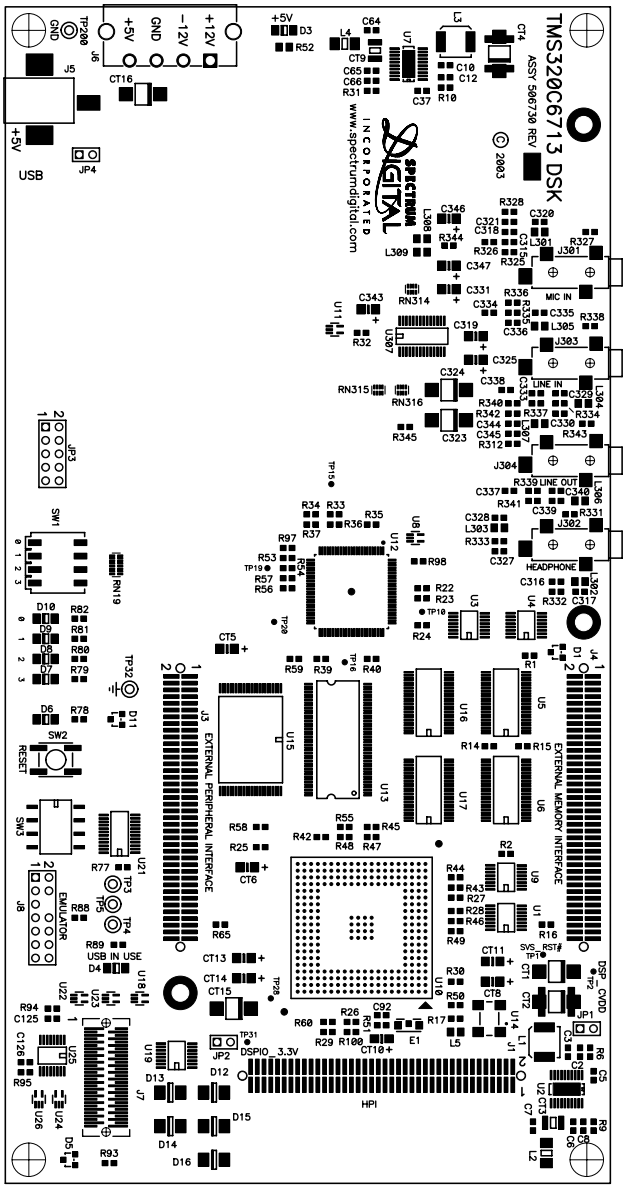
The product information on the Web Site or Materials is subject to change without notice. Do not finalize a design with this information.

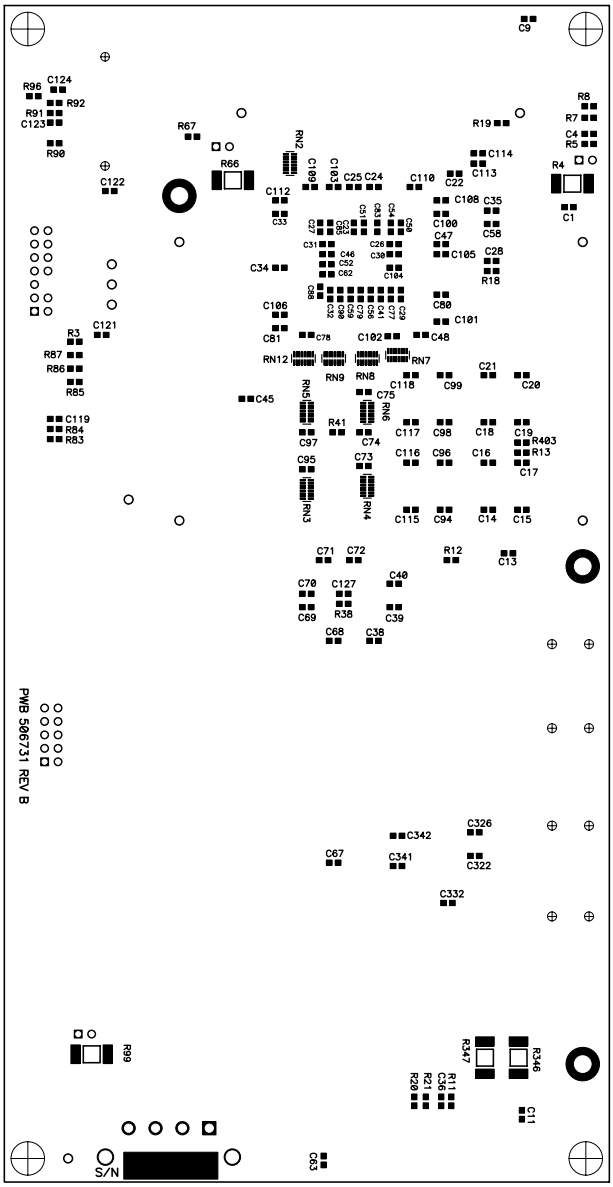
ARDUINO is a registered trademark.

Arduino Ethernet Shield - Top

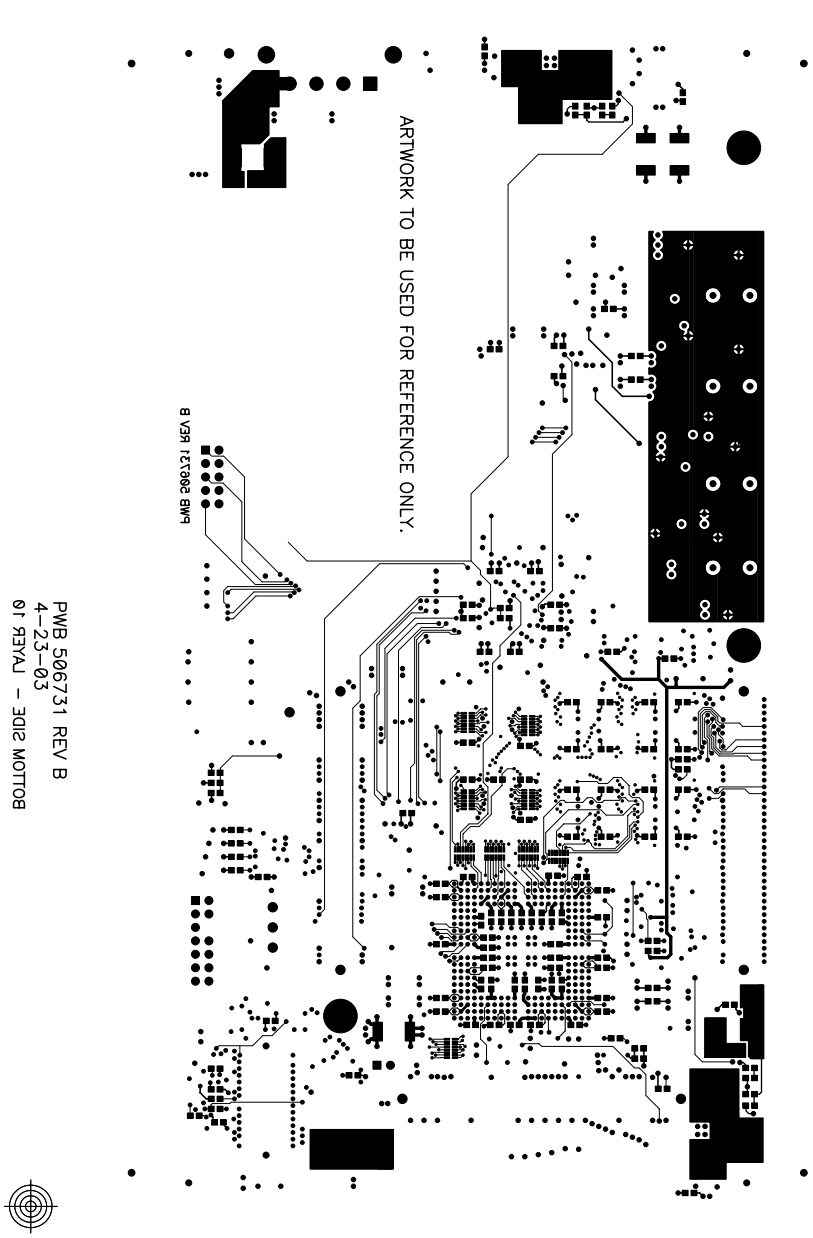


Arduino Ethernet Shield - Bottom

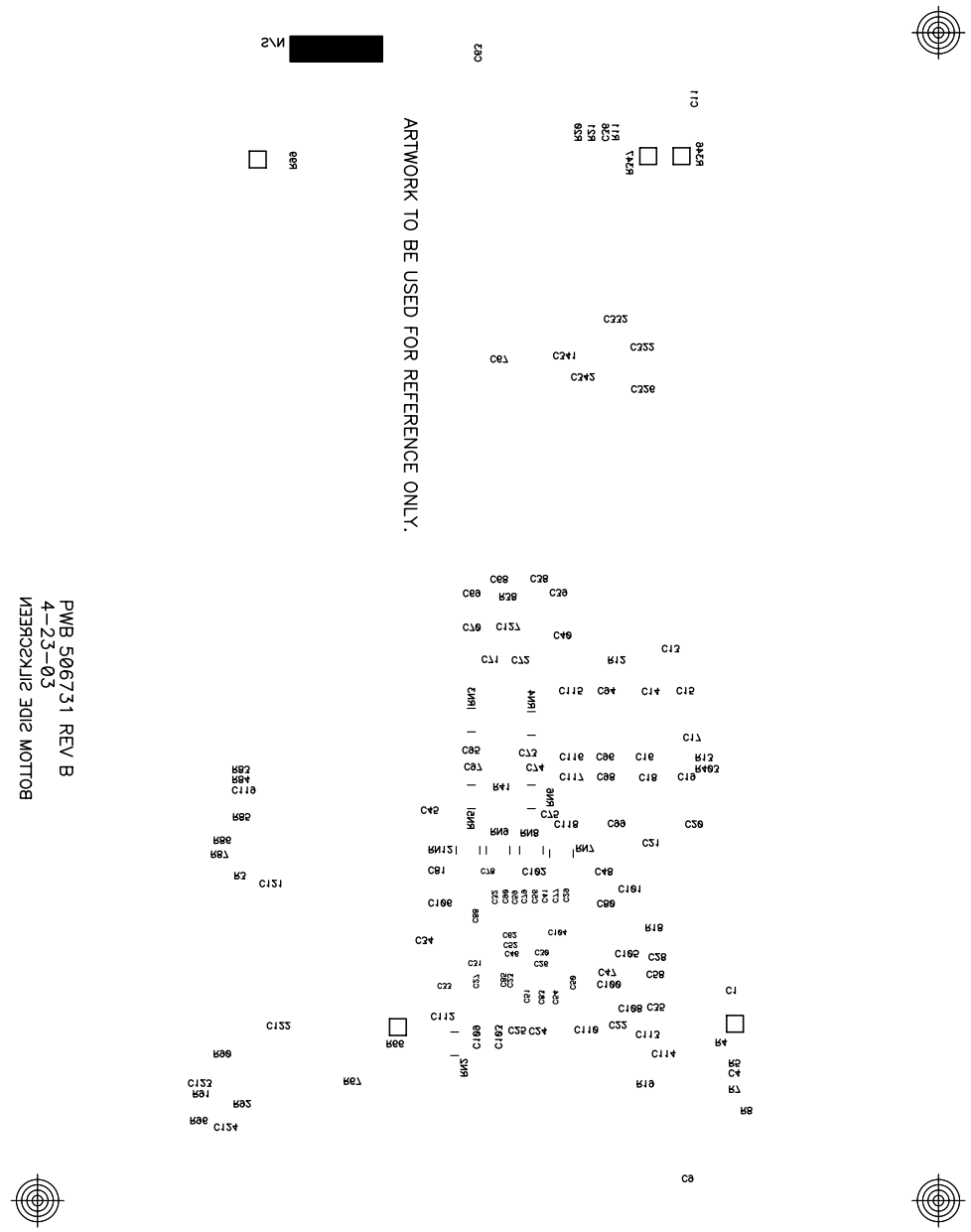

TMS320C6713 DSK
ASSY 506730 REV
© 2003
SPECTRUM DIGITAL INCORPORATED
www.spectrumdigital.com
USB
MIC IN
LINE IN
LINE OUT
HEADPHONE
EXTERNAL PERIPHERAL INTERFACE
EXTERNAL MEMORY INTERFACE
EMULATOR
RESET
USB IN USE
DSPIO_3.3V
HPI
DSP_CVDD


PWB 506731 REV B
S/N

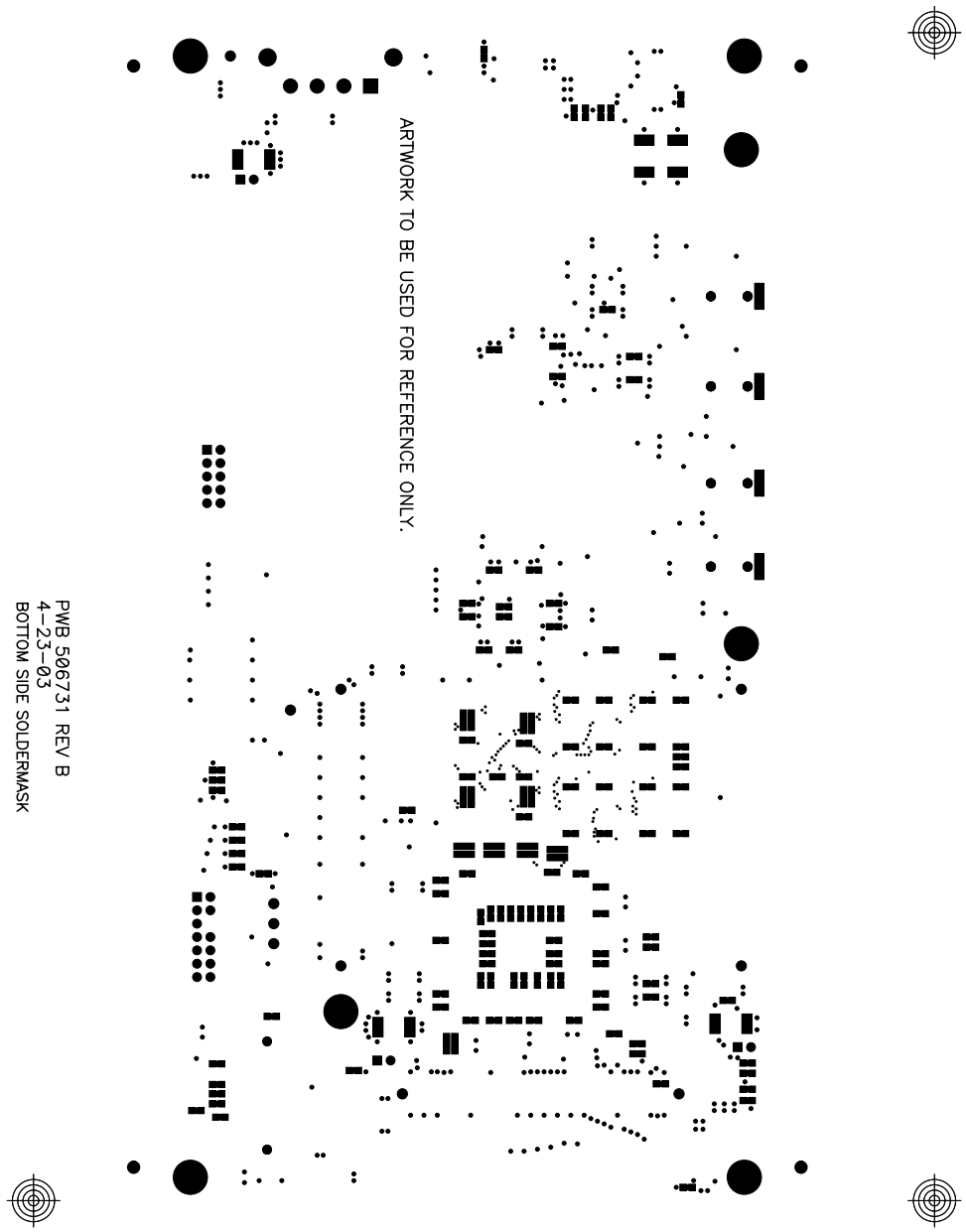
CAM350 V 6.0 : Thu May 08 10:57:54 2003 - (Untitled) : Title



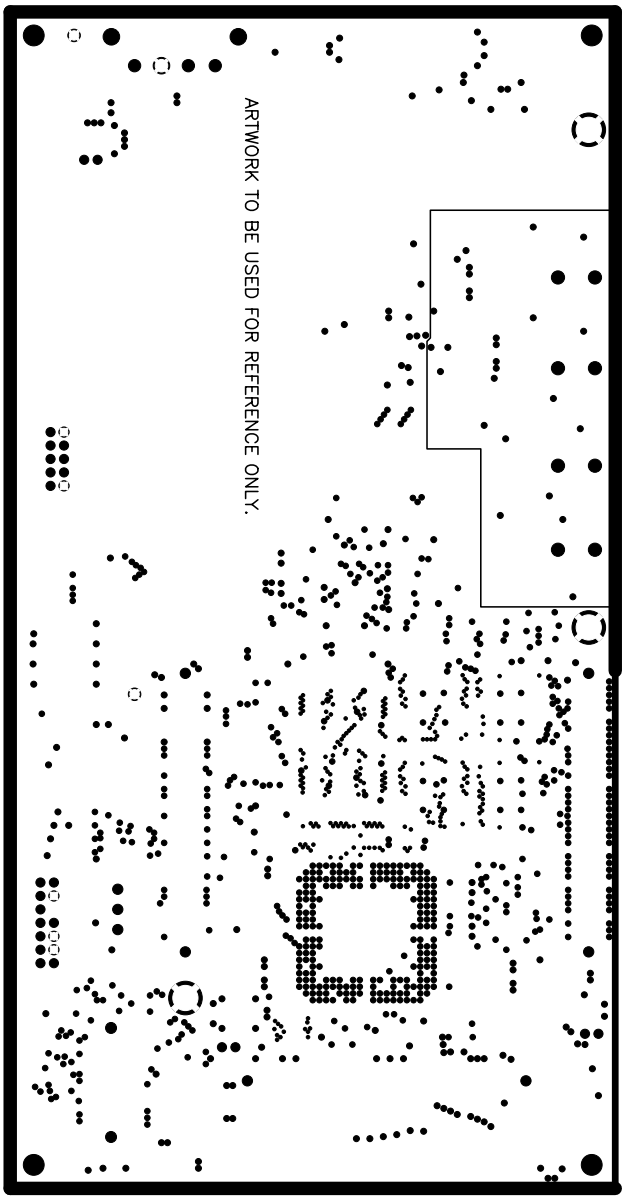
CAM350 V 6.0 : Thu May 08 10:57:54 2003 - (Untitled) : Title



CAM350 V 6.0 : Thu May 08 10:57:54 2003 - (Untitled) : Title

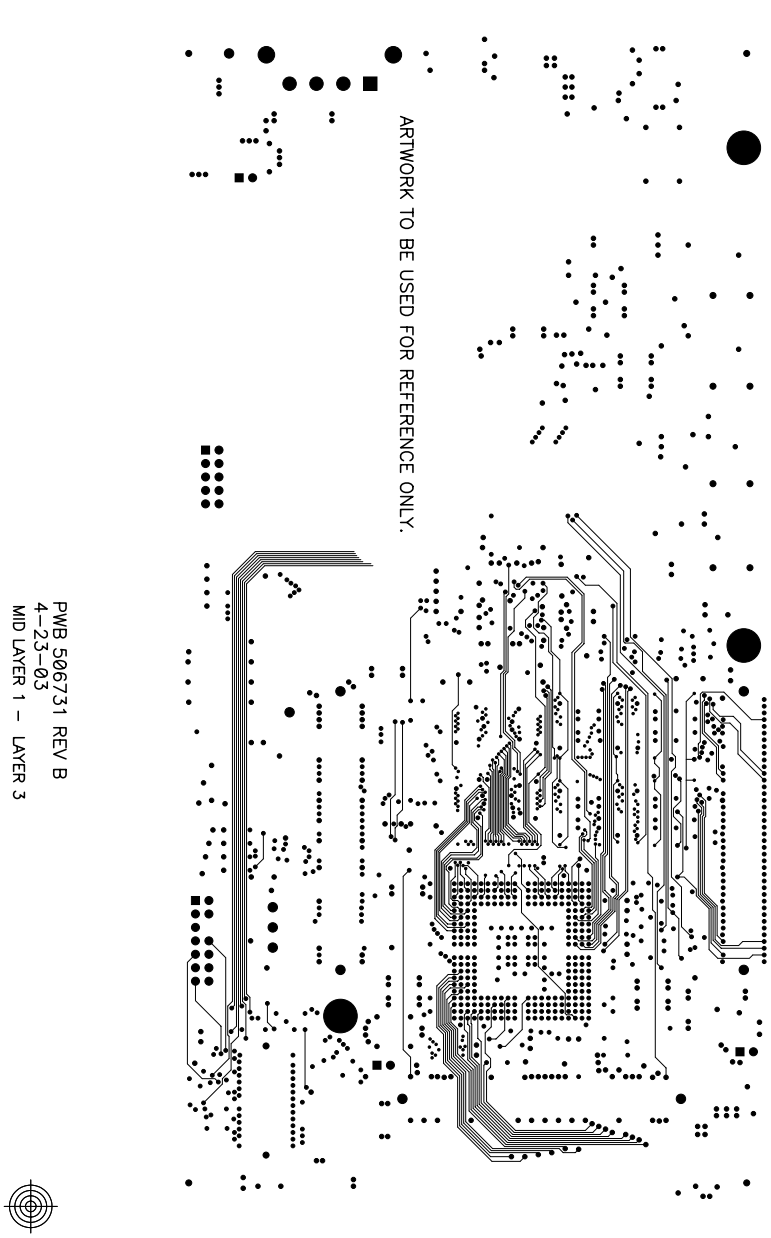


CAM350 V 6.0 : Thu May 08 10:57:55 2003 - (Untitled) : Title

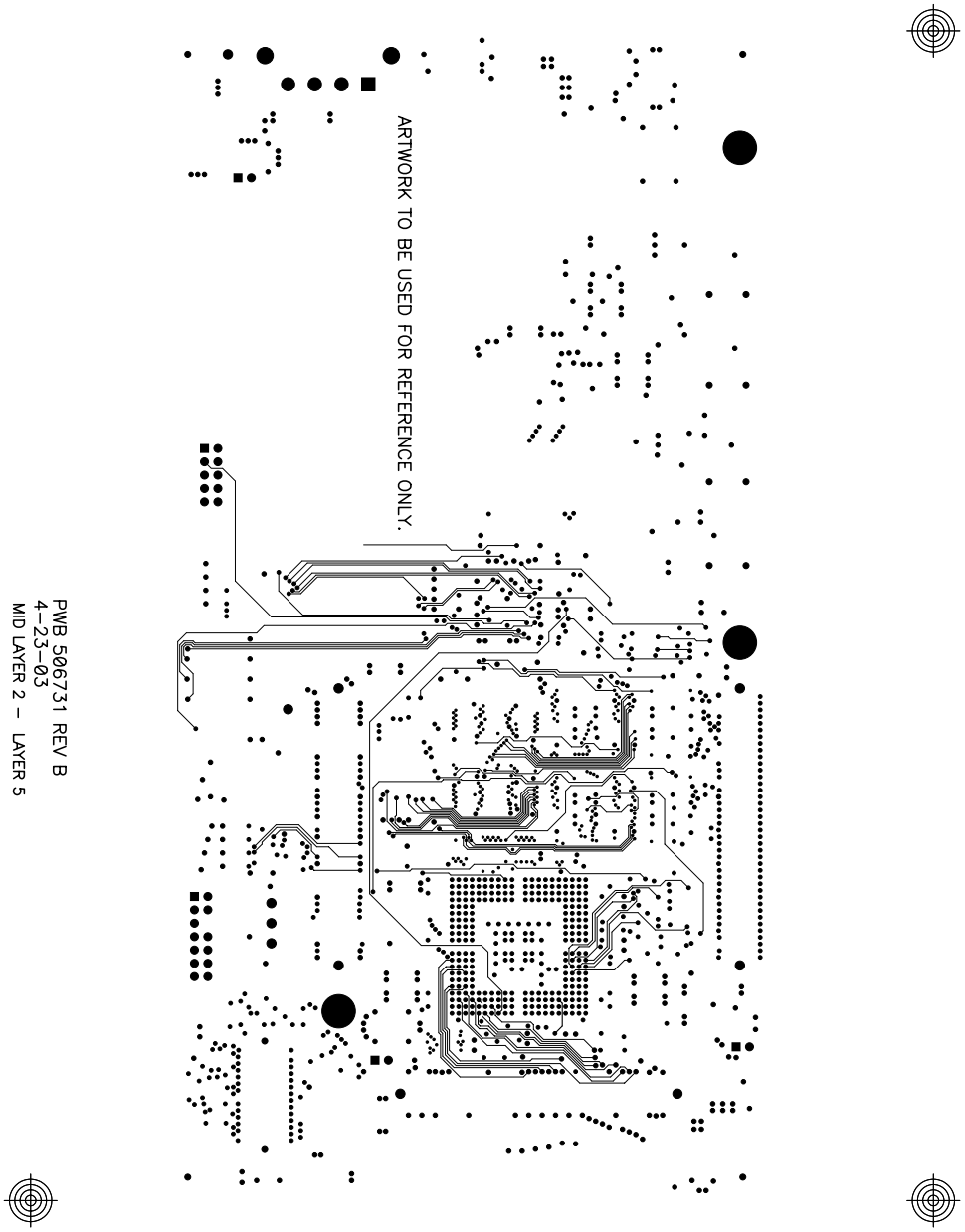


PWB 506731 REV B
4-23-03
GND PLANE - LAYER 2 & 9

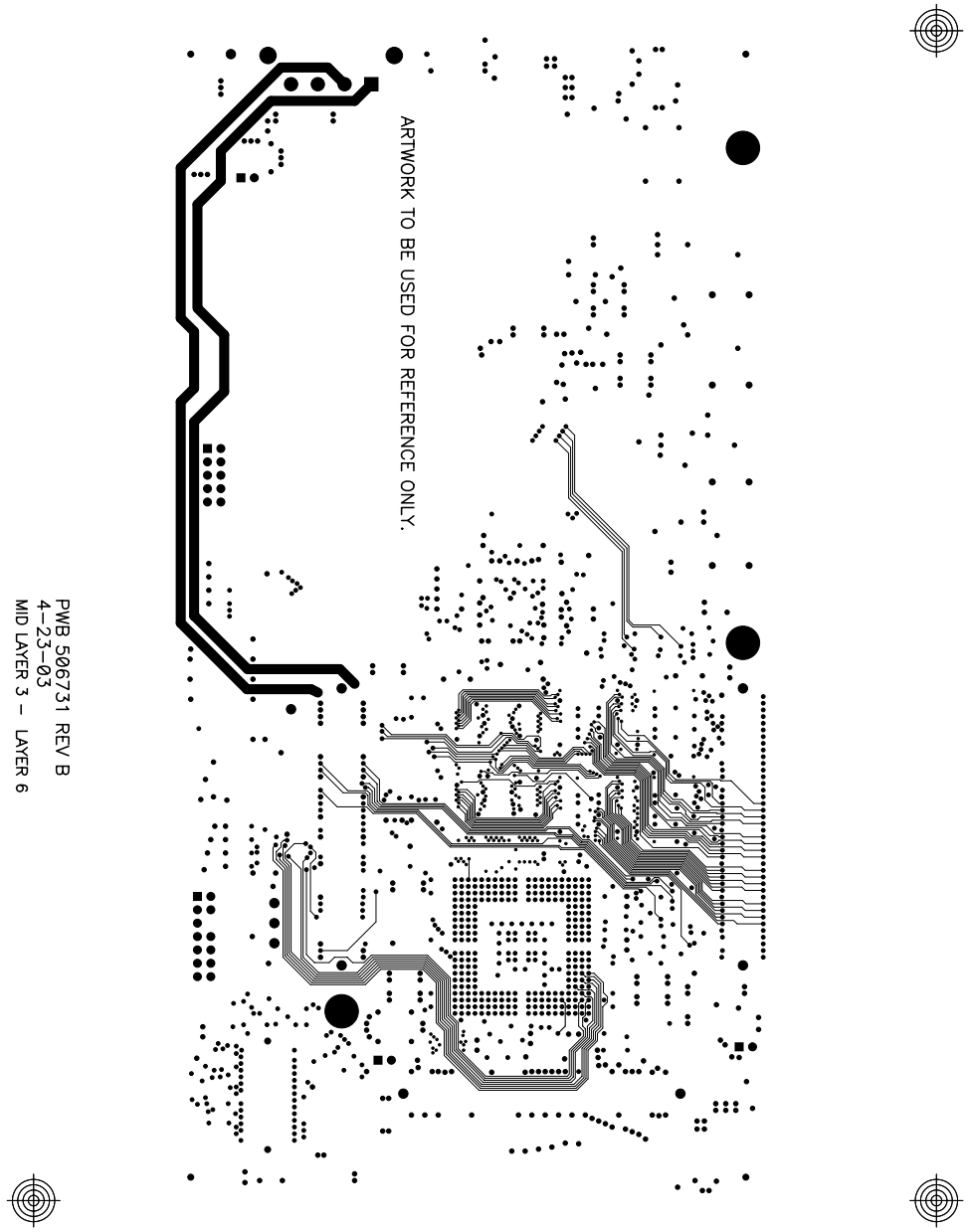
CAM350 V 6.0 : Thu May 08 10:57:55 2003 - (Untitled) : Title



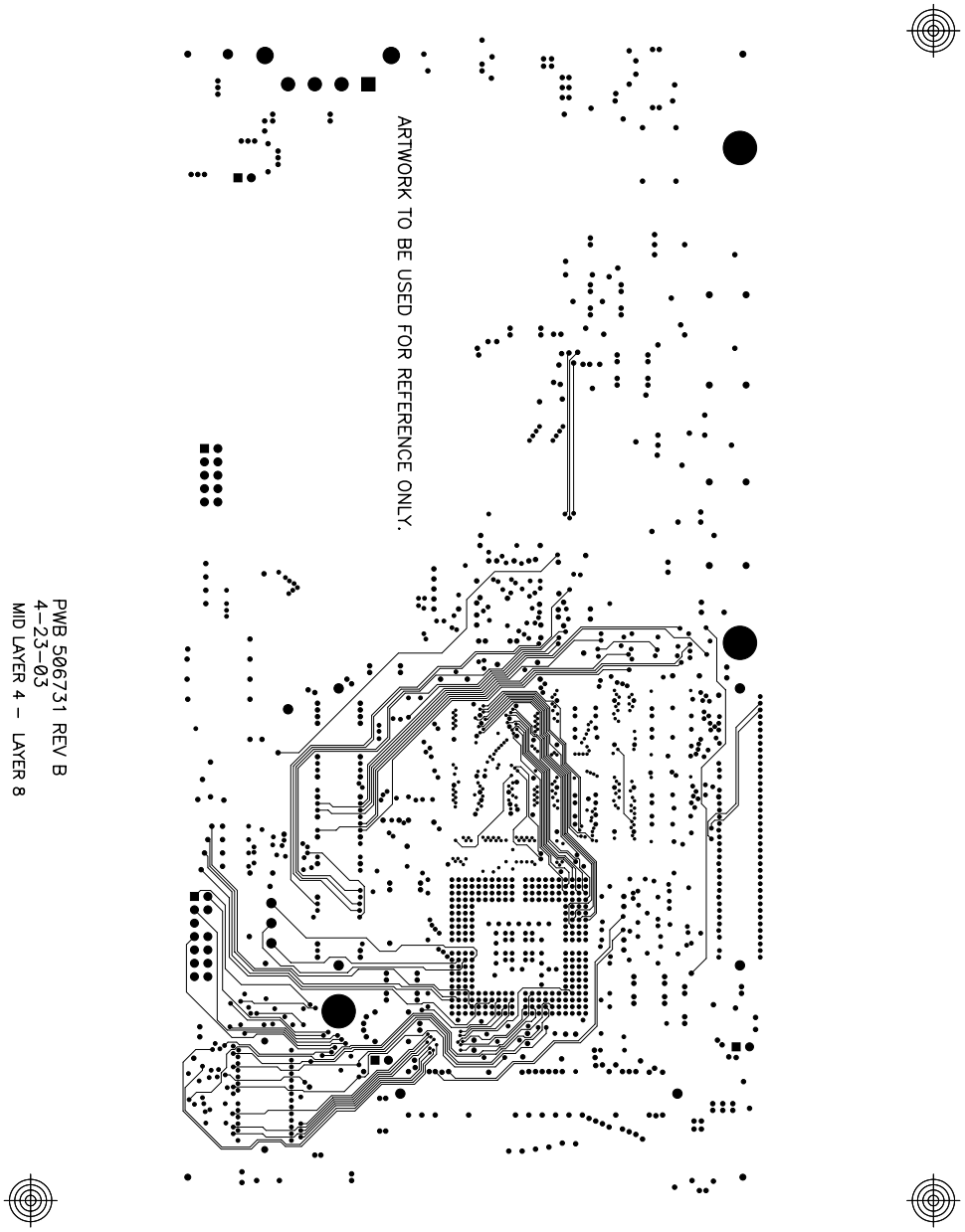
CAM350 V 6.0 : Thu May 08 10:57:55 2003 - (Untitled) : Title



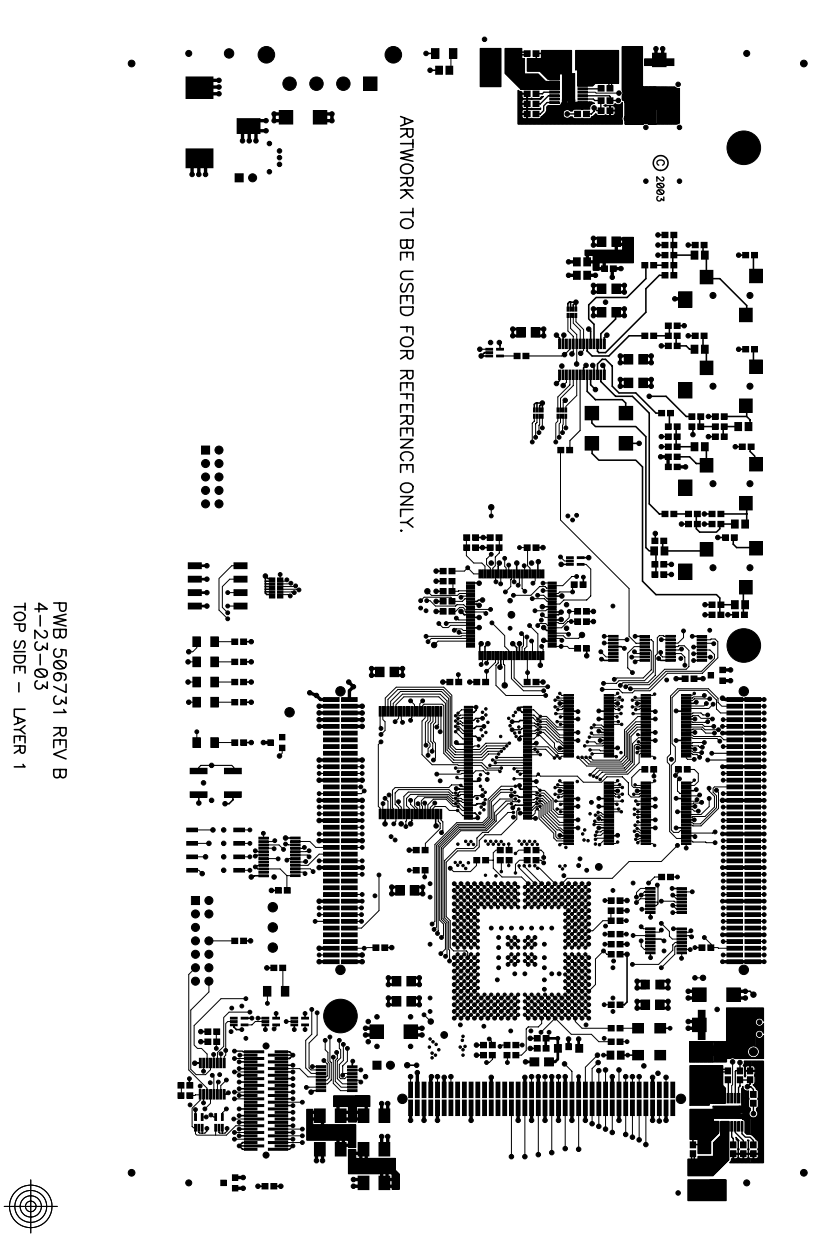
CAM350 V 6.0 : Thu May 08 10:57:56 2003 - (Untitled) : Title



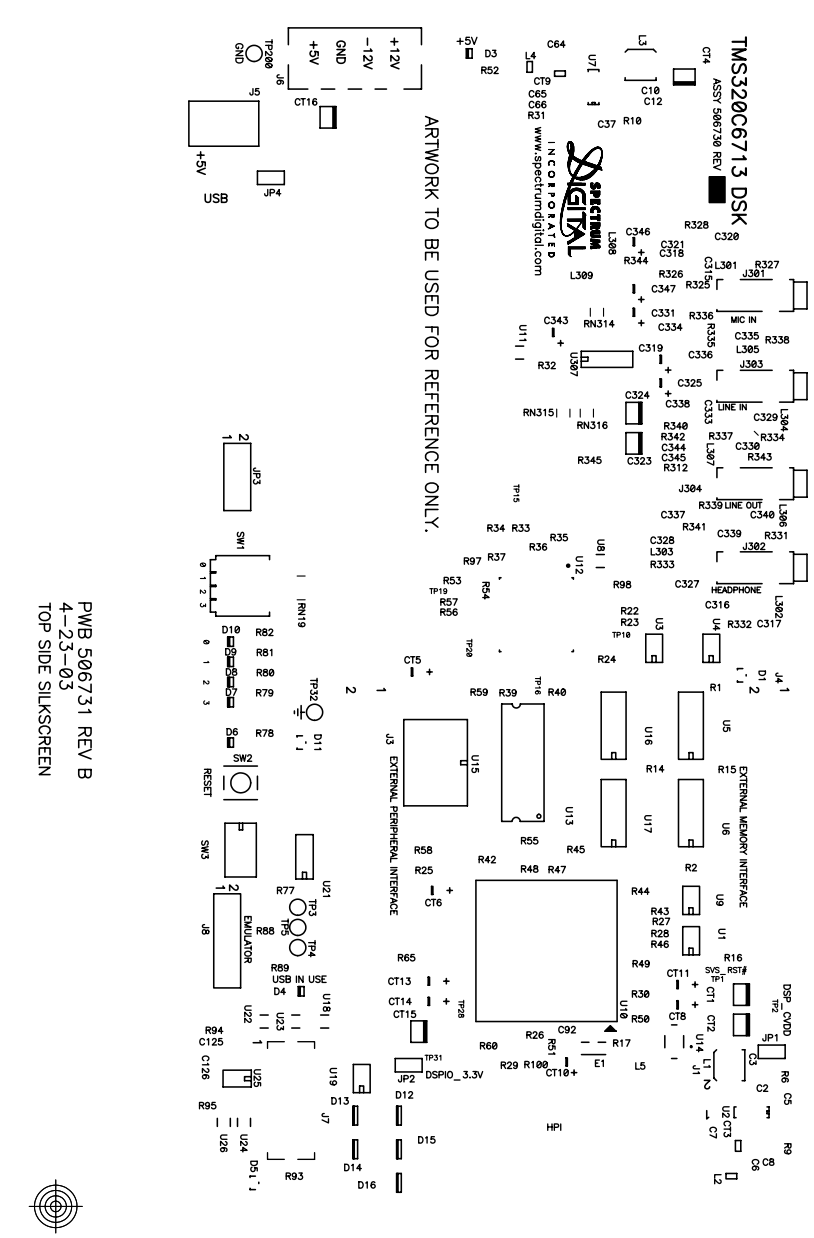
CAM350 V 6.0 : Thu May 08 10:57:56 2003 - (Untitled) : Title



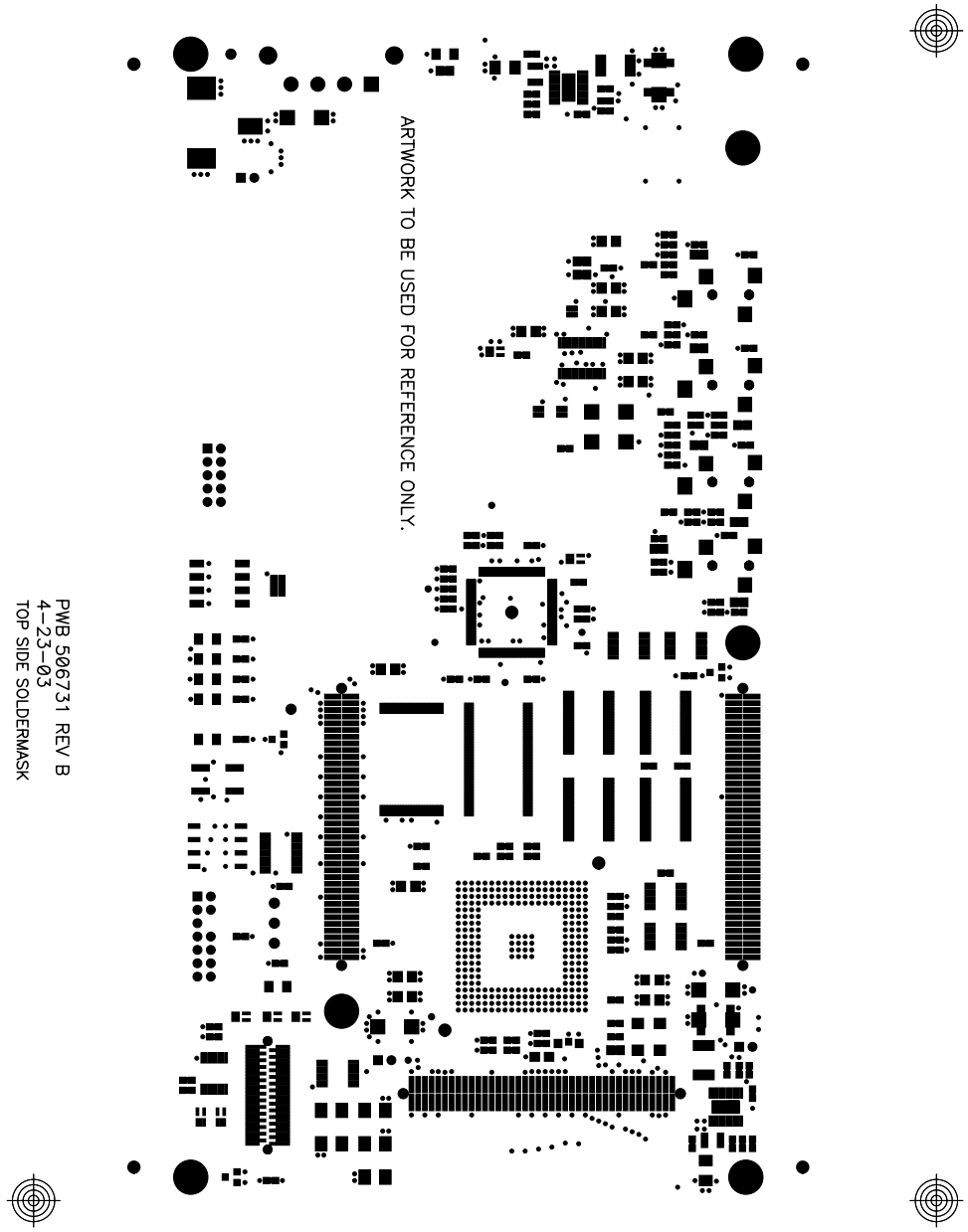
CAM350 V 6.0 : Thu May 08 10:57:57 2003 - (Untitled) : Title



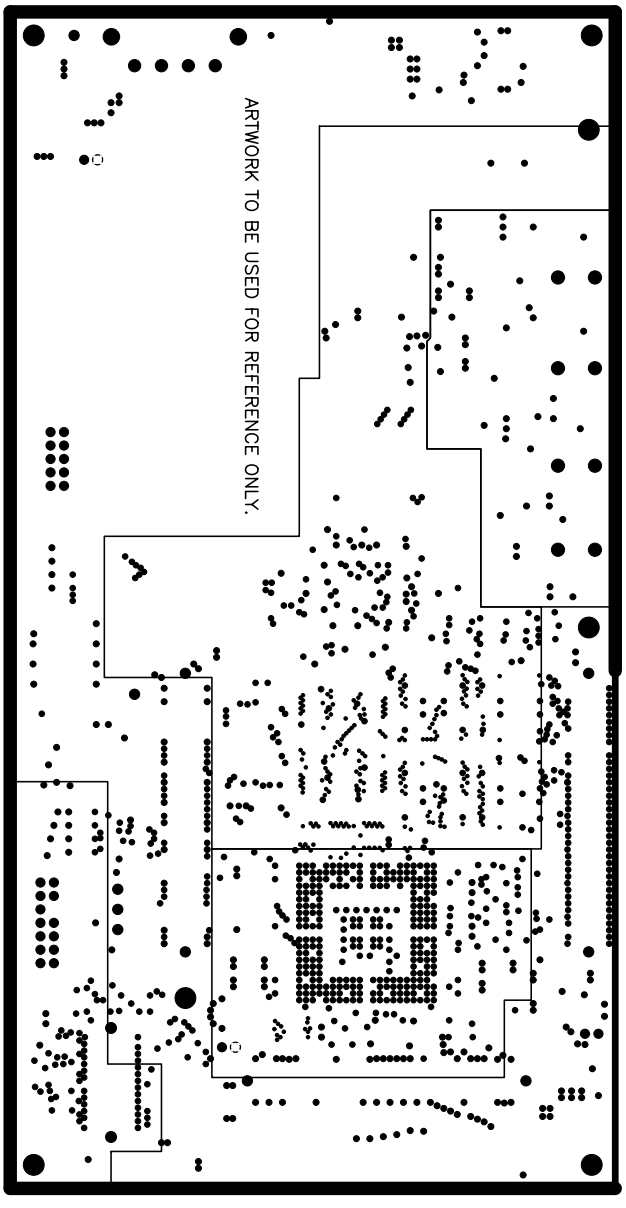
CAM350 V 6.0 : Thu May 08 10:57:58 2003 - (Untitled) : Title



CAM350 V 6.0 : Thu May 08 10:57:58 2003 - (Untitled) : Title

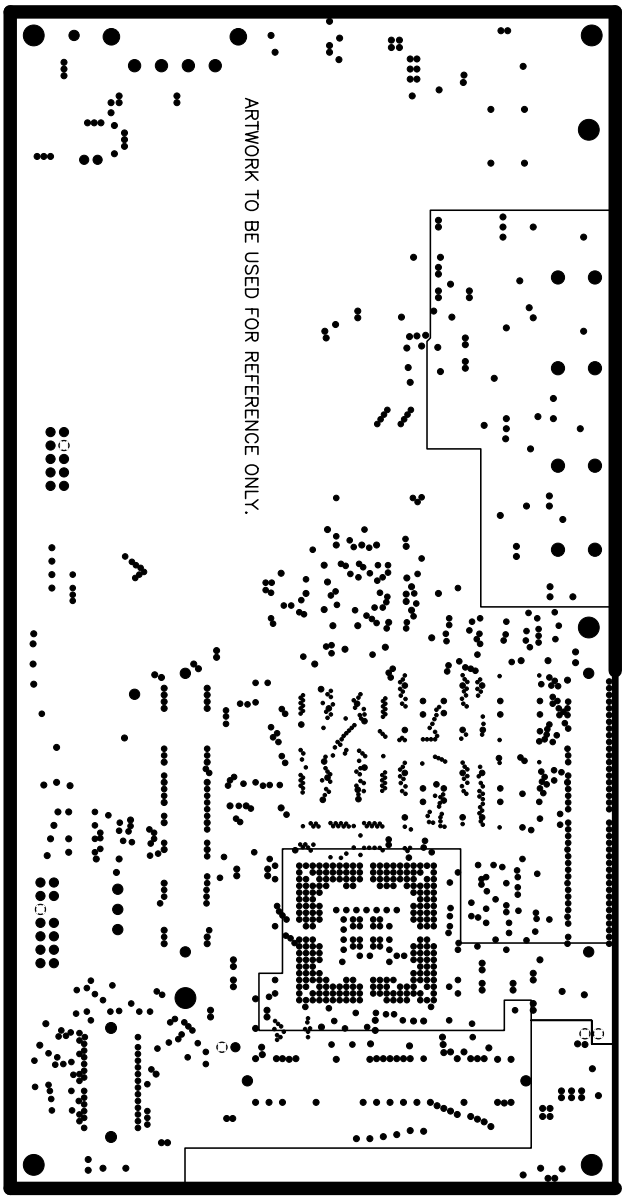


CAM350 V 6.0 : Thu May 08 10:57:58 2003 - (Untitled) : Title

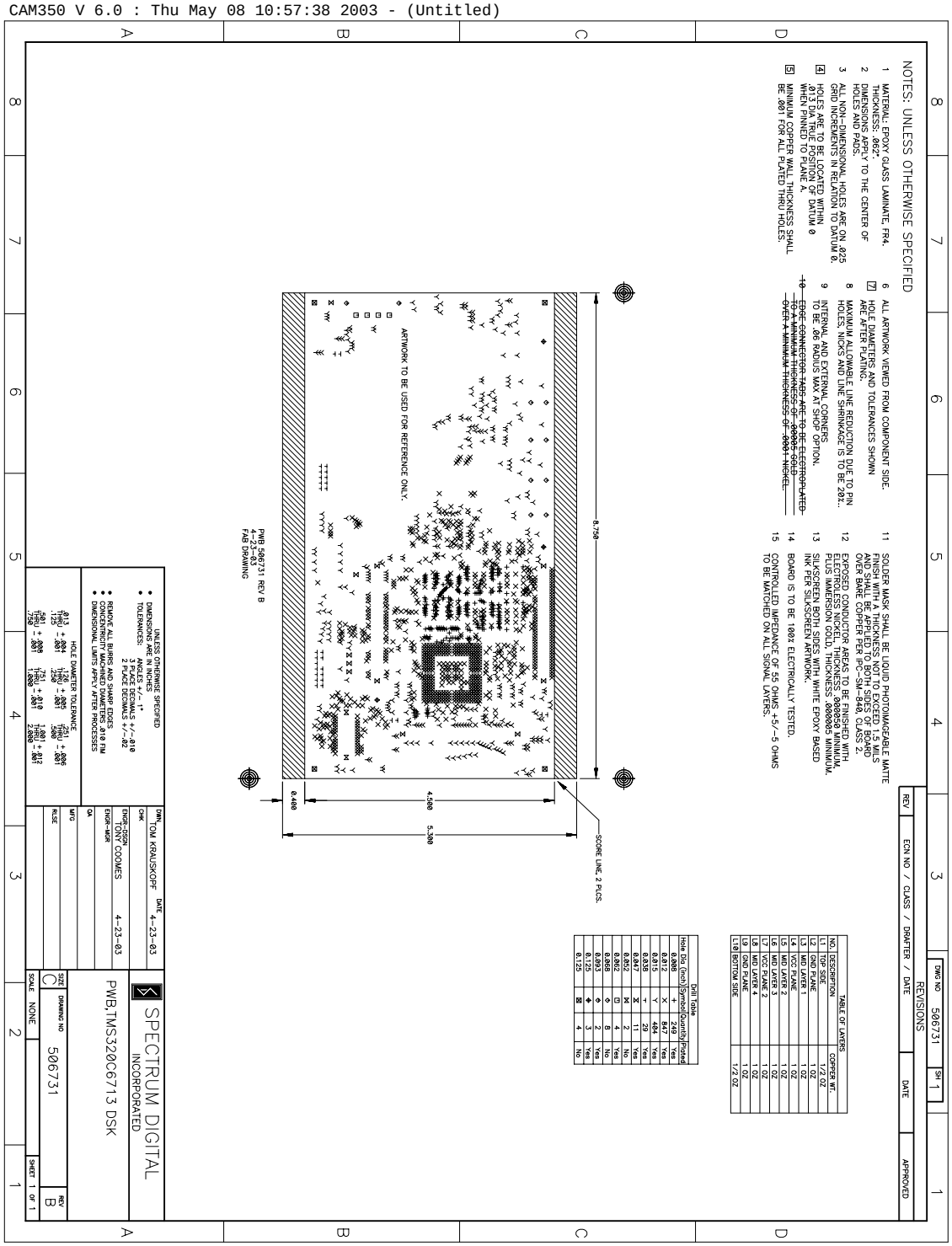


PWB 506731 REV B
4-23-03
VCC PLANE 2 - LAYER 7

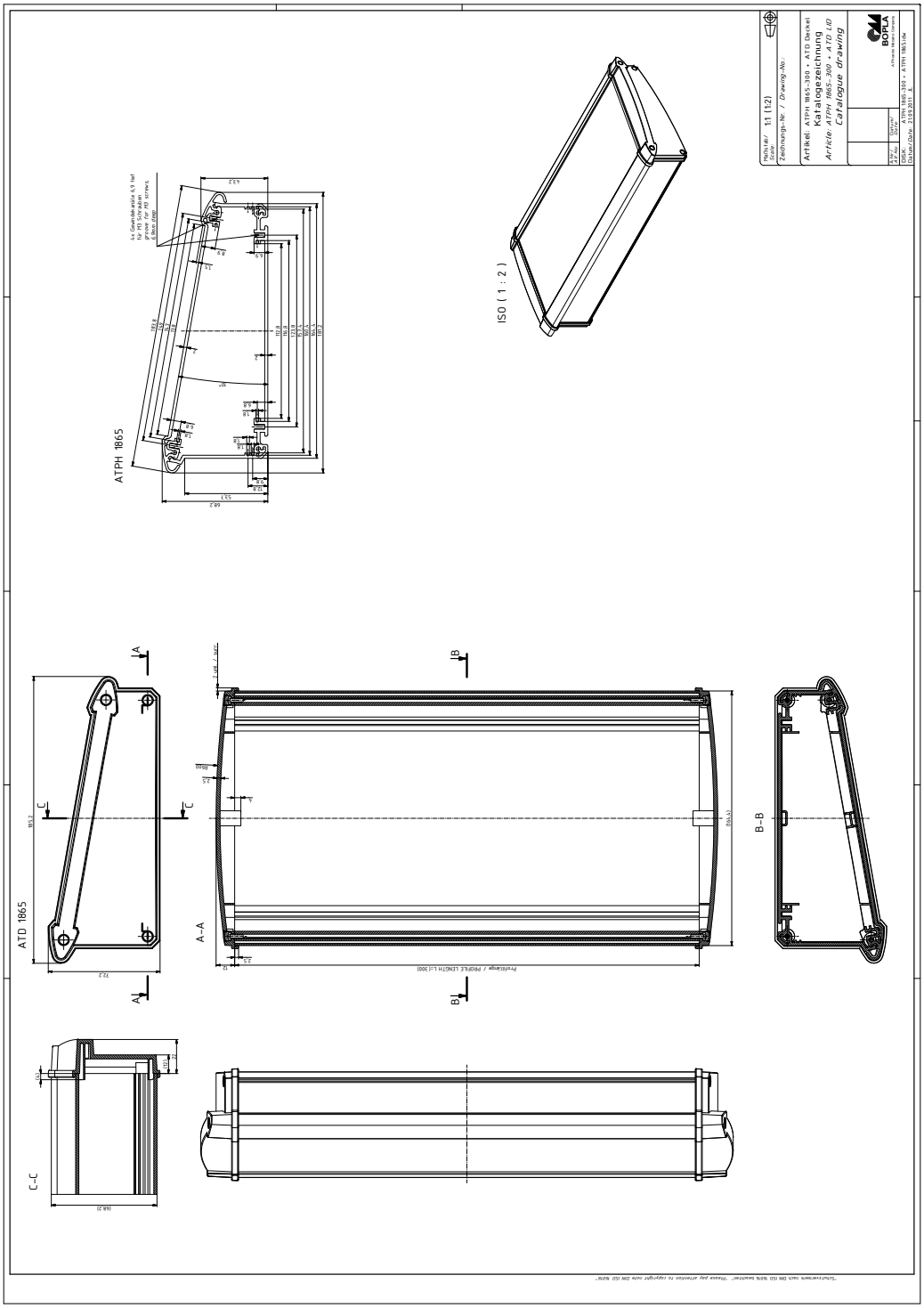
CAM350 V 6.0 : Thu May 08 10:57:59 2003 - (Untitled) : Title



PWB 506731 REV B
4-23-03
VCC PLANE - LAYER 4


CAM350 V 6.0 : Thu May 08 10:57:38 2003 - (Untitled)
NOTES: UNLESS OTHERWISE SPECIFIED
1 MATERIAL: EPOXY GLASS LAMINATE, FR4. THICKNESS: .062"
2 DIMENSIONS APPLY TO THE CENTER OF HOLES AND PADS.
3 ALL NON-DIMENSIONAL HOLES ARE ON .025 GRID INCREMENTS IN RELATION TO DATUM 0.
4 HOLES ARE TO BE LOCATED WITHIN 0.13 TRUE POSITION OF DATUM 0 WHEN PLANED TO FINISH SIZE.
5 MINIMUM COPPER WALL THICKNESS SHALL BE .001 FOR ALL PLATED THRU HOLES.
6 ALL ARTWORK VIEWED FROM COMPONENT SIDE.
7 HOLE DIAMETERS AND TOLERANCES SHOWN ARE AFTER PLATING.
8 MAXIMUM ALLOWABLE LINE REDUCTION DUE TO PIN HOLES, NICKS AND LINE SHRINKAGE IS TO BE 20%.
9 INTERNAL AND EXTERNAL CORNERS TO BE .06" RADIUS MAX. SHOP OPTION.
10 EDGE CONNECTOR TABS ARE TO BE ELECTROPLATED TO A MINIMUM THICKNESS OF .00005 GOLD OVER A MINIMUM THICKNESS OF .0001 NICKEL.
11 SOLDER MASK SHALL BE LIQUID PHOTOIMAGEABLE MATTE FINISH WITH A THICKNESS NOT TO EXCEED 1.5 MILS AND SHALL BE APPLIED TO BOTH SIDES OF BOARD OVER BARE COPPER PER IPC-SM-840, CLASS 2.
12 EXPOSED CONDUCTOR AREAS TO BE FINISHED WITH ELECTROLESS NICKEL/IMMERSION GOLD. THICKNESS: .000005 MINIMUM PLUS IMMERSION GOLD. THICKNESS: .000025 MINIMUM.
13 SILKSCREEN BOTH SIDES WITH WHITE EPOXY BASED INK PER SILKSCREEN ARTWORK.
14 BOARD IS TO BE 100% ELECTRICALLY TESTED.
15 CONTROLLED IMPEDANCE OF 55 OHMS +5/-5 OHMS TO BE MATCHED ON ALL SIGNAL LAYERS.
REVISIONS
REV | ECN NO / CLASS / DRAFTER / DATE | DATE | APPROVED
DWG NO 506731 SH 1
TABLE OF LAYERS
NO. | DESCRIPTION | COPPER WT.
L1 TOP SIDE 1/2 OZ
L2 GND PLANE 1 OZ
L4 VCC PLANE 1 OZ
L5 MID LAYER 2 1 OZ
L6 MID LAYER 3 1 OZ
L7 VCC PLANE 2 1 OZ
L9 GND PLANE 4 1 OZ
L10 BOTTOM SIDE 1/2 OZ
Drill Table
Hole Dia (inch) | Symbol | Quantity | Plated
0.012 X 847 Yes
0.015 Y 484 Yes
0.035 + 29 Yes
0.042 # 2 No
0.062 □ 4 Yes
0.098 ◇ 8 No
0.093 ◇ 2 Yes
0.125 ◆ 3 Yes
0.125 ⊠ 4 No
8.750
4.500
5.300
0.400
SCORE LINE, 2 PLCS.
ARTWORK TO BE USED FOR REFERENCE ONLY.
PWB 506731 REV B
4-23-03
FAB DRAWING
UNLESS OTHERWISE SPECIFIED
• DIMENSIONS ARE IN INCHES
• TOLERANCES: ANGLES +/- 1°
2 PLACE DECIMALS +/-.010
3 PLACE DECIMALS +/-.02
• REMOVE ALL BURRS AND SHARP EDGES
• CONCENTRICITY MACHINED DIAMETERS .010 FIM
• DIMENSIONAL LIMITS APPLY AFTER PROCESSES
HOLE DIAMETER TOLERANCE
.013 +.004 -.001 (.251 +.005 -.001)
.125 +.004 -.001 (3.20 +.005 -.001)
.250 +.004 -.001 (6.35 +.005 -.001)
.500 +.004 -.001 (12.70 +.005 -.001)
1.000 +.010 -.001 (25.40 +.013 -.001)
2.000 +.008 -.001 (50.80 +.017 -.001)
DWN TOM KRAUSKOPF DATE 4-23-03
CHK
ENGR-DSGN TONY COOMES 4-23-03
ENGR-MGR
QA
MFG
RLSE
SPECTRUM DIGITAL INCORPORATED
PWB,TMS320C6713 DSK
SIZE C DRAWING NO 506731 REV B
SCALE NONE SHEET 1 OF 1

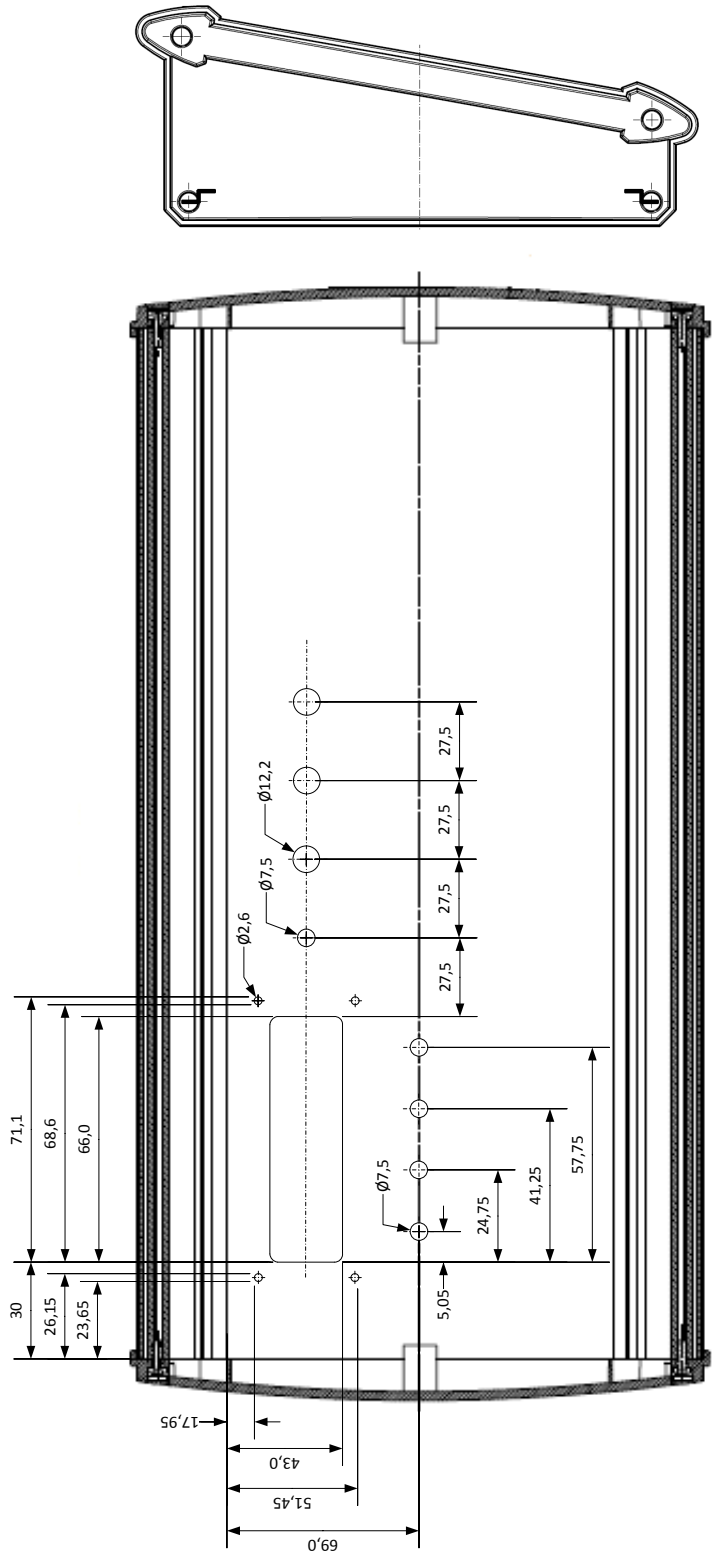
E.3. Konstruktionszeichnungen



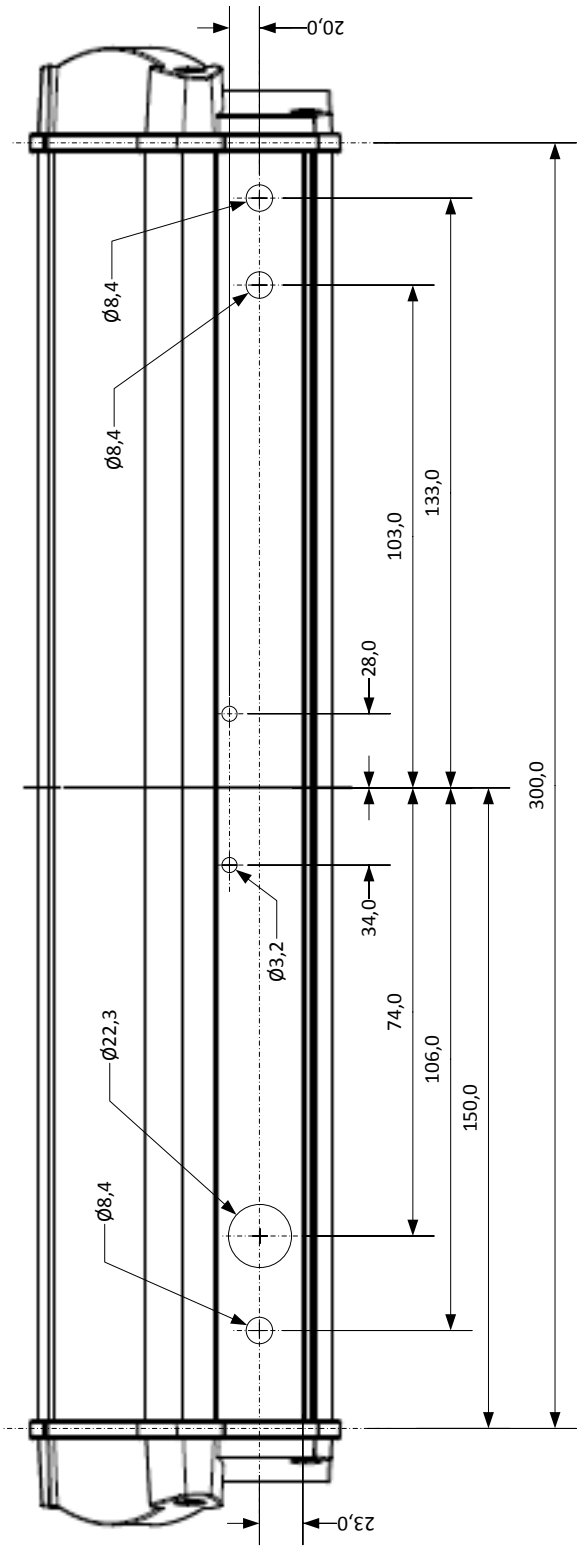
Gehäuse[23]

[23] Die Konstruktionszeichnung zum Gehäuse ist dem Datenblatt entnommen und ist auf der DVD als pdf-Datei enthalten. In der pdf-Datei kann in die Zeichnung hineingezoomt werden, um die Bemaßung besser lesen zu können.

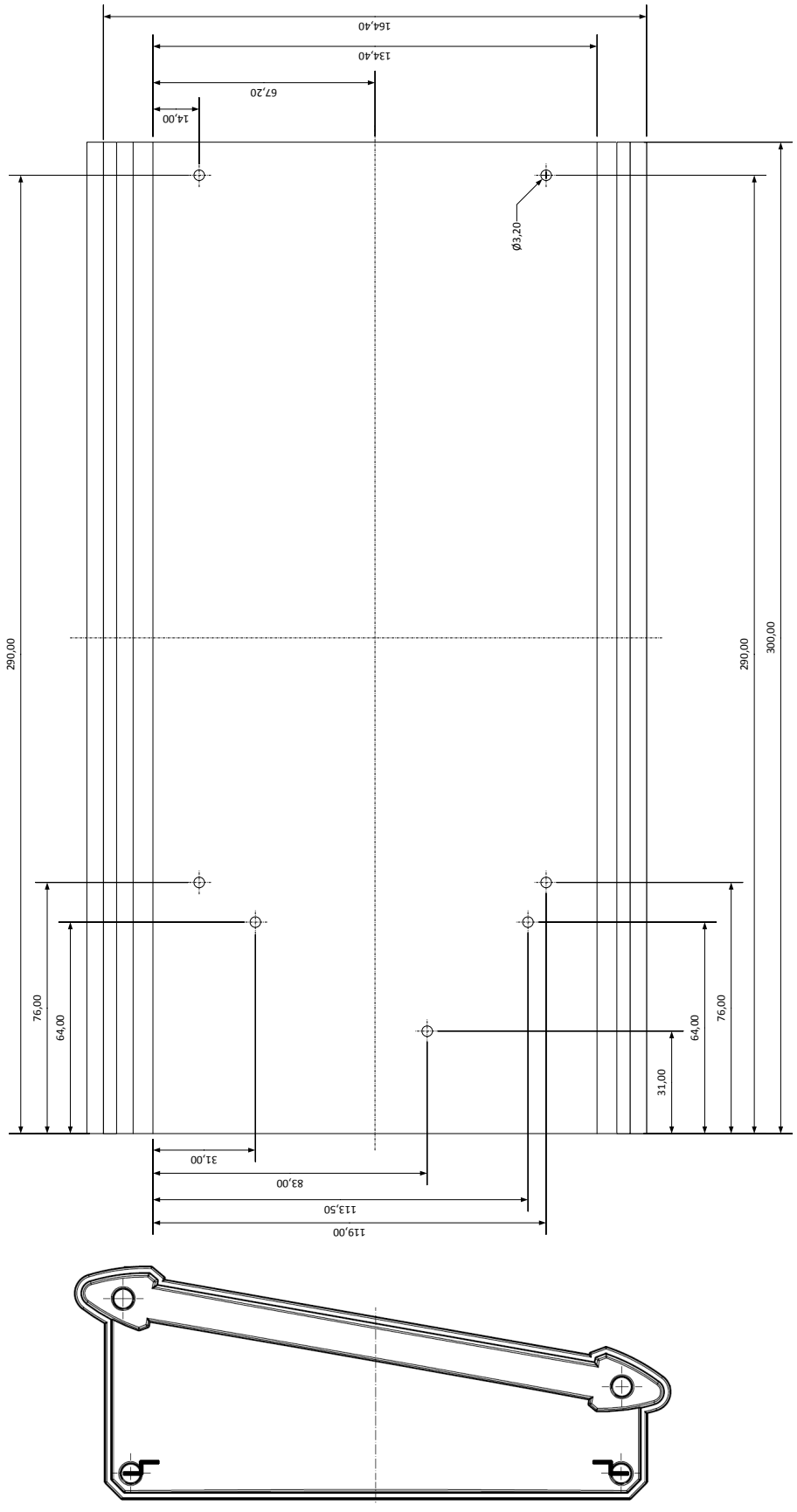
Die folgenden Konstruktionszeichnungen sind nicht maßstabsgetreu gezeichnet.



Gehäusedeckel mit Bohrungen



Gehäuserückseite mit Bohrungen



Gehäuseunterseite mit Bohrungen

F. Abgleichvorschrift

F.1. I²C-Bus LC-Display Modul

Auf dem I²C-Bus LC-Display Modul ist das Potentiometer P1. Mit diesem kann der Kontrast des LCD eingestellt werden. Dieses sollte vor dem Schließen des Gerätes geschehen. Dazu ist das Potentiometer im oder entgegen des Uhrzeigersinns zu drehen, bis der Kontrast nach belieben eingestellt ist.

Weiterhin sind sämtliche Jumper die auf dem JP1 header gesteckt sind zu entfernen. Damit wird die I²C-Schnittstelle auf die Adresse 0x27 eingestellt.

Die Jumper am header JP2 für die Pull-Up-Widerstände sind entsprechend der Beschriftung auf der Platine zu setzen.

F.2. TMS320C6713 DSK

Der Configuration Switch SW3 ist wie folgt einzustellen:

Switch 1:	off	- Little endian
Switch 2 & 3:	off	- EMIF boot from 8-bit Flash-Memory
Switch 4:	on	- McASP1 enabled on HPI pins

Ein Hinweis für die Fehlersuche. Sollte das Rücksetzen des DSP-Boards nicht funktionieren, bei Auswahl der Reset-Funktion über Taster oder Menü, so ist der Switch 4 zu prüfen. Wenn dieser auf off steht wird das Reset-Signal nicht weitergegeben.

Des Weiteren ist das Programm ASPP in den Flash-Speicher zu übertragen, vor der ersten Verwendung der Platine.

F.3. micro-SD card

Bei Wechsel oder Erstverwendung der micro-SD card ist auf diese das Webinterface aufzuspielen und die Speicherplätze einzurichten. Kopieren Sie dazu den Inhalt des Verzeichnisses */sd* der beigelegten DVD in das Root-Verzeichnis der micro-SD card.

Auch hier ein Hinweis zur Fehlersuche. Ist das Arduino Ethernet Shield aufgesteckt, die micro-SD card eingelegt und hängt sich die Klangregelung beim Startvorgang auf, dann kann dies daran liegen, dass die Datei *0.tc* (Speicherort 0 bzw. default) im Hauptverzeichnis der micro-SD card nicht vorhanden ist.

F.4. Arduino Due

Das Programm CUIP ist vor der ersten Verwendung des Arduino Due auf den internen Flash-Speicher der CPU zu übertragen.

G. Programmunterlagen

G.1. Control and User Interface Program (CUIP)

Die folgende Tabelle beschreibt das Menü und die Einstellmöglichkeiten.

Untermenü im Hauptmenü	Menüpunkt	Beschreibung
Lowpass	LP filtering	on Tiefen-Shelving-Filter ist aktiviert off Tiefen-Shelving-Filter ist deaktiviert
	LP left channel	on Tiefen-Shelving-Filter linker Kanal aktiviert off Tiefen-Shelving-Filter linker Kanal deaktiviert
	LP right channel	on Tiefen-Shelving-Filter rechter Kanal aktiviert off Tiefen-Shelving-Filter rechter Kanal deaktiviert
	LP Cuttoff freq.	Grenzfrequenz $f_{c_{lp}}$ des Tiefen-Shelving-Filters. Einstellbar im Bereich 20 Hz bis 1000 Hz in 1-Hz-Schritten.
	LP Amplification	logarithmischer Verstärkungsfaktor G_{lp} des Tiefen-Shelving-Filters. Einstellbar im Bereich -30 dB bis 30 dB in 1-dB-Schritten.
Highpass	HP filtering	on Höhen-Shelving-Filter ist aktiviert off Höhen-Shelving-Filter ist deaktiviert
	HP left channel	on Höhen-Shelving-Filter linker Kanal aktiviert off Höhen-Shelving-Filter linker Kanal deaktiviert
	HP right channel	on Höhen-Shelving-Filter rechter Kanal aktiviert off Höhen-Shelving-Filter rechter Kanal deaktiviert
	HP Cuttoff freq.	Grenzfrequenz $f_{c_{hp}}$ des Höhen-Shelving-Filters. Einstellbar im Bereich 1000 Hz bis 16000 Hz in 1-Hz-Schritten.

Untermenü im Hauptmenü	Menüpunkt	Beschreibung
	HP Amplification	logarithmischer Verstärkungsfaktor G_{hp} des Höhen-Shelving-Filters. Einstellbar im Bereich -30 dB bis 30 dB in 1-dB-Schritten.
Limiter	LT activation	on Limiter aktiviert off Limiter deaktiviert
	LT Threshold	Limiterschwelle *lt* einstellbar im Bereich 0 % und 100 % in 1-%-Schritten.
	LT Attack time	Limiter-Ansprechzeit t_a einstellbar im Bereich 20 μs und 10000 μs in 1-μs-Schritten
	LT Release time	Limiter-Rücklaufzeit t_r einstellbar im Bereich 2 ms und 5000 ms in 1-ms-Schritten
Audio	Volume	logarithmischer Verstärkungsfaktor G_V zur Einstellung der Lautstärke im Bereich von -30 dB bis 30 dB in 1-dB-Schritten.
	Mute	on Stummschaltung aktiviert off Stummschaltung deaktiviert
	Fade time	Ein-/Ausblendzeit t_F einstellbar im Bereich 0 s bis 10 s in 1-s-Schritten.
	Balance	Balancewert *B* einstellbar im Bereich -7 bis 7 in 1er-Schritten.
Audio-Codec	Line in Volume	logarithmischer Verstärkungsfaktor G_{ADC} wählbar im Bereich von -34,5 dB bis 12,0 dB in 1,5-dB-Schritten.
	De-Emphasis	Deakzentuierungsfilter des Audio-Codecs mit den Einstellmöglichkeiten: • off (Deakzentuierungsfilter deaktiviert) • 32 kHz • 44,1 kHz • 48 kHz
Network	Network mode	off Netzwerk deaktiviert static statische IP-Adresse DHCP IP-Adresse über DHCP
	IP address	Zeigt im Modus *off* keine IP-Adresse an, da keine vergeben ist. Im Modus *static* und *DHCP* wird die momentan verwendete IP-Adresse angezeigt. Im Modus *static* kann die IP-Adresse geändert werden.

Untermenü im Hauptmenü	Menüpunkt	Beschreibung
Options	BL turn-off time	Abschaltzeit t_{LCD} einstellbar im Bereich 0 s (off) und 60 s in 1-s-Schritten. Die Abschaltzeit gibt an, nach wie viel Sekunden die Hintergrundbeleuchtung des LCDs abgeschalten werden soll, wenn keine weiteren Eingaben erfolgen.
	Load config	Lädt eine Konfiguration aus dem angezeigten Speicherort. Der Speicherort kann an dieser Stelle auch geändert werden.
	Save config	Speichert die aktuelle Konfiguration der Klangregelung an den angezeigten Speicherort. Der Speicherort kann an dieser Stelle auch geändert werden.
	Reset	Hiermit kann ein Zurücksetzen der Klangregelung veranlasst werden.
	set to factory settings	Mit diesem Menüpunkt werden alle Konfigurationsparameter auf die ”Werkseinstellungen”gesetzt und die Konfiguration auf den Speicherort 0 (default) gespeichert. Anschließend wird das Reset ausgeführt, womit die Klangregelung neu gestartet und somit die Konfiguration auf dem Speicherort 0 (default) geladen und die AU darauf eingestellt wird.

Übersicht über das Menü des User Interfaces

In folgender Tabelle werden die Funktionen der Inkrementalgeber beschrieben, welche Sie bei einem bestimmten Zustand haben.

Inkrementalgeber	Hauptbildschirm	Balanceeinstellung	Hauptmenü	Untermenü
F1 (Taster)	Menü aufrufen	Balanceeinstellung verlassen	Untermenü aufrufen	Untermenü verlassen und zum Hauptmenü zurückkehren
F1 (Geber)	Balanceeinstellung aufrufen	Balance verändern	Untermenü auswählen	Menüpunkt auswählen
F2 (Taster)	Stummschaltung (Mute) aktivieren/-deaktivieren	—	—	Auslösen diverser Funktionen wenn die entsprechenden Menüpunkte ausgewählt sind (z.B. Reset, Save config usw.).
F2 (Geber)	Volume verändern	—	—	Parameterwerte eines entsprechenden Menüpunktes in 1er-Schritten verändern bzw. zwischen aktiv und inaktiv umschalten.
F3 (Taster)	Limiter aktivierien/-deaktivieren	—	—	—
F3 (Geber)	Limiterschwelle verändern	—	—	Parameterwerte eines entsprechenden Menüpunktes in 10er-Schritten verändern.
F4 (Taster)	Hochpass aktivieri-en/deaktivieren	—	—	—
F4 (Geber)	Verstärkung des Hochpasses ver-ändern	—	—	Parameterwerte eines entsprechenden Menüpunktes in 100er-Schritten verändern.
F5 (Taster)	Tiefpass aktivierien/-deaktivieren	—	—	—
F5 (Geber)	Verstärkung des Tief-passes ver- ändern	—	—	Parameterwerte eines entsprechenden Menüpunktes in 1000er-Schritten verändern.

Funktionen der Inkrementalgeber in Bezug auf den Zustand der UI

In der folgenden Tabelle ist aufgelistet an welcher Stelle ein bestimmter Parameterwert in einer tc-Datei steht. Eine tc-Datei enthält eine vollständige Konfiguration des Klangreglers.

Byte	Parameter
0 - 3	Line in Volume
4 - 7	De-Emphasis
8 - 11	DAC select
12 - 15	Bypass
16 - 19	Lowpass Bypass
20 - 23	Lowpass left channel bypass
24 - 27	Lowpass right channel bypass
28 - 31	Lowpass Cutoff frequency
32 - 35	Lowpass amplification
36 - 39	Highpass Bypass
40 - 43	Highpass left channel bypass
44 - 47	Highpass right channel bypass
48 - 51	Highpass Cutoff frequency
52 - 55	Highpass amplification
56 - 59	Volume
60 - 63	Fade time
64 - 67	Balance
68 - 71	Mute
72 - 75	Limiter bypass
76 - 79	Limiter threshold
80 - 83	Limiter attack time
84 - 87	Limiter release time
88 - 91	LCD Backlight turn-off time
92 - 95	Network mode
96 - 101	MAC address
102 - 105	IP address
106 - 109	Netmask
110 - 113	Gateway
114 - 117	DNS
118 - 121	TCP Port
122 - 125	Transmit UDP Port
126 - 129	Receive UDP Port

Dateiformat der Konfigurationsdateien des Klangreglers

Die folgende Tabelle beschreibt die UDP-Kommandos.

Request	Beschreibung
setIP	Abfrage der IP-Adresse
s_G_ADC	Line in Volume einstellen
s_d	De-Emphasis einstellen
s_DAC_en	DAC select einstellen
s_b	Bypass einstellen
s_lp_en	Lowpass bypass einstellen
s_lp_r_en	Lowpass right channel bypass einstellen
s_lp_l_en	Lowpass left channel bypass einstellen
s_lp_fc	Lowpass cutoff frequency einstellen
d_lp_G	Lowpass amplification einstellen
s_hp_en	Highpass bypass einstellen
s_hp_r_en	Highpass right channel bypass einstellen
s_hp_l_en	Highpass left channel bypass einstellen
s_hp_fc	Highpass cutoff frequency einstellen
s_hp_G	Highpass amplification einstellen
s_G_V	Volume einstellen
s_t_F	Fade time einstellen
s_B	Balance einstellen
s_m	Mute einstellen
s_lt_en	Limiter bypass einstellen
s_lt	Limiter threshold einstellen
s_ta	Limiter attack time einstellen
s_tr	Limiter release time einstellen
s_t_lcd	LCD backlight turn-off time einstellen
s_storage	Speicherplatz einstellen
s_load	Konfiguration laden
s_save	Konfiguration speichern
s_reset	Reset ausführen
s_factory	Alle Einstellungen auf Werkseinstellungen zurücksetzen und Reset ausführen
getConfig	Konfiguration abfragen

Übersicht über die Requestinhalte der UDP-Kommunikation

G.2. Tone Control Webinterface Program (TCWIP)

Die folgende Tabelle listet die Javascript-Funktionen des TCWIP auf und erläutert deren Aufgaben.

Funktion	Aufgabe
request(url, params)	Mit der Funktion request wird ein HTTPRequest ausgeführt. Zu übergeben sind die URL und die Parameter.
check_timeout()	Prüft ob ein Timeout beim HTTPRequest vorliegt. Ist dies der Fall wird dieser abgebrochen und eine Anfrage der Konfiguration des Klangreglers erneut gestartet.
get_config()	Führt einen HTTPRequest über die Funktion *request* aus, um die gesamte Konfiguration des Klangreglers abzurufen.
parse(a)	Zerlegt die Antwort auf das Abrufen der Gesamtkonfiguration des Klangreglers in die einzelnen Parameter und veranlasst das Ändern der jeweiligen Anzeigeelemente.
update(key)	Prüft ob ein Parameter sich geändert hat. Ist dies der Fall, so wird die Anzeige des jeweiligen Parameters entsprechend geändert. Über den Parameter *key* wird das Schlüsselwort des zu prüfenden Parameters übergeben.
get_checked(checked)	Wandelt die Werte einer Checkbox in Zahlenwerte um. (True = 1; False = 0)
set_param(cmd)	Veranlasst einen HTTPRequest über die Funktion *request* zur Änderung eines Parameterwertes im Klangregler.
update_enable()	Aktiviert oder Deaktiviert diverse Schaltflächen, Checkboxen und Auswahlfelder entsprechend des Zustandes des Klangreglers.

Funktion	Aufgabe
init_axes()	Mit der Funktion *init_axes* werden die Diagrammfelder auf eine bestimmte Größe eingestellt.
update_axes()	Wird aufgerufen um die Diagramme neu zeichnen zu lassen bzw. zu aktualisieren.
px2int(px)	Wandelt eine Zeichenkette mit einem Pixelwert in einen Integerwert um (z.B. “20 px “ = 20).
set_dimension(id)	Mit der Funktion set_dimension wird die Größe der Graphen festgelegt und die Offset-Werte (rechter, linker Rand usw.) initialisiert.
plot_ground(e, id)	Zeichnet den Hintergrund der Diagrammfelder.
plotXGrid(e,id)	Zeichnet die Gitternetzlinien der X-Achse.
calc_coeffs()	Ruft die Funktionen zum berechnen der Koeffizienten des Höhen- und Tiefen-Shelving-Filters auf.
calc_lp_coeffs()	Berechnet die Koeffizienten des Tiefen-Shelving-Filters.
calc_hp_coeffs()	Berechnet die Koeffizienten des Höhen-Shelving-Filters.
calc_transfer_function(id)	Berechnet die Übertragungsfunktionen des Tiefen-, Höhen-Shelving-Filters und der Kaskade aus beiden Filtern.
from_exp_to_complex(abs,arg)	Rechnet die komplexe Zahl aus Betrag (abs) und Argument (arg) in Real- und Imaginärteil um.
c_add(a,b)	Addiert zwei komplexe Zahlen.
c_sub(a,b)	Subtrahiert zwei komplexe Zahlen.
c_mul(a,b)	Multipliziert zwei komplexe Zahlen.
c_div(a,b)	Dividiert zwei komplexe Zahlen.
function c_abs(r,i)	Berechnet aus dem Realteil (r) und dem Imaginärteil (i) den Betrag der komplexen Zahl.
c_arg(r,i)	Berechnet aus Realteil (r) und Imaginärteil (i) das Argument (Phase).
plotYGrid(e, id, max, min)	Zeichnet die Gitternetzlinien der Y-Achse.
plotTitle(e, id)	Zeichnet den Diagrammtitel.
plotGraph(e, id, d, min)	Zeichnet den Graphen.
clear()	Löscht das Diagrammfeld.

Übersicht über die Aufgaben der Javascript-Funktionen

H. Softwareverzeichnis

Im Folgenden wird die verwendete Software zur Erstellung des Klangreglers aufgelistet. Dazu gibt die erste Tabelle Auskunft über die verwendeten Herstellerabkürzungen.

Hersteller	Abkürzung
Appcelerator, Inc. 440 N. Bernardo Ave. Mountain View CA 94043 U.S.A. http://www.aptana.com/	AC
Arduino http://arduino.cc/	Arduino
Digia USA Inc. 2350 Mission College Blvd., Suite 1020 Santa Clara, California 95054, U.S.A http://qt.digia.com/	digia
MathWorks Adalperostraße 45 85737 Ismaning GERMANY http://www.mathworks.de/	MW
Texas Instruments Incorporated P.O. Box 660199 Dallas, TX 75266-0199 http://www.ti.com/	TI

Abkürzungen zu den Herstellern

Software	Version	Hersteller	Anwendungsbereich
Aptana Studio 3	3.2.2	AC	IDE zur Erstellung des TCWIP
Arduino IDE	1.5.2	Arduino	Erstellung des CUIP und Programmierung des Flash-Speichers des Arduino Due mit dem CUIP.
Code Composer Studio	3.1.0	TI	IDE für ASPP
FlashBurn	5.90.0.110	TI	Programmierung des Flash-Speichers auf dem TMS320C6713 DSK.
Matlab	Student Version R2012a (7.14.0.739)	MW	mathematische Berechnungen und TCSP
Qt Creator	2.7.1 (based on Qt 5.0.2)	digia	IDE zur Erstellung des TCGUIP
Simulink	7.9 (R2012a)	MW	Erstellung und Ausführung TCSP

Übersicht über die verwendete Software

I. DVD-Inhaltsverzeichnis

Der Arbeit ist eine DVD beigelegt. Diese beinhaltet:

- Bachelorarbeit im pdf-Format
- aufgelistete Software im Abschnitt H
- Datenblätter zum verwendeten Material
- Produktdokumentationen zum TMS320C6713 DSK, Arduino Due und Arduino Ethernet Shield
- Quellcodes zu TCSP, ASPP, CUIP, TCGUIP und TCWIP
- Messergebnisse (z.B. Dateien mit den Messwerten)

Ernst-Abbe-Fachhochschule Jena
Hochschule für angewandte Wissenschaften

Fachbereich Elektrotechnik und Informationstechnik

Thesen zur Bachelorarbeit

zur Erlangung des akademischen Grades

Bachelor of Engineering (B.Eng.):

Implementierung einer Klangregelung durch Shelving-Filter auf einem Signalprozessor mit Steuerung der Parameter über einen Mikrocontroller

eingereicht von:	Jürgen Döffinger
geboren am:	14.09.1976 in Leipzig
Matrikel-Nummer:	63 15 51
Studiengang:	Kommunikations- und Medientechnik
Hochschulbetreuer:	Prof. Dr.-Ing. Frank Giesecke
Datum der Themenausgabe:	02.10.2013
Datum der Abgabe:	06.01.2014

Thesen zur Bachelorarbeit

1. Die Simulation mit MATLAB/SimuLink ist eine schnelle und einfache Möglichkeit die in den Grundlagen aufgestellten Verfahren der Audiosignalverarbeitung auf ihre Praxistauglichkeit zu überprüfen.

2. Mit einem Tiefen- und einem Höhen-Shelving-Filter lassen sich drei Frequenzbereiche (Tiefen, Mitten und Höhen) so verändern, dass eine subjektive Verbesserung der auditiven Wahrnehmung beim Hörer erreicht wird.

3. Mit der Klangregelung lässt sich die unterschiedliche Wahrnehmung der Frequenzen durch das menschliche Ohr in einem begrenzten Maße ausgleichen. Gleiches gilt auch für die Unzulänglichkeiten von Aufnahme- und Wiedergabegeräten.

4. Shelving-Filter sind geeigneter für den Klangregler als andere vergleichbare Filter.

5. Die IIR-Filterstruktur kommt zum Einsatz, da diese geringere Verzögerungen bei der Filterung des Audiosignals aufweist, als die FIR-Filterstruktur.

6. Auch wenn der nicht lineare Phasengang einer IIR-Filterstruktur ungünstig ist, so wirkt sich die Veränderung des Phasengangs durch den Klangregler kaum bis nicht wahrnehmbar aus.

7. Auch wenn nur jeweils ein Filter für die tiefen- und hohen Frequenzen zur Anwendung kommt, kann durch entsprechende Veränderung der Grenzfrequenzen und Verstärkungsfaktoren eine gewollte, aber auch eine ungewollte, Beeinflussung der mittleren Frequenzen erzielt werden.

8. Ein Überlauf beim Audio-Codec, welcher mit Festkommazahlen arbeitet lässt sich mithilfe eines Limiters verhindern ohne Verzerrungen zu verursachen, wie es bei der klassischen Begrenzung durch Abschneiden der Amplituden (Clipping) der Fall ist.

9. Die Beeinflussung der Dynamikänderung durch den Limiter lässt sich durch geschickte Wahl von Ansprech- und Rücklaufzeit auf ein Minimum reduzieren.

10. Die Wandlung der Festkommawerte des Audio-Codecs in Fließkommazahlen durch den

DSP vereinfacht die Audiosignalverarbeitung.

11. Durch eine entsprechende Struktur in der Software der Audiosignalverarbeitung kann eine einfache Erweiterung des Klangreglers mit weiteren Filtern, Effekten o.ä. sichergestellt werden.

12. Die Auslagerung der Steuerung und der Verarbeitung der Eingaben des Anwenders lässt dem DSP die zeitliche Möglichkeit der Erweiterung des Klangreglers um weitere Filter, Effekte o.ä.

13. Die Verwendung einer Menüstruktur auf der Seite der Benutzerschnittstelle lässt die Möglichkeit zu auf einfache Weise die Steuerbarkeit von weiteren Filtern, Effekten o.ä. zu implementieren.

14. Die Verwendung der I^2C-Schnittstellen zur Weitergabe der Parameteränderungen durch den Anwender ist schnell genug und einfacher zu implementieren, als die Verwendung anderer Schnittstellen.

15. Durch erstellen entsprechender Algorithmen kann eine Unterbrechnung oder andere Störung der Audiosignalverarbeitung bei Änderung von Parametern der Audiosignalverarbeitung vermieden werden.

16. Durch die Verwendung eines eigenen Übertragungsprotokolls zwischen Steuerung und Audiosignalverarbeitung kann eine Erweiterung des Klangreglers um weitere Filter, Effekte o.ä. erreicht werden.

Selbstständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Bachelorarbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe. Diese Arbeit lag in gleicher oder ähnlicher Weise noch keiner Prüfungsbehörde vor und wurde bisher noch nicht veröffentlicht.

Jena, den 10. Dezember 2013