Hochschule für Angewandte Wissenschaften Hamburg
*Hamburg University of Applied Sciences*

# Bachelorthesis

Robert Ostermann

## Experimentalaufbau einer kombinierten logarithmischen und linearen Verstärkerbank für ABS-Sensoren

*Fakultät Technik und Informatik*
*Department Informations- und Elektrotechnik*

*Faculty of Engineering and Computer Science*
*Department of Information and Electrical Engineering*

Robert Ostermann

# Experimentalaufbau einer kombinierten logarithmischen und linearen Verstärkerbank für ABS-Sensoren

Bachelorthesis eingereicht im Rahmen der Bachelorprüfung
im Studiengang Informations- und Elektrotechnik
am Department Informations- und Elektrotechnik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. -Ing. Karl-Ragmar Riemschneider
Zweitgutachter : Prof. Dr. -Ing. Jürgen Vollmer

Abgegeben am 24. Februar 2012

**Robert Ostermann**

## Thema der Bachelorthesis

Experimentalaufbau einer kombinierten logarithmischen und linearen Verstärkerbank für ABS-Sensoren

## Stichworte

ABS-Sensoren, AMR-Effekt, logarithmische Verstärker, lineare Verstärker, Verstärkbank, Mikrocontroller

## Kurzzusammenfassung

Diese Arbeit beschreibt die Entwicklung einer kombinierten logarithmisch und linearen Verstärkerbank, welche mittels linearer Verstärker eine logarithmische Funktion abschnittsweise approximiert. Die dadurch zur Verfügung stehenden Signale wurden dazu verwendet, die zweidimensionale Regelung von Offset und Verstärkung eines bestehenden Controllersystems auf eine Dimension, die Offsetregelung zu reduzieren. Durch das System sollen anschließend AMR-Sensorsignale verarbeitet werden, welche eine große Dynamik in Amplitude und Frequenz umfassen.

**Robert Ostermann**

## Title of the paper

Experimental combination of a logarithmical and linear amplifier bank for anti-lock brake sensor

## Keywords

ABS-sensors, AMR-effect, logarithmic amplifier, linear amplifier, amplifier bank, microcontroller

## Abstract

This paper is on the development of a combination of logarithmic and linear amplifier, which approximate the logarithm by piecewise linear segments. The available signals are used to reduce the two-dimensional offset- and gain control to only one-dimensional control loop, the offset compensation. The system processes AMR-sensor signals which include a wide dynamic in amplitude and frequency range.

# Danksagung

An dieser Stelle möchte ich mich ganz besonders bei Herrn Prof. Dr.-Ing. Karl-Ragmar Riemschneider dafür bedanken, dass er mir ermöglicht hat diese Arbeit zu erstellen und für sein unermüdliches Engagement im Rahmen der Forschungsprojekte ESZ-ABS und BAT-SEN.

Weiterhin danke ich Herrn Prof. Dr.-Ing. Jürgen Vollmer für seine Arbeit als Zweitprüfer und seine stete Bereitschaft mir mit fachlichem Rat zur Seite zu stehen.

Außerdem möchte ich Herrn Dipl.-Ing. Martin Krey für seine Unterstützung danken, die maßgeblich zum gelingen dieser Arbeit beigetragen hat.

Ein besonderer Dank geht an meine Eltern, die mir die Möglichkeit gegeben haben mein Studium zu absolvieren und mich immer unterstützt haben.

# Inhaltsverzeichnis

# 1 Einführung

## 1.1 Einleitung

Das Antiblockiersystem, kurz ABS, kommt heutzutage millionenfach in Kraftfahrzeugen zum Einsatz und ist maßgeblich für die Sicherheit im Straßenverkehr mitverantwortlich. Damit das System seinen Anforderungen gerecht wird ist es wichtig, dass stets aktuelle Informationen über die Geschwindigkeit des Fahrzeugs, bestimmt durch die Drehzahl der einzelnen Räder, vorliegen. Die Informationen werden von einem Steuergerät ausgewertet. Bei einer Gefahrenbremsung wird dann das Blockieren der Räder verhindert und das Fahrzeug bleibt kontrollierbar.

Zur Ermittlung der Drehzahl, der Räder, kommen ABS-Sensoren zum Einsatz. Da diese in unmittelbarer Nähe zur Radnabe montiert werden sind sie extremen Bedingungen wie Schmutz, Temperatur und Vibration ausgesetzt. Sensoren, die auf dem anisotropen (von der Raumrichtung abhängig) magnetoresistiven (AMR) Effekt beruhen eignen sich hier, da sie berührungslos arbeiten.

Der AMR-Effekt beschreibt nach [6] die Änderung eines elektrischen Widerstandes in Abhängigkeit eines äußeren Magnetfeldes. Ist das Magnetfeld in bzw. gegen die Stromrichtung ausgerichtet, ist der Widerstand am größten. Wird das Magnetfeld hingegen senkrecht zur Stromrichtung angelegt, ist der elektrische Widerstand am kleinsten.

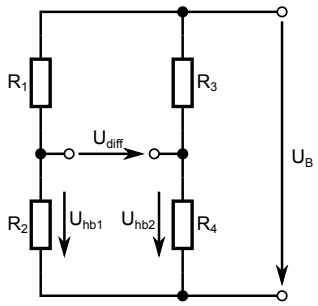



Abbildung 1.1: Wheatstonsche Messbrücke

Um die Änderung des Widerstandes nutzbar zu machen, sind typischerweise vier Widerstände, wie in Abbildung 1.1 dargestellt, zu einer Wheatstoneschen Messbrücke zusammen geschaltet.

Der AMR-Sensor ist mit einem Stützmagneten ausgestattet, welcher ein äußeres Magnetfeld hervorruft. Durch ein Encoderrad wird die Raumrichtung des Magnetfeldes innerhalb des Sensors beeinflusst und somit die Widerstände innerhalb des Sensors beeinflusst. Der Zusammenhang zwischen Encoderrad und der Brückenspannung $U_{diff}$ ist in Abbildung 1.2 dargestellt.

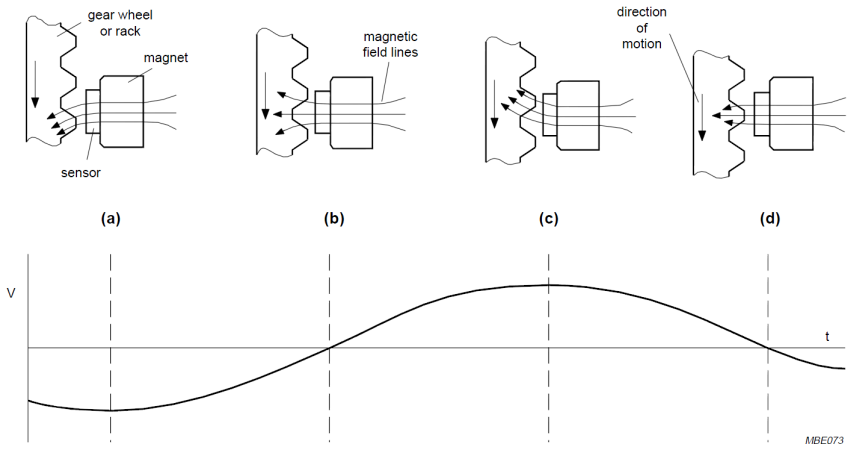



Abbildung 1.2: Zusammenhang zwischen Encoder und $U_{diff}$, Quelle:[16]

Die maximale Aussteuerung des Signals ist, bei konstanter Versorgungsspannung $U_B$ der Brücke, abhängig von der maximalen Änderung der Widerstände. Diese sind wiederum von der Ablenkung des Magnetfeldes abhängig. Ist die Distanz zwischen Encoderrad und Sensor gering, wird das Magnetfeld stärker abgelenkt als bei einer großen Distanz. Die Amplitude des Brückensignals ist somit direkt von der Distanz zwischen Encoder und Sensor abhängig.

Bedingt durch Fertigungstoleranzen können nicht alle vier Widerstände der Wheatstonschenbrücke exakt gleich hergestellt werden. Dies führt dazu, dass die Differenzspannung $U_{diff}$ einen konstanten, unerwünschten Gleichanteil aufweist. Zudem kann ein ungleichmäßig verteiltes Magnetfeld innerhalb des Sensors ebenfalls die Brücke verstimmen.

Da nicht davon ausgegangen werden kann, dass das Sensorsignal ideal ist, wie Abbildung 1.3 exemplarisch dargestellt, muss das Signal entsprechend aufbereitet werden. Dazu wird der Offset kompensiert und das Signal verstärkt. Anschließend kann anhand der Nulldurchgänge die Umdrehungsfrequenz des Encoderrads bestimmt werden. Im Rahmen des Forschungsprojekts ”Experimentelle digitale Signalverarbeitung und Zustandserkennung für ABS-Sensoren”, soll zudem durch eine Eigendiagnose des Sensors, seine Verlässlichkeit

erhöht werden [12]. Für dieses Vorhaben eignet sich nach [10] insbesondere eine Analyse der Harmonischen in Form des Klirrfaktors (THD).

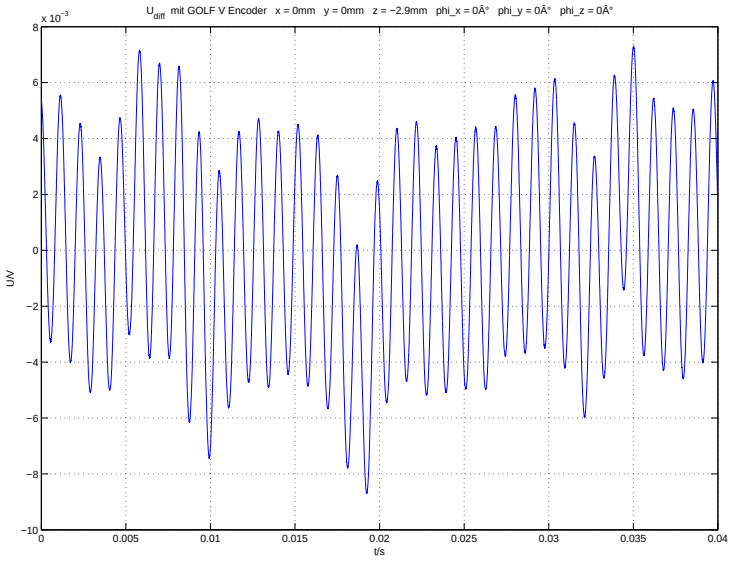



Abbildung 1.3: Stark unregelmäßiges Sensorsignal

## 1.2 Bestehendes System

In den vorangegangenen Arbeiten [10] und [19] wurde eine Demonstratorplattform geschaffen, die die wesentlichen Eigenschaften zur Nachbildung eines ABS-Sensors enthält. Außerdem wird das Signal der AMR-Messbrücke analysiert, um einen Indikator über den Zustand des Sensors zu gewinnen. In den Arbeiten [11] und [18] wurde entsprechende Verfahren zur Zustandserkennung untersucht.

### 1.2.1 Controllerplatine

Zur Aufbereitung der Spannungsversorgung und Sicherstellung der Stromversorgung für das gesamte System ist die Controllerplatine mit einem $+3V$ und einem $+5V$ Spannungsregler ausgestattet.

Um die Signalverarbeitung wie z.B. die Berechnung des THD und gleichzeitig die periodische Ausgabe des digitalen Strom-Ausgangssignals sicherzustellen, wurden diese Aufgaben auf zwei Mikrocontroller aufgeteilt. Der Signalcontroller ist für die Erkennung der

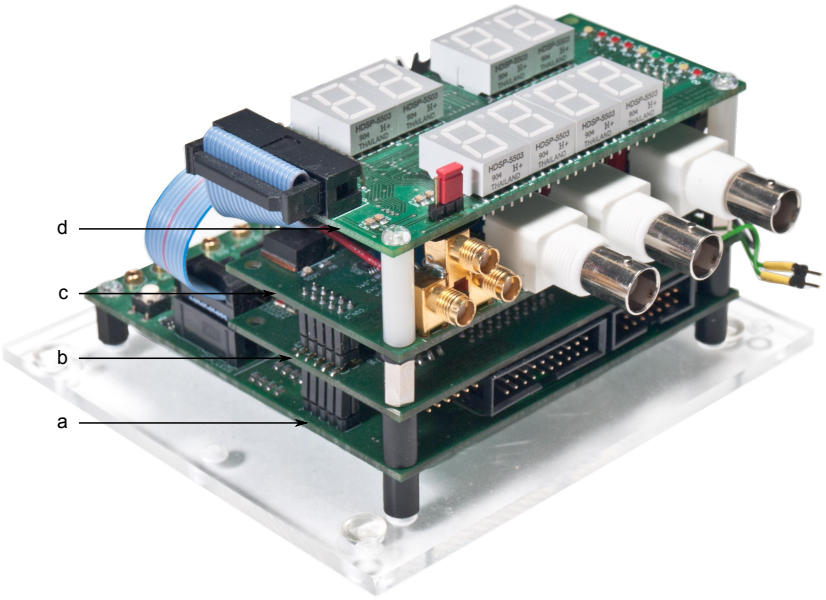



Abbildung 1.4: Bestehende Demonstratorplattform; a:Controllerplatine, b:Regelplatine, c:Verstärkerplatine, d:Displayplatine

Nulldurchgänge und die Berechnung des THD zuständig, während der Protokollcontroller die Ausgabe des digitalen Strom- Ausgangssignals übernimmt.

Zur Nachbildung des Übertragungsprotokolls nach [1] zwischen ABS-Sensor und ABS-Steuergerät, werden mehrere Konstantstromsenken eingesetzt. Neben einer Senke, die die Grundstromaufnahme des Sensors mit $7mA$ imitiert, besteht das System aus zwei weiteren schaltbaren Stromsenken mit jeweils $7mA$ und $14mA$ Stromaufnahmen.

Damit eine Kommunikation zwischen Demonstrator und PC stattfinden kann, ist die UART-Schnittstelle des Signalcontrollers auf eine Stiftleiste herausgeführt. Außerdem verfügt die Controllerplatine über zwei JTAG-Schnittstellen und eine Schnittstelle für die Displayplatine.

### 1.2.2 Regelplatine

Im Rahmen der Arbeit *Controllersystem zur Verstärkungsregelung und Offsetkompensation für ABS-Sensoren mit Diagnosefunktion* [19] von Martin Stahl ist die Regelplatine entstanden, welche die Demonstratorplattform [10] um eine zusätzliche Komponente erweitert.

Die Regelplatine, welche zwischen der Controllerplatine und der Verstärkerplatine in den Signalpfad eingebunden ist, verfügt über eine JTAG-Schnittstelle, eine UART-Schnittstelle

und eine Schnittstelle für die optionale Displayplatine. Außerdem befinden sich auf ihr ein MSP430 Mikrocontroller, eine Reset-Schaltung sowie mehrere Jumper um den Signalfluss zwischen der Controllerplatine und der Verstärkerplatine zu beeinflussen.

Da auf der ursprünglichen Demonstratorplattform zugunsten der Sensordiagnose die Offsetkompensation nur einmalig erfolgt und die Einstellung der Verstärkung händisch vorgenommen werde musste, wurde ein nebenläufiger Prozess entwickelt, der sowohl die Verstärkungsregelung als auch die Offsetkompensation übernimmt. Zur Offsetkompensation wird der Duty-Cycle der Brückendifferenzspannung $U_{diff}$ bestimmt und über einen der beiden DACs des Mikrocontrollers auf $50\,\%$ geregelt. Für die Verstärkungsregelung wird das Differenzsignal abgetastet und überprüft, ob das Signal in einem optimalen Aussteuerungsbereich liegt. Ist dies nicht der Fall, wird die Verstärkung auf der Verstärkerplatine entsprechend angepasst.

Zusätzlich kommuniziert der Regelcontroller mit dem Signalcontroller auf der Controllerplatine, um die nachfolgende Diagnosefunktion freizugeben. Befindet sich das System in der aktiven Regelung wird die weitere Signalverarbeitung gesperrt und erst wieder freigegeben, wenn das System ausgeregelt ist.

### 1.2.3 Verstärkerplatine

Auf der Verstärkerplatine werden die Signale, welche von der AMR-Messbrücke kommen für die weitere Signalverarbeitung aufbereitet. Dazu werden die beiden Halbbrückenspannungen $U_{hb1}$ und $U_{hb2}$ als Erstes von einem Tiefpassfilter 1. Ordnung, mit der Grenzfrequenz $f_g = 56{,}7\,\mathrm{kHz}$ gefiltert, um hochfrequente Störungen zu unterdrücken.

Die beiden Halbbrückenspannungen werden anschließend 25-fach verstärkt und an die weitere Signalverarbeitung weitergegeben. Da es durch Asymmetrien innerhalb der Messbrücke zu Offsets kommen kann, besteht die Möglichkeit, diese durch ein Potentiometer auszugleichen.

Zusätzlich wird die Differenz der beiden Halbbrückenspannungen gebildet. Dies geschieht über einen Differenzverstärker, der das Signal gleichzeitig noch um den Faktor 25 verstärkt.

$$U_{diff} = U_{hb1} - U_{hb2} \tag{1.1}$$

Da $U_{mathitDiff}$ ebenfalls Offset behaftet sein kann, muss dieser auch entsprechend kompensiert werden. Dazu wird mittels eines DAC (Digital Analog Umsetzer) des Regelcontrollers die Referenzspannung des nachfolgenden DVGA (Digital Variable Gain Amplifier)gesetzt. Durch den DVGA, dessen Verstärkung zwischen 1 und 128 eingestellt werden kann, wird das Signal weiter verstärkt. Somit beträgt die Verstärkung des Differenzsignals mindestens 25 und maximal 3200.

### 1.2.4 Displayplatine

Um diverse Parameter im laufenden Betrieb überprüfen zu können, kann der Demonstrator zusätzlich mit einer Displayplatine ausgestattet werden. Die Displayplatine stellt eine vierstellige und zwei zweistellige Siebensegmentanzeigen zur Visualisierung zur Verfügung.

Angesteuert werden kann die Displayplatine je nach Konfiguration, sowohl vom Regelcontroller, als auch vom Signalcontroller. Typischerweise wird die Displayplatine mit dem Signalcontroller verbunden, um relevante Systemparameter anzuzeigen. Dabei werden die Zahnfrequenz des Encoders auf der vierstelligen Anzeige visualisiert und der aktuelle Zustand des Zustandsautomaten des Signalcontrollers bzw. der THD des Brückendifferenzsignals auf den beiden zweistelligen Anzeigen.

## 1.3 Aufgabenstellung

In dieser Arbeit soll der lineare Verstärker der bestehenden Demonstratorplattform durch eine kombinierte logarithmisch und lineare Verstärkerbank ersetzt werden. Das Problem bei ABS-Sensoren, die auf dem AMR-Effekt beruhen ist, dass eine zweidimensionale Regelung des Sensorsignals erforderlich ist. Da zum einen der Herstellungsprozess und weitere physikalische Faktoren zu einer Verstimmung der Messbrücken führen, muss der dadurch entstehende Gleichspannungsanteil kompensiert werden. Zum anderen ist eine Regelung der Verstärkung notwendig, da die Signalgröße stark von der Einbaulage des Sensors abhängig ist. Durch den Einsatz der kombinierten logarithmisch und linearen Verstärkerbank soll die Regelung der Verstärkung entfallen. Sodass nur noch eine Offsetkompensation erforderlich ist.

Aus dem logarithmischen Signal soll zudem die Referenzfrequenz für die nachfolgende Signalverarbeitung gewonnen werden. Da bereits bei der Anfahrt des Fahrzeuges zuverlässige Informationen benötigt werden, müssen auch Signale von unter $1\,\mathrm{Hz}$ verarbeitet werden können. Dies ist problematisch, da herkömmliche Verfahren zur Gleichspannungsunterdrückung nicht zu Einsatz kommen können. Zudem müssen Signale mit einer Frequenz von bis zu $2{,}5\,\mathrm{kHz}$ verarbeitet werden können. Das Signal hat somit eine relativ große Bandbreite, die über mehrere Dekaden reicht.

Für die weitere Signalverarbeitung soll ein optimal ausgesteuertes, linear verstärktes Signal zur Verfügung stehen. Da durch die Verwendung einer Verstärkerbank gleichzeitig unterschiedlich stark verstärkte Signale zur Verfügung stehen, kann hieraus direkt ein geeignetes ausgewählt werden. Um eine Vergleichbarkeit zu dem bestehenden System zu schaffen, soll die Verstärkerbank so dimensioniert werden, dass diese ebenfalls eine Verstärkung von $25…3200$ abdeckt.

Für die Auswahl des linear verstärkten Signals, welches an die Signalverarbeitung weitergeleitet wird, sollen mehrere Auswahlmöglichkeiten untersucht werden. Eine Möglichkeit ist dabei die Abtastung aller durch die Verstärkerbank zur Verfügung stehenden Signale und Beurteilung anhand der Aussteuerungsgrenze. Eine weitere Möglichkeit ist die Auswahl der optimalen Verstärkung durch Analyse des logarithmierten Signals.

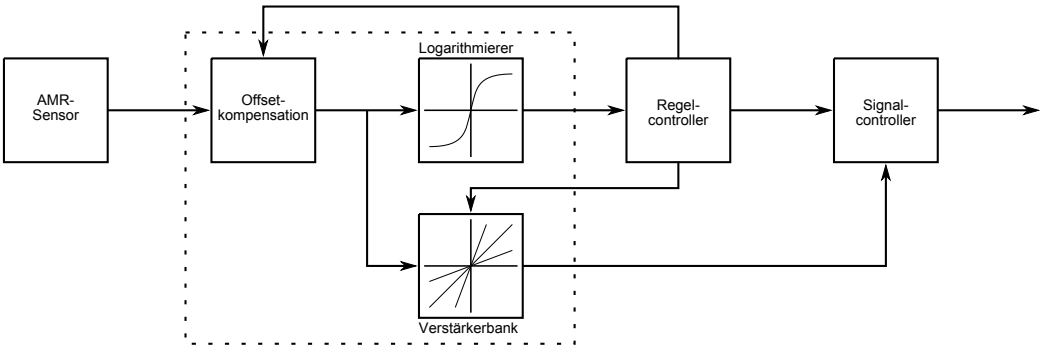



Abbildung 1.5: Schematisches Zusammenwirken der Grundfunktionalität

# 2 Analyse

## 2.1 Logarithmische Verstärkung

Damit die zweidimensionale Regelung des Eingangssignals, welche Offset- und Verstärkungsregelung umfasst, auf eine Dimension, die Offsetkompensation, reduziert werden kann, soll das Eingangssignal durch einen Logarithmierer vorverarbeitet werden.

### 2.1.1 Kontinuierlicher Logarithmierer

#### 2.1.1.1 Logarithmierer mit Diode

Wie in [20] beschrieben, kann die Shockley-Gleichung 2.1, welche die Kennlinie einer Diode beschreibt ausgenutzt werden, um ein Ausgangssignal zu erzeugen, welches proportional zum Logarithmus des Eingangssignals ist.

$$I_A = I_S \left( exp\left( \frac{U_{AK}}{n \cdot U_T} \right) - 1 \right) \tag{2.1}$$

Dabei wird der Diodenstrom $I_A$ durch den Sättigungsstrom $I_S$, der Temperaturspannung $U_T = \frac{k \cdot T}{e_0}$, der Anoden-Kathoden-Spannung $U_{AK}$ und einem Korrekturfaktor $n$ beschrieben. Ist der Diodenstrom viel größer als der Sättigungsstrom kann vereinfacht angenommen werden:

$$I_A = I_S \cdot \left( \exp \frac{U_{AK}}{n \cdot U_T} \right) \tag{2.2}$$

$$\Rightarrow U_{AK} = n \cdot U_T \cdot ln \left( \frac{I_A}{I_S} \right) \tag{2.3}$$

Um das logarithmische Ausgangssignal, welches vom Durchlassstrom $I_A$ der Diode abhängig ist, von einer Spannung abhängig zu machen, wird die Diode, wie in 2.1 dargestellt, in den Rückkopplungspfad eines invertierenden Verstärkers geschaltet. Damit ergibt sich für die Ausgangsspannung folgende Funktion:

$$U_a = -n \cdot U_T \cdot ln\left(\frac{U_e}{R_1 \cdot I_S}\right) \tag{2.4}$$

Der ausnutzbare Bereich dieser Schaltung wird dabei durch zwei Faktoren eingeschränkt. Zum einen besitzt die reale Diode einen Serienwiderstand, der dazu führt, dass die Kennlinie mit steigendem Strom $I_A$ linear ansteigt.

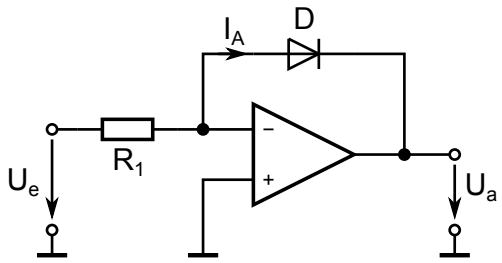



Abbildung 2.1: Logarithmierer mit Diode nach [20]

Zum anderen ist der Korrekturfaktor $n$ ebenfalls abhängig vom Durchflussstrom und verfälscht das Ergebnis zusätzlich. Eine hinreichende Genauigkeit der logarithmischen Kennlinie lässt sich nach [20] somit nur über ein bis zwei Dekaden der Eingangsspannung $U_e$ erreichen.

#### 2.1.1.2 Logarithmierer mit Transistor

Eine weitere Möglichkeit einen Logarithmierer zu realisieren, ist ebenfalls in [20] beschrieben. Dabei wird die Diode aus 2.1 durch einen Transistor, wie in Abbildung 2.2 dargestellt, ersetzt. Dadurch, das anstelle einer Diode ein Transistor eingesetzt wird, hat diese Schaltung den Vorteil, dass der Logarithmus unabhängig vom Korrekturfaktor $n$ ist. Mit dem Sättigungssperrstrom $I_{CS}$ des Transistors lässt sich das Ausgangssignal wie folgt beschreiben:

$$U_a = -U_T \cdot ln\left(\frac{U_e}{R_1 \cdot I_{CS}}\right) \tag{2.5}$$

Ein weiterer Vorteil dieser Schaltung ist, dass keine Verfälschung des Ergebnisses durch den Basis-Kollektor-Strom $I_{CB}$ auftritt, da die Basis-Kollektor-Spannung $U_{CB} = 0V$ ist. Außerdem geht die Stromverstärkung $B$ des Transistors nicht in das Ergebnis mit ein. Ein Nachteil der Schaltung ist allerdings, dass sie zum Schwingen neigt, da der Transistor die Schleifenverstärkung erhöht.

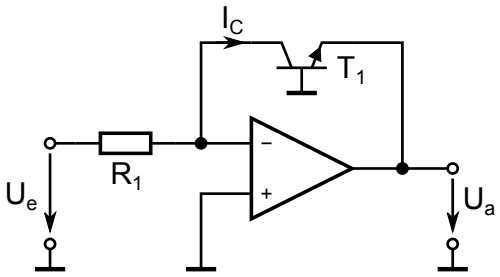



Abbildung 2.2: Logarithmierer mit Transistor nach [20]

#### 2.1.1.3 Zusammenfassung kontinuierlicher Logarithmierer

Wie gezeigt, kann mittels Diode bzw. Transistor relativ einfach ein Logarithmierer realisiert werden. Dieser hat allerdings den Nachteil, dass er durch die Temperaturspannung $U_T$ sehr empfindlich gegenüber Temperaturschwankungen ist. Ändert sich, wie in [20] beschrieben, die Temperatur beispielsweise von $20\,^{\circ}\mathrm{C}$ auf $50\,^{\circ}\mathrm{C}$, nimmt $U_T$ um $10\%$ zu. Der Sättigungssperrstrom des Transistors ist ebenfalls stark temperaturabhängig und verzehnfacht sich bei der gegebenen Temperaturänderung. Der Einfluss des Sperrstroms lässt sich allerdings durch Bilden der Differenz zweier Logarithmen eliminieren, siehe [20].

Ein weiterer Nachteil der beiden Schaltungen ist in Abbildung 2.3 dargestellt. Die beiden Schaltungen sind, wie der Logarithmus selbst, nur für positive Werte definiert und gehen bei negativen Eingangssignalen in die positive Begrenzung. Dies stellt bei der Auswertung von Wechselspannungen ein Problem da.

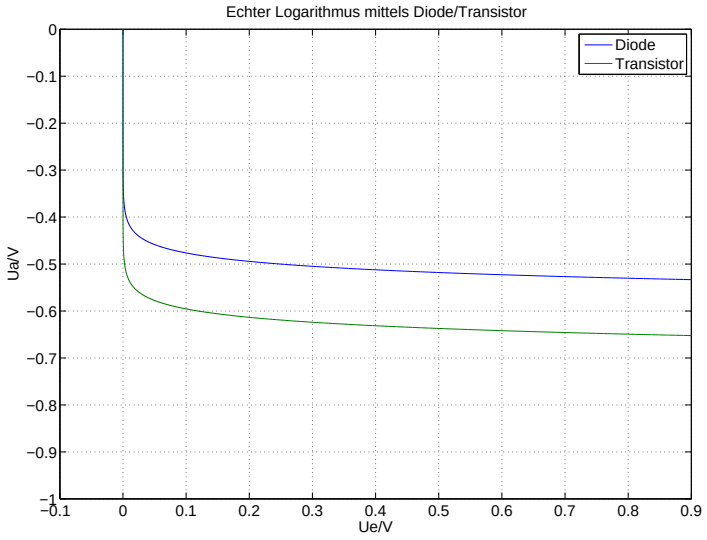



Abbildung 2.3: Kennlinien kontinuierlicher Logarithmierer

### 2.1.2 Stückweise lineare Approximation

Durch die stückweise lineare Approximation, wie in [14] beschrieben, ist es möglich den Logarithmus durch lineare Teilstücke nachzubilden.

Hierzu wird eine sogenannte Verstärkerbank aufgebaut, welche aus mehreren hintereinander geschalteten realen Verstärkern besteht. Auf eine alternative Parallelschaltung wird in Abschnitt 3.1.3 eingegangen. Die hintereinander geschalteten Verstärker haben dabei jeweils die Verstärkung $V$, ausgenommen dem ersten Verstärker, dessen Verstärkung $V-1$ beträgt. In Abbildung 2.4 ist der schematische Aufbau dargestellt.

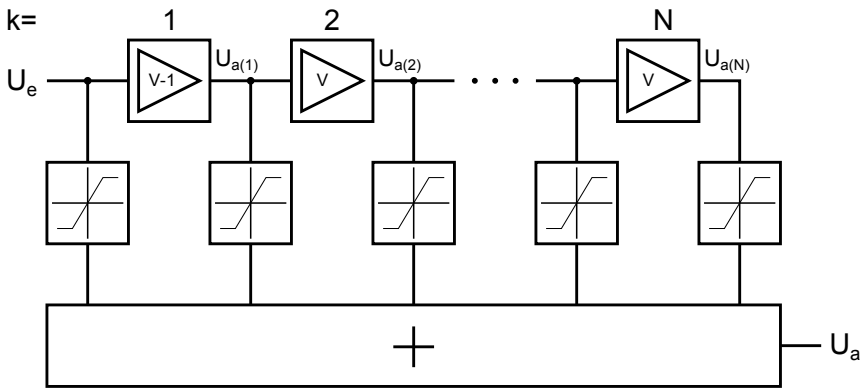



Abbildung 2.4: Schematischer Aufbau in Reihenschaltung

Reale Verstärker haben die Eigenschaft, dass sie nur bis zu einer maximalen Ausgangsspannung verstärken. Im Gegensatz zu idealen Verstärkern (vgl. Abbildung 2.5), die darüber hinaus weiterhin linear arbeiten, bleiben diese in der Sättigung. Die differenzielle Verstärkung wird dabei durch $\frac{\Delta U_a}{\Delta U_e}$ beschrieben. Geht der Verstärker in die Sättigung, so nimmt die Verstärkung entsprechend ab, da $_\Delta U_a$ klein wird.

Werden mehrere reale Verstärker wie beschrieben verschaltet, ist die Gesamtverstärkung des Systems durch die Anzahl der Stufen $N$ gegeben:

$$V_{ges} = (V-1) \cdot V^{N-1} \tag{2.6}$$

Dies gilt allerdings nur solange, bis der letzte Verstärker der Kette in die Sättigung geht.

Anders verhält es sich, wenn zusätzlich die Signale der einzelnen Stufen aufsummiert werden:

$$V_{ges} = 1 + \sum_{n=1}^{N} (V-1) \cdot V^{n-1} \tag{2.7}$$

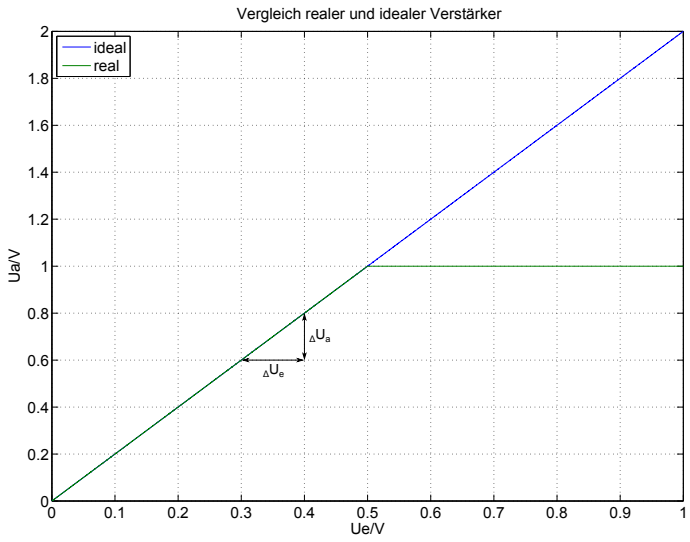



Abbildung 2.5: Vergleich idealer und realer Verstärker ($V = 2, U_{limit} = 1\,\mathrm{V}$)

Steigt das Eingangssignal in diesem Fall soweit an, dass die letzte Stufe in die Begrenzung geht, bleibt das System nicht in der Sättigung, sondern die Verstärkung des Systems nimmt um den letzten Summanden ab. Steigt das Eingangssignal weiter an, gehen nacheinander alle Verstärker in die Begrenzung. Die Verstärkung $V_{ges}$ kann dann anhand des Verstärkers $k$ bestimmt werden, welcher als Letztes in die Sättigung gegangen ist. Dabei sind die Verstärker, wie in Abbildung 2.4 dargestellt, der Reihe nach nummeriert.

$$V_{ges}(k) = 1 + \sum_{n=1}^{k-1} (V-1) \cdot V^{n-1} = V^{k-1} \tag{2.8}$$

Zudem kann daran die Funktion des Systems beschrieben werden. Die Verstärkung des Signals nimmt bei steigendem Eingangssignal, wie bereits beschrieben, ab. Für jede Stufe, die nicht mehr linear arbeitet, wird eine Konstante $U_{Limit}$ auf das Ausgangssignal aufaddiert. Diese entspricht der Höhe der Sättigung.

$$U_a(U_e,k) = U_e \cdot V^{k-1} + (N - k + 1) \cdot U_{Limit} \tag{2.9}$$

Zusätzlich kann anhand der Verstärkung die Ausgangsspannung der einzelnen Verstärkerstufen in Abhängigkeit der Eingangsspannung berechnet werden:

$$U_a(k) = (V-1) \cdot V^{k-1} \cdot U_e \tag{2.10}$$

Wenn die Ausgangsspannung der Stufen gleich der Konstanten $U_{Limit}$ gesetzt wird, kann daraus berechnet werden bei welchen Eingangsspannungen dieser Wert erreicht wird:

$$U_{e,s}(k) = \frac{U_{Limit}}{(V-1) \cdot V^{k-1}} \tag{2.11}$$

Werden die Spannungen $U_{Limit}$ und $U_{e,s}$ auf 1 V normiert, kann aus Gl. 2.11 der Wert $k$ berechnet werden:

$$V^{k-1} = \frac{U_{Limit,N}}{(V-1) \cdot U_{e,s,N}(k)}$$

$$k - 1 = \log_V \left( \frac{U_{Limit,N}}{(V-1) \cdot U_{e,s,N}(k)} \right)$$

$$k = log_V(U_{Limit,N}) - \log_V(U_{e,s,N}(k)) + \log_V(V-1) + 1 \tag{2.12}$$

Wird Gl. 2.12 in Gl. 2.9 eingesetzt und $U_a$ an den Stellen $U_e = U_{e,s}(k)$ berechnet, folgt daraus:

$$\begin{aligned} U_a\big|_{U_e=U_{e,s}(k)} \quad = \quad & U_{Limit}\left[\tfrac{1}{V-1} + N - \log_V(U_{Limit,N})\right. \\ & \left. + \log_V(V-1) + \log_V(U_{e,s,N}(k))\right] \end{aligned} \tag{2.13}$$

Anhand von Gl. 2.13 können die Werte des Ausgangssignals berechnet werden, bei denen die einzelnen Stufen gerade in die Begrenzung gehen. Betrachtet man die Formel genauer, wird ersichtlich, dass nur der letzte Term abhängig vom Eingangssignal ist und alle weiteren als eine Konstante zusammengefasst werden können:

$$\alpha = \frac{1}{V-1} + N - \log_V(U_{Limit,N}) + \log_V(V-1) \tag{2.14}$$

Die Funktion kann mit Hilfe der Logarithmengesetze wie folgt ausgedrückt werden:

$$U_a\big|_{U_e=U_{e,s}(k)} = U_{Limit} \cdot \log_V \left( V^{\alpha} \cdot \frac{U_{e,s}(k)}{1\,\mathrm{V}} \right) \tag{2.15}$$

Die Punkte, an denen die einzelnen Verstärker in die Sättigung gehen, können somit durch einen Logarithmus zur Basis $V$ beschrieben werden.

In Abbildung 2.6 ist das Ausgangssignal als Funktion des Eingangssignals der beschriebenen Verstärkeranordnung dargestellt. Dazu wurde das beschriebene Verhalten mittels MATLAB simuliert. Der entsprechende Quellcode befindet sich in Anhang D.2.1. Für

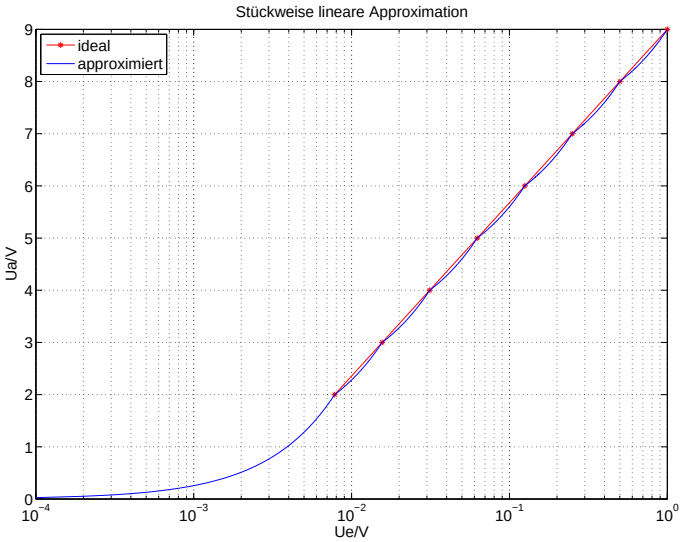



Abbildung 2.6: Stückweise lineare Approximation, eigene Simulation nach [14]

die Simulation wurde eine Verstärkung von $V = 2$ bei insgesamt $8$ Verstärkerstufen und $U_{Limit} = 1\,\mathrm{V}$gewählt.

Zudem sind die mit Gl. 2.15 berechneten Ausgangssignale in Abhängigkeit der Eingangssignale dargestellt, an denen die einzelnen Stufen in die Begrenzung gehen. Da diese in der logarithmischen Darstellung eine Gerade bildet, stellt sie eine logarithmische Funktion dar. Die Approximation durch die Verstärkerkette, welche entlang der Geraden verläuft und diese stellenweise berührt bildet somit eine gute Näherung der logarithmischen Funktion. Für Signale oberhalb des ersten Knickpunktes können die Ausgangssignale somit mit Gl. 2.16 berechnet werden.

$$U_a = U_{Limit} \cdot \log_V \left( V^{\alpha} \cdot \frac{U_e}{1\,\mathrm{V}} \right) \tag{2.16}$$

Die Näherung gilt allerdings nur, wie in Abbildung 2.6 zu sehen ist, für Bereiche oberhalb des ersten Knickpunktes. Unterhalb dieses Punktes wird das System durch einen linearen Zusammenhang beschrieben, welcher der maximalen Verstärkung des Systems entspricht und dafür sorgt, dass die Kennlinie durch den Ursprung läuft. In Abbildung 2.7 wird zudem eine weitere Eigenschaft der Approximation deutlich. Da die Verstärker nicht nur für positive Eingangssignale in die Begrenzung gehen, sondern auch für negative, ist die Kennlinie des Logarithmierers punktsymmetrisch. Somit ist dieses Verfahren auch für Wechselspannungssignale geeignet.

Für die Verarbeitung von bipolaren Signalen ist jedoch auch eine bipolare Versorgung notwendig. In einem Kraftfahrzeug steht nur eine unipolare Spannungsversorgung zur Verfügung, welches ein generelles Problem bei der Verarbeitung von negativen Signalen darstellt.

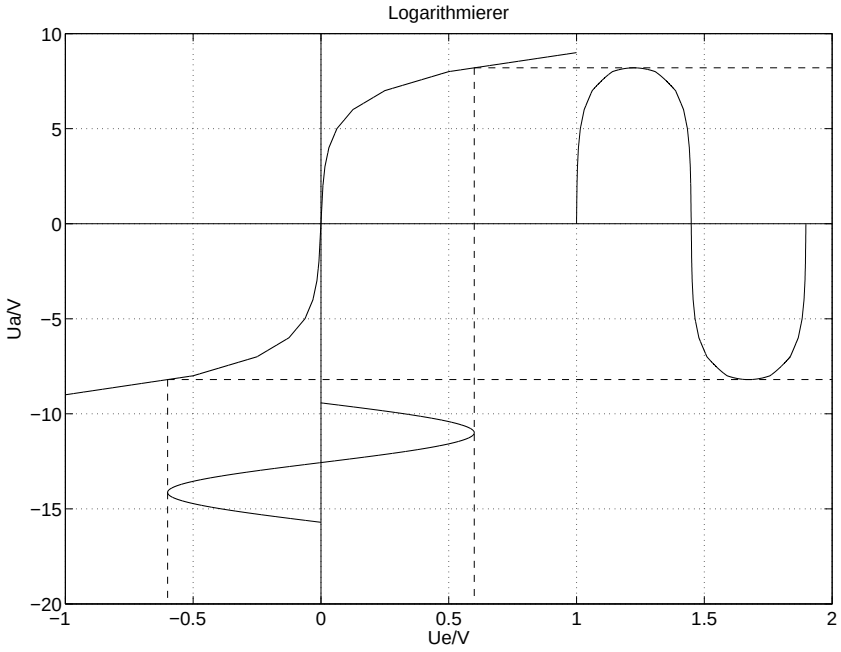



Abbildung 2.7: Logarithmierer Kennlinie

Eine Lösung dieses Problems ist die Verschiebung der Funktion in den ersten Quadranten. Dies bietet sich zudem an, da das Signal der AMR-Messbrücke auf der halben Betriebspannung der Brücke liegt. Die Verschiebung kann dadurch erfolgen, dass das Bezugspotential der Verstärker beim Schaltungsentwurf auf eine virtuelle Masse (halbe Brückenspannung) gelegt wird. Der Logarithmierer arbeitet dann wie in Abbildung 2.8 dargestellt.

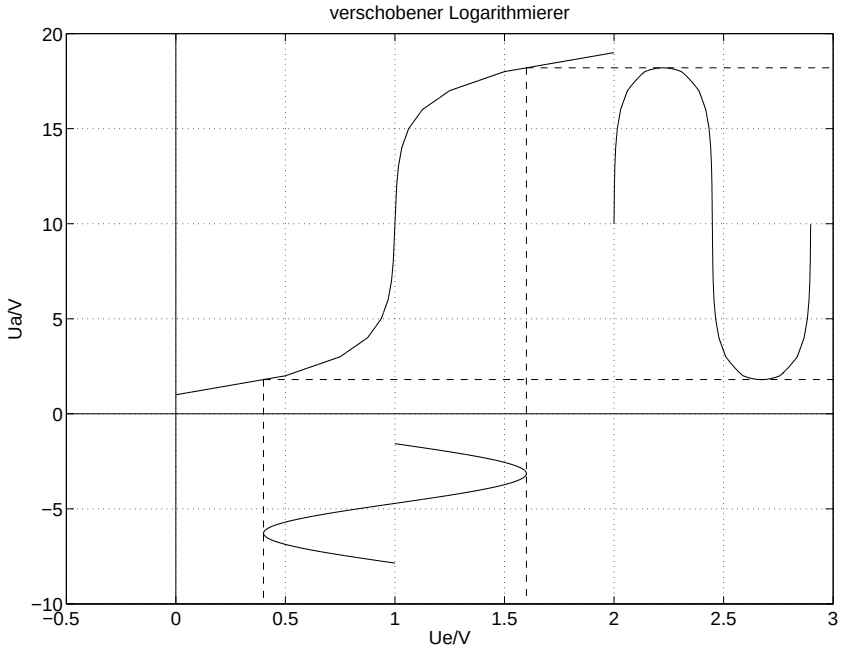



Abbildung 2.8: Kennlinie verschobener Logarithmierer

## 2.2 Delogarithmieren

In diesem Kapitel werden Möglichkeiten untersucht, um aus dem logarithmierten Signal die Eingangsgröße zurück zu gewinnen. Dabei wird zum einen eine analytische Methode und zum anderen die Speicherung der Umkehrfunktion in Form einer Lookup-Tabelle untersucht.

### 2.2.1 Analytische Umkehrfunktion

Unter der Annahme, dass durch die stückweise lineare Approximation ein Logarithmus beschrieben wird, kann das System, wie in Abbildung 2.9 zu sehen ist, vereinfacht dargestellt werden. Dabei dient $V_V$ der Vorstärkung des Signals und $V_N$ der Anpassung des logarithmischen Signals an den nachfolgenden ADC.

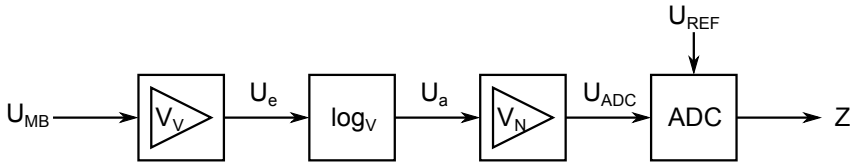



Abbildung 2.9: Schematische Systemdarstellung

Der digitalisierte Wert $Z$ lässt sich dann für gleichspannungsfreie positive Messbrückensignale $U_{MB}$ durch folgende Formel beschreiben:

$$Z = \text{round}\left(\frac{U_{ADC}}{U_{LSB}}\right) = \frac{U_{ADC}}{U_{LSB}} + q \tag{2.17}$$

Wird der Logarithmierer durch Gl. 2.16 beschrieben, kann Z, wie in Gl. 2.18 bzw. Gl. 2.19 dargestellt, gebildet werden.

$$Z = \frac{1}{U_{LSB}} \cdot V_N \cdot U_{Limit} \cdot log_V \left(V_V \cdot V^{\alpha} \cdot \frac{U_{MB}}{1\,\text{V}}\right) + q \tag{2.18}$$

$$Z = \frac{2^n}{U_{Ref}} \cdot V_N \cdot U_{Limit} \cdot log_V \left(V_V \cdot V^{\alpha} \cdot \frac{U_{MB}}{1\,\text{V}}\right) + q \tag{2.19}$$

Nach Gl. 2.20 kann, unter Vernachlässigung des Quantisierungsfehler $q$, das Eingangssignal aus dem digitalisierten Wert $Z$ zurück gewonnen werden.

$$U_{MB} = \frac{1\,\text{V}}{V_V \cdot V^{\alpha}} \cdot V^{\left(\frac{U_{Ref}}{2^n \cdot V_N \cdot U_{Limit}} \cdot Z\right)} \tag{2.20}$$

Findet die weitere Verarbeitung auf einem Mikrocontroller statt, so bietet sich ein Logarithmus zur Basis zwei an. Dadurch wird die Delogarithmierung zu einer $2^x$ Funktion, welche auf einem binären Rechensystem durch eine einfache Schiebeoperation gelöst werden kann.

$$x = \frac{U_{Ref}}{2^n \cdot V_N \cdot U_{Limit}} \cdot Z \tag{2.21}$$

Dies gilt allerdings nur, wenn $x$ ganzzahlig ist. Da $x$ im vorliegenden Fall von mehreren Parametern abhängig ist, müssen diese passend gewählt werden. Am einfachsten kann dabei die Verstärkung $V_N$ verändert werden. Dann ist allerdings nicht gewährleistet, dass der ADC optimal ausgesteuert wird. Eine Änderung der Referenzspannung hat den selben Effekt. Idealerweise wird die Spannung $U_{Limit}$ angepasst; da diese allerdings, wie später gezeigt,

ebenfalls nicht frei gewählt werden kann, stellt auch diese Möglichkeit eine unzureichende Lösung dar.

Ist $x$ nicht ganzzahlig kann die Delogarithmierung in die zwei Terme $2^{x_a+x_b} = 2^{x_a} \cdot 2^{x_b}$ geteilt werden. Dabei repräsentiert $x_a$ die Vorkommastelle und $x_b$ die Nachkommastelle. Der erste Teil des Terms, welcher somit ganzzahlig ist, lässt sich dann durch eine Schiebeoperation auf dem Mikrocontroller lösen. Da der zweite Term $2^{x_b}$ mit $x_b < 1$ eine Wurzelfunktion darstellt und die Lösung einer entsprechenden Funktion auf einem Mikrocontroller sehr aufwendig ist, bietet sich als Alternative an dieser Stelle die Verwendung einer Lookup-Tabelle an.

### 2.2.2 Delogarithmieren durch Lookup-Tabelle

Eine weitere Möglichkeit, um aus dem logarithmierten Signal auf das Eingangssignal $U_e$ zu schließen ist, die Funktion des System zu messen und die Umkehrfunktion in Form einer Lookup-Tabelle auf dem Regelcontroller zu speichern. Dazu wird das System mit einem Signal gespeist, welches am Ausgang $U_{ADC}$ den gesamten Spannungsbereich des ADCs von $0$ bis $2{,}5\,\mathrm{V}$ abdeckt. Die beiden Signale $U_e$ und $U_{ADC}$ werden dabei in digitaler Form gespeichert. Zusätzlich werden die theoretischen Spannungswerte berechnet, welche der ADC digitalisieren kann $\{0, U_{LSB}, 2 \cdot U_{LSB}, ..., (2^n - 1) \cdot U_{LSB}\}$. Anschließend wird die Differenz zwischen den gemessenen Eingangswerten des ADCs am Ende des Systems und einem theoretischem Wert ermittelt. Für das Minimum des Betrags der Differenz wird der entsprechende Eingangswert des Systems ermittelt und als Wert in einer Lookup-Tabelle gespeichert. Dieser Vorgang wird für alle theoretischen Werte wiederholt. Für die Erstellung der Lookup-Tabelle wurde ein MATLAB Skript entwickelt, welches diesen Vorgang automatisiert. Der entsprechende Quellcode ist in Anhang D.2.4 beigefügt. Diese Methode hat zudem den Vorteil, dass auch im unteren Spannungsbereich, in denen das System linear arbeitet, auf die richtigen Eingangswerte geschlossen werden kann.

## 2.3 Offsetkompensation

Die Wheatstonsche Messbrücke besteht, wie in Abbildung 1.1 dargestellt, aus vier Widerständen. Mithilfe dieser Anordnung können Widerstandsänderungen in Spannungsänderungen überführt werden. Der Zusammenhang zwischen Widerstand und Spannung lässt sich durch 2.22 beschreiben.

$$U_{diff} = U_B \cdot \left( \frac{R_2}{R_1 + R_2} - \frac{R_4}{R_3 + R_4} \right) \tag{2.22}$$

Sind die Widerstände $R_1 ... R_4$ gleich gewählt, so ist die Differenzspannung der Brücke $U_{diff} = 0\,\mathrm{V}$. Ändert sich einer der Widerstände, so ändert sich gemäß 2.22 auch die Brückenspannung.

Der AMR-Sensor besteht aus vier vom Magnetfeld abhängigen Widerständen. Dabei sind diese, wie in [17] beschrieben, so entworfen, dass die Widerstände $R_1$ und $R_4$ als auch $R_2$ und $R_3$ in gleicher Weise vom Magnetfeld abhängen. Durch diese Eigenschaft wird die Aussteuerung von $U_{diff}$ verdoppelt.

Der Einsatz einer Messbrücke hat zudem den Vorteil, dass die Temperaturempfindlichkeit des Sensors reduziert wird. Da sich die einzelnen Widerstände relativ nahe beieinander befinden, kann davon ausgegangen werden, dass diese thermisch gekoppelt sind. Das heißt, wenn die Widerstände exakt gleich groß sind, wirkt sich eine temperaturbedingte Widerstandsänderung auf alle vier Widerstände gleich aus. Somit ist die Abgleichbedingung der Messbrücke nach Gl. 2.23 weiterhin erfüllt.

$$\frac{R_1}{R_2} = \frac{R_3}{R_4} \tag{2.23}$$

Da die vier Widerstände des Sensors nicht beliebig genau produziert werden können, existieren, wie in [12] beschrieben, innerhalb des Sensors zwei Trimmstrukturen, um die Messbrücke abzugleichen. Mithilfe der Trimmstrukturen kann der elektrische Offset der Messbrücke minimiert werden. Dieser kann allerdings nicht vollständig durch die Trimmung eliminiert werden. Daher ist auch die Gl. 2.23 nicht erfüllt, wodurch sich eine temperaturbedingte Widerstandsänderung als Offset auswirkt.

Zu dem elektrischen Offset kommt nach [12] weiterhin ein magnetischer Offset. In unmittelbarer Nähe der Messbrücke ist ein Stützmagnet montiert, welcher ein Magnetfeld hervorruft. Dieses Feld durchflutet die Messbrücke und stellt den Arbeitspunkt der vom Magnetfeld abhängigen Widerstände ein. Wird durch den Magneten ein ungleichmäßiges Magnetfeld hervorgerufen, bzw. die vier Brückenwiderstände ungleich von dem Magnetfeld durchflutet, sorgt dies dafür, dass die Widerstände leicht unterschiedliche Werte haben. Nach 2.22 führt dies ebenfalls zu einem unerwünschten Gleichanteil der Brückendifferenzspannung $U_{diff}$.

### 2.3.1 Offsetkompensation durch Wechselstromkopplung

Eine Möglichkeit um ein, von Gleichspannung freies Signal zu erzeugen ist, den Gleichanteil herauszufiltern. Dies kann mithilfe eines RC-Hochpasses geschehen. Da der Widerstand des Kondensators von der Frequenz abhängig ist und mit steigender Frequenz abnimmt, werden niederfrequente Signale praktisch nicht durchgelassen. Hochfrequente Signale werden dagegen verhältnismäßig gering beeinflusst.

Die Sensoren sind nach [15] für einen Frequenzbereich von $0\,\mathrm{Hz}$ bis $2{,}5\,\mathrm{kHz}$ spezifiziert. Das heißt, dass diese Signale vom Hochpass nicht beeinflusst werden dürfen. Da bereits die Entwicklung eines Hochpasses mit einer Grenzfrequenz von $f_g = 1\,\mathrm{Hz}$ große Kapazitäten benötigt, welche auf Chipebene nicht realisierbar sind, kommt auch eine Realisierung eines Hochpasses mit einer Grenzfrequenz von $f_g < 1\,\mathrm{Hz}$ nicht infrage.

### 2.3.2 Offsetkompensation mittels Duty-Cycle

In den Arbeiten [10] und [19] wurde gezeigt, dass sich die Offsetkompensation anhand des Duty-Cycle als Verfahren gut eignet. Gerade in Hinblick auf einen Chipentwurf hat dieses Verfahren Vorteile, da es digital arbeitet und somit eine Integration vereinfacht.

Der Duty-Cycle $\delta$ beschreibt das Verhältnis zwischen Impulsdauer $t_p$ und Periodendauer $T$. Bei einem sinusförmigen Signal (vgl. Abbildung 2.10) werden diese beiden Parameter anhand der Nulldurchgänge bestimmt. Die Zeit vom ersten positiven Nulldurchgang bis zum darauffolgenden Nulldurchgang wird dabei als Impulsdauer gewertet. Die Zeit zwischen zwei positiven Nulldurchgängen beschreibt die Periodendauer.

$$\delta = \frac{t_p}{T} \cdot 100\% \tag{2.24}$$

Sind die zeitlichen Abstände zwischen den einzelnen Nulldurchgängen gleich, so beträgt das Tastverhältnis des Signals $t_p = 50\,\%$ und das Signal ist offsetfrei. Ist der Abstand ungleich, so weist das Signal einen Gleichanteil auf und das Tastverhältnis ist entsprechend $t_p \neq 50\,\%$.

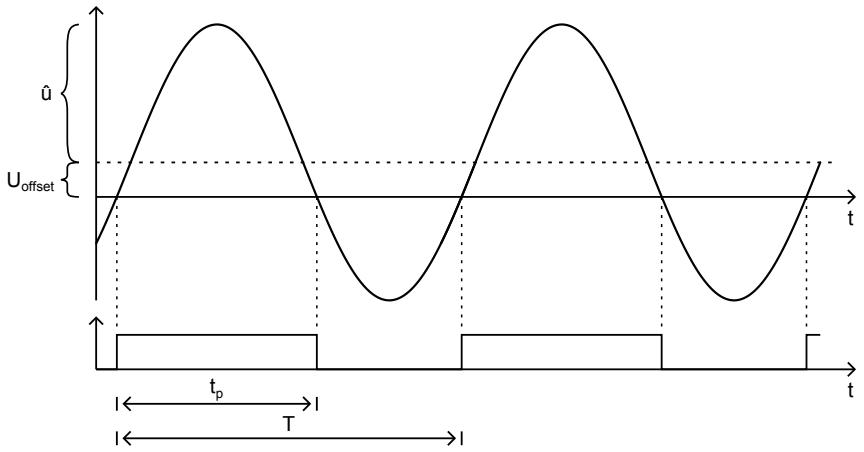



Abbildung 2.10: Ermittlung des Tastverhältnisses, schematisch nach [10]

Für sinusförmige Signale kann mithilfe der Sinusfunktion aus dem Tastgrad direkt der Offset bestimmt werden:

$$U_{offset} = \sin\left(\pi \frac{\delta}{100\%} - \frac{\pi}{2}\right) \tag{2.25}$$

Durch die Näherung wird die Logarithmus Funktion im Ursprung durch eine Gerade beschrieben:

$$f(x) = m \cdot x \tag{2.26}$$

Wodurch $f(x) = 0$ für $x = 0$ ist. D.h., dass die logarithmierte Funktion zu denselben Zeitpunkten die Zeitachse schneidet wie die nicht logarithmierte Funktion. Für die Ermittlung des Tastverhältnisses $\delta$ wird, wie beschrieben, die verstrichene Zeit zwischen den Nulldurchgängen ausgewertet. Da die Zeitpunkte der Nulldurchgänge durch das Logarithmieren nicht beeinflusst werden, wird auch das Tastverhältnis nicht beeinflusst.

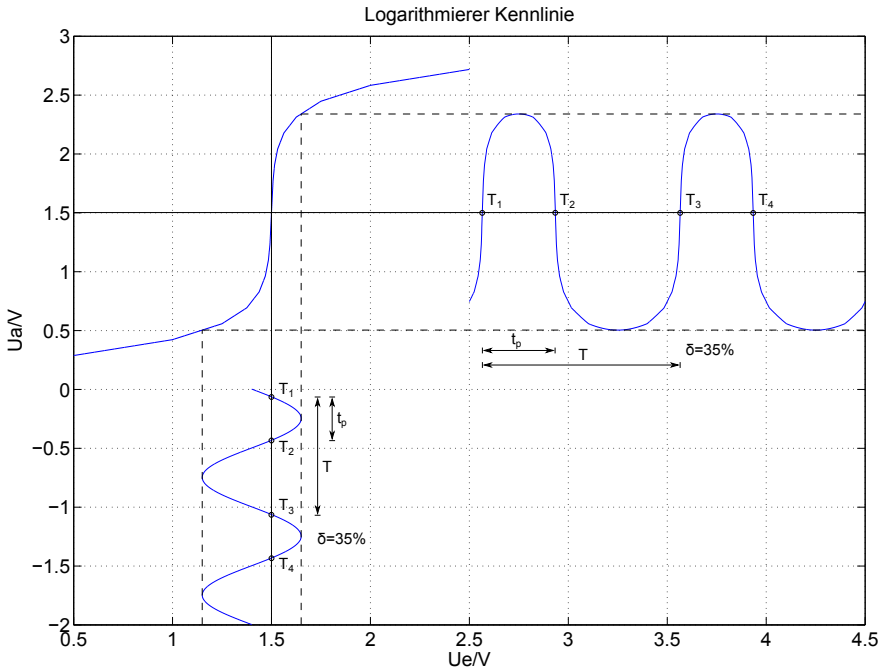



Abbildung 2.11: Logarithmierer mit Duty-Cycle erhaltendes Zeitverhalten

Da der Offset, wie in Kapitel 3.7 beschrieben korrigiert wird, bevor das Signal verstärkt wird, muss aus dem logarithmierten Signal der Gleichspannungsanteil des Eingangssignals bestimmt werden. Aus dem Tastverhältnis des logarithmierten Signals kann allerdings nicht direkt auf den Offset des Eingangssignals geschlossen werden. Dazu muss zusätzlich die Amplitude des Eingangssignals bekannt sein. Stehen beide Informationen zur Verfügung, kann der Offset gemäß Gl. 2.27 berechnet werden. Dabei kann die Amplitude anhand des

Minimums und Maximums bestimmt werden, welche sich am einfachsten detektieren lassen.

$$U_{offset} = \frac{U_{max} - U_{min}}{2} \cdot \sin\left(\pi \frac{\delta}{100\%} - \frac{\pi}{2}\right) \tag{2.27}$$

## 2.4 Verstärkungsauswahl

Die Widerstandsänderung der AMR-Messbrücke ist von der Richtung und der Stärke des Magnetfelds abhängig, welches den Sensor durchdringt. Dabei wird das Feld stärker beeinflusst, wenn sich das Encoderrad unmittelbar vor dem Sensor befindet. Wird die Entfernung größer, nimmt der Einfluss des Encoders auf das Magnetfeld ab und die Widerstandsänderung wird geringer. Die Amplitude des Ausgangssignals der AMR-Messbrücke ist also direkt abhängig von der Entfernung zwischen Sensor und Encoder. In [10] wurde gezeigt, dass sich die Änderung des Signals bei einer Distanz zwischen $0$ und $6\,\mathrm{mm}$ über drei Dekaden erstreckt. Somit ist es unerlässlich, die Verstärkung des Eingangssignals an den Aussteuerungsbereich des ADCs anzupassen.

### 2.4.1 Verstärkungsregelung bei linearer Verstärkung

In [19] wurde eine Verstärkungsregelung für einen linearen Verstärker implementiert. Dazu wurden für die obere und untere Halbwelle jeweils zwei Schwellwerte festgelegt, innerhalb denen sich die Maxima bzw. Minima des Signals bewegen müssen. Zusätzlich wird das Signal durch einen freilaufenden ADC ständig abgetastet und am Ende einer Periode überprüft, wie oft die einzelnen Schwellen innerhalb einer Periode überschritten wurden. Wurde keine der Schwellen überschritten, so ist das Signal nicht ausreichend verstärkt. Bei Überschreitung der unteren Schwelle und keiner Überschreitung der oberen Schwelle ist das Signal optimal ausgesteuert. Wird allerdings zusätzlich zu der unteren auch die obere Schwelle überschritten, so ist das Signal übersteuert und die Verstärkung muss verringert werden. Bei dieser Art der Regelung muss die Abtastfrequenz allerdings deutlich höher gewählt werden als es nach dem Abtasttheorem $f_{sample} > 2 \cdot f_{sig,max}$ nötig wäre. Würden nur zwei Abtastwerte pro Periode aufgenommen, so ist nicht sichergestellt, dass das Minimum bzw. Maximum detektiert und beispielsweise ein Übersteuern festgestellt wird.

### 2.4.2 Verstärkerauswahl durch schnelle Abtastung

Stehen durch eine Verstärkerbank mehrere linear verstärkte Signale zur Verfügung, kann anhand der Aussteuerung der einzelnen Signale die optimale Verstärkung ausgewählt werden. Dazu werden alle Signale abgetastet und digital ausgewertet. Ist das Signal zu groß, so

führt dies zu einem Übersteuern des ADCs bzw. zu einem Überschreiten eines definierten Schwellwertes und das Signal ist für die weitere Signalanalyse ungeeignet. Das Signal mit der größten Aussteuerung, welches nicht zum Übersteuern des ADCs geführt hat, stellt das optimale Signal zur weiteren Verarbeitung dar.

Wie in [19] gezeigt, ist bei einer maximalen Signalfrequenz von $2{,}5\,\mathrm{kHz}$ eine minimale Abtastfrequenz von $17{,}4\,\mathrm{kHz}$ notwendig, wenn Überschreitungen von $90\,\%$ der maximal zulässigen Spannung detektiert werden sollen. Steht zur Digitalisierung nur ein ADC zur Verfügung, wie es im MSP430F1611, welcher als Regelcontroller dient, der Fall ist, so muss die Abtastfrequenz für jeden zusätzlichen Kanal, der abgetastet werden soll, um $17{,}4\,\mathrm{kHz}$ erhöht werden. Sollen also beispielweise acht Signale detektiert werden, so ist eine Abtastfrequenz von $f_{sample} = 139{,}2\,\mathrm{kHz}$ notwendig. Die maximale Abtastfrequenz bei Verwendung des internen Oszillators des MSPs wurde in [19] zu $f_{sample,max} = 77{,}1\,\mathrm{kHz}$ bestimmt und ist somit nicht schnell genug, um acht Kanäle mit der notwendigen Abtastfrequenz abzutasten.

### 2.4.3 Verstärkerauswahl durch Delogarithmierung

Ein anderer Ansatz die optimale Verstärkung auszuwählen ist, aus dem logarithmierten Signal die Amplitude des Eingangssignals zu ermitteln. Ist weiterhin bekannt, bei welcher Eingangsspannung die einzelnen Verstärkerstufen in die Begrenzung gehen und somit das Signal verzerren, kann aus diesen beiden Parametern die maximal zulässige Verstärkung des Eingangssignals bestimmt werden, ohne dass dieses durch die Verstärker verzerrt wird.

# 3 Schaltungsentwurf und Simulation

Im Folgenden wird auf die Umsetzung der geforderten Verstärkerschaltung eingegangen. Dabei werden insbesondere Reihen- und Parallelschaltung von Verstärkern zur Realisierung der linearen Approximation näher betrachtet und gegenübergestellt.

## 3.1 Logarithmischer Verstärker

Um die in Kapitel 2.1.2 beschriebene stückweise lineare Approximation zu realisieren, gibt es zwei Möglichkeiten. Zum einen kann die Verstärkerkette aus hintereinander geschalteten Verstärkern mit der gleichen Verstärkung realisiert werden. Zum anderen können mehrere Verstärkerstufen parallel geschaltet werden und die Verstärkung jeweils um den Faktor V vergrößert werden.

### 3.1.1 Begrenzungsverzerrung von Operationsverstärkern

Um die Verstärkerstufen zu realisieren bieten sich Operationsverstärker an. Diese zählen zu den Standardbauelementen und sind in vielen Ausführungen und Variationen auf dem Markt erhältlich. Zudem können sie durch äußere Beschaltung in ihren Funktionen beeinflusst werden, weshalb sie sich gut für einen experimentellen Aufbau eignen.

Wichtig für die Approximation der logarithmischen Kennlinie sind die Eigenschaften der Verstärker, wenn diese in die Sättigung gehen, da diese Eigenschaft der Approximation zu Grunde liegt.

Um eine Aussage über die Begrenzungszerrung von Operationsverstärkern zu machen, wurden mehrere Typen untersucht, die den Anforderungen entsprechen. Dabei ist entscheidend, dass die OPs mit einer unipolaren Spannungsversorgung arbeiten, da in Kraftfahrzeugen nur eine positive Versorgungsspannung zur Verfügung steht. Des weiteren müssen die Verstärker ein ausreichendes Verstärkungs-Bandbreite-Produkt aufweisen, damit höherfrequente Signalanteile nicht durch den Verstärker gedämpft werden. Dies würde zu einer Verfälschung des THD bei der späteren Signalanalyse führen. Eine weitere wichtige Eigenschaft ist ein geringer Gleichspannungs-Offset der Verstärker. Da das gesamte System empfindlich

| Bezeichnung | Offset | GBW | Slew Rate |
|---|---|---|---|
| AD4051 | $15\mu V$ | $125kHz$ | $0.06V/\mu s$ |
| AD4528 | $2{,}5\mu V$ | $4MHz$ | $0.45V/\mu s$ |

Tabelle 3.1: Liste der untersuchten Verstärker

auf Offsets reagiert, sind zusätzliche Gleichspannungsanteile zu vermeiden. Eine Liste der untersuchten Verstärker ist in Tabelle 3.1 aufgelistet.

Für die Untersuchung der Begrenzungsverzerrung wurden die Operationsverstärker in nichtinvertierender Schaltung betrieben. Dabei wurde eine Verstärkung von Zwei und als Betriebsspannung $3V$ gewählt. Zudem wurde eine virtuelle Masse bei der halben Betriebsspannung festgelegt. Als Eingangssignal diente ein Sinussignal mit $\hat{u} = 0.9V$ Amplitude bei einer Signalfrequenz von $f = 1kHz$.

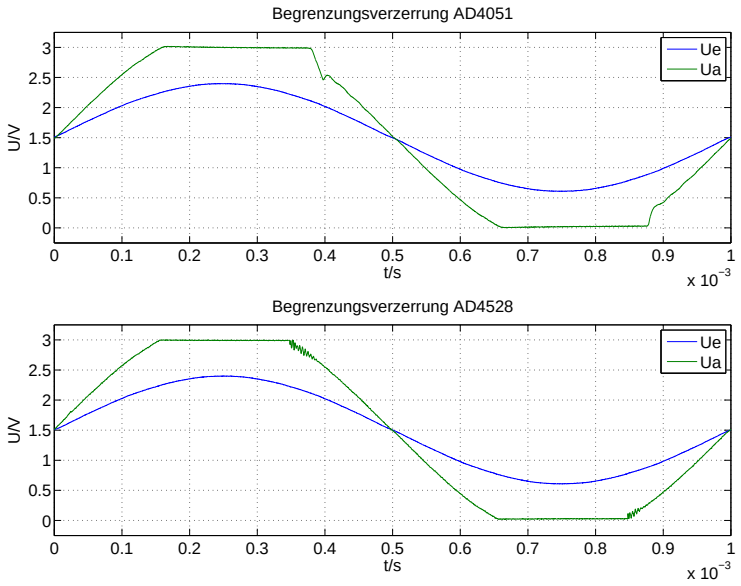



Abbildung 3.1: OP Begrenzungsverzerrung

Wie in Abbildung 3.1 zu sehen, verzerren die Operationsverstärker das Signal deutlich, wenn es aus der Sättigung zurückkommt. Mit steigender Bandbreite der OPs wird dieses Phänomen zwar kleiner, ist für die Approximation allerdings nicht zu vernachlässigen. Zudem neigen Verstärker mit großer Bandbreite eher zum Schwingen.

### 3.1.2 Reihenschaltung von Verstärkern

Eine Möglichkeit die Verstärkerkette für die stückweise lineare Approximation zu realisieren, ist die Reihenschaltung von Verstärkern. Dabei werden die einzelnen Verstärker so zusammengeschaltet, dass der Ausgang des Vorherigen, den Eingang des darauf Folgenden speist. Die Ausgänge der einzelnen Stufen, sowie das Eingangssignal selbst, werden durch Addition zu dem Ausgangssignal zusammengefasst. Der Aufbau ist schematisch in Abbildung 3.2 dargestellt.

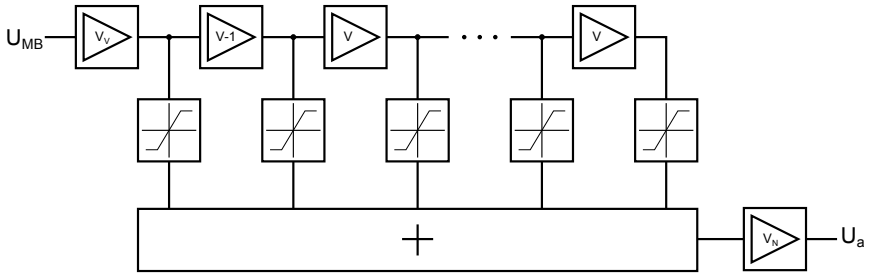



Abbildung 3.2: Approximation durch Reihenschaltung

Damit alle Signale, die aufsummiert werden, die gleiche Phasenlage haben, müssen die Verstärkerstufen nicht invertierend sein. Das beschriebene Problem der Begrenzungsverzerrung kann, bei nicht invertierenden Verstärkern, durch das Begrenzen des Eingangssignals mittels Dioden, umgangen werden. Somit findet das Abknicken der Kennlinie nicht durch den eigentlichen Verstärker sondern bereits davor statt. In Abbildung 3.3 ist die verwendete Schaltung dargestellt. Hierbei ist die Verstärkung allerdings so zu wählen, dass das maximale Eingangssignal des Verstärkers nicht zu dessen Übersteuerung führt. Hat der Verstärker beispielweise eine maximale Betriebspannung von $3\,\mathrm{V}$, eine Verstärkung von $2$ und eine virtuelle Masse bei $1{,}5\,\mathrm{V}$, so darf das maximale Eingangssignal $\pm 0{,}75\,\mathrm{V}$ zur virtuellen Masse nicht überschreiten. Wird das Eingangssignal durch eine Diode mit $U_f = 0{,}7\,\mathrm{V}$ begrenzt, wird diese Bedingung erfüllt.

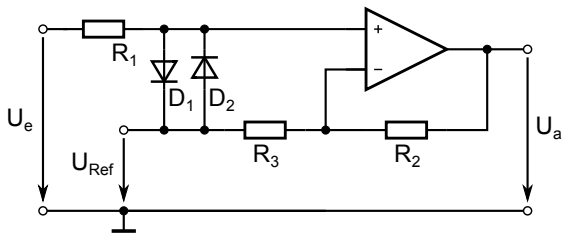



Abbildung 3.3: Nichtinvertierender Verstärker mit Begrenzung der OP Eingangsspannung

Diese Anordnung hat den Vorteil, dass alle Verstärkerstufen exakt gleich aufgebaut werden können. In Hinblick auf eine Chipintegration muss somit nur eine Verstärkerstufe entworfen werden, welche dann mehrfach auf dem Chip integriert werden kann. Ein Nachteil dieser Anordnung ist, dass das Signal durch jede Stufe ggf. zeitlich verzögert wird. Dies kann, wie in Kapitel 3.1.6 beschrieben, zu einer Verzerrung des Signals führen.

### 3.1.3 Parallelschaltung von Verstärkern

Die Reihenschaltung der Verstärkerkette kann in eine Parallelschaltung überführt werden. Dabei werden, wie in Abbildung 3.4 illustriert, mehrere Stufen mit unterschiedlicher Verstärkung verwendet. Die Verstärkung der einzelnen Stufen, mit Ausnahme der ersten Stufe, ist dabei um den Faktor $V$ gestaffelt. Die einzelnen Stufen werden alle mit demselben Eingangssignal gespeist und die Summe der Ausgangssignale zusammen mit dem Eingangssignal bilden den Logarithmus des Eingangssignals.

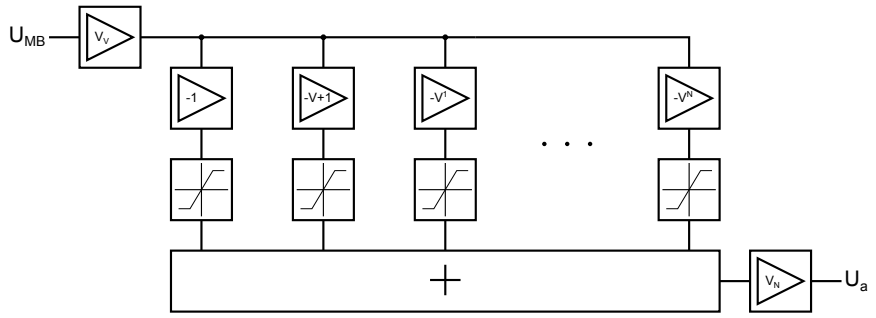



Abbildung 3.4: Approximation durch Parallelschaltung

Vorteil dieser Variante ist, dass das Signal nicht durch eine vorherige Stufe zeitlich verzögert wird, sondern dass alle Stufen mit demselben Signal gespeist werden. Ein Nachteil dieser Anordnung ist allerdings, dass alle Stufen eine unterschiedliche Verstärkung aufweisen und somit mehrere Verstärker realisiert werden müssen, da nicht auf einen Einzelnen zurück gegriffen werden kann.

Zudem wächst die Verstärkung der einzelnen Stufen sehr schnell an. Wird beispielsweise ein Logarithmierer zur Basis zwei mit acht Stufen entworfen, so ist die Verstärkung der letzten Stufe gegenüber der ersten bereits um den Faktor $128$ größer.

Da in diesem Fall alle Verstärker eine unterschiedliche Verstärkung haben, kann die Begrenzungsverzerrung der Operationsverstärker nicht dadurch umgangen werden, dass das Eingangssignal mittels Dioden beschränkt wird. Hat der Verstärker eine Verstärkung von beispielsweise $128$ und arbeitet mit einer Betriebsspannung von $3\,\mathrm{V}$, so darf das Eingangssignal maximal $\pm 11{,}7\,\mathrm{mV}$ relativ zur virtuellen Masse nicht überschreiten. Dioden haben

allerdings eine Flussspannung von einigen Hundert Millivolt und sind somit ungeeignet für diesen Anwendungsfall.

Eine weitere Möglichkeit sicherzustellen, dass der Verstärker nicht in die Sättigung geht, ist, dessen Verstärkung bei großen Eingangssignalen zu reduzieren. Dazu kommt ein invertierender Verstärker zu Einsatz, zu dessen Rückkopplungswiderstand eine Diode parallel geschaltet wird. Steigt die Spannung am Ausgang des Verstärkers nun soweit an, dass die Diode zu leiten beginnt, wird der Rückkopplungswiderstand kleiner. Dadurch sinkt die Verstärkung und der Verstärker geht nicht in die Sättigung. Der Aufbau ist in Abbildung 3.5 dargestellt.

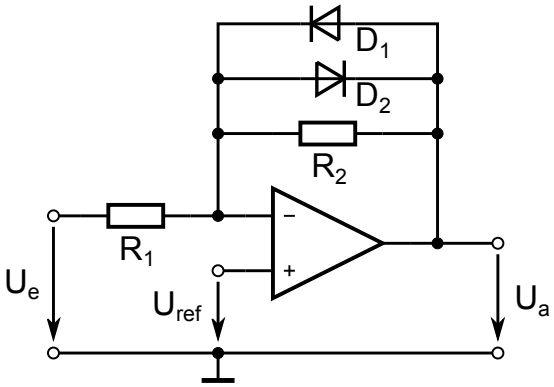



Abbildung 3.5: Invertierender Verstärker mit sanfter Begrenzung

Hierbei ist allerdings zu beachten, dass durch die Verwendung eines invertierenden Verstärkers die verstärkten Signale eine andere Phasenlage haben als das Eingangssignal. Sollen anschließend alle Signale aufsummiert werden, so muss das Eingangssignal ebenfalls invertiert werden, bevor es der Summe zugeführt wird. Alternativ kann bei einem Logarithmus zur Basis 2 das Ausgangssignal der ersten Stufe doppelt aufaddiert werden, da dies eine Verstärkung von eins aufweist und somit dem Eingangssignal entspricht.

### 3.1.4 Analoger Addierer

Um den Logarithmus des Eingangssignals zu bilden, müssen die unterschiedlich verstärkten Signale aufsummiert werden. Dabei kann nach [20] ein Addierer zum Einsatz kommen. Die Eingangssignale werden dabei über Widerstände an den negativen Eingang eines Operationsverstärkers gelegt. Der Ausgang ist ebenfalls durch einen Widerstand an den Knoten zurückgekoppelt, so dass sich die Ausgangsspannung nach 3.1 berechnen lässt.

$$U_a = -\frac{R_N}{R_1}U_1 - \frac{R_N}{R_2}U_2 - \ldots - \frac{R_N}{R_n}U_n \tag{3.1}$$

Da die Verstärker eine von $0\,\mathrm{V}$ verschobene Mittellage haben, muss dies bei dem Addierer ebenfalls beachtet werden. Dazu kann der positive Eingang des Operationsverstärkers auf die virtuelle Masse der Verstärker gelegt werden. Somit befindet sich die Ruhelage des Addierers, wie gewünscht, ebenfalls auf der virtuellen Masse. Die Gl. 3.1 kann weiterhin verwendet werden, wenn die Signale relativ zu der virtuellen Masse betrachtet werden. Außerdem ist zu beachten, dass die Summe schaltungsbedingt mit minus eins multipliziert wird. Dies stellt allerdings kein Problem dar, da es sich bei dem Eingangssignal um ein Wechselspannungssignal handelt. Ein weiterer Vorteil dieser Schaltung ist, dass durch das Widerstandsverhältnis $R_N$ zu $R_n$ bestimmt werden kann, mit welchem Faktor die Eingangsspannungen in die Addition eingehen. Hierdurch kann das Ausgangssignal für die darauffolgende Schaltung angepasst werden.

### 3.1.5 Simulation der logarithmischen Verstärker

Es wurden zwei unterschiedliche Möglichkeiten zur Realisierung des logarithmischen Verstärkers erläutert. Mittels PSpice sollen beide Varianten näher untersucht und in ihrer Funktionsweise miteinander verglichen werden.

Für die Simulation wurde das PSpice-Model des Operationsverstärkers AD8608 von Analog Devices verwendet. Dieser hat, wie in [5] angegeben, einen geringen Offset von maximal $65\,\mu\mathrm{V}$ und eine unipolare Spannungsversorgung von $2{,}7\,\mathrm{V}...5{,}5\,\mathrm{V}$. Außerdem hat er ein Verstärker-Bandbreite-Produkt von $10\,\mathrm{MHz}$, wodurch auch bei einer Verstärkung von maximal $128$ noch eine ausreichende Bandbreite von $78{,}125\,\mathrm{kHz}$ zur Verfügung steht. Eine weitere Besonderheit dieses Operationsverstärkers ist, dass er in einem 14-Pin-TSSOP Gehäuse zur Verfügung steht, welches vier Verstärker beinhaltet. Dies vereinfacht eine spätere Hardware Realisation.

Das System nach [10] besteht aus einem Vorverstärker, gefolgt von einem DVGA mit Verstärkungen von $V \in \{1{,}2{,}4{,}8{,}16{,}32{,}64{,}128\}$. Durch den gewählten Logarithmus zur Basis zwei in Kombination mit den acht Stufen, decken beide Systeme eine Verstärkung von $25$ bis $3200$ ab. Somit stehen dieselben linear verstärkten Signale für die weitere Signalverarbeitung zur Verfügung und können entsprechend miteinander verglichen werden.

Anhand der Simulation wurde die Ausgangsspannung $U_e$ als Funktion der Eingangsspannung $U_{MB}$, als auch das Frequenzverhalten, der beiden Schaltungen untersucht. Bei der Simulation des Frequenzverhalten wird der negative Eingang des Differenzverstärkers auf ein festes Potenzial von $1{,}5\,\mathrm{V}$ gelegt. Der positive Eingang wird mit einer Wechselspannung plus einem Offset von $1{,}5\,\mathrm{V}$ gespeist. Die Amplitude der Wechselspannung beträgt $100\,\mu\mathrm{V}$, damit auch bei einer maximalen Verstärkung von 3200 alle Verstärker im linearen Bereich arbeiten und das Signal durch keine der Verstärkerstufen begrenzt und somit verzerrt wird.

Wie in den Simulationsergebnissen, in Anhang A.3, zu sehen ist, nehmen die Amplitudengänge bei beiden Designs wie erwartet mit steigender Frequenz ab. Dieses Verhalten wird in beiden Fällen maßgeblich vom vorgeschalteten Tiefpassfilter bestimmt. Bei der Parallelschaltung ist außerdem zu sehen, dass mit steigender Verstärkung der Amplitudengang eher anfängt abzunehmen. Dies lässt sich dadurch begründen, dass die Bandbreite der Verstärker mit zunehmender Verstärkung abnimmt.

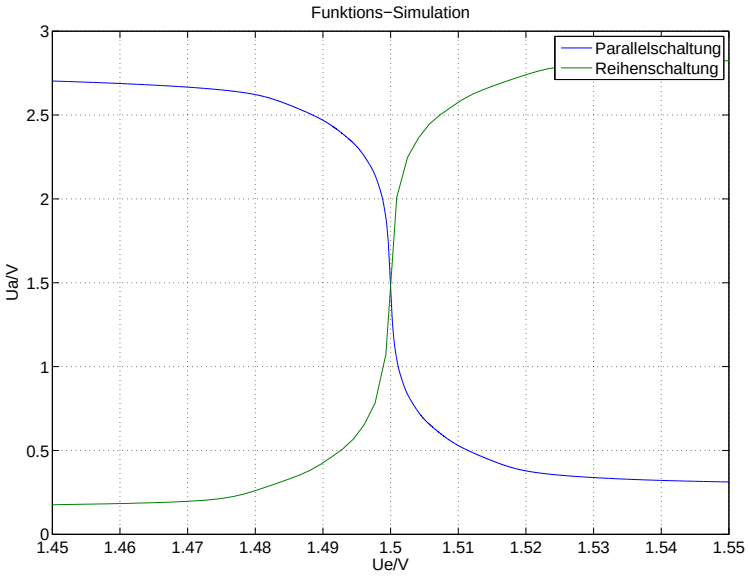



Abbildung 3.6: Funktion der Logarithmierer; Parallelschaltung ist bedingt durch den Aufbau invertiert

Der Phasengang, dargestellt in Anhang A.3 und Abbildung 3.7, verhält sich äquivalent zum Amplitudengang. Mit steigender Frequenz wird die Phasenverschiebung zwischen Eingangssignal und Ausgangssignal größer. Da bei der Parallelschaltung invertierende Verstärker eingesetzt werden, ist das Signal ebenfalls entsprechend invertiert. Dies macht sich im Phasengang dadurch bemerkbar, dass die Phase bei niedrigen Frequenzen $180\,^\circ$ beträgt, anstatt wie erwartet $0\,^\circ$. Die Phasenverschiebung der Parallelschaltung nimmt mit steigender Verstärkung früher zu. Dies ist bedingt durch die geringere Bandbreite der Verstärker bei hoher Verstärkung. Wie ursprünglich erwartet, tritt bei der Reihenschaltung keine signifikante Phasenverschiebung durch das hintereinanderschalten der einzelnen Stufen auf. Hierbei ist allerdings auch zu beachten, dass in der Simulation keine realen Dioden verwendet wurden. In der Realität wirkt sich die Kapazität der Diode, welche in Kombination mit dem Vorwiderstand, Tiefpassverhalten aufweist ggf. stärker auf die Phase aus.

In Abbildung 3.6 ist die Ausgangsspannung der simulierten Schaltungen als Funktion der Eingangsspannung dargestellt. Dabei wurde das Eingangssignal zwischen $1{,}45\,\mathrm{V}$ und

$1{,}55\,\mathrm{V}$ variiert, was einem Spitze-Spitze-Wert von $100\,\mathrm{mV}$ entspricht. Auffällig beim Vergleich der beiden Funktionen ist, dass sie invertiert zueinander sind, was durch die invertierenden Verstärker der Parallelschaltung zu erklären ist. Bei der Reihenschaltung wird das Signal nicht zusätzlich invertiert, da nicht-invertierende Verstärker zum Einsatz kommen.

Zudem fällt auf, dass das Ausgangssignal der Reihenschaltung einen größeren Spannungsbereich abdeckt als dass der Parallelschaltung. Da bei beiden Schaltungen die Begrenzung unterschiedlich realisiert wurde und somit auch nicht exakt gleich arbeitet, kommt es zu unterschiedlichen Spannungen $U_{Limit}$ bei denen die Verstärker in die Begrenzung gehen. In der Summe machen sich diese als Streckung bzw. Stauchung der Funktion bemerkbar. Durch einen nachgeschalteten Verstärker können die beiden Funktionen allerdings aneinander angepasst werden.

### 3.1.6 Phasengangkorrektur

Die Simulation zeigt, wie in Abbildung 3.7 dargestellt, dass die verstärkten Signale im Falle der Parallelschaltung unterschiedlich stark in ihrer Phase beeinflusst werden.

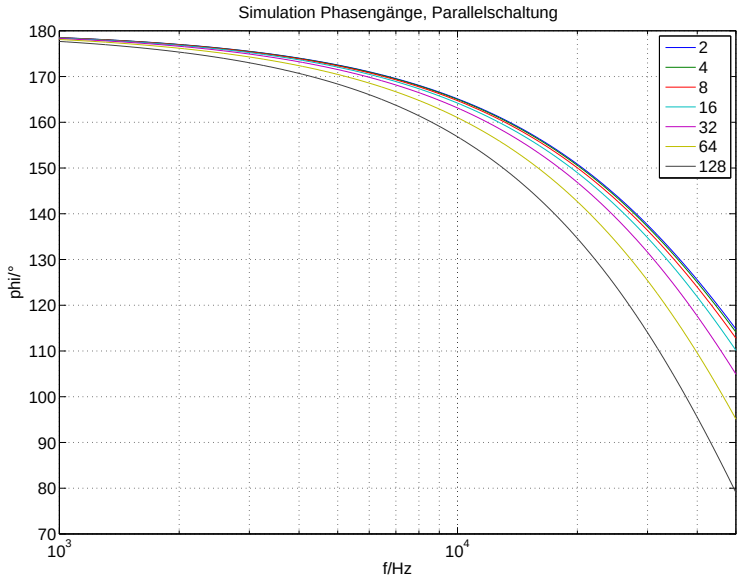



Abbildung 3.7: Simulierter Phasengang bei parallel aufgebauten Verstärkerstufen

Werden mehrere phasenverschobene sinusförmige Schwingungen gleicher Frequenz addiert, ist das Ergebnis nach [7] ebenfalls eine sinusförmige Schwingung. Gl. 3.2 verdeutlicht

diesen Zusammenhang.

$$A_1 \sin(\omega t + \varphi_1) + A_2 \sin(\omega t + \varphi_2) = A \sin(\omega t + \varphi) \tag{3.2}$$

Dieser Zusammenhang kann allerdings nicht angenommen werden, wenn die Schwingungen begrenzt werden, wie es bei der linearen Approximation der Fall ist, da durch die Begrenzung zusätzliche höherfrequente Schwingungen hinzukommen. In Abbildung 3.8 wurden exemplarisch fünf Signale mit gleicher Frequenz und unterschiedlicher Amplitude begrenzt und aufsummiert. Dabei haben die Signale im ersten Fall die gleiche Phasenlage, im Zweiten sind sie um einige Grad zueinander verschoben, d.h. Signale mit einer größeren Amplitude haben einen größeren Phasenwinkel. Dabei wird ersichtlich, dass sich die Formen der beiden Signale deutlich unterscheiden. So nimmt die Summe der phasenverschobenen Signale annähernd die Form einer Haifischflosse an.

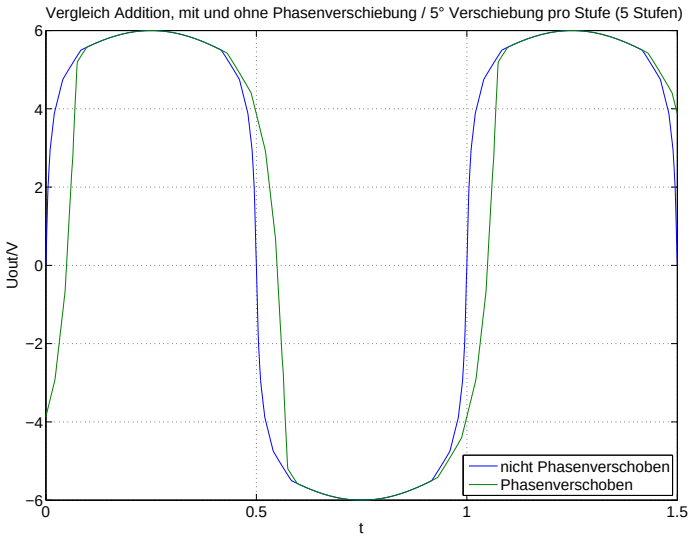



Abbildung 3.8: Vergleich Addition mit und ohne verschobener Phase

Um dieses Phänomen zu verhindern, müssen alle Signale, die aufsummiert werden, dieselbe Phasenlage haben. Zur Korrektur des Phasengangs aus Abbildung 3.7 kann parallel zu den Dioden, welche die sanfte Begrenzung ermöglichen, eine Kapazität geschaltet werden. Diese sorgt dafür, dass die Verstärkerstufen ein zusätzliches Tiefpassverhalten bekommen. Dadurch können die Verstärker so entworfen werden, dass alle den gleichen Phasengang haben. Mittels PSpice-Simulation wurden die optimalen Werte für die Kondensatoren bestimmt, welche zwischen $10\,\mathrm{pF}$ und $24\,\mathrm{pF}$ liegen. Der korrigierte Phasengang der Simulation ist in Abbildung 3.9 abgebildet. Bei der Reihenschaltung kann dies so nicht realisiert werden, da sich eine Veränderung der Phase durch die gesamte Kette fortsetzt.

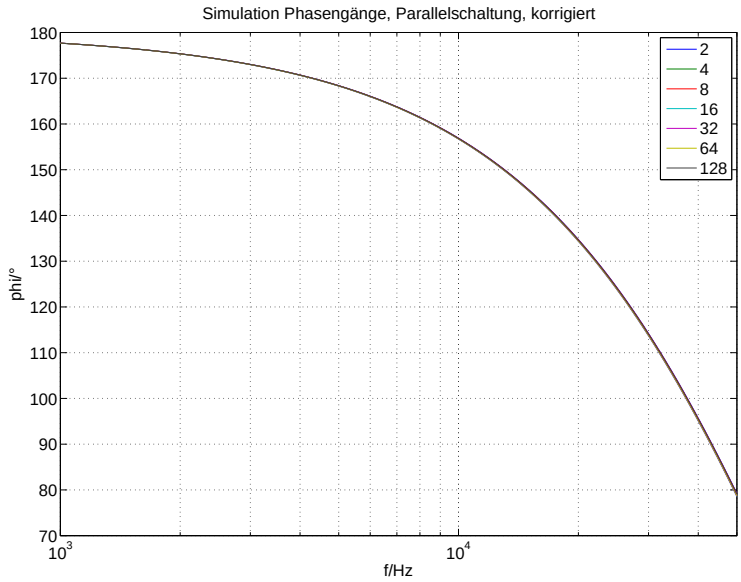



Abbildung 3.9: Simulierter korrigierter Phasengang bei parallel aufgebauten Verstärkerstufen

## 3.2 Zusammenfassung Reihen- und Parallelschaltung

Sowohl die Parallelschaltung als auch die Reihenschaltung eignen sich für die Approximation der logarithmischen Funktion. Dabei haben allerdings beide Varianten Vor- und Nachteile auf die im folgenden näher eingegangen wird.

### 3.2.1 Theoretische Gegenüberstellung

Theoretisch betrachtet hat die Parallelschaltung den Vorteil, dass die Verstärkung der einzelnen Stufen frei gewählt werden kann und somit auch ein Logarithmus zu einer beliebigen Basis gebildet werden kann. Zudem ist durch die Schaltungsanordnung der Verstärker keine sich fortsetzende Laufzeitverzögerung zu erwarten. Sie hat allerdings den Nachteil, dass die einzelnen Stufen jeweils eine unterschiedliche Verstärkung aufweisen müssen, wodurch diese entsprechend unterschiedlich ausgelegt werden müssen. Außerdem ist durch die höhere Verstärkung, welche die einzelnen Stufen erbringen müssen, die benötigte Bandbreite größer.

Die Reihenschaltung zeichnet sich hingegen dadurch aus, dass die Verstärkung der einzelnen Stufen, mit einer Ausnahme, gleich ist. Dadurch können die gleichen Verstärker verwendet werden, was den Aufbau vereinfacht. Durch diese Anordnung kann es aber zu

| | Parallelschaltung | Reihenschaltung |
|---|---|---|
| Vorteile | • geringe Laufzeitverzögerung<br>• Phasengang abgleichbar<br>• wählbare Verstärkung<br>• invertierende und nicht-invertierende Verstärker einsetzbar | • konstante Verstärkung jeder Stufe<br>• gleicher Aufbau<br>• niedriges GBWP |
| Nachteile | • hohe und unterschiedliche Verstärkung<br>• ungleicher Aufbau<br>• OP mit großer Bandbreite benötigt | • kompliziert zu beeinflussender Phasengang<br>• feste Verstärkung<br>• Laufzeitverzögerung |

Tabelle 3.2: Gegenüberstellung Parallel- und Reihenschaltung

Laufzeitverzögerungen durch den Verstärker kommen, was zu einer Verzerrung des logarithmierten Signals führt, wie in Kapitel 3.1.6 beschrieben. Ein weiterer Nachteil der Reihenschaltung ist, dass die Verstärkung der einzelnen Stufen nicht frei gewählt werden kann, sondern abhängig von den zur Begrenzung verwendeten Dioden ist. Dadurch kann kein beliebiger Logarithmus gebildet werden. In Tabelle 3.2 sind die beschriebenen Vor- und Nachteile aufgelistet.

### 3.2.2 Praktische Gegenüberstellung

Versuchsaufbauten haben gezeigt, dass sich beide Varianten unterschiedlich gut für den praktischen Einsatz eignen. Wird das System beispielsweise mit einem Eingangssignal mit einer Frequenz von $f = 0{,}5\,\mathrm{kHz}$ und einer Amplitude von $U_{MB} = 5\,\mathrm{mV}$ gespeist, neigt die Reihenschaltung der Verstärker, wie in Abbildung 3.10 dargestellt, zu starkem Rauschen in den Amplituden des Signals. Das Signal wurde dabei nach der ersten Verstärkerstufe gemessen und hat somit eine Verstärkung von $V = 50$ erfahren. Da das Signal durch die Reihenschaltung weiter verstärkt wird, wird auch das Rauschen entsprechend weiter verstärkt. Dadurch eignet es sich nicht für die weitere Verarbeitung. Da die Ursache für dieses Verhalten nicht ermittelt werden konnte und die Parallelschaltung kein solches Verhalten aufweist, wird im Weiteren nur noch die Parallelschaltung betrachtet.

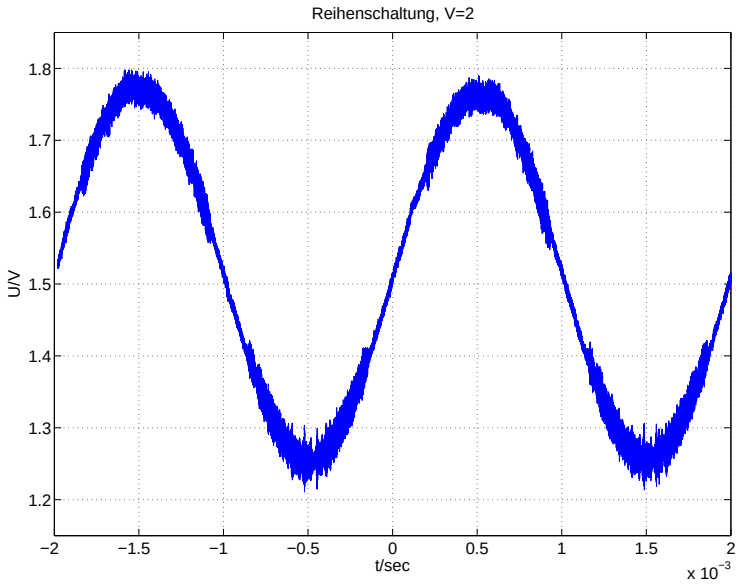



Abbildung 3.10: Reihenschaltung nach erster Verstärkerstufe, $V = 50$

## 3.3 Lineare Verstärkung

Der Logarithmierer besteht aus mehreren linearen Verstärkerstufen. Werden die Signale der einzelnen Stufen zusätzlich abgegriffen, so stehen neben dem logarithmierten Signal auch linear verstärkte Signale, wie in Abbildung 3.11 bzw. 3.12 dargestellt, zur Verfügung. Diese Signale sind allerdings nur solange linear, bis die einzelnen Stufen in die Begrenzung gehen. Da der nachfolgende Signal- bzw. Regelcontroller nur eine begrenzte Anzahl an analogen Eingänge besitzt, werden die linearen Signale über einen Analogmultiplexer bereitgestellt. So können durch Umschalten des Multiplexers die einzelnen Signale für die Weiterverarbeitung schnell ausgewählt werden.

Als Multiplexer wurde der ADG708 [3] von Analog Devices ausgewählt. Dieser bietet die Möglichkeit, aus acht Kanälen einen auszuwählen. Die Auswahl des Kanals erfolgt über ein 3-Bit breites paralleles Interface, wodurch die Umschaltung durch den Regelcontroller sehr schnell erfolgen kann. Typischerweise benötigt die Umschaltung zwischen zwei Kanälen $18\,\mathrm{ns}$ [3], womit diese deutlich schneller ist als das Sensorsignal, welches im $\mathrm{ms}$-Bereich liegt. Zudem kann er mit einer unipolaren Versorgungsspannung zwischen $1{,}8\,\mathrm{V}$ und $5{,}5\,\mathrm{V}$ versorgt werden.

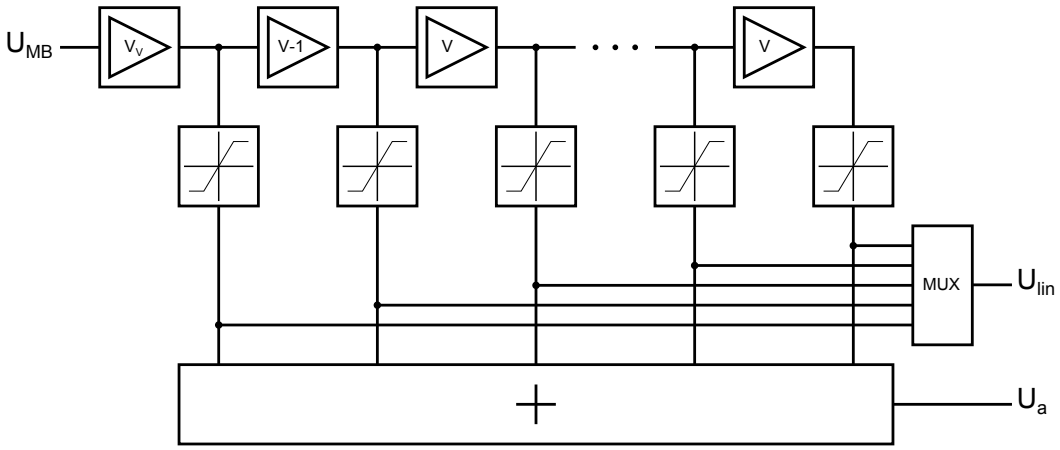



Abbildung 3.11: Signalpfad der linearen Signale bei der Reihenschaltung

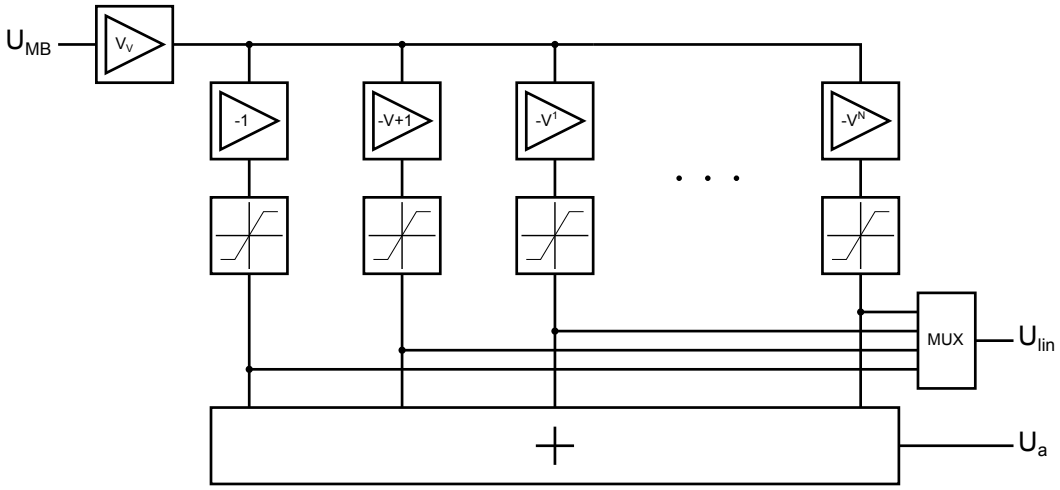



Abbildung 3.12: Signalpfad der linearen Signale bei der Parallelschaltung

## 3.4 Energieversorgung der Messbrücke

Bisher wird die Messbrücke extern mit Spannung versorgt. Eine Versorgung über den Demonstrator ist somit nicht vorgesehen. Um dem Messaufbau zu vereinfachen und das System zu vereinheitlichen, soll die Brücke über den Demonstrator mitversorgt werden.

Dazu soll zum einen die Möglichkeit geschaffen werden die Brücke, wie bisher, mit einer konstanten Spannung zu versorgen. Hierzu kann die positive $3\,\mathrm{V}$ Spannungsversorgung des Regelcontrollers genutzt werden. Um nicht nur eine feste Spannung einstellen zu können, kann das Spannungslevel zudem durch einen Digital-Analog-Umsetzer des Regelcontrollers gestellt werden. Das Signal des DAC wird dabei durch einen nicht-invertierenden Verstärker aufbereitet, um zum einen den nötigen Strom liefern zu können und zum anderen um die maximalen $2{,}5\,\mathrm{V}$ des DACs, bedingt durch die Referenzspannung, auf mindestens $3\,\mathrm{V}$ anzuheben.

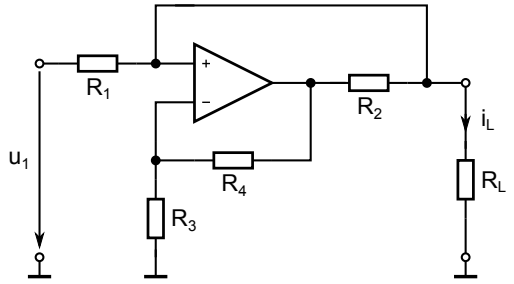



Abbildung 3.13: Spannung-Strom-Wandler, Quelle: [13]

Wie in [17] beschrieben kann die Messbrücke auch durch eine Stromquelle versorgt werden. Die Versorgung mittels Stromquelle hat den Vorteil, dass die Temperaturabhängigkeit der Brückenempfindlichkeit um den Faktor zwei gegenüber der Spannungsversorgung abnimmt. Hierbei ist allerdings auch zu beachten, dass die Versorgungsspannung entsprechend größer gewählt werden muss, da über die Stromquelle selbst eine Spannung abfällt.

Als Stromquelle kann dem nicht-invertierenden Verstärker der Spannungsversorgung ein Spannungs-Strom-Wandler nachgeschaltet werden. Dadurch ist zusätzlich die Möglichkeit gegeben, den Strom durch den Regelcontroller zu steuern. Als Spannungs-Strom-Wandler kommt eine Schaltung nach [13] zum Einsatz. Unter der Voraussetzung, dass das Widerstandsverhältnis $R_2$ zu $R_1$ gleich dem Verhältnis von $R_4$ zu $R_3$ gewählt wird, kann der Ausgangsstrom wie folgt berechnet werden.

$$i_L = \frac{u_1}{R_1} \tag{3.3}$$

Die in Abbildung 3.13 dargestellte Schaltung hat zudem den Vorteil, dass sie massebezogen ist. Dadurch wird der Wechsel zwischen den verschiedenen Versorgungsmöglichkeiten vereinfacht, da nur Änderungen im positiven Versorgungspfad durchgeführt werden müssen. Außerdem sind die Halbbrückensignale der Messbrücke somit immer massebezogen.

## 3.5 Vorverstärker

Bevor die Sensorsignale weiter verarbeitet werden können, müssen sie vorverstärkt werden. Dazu wird, wie bereits in [10], der Instrumentenverstärker AD8226 [4] verwendet. Instrumentenverstärker eigenen sich besonders um die Differenz zweier Signale, wie im vorliegenden Fall die Halbbrückensignale des AMR-Sensors, zu verstärken. Des Weiteren haben sie geringe Eingangs-Offsets und eine gute Gleichtaktunterdrückung, weshalb sie sich für den gegebenen Anwendungsfall gut eignen.

Der Eingangstiefpassfilter, welcher in [10] vorgesehen wurde, wird ebenfalls übernommen. Dieser unterdrückt hochfrequente Störanteile, die auf den Signalleitungen zwischen Sensor und Verstärkerplatine eingestreut werden können. Die Grenzfrequenz des Filters wurde zu $f_g = 56{,}6\,\mathrm{kHz}$ gewählt, sodass die ersten fünf Harmonischen nicht beeinträchtigt werden.

## 3.6 Verstärkung der Halbbrückensignale

Eine Signalaufbereitung in Form von Offsetkompensation und Verstärkungsreglung fand, wie in [10] und [19] beschrieben, bisher nur für die Brückendifferenzspannung $U_{diff}$ statt. Da die beiden Halbbrückensignale für Funktionen wie z.B. die Drehrichtungserkennung nicht minder von Bedeutung sind, sollen diese auch entsprechend aufbereitet werden. Dazu wird die kombinierte logarithmisch und lineare Verstärkerbank so gestaltet, dass sie sowohl für die beiden Halbbrückensignale als auch für die Differenz der beiden Signale eingesetzt werden kann. Infolgedessen werden drei äquivalente Verstärker, wie in Abbildung 3.14 dargestellt, für die Verstärkung der Halbbrückensignale als auch für die Verstärkung des Differenzsignals der Messbrücke verwendet.

Für die Offsetkompensation wird das Tastverhältnis des logarithmierten Signals mit Hilfe eines Smart-Comparators, beschrieben in [19], ermittelt. Da der MSP430F1611, welcher als Regelcontroller eingesetzt wird, allerdings nur über einen Komparator verfügt, ist die Offsetkompensation aller drei Signale nicht möglich. Der Gleichanteil der beiden Halbbrückensignale kann somit nicht geregelt werden und muss manuell abgeglichen werden. Für die Verstärkung der Halbbrückensignale kann dieselbe Verstärkung wie für das Differenzsignal ausgewählt werden, da diese eine geringere maximale Aussteuerung als die Differenzspannung $U_{diff}$ aufweisen.

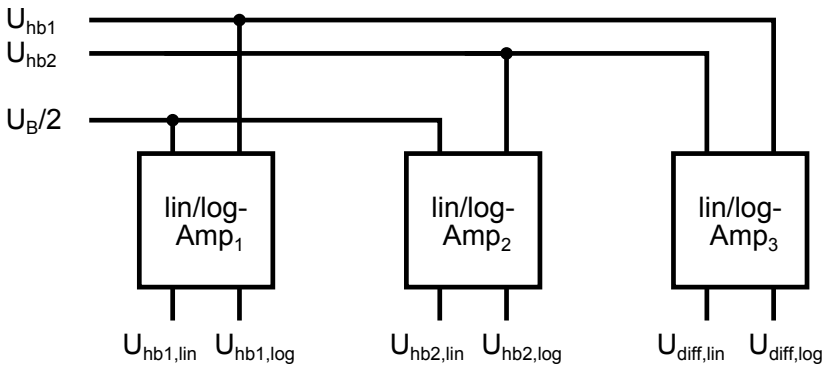



Abbildung 3.14: Parallel angeordnete Verstärker für Halbbrückensignale und Differenzsignal

## 3.7 Offset-Stellglied

Wie in Kapitel 2.3 bereits erläutert, ist eine Offsetregelung des Systems unerlässlich. Dazu muss an mindestens einer Stelle des Systems der Offset gestellt werden. Bisher wurde dazu der negative Eingang des DVGA genutzt. Da dieser Verstärker durch einen logarithmischen Verstärker ersetzt werden soll, muss die Stellgröße an einer anderen Stelle dem System zugeführt werden. Der Vorverstärker AD8226 besitzt einen extra Eingang, der die Nulllage des Ausgangssignals festlegt. Dieser eignet sich somit optimal, um den Offset zu stellen. Des Weiteren ist es möglich bei der Verstärkung der Halbbrückensignale den Offset am Eingang des Differenzverstärkers zu stellen. Dazu wird auf einen Eingang das Halbbrückensignal gegeben und auf den zweiten Eingang die Stellgröße geführt. Somit wird zudem eine grobe und feine Einstellung des Offsets ermöglicht.

Wenn die Stellgröße allerdings so früh dem System zugeführt wird, ist eine sehr feine Auflösung erforderlich. Eine an dieser Stelle auftretende Abweichung von zum bsp. $1\,\mathrm{mV}$, führt bei einer Verstärkung von maximal $3200$ zu einem Fehler von $3{,}2\,\mathrm{V}$. Da der ADC des Signalcontrollers maximal $2{,}5\,\mathrm{V}$ verarbeiten kann, ist der Fehler unzulässig hoch. Aus diesem Grund wurde für die Einstellung des Offsets der 16-Bit Digital-Analog-Umsetzer AD5667 [2] ausgewählt.

$$U_{LSB} = \frac{U_{Ref}}{2^n} = \frac{3V}{2^{16}} = 45{,}78\mu V \tag{3.4}$$

Bei einer Referenzspannung von $3\,\mathrm{V}$ führt dies nach 3.4 zu einer Auflösung von ca. $0{,}05\,\mathrm{mV}$. Ein daraus anzunehmender Stellfehler von $25\,\mu\mathrm{V}$ führt bei der maximalen Verstärkung zu einem Offsetfehler von $80\,\mathrm{mV}$ am Ende der Verstärkung.

# 4 Systemintegration

Soll der neu entworfene Verstärker mit dem bestehenden System aus Controllerplatine, Regelplatine zusammenarbeiten, müssen sowohl die Hardware als auch die Software aufeinander abgestimmt werden. An dieser Stelle wird dabei auf die zu beachtenden Punkte näher eingegangen.

## 4.1 Hardwareintegration

Der bisherige Verstärker besteht aus einer einzelnen Platine, welche die Verstärkerstufen für alle drei relevanten Signale: $U_{diff}$, $U_{hb1}$ und $U_{hb2}$ enthält. Dabei erfahren die beiden Halbbrückensignale allerdings nur eine fest eingestellte Verstärkung von $25$ und nur die Differenzspannung wird durch den DVGA weiter verstärkt. Bei dem Neuentwurf des Verstärkers sollen auch die Halbbrückensignale weiter verstärkt werden. Um diese Möglichkeit zu schaffen, wurde die logarithmisch und lineare Verstärkerbank auf ein steckbares Modul gebaut, welche sowohl für die Verstärkung der Halbbrückensignale als auch für die der Differenzspannung eingesetzt werden kann.

Damit derselbe Verstärker für alle drei Signale eingesetzt werden kann, wurde eine Hauptplatine entwickelt, welche drei Verstärker-Module aufnehmen kann. Auf der Hauptplatine, befindet sich die Energieversorgung der Messbrücke. Zudem wird die Hauptplatine mit dem Signal- bzw. Regelcontroller verbunden und stellt die Anschlüsse für die AMR-Messbrücke zur Verfügung.

## 4.2 Signalführung

Damit die einzelnen Verstärker ihre Funktionen erfüllen können, müssen ihnen sowohl digitale als auch analoge Signale zur Verfügung stehen. Insgesamt werden die im folgenden aufgeführten Signale benötigt:

- 2 x analoge Eingangssignale $U_{hb1}$ und $U_{hb1}$
- 2 x analoge Ausgangssignale: linear und logarithmisch verstärktes Signal

- 3 x digitale Kanäle zur Ansteuerung des Multiplexers
- 2 x digitale Kanäle ($I^2C$) zur Ansteuerung des DACs
- 1 x digitaler Kanal zur Ansteuerung des Wechselschalters
- Masse und positive Spannungsversorgung

Die Hauptplatine muss die Signale der einzelnen Verstärker-Module aufnehmen und diese an die Platine des Regelcontrollers weiterleiten. Außerdem muss sie die Anschlüsse für die Halbbrückensignale bereitstellen und die Messbrücke mit Energie versorgen. Für die Energieversorgung wird zusätzlich ein analoger Ausgang des Regelcontrollers benötigt und einen zusätzlicher Anschluss, für die Messbrücke. In der nachfolgenden Abbildung 4.1 ist eine Übersicht der Signalführung dargestellt.

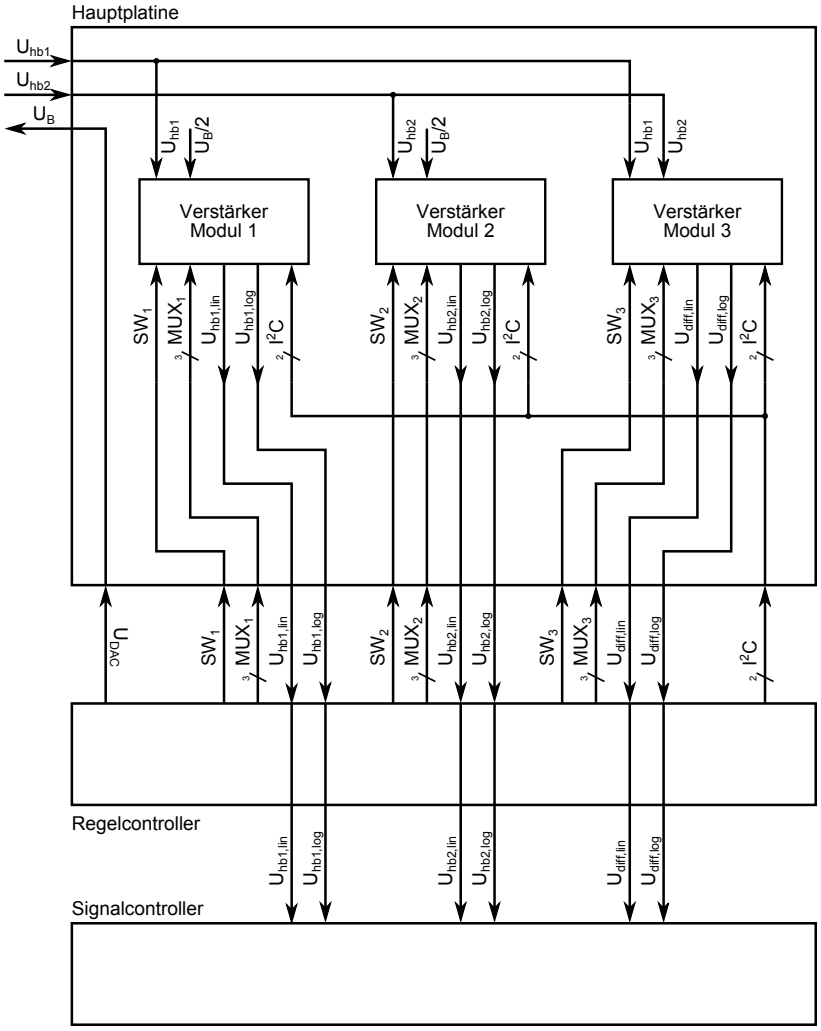



Abbildung 4.1: Signalführung auf der Hauptplatine

## 4.3 Softwareintegration

Für das neue Verstärkerkonzept muss die Software des bestehenden Systems und insbesondere die des Regelcontrollers an die neue Hardware angepasst werden. Im Folgenden werden die vorgenommenen Änderungen beschrieben.

### 4.3.1 Regelcontroller Software

Die Software für den Regelcontroller nach [19] besteht im Wesentlichen, wie in Abbildung 4.2 dargestellt, aus einem Zustandsautomaten mit den drei Zuständen *Initial_0Hz*, *Initial_1Hz* und *Active*. Im Zustand *Initial_0Hz* befindet sich das System nach dem Start und verharrt hier solange, bis eine Drehung des Rades erkannt wird. Wird eine Drehung erkannt, wird in den Zustand *Initial_1Hz* gewechselt. Hier werden abwechselnd Verstärkung und Offset des Systems geregelt, bis das Signal optimal verstärkt und gleichspannungsfrei ist. Anschließend wechselt das System in den Zustand *Active*, wo die Diagnose des Signals freigegeben und eine langsame Regelung zum Einsatz kommt, welche Offset- und Amplituden- Drifts ausgleicht.

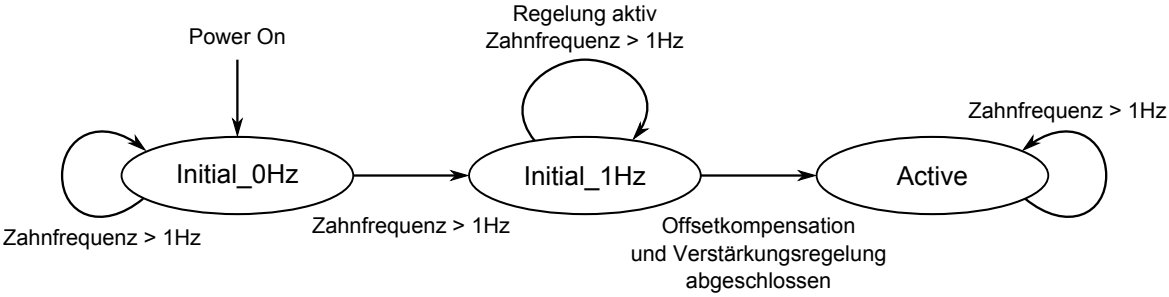



Abbildung 4.2: übergeordneter Zustandsautomat, nach [19]

Der Zustande *Initial_1Hz* ist in weitere sechs Zustände unterteilt (vgl. linker Teil Abbildung 4.3). Im Zustand *Search_S1* wird auf einen positiven Nulldurchgang des Differenzsignals $U_{diff}$ gewartet, der Zählerstand von Timer A für die Berechnung des Duty-Cycle gespeichert und der Timer zurück gesetzt. Der Zustand *Search_S2* ist für die Behandlung eines negativen Nulldurchgangs zuständig und sichert ebenfalls den Wert von Timer A. Wurden die beiden Zustände durchlaufen, wird in den Zustand *Search_S1_Calc* gewechselt. Hier wird zudem das Tastverhältnis bestimmt und die Funktion `calc_offset()` aufgerufen, in der ein Offsetkorrekturfaktor bestimmt wird. Danach wechselt der Automat in den Zustand *Search_S2_Pre*, von wo aus er am Ende einer Periode wieder in den Zustand *Search_S1* übergeht. Wurde die Offsetkompensation abgeschlossen, geht der Automat in den Zustand *Search_S2_Check_Gain*, in dem die obere Halbwelle des Signals abgetastet

wird und anschließend in *Search_S1_Check_Gain*, in welcher die untere Halbwelle behandelt wird. Im letzteren Zustand wird die Funktion `check_gain()` aufgerufen, in der, wie in Kapitel 2.4.1 beschrieben, ggf. auftretende Überschreitungen der Schwellwerte ausgewertet werden und die Verstärkung entsprechend gestellt wird. Anschließend geht der Automat wieder in den Zustand *Search_S2_Pre*.

Ist die Offsetkompensation abgeschlossen und die Verstärkung richtig eingestellt, geht der übergeordnete Automat in den Zustand *Active* über, welcher die gleichen Subzustände enthält (vgl. Abbildung 4.3, rechter Teil). Der wesentliche Unterschied zum Zustand *Initial_1Hz* ist, dass hier das Tastverhältnis und die Verstärkung überwacht werden und nur, wenn sich einer der beiden Werte unzulässig ändert, in das System eingegriffen wird.

### 4.3.2 Initialisierung

Damit das bestehende System mit dem neuen Verstärkerkonzept zusammenarbeitet, müssen an der Regelcontroller-Software einige Änderungen vorgenommen werden. Dazu zählt als Erstes die Initialisierungsphase des Controllers. Da der Controller mehrere Multiplexer und Schalter ansteuern muss, müssen die verwendeten Port-Pins entsprechend konfiguriert werden.

Des Weiteren wurde der ADC neu konfiguriert. Da das logarithmierte Signal auf einem anderen Eingang an dem Controller anliegt, muss dieser entsprechend umkonfiguriert werden. Bisher wurde der ADC zudem nur bei Bedarf eingeschaltet und nach Abtasten einer Periode wieder abgeschaltet. Da für mehrere Aufgaben im System Angaben über die Amplitude des Signals unmittelbar zur Verfügung stehen müssen, wurde der ADC so konfiguriert, dass ständig eine Abtastung stattfindet. Die Abtastfrequenz wurde dabei nicht verändert.

Außerdem muss für die Ansteuerung der einzelnen Digital-Analog-Umsetzer die $I^2C-$Schnittstelle des Controllers initialisiert und konfiguriert werden. Ist die Schnittstelle funktionsfähig müssen über diese die externen DACs initialisiert werden.

### 4.3.3 Abtastung

Da die Abtastung des Regelcontrollers und die weitere Verarbeitung der Samples an den logarithmischen Verstärker angepasst wurden, muss auch die entsprechende Interrupt Service Routine (ISR) angepasst werden. Die ISR wird nach jeder vollständig abgeschlossenen Analog-Digital-Umsetzung aufgerufen.

In der ISR, welche in Anhang D.1 aufgeführt ist, werden die digitalisierten Werte als Erstes delogarithmiert. Dazu wurde das System, wie in Abschnitt 2.2.2 beschrieben, vermessen und die Umkehrfunktion in Form einer Lookup-Tabelle gespeichert. Da dabei allerdings für jeden möglichen Abtastwert (0 - 4095) ein 16-Bit Wert in der Tabelle abgelegt wird,

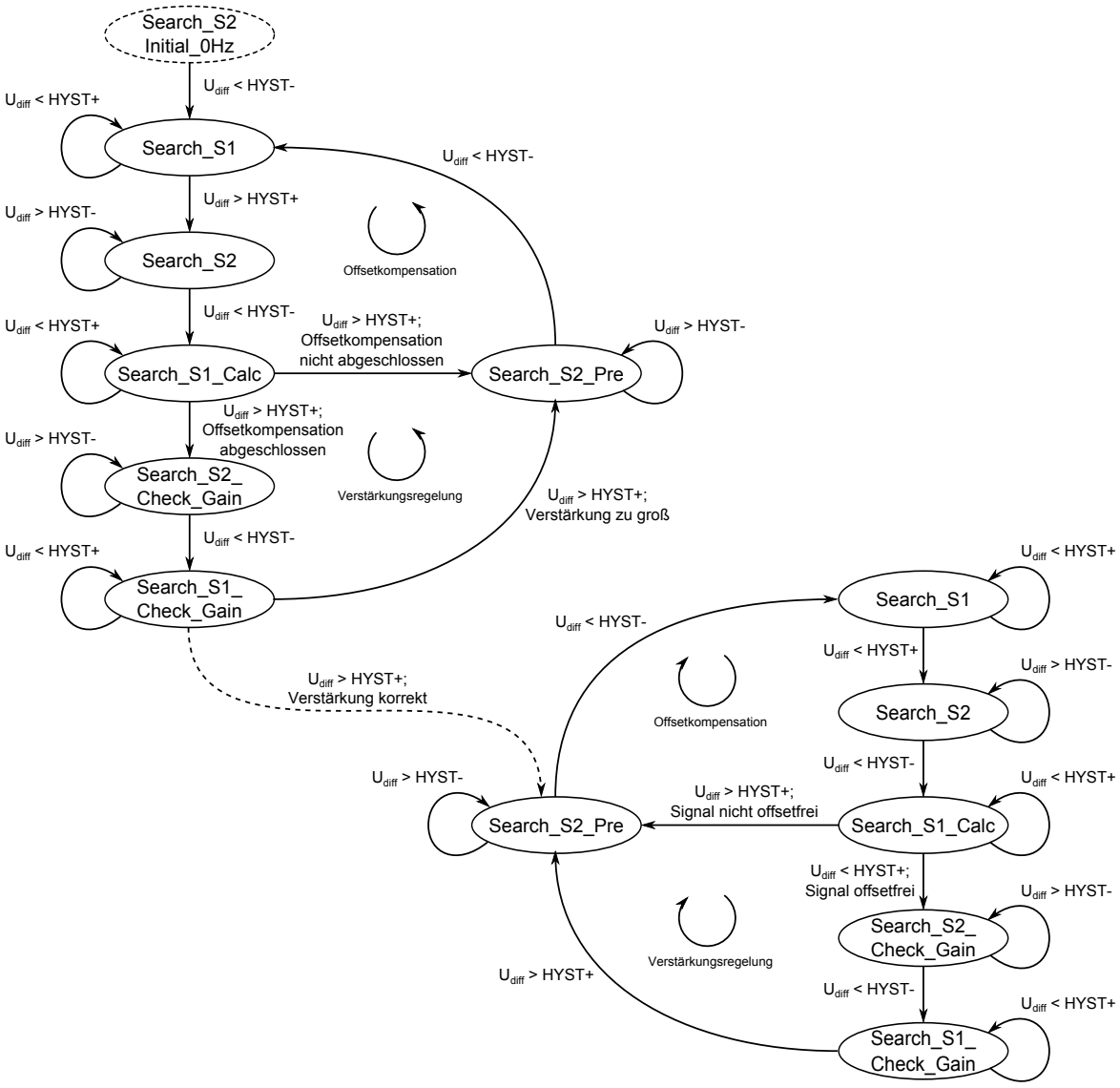



Abbildung 4.3: Zustandsautomat nach [19]

wird diese sehr groß. Im Gegenzug kann mit dem ermittelten Wert direkt auf ein Array zugegriffen werden und es ist keine weitere Berechnung notwendig.

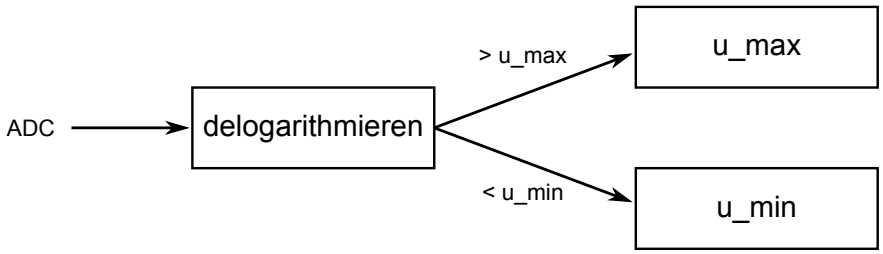



Abbildung 4.4: Ausschnitt der ADC Interrupt Service Routine

Anschließend wird der ermittelte Wert mit den Variablen `g_adc.u_min_offset` bzw. `g_adc.u_max_offset` verglichen und überprüft, ob er kleiner bzw. größer ist. Ist dies der Fall wird der Wert als neues Minimum bzw. Maximum in der entsprechenden Variablen gespeichert. Somit stehen nach einer Periode das Maximum und Minimum des Eingangssignals in den beiden Variablen zur weiteren Verarbeitung zur Verfügung. Wurden die Werte verarbeitet, müssen die Variablen zurückgesetzt werden, da sonst ein Kleinerwerden der Aussteuerung nicht detektiert werden würde. Da das Maximum und das Minimum sowohl für die Offsetkompensation, als auch für die Auswahl der Verstärkung benötigt werden, allerdings zu unterschiedlichen Zeitpunkten, werden diese zusätzlich noch in den Variablen `g_adc.u_min_gain` und `g_adc.u_max_gain` gespeichert.

### 4.3.4 Zustandsautomat

An dem Zustandsautomaten des Regelcontrollers nach Abbildung 4.3, müssen ebenfalls einige Änderungen vorgenommen werden, damit dieser weiterhin verwendet werden kann. Der Zustandsautomat führt, unter Berücksichtigung seines Zustandes, beim Erkennen eines Nulldurchgangs unterschiedliche Aktionen aus. Dazu gehört das Ein- bzw. Ausschalten des ADCs, welche aus allen Zuständen entfernt wurde, da der ADC dauerhaft samplen soll. In allen Zuständen, die einen positiven Nulldurchgang behandeln, wurde zudem implementiert, dass das letzte detektierte Minimum für die Offsetkompensation zu der Variablen `g_adc.u_min_middle` hinzu addiert wird. Außerdem wird das Minimum zurückgesetzt und ein entsprechender Zähler inkrementiert, der die Anzahl der Additionen speichert. In allen Zuständen, die einen negativen Nulldurchgang behandeln, wird zusätzlich die Addition des Maximums zur Variablen `g_sig.u_max_middle` implementiert. Hierbei wird ebenfalls das letzte Maximum entsprechend zurückgesetzt und eine Variable inkrementiert, in der die Anzahl der Additionen gespeichert wird.

### 4.3.5 Offsetkompensation

Mittels der Funktion `calc_offset()` wird der Gleichanteil des Signals bestimmt. Dafür wurde bei sinusförmigen Signalen bislang nur das Tastverhältnis und eine Angabe über die Verstärkung benötigt. Da aus dem logarithmierten Signal mit Hilfe des Tastverhältnisses nicht direkt auf den Gleichanteil geschlossen werden kann, muss diese Funktion entsprechend angepasst werden.

Durch den logarithmischen Verstärker wird das Tastverhältnis des Eingangssignals, wie in Kapitel 2.3 beschrieben, nicht verändert. Dadurch kann dies weiterhin für die Berechnung des Gleichanteils genutzt werden. Wird weiterhin davon ausgegangen, dass das Eingangssignal sinusförmig ist und dessen Amplitude bekannt ist kann der Offset mit der Formel 2.27 berechnet werden.

Da das Minimum und Maximum des Eingangssignals in den beiden Variablen `g_adc.u_min_middle` und `g_adc.u_max_middle` gespeichert ist, kann aus diesen beiden Angaben die Amplitude des Signals direkt bestimmt werden. Der Duty-Cycle des Signals ist in der Variablen `g_sig.duty` in Prozent gespeichert. Der Wert ist zudem auf $128$ skaliert: $100\,\% \mathrel{\hat{=}} 128$.

Der Vorverstärker wird mit einer Betriebsspannung von $3V$ betrieben, womit sich für den darauffolgenden logarithmischen Verstärker ein maximaler Eingangsspannungshub von $0\,\mathrm{V}$ bis $3\,\mathrm{V}$ ergibt. In der Lookup-Tabelle, mit deren Hilfe die Eingangswerte des logarithmischen Verstärkers zurückgewonnen werden, werden die Werte in $0{,}1\,\mathrm{mV}$ Schritten gespeichert. Für die Amplitude kann somit ein maximaler Wert von $30000$ ermittelt werden. Für die Berechnung des Gleichanteils muss die Amplitude mit einer Sinusfunktion multipliziert werden. Das Ergebnis der Sinusfunktion ist vom Tastverhältnis des Signals abhängig. Für eine einfachere Implementierung wurde die Sinusfunktion ebenfalls in einer Lookup-Tabelle abgelegt. Da die Funktion, wie in [10] gezeigt, punktsymmetrisch ist, ist es ausreichend nur Werte für ein Tastverhältnis zwischen $0\,\%$ und $50\,\%$ in der Tabelle abzulegen. Für Tastverhältnisse größer $50\,\%$ können diese aufgrund der Punktsymmetrie entsprechend berechnet werden.

$$D = \frac{U_{out} \cdot 2^n}{U_{Ref}} = Offset \cdot \frac{2^{16} \cdot 0{,}1\,\mathrm{mV}}{2^{15} \cdot 3V} = Offset \cdot \frac{1}{15000} \tag{4.1}$$

Mittels der Gl. 4.1, welche in [2] angegeben ist, kann berechnet werden welcher Wert im DAC gesetzt werden muss, damit eine bestimmte Spannung ausgegeben wird. Da die Sinusfunktion, die für die Berechnung des Offsets verwendet wird, auf $2^{15}$ skaliert ist und die Amplitude in $0{,}1\,\mathrm{mV}$ angeben ist, müssen diese Faktoren zuvor wieder herausgerechnet werden. Zusätzlich ist die ausgegebene Spannung von der Referenzspannung $U_{Ref}$ und der Auflösung $2^n$ des DACs abhängig. Diese vier Faktoren können zu einer Konstanten zusammengefasst werden. Ist der Offset ermittelt, wird dieser zu dem aktuellen hinzu addiert. Der dazugehörige Quellcode, befindet sich in Anhang D.1.

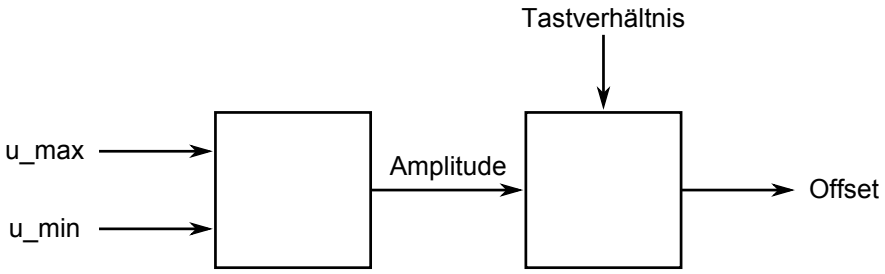



Abbildung 4.5: Schematische Darstellung `calc_offset()`

### 4.3.6 Verstärkungsauswahl

Wie bereits beschrieben, findet die Auswahl der Verstärkung in der Funktion `check_gain()` statt. Hierbei werden die Anzahl der Samples, die innerhalb einer Periode die vier Schwellwerte überschritten haben, ausgewertet und die Verstärkung entsprechend gestellt. Außerdem wird festgelegt, in welchen Zustand die beiden Zustandsautomaten wechseln.

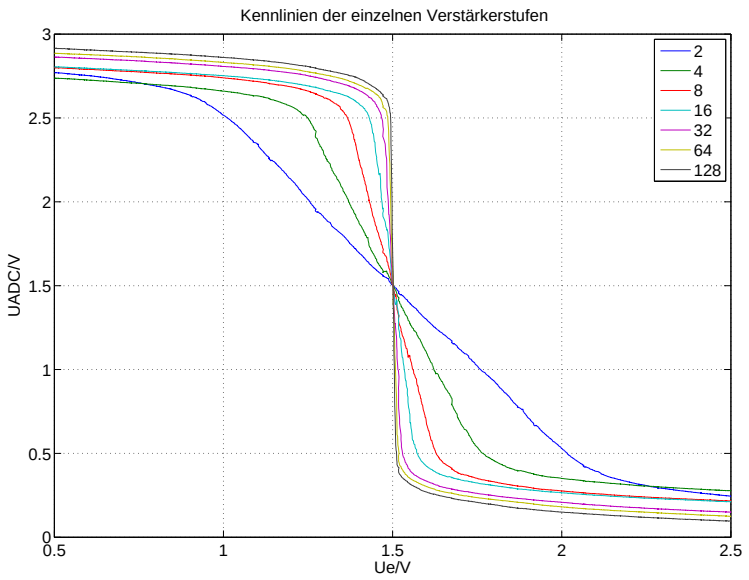



Abbildung 4.6: Kennlinien der einzelnen Verstärkerstufen

Für die Verstärkerbank bzw. logarithmische Verstärkung muss diese Funktion angepasst werden. Da der ADC so konfiguriert wurde, dass er dauerhaft sampelt, steht schon beim ersten Aufruf der Funktion das Minimum und Maximum des Eingangssignals zur Verfügung. Daraus kann unmittelbar die Amplitude des Eingangssignals bestimmt werden.

In Abbildung 4.6 sind die Kennlinien der einzelnen Verstärkerstufen aufgetragen. Daraus kann abgelesen werden, dass alle Stufen in einem Ausgangsspannungsbereich von $0{,}6\,\mathrm{V}$ bis $2{,}4\,\mathrm{V}$ linear arbeiten, also einem maximalen Spitze-Spitze-Wert von $1{,}8\,\mathrm{V}$. Da weiterhin die Verstärkungsfaktoren bekannt sind, können daraus, wie in Tabelle 4.1 aufgelistet, die maximalen Eingangsspannungswerte für die einzelnen Verstärkerstufen berechnet werden.

Mittels mehrerer if-Bedingungen wird in der Funktion `check_gain()` daraufhin die Größe des Eingangssignals abgefragt und die passende Verstärkung ausgewählt.

| Verstärkung | maximale Eingangsspannung $U_{SS}$ |
|---|---|
| 2 | 900 mV |
| 4 | 450 mV |
| 8 | 225 mV |
| 16 | 112,5 mV |
| 32 | 56,3 mV |
| 64 | 28,1 mV |
| 128 | 14,4 mV |

Tabelle 4.1: Maximale Eingangsspannung bei gegebener Verstärkung

Zudem wird der Folgezustand des übergeordneten Zustandsautomaten auf *Active* gesetzt, wodurch das System die Signaldiagnose frei gibt. Eine Regelung der Verstärkung im Zustand *Initial_1Hz* entfällt somit, der vereinfachte Zustandsautomat ist in Abbildung 4.7 dargestellt. Am Ende der Funktion werden die Variablen `g_adc.u_min_gain` und `g_adc.u_max_gain` zurückgesetzt, damit innerhalb der nächsten Periode das Minimum und Maximum neu detektiert werden können. Der zugehörige Quellcode für den Regelcontroller befindet sich in Anhang D.1.

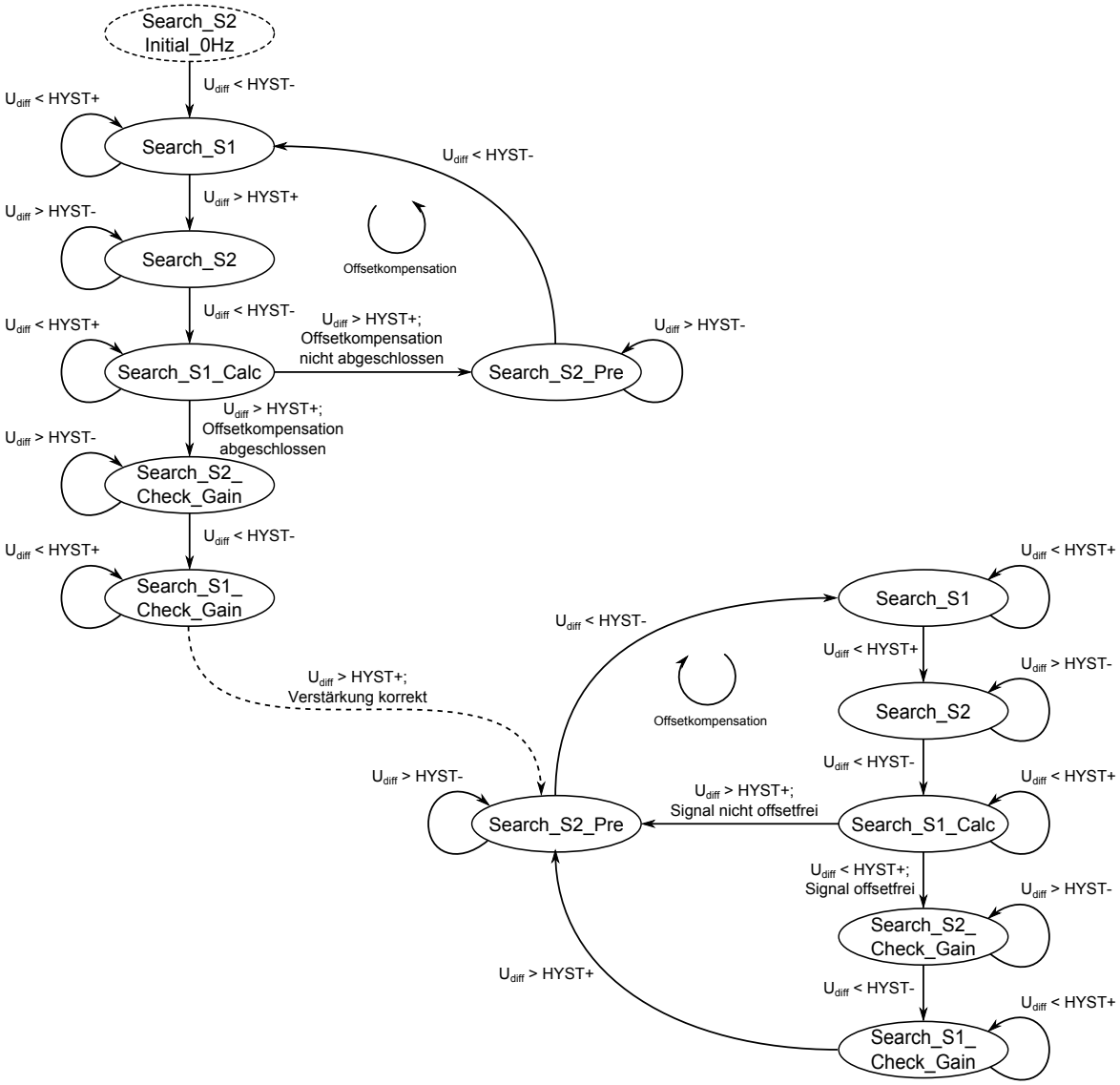



Abbildung 4.7: Zustandsautomat nach [19], hier vereinfacht nach Wegfall der initialen Verstärkungsregelung

# 5 Messungen und Funktionsnachweis

Anhand von Messergebnissen soll die ordnungsgemäße Funktion des Systems nachgewiesen werden. Dabei werden in erster Instanz die einzelnen Verstärker der in Abbildung 5.1 abgebildeten logarithmisch und linearen Verstärkerbank vermessen und mit den Simulationsergebnissen verglichen. Anschließend folgt eine Überprüfung der einzelnen Funktionen, wie Offsetkompensation und die Auswahl der Verstärkung. Abschließend werden THD-Kennfelder des bestehenden Systems dem neu Entworfenen gegenüber gestellt.

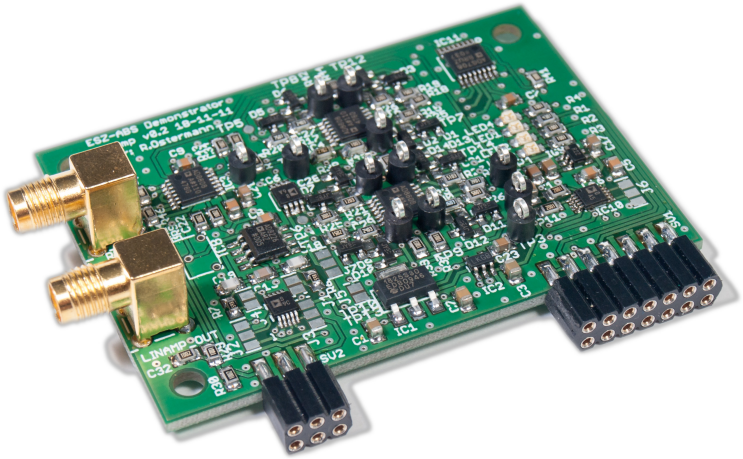


Abbildung 5.1: Kombinierte logarithmisch und lineare Verstärkerbank

## 5.1 Amplitudengang

Wie bereits in der Simulation wurde der Amplitudengang der einzelnen Verstärker gemessen. Dabei wurde eine einzelne logarithmisch und lineare Verstärkerbank so konfiguriert, dass sie das Halbbrückensignal $U_{hb1}$ verstärkt. An Stelle des Halbbrückensignals wurde eine in der frequenzveränderliche Wechselspannung mit einem Effektivwert von $U_{\mathit{eff}} = 0{,}1\,\mathrm{mV}$

und einem Gleichspannungsanteil von $1{,}5\,\mathrm{V}$ eingespeist. Die Amplitude des Wechselspannungssignals wurde dabei, wie bereits in der Simulation, so gewählt, dass keine der Verstärkerstufen in die Begrenzung geht. Als Messgerät wurde der UVP Audio Analyzer der Firma Rohde & Schwarz verwendet. Des Weiteren wurde für die Messung der Regelcontroller angehalten, sodass keine Offsetkompensation bzw. automatische Auswahl der Verstärkung statt fand. Diese wurden manuell eingestellt.

In Abbildung 5.2 sind die Messergebnisse dargestellt. Dabei wurden, um eine bessere Vergleichbarkeit zu erzielen, die unterschiedlich verstärkten Signale normiert. In der Abbildung ist, wie auch die Simulation gezeigt hat, gut zu erkennen, dass die Bandbreite mit steigender Verstärkung abnimmt. So wird das Signal mit der größten Verstärkung deutlich eher gedämpft als ein weniger stark verstärktes Signal. Dass sich die Bandbreite nicht mit Halbierung der Verstärkung verdoppelt, ist bedingt durch den Eingangsfilter, der sich bei den geringer verstärkten Signalen auswirkt.

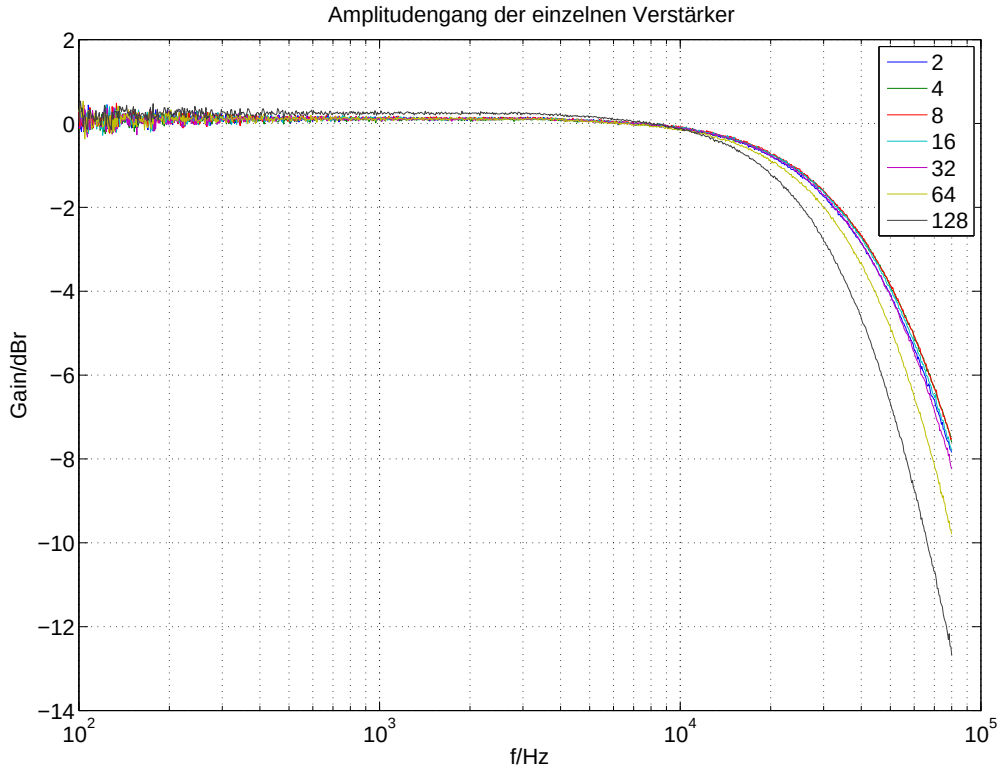



Abbildung 5.2: Gemessene Amplitudengänge

## 5.2 Phasengang

Neben dem Amplitudengang wurde auch der Phasengang der einzelnen Verstärkerstufen gemessen. Dieser entspricht im Verlauf ebenfalls der Simulation. Allerdings setzt eine spürbare Phasenverschiebung bei den gemessenen Signalen eher ein als in der Simulation. Dies kann durch parasitäre Effekte der Schaltung hervorgerufen werden, welche in der Simulation nicht berücksichtigt wurden. In der Simulation wurde gezeigt, dass eine zu den Dioden parallele geschaltete Kapazität im $\mathrm{pF}$-Bereich bereits zu einer merklichen Veränderung des Phasengangs führt. Da die gewählten Dioden [8] eine Kapazität von $2\,\mathrm{pF}$ aufweisen, kann dies den ungleichen Phasengang erklären.

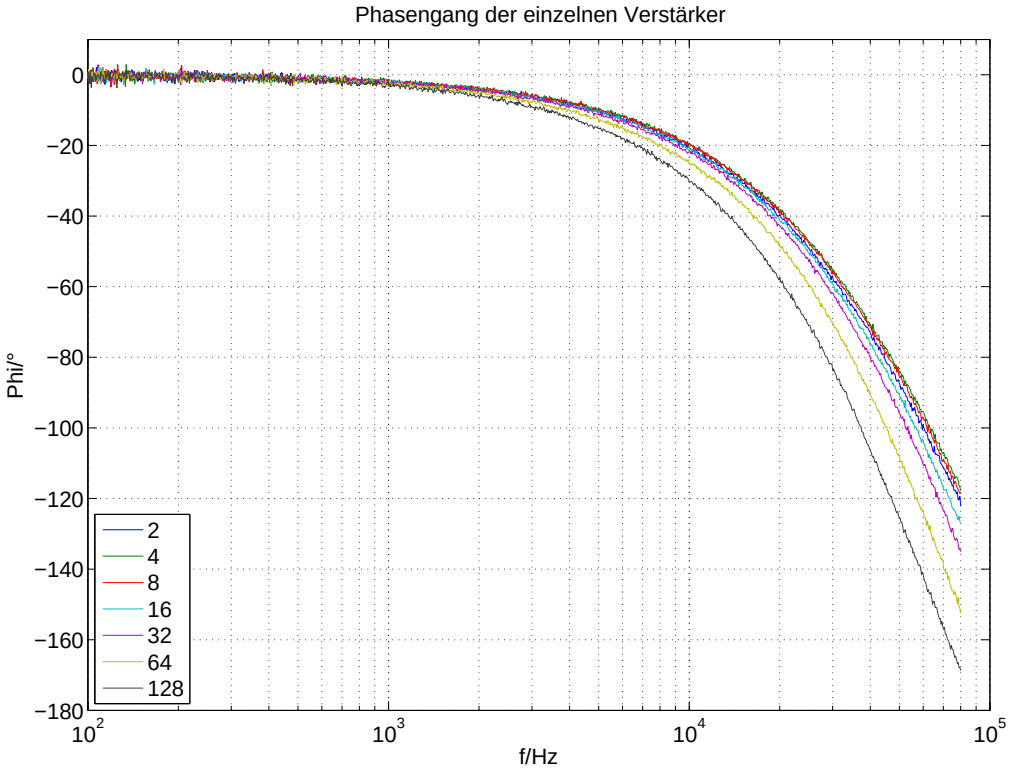



Abbildung 5.3: Gemessene Phasengänge

## 5.3 Verstärkungsauswahl

Für die nachfolgenden Messungen wurde der Versuchsaufbau nach Abbildung 5.4 verwendet. Das System wird dabei an dem Eingang $U_{hb1}$ über den Funktionsgenerator AFG3022B der Firma Tektronix gespeist. Der zweite Eingang wurde für die Messungen auf ein festes Potenzial von $U_{hb2} = 1{,}5\,\mathrm{V}$ gelegt. Zur Aufnahme der Messergebnisse, wurde das Oszilloskop MSO3034 der Firma Tektronix eingesetzt. Mit diesem wurde sowohl das Eingangssignal $U_{hb1}$ als auch das linear Verstärkte Ausgangssignal $U_{diff,lin}$ aufgezeichnet.

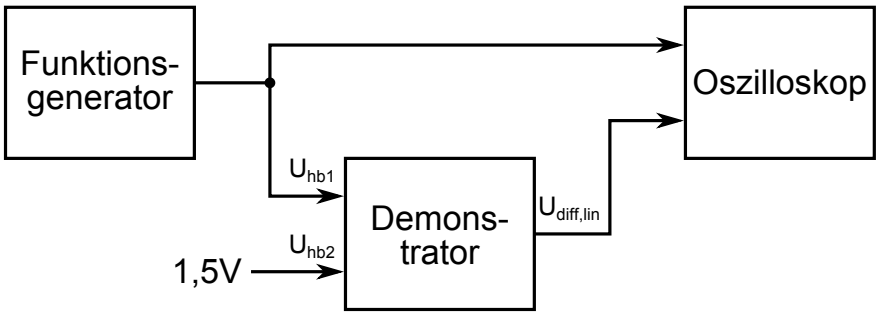



Abbildung 5.4: Versuchsaufbau

Für die weitere Signalverarbeitung ist die Auswahl der optimalen linearen Verstärkung entscheidend. Um die Funktionsfähigkeit des Systems zu verifizieren, wurde die Auswahl der optimalen Verstärkung in mehrere Fällen untersucht. Zum einen wurde der Systemstart näher betrachtet, zum anderen wurde ein, in der Signalamplitude kontinuierlich abnehmendes, Eingangssignal und eine sprunghafte Änderung der Amplitude untersucht.

In Abbildung 5.5 ist das Verhalten des Systems beim Start zu sehen. Im oberen Teil der Abbildung ist das Eingangssignal $U_{hb1,pp} = 50\,\mathrm{mV}$ und einer Frequenz von $f = 1\,\mathrm{kHz}$ abgebildet. Im unteren Graphen ist das linear verstärkte Signal beim Start des System dargestellt. Im linken Teil arbeitet der Regelcontroller noch nicht und seine Ausgänge sind dementsprechend hochohmig, es wird somit nicht kontrolliert beeinflusst, welches Signal der Multiplexer durchschaltet. Anschließend wird auf dem Controller die Initialisierungsphase durchgeführt. Dabei werden alle Ausgänge abgeschaltet, weshalb der Multiplexer ein Signal mit geringerer Verstärkung durchschaltet. Im Anschluss an die Initialisierung wird eine Offsetkompensation durchgeführt, welche mehrere Perioden andauert. Ist die Offsetkompensation abgeschlossen, wird die optimale lineare Verstärkung schnell ermittelt und direkt durch den Multiplexer gesetzt. Eine länger andauernde Regelung der Verstärkung entfällt an dieser Stelle.

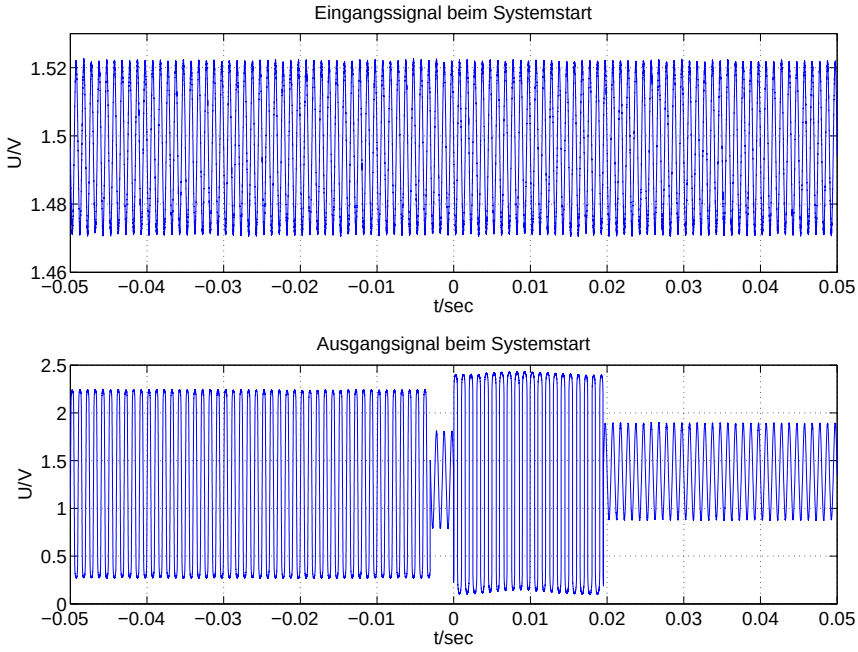



Abbildung 5.5: Verstärkungsauswahl bei Systemstart

Außerdem wurde die Auswahl der optimalen Verstärkung bei ab- und zunehmender Amplitude des Eingangssignals beobachtet. Dabei wurde der Spitzen-Spitzen-Wert des Eingangssignals zwischen $U_{hb1,pp} = 65\,\mathrm{mV}$ und $U_{hb1,pp} = 5\,\mathrm{mV}$, bei einer Signalfrequenz von $f = 107\,\mathrm{Hz}$ variiert. In Abbildung 5.6 ist der entsprechende Spannungsverlauf abgebildet. Zudem ist in der Abbildung das linear verstärkte Ausgangssignal dargestellt. Dabei ist deutlich zu sehen, dass bei Veränderung der Signalamplitude entsprechend die beste Verstärkung ausgewählt wird.

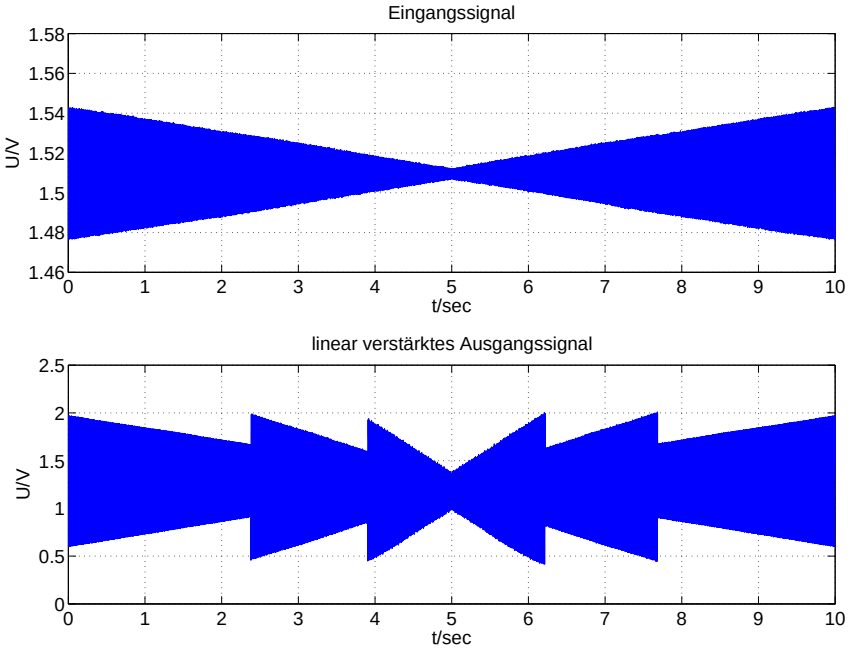



Abbildung 5.6: Verstärkungsauswahl bei kontinuierlicher Änderung der Signalamplitude

Zuletzt wurden untersucht, wie sich das System auf Sprünge der Signalamplitude verhält. Dazu wurde das System mit einem Sprung des Signals von $U_{hb1,pp} = 10\,\mathrm{mV}$ auf $U_{hb1,pp} = 20\,\mathrm{mV}$ bzw. von $U_{hb1,pp} = 20\,\mathrm{mV}$ auf $U_{hb1,pp} = 10\,\mathrm{mV}$ gespeist. In Abbildung 5.7 ist zu sehen, dass das System bereits eine Periode nach dem Sprung die Verstärkung angepasst hat. Dabei ist allerdings nicht garantiert, dass dies immer unmittelbar passiert. Nach wie vielen Perioden die Verstärkung angepasst wird, ist davon abhängig in welchem Zustand sich der Zustandsautomat befindet. Wie in Abbildung 5.8 zu sehen, kann die richtige Auswahl der Verstärkung somit einige Perioden dauern.

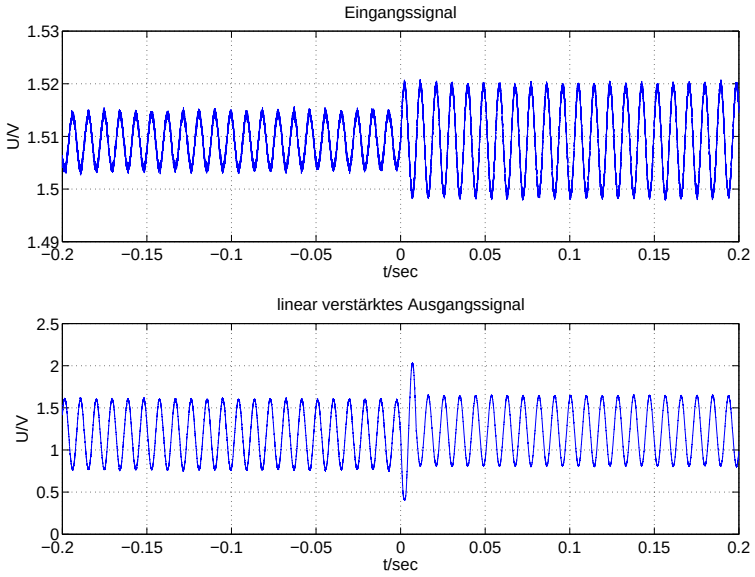



Abbildung 5.7: Messung positiver Amplitudensprung mit schneller Anpassung der Verstärkung

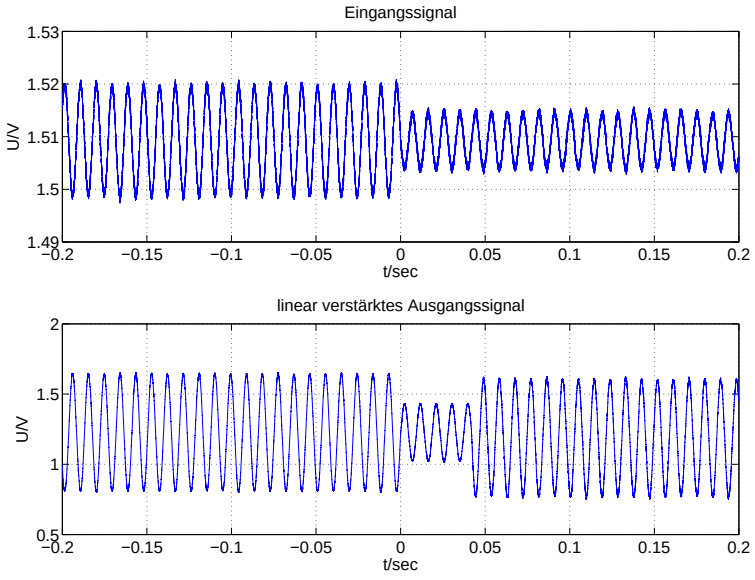



Abbildung 5.8: Messung negativer Amplitudensprung mit verzögerter Anpassung der Verstärkung

## 5.4 Offsetkompensation

Um die Funktionsweise der Offsetkompensation zu überprüfen, wurde das System mit einem Wechselspannungssignal gespeist, welches im oberen Graph in Abblidung 5.9 dargestellt ist. Die Spitzen-Spitzen-Spannung des Brückensignals betrug $U_{hb1,pp} = 25\,\mathrm{mV}$ bei einer Signalfrequenz von $f = 107\,\mathrm{Hz}$. Der Gleichanteil wurde über einen Zeitraum von $40\,\mathrm{s}$ um $40\,\mathrm{mV}$ variiert. Im zweiten Graphen der Abbildung ist das linear verstärkte Ausgangssignal dargestellt. Dabei sieht man deutlich, dass der Offset in regelmäßigen Abständen korrigiert wird. Die Abstände sind dadurch zu begründen, dass der Regelcontroller im Zustand *Active* eine Mittelung des Offsets über mehrere Perioden durchführt und erst anschließend korrigierend in das System eingreift.

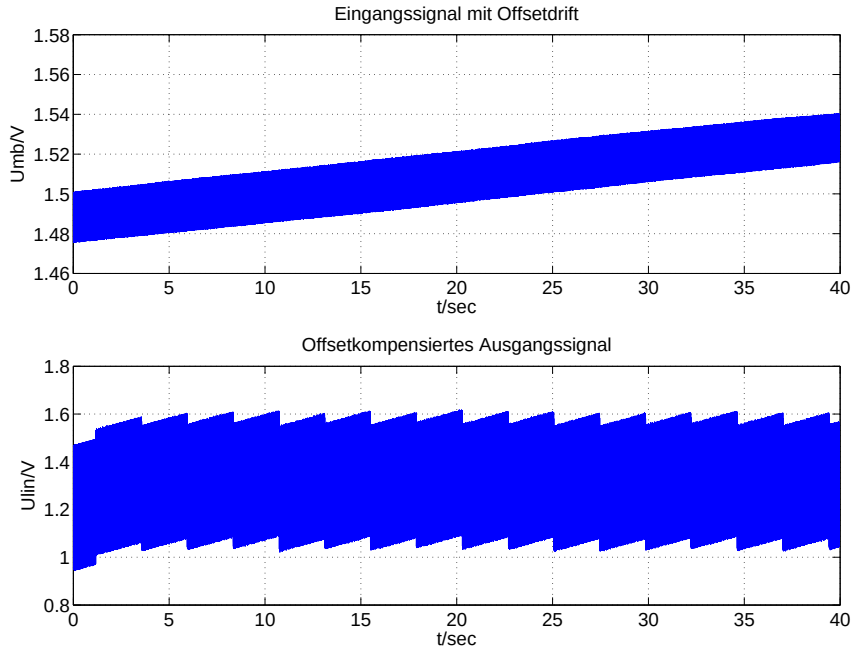



Abbildung 5.9: Offsetkompensation bei Offsetsweep (Die Signalfrequenz von $f = 107\,\mathrm{Hz}$ ist hoch gegenüber Zeitbereichsdarstellung)

Des Weiteren wurde untersucht, wie sich das System verhält, wenn sich der Gleichanteil des Eingangssignals $U_{hb1}$ schlagartig ändert. Dazu wurde das System wieder mit einem Wechselspannungssignal mit $f = 107\,\mathrm{Hz}$ und einem Spitze-Spitze-Wert von $U_{hb1,pp} = 20\,\mathrm{mV}$ gespeist. Der Gleichanteil wurde zudem um $5\,\mathrm{mV}$ verändert, was $50\,\%$ der maximalen Aussteuerung des Signals entspricht. In Abbildung 5.10 ist zu sehen, dass das System die Änderung des Gleichanteils erkennt, kompensiert und das linear verstärkte Ausgangssignal somit zurück in den optimalen Mittellage bringt.

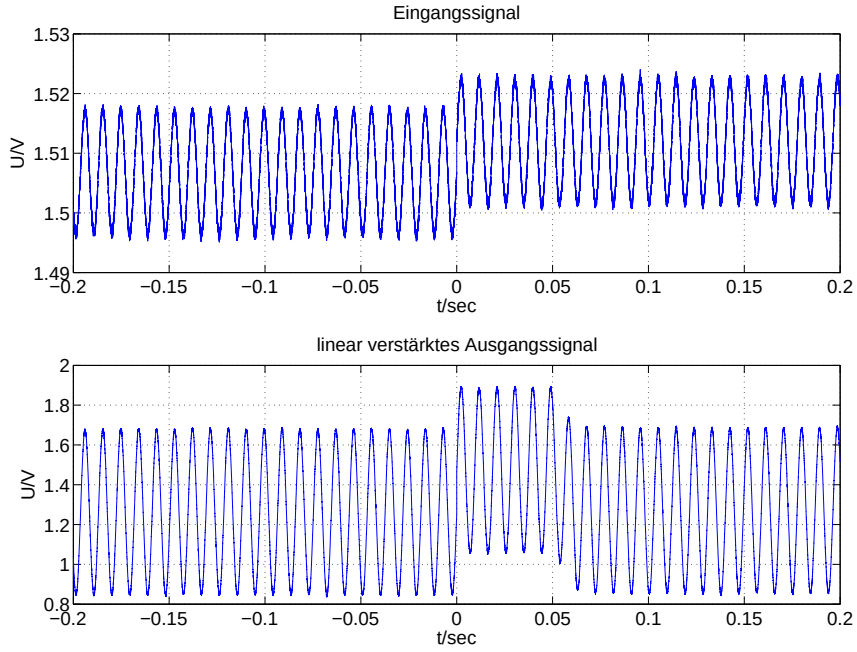



Abbildung 5.10: Offsetkompensation bei Offsetsprung

## 5.5 Brückenversorgung

Mit der entworfenen Hauptplatine besteht die Möglichkeit, die AMR-Messbrücke auf unterschiedliche Weise mit Energie zu versorgen. Zum einen kann die Messbrücke direkt mit einer konstanten Spannung von $3\,\mathrm{V}$ versorgt werden. Zum anderen besteht die Möglichkeit die Versorgungsspannung zwischen $0\,\mathrm{V}$ und maximal $5\,\mathrm{V}$ digital durch den Regelcontroller einzustellen. Eine dritte Möglichkeit ist die Verwendung einer Stromquelle anstatt einer Spannungsquelle.

Um die Funktion der Stromquelle zu verifizieren, wurde die Spannung am Ausgang des DACs zwischen $0$ und $2{,}5\,\mathrm{V}$ variiert und der Strom am Ausgang des Spannungs-Strom-Wandlers gemessen. Dazu diente ein $100\,\Omega$ Widerstand als Messwiderstand. Wie in Abbildung 5.11 dargestellt, ist der Zusammenhang zwischen Brückenstrom und eingestellter Spannung linear. Der Brückenstrom kann somit sehr gut durch den Regelcontroller gesteuert werden.

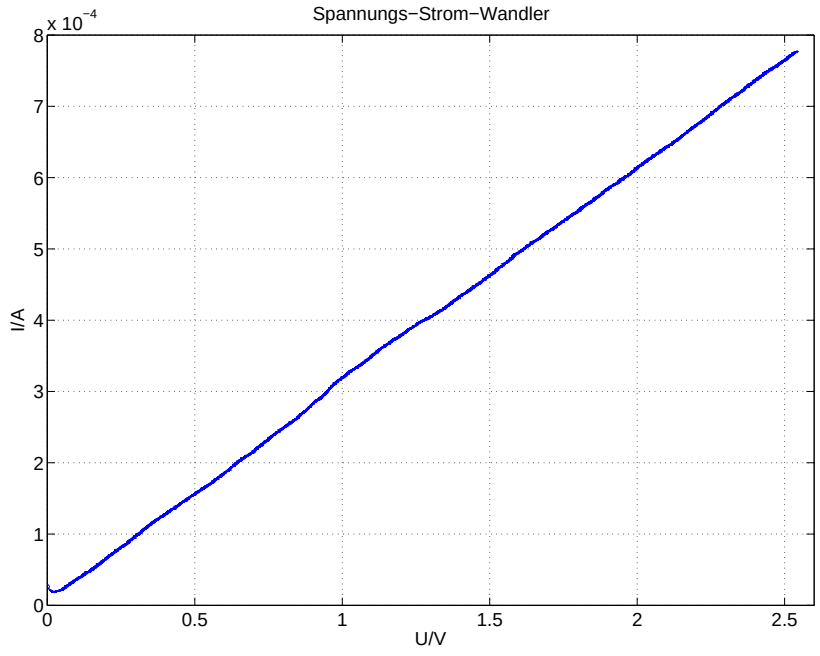



Abbildung 5.11: Kennlinie Spannungs-Strom-Wandler

## 5.6 Radmessplatz

Um eine Vergleichbarkeit des Demonstrators mit logarithmisch und linearer Verstärkerbank und Demonstrator mit der Verstärkerplatine nach [10] herzustellen, wurden mehrere Messungen mit beiden Systemen am Radmessplatz durchgeführt. Der Radmessplatz bietet die Möglichkeit eine AMR-Messbrücke in unterschiedlichen Distanzen und Verkippungswinkeln zu einem Encoderrad zu platzieren. Somit können mit dem Radmessplatz der Reihe nach mehrere Positionen angefahren werden und verschiedene Messwerte des Demonstrators aufgenommen werden, welche anschließend visualisiert werden.

Im folgenden sind Messergebnisse abgebildet, die am Radmessplatz durchgeführt wurden. Dabei wurde der Sensor um einen Winkel von $\varphi = \pm 15\,^{\circ}$ in der y-Achse verkippt und auf der z-Achse in einem Abstand von $0\,\mathrm{mm}$ bis $0{,}9\,\mathrm{mm}$ zum Encoderrad verfahren. Zur besseren Vergleichbarkeit sind immer zwei Messungen gegenüber gestellt. Wobei die erste Messung mit dem Demonstrator mit kombinierter logarithmisch und linearer Verstärkerbank durchgeführt wurde und die Zweite mit dem System nach [9], welches die Verstärkungsregelung und Offsetkompensation aufwendig in MATLAB durchführt.

In den beiden Abbildung 5.12 und 5.13, ist der gemessene HDI aufgetragen. Dieser ist nach [11] und [18] vergleichbar mit dem THD, beruht allerdings auf einem anderen Verfahren. Ein Vergleich der beiden Bilder zeigt, dass der HDI in beiden Fällen nahezu identisch ist. Eine zusätzliche Verzerrung durch die kombinierte logarithmisch und lineare Verstärkerbank, kann somit ausgeschlossen werden. Auch die gewählte Verstärkung, welche in Abbildung 5.14 und 5.15 dargestellt ist, zeigt dass die Systeme in vielen Bereichen gleich arbeiten. Weitere Messungen sind in Anhang B beigefügt.

Auffallend in den Messungen sind, zum einen, die mit einem weißen Kreuz markierten Felder, welche eine Frequenzverdopplung markieren. Zum anderen die vollständig weißen Felder in der jeweils zweiten Abbildung. In diesen Fällen wurde kein gültiger Messwert aufgenommen. Das kein Messwert aufgenommen wurde, ist dadurch zu begründen, dass das System ständig den Offset nachgeregelt hat und somit der Zeitinterval für die HDI Ermittlung zu kurz war. Davon abgesehen, weisen die beiden Grafiken allerdings eine hohe Ähnlichkeit auf.

Um ein Gefühl dafür zu bekommen, wie der logarithmische Verstärker arbeitet, sind in Abbildung 5.16 und 5.17 zwei verstärkte Sensorsignal im Zeitbereich dargestellt. Dabei wurde im zweiten Bild der Sensor soweit auf der y-Achse verkippt, dass eine Frequenzverdopplung des Signals auftritt.

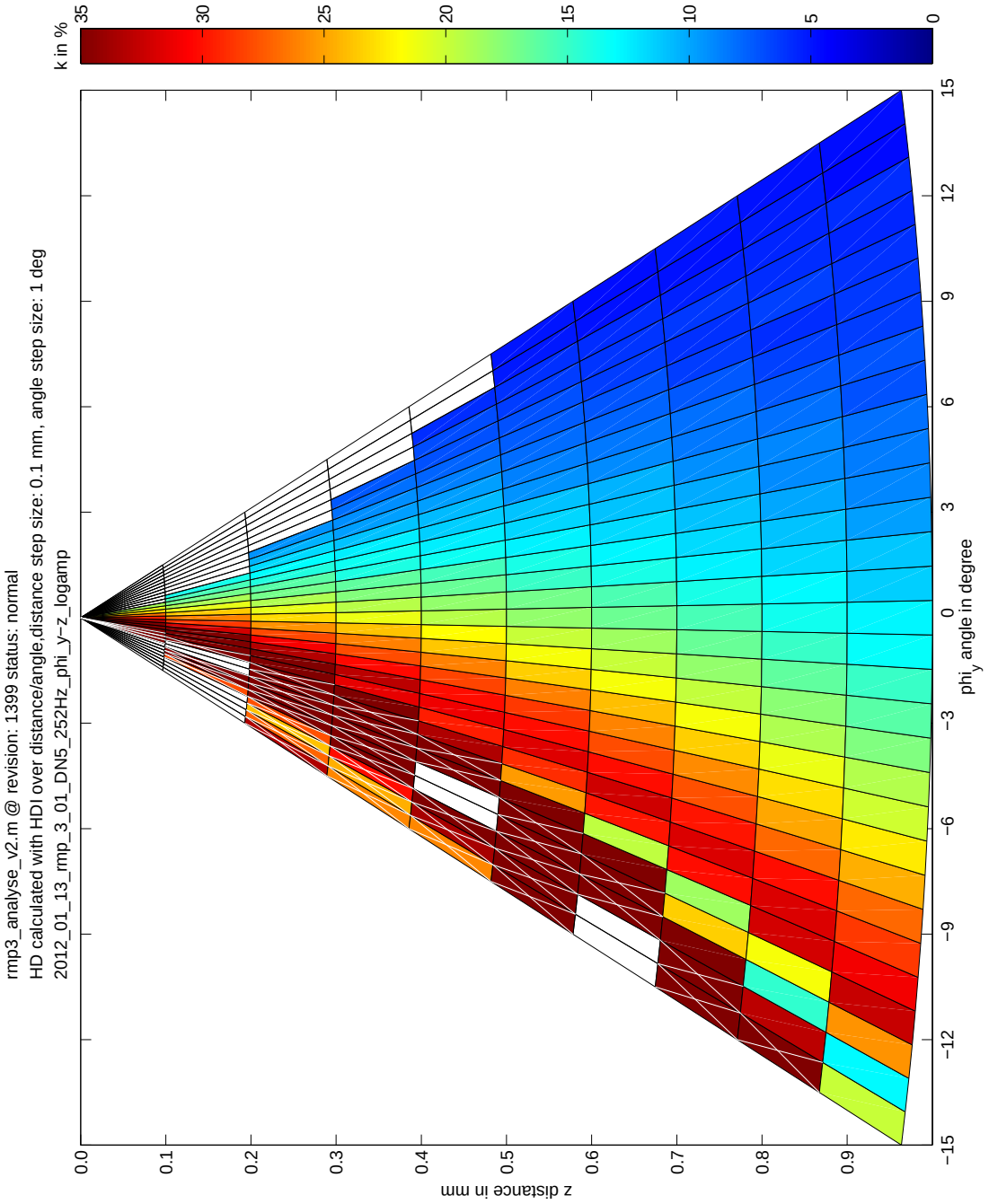



Abbildung 5.12: HDI mittels kombinierter logarithmisch und linearer Verstärkerbank

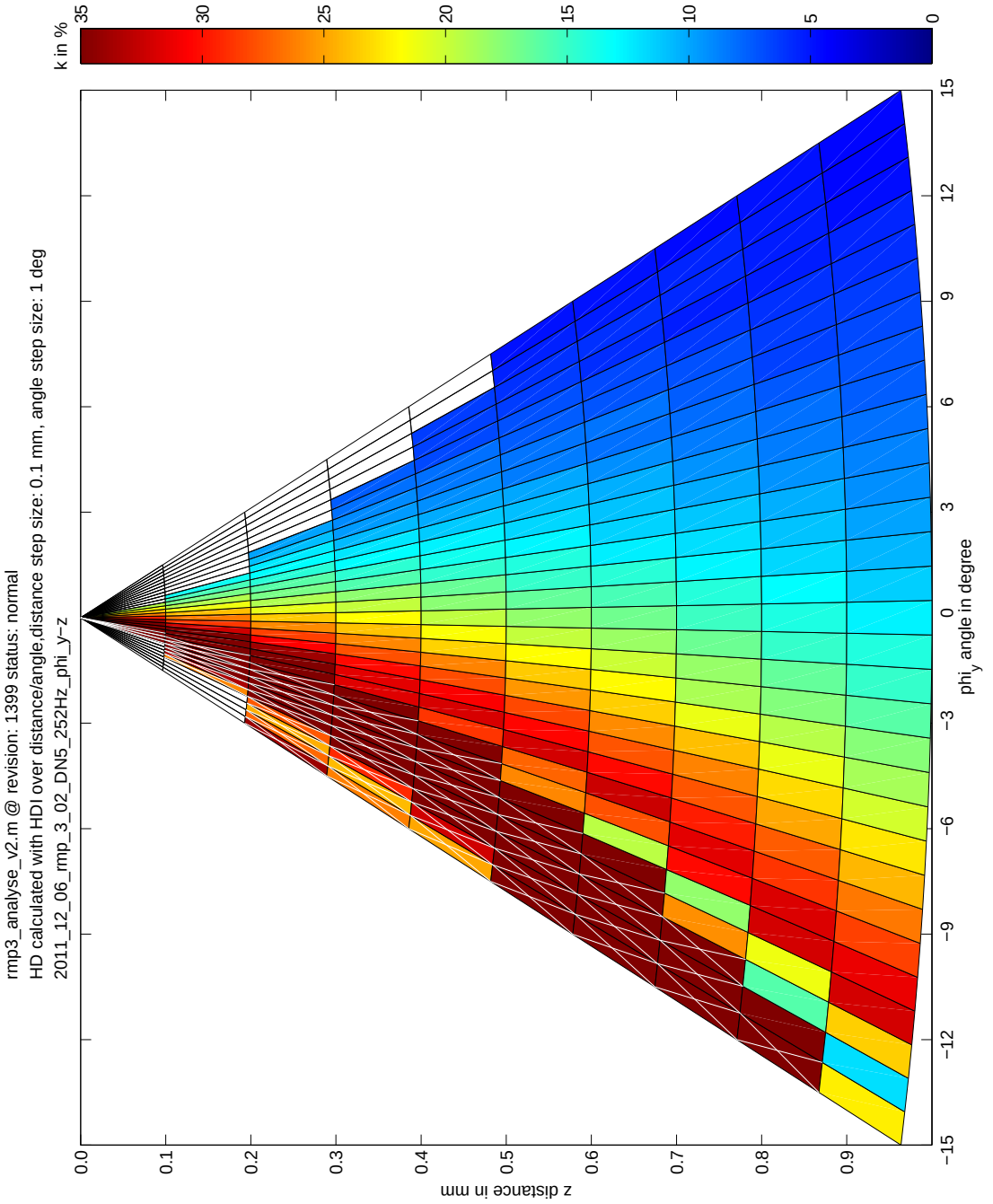



Abbildung 5.13: HDI mittels Verstärkerplatine nach [10]

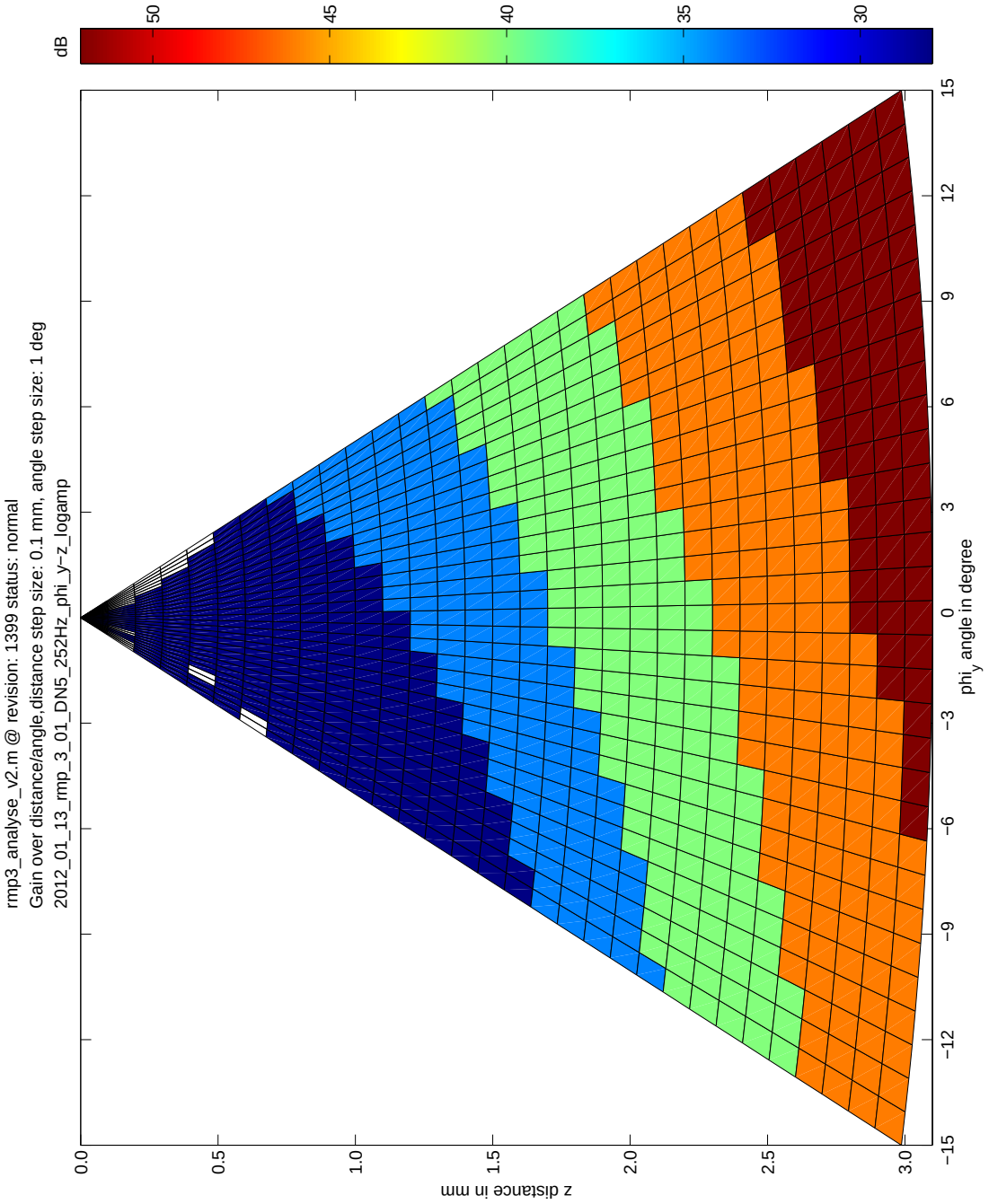



Abbildung 5.14: Verstärkung bei kombinierter logarithmisch und linearer Verstärkerbank

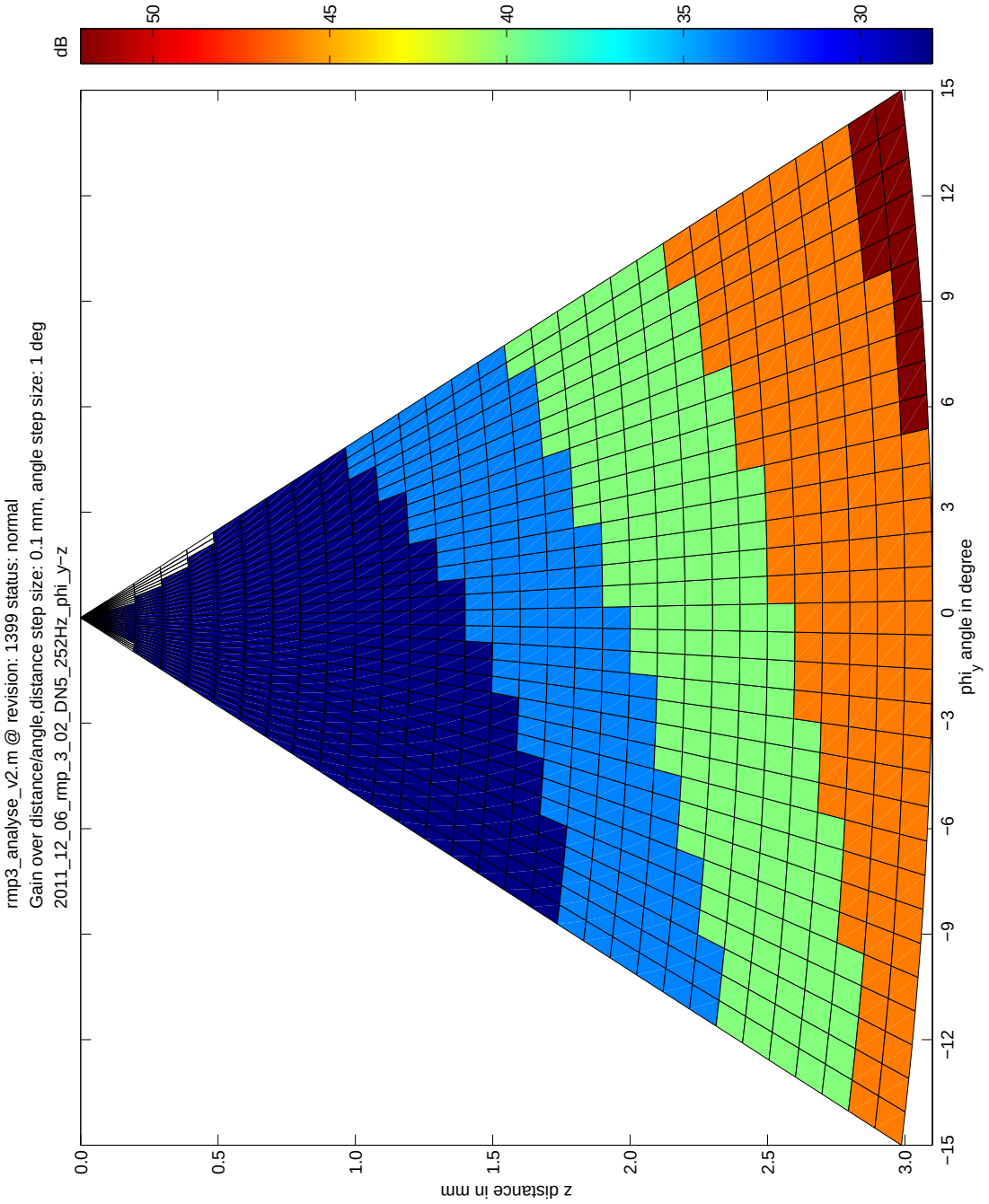



Abbildung 5.15: Verstärkung bei Verstärkerplatine nach [10]

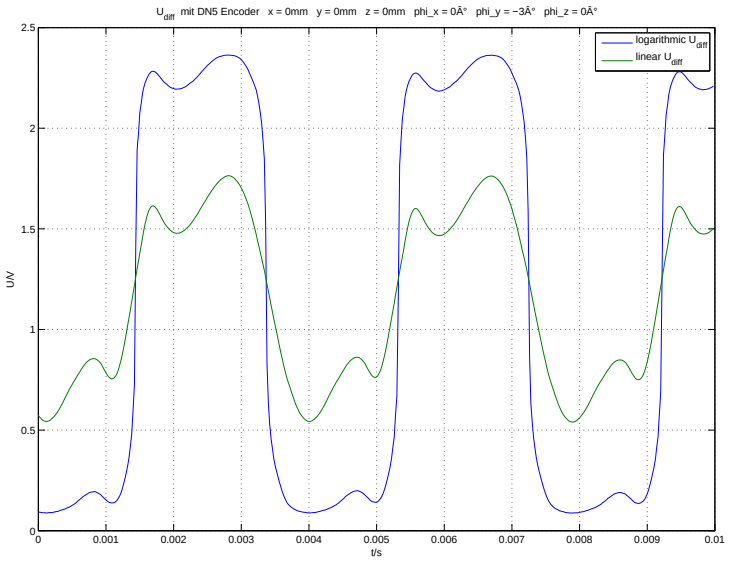



Abbildung 5.16: Linear und logarithmisch verstärktes und Sensorsignal im Zeitbereich

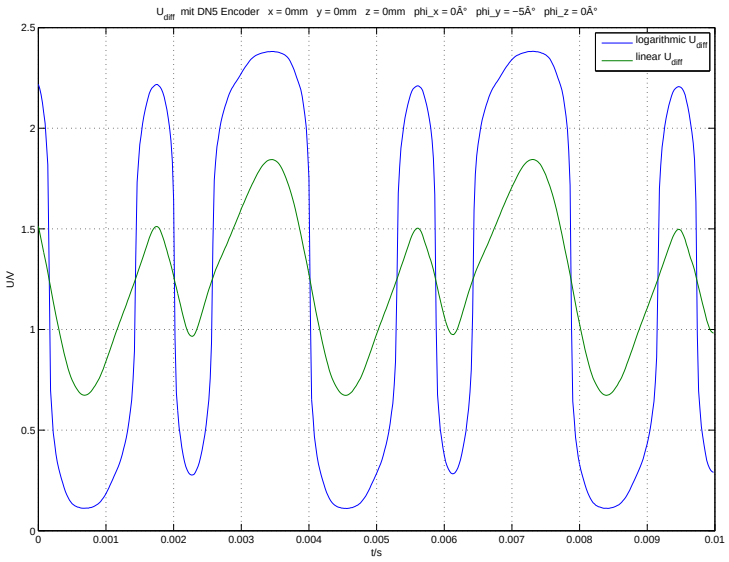



Abbildung 5.17: Linear und logarithmisch verstärktes und Sensorsignal im Zeitbereich, mit Frequenzverdopplung durch Verkippung

# 6 Bewertung und Ausblick

## 6.1 Bewertung

In dieser Arbeit wurde die Verstärkerplatine der bestehenden Demonstratorplattform erfolgreich durch eine kombinierte logarithmisch und lineare Verstärkerbank ersetzt. Die Verstärkerbank besteht dabei aus sieben parallel aufgebauten Verstärkerstufen, welche dieselbe Verstärkung des bestehenden Systems abdeckt. Durch die sanfte Begrenzung der einzelnen Verstärkerstufen und Aufsummieren aller Stufen wurde zudem erreicht, dass das Eingangssignal logarithmiert wird. Somit steht neben dem linear verstärkten Brückensignal auch das logarithmierte Signal zur Verfügung, welches den Vorteil hat, dass es immer im optimalen Aussteuerungsbereich des Analog-Digital-Umsetzers liegt und somit zu jedem Zeitpunkt auswertbar ist.

Für den Aufbau der kombinierten logarithmisch und linearen Verstärkerbank wurde sowohl eine Schaltung entwickelt, welche auf die Reihenschaltung von Verstärkern setzt, als auch eine Parallelschaltung der Verstärker. Die beiden Aufbauten wurden mit Hilfe des Simulationsprogramms PSpice simuliert und in ihren Eigenschaften untersucht. In der Praxis hat sich die Reihenschaltung der Verstärkerstufen allerdings als untauglich herausgestellt, da diese zum Schwingen neigt. Die Ursache hierfür konnte nicht lokalisiert werden.

Des Weiteren konnte die zweidimensionale Offset- und Verstärkungsregelung mit Hilfe der gleichzeitig zur Verfügung stehenden linear verstärkten Signale auf die Offsetkompensation reduziert werden. Hierbei wird das logarithmierte Eingangssignal auf dem Regelcontroller delogarithmiert und zur Auswahl der optimalen Verstärkung genutzt. Für die Auswahl der Verstärkung, als auch für die Offsetkompensation, mussten einige Änderungen am Regelcontroller vorgenommen werden, die die grundlegende Softwarestruktur allerdings nicht beeinflussen.

Auch stellt die Delogarithmierung mittels Lookup-Tabelle noch keine geeignete Methode dar, da diese in Form eines Speichers auf dem Sensorchip hinterlegt werden muss. Die hohen Anforderungen an den Sensor, gerade in Bezug auf Temperatur, erweisen sich hier als besonders kritisch.

Mit den Ergebnissen diese Arbeit konnte ein weiterer Beitrag zum Forschungsprojekt ”Experimentelle digitale Signalverarbeitung und Zustandserkennung für ABS-Sensoren” (ESZ-ABS) geleistet werden. Außerdem erschließen sich mit der entwickelten kombinierten lo-

garithmisch und linearen Verstärkerbank neue Möglichkeiten, was die Auswertung der Brückensignale, als auch die weitere Signalverarbeitung betrifft.

## 6.2 Ausblick

Da der verwendete Mikrocontroller nur über einen Smart-Comparator verfügt, ist eine Offsetkompensation nur für das Brückendifferenzsignal möglich. Sollen die Halbbrückensignale mit in die Zustandserkennung des Sensors einbezogen werden, empfiehlt es sich auf einen Mikrocontroller umzusteigen, der die Offsetkompensation aller drei Signale ermöglicht.

Des Weiteren empfiehlt sich eine genauere Betrachtung des logarithmierten Signals in Bezug auf die Berechnung des Klirrfaktors, da sich dadurch ggf. die Abtastung der linear verstärkten Signale einsparen lässt.

Mit dieser Arbeit wurde gezeigt dass die Regelung des Sensorsignals durch den Einsatz einer kombinierten logarithmisch und linearen Verstärkerbank deutlich vereinfacht werden kann. Es bleibt allerdings offen, ob die gewonnen Vorteile, den Mehraufwand rechtfertigen. Gerade in Hinblick auf eine Chipintegration, müssen diese beiden Faktoren gegeneinander abgewogen werden.

# Literaturverzeichnis


[1] AG, Daimler: *Requirement Specifications for Standardized Interface for Wheel Speed Sensors with Additional Information „AK-Protokoll“*. 4.0. Februar 2008

[2] ANALOG DEVICES: *AD5627R/AD5647R/AD5667R, AD5627/AD5667 - Dual, 12-/14-/16-Bit nanoDACs with 5 ppm/ °C On-Chip Reference, $I^2C$ Interface*. Version: 2007. http://www.analog.com/static/imported-files/data_sheets/AD5627R_5647R_5667R_5627_5667.pdf, Abruf: 17.01.2012

[3] ANALOG DEVICES: *ADG708/ADG709 - CMOS, 1.8V to 5.5V/±2.5V,3Ω Low Voltage 4-/8-Channel Multiplexers*. Version: C, 2009. http://www.analog.com/static/imported-files/data_sheets/ADG708_709.pdf, Abruf: 26.01.2012

[4] ANALOG DEVICES: *AD8226 - Wide Supply Range, Rail-to-Rail Output Instrumentation Amplifier*. Version: B, 2011. http://www.analog.com/static/imported-files/data_sheets/ADG884.pdf, Abruf: 26.01.2012

[5] ANALOG DEVICES: *AD8605/AD8606/AD8608 - Precision, Low Noise, CMOS, Rail-to-Rail, Input/Output Operational Amplifiers*. Version: K, 2011. http://www.analog.com/static/imported-files/data_sheets/AD8605_8606_8608.pdf, Abruf: 02.12.2011

[6] BOLL, R. ; OVERSHOTT, K. J.: Magnetic Sensors. In: GÖPEL, W. (Hrsg.) ; HESSE, J. (Hrsg.) ; ZEMEL, J. N. (Hrsg.): *Sensors a Comprehensive Survey* Bd. 5. Weinheim : VCH, 1989. – ISBN 3–527–26771–9. – insb. Kap. 9 von U. Dibbern

[7] BRONSTEIN, I.N. ; SEMENDJAJEW, K.A. ; MUSIOL, G. ; MÜHLIG, H.: *Taschenbuch der Mathematik*. 7, Aufl. Frankfurt am Main : Harri Deutsch, 2008. – ISBN 978–3–8171–2017–8

[8] INFINEON TECHNOLOGIES: *BAV199 Silicon Low Leakage Diode*. Version: 2004. http://www.infineon.com/dgdl/bav199series.pdf?folderId=db3a30431400ef6801141c748874044e&fileId=db3a30431400ef6801141cba733104e5, Abruf: 26.01.2012

[9] IVANOV, Kalin: *Fehlersichere Automatisierung eines Encoder-Messplatzes zur Untersuchung von ABS-Sensoren*, Hochschule für Angewandte Wissenschaften Hamburg, laufende Diplomarbeit

[10] JEGENHORST, N.: *Entwicklung eines Controllersystems zur Zustandserkennung von ABS-Sensoren*, Hamburg, Hochschule für Angewandte Wissenschaften, Diplomarbeit, 2009

[11] KOCH, Lennart: *Aufwandsminimierte Schätzung von Harmonischen zur Zustandsbestimmung von ABS-Sensoren*, Hochschule für Angewandte Wissenschaften Hamburg, Diplomarbeit, April 2010

[12] KREY, Martin ; RIEMSCHNEIDER, Karl-Ragmar: Signalverarbeitung zur Funktionsdiagnose bei magnetischen Sensoren. In: *E-Forum Magazin des Departments Informations- und Elektrotechnik* (2011). – und pers. Kommunikation. Hamburg, Hochschule für Angewandte Wissenschaften

[13] KRUCKER, G.: *Analoge Systeme 2 (ELA5)*. Version: August 2004. http://www.krucker.ch/skripten-uebungen/AnSys/ELA5-OpAmp.pdf

[14] M. BIEGERT: *Learning How Electronic Parts Work*. Version: 2011. http://mathscinotes.wordpress.com/2011/04/28/learning-how-electronic-parts-work/, Abruf: 30.12.2011

[15] PHILIPS: *KMI22/1 Rotational speed sensor for extended air gap application and direction detection*. 2000

[16] PHILIPS SEMICONDUCTORS: *General Rotational speed measurement*. 1998

[17] PHILIPS SEMICONDUCTORS: *General Magnetoresistive sensors for magnetic field measurement*. 2000

[18] POPPINGA, Heiko: *Controller-Implementation und messtechnische Erprobung der Signalverarbeitung für die Diagnosefunktion von ABS-Sensoren*, Hochschule für Angewandte Wissenschaften Hamburg, Bachelorarbeit, Mai 2011

[19] STAHL, M.: *Controllersystem zur Verstärkungsregelung und Offsetkompensation für ABS-Sensoren mit Diagnosefunktion*. Hamburg, Hochschule für Angewandte Wissenschaften, Bachelorthesis, 2010

[20] TIETZE, U. ; SCHENK, Ch. ; GAMM, E.: *Halbleiter-Schaltungstechnik*. 13, Aufl. Heidelberg : Springer, 2010. – ISBN 978–3–642–01621–9

# A Simulation

## A.1 Parallelschaltung

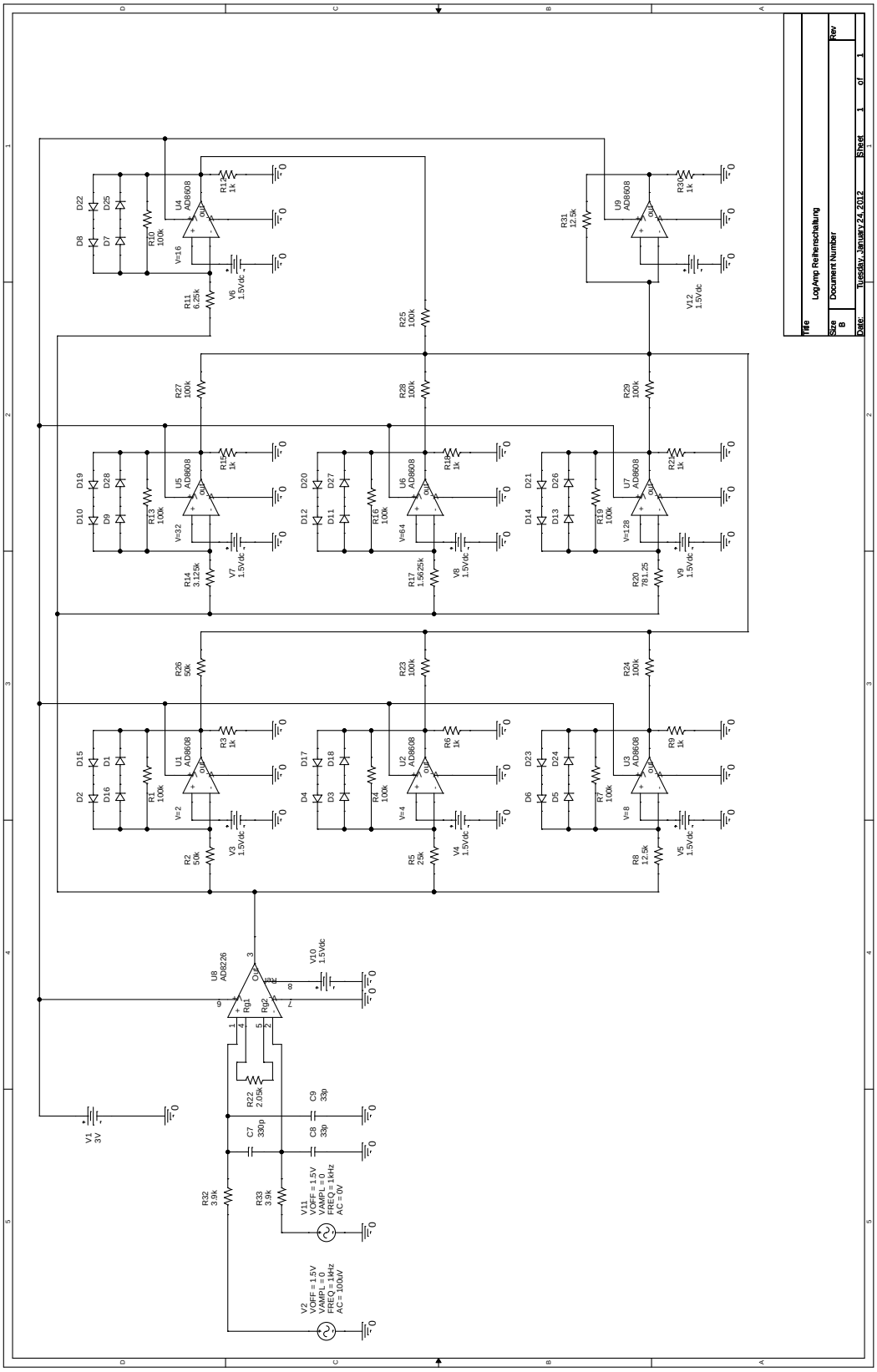



Abbildung A.1: PSpice Schaltbild Parallelschaltung

## A.2 Reihenschaltung

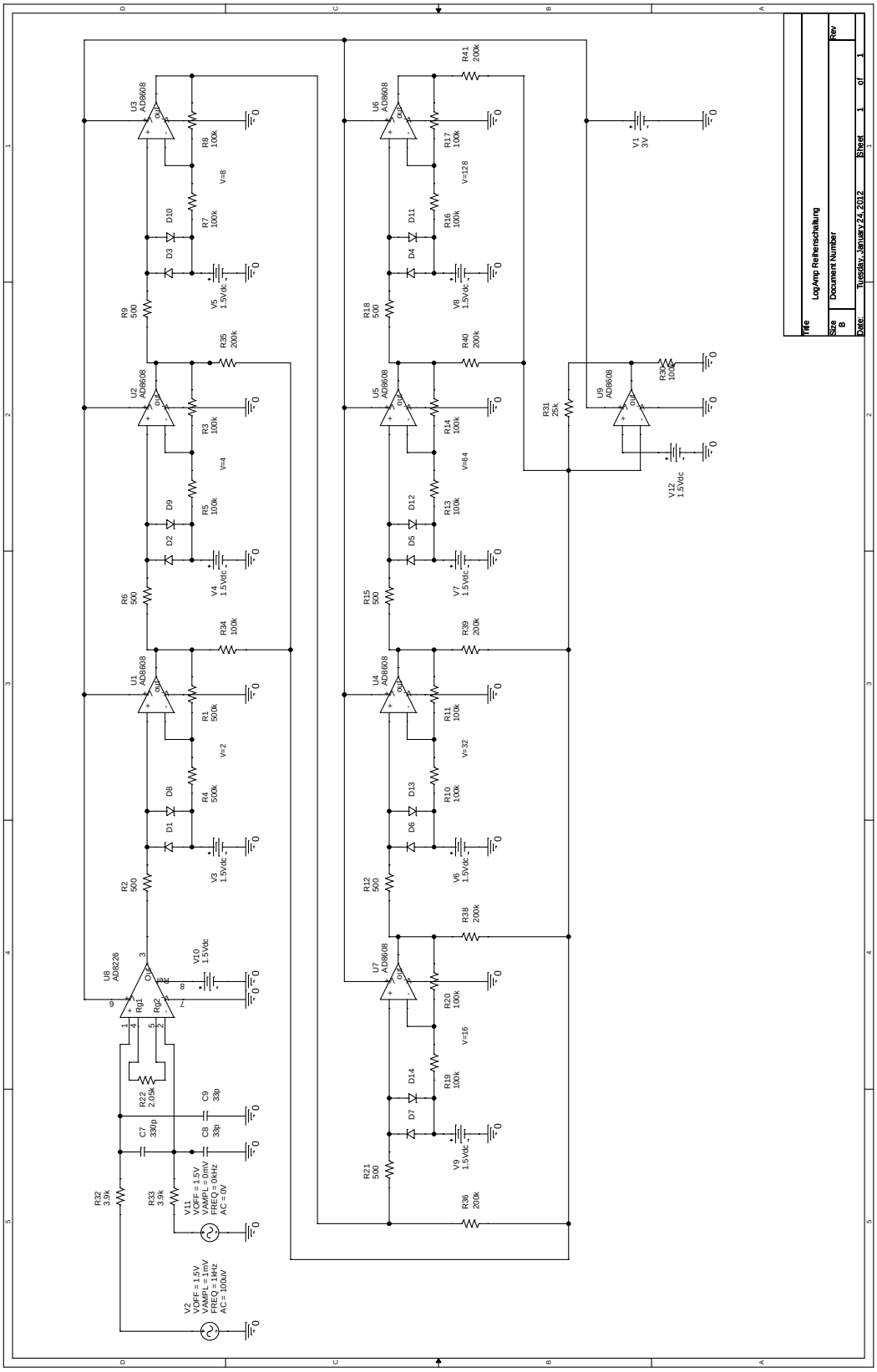



Abbildung A.2: PSpice Schaltbild Reihenschaltung

## A.3 Simulationsergebnisse

### A.3.1 Amplituden- und Phasengang Reihenschaltung

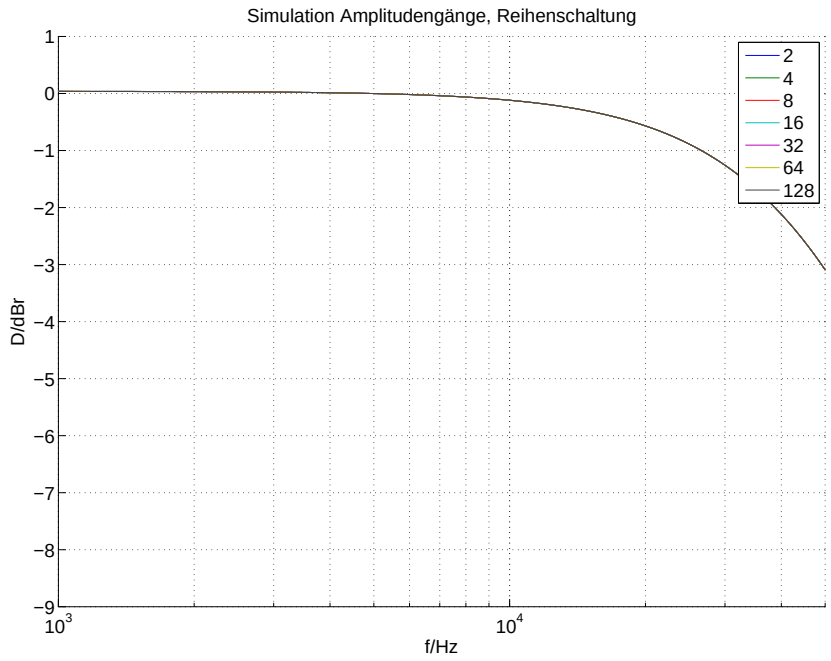



Abbildung A.3: PSpice Simulation Amplitudengang Reihenschaltung

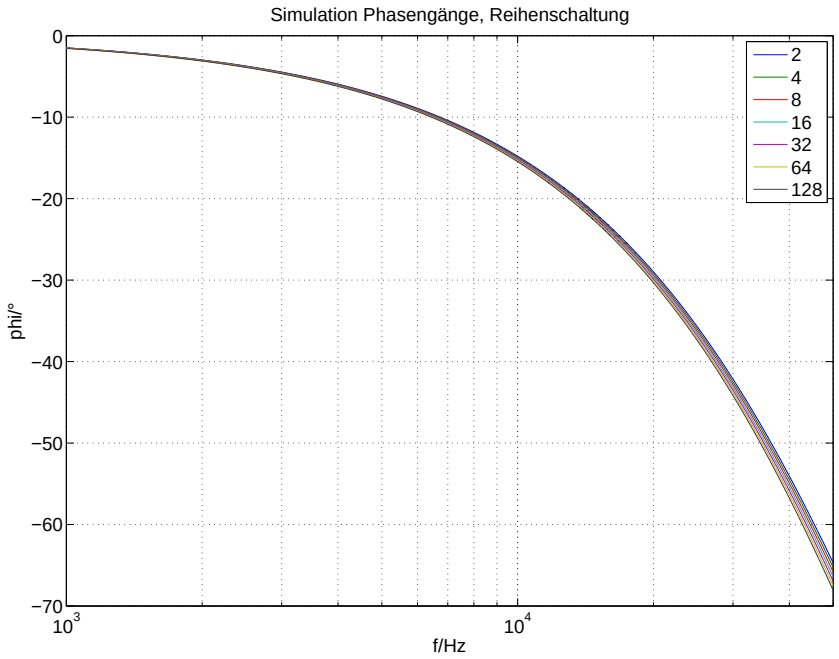



Abbildung A.4: PSpice Simulation Phasengang Reihenschaltung

### A.3.2 Amplituden- und Phasengang Parallelschaltung

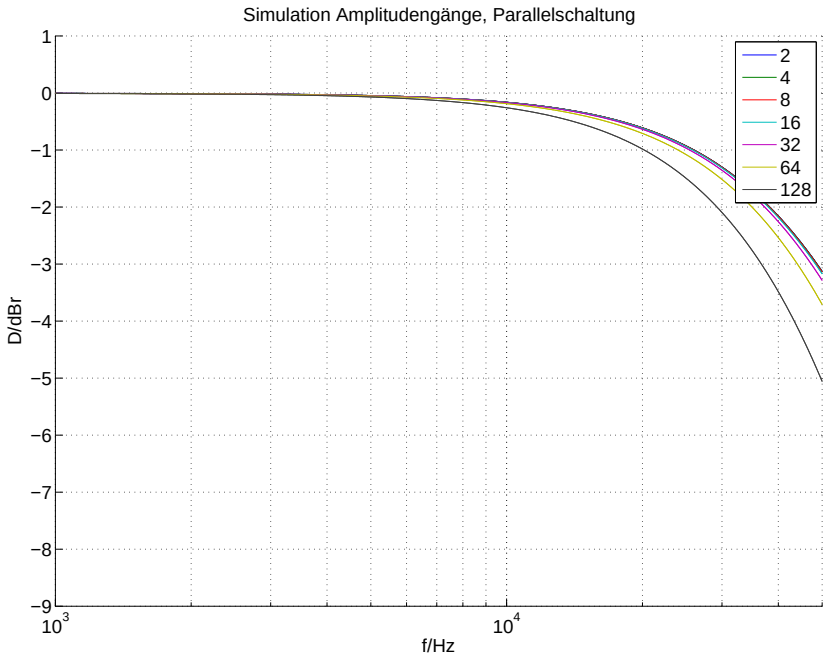



Abbildung A.5: PSpice Simulation Amplitudengang Parallelschaltung

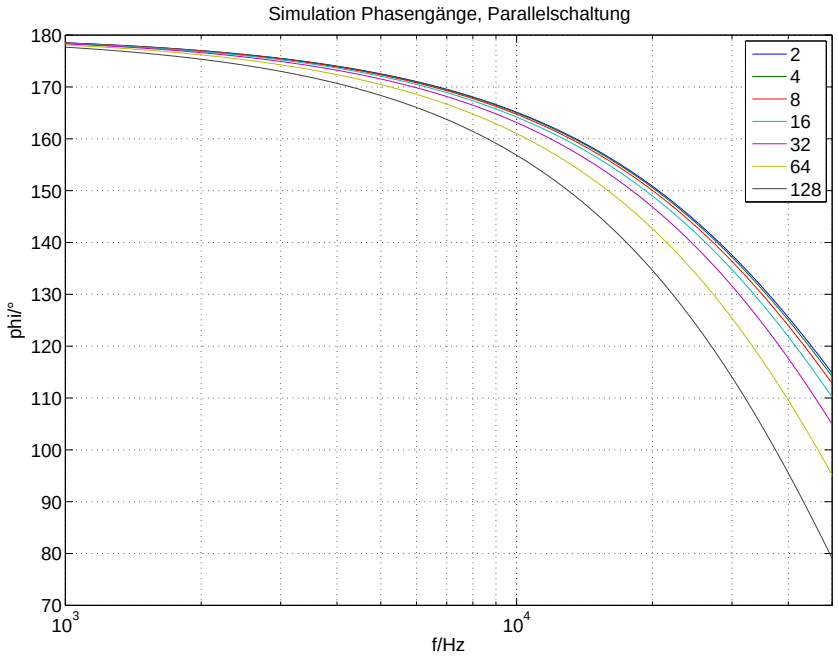



Abbildung A.6: PSpice Simulation Phasengang Parallelschaltung

# B Weitere Messungen

## B.1 Radmessplatz

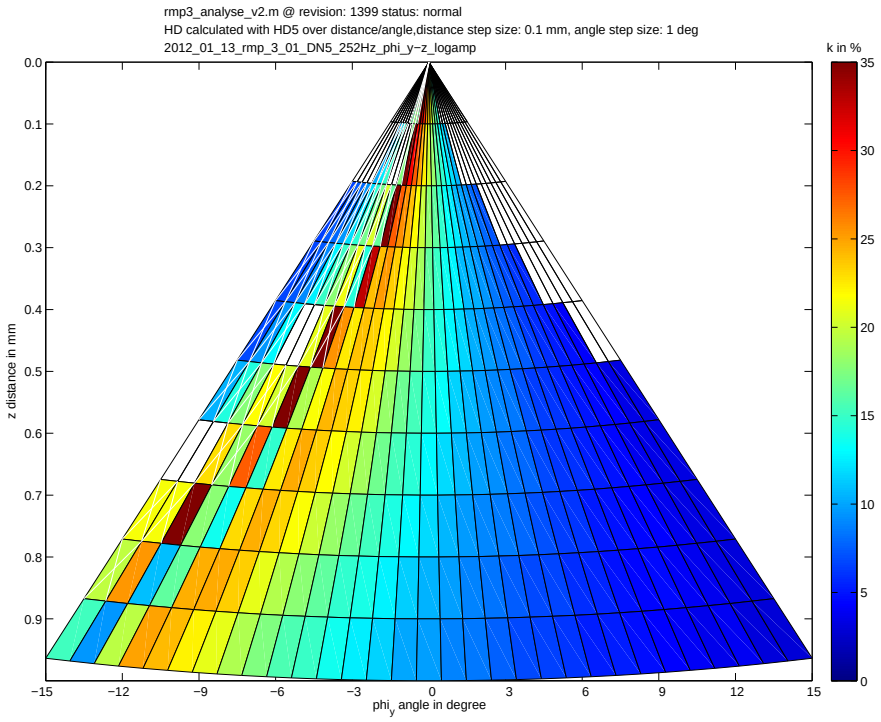



Abbildung B.1: HD5 bei kombinierter logarithmisch und linearen Verstärkerbank

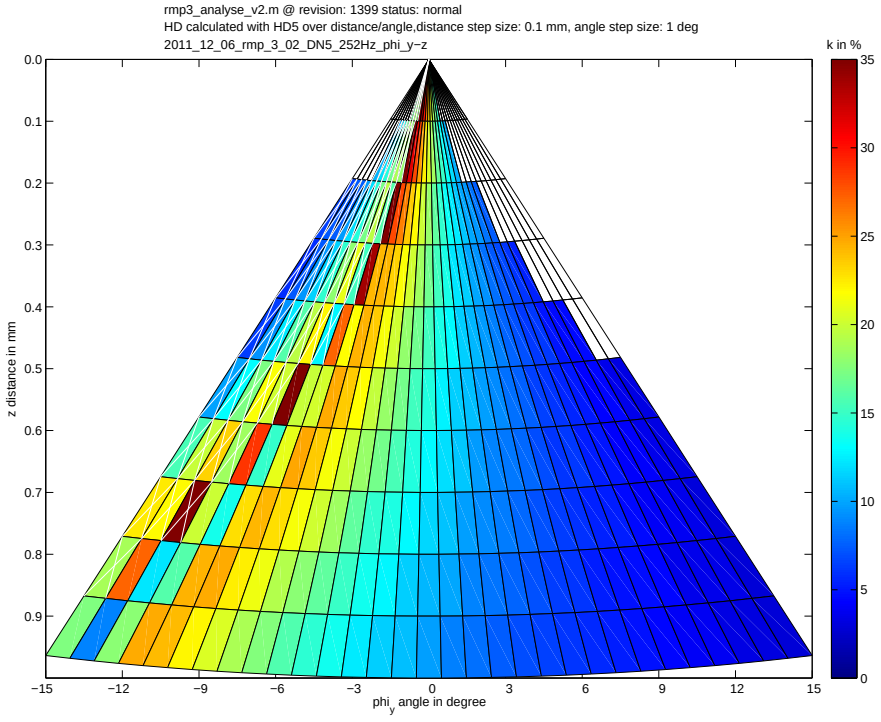



Abbildung B.2: HD5 bei Verstärkerplatine nach [10]

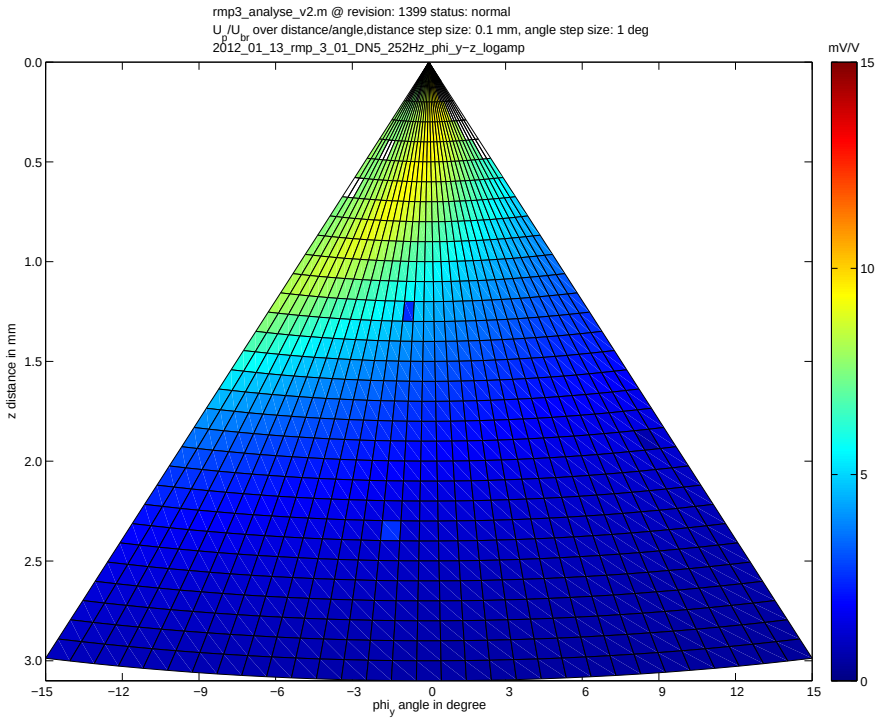



Abbildung B.3: U-Peak bei kombinierter logarithmisch und linearen Verstärkerbank

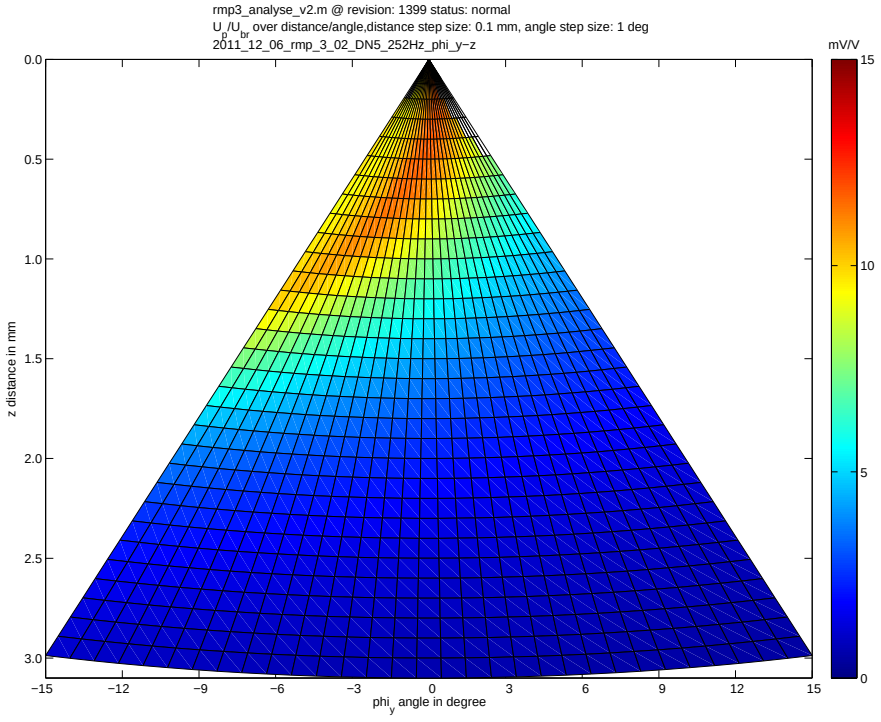



Abbildung B.4: U-Peak bei Verstärkerplatine nach [10]

# C Schaltpläne

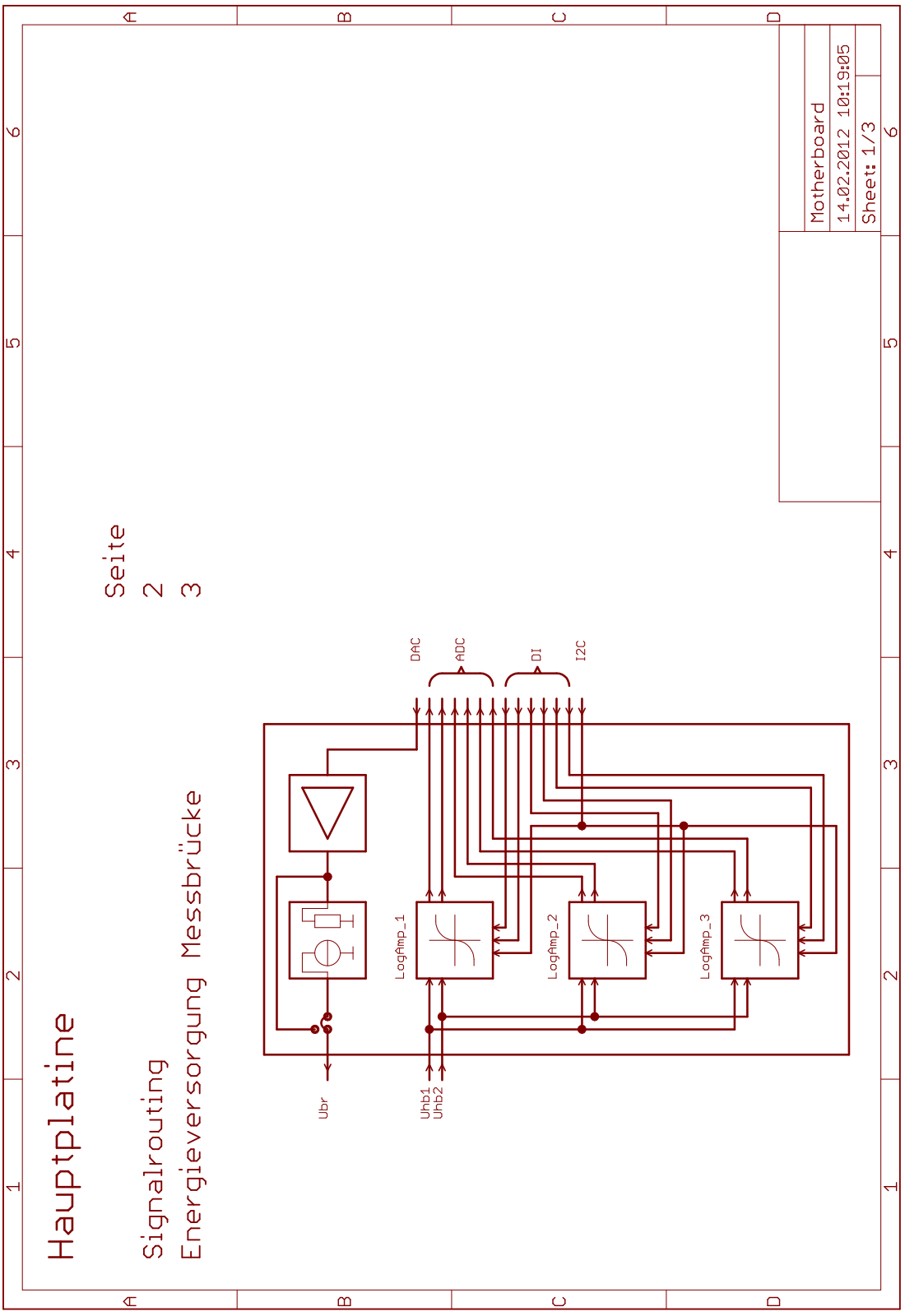
## C.1 Pinbelegung Regelcontroller

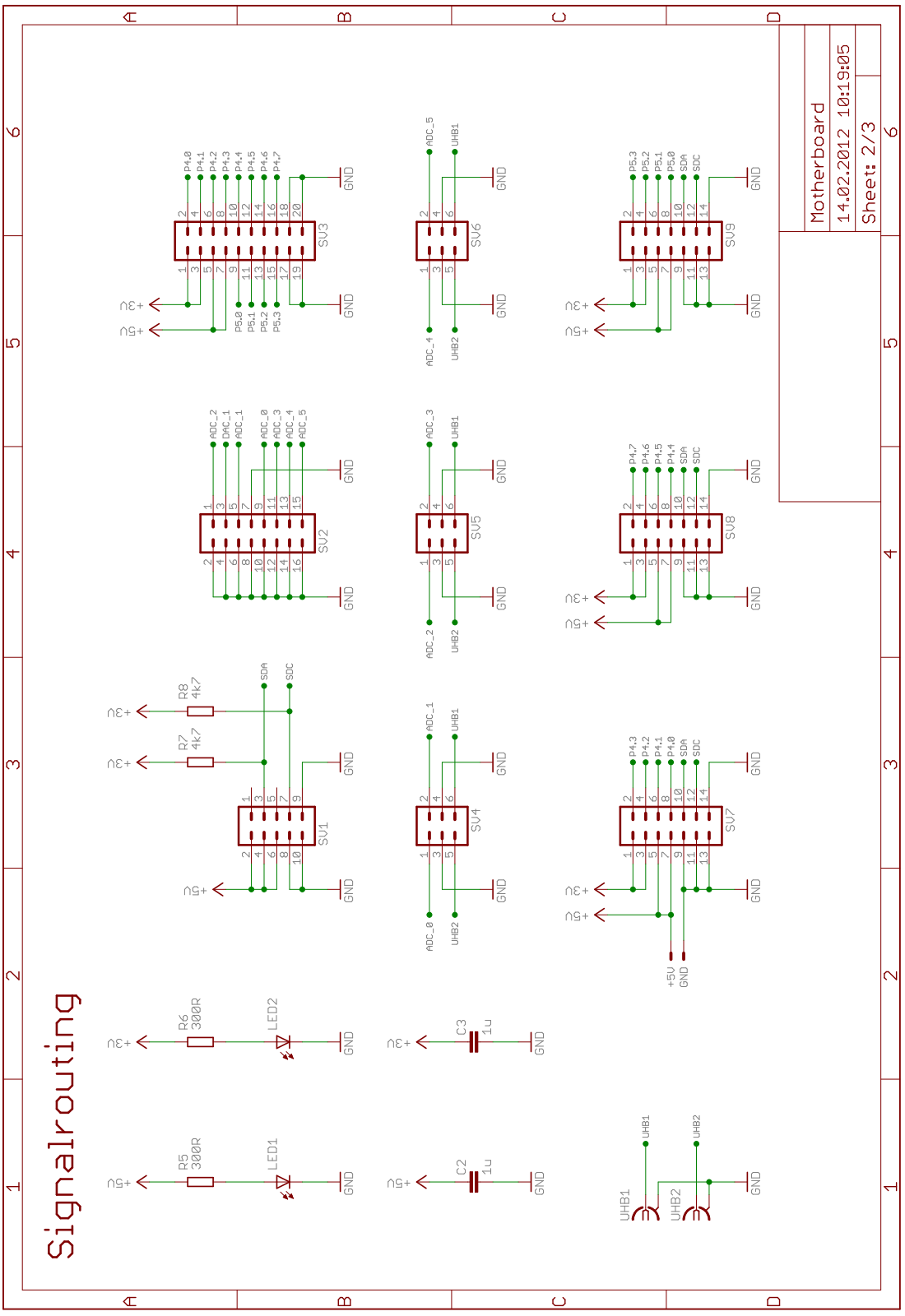
| Port | Mögliche Funktion | Funktion | Datenrichtung | Verwendung |
|---|---|---|---|---|
| 1 | P1.0 / TACLK | P1.0 | input | Jumper 1 |
| | P1.1 / TA0 | P1.1 | input | Jumper 2 |
| | P1.2 / TA1 | P1.2 | input | Jumper 3 |
| | P1.3 / TA2 | P1.3 | input | Jumper 4 |
| | P1.4 / SMCLK | P1.4 | output | LED 1 |
| | P1.5 / TA0 | P1.5 | output | LED 2 |
| | P1.6 / TA1 | P1.6 | output | LED 3 |
| | P1.7 / TA2 | P1.7 | output | LED 4 |
| 2 | P2.0 / ACLK | P2.0 | input | SW_3 |
| | P2.1 / TAINCLK | P2.1 | input | |
| | P2.2 / CAOUT/TA0 | P2.2 | input | COMP_OUT |
| | P2.3 / CA0/TA1 | P2.3 | input | LogAmp1 |
| | P2.4 / CA1/TA2 | P2.4 | input | COMP_IN |
| | P2.5 / Rosc | P2.5 | input | COMM_6 |
| | P2.6 / ADC12CLK/DMAE0 | P2.6 | input | COMM_7 |
| | P2.7 / TA0 | P2.7 | input | COMM_8 |
| 3 | P3.0 / STE0 | P3.0 | input | |
| | P3.1 / SIMO0/SDA | SDA | input/output | I2C |
| | P3.2 / SOMI0 | P3.2 | input | |
| | P3.3 / UCCLK0/SCL | SCL | output | I2C |
| | P3.4 / UTXD0 | P3.4 | input | COMM_4 |
| | P3.5 / URXD0 | P3.5 | input | COMM_5 |
| | P3.6 / UTXD1 | UTXD1 | input | UART-TX |
| | P3.7 / URXD1 | URXD1 | input | UART-RX |

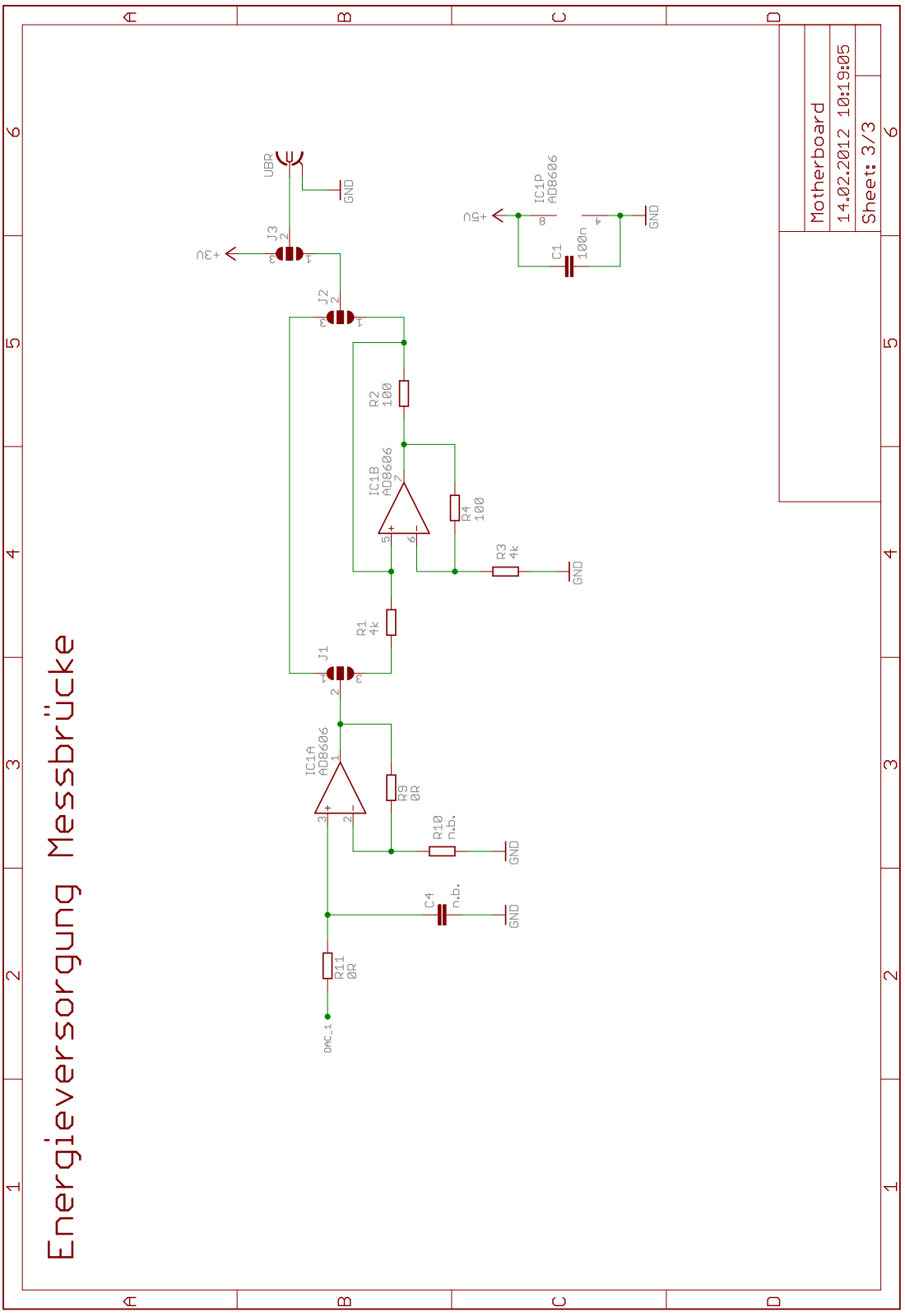
| Port | Mögliche Funktion | Funktion | Datenrichtung | Verwendung |
|---|---|---|---|---|
| 4 | P4.0 / TB0 | P4.0 | output | AMP1 / MUX-AD0 |
| | P4.1 / TB1 | P4.1 | output | AMP1 / MUX-AD1 |
| | P4.2 / TB2 | P4.2 | output | AMP1 / MUX-AD2 |
| | P4.3 / TB3 | P4.3 | output | AMP1 / Switch |
| | P4.4 / TB4 | P4.4 | output | AMP2 / MUX-AD0 |
| | P4.5 / TB5 | P4.5 | output | AMP2 / MUX-AD1 |
| | P4.6 / TB6 | P4.6 | output | AMP2 / MUX-AD2 |
| | P4.7 / TBCLK | P4.7 | output | AMP2 / Switch |
| 5 | P5.0 / STE1 | P5.0 | output | AMP3 / MUX-AD0 |
| | P5.1 / SIMO1 | P5.1 | output | AMP3 / MUX-AD1 |
| | P5.2 / SOMI1 | P5.2 | output | AMP3 / MUX-AD2 |
| | P5.3 / UCLK1 | P5.3 | output | AMP3 / Switch |
| | P5.4 / MCLK | MCLK | input | CPU-Takt |
| | P5.5 / SMCLK | SMCLK | input | Peripherie-Takt |
| | P5.6 / ACLK | P5.6 | input | |
| | P5.7 / TBOUTH/SVSOUT | P5.7 | input | |
| 6 | P6.0 / A0 | A0 | input | LinAmp1 |
| | P6.1 / A1 | A1 | input | LogAmp1 |
| | P6.2 / A2 | A2 | input | LinAmp2 |
| | P6.3 / A3 | A3 | input | LogAmp2 |
| | P6.4 / A4 | A4 | input | LinAmp3 |
| | P6.5 / A5 | A5 | input | LogAmp3 |
| | P6.6 / A6/DAC0 | P6.6 | output | COMP_IN |
| | P6.7 / A7/DAC1/SVSIN | DAC1 | output | Brückenspeisung |

Tabelle C.1: Zuordnung, Pins des Regelcontrollers

## C.2 Hauptplatine


Hauptplatine
Signalrouting
Energieversorgung Messbrücke
Seite
2
3
Motherboard
14.02.2012 10:19:05
Sheet: 1/3
DAC
ADC
DI
I2C
LogAmp_1
LogAmp_2
LogAmp_3
Ubr
Uhb1
Uhb2


Signalrouting
Motherboard
14.02.2012 10:19:05
Sheet: 2/3
SV1
SV2
SV3
SV4
SV5
SV6
SV7
SV8
SV9
R5 300R
R6 300R
R7 4k7
R8 4k7
LED1
LED2
C2 1u
C3 1u
UHB1
UHB2
GND
+5V
+3V
SDA
SDC


Energieversorgung Messbrücke
Motherboard
14.02.2012 10:19:05
Sheet: 3/3
UBR
GND
J3
J2
J1
IC1A AD8606
IC1B AD8606
IC1P AD8606
R1 4k
R2 100
R3 4k
R4 100
R9 0R
R10 n.b.
R11 0R
C1 100n
C4 n.b.
DAC_1

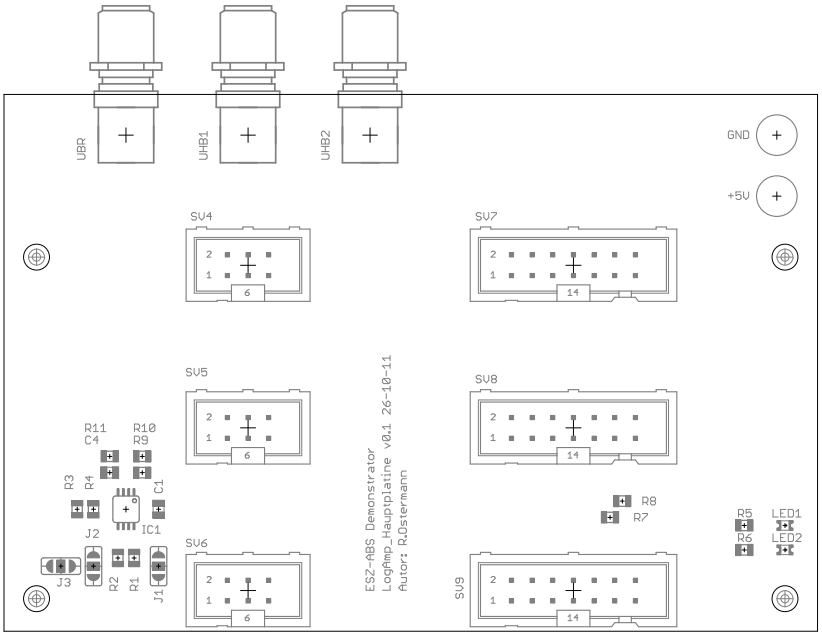



Abbildung C.1: Bestückung Oberseite Hauptplatine

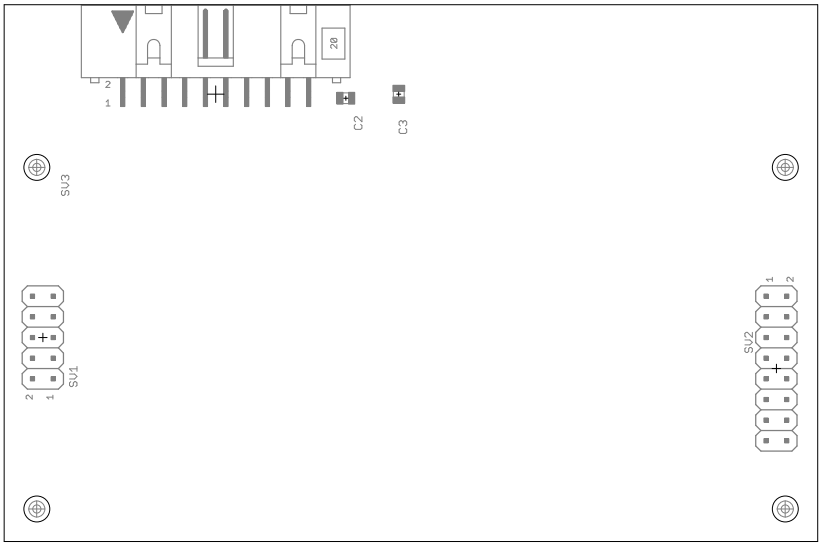
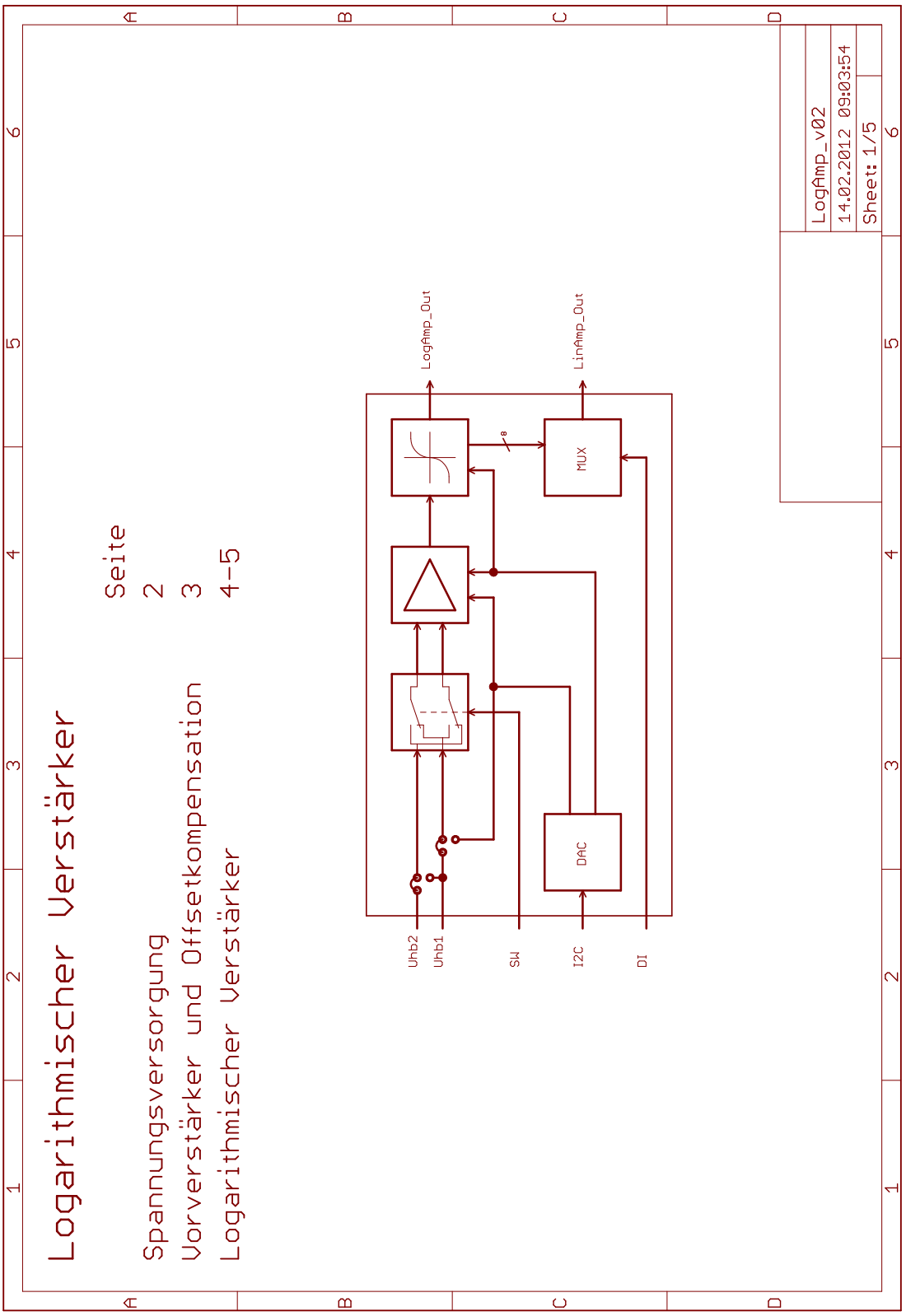
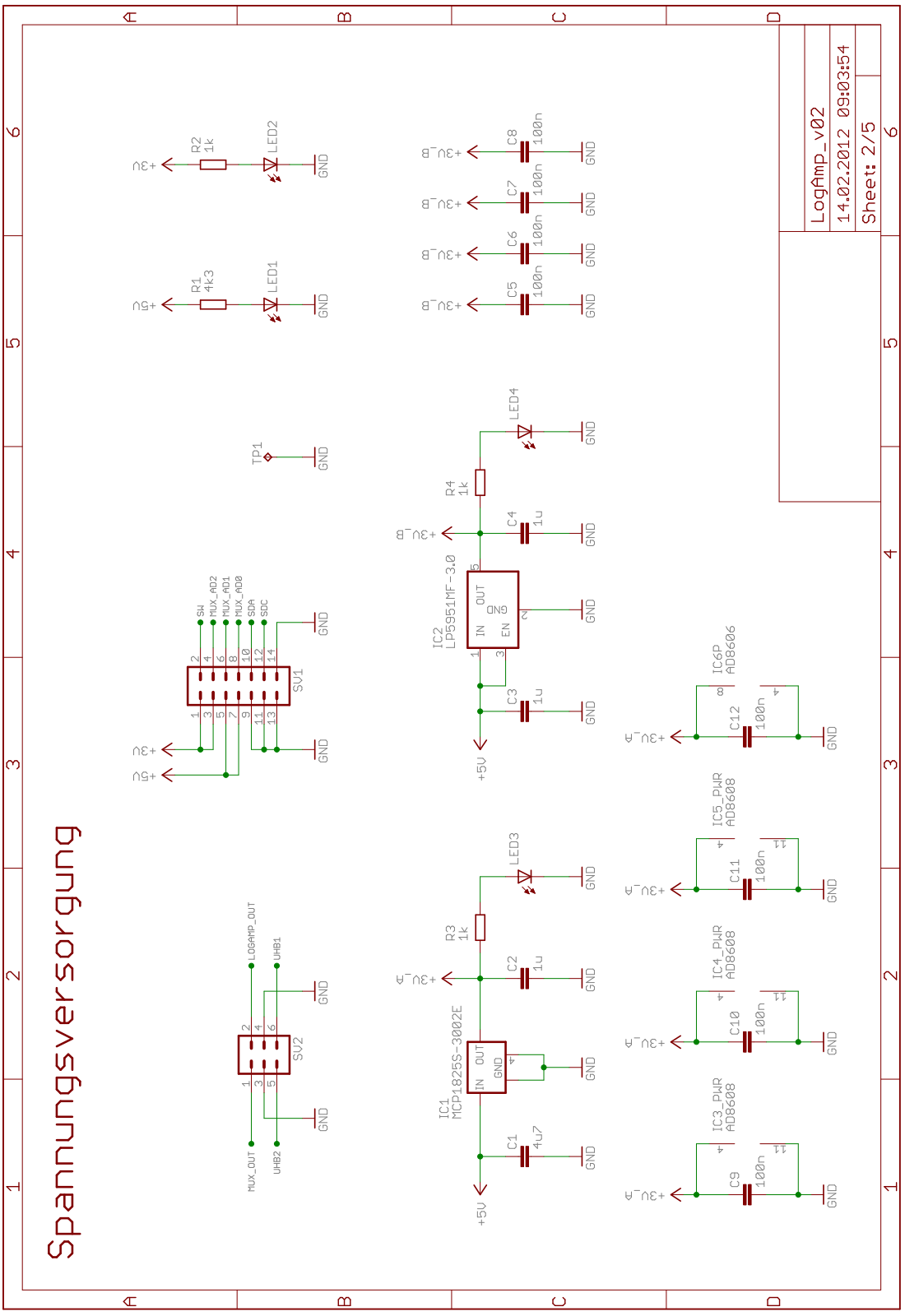
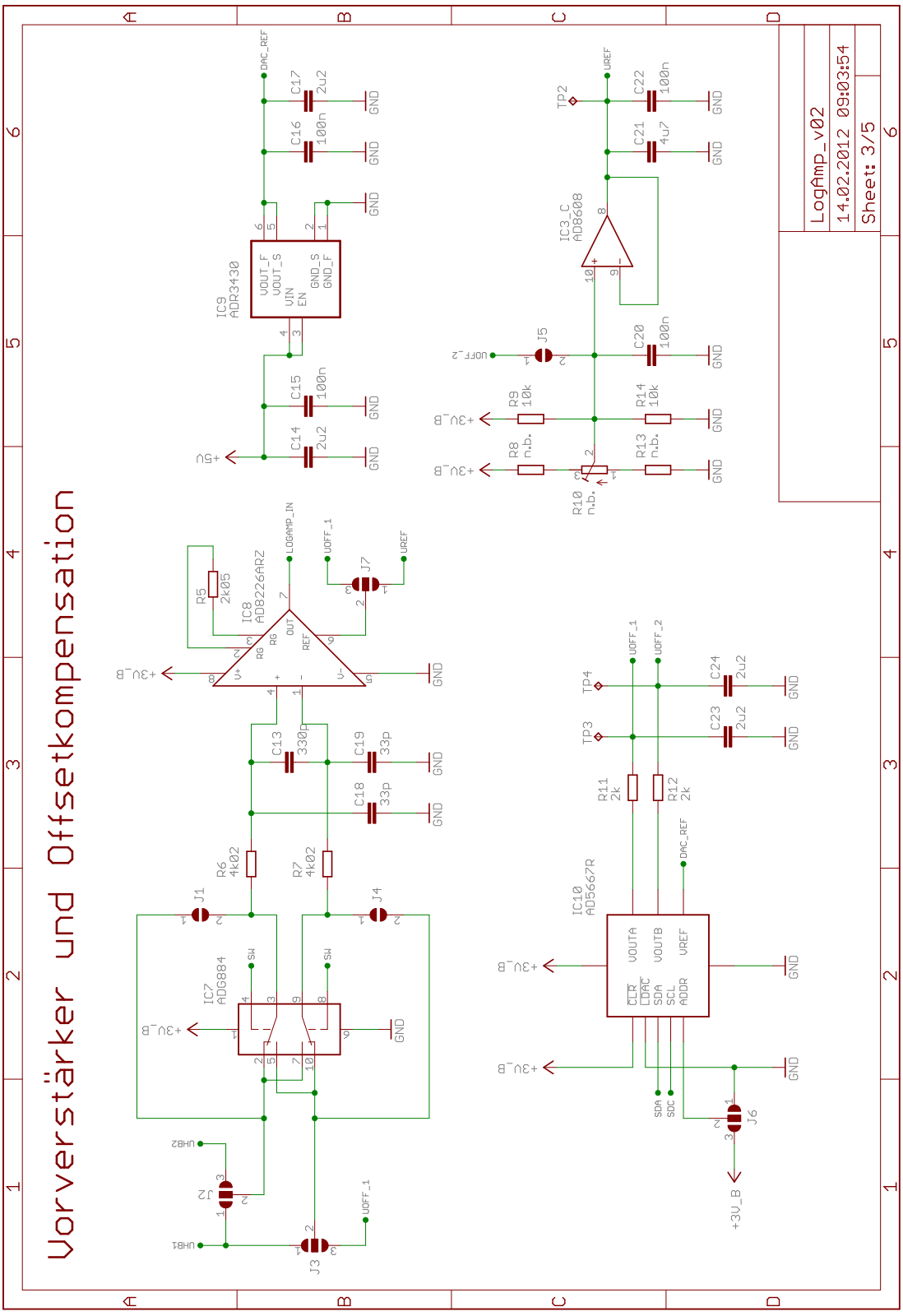


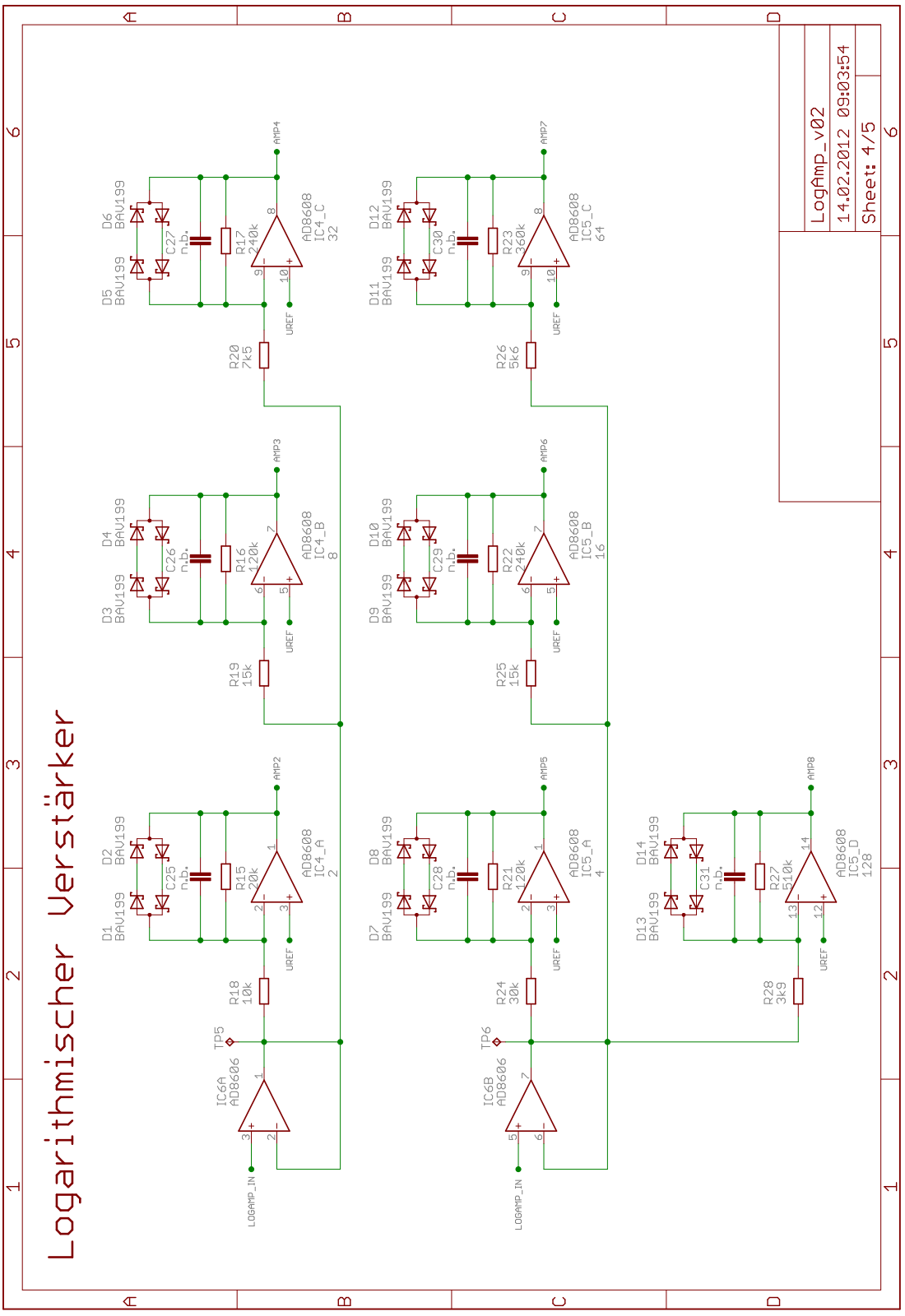

Abbildung C.2: Bestückung Unterseite Hauptplatine

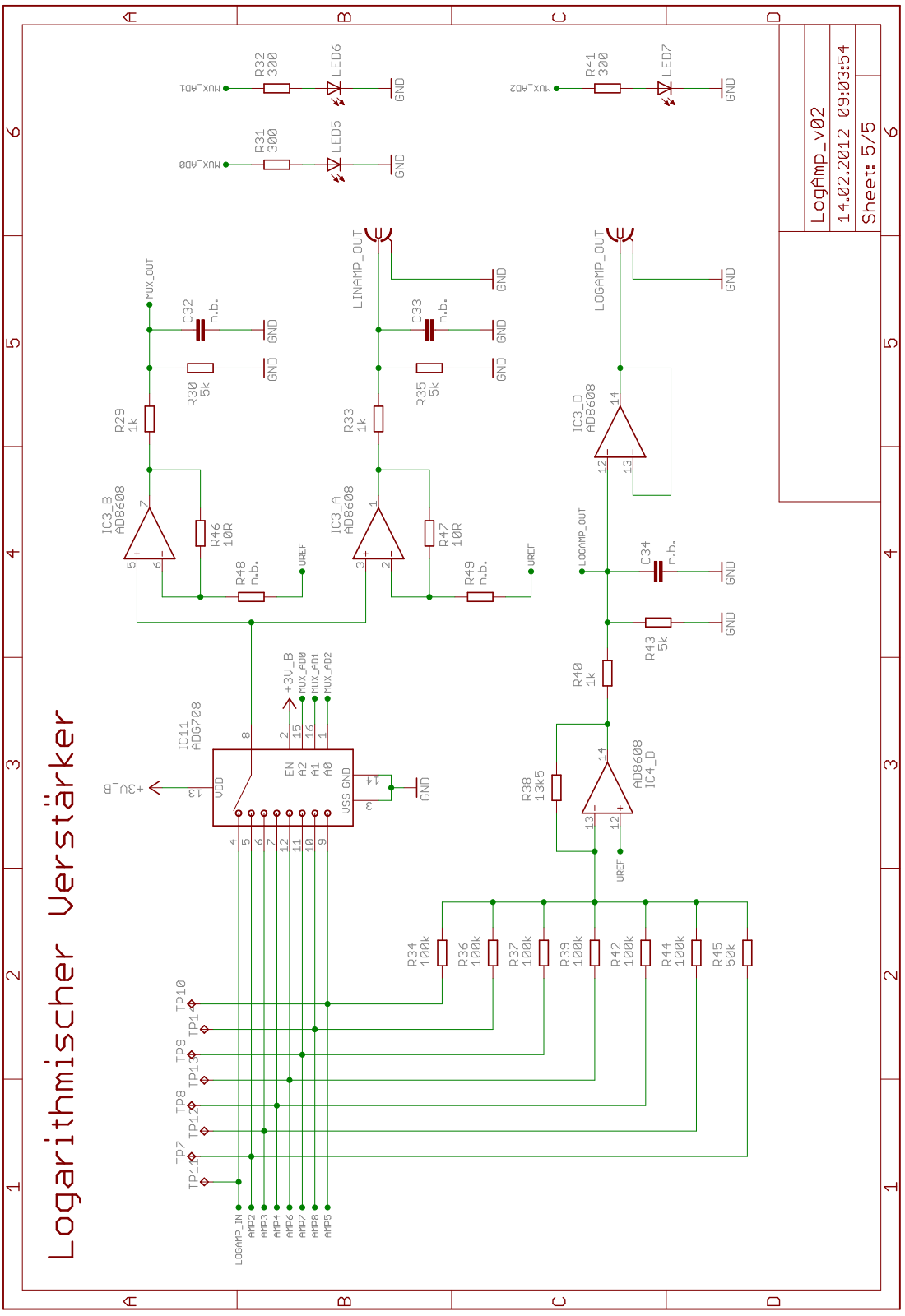
## C.3 Parallelschaltung


Logarithmischer Verstärker
Seite
Spannungsversorgung 2
Vorverstärker und Offsetkompensation 3
Logarithmischer Verstärker 4-5
LogAmp_v02
14.02.2012 09:03:54
Sheet: 1/5
LogAmp_Out
LinAmp_Out
MUX
DAC
Uhb2
Uhb1
SW
I2C
DI


Spannungsversorgung
LogAmp_v02
14.02.2012 09:03:54
Sheet: 2/5
R2 1k
LED2
R1 4k3
LED1
TP1
C8 100n
C7 100n
C6 100n
C5 100n
LED4
R4 1k
C4 1u
IC2 LP5951MF-3.0
C3 1u
SV1
SW
MUX_AD2
MUX_AD1
MUX_AD0
SDA
SCL
LED3
R3 1k
C2 1u
IC1 MCP1825S-3002E
C1 4u7
SV2
LOGAMP_OUT
UHB1
UHB2
MUX_OUT
IC6P AD8606
C12 100n
IC5_PWR AD8608
C11 100n
IC4_PWR AD8608
C10 100n
IC3_PWR AD8608
C9 100n
+3V3
+5V
GND


Vorverstärker und Offsetkompensation
LogAmp_v02
14.02.2012 09:03:54
Sheet: 3/5
IC9 ADR3430
IC8 AD8226ARZ
IC7 AD6884
IC3_C AD8608
IC10 AD5667R
TP2
TP3
TP4
C17 2u2
C16 100n
C15 100n
C14 2u2
C13 330p
C19 33p
C18 33p
C20 100n
C21 4u7
C22 100n
C23 2u2
C24 2u2
R5 2k05
R6 4k02
R7 4k02
R8 n.b.
R9 10k
R10 n.b.
R11 2k
R12 2k
R13 n.b.
R14 10k
J1
J2
J3
J4
J5
J6
J7
GND
+3V_B
+5V
DAC_REF
UREF
UOFF_1
UOFF_2
LOGAMP_IN
SDA
SDC


Logarithmischer Verstärker
LogAmp_v02
14.02.2012 09:03:54
Sheet: 4/5
A
B
C
D
1
2
3
4
5
6
AMP1
AMP2
AMP3
AMP4
AMP5
AMP6
AMP7
AMP8
D1 BAV199
D2 BAV199
D3 BAV199
D4 BAV199
D5 BAV199
D6 BAV199
D7 BAV199
D8 BAV199
D9 BAV199
D10 BAV199
D11 BAV199
D12 BAV199
D13 BAV199
D14 BAV199
C25 n.b.
C26 n.b.
C27 n.b.
C28 n.b.
C29 n.b.
C30 n.b.
C31 n.b.
R15 20k
R16 120k
R17 240k
R18 10k
R19 15k
R20 7k5
R21 120k
R22 240k
R23 360k
R24 30k
R25 15k
R26 5k6
R27 510k
R28 3k9
AD8608 IC4_A 2
AD8608 IC4_B 8
AD8608 IC4_C 32
AD8608 IC5_A 4
AD8608 IC5_B 16
AD8608 IC5_C 64
AD8608 IC5_D 128
IC6A AD8606
IC6B AD8606
TP5
TP6
UREF
LOGAMP_IN


Logarithmischer Verstärker
LogAmp_v02
14.02.2012 09:03:54
Sheet: 5/5
MUX_A01
MUX_A00
MUX_A02
R32 300
R31 300
R41 300
LED6
LED5
LED7
GND
MUX_OUT
LINAMP_OUT
LOGAMP_OUT
C32 n.b.
C33 n.b.
C34 n.b.
R30 5k
R35 5k
R43 5k
R29 1k
R33 1k
R40 1k
IC3_B AD8608
IC3_A AD8608
IC3_D AD8608
R46 10R
R47 10R
R48 n.b.
R49 n.b.
UREF
IC11 ADG708
+3V_B
VDD
VSS
GND
EN
A2
A1
A0
R38 13k5
IC4_D AD8608
R34 100k
R36 100k
R37 100k
R39 100k
R42 100k
R44 100k
R45 50k
TP10
TP14
TP9
TP13
TP8
TP12
TP7
TP11
LOGAMP_IN
AMP2
AMP3
AMP4
AMP6
AMP7
AMP8
AMP5

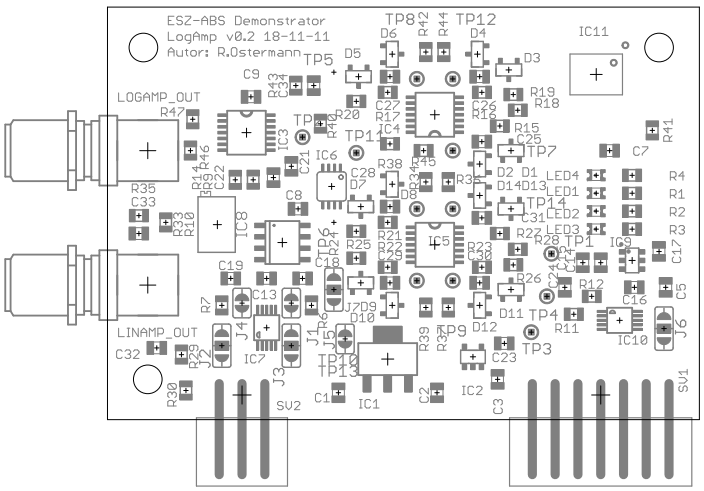



Abbildung C.3: Bestückung Oberseite logarithmisch und lineare Verstärkerbank, Parallelschaltung

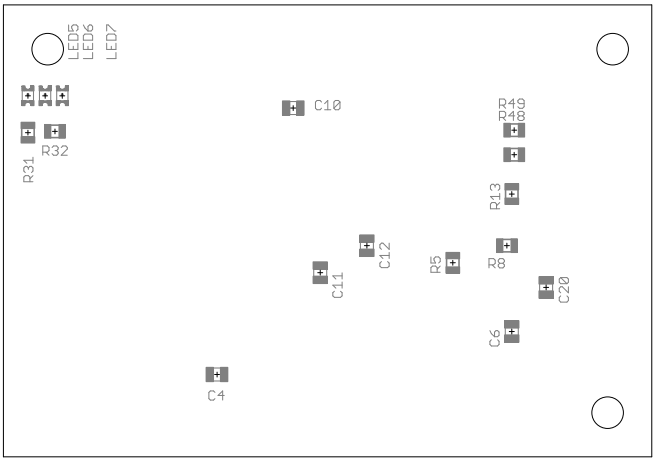
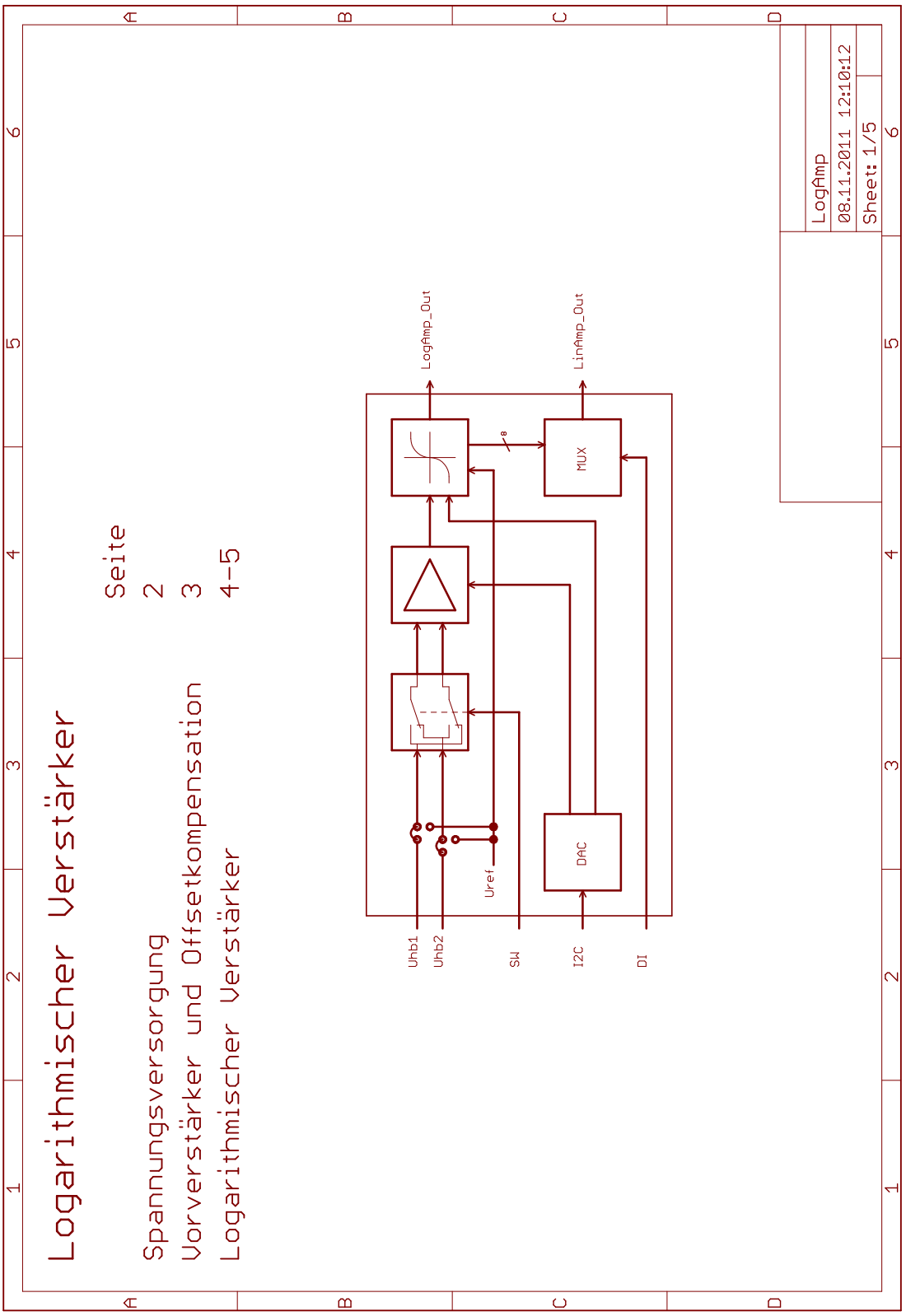
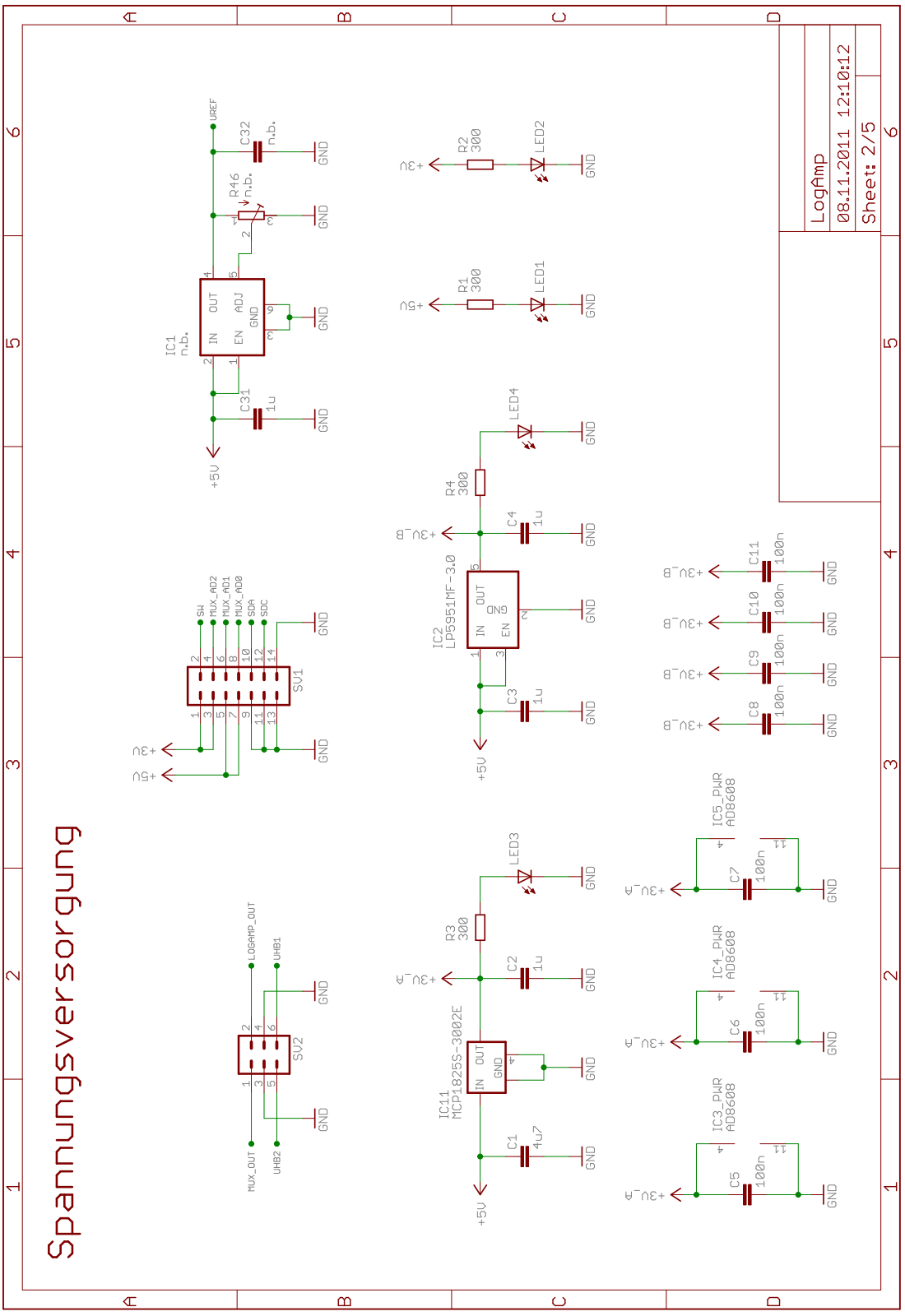
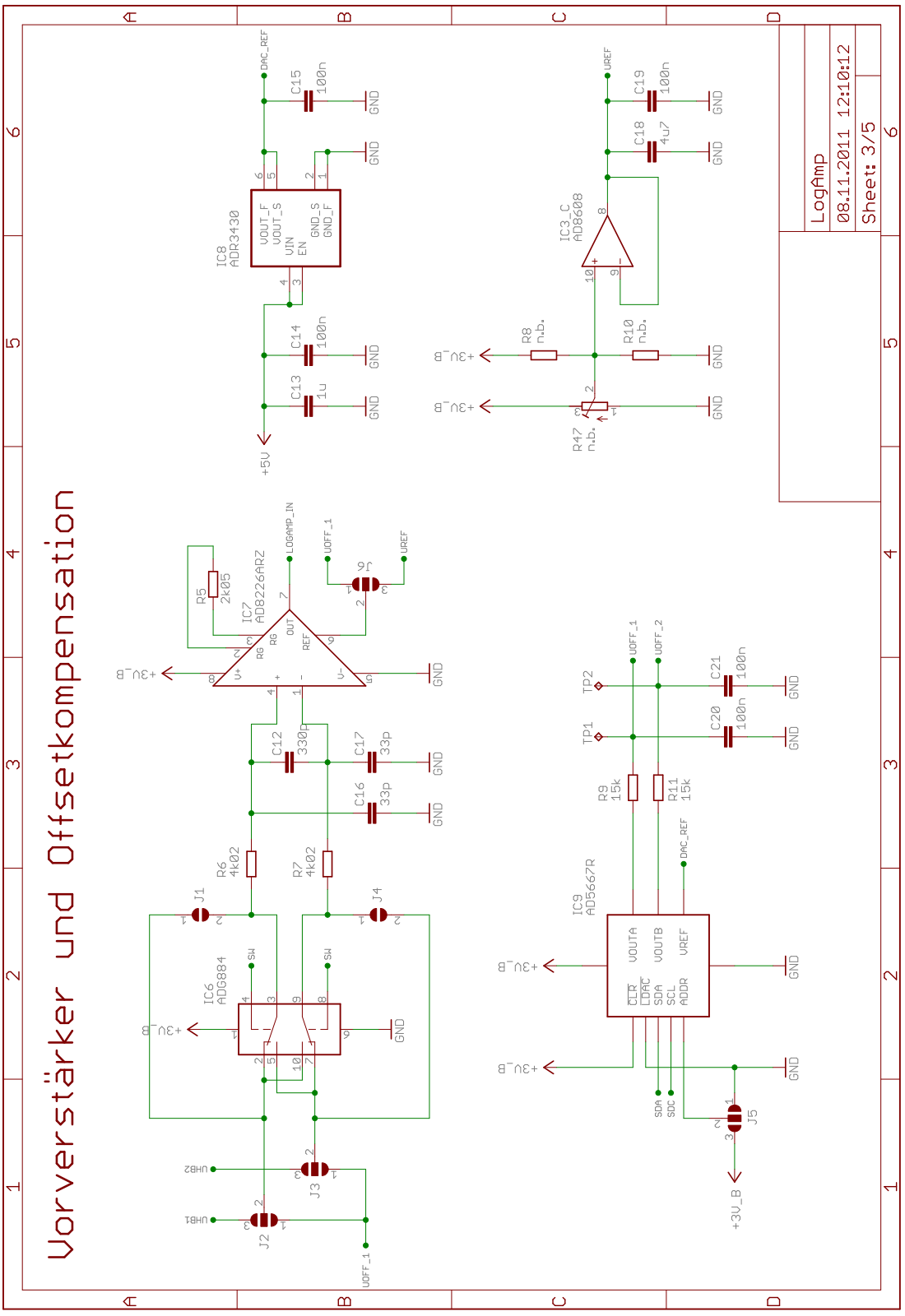


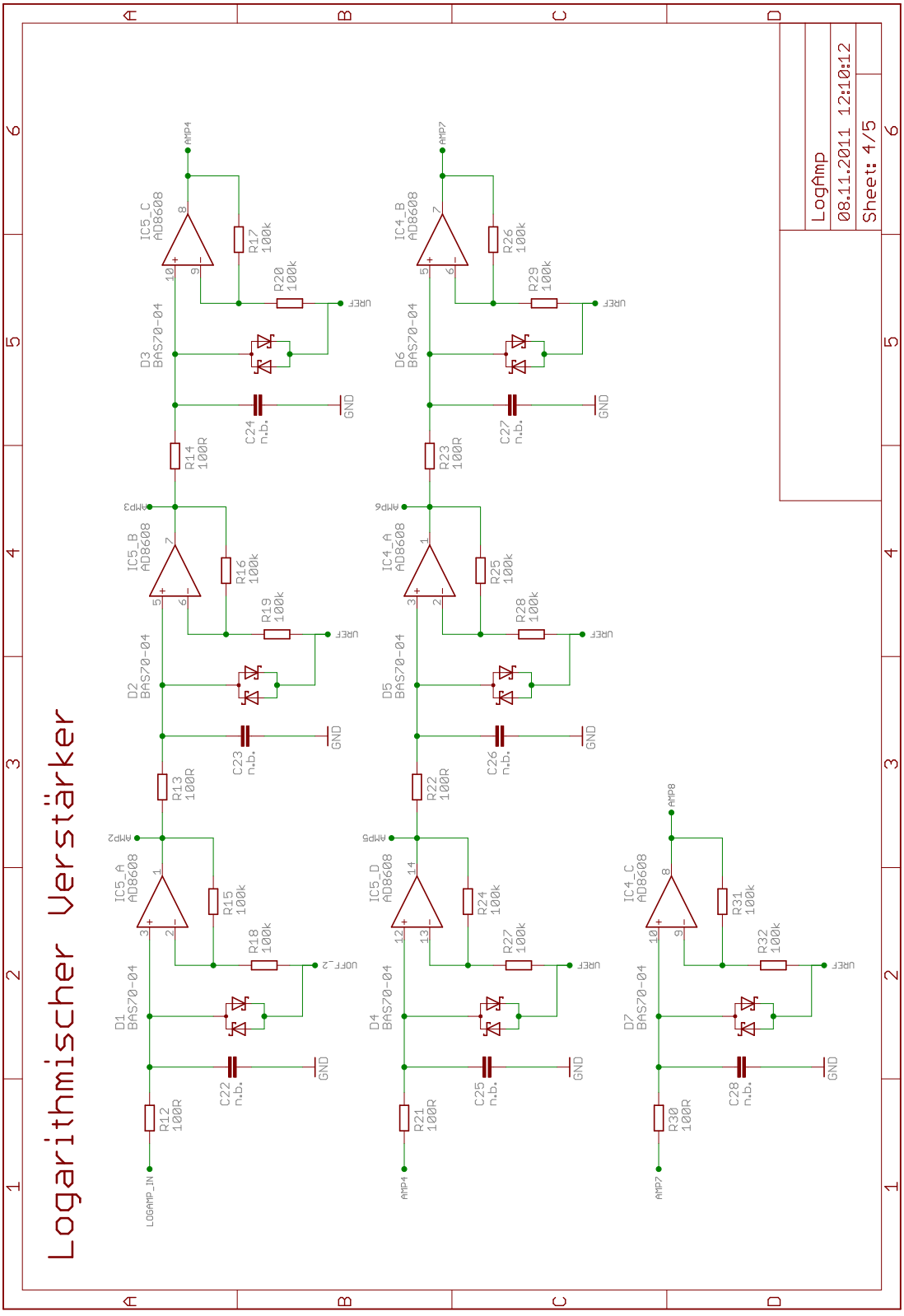

Abbildung C.4: Bestückung Unterseite logarithmisch und lineare Verstärkerbank, Parallelschaltung

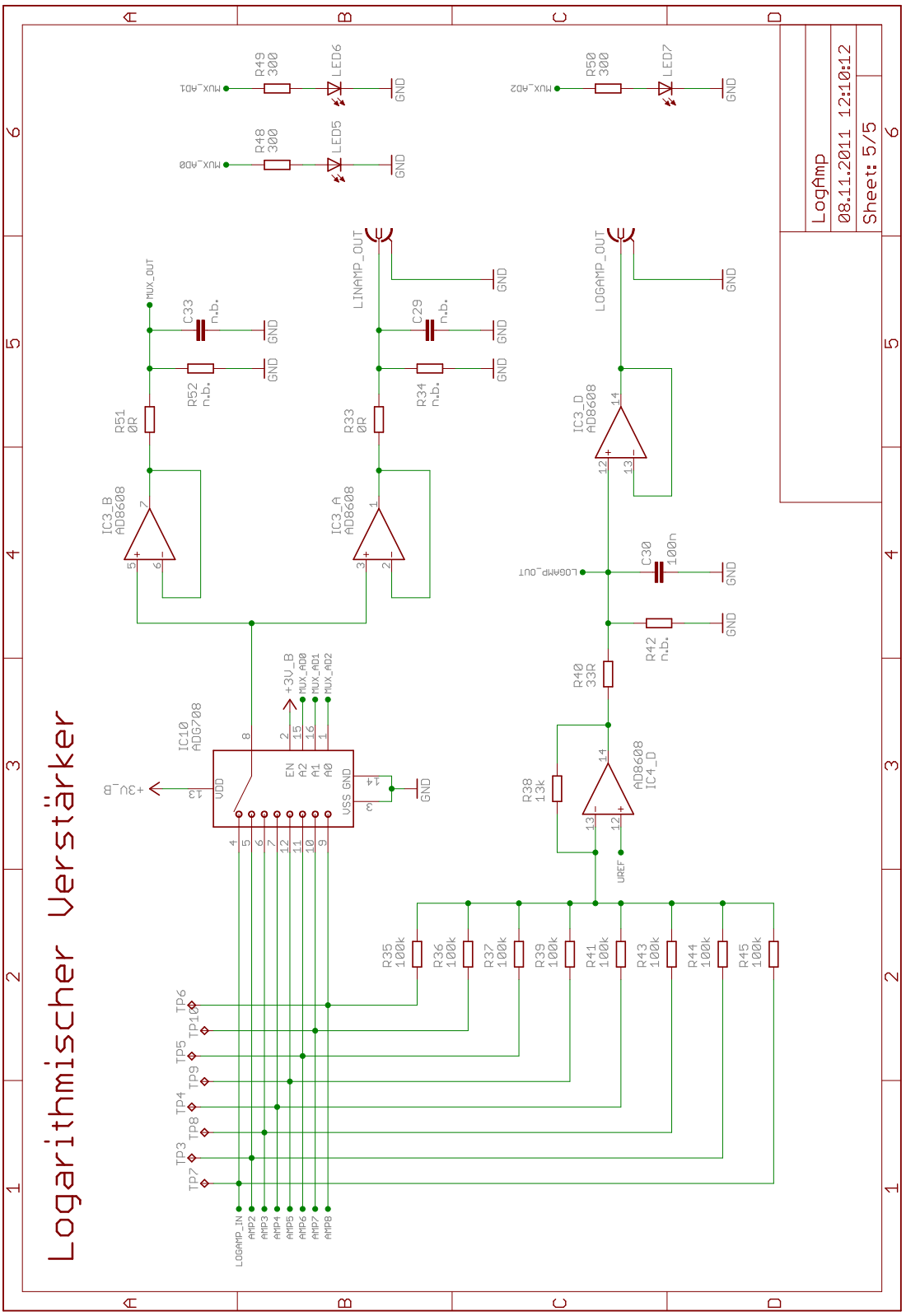
## C.4 Reihenschaltung


Logarithmischer Verstärker
Seite
Spannungsversorgung 2
Vorverstärker und Offsetkompensation 3
Logarithmischer Verstärker 4-5
LogAmp_Out
LinAmp_Out
MUX
8
DAC
Uref
Uhb1
Uhb2
SW
I2C
DI
LogAmp
08.11.2011 12:10:12
Sheet: 1/5


Spannungsversorgung
LogAmp
08.11.2011 12:10:12
Sheet: 2/5
UREF
C32
n.b.
R46
n.b.
IC1
n.b.
IN OUT ADJ
EN GND
C31
1u
+5V
GND
R2
300
LED2
R1
300
LED1
LED4
R4
300
C4
1u
IC2
LP5951MF-3.0
IN OUT
GND
EN
C3
1u
LED3
R3
300
C2
1u
IC11
MCP1825S-3002E
IN OUT
GND
C1
4u7
SV1
SW
MUX_AD2
MUX_AD1
MUX_AD0
SDA
SDC
SV2
LOGAMP_OUT
UHB1
MUX_OUT
UHB2
IC5_PWR
AD8608
IC4_PWR
AD8608
IC3_PWR
AD8608
C11
C10
C9
C8
C7
C6
C5
100n
+3V_A
+3V_B
+5V


Vorverstärker und Offsetkompensation
LogAmp
08.11.2011 12:10:12
Sheet: 3/5
IC8 ADR3430
IC7 AD8226ARZ
IC6 AD6884
IC3_C AD8608
IC9 AD5667R
R5 2k05
R6 4k02
R7 4k02
R8 n.b.
R9 15k
R10 n.b.
R11 15k
R47 n.b.
C12 330p
C13 1u
C14 100n
C15 100n
C16 33p
C17 33p
C18 4u7
C19 100n
C20 100n
C21 100n
J1
J2
J3
J4
J5
TP1
TP2
GND
+3V_B
+5V
UOFF_1
UOFF_2
UREF
DAC_REF
LOGAMP_IN
SDA
SDC


Logarithmischer Verstärker
LogAmp
08.11.2011 12:10:12
Sheet: 4/5
LOGAMP_IN
AMP1
AMP2
AMP3
AMP4
AMP5
AMP6
AMP7
AMP8
UREF
GND
IC5_A AD8608
IC5_B AD8608
IC5_C AD8608
IC5_D AD8608
IC4_A AD8608
IC4_B AD8608
IC4_C AD8608
D1 BAS70-04
D2 BAS70-04
D3 BAS70-04
D4 BAS70-04
D5 BAS70-04
D6 BAS70-04
D7 BAS70-04
R12 100R
R13 100R
R14 100R
R15 100k
R16 100k
R17 100k
R18 100k
R19 100k
R20 100k
R21 100R
R22 100R
R23 100R
R24 100k
R25 100k
R26 100k
R27 100k
R28 100k
R29 100k
R30 100R
R31 100k
R32 100k
C22 n.b.
C23 n.b.
C24 n.b.
C25 n.b.
C26 n.b.
C27 n.b.
C28 n.b.


Logarithmischer Verstärker
LogAmp
08.11.2011 12:10:12
Sheet: 5/5
LINAMP_OUT
LOGAMP_OUT
MUX_OUT
IC3_B AD8608
IC3_A AD8608
IC3_D AD8608
AD8608 IC4_D
IC10 ADG708
R51 0R
R33 0R
R52 n.b.
R34 n.b.
R40 33R
R38 13k
R42 n.b.
C33 n.b.
C29 n.b.
C30 100n
R49 300
R48 300
R50 300
LED6
LED5
LED7
MUX_A01
MUX_A00
MUX_A02
GND
+3U_B
VREF
R35 100k
R36 100k
R37 100k
R39 100k
R41 100k
R43 100k
R44 100k
R45 100k
TP6
TP10
TP5
TP9
TP4
TP8
TP3
TP7
LOGAMP_IN
AMP2
AMP3
AMP4
AMP5
AMP6
AMP7
AMP8

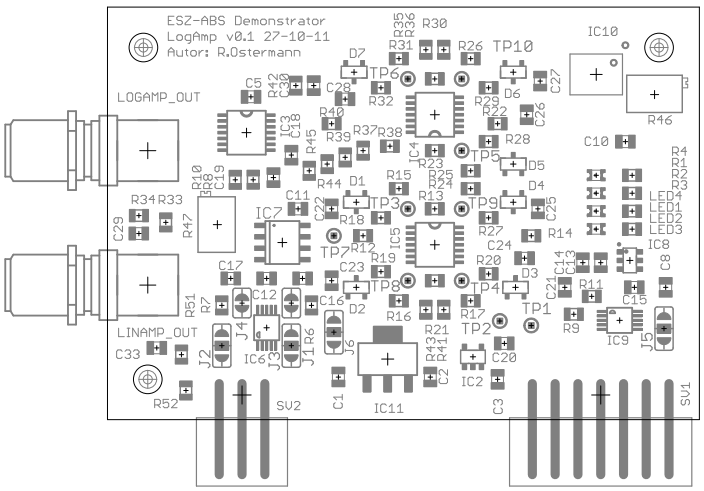



Abbildung C.5: Bestückung Oberseite logarithmisch und lineare Verstärkerbank, Reihenschaltung

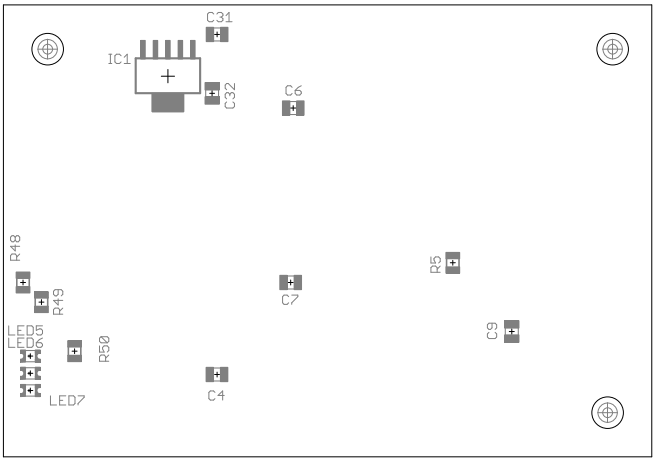



Abbildung C.6: Bestückung Unterseite logarithmisch und lineare Verstärkerbank, Reihenschaltung

# D Quellcode

## D.1 Regelcontroller

In der Tabelle D.1 sind die Quelldateien mit den enthaltenden Funktionen aufgelistet.

| Datei | Funktion | Seite |
|---|---|---|
| main.c | main() | 108 |
| main.h | - | 110 |
| global.h | - | 112 |
| init.c | init() | 116 |
| calc_other.c | check_gain()<br>calc_offset() | 117 |
| cleanup.c | cleanup() | 118 |
| isr_adc12.c | isr_adc12() | 119 |
| isr_comp_a.c | isr_comp_a() | 124 |
| isr_timer_a1.c | isr_timer_a1() | 126 |
| send.c | set_compensation_state()<br>send_usart() | 127 |
| set_gain.c | set_gain() | 129 |
| i2c.h | - | 130 |
| i2c.c | i2c_init()<br>i2c_send()<br>i2c_isr() | 133 |
| ad5667.h | - | 134 |
| ad5667.c | ad5667_init()<br>ad5667_set() | 135 |
| delog_table.h | - | 142 |
| offset_sin.h | - | 143 |

Tabelle D.1: Zuordnung von Quelldateien und Funktionen des Regelcontrollers

```
1  /*****************************************************************************/
2  /*  Autor:          Martin Stahl                                             */
3  /*  Datum:          16.12.2009                                               */
4  /*  Hardware:       dm_off_amp_ctrl v0.1                                     */
5  /*  Controller:     MSP430f1611 => "Regelcontroller"                         */
6  /*  Toolchain:      MSPGCC v.20060502, Eclipse ganymede-SR2, USBExpress v1.021*/
7  /*                                                                           */
8  /*  Datei:          main.c                                                   */
9  /*  Beschreibung:   Aufruf der Initialisierung des Systems                   */
10 /*                  In der Endlosschleife erfolgt lediglich die Ausgabe von Daten*/
11 /*                  auf der Display-Platine => Niedrigste Priorität          */
12 /*                  Alle übrigen Funktionen werden interruptgesteuert ausgeführt*/
13 /*                                                                           */
14 /*  Geändert von:   Robert Ostermann                                         */
15 /*  Datum:          01.09.2011 - 24.02.2012                                  */
16 /*  Änderungen:     Anpassung an kombinierte logarithmisch und lineare Verstärkerbank*/
17 /*                  Ausgabe auf der Display-Platine entfernt, da Ports für   */
18 /*                  Multiplexer verwendet werden                             */
19 /*                                                                           */
20 /*  Funktion:       ok                                                       */
21 /*                                                                           */
22 /*****************************************************************************/
23
24 #define EXTERN
25
26 #include "main.h"
27 #include "global.h"
28 #include "i2c.h"
29
30 /*************************************************************/
31 /*                        Hauptschleife                      */
32 /*************************************************************/
33 int main(void)
34 {
35     init(); // Initialisierung des Systems
36
37     while(1)
38     {
39
40     }// Ende while(TRUE)
41 } // Ende main()
```

```
1  /*******************************************************************************/
2  /*  Autor:          Martin Stahl                                               */
3  /*  Datum:          16.12.2009                                                 */
4  /*  Hardware:       dm_off_amp_ctrl v0.1                                       */
5  /*  Controller:     MSP430f1611 => "Regelcontroller"                           */
6  /*  Toolchain:      MSPGCC v.20060502, Eclipse ganymede-SR2, USBExpress v1.021  */
7  /*                                                                             */
8  /*  Datei:          main.h                                                     */
9  /*  Beschreibung:   Header-File - enthält alle Definitionen und Prototypen     */
10 /*                                                                             */
11 /*  Geändert von:   Robert Ostermann                                           */
12 /*  Datum:          01.09.2011 - 24.02.2012                                    */
13 /*  Änderungen:     Anpassung an kombinierte logarithmisch und lineare Verstärkerbank   */
14 /*                  Offset Anfangswert für externe DAC hinzugefügt             */
15 /*                  define für DVGA und Displayplatine entfernt                */
16 /*                                                                             */
17 /*  Funktion:       ok                                                         */
18 /*                                                                             */
19 /*******************************************************************************/
20
21 #ifndef MAIN_H_
22 #define MAIN_H_
23
24     #include <msp430x16x.h>
25     #include <signal.h>
26
27
28 /******* Allgemeine Definitionen ******/
29     #define FALSE   0
30     #define TRUE    !FALSE
31
32
33 /******* Spezielle Definitionen *******/
34
35     #define __OFFSET_CAL            // definieren wenn Offset Kompensation
36     #define __INIT_FALL_BACK        // Rücksprung von Active --> Init_0Hz
37
38 /******* Makros für Offsetkompensation  *******/
39
40                     // Regelung im Active-Mode wird zugelassen
41     #define __ACTIVE_COMPENSATION
42                 // 2^x Perioden fuer die Mittelwertbildung des Duty-Cycles
43     #define PERIODES_FOR_MEAN_DUTY  5 //7
44                 //Toleranzfenster für die Duty-Cycle-Korrektur
45     #define DUTY_TOLERANCE  4
46                 //Periodenanzahl abhängig vom Verstärkungsfaktor
47     //#define __MEAN_DUTY_DEPEND_ON_GAIN
48         // Anzahl an durchzufuehrenden Offset-Kompensationen im Zustand Initial_1Hz
49     #define COMPENSATIONS   6
50
51     #define GAIN_START      7   // Gain in Exponent von 2
52                 //Schwellwert-Differenz zum Offset => Mindestverstärkrung (300mV)
53     #define LEVEL_GMIN      600//492
54                 //Schwellwert-Differenz zum Offset => maximale Verstärkung (940mV)
55     #define LEVEL_GMAX      700//1542//+200
56                 //Anzahl der Samples, die die Schwellwerte überschreiten müssen
57     #define MIN_SAMPLES_PASSED 2
58
59 /********* Timer-Definitionen *********/
60
61     #define TIMER_A_PER     13488       // Periodendauer Timer A in ns
62                                         //  = T_SMCLK * 8 in ns
63     #define TCCR_TEST       30000       // --> T_SMCLK = 1686ns (div1)
64
65     #define SAMPLETIME      400         // 4,9MHz/800=12,250kHz
66     #define TIMER_B_PER_F   169247      // Periodendauer Timer B in ns * 10
67     #define TIMER_B_PER     1693        // Periodendauer Timer B in ns
68                                         // Festlegung des Prescalers Timer B7:
69 /********** DAC-Definitionen **********/
70
71     #define OFFSET_START        2048    // Offset, Anfangswert (1.25V) 2048 RO
72     #define LEVEL_SX            80      // Schwellwert-Differenz zum Offset(0.1V) 164
73     #define OFFSET_EXT_START    32768   // Offset, Anfangswert (1.5V)
74
75     #define MIN_TIME_S1     4           // Minimale Zeit am Punkt S1
76     #define MIN_TIME_S2     2           // Minimale Zeit am Punkt S2
77
78 /********** Pin-Definitionen **********/
79     #define PIN0        0x01
```

```
80      #define PIN1        0x02
81      #define PIN2        0x04
82      #define PIN3        0x08
83      #define PIN4        0x10
84      #define PIN5        0x20
85      #define PIN6        0x40
86      #define PIN7        0x80
87
88      #define PORT_LED    P1OUT
89      #define LED1        PIN4
90      #define LED2        PIN5
91      #define LED3        PIN6
92      #define LED4        PIN7
93
94      #define COMM_4      PIN4
95      #define COMM_5      PIN5
96      #define COMM_6      PIN5
97      #define COMM_7      PIN6
98      #define COMM_8      PIN7
99
100 /******** Zustandsdefinitionen ********/
101
102     typedef enum {  // Systemzustände Sensor
103         Initial_0Hz,
104         Initial_1Hz,
105         Active
106     } states;
107
108     typedef enum {  // Systemzustände Periodenerkennung
109         Search_S1,
110         Search_S1_Calc,
111         Search_S2,
112         Search_S2_Pre,
113         Search_S1_Check_Gain,
114         Search_S2_Check_Gain
115     } states_per;
116
117 /******* Funktions-Prototypen ********/
118
119     void init(void);
120     void set_7seg(uint16_t frequenz, uint8_t thd, uint8_t status);
121     uint8_t send_usart(uint8_t value);
122     void set_gain(void);
123     void cleanup(void);
124     void calc_offset(void);
125     void check_gain(void);
126     void set_compensation_state(uint8_t state);
127     uint16_t calc_crc16(uint16_t crc, uint8_t a);
128     void send_usart_data();
129
130 #endif
```

```
1  /*****************************************************************************************/
2  /*  Autor:          Martin Stahl                                                         */
3  /*  Datum:          22.12.2009                                                           */
4  /*  Hardware:       dm_off_amp_ctrl v0.1                                                 */
5  /*  Controller:     MSP430f1611 => "Regelcontroller"                                     */
6  /*  Toolchain:      MSPGCC v.20060502, Eclipse ganymede-SR2, USBExpress v1.021           */
7  /*                                                                                       */
8  /*  Datei:          global.h                                                             */
9  /*  Beschreibung:   Header-File - enthält globale Variablen                              */
10 /*                  Namensgebung basiert auf Diplomarbeit NJ                             */
11 /*                                                                                       */
12 /*  Geändert von:   Robert Ostermann                                                     */
13 /*  Datum:          01.09.2011 - 24.02.2012                                              */
14 /*  Änderungen:     Anpassung an kombinierte logarithmisch und lineare Verstärkerbank    */
15 /*                  Variablen für Anzahl überschreitungen der Schwellen entfernt         */
16 /*                  Variablen für Amplituden eingefügt                                   */
17 /*                                                                                       */
18 /*  Funktion:       ok                                                                   */
19 /*                                                                                       */
20 /*****************************************************************************************/
21
22 #ifndef GLOBAL_H_
23 #define GLOBAL_H_
24
25     #ifndef EXTERN
26     #define EXTERN extern
27     #endif
28
29     struct {
30         uint32_t duty;              // Tastverhältnis
31         uint32_t offset;            // Ausgangswert für DAC12_1
32         int16_t  offset_faktor;     // Faktor für Offset-Änderung
33         uint8_t  compensations;     // Kompensationen der Offset Kompensation
34         uint8_t  gain;              // Verstärung des 2.TIA
35         uint8_t  lgain;
36         uint32_t duty_sum;          // Summe von x Duty-Cycles zut Mittelwertbildung
37         uint16_t  duty_count;       // Zähler für aufsummierte Duty_Cycles
38     } EXTERN volatile g_sig;
39
40     struct {                        // ISR Comperator:
41         uint16_t tp;                // halbe Periode
42         uint16_t tt;                // Periodendauer
43         uint16_t frequenz;          // Frequenz in Hz
44         uint16_t sample_inst;       // Zeitpunkt zum Samplen
45     } EXTERN volatile g_time;
46
47     struct {                        // ADC:
48         uint16_t u_max_offset;      // Maximum für Offset
49         uint16_t u_min_offset;      // Minimum für Offset
50         uint16_t u_max_gain;        // Maximum für Verstärkung
51         uint16_t u_min_gain;        // Minimum für Verstärkung
52         uint32_t u_max_middle;      // Maximum für Offset Mittelwert
53         uint32_t u_min_middle;      // Minimum für Offset Mittelwert
54         uint16_t min_cnt;
55         uint16_t max_cnt;
56         uint16_t u_diff;            // Differenz Signal
57         int16_t  u_diff_b;          // Differenz Signal ohne Offset
58         uint16_t u_hb1;             // Halbbrücken Signal 1
59         uint16_t u_hb2;             // Halbbrücken Signal 2
60         uint16_t u_sum1;            // Summe HB1 + HB2
61     } EXTERN volatile g_adc;
62
63     struct {                                // ISR USART1:
64         uint8_t  send_value;                // Daten die gesendet werden
65         uint32_t data_value;                // Daten aus Variable
66         uint16_t  data_length;              // Anzahl an Elementen der jeweiligen Daten
67         uint16_t factor;                    // Universeller Faktor
68         uint8_t  data_to_send;              // Index für zu sendene Variable
69         uint8_t  select;                    // Selection eines Worts einer Variable
70         uint16_t  i;                        // Index für Daten der Variable
71         uint8_t  frame_pos;                 // Position in der Übertragung
72         uint8_t  send_all;                  // Flag
73         uint8_t  send_part;
74         uint16_t crc;                       // current crc value
75     } EXTERN volatile g_usart;
76
77 /************** Zustaende des Systems  **************/
78     EXTERN volatile states      sensor_state;
79     EXTERN volatile states_per  state_periode;
```

```
80 |
81 |     struct{
82 |         unsigned int middle;
83 |         unsigned int cur;
84 |         unsigned long sum;
85 |         unsigned int cnt;
86 |     } EXTERN volatile ad;
87 |
88 | #endif
```

```
1  /*****************************************************************************/
2  /*  Autor:          Martin Stahl                                             */
3  /*  Datum:          16.12.2009                                               */
4  /*  Hardware:       dm_off_amp_ctrl v0.1                                     */
5  /*  Controller:     MSP430f1611 => "Regelcontroller"                         */
6  /*  Toolchain:      MSPGCC v.20060502, Eclipse ganymede-SR2, USBExpress v1.021*/
7  /*                                                                           */
8  /*  Datei:          init.c                                                   */
9  /*  Beschreibung:   Initialisierung der Controller-Register und -Peripherie  */
10 /*                  Verwendet sind interner DCO, USART1, TimerA, TimerB, ComparartorA,  */
11 /*                  ADC12, DAC12_0, DAC12_1, div. I/O-Pins                   */
12 /*                                                                           */
13 /*  Geändert von:   Robert Ostermann                                         */
14 /*  Datum:          01.09.2011 - 24.02.2012                                  */
15 /*  Änderungen:     Anpassung an kombinierte logarithmisch und lineare Verstärkerbank   */
16 /*                  Initialisierung der Portpins für Verstärkerbank angepasst*/
17 /*                  i2c initialisierung und initialisierung der externer DAC einfügt    */
18 /*                  ADC aktivierung eingefügt                                */
19 /*                                                                           */
20 /*  Funktion:       ok                                                       */
21 /*                                                                           */
22 /*****************************************************************************/
23
24 #include "main.h"
25 #include "global.h"
26 #include "i2c.h"        // I2C-Schnittstelle konfigurieren
27 #include "ad5667.h"     // DAC-Funktionen bereitstellen
28
29
30 /*************************************************************/
31 /*              Initialisierungsroutine                      */
32 /*************************************************************/
33
34 void init(void)
35 {
36 /************ Initialisierung interner Oszillator DCO = 4.9 MHz **************/
37
38     // DCO  = 7, Mod ignored
39     DCOCTL   = DCO2 | DCO1 | DCO0;
40     // RSEL = 7, XT2 Oscillator off, ACLK Div = 1
41     BCSCTL1  = RSEL2 | RSEL1 | RSEL0 | XT2OFF;
42     // MCLK von DCOCLK, SMCLK von DCOCLK, SMCLK Div = 8
43     //BCSCTL2 |= (DIVS0 | DIVS1);
44
45 /********************* Initialisierung USART1 ***********************/
46     //zum debuggen
47
48         // USART1 Control Reg.: Parität = even, StopBits = 2, CharLength = 8, UART Mode
49     U1CTL  = PENA | PEV | SPB | CHAR;
50         // USART1 Transmit Control Reg.:    Clock = SMCLK, Transmit Empty Flag = 1
51     U1TCTL = SSEL_SMCLK | TXEPT;
52         // USART1 Receive Control Reg.:
53     U1RCTL = 0x00;
54         // USART1 Baud Rate Control Reg.0:  (U1BR1+U1BR0) = U1BR = SMCLK / BaudRate
55         //                                  BaudRate = 115200 --> N = 5.1485, UBR = 5
56     U1BR0  = 0x05;
57         // USART1 Baud Rate Control Reg.1:
58     U1BR1  = 0x00;
59         // USART1 Modulation Control Reg.:  1/n * UMOD = 1/12 * 2 = 0.166 --> N = 5.166
60     U1MCTL = 0x90;
61         // Module Enable Reg.2:     Transmitter disable, Receiver disable
62     ME2   &= ~(UTXE1 | URXE1);
63         // Interrupt Flag Reg.2:    Flags löschen
64     IFG2  &= ~(UTXIFG1 | URXIFG1);
65         // Interrupt Enable Reg.2:  Transmit IRQ disable
66     IE2   &= ~UTXIE1;
67
68 /********************* Initialisierung Timer A ***********************/
69     // Zur Stillstandserkennung und Bestimmung des Tastverhältnisses
70
71     // Timer Control Reg.: SMCLK, prescaler=8, clear TAR, IRQ en
72     TACTL   = TACLR;
73     TACTL   = TASSEL1 | ID1 | ID0 | TAIE;
74     TAR     = 0x0000;
75
76     // Timer Capture/Compare Control Reg.0:
77     TACCTL0 = 0x0000;                       // IRQ aus
78     TACCR0  = 0x0000;
79     TACCTL1 = 0x0000;
```

```
80  |     TACCR1  = 0x0000;
81  |     TACCTL2 = 0x0000;
82  |     TACCR2  = 0x0000;
83  |
84  | /********************* Initialisierung Timer B ***********************/
85  |     //Triggert den ADC
86  |
87  |     // Timer Control Reg.: SMLK, prescaler=1, clear TAR, IRQ en
88  |     TBCTL   = TBCLR;
89  |     TBCTL   = TBSSEL1;
90  | //  TBCCTL0 |= CCIE;
91  |     TBR     = 0x0000;
92  |
93  |     // Timer Capture/Compare Control Reg.0:
94  |     TBCCTL0 |= OUTMOD_3;
95  |     TBCCR0  = SAMPLETIME-1;             // 0xFFFF - 1
96  |     TBCCTL1 = 0x0000;                   // Output Mode 1 wird erst später gesetzt
97  |     TBCCR1  = 0x0000;
98  |     TBCCTL2 = 0x0000;
99  |     TBCCR2  = 0x0000;
100 |     TBCCTL3 = 0x0000;
101 |     TBCCR3  = 0x0000;
102 |     TBCCTL4 = 0x0000;
103 |     TBCCR4  = 0x0000;
104 |     TBCCTL5 = 0x0000;
105 |     TBCCR5  = 0x0000;
106 |     TBCCTL6 = 0x0000;
107 |     TBCCR6  = 0x0000;
108 |
109 | /********************* Initialisierung Comparator ***********************/
110 |
111 |     // Comperator Control Reg.1: no exchange, Internal reference off, Comperator off,
112 |     //                           IRQ rising edge, IRQ disable, Flag delete,
113 |     //                           CA0 pos. - CA1 neg. Input
114 |     CACTL1 = 0x00;
115 |     // Comperator Control Reg.2: P2CA1 -> CA1, P2CA0 -> CA0, output filtered
116 |     CACTL2 = P2CA1 | P2CA0 | CAF;
117 |     // Comperator Port Disable Reg.: input buffer P2.2, P2.3, P2.4 disable
118 |     CAPD   = CAPD2 | CAPD3 | CAPD4;
119 |
120 | /************************ Initialisierung ADC ************************/
121 |
122 |     // ADC12 Control Reg.0: Sample & Hold Time = 32 , Reference = 2.5V, Reference on,
123 |     //                      ADC12 off, IRQ MEMx overflow,
124 |     //                      IRQ conversion time overflow,
125 |     ADC12CTL0 = SHT0_DIV8 | REF2_5V | REFON | ADC12OVIE | ADC12ON;  // | ADC12TOVIE
126 |     // ADC12 Control Reg.1: MCTL0 = first sample, Clock = MCLK, Sequence of channels
127 |     //                      Trigger by TBCCR0
128 |     ADC12CTL1 = CSTARTADD_1 | ADC12SSEL_MCLK | SHS_TBCCR0 | ADC12DIV_0 |
129 |                 SHP | CONSEQ_REPEAT_SINGLE;
130 |     // ADC12 Conversion Memory Control Reg.0: Reference = Vref+ / AVss, Channel = A0
131 |     //ADC12MCTL0 = SREF_VREF_AVSS | INCH_0;     // RO
132 |     // ADC12 Conversion Memory Control Reg.1: Reference = Vref+ / AVss, Channel = A1
133 |     ADC12MCTL1 = SREF_VREF_AVSS | INCH_1;       //RO
134 |     // ADC12 Conversion Memory Control Reg.1: Reference = Vref+ / AVss, Channel = A2
135 |     //                                        End of Sequence
136 |     //ADC12MCTL2 = SREF_VREF_AVSS | INCH_2 | EOS;
137 |     // ADC12 Interrupt Enable Reg: MCTL 2 enable IRQ
138 |     ADC12IE = 0x0002;   // 0x0001 RO
139 |
140 | /************************ Initialisierung interner DAC ************************/
141 |
142 |     // DAC12_0 Control Reg.: Internal reference, 12-Bit, load when written,
143 |     //                       Fullscale = 1*ref, Amp = high speed, binary format,
144 |     //                       no IRQ, DAC12 off, not grouped, calibration on
145 |     DAC12_0CTL = DAC12SREF_0 | DAC12LSEL_0 | DAC12CALON | DAC12IR | DAC12AMP_7;
146 |     DAC12_0DAT = OFFSET_START + LEVEL_SX;   // zu Begin Level S1 ausgeben
147 |     // DAC12_1 Control Reg.: Internal reference, 12-Bit, load when written,
148 |     //                       Fullscale = 1*ref, Amp = high speed, binary format,
149 |     //                       no IRQ, DAC12 off, not grouped, calibration on
150 |     DAC12_1CTL = DAC12SREF_0 | DAC12LSEL_0 | DAC12CALON | DAC12IR | DAC12AMP_7;
151 |     DAC12_1DAT = 0x0000;                // zu Begin 1.25V ausgeben
152 |
153 | /********************* Initialisierung der I/O - Ports ***********************/
154 |
155 |     P1OUT  = 0x00;                                 // Ausgangregister Port1 löschen
156 |     P1DIR |= LED1 | LED2 | LED3 | LED4 | PIN3|PIN0;         // LED-Pins als Ausgang
157 |     P1DIR &= ~( /*PIN0 |*/ PIN1 | PIN2);// | PIN3 );    // JP209...212 als Eingang
158 |     P1SEL &= ~( LED1 | LED2 | LED3 | LED4 );        // IO-Mode für LED-Pins
159 |     P1SEL &= ~( PIN0 | PIN1 | PIN2 | PIN3 );        // IO-Mode für JP209...212
```

```
160     //P1OUT |= (LED1 | LED2 | LED3 | LED4);         // set LEDs on
161
162     P2OUT  = 0x00;                                  // Port2 Ausgangsregister LOW
163     P2DIR  |= PIN2 | COMM_6 | COMM_7 | COMM_8;      //Ausgaenge an Signalcontroller
164     P2SEL &= ~( PIN0 | PIN1 | PIN5 | PIN6 | PIN7 ); // IO-Mode
165     P2SEL  |= PIN2 | PIN3 | PIN4;                   // Comparator Pins
166
167     P3OUT  = 0x00;                                  // Port3 Ausgangsregister LOW
168     P3DIR |= PIN0 | PIN1 | PIN2 | PIN3;
169     P3DIR |= COMM_5; //Ausgang U_Diff not valid => Regelung aktiv wenn high
170     P3SEL &= ~( PIN0 | PIN1 | PIN2 | PIN3 | PIN4 | PIN5 );  // IO-Mode
171     P3SEL |= PIN6 | PIN7;                                   // UART Rx & Tx
172
173     // set outputs for I2C
174     P3SEL |= (1<<1);                    // P3.1 is SDA
175     P3SEL |= (1<<3);                    // P3.3 is SCL
176
177
178     P4OUT  = 0x00;
179     P4DIR |= 0xFF;  // 7Seg_LE0...7 als Ausgang
180     P4SEL &= ~0xFF; // IO-Mode für 7Seg_LE0...7
181
182
183     P5OUT  = 0x00;
184     P5DIR |= PIN0 | PIN1 | PIN2 | PIN3;         // 7Seg_DB0...3 als Ausgang
185     P5SEL |= PIN4 | PIN5;
186     P5SEL &= ~( PIN0 | PIN1 | PIN2 | PIN3 );    // IO-Mode für 7Seg_DB0...3
187
188
189     P6OUT  = 0x00;
190     P6DIR  |= PIN6 | PIN7;  // output DAC
191     P6SEL  |= PIN0 | PIN1 | PIN2 | PIN3 | PIN4 | PIN5 | PIN6 | PIN7;    // ADC- und DAC-Pins
192
193 /******************** Initialisierung externer DAC **********************/
194
195     i2c_init();                 // initiate the I2C
196
197     eint(); // Interrupts zulassen
198
199     ad5667_init(0x0C);          // initialize dac_1
200     ad5667_set(0x0C,0,32350);   // set default value for duty cycle
201     while(i2c_frame.status != 0);
202
203     ad5667_set(0x0C,1,OFFSET_EXT_START);    // set default value for the middle of the signal
204     while(i2c_frame.status != 0);
205
206     ad5667_init(0x0E);          // initialize dac_2
207     ad5667_set(0x0E,0,OFFSET_EXT_START);    // set default value for duty cycle
208     while(i2c_frame.status != 0);
209
210     ad5667_set(0x0E,1,OFFSET_EXT_START);    // set default value for the middle of the signal
211     while(i2c_frame.status != 0);
212
213     ad5667_init(0x0F);          // initialize dac_3
214     ad5667_set(0x0F,0,32350);   // set default value for duty cycle
215     while(i2c_frame.status != 0);
216
217     ad5667_set(0x0F,1,OFFSET_EXT_START);    // set default value for the middle of the signal
218     while(i2c_frame.status != 0);
219
220 /******************** Initialisierung des Systems ***********************/
221
222     // globale Variable zuruecksetzen
223     cleanup();
224
225     // Verstärker
226     g_sig.gain = GAIN_START; //maximale Verstärkung beim PowerOn
227     set_gain();
228     P3OUT |= COMM_5; //Regelung aktiv, U_Diff ungueltig
229
230     sensor_state    = Initial_0Hz;                  // Power on --> Initial_0Hz
231     state_periode   = Search_S1;
232
233     while(TRUE) {   // warten bis DAC-Kalibration beendet
234         if( ((DAC12_0CTL & DAC12CALON) != DAC12CALON) &&
235             ((DAC12_1CTL & DAC12CALON) != DAC12CALON)   ) break;
236     }
237
238     //Timer_B triggert den ADC
239     TBCTL |= TBCLR;                 // Timer B zurücksetzen
```

```
240     TBCTL |= MC_1;                  // Timer B Start compare mode
241     ADC12CTL0 |= ENC;               //enable ADC    RO
242     CACTL1 |= CAON | CAIE;          // Comperator on, IRQ enable
243
244 }//Ende init()
```

```
1  /*****************************************************************************/
2  /*  Autor:          Robert Ostermann                                         */
3  /*  Datum:          01.09.2011 - 24.02.2012                                  */
4  /*  Controller:     MSP430f1611 => "Regelcontroller"                         */
5  /*                                                                           */
6  /*  Datei:          calc_other.c                                             */
7  /*  Beschreibung:   Auswahl der Verstärkung, basierend auf Bachelorthesis Martin Stahl  */
8  /*                  Berechnung des Offsetkorrekturfaktors basiert auf Diplomarbeit      */
9  /*                  Niels Jegenhorst und Bachelorthesis Martin Stahl         */
10 /*                                                                           */
11 /*****************************************************************************/
12
13 #include "main.h"
14 #include "global.h"
15 #include "offset_sin.h"
16
17 /*************************************************************/
18 /*              Überprüfung der Verstärkung                  */
19 /*************************************************************/
20 void check_gain(void)
21 {
22     unsigned int amp;
23
24     amp = (g_adc.u_max_gain-g_adc.u_min_gain);
25
26     if(amp>=9000)       {g_sig.gain = 0;}   // Auswahl der richtigen Verstärkung
27     else if(amp>=4500)  {g_sig.gain = 1;}
28     else if(amp>=2250)  {g_sig.gain = 2;}
29     else if(amp>=1130)  {g_sig.gain = 3;}
30     else if(amp>=560)   {g_sig.gain = 4;}
31     else if(amp>=280)   {g_sig.gain = 5;}
32     else if(amp>=140)   {g_sig.gain = 6;}
33     else if(amp<140)    {g_sig.gain = 7;}
34
35     if(g_sig.gain != g_sig.lgain)
36     {
37         set_compensation_state(TRUE);   //Regelung aktiv
38         set_gain();
39         set_compensation_state(FALSE);  //Diagnose freigeben
40         g_sig.lgain = g_sig.gain;
41     }
42
43     // nächster Zustand
44     sensor_state  = Active;
45     state_periode = Search_S2_Pre;
46
47     // Zurücksetzen der Werte
48     g_adc.u_max_gain = 0x0000;
49     g_adc.u_min_gain = 0xffff;
50 }
51
52 /*************************************************************/
53 /*              Berechnung des Offsets                       */
54 /*************************************************************/
55 void calc_offset(void)
56 {
57     unsigned int amp;
58
59     // Berechnung des Offsets anhand des Duty-Cycle und der Amplitude
60     amp = (g_adc.u_max_middle/g_adc.max_cnt-g_adc.u_min_middle/g_adc.min_cnt)/2;
61
62     if(g_sig.duty>64)
63     {
64         g_sig.offset -= ((uint32_t)amp*(uint32_t)offset_sin[128-g_sig.duty])/15000; //-
65     }
66     else
67     {
68         g_sig.offset += ((uint32_t)amp*(uint32_t)offset_sin[g_sig.duty])/15000;     //+
69     }
70
71     // Zurücksetzen der Werte
72     g_adc.u_max_middle = 0x00000000;
73     g_adc.u_min_middle = 0x00000000;
74     g_adc.min_cnt = 0x0000;
75     g_adc.max_cnt = 0x0000;
76 }
```

```
/*******************************************************************************/
/*  Autor:          Martin Stahl                                               */
/*  Datum:          22.12.2009                                                 */
/*  Hardware:       dm_off_amp_ctrl v0.1                                       */
/*  Controller:     MSP430f1611 => "Regelcontroller"                           */
/*  Toolchain:      MSPGCC v.20060502, Eclipse ganymede-SR2, USBExpress v1.021  */
/*                                                                             */
/*  Datei:          cleanup.c                                                  */
/*  Beschreibung:   Funktion zum zuruecksetzten aller globalen Variablen       */
/*                  Basiert auf Diplomarbeit NJ                                */
/*                                                                             */
/*  Geändert von:   Robert Ostermann                                           */
/*  Datum:          01.09.2011 - 24.02.2012                                    */
/*  Änderungen:     Anpassung an kombinierte logarithmisch und lineare Verstärkerbank   */
/*                  Variablen für Anzahl überschreitungen der Schwellen entfernt*/
/*                  Variablen für Amplituden eingefügt                         */
/*                                                                             */
/*  Funktion:       ok                                                         */
/*                                                                             */
/*******************************************************************************/

#include "main.h"
#include "global.h"


/*************************************************************/
/*             Zuruecksetzten der globalen Variablen         */
/*************************************************************/

void cleanup(void)
{
    PORT_LED &= ~(LED1 | LED2 | LED3 | LED4); //LEDs ausschalten
    P3OUT |= COMM_5; //Regelung aktiv => U_Diff not valid

    g_sig.offset        = OFFSET_EXT_START;//OFFSET_START;
    g_sig.compensations = 0x00;
    g_sig.duty          = 0x00000000;
    g_sig.gain          = GAIN_START;
    g_sig.lgain         = 8;
    g_sig.offset_faktor = 0x0000;
    g_sig.duty_sum      = 0x00000000;
    g_sig.duty_count    = 0x0000;

    set_gain();

    g_adc.u_max_gain    = 0x0000;       //RO
    g_adc.u_min_gain    = 0xffff;       //RO
    g_adc.u_max_offset  = 0x0000;       //RO
    g_adc.u_min_offset  = 0xffff;       //RO
    g_adc.u_max_middle  = 0x00000000;   //RO
    g_adc.u_min_middle  = 0x00000000;   //RO
    g_adc.u_diff    = 0x0000;
    g_adc.u_hb1     = 0x0000;
    g_adc.u_hb2     = 0x0000;
    g_adc.u_diff_b  = 0x0000;
    g_adc.u_sum1    = 0x0000;

    g_time.tp       = 0x0000;
    g_time.tt       = 0x0000;

    ad.cur = OFFSET_EXT_START;
}
```

```
1  /*****************************************************************************/
2  /*  Autor:          Martin Stahl                                             */
3  /*  Datum:          05.01.2010                                               */
4  /*  Hardware:       dm_off_amp_ctrl v0.1                                     */
5  /*  Controller:     MSP430f1611 => "Regelcontroller"                         */
6  /*  Toolchain:      MSPGCC v.20060502, Eclipse ganymede-SR2, USBExpress v1.021*/
7  /*                                                                           */
8  /*  Datei:          isr_adc12.c                                              */
9  /*  Beschreibung:   Interrupt Service Routine für ADC12                      */
10 /*                  Ueberprüft aktuellen Sample auf Ueberschreitung der Scwhellen*/
11 /*                                                                           */
12 /*  Geändert von:   Robert Ostermann                                         */
13 /*  Datum:          01.09.2011 - 24.02.2012                                  */
14 /*  Änderungen:     Anpassung an kombinierte logarithmisch und lineare Verstärkerbank*/
15 /*                  inkrementierung der überschrittenen Schwellen entfernt   */
16 /*                  Ermittlung von min und max für Offset und Verstärkung eingefügt*/
17 /*                                                                           */
18 /*  Funktion:       ok                                                       */
19 /*                                                                           */
20 /*****************************************************************************/
21
22 #include "main.h"
23 #include "global.h"
24 #include "i2c.h"
25 #include "ad5667.h"
26 #include "delog_table.h"
27
28 /*************************************************************/
29 /*         Interrupt Service Routine ADC_12                  */
30 /*************************************************************/
31 interrupt ( ADC12_VECTOR ) isr_adc12 ( void )
32 {
33     unsigned int adc;
34
35     if(ADC12IFG & 0x0002)   // channel 1 interrupt
36     {
37         adc = delog_table[ADC12MEM1];
38         if(adc<g_adc.u_min_offset)  // search minimum for offset
39         {
40             g_adc.u_min_offset = adc;
41         }
42         if(adc>g_adc.u_max_offset)  // search maximum for offset
43         {
44             g_adc.u_max_offset = adc;
45         }
46         if(adc<g_adc.u_min_gain)    // search minimum for offset
47         {
48             g_adc.u_min_gain = adc;
49         }
50         if(adc>g_adc.u_max_gain)    // search maximum for offset
51         {
52             g_adc.u_max_gain = adc;
53         }
54     }
55
56     if(ADC12IV == 0x0002){} // ADC12MEMx overflow
57     if(ADC12IV == 0x0004){} // conversion time overflow
58
59     ADC12IFG = 0x0000;
60     ADC12IE = 0x0002;
61 }
```

```
1  /*****************************************************************************/
2  /*  Autor:          Martin Stahl                                             */
3  /*  Datum:          05.01.2010                                               */
4  /*  Hardware:       dm_off_amp_ctrl v0.1                                     */
5  /*  Controller:     MSP430f1611 => "Regelcontroller"                         */
6  /*  Toolchain:      MSPGCC v.20060502, Eclipse ganymede-SR2, USBExpress v1.021*/
7  /*                                                                           */
8  /*  Datei:          isr_comp_a.c                                             */
9  /*  Beschreibung:   Interrupt Service Routine für Comparator A               */
10 /*                  Beinhaltet die Steuerung der geschachtelten Zustandsautomaten*/
11 /*                  Namensgebungen basieren auf Diplomarbeit NJ              */
12 /*                                                                           */
13 /*  Geändert von:   Robert Ostermann                                         */
14 /*  Datum:          01.09.2011 - 24.02.2012                                  */
15 /*  Änderungen:     Anpassung an kombinierte logarithmisch und lineare Verstärkerbank*/
16 /*                  Ein- und Ausschalten des ADCs aus den einzelnen Zuständen entfernt*/
17 /*                  Setzen des Offsets an externen DAC angepasst             */
18 /*                  Mittelwertbildung des max und min für Offset eingefügt   */
19 /*                                                                           */
20 /* Funktion: ok                                                              */
21 /*                                                                           */
22 /*****************************************************************************/
23
24 #include "main.h"
25 #include "global.h"
26 #include "i2c.h"
27 #include "ad5667.h"
28
29 /*************************************************************/
30 /*         Interrupt Service Routine Comparator A            */
31 /*************************************************************/
32
33 interrupt ( COMPARATORA_VECTOR ) isr_comp_a ( void )
34 {
35     TBCTL |= TBCLR;             // Timer B zurücksetzen
36     TBCTL |= MC_1;              // Timer B Start compare mode
37     ADC12CTL0 &= ~ENC;          /* disable ADC */
38
39     switch (sensor_state) {
40
41 /********************** Systemstart ***********************/
42         case Initial_0Hz: {
43             switch (state_periode) {
44
45 /******************** Periodenanfang **********************/
46                 case Search_S1:
47                         // Timer ist aus || Entprellzeit erreicht?
48                         if((TAR == 0x0000) || (TAR > MIN_TIME_S1)) {
49                             TACTL |= TACLR;                     // Timer zurücksetzen
50                             TACTL |= (ID_3 | MC_2);             // Timer start cont. mode
51                             DAC12_0DAT = OFFSET_START - LEVEL_SX;   // Level S2
52                             CACTL1 |= CAIES;                    // IRQ fallende Flanke
53                             state_periode = Search_S2;
54
55                             g_adc.u_min_middle += g_adc.u_min_offset;   // RO
56                             g_adc.u_min_offset = 0xffff;
57                             g_adc.min_cnt++;
58                         }
59                         break;
60
61 /******************** Periodemitte ***********************/
62                 case Search_S2:
63                         if(TAR > MIN_TIME_S2) {                 // Entprellzeit erreicht
64                             DAC12_0DAT = OFFSET_START + LEVEL_SX;   // Level S1
65                             CACTL1 &= ~CAIES;                   // IRQ steigende Flanke
66                             state_periode = Search_S1;
67                             sensor_state  = Initial_1Hz;
68                             //sensor_state = Active;
69                             g_adc.u_max_middle += g_adc.u_max_offset;   // RO
70                             g_adc.u_max_offset = 0x0000;
71                             g_adc.max_cnt++;
72                         }
73                         break;
74                 default:
75                         state_periode = Search_S1;
76             }
77             break;
78         }
79
```

```
/********************** Regelung beim PowerOn ************************/
        case Initial_1Hz: {
            switch (state_periode) {

                case Search_S1:      //_ Anfang einer Periode zum Messen des Duty __
                        if(TAR > MIN_TIME_S1) {                 // Entprellzeit erreicht?
                            TACTL |= TACLR;                     // Timer zurücksetzen
                            TACTL |= (ID_3 | MC_2);             // Timer start cont. mode
                            DAC12_0DAT = OFFSET_START - LEVEL_SX;   // Level S2
                            CACTL1 |= CAIES;                    // IRQ fallende Flanke
                            #ifdef __LOG_TIME
                                g_log.new_time = TRUE;          // für log_time()
                            #endif
                            state_periode = Search_S2;
                            g_adc.u_min_middle += g_adc.u_min_offset;   // RO
                            g_adc.u_min_offset = 0xffff;
                            g_adc.min_cnt++;
                        }
                        break;

                case Search_S2:      //_ Mitte einer Periode zum Messen des Duty ___
                        if(TAR > MIN_TIME_S2) {                 // Entprellzeit erreicht
                            g_time.tp = TAR;                    // Zählerstand sichern
                            DAC12_0DAT = OFFSET_START + LEVEL_SX;   // Level S1
                            CACTL1 &= ~CAIES;                   // IRQ steigende Flanke
                            state_periode = Search_S1_Calc;
                            //state_periode = Search_S2;
                            //sensor_state  = Active;       // next Sensor State
                            g_adc.u_max_middle += g_adc.u_max_offset;   // RO
                            g_adc.u_max_offset = 0x0000;
                            g_adc.max_cnt++;
                        }
                        break;

/******************** Periodenende => Duty-Cycle berechnen **********************/
                case Search_S1_Calc:

                    if(TAR > MIN_TIME_S1) {                 // Entprellzeit erreicht?
                            g_time.tt = TAR;                    // Zählerstand sichern
                            TACTL |= TACLR;                     // Timer zurücksetzen
                            TACTL |= (ID_3 | MC_2);             // Timer start cont. mode
                            DAC12_0DAT = OFFSET_START - LEVEL_SX;   // Level S2
                            CACTL1 |= CAIES;                    // IRQ fallende Flanke

                            g_adc.u_min_middle += g_adc.u_min_offset;   // RO
                            g_adc.u_min_offset = 0xffff;
                            g_adc.min_cnt++;
                                                    // tp/T * 128 = duty-cycle in %
                            g_sig.duty = ((uint32_t)g_time.tp << 7)  / g_time.tt;

                            #ifdef __OFFSET_CAL
                                if(i2c_frame.status == 0)
                                {
                                    calc_offset();
                                    ad5667_set(0x0E,0,g_sig.offset);
                                }
                            #endif

                            g_sig.compensations++;

                            if(g_sig.compensations == COMPENSATIONS) {
                                g_sig.compensations = 0;
                                state_periode = Search_S2_Check_Gain;
                            }
                            else {  // eine Periode überspringen für neue Kompensation
                                state_periode = Search_S2_Pre;
                            }
                        }
                        break;

                case Search_S2_Pre:  //_ Mitte einer Periode, vor Search_S1

                        if(TAR > MIN_TIME_S2) {                 // Entprellzeit erreicht
                            g_time.tp = TAR;
                            DAC12_0DAT = OFFSET_START + LEVEL_SX;   // Level S1
                            CACTL1 &= ~CAIES;                   // IRQ steigende Flanke
                            state_periode = Search_S1;

                            g_adc.u_max_middle += g_adc.u_max_offset;   // RO
                            g_adc.u_max_offset = 0x0000;
```

```
160                         g_adc.max_cnt++;
161                     }
162                     break;
163
164                 case Search_S2_Check_Gain:
165                     if(TAR > MIN_TIME_S2) {                 // Entprellzeit erreicht
166
167                         g_time.tp = TAR;                    // Zählerstand Timer A sichern
168                         DAC12_0DAT = OFFSET_START + LEVEL_SX;   // Level S1
169                         CACTL1 &= ~CAIES;                   // IRQ steigende Flanke
170                         state_periode = Search_S1_Check_Gain;
171                         //sensor_state  = Active;       // next Sensor State
172
173                         g_adc.u_max_middle += g_adc.u_max_offset;   // RO
174                         g_adc.u_max_offset = 0x0000;
175                         g_adc.max_cnt++;
176                     }
177                     break;
178
179                 case Search_S1_Check_Gain:
180                     if(TAR > MIN_TIME_S1) {                 // Entprellzeit erreicht?
181
182                         TACTL |= TACLR;                     // Timer zurücksetzen
183                         TACTL |= (ID_3 | MC_2);             // Timer start cont. mode
184                         DAC12_0DAT = OFFSET_START - LEVEL_SX;   // Level S2
185                         CACTL1 |= CAIES;                    // IRQ fallende Flanke
186
187                         g_adc.u_min_middle += g_adc.u_min_offset;   // RO
188                         g_adc.u_min_offset = 0xffff;
189                         g_adc.min_cnt++;
190
191                         check_gain(); //Verstärkung überprüfen und Folgezustand festlegen
192                     }
193                     break;
194             }
195             break;
196         }
197
198 /********************** Normaler Betrieb "ACTIVE" ************************/
199         case Active:
200             switch (state_periode) {
201
202 /******************** Periodenanfang *********************/
203                 case Search_S1:
204                     g_time.tt = TAR;                // Zählerstand sichern
205                     TACTL |= TACLR;                 // Timer A zurücksetzen
206                     TACTL |= (ID_3 | MC_2);         // Timer A Start Continuis - Mode
207                     DAC12_0DAT = OFFSET_START - LEVEL_SX;   // Level S2
208                     CACTL1 |= CAIES;                // IRQ Comp. bei fallender Flanke
209                     state_periode = Search_S2;
210                     g_adc.u_min_middle += g_adc.u_min_offset;   // RO
211                     g_adc.u_min_offset = 0xffff;
212                     g_adc.min_cnt++;
213                     break;
214
215 /******************** Periodenmitte *********************/
216                 case Search_S2:
217                     g_time.tp = TAR;                    // Zählerstand Timer A sichern
218                     DAC12_0DAT = OFFSET_START + LEVEL_SX;   // Level S1
219                     CACTL1 &= ~CAIES;                   // IRQ Comp. bei steigender Flanke
220
221                     g_adc.u_max_middle += g_adc.u_max_offset;   // RO
222                     g_adc.u_max_offset = 0x0000;
223                     g_adc.max_cnt++;
224
225                     #ifdef __ACTIVE_COMPENSATION
226                         state_periode = Search_S1_Calc;
227                     #else
228                         state_periode = Search_S1;
229                     #endif
230
231                     break;
232
233
234
235 /******************** Periodenende => Duty-Cycle berechnen *********************/
236                 case Search_S1_Calc:
237
238                     if(TAR > MIN_TIME_S1)                   // Entprellzeit erreicht?
239                     {
```

```
240                         g_time.tt = TAR;                    // Zählerstand sichern
241                         TACTL |= TACLR;                     // Timer zurücksetzen
242                         TACTL |= (ID_3 | MC_2);             // Timer start cont. mode
243                         DAC12_0DAT = OFFSET_START - LEVEL_SX;   // Level S2
244                         CACTL1 |= CAIES;                    // IRQ fallende Flanke
245
246                         g_adc.u_min_middle += g_adc.u_min_offset;   // RO
247                         g_adc.u_min_offset = 0xffff;
248                         g_adc.min_cnt++;
249
250                         // aktuellen Duty-Cycle ausrechnen und aufsummieren
251                         #ifdef __MEAN_DUTY_DEPEND_ON_GAIN
252                         if(g_sig.duty_count < (1<<(g_sig.gain+2)))
253                         {
254                         #else
255                         if(g_sig.duty_count < (1<<PERIODES_FOR_MEAN_DUTY))
256                         {
257                         #endif
258                             g_sig.duty = ((uint32_t)g_time.tp << 7)  / g_time.tt; //Duty rechnen
259                             g_sig.duty_sum += g_sig.duty;
260                             g_sig.duty_count++;
261                             state_periode = Search_S2_Pre;
262                         }
263                         else // Bildung eines Mittelwertes für den Duty-Cycle über x Perioden
264                         {
265                             //Division durch Anzahl der aufsummierten Duty-Cycles
266                             #ifdef __MEAN_DUTY_DEPEND_ON_GAIN
267                             g_sig.duty = g_sig.duty_sum >> (g_sig.gain+2);
268                             #else
269                             g_sig.duty = g_sig.duty_sum >> PERIODES_FOR_MEAN_DUTY;
270                             #endif
271                             g_sig.duty_count = 0;
272                             g_sig.duty_sum = 0;
273
274                             //Signal ist offsetfrei
275                             if(g_sig.duty > 64-DUTY_TOLERANCE && g_sig.duty < 64+DUTY_TOLERANCE)
276                             {
277                                 state_periode = Search_S2_Check_Gain;
278                             }
279                             else
280                             {
281                                 set_compensation_state(TRUE); //Regelung aktiv
282
283                                 #ifdef __OFFSET_CAL
284                                     calc_offset();
285                                     if(i2c_frame.status == 0)
286                                     {
287                                         ad5667_set(0x0E,0,g_sig.offset);
288                                     }
289                                 #endif
290
291                                 set_compensation_state(FALSE); //Regelung abgeschlossen
292                                 state_periode = Search_S2_Pre;
293                             }
294                         }
295                     }
296
297                     break;
298
299 /******************* Periodenmitte *********************/
300                 case Search_S2_Pre:
301
302                     if(TAR > MIN_TIME_S2) {                 // Entprellzeit erreicht?
303                         g_time.tp = TAR;
304                         DAC12_0DAT = OFFSET_START + LEVEL_SX;   // Level S1
305                         CACTL1 &= ~CAIES;                   // IRQ steigende Flanke
306                         state_periode = Search_S1;
307
308                         g_adc.u_max_middle += g_adc.u_max_offset;   // RO
309                         g_adc.u_max_offset = 0x0000;
310                         g_adc.max_cnt++;
311                     }
312                     break;
313
314
315                 case Search_S2_Check_Gain:
316                     if(TAR > MIN_TIME_S2) {                 // Entprellzeit erreicht?
317
318                         g_time.tp = TAR;                    // Zählerstand Timer A sichern
319                         DAC12_0DAT = OFFSET_START + LEVEL_SX;   // Level S1
```

```
320                         CACTL1 &= ~CAIES;                   // IRQ steigende Flanke
321                         state_periode = Search_S1_Check_Gain;
322
323                         g_adc.u_max_middle += g_adc.u_max_offset;   // RO
324                         g_adc.u_max_offset = 0x0000;
325                         g_adc.max_cnt++;
326                     }
327                     break;
328
329                 case Search_S1_Check_Gain:
330                     if(TAR > MIN_TIME_S1) {                 // Entprellzeit erreicht?
331
332                         TACTL |= TACLR;                     // Timer zurücksetzen
333                         TACTL |= (ID_3 | MC_2);             // Timer start cont. mode
334                         DAC12_0DAT = OFFSET_START - LEVEL_SX;   // Level S2
335                         CACTL1 |= CAIES;                    // IRQ fallende Flanke
336
337                         g_adc.u_min_middle += g_adc.u_min_offset;   // RO
338                         g_adc.u_min_offset = 0xffff;
339                         g_adc.min_cnt++;
340
341                         check_gain(); //Verstärkung überprüfen und Folgezustand festlegen
342                     }
343                     break;
344
345
346                 default:
347                     state_periode = Search_S1;
348                     sensor_state  = Initial_0Hz;
349                     break;
350             }
351
352     } // Ende switch(sensor_state)
353
354
355     CACTL1 &= ~CAIFG;   // IRQ Flag löscht automatisch, aber da bei niedrigen Frequenzen
356                         // der Comparator prellt muss das widerholt gelöscht werden
357
358     ADC12CTL0 |= ENC;   //enable ADC
359 }
```

```
1  /*******************************************************************************/
2  /*  Autor:          Martin Stahl                                               */
3  /*  Datum:          05.01.2010                                                 */
4  /*  Hardware:       dm_off_amp_ctrl v0.1                                       */
5  /*  Controller:     MSP430f1611 => "Regelcontroller"                           */
6  /*  Toolchain:      MSPGCC v.20060502, Eclipse ganymede-SR2, USBExpress v1.021  */
7  /*                                                                             */
8  /*  Datei:          isr_timer_a1.c                                             */
9  /*  Beschreibung:   Interrupt Service Routine für Timer A                      */
10 /*                  Zur Stillstandserkennung                                   */
11 /*                  Basiert auf Diplomarbeit NJ                                */
12 /*                                                                             */
13 /*  Geändert von:   Robert Ostermann                                           */
14 /*  Datum:          01.09.2011 - 24.02.2012                                    */
15 /*  Änderungen:     Anpassung an kombinierte logarithmisch und lineare Verstärkerbank   */
16 /*                  Setzen des Offsets an externen DAC angepasst               */
17 /*                                                                             */
18 /*  Funktion:       ok                                                         */
19 /*                                                                             */
20 /*******************************************************************************/
21
22 #include "main.h"
23 #include "global.h"
24 #include "ad5667.h"
25 #include "i2c.h"
26
27 /*************************************************************/
28 /*         Interrupt Service Routine Timer A                 */
29 /*************************************************************/
30
31 interrupt ( TIMERA1_VECTOR ) isr_timer_a1 ( void )
32 {
33         // Routine wird aufgerufen, wenn Überlauf des Zeitgeber Timer_A auftritt.
34         // Idealerweise, wenn die Zahnfrequnez unter 1Hz fällt. Real aufgrund der Länge
35         // des Zeitgebers, sowie dessen Taktes bereits nach 838ms (1,2Hz).
36         // Somit weicht dieser Punkt von der eigentlichen Spezifikation ab.
37
38     if((TACTL & TAIFG) == TAIFG) {          // Timer Overflow
39         TACTL |= TACLR;                     // Timer zurücksetzen
40         TACTL |= (ID_3 | MC_2);             // Timer start cont. mode
41
42                                             // Wenn Sensor vom Zustand Aktiv in den
43                                             // Zustand Init0_Hz zurückfallen soll
44         #ifdef __INIT_FALL_BACK
45             cleanup();
46             DAC12_0DAT = OFFSET_START + LEVEL_SX;   // Schwellwert Comp. Level S1
47             //DAC12_1DAT = OFFSET_START;            // Anfangswert Offsetcompensation
48             if(i2c_frame.status == 0)
49             {
50                 ad5667_set(0x0E,0,OFFSET_EXT_START);
51             }
52             CACTL1 &= ~CAIES;                       // IRQ Comparator steigende Flanke
53
54             g_sig.compensations = 0x00;             // Anzahl der Offsetcomp = 0
55             g_sig.offset = OFFSET_EXT_START;        // Anfangswert Offsetcompensation
56
57             state_periode = Search_S1;              // Automaten zurücksetzen
58             sensor_state  = Initial_0Hz;
59
60
61         #else                           // Wenn Sensor nicht zurückfallen soll,
62                                         // z.B. bei Messreihenaufnahme
63             if(sensor_state == Active) {    // nur State_Calc zurücksetzen wenn
64                                             // im Zustand Active
65         //      state_calc = S1_Prepare_Measure_Periode;
66             }
67             else {                          // nur zurückfallen, wenn nicht im
68                                             // Zustand Active
69                 DAC12_0DAT = OFFSET_START + LEVEL_SX;   // Schwellwert Comp. Level S1
70                 DAC12_1DAT = OFFSET_START;              // Anfangswert Offsetcompensation
71                 CACTL1    &= ~CAIES;                    // IRQ Comparator steigende Flanke
72
73                 g_sig.compensations = 0x00;             // Anzahl der Offsetcomp = 0
74                 g_sig.offset        = OFFSET_START;     // Anfangswert Offsetcompensation
75
76                 state_periode = Search_S1;              // Automaten zurücksetzen
77                 sensor_state  = Initial_0Hz;
78             }
79         #endif
```

```
80 |
81 |         g_time.frequenz           = 0x0000; // Frequenzanzeige = 0
82 |
83 |         TACTL &= ~TAIFG;                    // Flag loeschen
84 |     }
85 | }
```

```
1  /*****************************************************************************/
2  /*  Autor:          Martin Stahl                                             */
3  /*  Datum:          17.12.2009                                               */
4  /*  Hardware:       dm_off_amp_ctrl v0.1                                     */
5  /*  Controller:     MSP430f1611 => "Regelcontroller"                         */
6  /*  Toolchain:      MSPGCC v.20060502, Eclipse ganymede-SR2, USBExpress v1.021*/
7  /*                                                                           */
8  /*  Datei:          send.c                                                   */
9  /*  Beschreibung:   Funktionen zur Datenausgabe auf der seriellen Schnittstelle*/
10 /*                  und zur Kommunikation mit dem Signalcontroller           */
11 /*                                                                           */
12 /*  Funktion:       ok                                                       */
13 /*                                                                           */
14 /*****************************************************************************/
15
16 #include "main.h"
17 #include "global.h"
18
19 /*************************************************************/
20 /*           Zur Kommunikation mit Signalcontroller          */
21 /*************************************************************/
22 void set_compensation_state(uint8_t state) {
23
24     //Regelung aktiv => U_Diff not valid
25     if(state) {
26         P3OUT |= COMM_5;
27         PORT_LED |= PIN3;
28         PORT_LED &= ~(LED2 | LED3);
29         PORT_LED |= LED1; //rote LED ein
30
31     }
32
33     //Sensordiagnose freigeben
34     else {
35         P3OUT &= ~COMM_5;
36         PORT_LED &= ~PIN3;
37         PORT_LED &= ~(LED1 | LED3);
38         PORT_LED |= LED2; //grüne LED ein
39     }
40 }
41
42 /*************************************************************/
43 /*              Ausgabe auf serieller Schnittstelle          */
44 /*************************************************************/
45 //--- Funktion send_usart ---------------------------------------------------|
46 uint8_t send_usart(uint8_t value) {
47
48         // Ausgabe einer 8-Bit Variablen mit dem USART1.
49         // Wenn eine Ausgabe noch aktiv ist, wird nichts geändert
50
51     if((U1TCTL & TXEPT) == TXEPT) {     // wenn Buffer & Register leer --> Daten senden
52         ME2  &= ~UTXE1;                 // USART1 TX disable
53         U1TXBUF = value;                // Daten in Transmitter Buffer (max 8Bit)
54         ME2  |= UTXE1;                  // USART1 TX enable
55         return TRUE;
56     }
57     else
58         return FALSE;
59 }
```

```
1  /*****************************************************************************/
2  /*  Autor:          Martin Stahl                                             */
3  /*  Datum:          22.12.2009                                               */
4  /*  Hardware:       dm_off_amp_ctrl v0.1                                     */
5  /*  Controller:     MSP430f1611 => "Regelcontroller"                         */
6  /*  Toolchain:      MSPGCC v.20060502, Eclipse ganymede-SR2, USBExpress v1.021*/
7  /*                                                                           */
8  /*  Datei:          set_gain.c                                               */
9  /*  Beschreibung:   Stellt den Multiplexer auf die ausgewählte Verstärkung   */
10 /*                  Gibt Verstaerkung zusaetzlich auf COMM_6...8 aus          */
11 /*                  (Kommunikationsleitungen zum Signalcontroller)           */
12 /*                  Einzustellende Verstärkung ist in g_sig.gain gespeichert */
13 /*                                                                           */
14 /*  Geändert von:   Robert Ostermann                                         */
15 /*  Datum:          01.09.2011 - 24.02.2012                                  */
16 /*  Änderungen:     Anpassung an kombinierte logarithmisch und lineare Verstärkerbank */
17 /*                  Auswahl der Verstärkung an Multiplexer angepasst         */
18 /*                                                                           */
19 /*  Funktion:       ok                                                       */
20 /*                                                                           */
21 /*****************************************************************************/
22
23 #include "main.h"
24 #include "global.h"
25 #include "signal.h"
26
27 /*************************************************************/
28 /*                  Einstellung der Verstärkung              */
29 /*************************************************************/
30 void set_gain()
31 {
32     if(g_sig.gain == 0)                       // Gain 1
33     {
34         P4OUT = (P4OUT&0xf8) + 0;             // diff
35         P4OUT = (P4OUT&0x8f) + (0<<4);        // hb1
36         P5OUT = (P5OUT&0xf8) + 0;             // hb2
37         P2OUT &= ~(COMM_6 | COMM_7 | COMM_8);
38     }
39     else if(g_sig.gain == 1)                  // Gain 2
40     {
41         P4OUT = (P4OUT&0xf8) + 4;             // diff
42         P4OUT = (P4OUT&0x8f) + (4<<4);        // hb1
43         P5OUT = (P5OUT&0xf8) + 4;             // hb2
44         P2OUT &= ~(COMM_6 | COMM_7 | COMM_8);
45         P2OUT |= COMM_6;
46     }
47     else if(g_sig.gain == 2)                  // Gain 4
48     {
49         P4OUT = (P4OUT&0xf8) + 7;             // diff
50         P4OUT = (P4OUT&0x8f) + (7<<4);        // hb1
51         P5OUT = (P5OUT&0xf8) + 7;             // hb2
52         P2OUT &= ~(COMM_6 | COMM_7 | COMM_8);
53         P2OUT |= COMM_7;
54     }
55     else if(g_sig.gain == 3)                  // Gain 8
56     {
57         P4OUT = (P4OUT&0xf8) + 2;             // diff
58         P4OUT = (P4OUT&0x8f) + (2<<4);        // hb1
59         P5OUT = (P5OUT&0xf8) + 2;             // hb2
60         P2OUT &= ~(COMM_6 | COMM_7 | COMM_8);
61         P2OUT |= COMM_6 | COMM_7;
62     }
63     else if(g_sig.gain == 4)                  // Gain 16
64     {
65         P4OUT = (P4OUT&0xf8) + 1;             // diff
66         P4OUT = (P4OUT&0x8f) + (1<<4);        // hb1
67         P5OUT = (P5OUT&0xf8) + 1;             // hb2
68         P2OUT &= ~(COMM_6 | COMM_7 | COMM_8);
69         P2OUT |= COMM_8;
70     }
71     else if(g_sig.gain == 5)                  // Gain 32
72     {
73         P4OUT = (P4OUT&0xf8) + 6;             // diff
74         P4OUT = (P4OUT&0x8f) + (6<<4);        // hb1
75         P5OUT = (P5OUT&0xf8) + 6;             // hb2
76         P2OUT &= ~(COMM_6 | COMM_7 | COMM_8);
77         P2OUT |= COMM_6 | COMM_8;
78     }
79     else if(g_sig.gain == 6)                  // Gain 64
```

```
80      {
81          P4OUT = (P4OUT&0xf8) + 5;               // diff
82          P4OUT = (P4OUT&0x8f) + (5<<4);          // hb1
83          P5OUT = (P5OUT&0xf8) + 5;               // hb2
84          P2OUT &= ~(COMM_6 | COMM_7 | COMM_8);
85          P2OUT |= COMM_7 | COMM_8;
86      }
87      else if(g_sig.gain == 7)                    // Gain 128
88      {
89          P4OUT = (P4OUT&0xf8) + 3;               // diff
90          P4OUT = (P4OUT&0x8f) + (3<<4);          // hb1
91          P5OUT = (P5OUT&0xf8) + 3;               // hb2
92          P2OUT &= ~(COMM_6 | COMM_7 | COMM_8);
93          P2OUT |= COMM_6 | COMM_7 | COMM_8;
94      }
95  }
```

```
1  /*******************************************************************************/
2  /*  Autor:          Robert Ostermann                                           */
3  /*  Datum:          01.09.2011 - 24.02.2012                                    */
4  /*  Controller:     MSP430f1611 => "Regelcontroller"                           */
5  /*                                                                             */
6  /*  Datei:          i2c.h                                                      */
7  /*  Beschreibung:   Headerdatei zu ic2.c                                       */
8  /*                                                                             */
9  /*******************************************************************************/
10
11 #ifndef I2C_H_
12 #define I2C_H_
13
14     #ifndef EXTERN
15     #define EXTERN extern
16     #endif
17
18     struct {
19         unsigned char device;       // 7bit address of the device
20         unsigned char stop;         // stop condition of the frame
21         unsigned char buffer[100];
22         unsigned char length;       // length of the message
23         char status;                // high: currently data will be send / low: unit ready to send data
24         unsigned char cnt;          // count the transmitted data
25         unsigned int duty;
26         unsigned long Th;
27         unsigned int Tl;
28     } EXTERN volatile i2c_frame;
29
30     extern void i2c_init(void);
31     extern char i2c_send(void);
32     extern char i2c_receive(void);
33
34 #endif /* I2C_H_ */
```

```c
/*****************************************************************************/
/*  Autor:          Robert Ostermann                                         */
/*  Datum:          01.09.2011 - 24.02.2012                                  */
/*  Controller:     MSP430f1611 => "Regelcontroller"                         */
/*                                                                           */
/*  Datei:          i2c.c                                                    */
/*  Beschreibung:   Initialisierung der I2C-Schnittstelle des MSP430F1611    */
/*                  Funktionen zum senden und empfangen von Datanpaketen     */
/*                  Interrupt Service Handler der I2C-Schnittstelle          */
/*                                                                           */
/*****************************************************************************/

#include <msp430x16x.h>
#include <signal.h>

#include "i2c.h"

/**
 * initialize the i2c-bus on USART0
 */
void i2c_init(void)
{
    i2c_frame.cnt = 0;
    i2c_frame.device = 0;
    i2c_frame.stop = 1;
    i2c_frame.length = 0;
    i2c_frame.status = 0;

    P3SEL |= (1<<1);                    // P3.1 is SDA
    P3SEL |= (1<<3);                    // P3.3 is SCL

    U0CTL |= ( (1<<5)|(1<<2) );         // set I2C and SNYC
    U0CTL &= ~(1<<0);                   // disable I2C (~I2CEN)

    I2CTCTL |= ( (1<<4)|(1<<5) );       // SMCLK is clock source for I2C

    I2CPSC = 0x00;                      // prescaler = 1
    I2CSCLH = 0x0A;                     // 12 Takte vom prescaler bilden high period
    I2CSCLL = 0x0A;                     // 12 clocks from prescaler build the low period

    U0CTL |= (1<<0);                    // enable I2C
}

/**
 * send the frame, saved in i2c_frame.buffer
 */
char i2c_send(void)
{
    if(i2c_frame.status == 1)           // unit not ready to send data
    {
        return 1;
    }

    i2c_frame.status = 1;               // block the unit
    i2c_frame.cnt = 0;                  // set frame count to zero

    U0CTL &= ~(1<<0);                   // disable I2C (~I2CEN)

    I2CNDAT = i2c_frame.length;         // set length of data to be transmit
    I2CSA = i2c_frame.device & 0x7f;    // set the device address
    U0CTL |= (1<<1);                    // master, must set because is cleared after a STOP condition
    I2CTCTL |= (1<<3);                  // Transmitter

    I2CIE = 0x00;                       // disable all interrupts

    U0CTL |= (1<<0);                    // enable I2C

    // send STOP condition?
    if( (i2c_frame.stop == 1) ? (I2CTCTL |= (1<<1)) : (I2CTCTL &= ~(1<<1)) );
    I2CTCTL |= (1<<0);                  // send a START condition

    I2CIE |= (1<<1);                    // no acknowledge interrupt
    I2CIE |= (1<<5);                    // enable transmit ready interrupt
    I2CIE |= (1<<3);                    // register access ready interrupt

    return 0;                           // transmission start correctly
}

/**
```

```
80  | * received a frame, will be saved in i2c_frame.buffer
81  | */
82  |char i2c_receive(void)
83  |{
84  |    if(i2c_frame.status == 1)           // unit not ready to send data
85  |    {
86  |        return 1;
87  |    }
88  |
89  |    i2c_frame.status = 1;               // block the unit
90  |    i2c_frame.cnt = 0;                  // set frame count to zero
91  |
92  |    U0CTL &= ~(1<<0);                   // disable I2C (~I2CEN)
93  |
94  |    I2CNDAT = i2c_frame.length;         // set length of data to be transmit
95  |    I2CSA = i2c_frame.device & 0x7f;    // set the device address
96  |    U0CTL |= (1<<1);                    // master, must set because is cleared after a STOP condition
97  |    I2CTCTL &= ~(1<<3);                 // Transmitter
98  |
99  |    I2CIE = 0x00;                       // disable all interrupts
100 |
101 |    U0CTL |= (1<<0);                    // enable I2C
102 |
103 |    // send STOP condition?
104 |    if( (i2c_frame.stop == 1) ? (I2CTCTL |= (1<<1)) : (I2CTCTL &= ~(1<<1)) );
105 |    I2CTCTL |= (1<<0);                  // send a START condition
106 |
107 |    I2CIE |= (1<<4);                    // enable receive ready interrupt
108 |    I2CIE |= (1<<1);                    // no acknowledge interrupt
109 |    return 0;
110 |}
111 |
112 |/**
113 | * I2C interrupt handler
114 | */
115 |interrupt ( UART0TX_VECTOR ) i2c_isr(void)
116 |{
117 |    switch(I2CIV&0x00ff)
118 |    {
119 |        case 0x00: break;               // no interrupt
120 |
121 |        case 0x02: break;               // arbitration lost
122 |
123 |        case 0x04:                      // no acknowledgment
124 |            I2CIE = 0x00;               // disable all interrupts
125 |            i2c_frame.status = 2;       // block für neue übertragung freigeben
126 |            break;
127 |
128 |        case 0x06: break;               // own address
129 |
130 |        case 0x08:                      // register access ready
131 |            if(i2c_frame.cnt == i2c_frame.length)   // all data transmit
132 |            {
133 |                i2c_frame.status = 0;   // release the unit
134 |                I2CIE = 0x00;           // disable all interrupts
135 |            }
136 |            break;
137 |
138 |        case 0x0A:                      // receive data ready
139 |            i2c_frame.buffer[i2c_frame.cnt] = I2CDRB;
140 |            i2c_frame.cnt++;
141 |            if(i2c_frame.cnt == i2c_frame.length)   // all data received
142 |            {
143 |                i2c_frame.status = 0;   // release the unit
144 |                I2CIE   = 0x00;         // all data received, disable all interrupts
145 |            }
146 |            break;
147 |
148 |        case 0x0C:                      // Transmit data ready
149 |            if(i2c_frame.cnt == i2c_frame.length)   // all data transmit
150 |            {
151 |                I2CIE &= ~(1<<5);       // disable transmit ready interrupt
152 |                break;
153 |            }
154 |            I2CDRB = i2c_frame.buffer[i2c_frame.cnt];
155 |            i2c_frame.cnt++;
156 |            break;
157 |
158 |        case 0x0E: break;               // general call
159 |
```

```
160         case 0x10: break;               // START condition received
161     }
162 }
```

```
1  /*****************************************************************************/
2  /*  Autor:          Robert Ostermann                                         */
3  /*  Datum:          01.09.2011 - 24.02.2012                                  */
4  /*  Controller:     MSP430f1611 => "Regelcontroller"                         */
5  /*                                                                           */
6  /*  Datei:          ad5667.h                                                 */
7  /*  Beschreibung:   Headerdatei zu ad5667.c                                  */
8  /*                                                                           */
9  /*****************************************************************************/
10
11 #ifndef AD5667_H_
12 #define AD5667_H_
13
14     extern void ad5667_init(unsigned char address);
15     extern void ad5667_set(unsigned char address, unsigned char dac, unsigned int value);
16
17 #endif /* AD5776_H_ */
```

```
1  /*****************************************************************************************/
2  /*  Autor:          Robert Ostermann                                                     */
3  /*  Datum:          01.09.2011 - 24.02.2012                                              */
4  /*  Controller:     MSP430f1611 => "Regelcontroller"                                     */
5  /*                                                                                       */
6  /*  Datei:          ad5667.c                                                             */
7  /*  Beschreibung:   Initialisierung des DAC AD5667 von Analog Devices                    */
8  /*                  Setzen der analogen Ausgangsspannung                                 */
9  /*                                                                                       */
10 /*****************************************************************************************/
11
12 #include "i2c.h"
13
14 /**
15  * Initialize the DAC ad5667
16  */
17 void ad5667_init(unsigned char address)
18 {
19     i2c_frame.device = address;     // set slave address
20     i2c_frame.length = 3;           // send 3 bytes
21     i2c_frame.stop = 1;             // send a stop condition
22     i2c_frame.buffer[0] = 0x20;     // disable LDAC
23     i2c_frame.buffer[1] = 0x00;     // disable LDAC
24     i2c_frame.buffer[2] = 0x03;     // disable LDAC
25
26     i2c_send(); // send the data
27     while(i2c_frame.status == 1);
28
29     //  set internal reference off, default off
30     i2c_frame.buffer[0] = 0x38;     // set internal reference
31     i2c_frame.buffer[1] = 0x00;     // set internal reference
32     i2c_frame.buffer[2] = 0x00;     // set internal reference off
33
34     i2c_send(); // send the data
35     while(i2c_frame.status == 1);
36 }
37
38 /**
39  * set the output voltage, on the device with the given address and selected channel
40  */
41 void ad5667_set(unsigned char address, unsigned char dac, unsigned int value)
42 {
43     i2c_frame.device = address;                 // set slave address
44     i2c_frame.length = 3;                       // send 3 bytes
45     i2c_frame.stop = 1;                         // send a stop condition
46
47     i2c_frame.buffer[0] = (dac&0x07) + 0x18;    // disable LDAC
48     i2c_frame.buffer[2] = value&0x00ff;         // set low value
49     i2c_frame.buffer[1] = (value>>8)&0x00ff;    // set upper value
50
51     i2c_send(); // send the data
52 }
```

```
/*******************************************************************************/
/*  Autor:          Robert Ostermann                                           */
/*  Datum:          01.09.2011 - 24.02.2012                                    */
/*  Controller:     MSP430f1611 => "Regelcontroller"                           */
/*                                                                             */
/*  Datei:          delog_table.h                                              */
/*  Beschreibung:   Lookup-Tabelle der Delogarithmierung                       */
/*                                                                             */
/*******************************************************************************/

#ifndef DELOG_TABLE_H
#define DELOG_TABLE_H

const unsigned int delog_table[4096] = {
      773,    773,    773,    773,    773,    773,    773,    773,
      773,    773,    773,    773,    773,    773,    773,    773,
      773,    773,    773,    773,    773,    773,    773,    773,
      773,    773,    773,    773,    773,    773,    773,    773,
      773,    773,    773,    773,    773,    773,    773,    773,
      773,    773,    773,    773,    773,    773,    773,    773,
      773,    773,    773,    773,    773,    773,    773,    773,
      773,    773,    773,    773,    773,    773,    773,    819,
      974,   1041,   1134,   1283,   1362,   1375,   1478,   1534,
     1727,   1768,   1892,   1853,   1970,   2079,   2174,   2274,
     2188,   2370,   2366,   2552,   2634,   2699,   2714,   2874,
     2909,   2894,   3111,   3078,   3073,   3353,   3379,   3352,
     3472,   3562,   3663,   3720,   3766,   3871,   3956,   4017,
     4059,   4209,   4189,   4314,   4341,   4392,   4551,   4598,
     4662,   4708,   4787,   4840,   4875,   4966,   4950,   5029,
     5113,   5136,   5226,   5274,   5332,   5433,   5465,   5509,
     5520,   5617,   5632,   5762,   5768,   5905,   5952,   6019,
     6071,   6056,   6124,   6159,   6260,   6276,   6317,   6469,
     6571,   6546,   6653,   6673,   6705,   6748,   6826,   6885,
     6870,   6991,   7012,   7078,   7143,   7166,   7222,   7264,
     7237,   7353,   7384,   7418,   7464,   7491,   7550,   7560,
     7631,   7656,   7765,   7745,   7870,   7837,   7891,   7922,
     7955,   7984,   8007,   8052,   8074,   8127,   8107,   8178,
     8205,   8287,   8327,   8348,   8371,   8399,   8416,   8444,
     8473,   8498,   8508,   8541,   8573,   8609,   8624,   8641,
     8666,   8724,   8723,   8741,   8760,   8809,   8830,   8880,
     8894,   8909,   8913,   8967,   8990,   9018,   9005,   9045,
     9073,   9061,   9093,   9126,   9129,   9152,   9172,   9191,
     9225,   9236,   9240,   9273,   9284,   9319,   9323,   9345,
     9356,   9366,   9384,   9416,   9434,   9455,   9485,   9475,
     9512,   9537,   9530,   9566,   9574,   9558,   9616,   9614,
     9635,   9652,   9682,   9689,   9689,   9720,   9738,   9723,
     9752,   9757,   9748,   9788,   9795,   9838,   9825,   9837,
     9856,   9859,   9891,   9895,   9919,   9916,   9930,   9955,
     9945,   9974,   9988,   9991,  10024,  10007,  10042,  10032,
    10061,  10078,  10088,  10082,  10109,  10127,  10134,  10159,
    10158,  10167,  10180,  10192,  10211,  10222,  10216,  10246,
    10259,  10266,  10291,  10291,  10321,  10317,  10332,  10352,
    10352,  10377,  10383,  10387,  10404,  10426,  10426,  10441,
    10459,  10459,  10477,  10468,  10484,  10501,  10513,  10523,
    10528,  10552,  10549,  10556,  10573,  10577,  10590,  10605,
    10611,  10639,  10622,  10638,  10639,  10663,  10665,  10685,
    10698,  10708,  10711,  10733,  10724,  10747,  10760,  10754,
    10769,  10775,  10786,  10806,  10805,  10827,  10825,  10839,
    10842,  10861,  10863,  10863,  10905,  10891,  10887,  10914,
    10926,  10945,  10942,  10963,  10980,  10990,  10998,  10998,
    11012,  11022,  11040,  11042,  11044,  11069,  11076,  11078,
    11097,  11110,  11119,  11116,  11138,  11153,  11157,  11164,
    11164,  11181,  11177,  11191,  11205,  11215,  11230,  11232,
    11237,  11245,  11263,  11280,  11288,  11286,  11295,  11300,
    11322,  11336,  11328,  11348,  11369,  11362,  11386,  11398,
    11400,  11423,  11420,  11429,  11455,  11441,  11463,  11476,
    11489,  11508,  11513,  11522,  11534,  11543,  11550,  11583,
    11578,  11581,  11603,  11613,  11615,  11628,  11642,  11638,
    11659,  11668,  11682,  11689,  11695,  11697,  11714,  11720,
    11727,  11730,  11748,  11745,  11768,  11775,  11773,  11788,
    11793,  11809,  11827,  11827,  11830,  11841,  11848,  11847,
    11861,  11870,  11877,  11884,  11898,  11898,  11918,  11934,
    11931,  11943,  11955,  11963,  11976,  11983,  11995,  11989,
    12008,  12018,  12031,  12035,  12045,  12057,  12072,  12081,
    12089,  12089,  12093,  12112,  12116,  12113,  12120,  12136,
    12151,  12155,  12171,  12175,  12188,  12188,  12197,  12205,
    12202,  12224,  12231,  12234,  12239,  12246,  12259,  12263,
    12277,  12277,  12290,  12298,  12298,  12315,  12315,  12325,
```

```
    12327,  12337,  12337,  12346,  12355,  12370,  12378,  12378,
    12377,  12402,  12397,  12393,  12411,  12424,  12423,  12434,
    12446,  12454,  12454,  12471,  12486,  12486,  12482,  12495,
    12501,  12505,  12516,  12519,  12540,  12544,  12537,  12555,
    12563,  12563,  12584,  12587,  12587,  12605,  12618,  12611,
    12634,  12632,  12639,  12643,  12655,  12668,  12665,  12677,
    12689,  12698,  12706,  12713,  12713,  12735,  12738,  12745,
    12734,  12752,  12752,  12766,  12770,  12782,  12786,  12795,
    12802,  12805,  12810,  12813,  12820,  12820,  12838,  12842,
    12854,  12842,  12842,  12865,  12867,  12875,  12885,  12883,
    12895,  12904,  12897,  12902,  12916,  12919,  12912,  12922,
    12931,  12938,  12940,  12940,  12949,  12959,  12967,  12959,
    12977,  12977,  12967,  12995,  12995,  12994,  13003,  13005,
    13007,  13017,  13016,  13032,  13040,  13041,  13041,  13055,
    13055,  13054,  13055,  13075,  13067,  13068,  13081,  13082,
    13082,  13102,  13100,  13100,  13112,  13109,  13109,  13127,
    13127,  13139,  13143,  13160,  13160,  13150,  13153,  13177,
    13180,  13180,  13180,  13198,  13198,  13191,  13211,  13213,
    13208,  13208,  13226,  13226,  13234,  13249,  13252,  13245,
    13259,  13260,  13257,  13280,  13281,  13280,  13296,  13297,
    13297,  13300,  13314,  13314,  13334,  13334,  13327,  13330,
    13344,  13341,  13352,  13356,  13366,  13366,  13371,  13377,
    13377,  13386,  13392,  13400,  13399,  13406,  13413,  13413,
    13420,  13420,  13431,  13439,  13434,  13443,  13452,  13458,
    13458,  13456,  13466,  13475,  13485,  13480,  13487,  13495,
    13484,  13484,  13513,  13508,  13511,  13511,  13529,  13527,
    13527,  13538,  13538,  13546,  13559,  13559,  13567,  13559,
    13559,  13573,  13576,  13587,  13591,  13591,  13588,  13588,
    13613,  13597,  13608,  13627,  13627,  13627,  13636,  13636,
    13634,  13643,  13643,  13648,  13648,  13659,  13659,  13659,
    13678,  13678,  13680,  13680,  13694,  13694,  13701,  13700,
    13701,  13714,  13714,  13714,  13723,  13723,  13735,  13733,
    13741,  13740,  13740,  13748,  13757,  13757,  13757,  13771,
    13771,  13770,  13770,  13781,  13788,  13788,  13786,  13786,
    13809,  13809,  13802,  13808,  13808,  13816,  13813,  13813,
    13826,  13830,  13830,  13843,  13839,  13850,  13841,  13841,
    13857,  13857,  13863,  13866,  13866,  13877,  13877,  13880,
    13884,  13880,  13880,  13880,  13904,  13902,  13902,  13913,
    13905,  13905,  13926,  13927,  13927,  13927,  13934,  13942,
    13942,  13940,  13940,  13953,  13955,  13963,  13967,  13967,
    13960,  13960,  13980,  13980,  13982,  13982,  13989,  13999,
    13999,  14004,  14004,  14021,  14021,  14014,  14013,  14013,
    14023,  14023,  14038,  14042,  14042,  14037,  14054,  14054,
    14054,  14059,  14059,  14059,  14058,  14058,  14057,  14057,
    14057,  14066,  14066,  14080,  14082,  14082,  14085,  14085,
    14088,  14088,  14102,  14102,  14105,  14105,  14105,  14119,
    14119,  14117,  14117,  14117,  14127,  14127,  14127,  14127,
    14139,  14139,  14145,  14145,  14145,  14145,  14157,  14157,
    14157,  14151,  14165,  14165,  14174,  14174,  14174,  14174,
    14180,  14180,  14191,  14191,  14194,  14194,  14202,  14202,
    14198,  14205,  14205,  14218,  14218,  14221,  14224,  14218,
    14218,  14236,  14236,  14236,  14237,  14237,  14254,  14252,
    14252,  14247,  14262,  14262,  14263,  14260,  14260,  14270,
    14270,  14270,  14270,  14286,  14286,  14286,  14288,  14290,
    14290,  14294,  14294,  14300,  14300,  14301,  14301,  14317,
    14317,  14317,  14317,  14317,  14317,  14324,  14324,  14324,
    14335,  14335,  14335,  14337,  14341,  14341,  14341,  14351,
    14351,  14351,  14355,  14355,  14358,  14358,  14365,  14365,
    14369,  14369,  14371,  14371,  14371,  14378,  14385,  14385,
    14387,  14387,  14387,  14387,  14396,  14396,  14396,  14396,
    14413,  14413,  14409,  14409,  14409,  14427,  14427,  14427,
    14427,  14431,  14431,  14434,  14434,  14434,  14434,  14443,
    14443,  14443,  14440,  14448,  14457,  14448,  14448,  14448,
    14464,  14464,  14464,  14471,  14471,  14477,  14477,  14479,
    14484,  14477,  14477,  14477,  14477,  14504,  14504,  14504,
    14499,  14488,  14504,  14504,  14504,  14504,  14516,  14516,
    14516,  14516,  14516,  14516,  14516,  14516,  14516,  14531,
    14531,  14531,  14531,  14538,  14536,  14536,  14536,  14548,
    14548,  14548,  14547,  14557,  14557,  14556,  14556,  14556,
    14570,  14570,  14570,  14570,  14571,  14571,  14571,  14571,
    14580,  14580,  14580,  14590,  14590,  14590,  14590,  14596,
    14596,  14596,  14602,  14602,  14602,  14594,  14605,  14605,
    14605,  14614,  14614,  14614,  14613,  14610,  14610,  14610,
    14619,  14619,  14619,  14616,  14616,  14625,  14625,  14625,
    14625,  14625,  14636,  14636,  14636,  14636,  14636,  14645,
    14645,  14645,  14645,  14648,  14648,  14648,  14648,  14648,
    14655,  14655,  14655,  14655,  14655,  14655,  14655,  14670,
    14670,  14670,  14670,  14670,  14661,  14661,  14679,  14679,
    14679,  14679,  14679,  14679,  14679,  14689,  14689,  14689,
    14689,  14689,  14687,  14682,  14682,  14682,  14682,  14682,
```

```
    14695,  14695,  14695,  14695,  14695,  14695,  14695,  14695,
    14703,  14703,  14703,  14709,  14709,  14709,  14709,  14709,
    14709,  14709,  14711,  14711,  14711,  14723,  14723,  14723,
    14721,  14721,  14721,  14718,  14718,  14718,  14730,  14730,
    14730,  14730,  14730,  14733,  14733,  14733,  14731,  14731,
    14731,  14731,  14738,  14738,  14738,  14738,  14738,  14738,
    14744,  14744,  14744,  14744,  14748,  14748,  14748,  14748,
    14750,  14750,  14748,  14755,  14755,  14755,  14755,  14759,
    14759,  14759,  14759,  14762,  14762,  14762,  14767,  14767,
    14767,  14763,  14763,  14763,  14769,  14769,  14769,  14778,
    14778,  14778,  14778,  14778,  14775,  14775,  14775,  14775,
    14776,  14776,  14776,  14776,  14776,  14776,  14776,  14776,
    14776,  14788,  14788,  14788,  14788,  14788,  14785,  14785,
    14785,  14785,  14785,  14785,  14785,  14785,  14785,  14785,
    14801,  14801,  14801,  14801,  14801,  14801,  14801,  14801,
    14801,  14802,  14802,  14802,  14805,  14805,  14805,  14805,
    14805,  14809,  14809,  14809,  14809,  14809,  14813,  14813,
    14813,  14813,  14813,  14813,  14813,  14813,  14822,  14822,
    14822,  14822,  14822,  14822,  14822,  14822,  14827,  14827,
    14827,  14827,  14827,  14827,  14823,  14823,  14823,  14823,
    14823,  14827,  14827,  14827,  14827,  14827,  14827,  14835,
    14835,  14835,  14835,  14835,  14837,  14837,  14837,  14837,
    14837,  14837,  14837,  14840,  14840,  14840,  14840,  14840,
    14840,  14848,  14848,  14848,  14848,  14849,  14849,  14849,
    14849,  14849,  14849,  14858,  14858,  14858,  14858,  14858,
    14858,  14858,  14858,  14858,  14859,  14859,  14859,  14859,
    14859,  14859,  14861,  14861,  14861,  14861,  14861,  14861,
    14861,  14877,  14877,  14877,  14877,  14877,  14877,  14877,
    14877,  14877,  14877,  14877,  14873,  14873,  14873,  14873,
    14873,  14873,  14873,  14873,  14873,  14873,  14873,  14880,
    14880,  14880,  14880,  14880,  14880,  14880,  14883,  14881,
    14881,  14881,  14881,  14881,  14881,  14895,  14895,  14895,
    14895,  14895,  14895,  14895,  14895,  14895,  14895,  14895,
    14898,  14898,  14898,  14898,  14898,  14898,  14898,  14891,
    14891,  14891,  14891,  14891,  14891,  14891,  14891,  14910,
    14910,  14910,  14910,  14910,  14910,  14910,  14910,  14910,
    14909,  14909,  14909,  14909,  14908,  14908,  14908,  14908,
    14908,  14915,  14915,  14915,  14915,  14915,  14915,  14915,
    14915,  14915,  14915,  14915,  14915,  14915,  14915,  14925,
    14925,  14925,  14925,  14925,  14925,  14925,  14925,  14925,
    14925,  14925,  14925,  14932,  14932,  14932,  14936,  14936,
    14936,  14936,  14936,  14936,  14942,  14942,  14942,  14942,
    14942,  14942,  14942,  14942,  14938,  14938,  14938,  14938,
    14942,  14942,  14942,  14942,  14942,  14942,  14942,  14942,
    14942,  14942,  14942,  14942,  14942,  14942,  14942,  14945,
    14945,  14945,  14945,  14945,  14945,  14945,  14945,  14945,
    14945,  14945,  14945,  14945,  14945,  14945,  14953,  14953,
    14953,  14953,  14953,  14955,  14955,  14955,  14955,  14955,
    14955,  14955,  14955,  14955,  14955,  14955,  14955,  14955,
    14955,  14955,  14955,  14955,  14955,  14965,  14965,  14965,
    14965,  14965,  14965,  14965,  14965,  14965,  14965,  14965,
    14965,  14965,  14965,  14965,  14965,  14963,  14963,  14963,
    14963,  14963,  14963,  14963,  14963,  14963,  14963,  14963,
    14963,  14963,  14963,  14963,  14963,  14970,  14970,  14970,
    14970,  14970,  14970,  14970,  14970,  14970,  14970,  14970,
    14977,  14977,  14979,  14979,  14979,  14979,  14979,  14979,
    14979,  14979,  14979,  14979,  14979,  14979,  14979,  14979,
    14979,  14979,  14979,  14979,  14979,  14979,  14979,  14984,
    14984,  14984,  14984,  14984,  14984,  14984,  14984,  14984,
    14984,  14984,  14984,  14984,  14984,  14984,  14984,  14984,
    14984,  14984,  14984,  14984,  14984,  14993,  14993,  14993,
    14993,  14993,  14993,  14988,  14988,  14988,  14988,  14988,
    14988,  14988,  14988,  14988,  14988,  14988,  14988,  14988,
    14988,  14988,  14988,  14988,  14988,  14988,  14988,  14988,
    14988,  14988,  14994,  14994,  14994,  14994,  14994,  14994,
    14994,  14994,  14994,  14994,  14994,  14994,  14994,  14994,
    14994,  14994,  14994,  14994,  14994,  14994,  14994,  14998,
    14998,  14998,  14998,  14998,  14998,  14998,  14998,  14997,
    14997,  14997,  14997,  14997,  14997,  14997,  14997,  14997,
    14997,  14997,  14997,  14997,  14997,  14997,  15001,  15001,
    15001,  15001,  15001,  15001,  15001,  15001,  15001,  15001,
    15001,  15001,  15002,  15002,  15002,  15002,  15002,  15002,
    15002,  15002,  15002,  15002,  15002,  15002,  15009,  15009,
    15009,  15009,  15009,  15009,  15009,  15009,  15009,  15009,
    15009,  15009,  15009,  15009,  15009,  15009,  15009,  15009,
    15009,  15009,  15009,  15009,  15009,  15009,  15009,  15009,
    15009,  15008,  15008,  15008,  15008,  15008,  15008,  15008,
    15008,  15008,  15008,  15008,  15008,  15008,  15008,  15008,
    15008,  15008,  15008,  15008,  15008,  15008,  15008,  15008,
```

```
240 |    15008,  15017,  15017,  15017,  15017,  15013,  15013,  15013,                |
241 |    15013,  15013,  15013,  15013,  15013,  15013,  15013,  15013,                |
242 |    15013,  15013,  15013,  15013,  15013,  15013,  15013,  15013,                |
243 |    15013,  15013,  15013,  15013,  15013,  15013,  15013,  15013,                |
244 |    15013,  15013,  15013,  15013,  15013,  15013,  15013,  15013,                |
245 |    15013,  15013,  15013,  15013,  15013,  15016,  15016,  15016,                |
246 |    15016,  15016,  15016,  15016,  15016,  15016,  15016,  15016,                |
247 |    15016,  15016,  15016,  15016,  15016,  15016,  15016,  15016,                |
248 |    15016,  15016,  15016,  15016,  15016,  15016,  15016,  15016,                |
249 |    15016,  15016,  15016,  15016,  15016,  15016,  15016,  15016,                |
250 |    15016,  15016,  15016,  15016,  15016,  15016,  15016,  15024,                |
251 |    15024,  15024,  15024,  15024,  15024,  15024,  15024,  15024,                |
252 |    15030,  15030,  15030,  15030,  15030,  15030,  15030,  15030,                |
253 |    15030,  15030,  15030,  15030,  15030,  15030,  15030,  15030,                |
254 |    15030,  15030,  15030,  15030,  15030,  15030,  15030,  15030,                |
255 |    15030,  15030,  15030,  15030,  15030,  15030,  15030,  15030,                |
256 |    15030,  15030,  15030,  15030,  15030,  15033,  15033,  15033,                |
257 |    15033,  15033,  15033,  15033,  15033,  15033,  15033,  15033,                |
258 |    15033,  15033,  15033,  15033,  15033,  15033,  15033,  15033,                |
259 |    15033,  15033,  15033,  15033,  15033,  15033,  15033,  15033,                |
260 |    15033,  15033,  15033,  15033,  15033,  15033,  15033,  15033,                |
261 |    15033,  15033,  15033,  15033,  15030,  15030,  15030,  15030,                |
262 |    15030,  15030,  15030,  15030,  15030,  15030,  15030,  15030,                |
263 |    15030,  15030,  15030,  15030,  15030,  15030,  15038,  15038,                |
264 |    15038,  15038,  15038,  15038,  15038,  15038,  15038,  15038,                |
265 |    15038,  15038,  15038,  15038,  15038,  15038,  15038,  15038,                |
266 |    15038,  15040,  15040,  15040,  15040,  15040,  15040,  15040,                |
267 |    15040,  15040,  15040,  15040,  15040,  15040,  15040,  15040,                |
268 |    15040,  15040,  15040,  15040,  15040,  15040,  15040,  15040,                |
269 |    15040,  15040,  15040,  15041,  15041,  15041,  15041,  15041,                |
270 |    15041,  15042,  15042,  15042,  15042,  15042,  15042,  15042,                |
271 |    15042,  15042,  15042,  15042,  15042,  15042,  15042,  15042,                |
272 |    15042,  15042,  15042,  15042,  15042,  15042,  15042,  15042,                |
273 |    15042,  15042,  15042,  15042,  15047,  15047,  15047,  15047,                |
274 |    15047,  15047,  15047,  15047,  15047,  15047,  15047,  15047,                |
275 |    15047,  15047,  15047,  15047,  15047,  15047,  15047,  15047,                |
276 |    15047,  15047,  15047,  15047,  15047,  15047,  15047,  15047,                |
277 |    15047,  15047,  15047,  15047,  15047,  15047,  15047,  15047,                |
278 |    15048,  15048,  15048,  15048,  15048,  15048,  15048,  15048,                |
279 |    15048,  15048,  15048,  15048,  15048,  15048,  15048,  15048,                |
280 |    15048,  15048,  15048,  15048,  15048,  15048,  15048,  15048,                |
281 |    15048,  15048,  15048,  15048,  15048,  15065,  15065,  15065,                |
282 |    15065,  15065,  15065,  15065,  15065,  15065,  15065,  15065,                |
283 |    15065,  15065,  15065,  15065,  15065,  15065,  15065,  15065,                |
284 |    15065,  15065,  15065,  15065,  15065,  15065,  15058,  15058,                |
285 |    15058,  15058,  15058,  15058,  15062,  15062,  15062,  15062,                |
286 |    15062,  15062,  15062,  15062,  15062,  15062,  15062,  15062,                |
287 |    15062,  15062,  15062,  15062,  15062,  15062,  15062,  15062,                |
288 |    15062,  15062,  15062,  15062,  15062,  15062,  15062,  15062,                |
289 |    15062,  15062,  15062,  15062,  15062,  15062,  15062,  15062,                |
290 |    15062,  15062,  15071,  15071,  15071,  15071,  15071,  15071,                |
291 |    15071,  15071,  15071,  15071,  15071,  15071,  15071,  15071,                |
292 |    15071,  15071,  15071,  15071,  15071,  15071,  15071,  15071,                |
293 |    15071,  15071,  15071,  15071,  15071,  15071,  15071,  15071,                |
294 |    15071,  15071,  15071,  15071,  15071,  15071,  15071,  15071,                |
295 |    15071,  15071,  15071,  15071,  15071,  15073,  15073,  15073,                |
296 |    15073,  15073,  15073,  15073,  15073,  15073,  15073,  15073,                |
297 |    15073,  15073,  15073,  15073,  15073,  15073,  15073,  15073,                |
298 |    15073,  15073,  15073,  15073,  15073,  15073,  15073,  15084,                |
299 |    15084,  15084,  15084,  15084,  15084,  15084,  15084,  15084,                |
300 |    15084,  15084,  15084,  15084,  15084,  15084,  15084,  15084,                |
301 |    15084,  15084,  15084,  15084,  15084,  15084,  15084,  15084,                |
302 |    15088,  15088,  15088,  15088,  15088,  15088,  15088,  15088,                |
303 |    15088,  15088,  15088,  15088,  15088,  15088,  15088,  15088,                |
304 |    15088,  15088,  15088,  15088,  15088,  15088,  15088,  15088,                |
305 |    15095,  15095,  15095,  15095,  15095,  15095,  15095,  15095,                |
306 |    15095,  15095,  15095,  15095,  15095,  15095,  15095,  15095,                |
307 |    15095,  15095,  15095,  15095,  15095,  15095,  15095,  15095,                |
308 |    15095,  15095,  15095,  15095,  15095,  15095,  15095,  15095,                |
309 |    15095,  15095,  15095,  15095,  15095,  15095,  15095,  15095,                |
310 |    15095,  15095,  15095,  15095,  15095,  15095,  15095,  15095,                |
311 |    15095,  15095,  15095,  15095,  15095,  15093,  15093,  15093,                |
312 |    15093,  15093,  15093,  15093,  15093,  15093,  15093,  15093,                |
313 |    15093,  15093,  15093,  15093,  15093,  15093,  15093,  15093,                |
314 |    15093,  15093,  15093,  15093,  15103,  15103,  15103,  15103,                |
315 |    15103,  15103,  15103,  15103,  15103,  15103,  15103,  15103,                |
316 |    15103,  15103,  15103,  15103,  15103,  15103,  15103,  15103,                |
317 |    15103,  15103,  15103,  15103,  15103,  15103,  15103,  15103,                |
318 |    15102,  15102,  15102,  15102,  15102,  15102,  15102,  15102,                |
319 |    15102,  15102,  15102,  15102,  15102,  15102,  15102,  15102,                |
```

```
320 |    15102,  15102,  15102,  15105,  15105,  15105,  15105,  15105,                |
321 |    15105,  15105,  15105,  15105,  15105,  15105,  15105,  15105,                |
322 |    15105,  15105,  15105,  15105,  15105,  15105,  15105,  15105,                |
323 |    15105,  15105,  15120,  15120,  15120,  15120,  15120,  15120,                |
324 |    15120,  15120,  15120,  15120,  15120,  15120,  15120,  15120,                |
325 |    15120,  15120,  15120,  15120,  15120,  15120,  15120,  15120,                |
326 |    15120,  15120,  15120,  15120,  15120,  15120,  15120,  15120,                |
327 |    15120,  15120,  15120,  15120,  15129,  15129,  15129,  15129,                |
328 |    15129,  15129,  15129,  15129,  15129,  15129,  15129,  15129,                |
329 |    15130,  15130,  15130,  15130,  15130,  15130,  15130,  15130,                |
330 |    15136,  15136,  15136,  15136,  15136,  15136,  15136,  15136,                |
331 |    15136,  15136,  15136,  15136,  15136,  15136,  15136,  15136,                |
332 |    15136,  15136,  15136,  15136,  15136,  15136,  15136,  15149,                |
333 |    15149,  15149,  15149,  15149,  15149,  15149,  15149,  15149,                |
334 |    15149,  15149,  15149,  15149,  15149,  15149,  15149,  15149,                |
335 |    15149,  15142,  15148,  15148,  15148,  15148,  15148,  15148,                |
336 |    15148,  15148,  15148,  15148,  15148,  15148,  15148,  15148,                |
337 |    15148,  15148,  15148,  15148,  15164,  15164,  15164,  15164,                |
338 |    15164,  15164,  15164,  15164,  15164,  15164,  15164,  15164,                |
339 |    15164,  15164,  15164,  15164,  15164,  15164,  15164,  15164,                |
340 |    15157,  15157,  15157,  15157,  15157,  15157,  15170,  15170,                |
341 |    15170,  15170,  15170,  15170,  15170,  15170,  15170,  15170,                |
342 |    15170,  15175,  15175,  15175,  15175,  15175,  15175,  15175,                |
343 |    15175,  15175,  15175,  15175,  15175,  15175,  15175,  15175,                |
344 |    15175,  15175,  15175,  15175,  15175,  15175,  15175,  15175,                |
345 |    15175,  15184,  15184,  15184,  15184,  15184,  15191,  15191,                |
346 |    15191,  15191,  15191,  15191,  15191,  15191,  15191,  15191,                |
347 |    15191,  15195,  15195,  15195,  15195,  15195,  15195,  15195,                |
348 |    15195,  15195,  15195,  15195,  15205,  15205,  15205,  15205,                |
349 |    15205,  15211,  15211,  15211,  15211,  15207,  15207,  15207,                |
350 |    15207,  15207,  15207,  15207,  15207,  15207,  15207,  15207,                |
351 |    15207,  15215,  15215,  15215,  15215,  15215,  15215,  15215,                |
352 |    15215,  15215,  15215,  15215,  15215,  15222,  15222,  15222,                |
353 |    15222,  15222,  15222,  15222,  15222,  15222,  15229,  15229,                |
354 |    15229,  15233,  15233,  15233,  15233,  15233,  15233,  15233,                |
355 |    15233,  15233,  15232,  15232,  15232,  15232,  15232,  15232,                |
356 |    15232,  15232,  15239,  15239,  15239,  15239,  15239,  15239,                |
357 |    15239,  15255,  15255,  15255,  15255,  15255,  15255,  15255,                |
358 |    15255,  15255,  15253,  15253,  15253,  15253,  15253,  15253,                |
359 |    15259,  15259,  15259,  15259,  15259,  15259,  15259,  15259,                |
360 |    15264,  15264,  15264,  15264,  15264,  15264,  15273,  15273,                |
361 |    15273,  15273,  15270,  15270,  15270,  15270,  15270,  15270,                |
362 |    15270,  15270,  15270,  15270,  15282,  15282,  15282,  15282,                |
363 |    15282,  15282,  15282,  15282,  15282,  15282,  15282,  15290,                |
364 |    15290,  15290,  15288,  15288,  15288,  15288,  15288,  15288,                |
365 |    15288,  15299,  15299,  15299,  15299,  15299,  15299,  15305,                |
366 |    15305,  15305,  15305,  15305,  15305,  15305,  15305,  15308,                |
367 |    15308,  15308,  15308,  15308,  15308,  15308,  15308,  15313,                |
368 |    15313,  15313,  15313,  15313,  15321,  15321,  15321,  15321,                |
369 |    15321,  15321,  15321,  15321,  15321,  15331,  15331,  15331,                |
370 |    15331,  15331,  15331,  15331,  15331,  15330,  15330,  15332,  15332,       |
371 |    15332,  15332,  15332,  15332,  15332,  15337,  15337,  15337,                |
372 |    15337,  15337,  15345,  15345,  15345,  15345,  15345,  15355,                |
373 |    15355,  15355,  15355,  15355,  15355,  15355,  15358,  15358,                |
374 |    15358,  15358,  15358,  15358,  15358,  15361,  15361,  15361,                |
375 |    15361,  15361,  15361,  15361,  15361,  15380,  15380,  15380,                |
376 |    15380,  15376,  15376,  15375,  15375,  15375,  15375,  15375,                |
377 |    15386,  15386,  15386,  15386,  15386,  15386,  15392,  15392,  15392,       |
378 |    15392,  15392,  15392,  15399,  15399,  15399,  15399,  15398,                |
379 |    15402,  15402,  15402,  15402,  15402,  15409,  15409,  15409,                |
380 |    15409,  15409,  15414,  15414,  15416,  15416,  15416,  15416,                |
381 |    15416,  15416,  15416,  15426,  15426,  15426,  15426,  15426,                |
382 |    15426,  15426,  15430,  15430,  15430,  15434,  15434,  15434,                |
383 |    15434,  15439,  15439,  15439,  15439,  15439,  15450,  15450,                |
384 |    15450,  15450,  15450,  15450,  15450,  15450,  15453,  15453,                |
385 |    15453,  15454,  15452,  15452,  15452,  15464,  15464,  15464,                |
386 |    15464,  15464,  15464,  15470,  15470,  15470,  15470,  15470,                |
387 |    15470,  15470,  15477,  15477,  15477,  15477,  15477,  15477,                |
388 |    15485,  15485,  15485,  15485,  15485,  15496,  15496,  15496,                |
389 |    15496,  15496,  15498,  15498,  15498,  15498,  15498,  15498,                |
390 |    15498,  15514,  15514,  15514,  15514,  15514,  15513,  15513,                |
391 |    15513,  15513,  15526,  15526,  15526,  15526,  15528,  15528,                |
392 |    15528,  15537,  15537,  15537,  15537,  15537,  15537,  15537,                |
393 |    15543,  15543,  15543,  15543,  15543,  15543,  15554,  15554,                |
394 |    15560,  15560,  15560,  15557,  15566,  15566,  15566,  15571,                |
395 |    15571,  15571,  15571,  15571,  15581,  15581,  15581,  15581,                |
396 |    15581,  15586,  15586,  15586,  15583,  15583,  15594,  15594,                |
397 |    15594,  15598,  15598,  15607,  15607,  15607,  15607,  15609,                |
398 |    15609,  15609,  15609,  15609,  15617,  15617,  15625,  15625,                |
399 |    15625,  15632,  15632,  15635,  15635,  15635,  15644,  15644,                |
```

```
    15644,  15645,  15652,  15652,  15652,  15659,  15659,  15659,
    15659,  15668,  15668,  15675,  15675,  15675,  15682,  15682,
    15682,  15682,  15682,  15693,  15693,  15694,  15694,  15694,
    15709,  15709,  15709,  15709,  15709,  15716,  15716,  15711,
    15730,  15730,  15730,  15734,  15734,  15734,  15738,  15734,
    15734,  15734,  15751,  15751,  15751,  15757,  15758,  15763,
    15757,  15767,  15770,  15770,  15771,  15771,  15778,  15781,
    15781,  15781,  15798,  15798,  15798,  15791,  15791,  15799,
    15799,  15808,  15808,  15808,  15808,  15808,  15820,  15820,
    15830,  15833,  15833,  15834,  15834,  15841,  15841,  15841,
    15850,  15850,  15855,  15855,  15858,  15858,  15856,  15856,
    15856,  15887,  15887,  15887,  15877,  15877,  15898,  15898,
    15891,  15895,  15895,  15908,  15908,  15908,  15905,  15905,
    15916,  15916,  15926,  15926,  15926,  15941,  15941,  15941,
    15933,  15938,  15938,  15943,  15943,  15948,  15948,  15956,
    15955,  15955,  15963,  15963,  15963,  15976,  15976,  15976,
    15975,  15977,  15977,  15985,  15985,  15991,  15996,  15996,
    15998,  15998,  15998,  16004,  16004,  16004,  16015,  16015,
    16015,  16013,  16015,  16015,  16015,  16041,  16041,  16041,
    16037,  16037,  16037,  16041,  16041,  16048,  16048,  16048,
    16059,  16059,  16059,  16059,  16069,  16069,  16069,  16074,
    16074,  16078,  16078,  16081,  16081,  16092,  16092,  16086,
    16106,  16106,  16106,  16112,  16112,  16112,  16107,  16122,
    16122,  16122,  16129,  16129,  16129,  16135,  16135,  16145,
    16145,  16145,  16149,  16147,  16147,  16157,  16157,  16163,
    16163,  16174,  16166,  16173,  16173,  16173,  16184,  16184,
    16184,  16195,  16200,  16209,  16207,  16207,  16207,  16211,
    16211,  16220,  16220,  16226,  16231,  16231,  16236,  16236,
    16245,  16248,  16250,  16250,  16259,  16259,  16259,  16266,
    16266,  16277,  16277,  16282,  16282,  16282,  16291,  16295,
    16298,  16298,  16306,  16307,  16318,  16318,  16321,  16317,
    16331,  16331,  16338,  16338,  16341,  16341,  16348,  16348,
    16359,  16359,  16364,  16364,  16375,  16375,  16379,  16384,
    16385,  16390,  16392,  16392,  16399,  16413,  16413,  16420,
    16420,  16420,  16430,  16423,  16436,  16441,  16437,  16446,
    16458,  16458,  16461,  16461,  16468,  16473,  16474,  16478,
    16478,  16492,  16495,  16498,  16502,  16502,  16515,  16519,
    16526,  16521,  16527,  16527,  16538,  16538,  16546,  16544,
    16544,  16561,  16561,  16561,  16577,  16576,  16589,  16589,
    16590,  16602,  16602,  16601,  16601,  16618,  16626,  16626,
    16619,  16639,  16635,  16635,  16658,  16654,  16654,  16661,
    16666,  16666,  16678,  16682,  16682,  16692,  16692,  16702,
    16706,  16706,  16716,  16719,  16724,  16737,  16737,  16736,
    16743,  16752,  16755,  16758,  16774,  16774,  16784,  16784,
    16779,  16801,  16801,  16796,  16808,  16813,  16826,  16819,
    16819,  16835,  16834,  16843,  16843,  16847,  16855,  16854,
    16863,  16863,  16877,  16876,  16881,  16881,  16898,  16894,
    16894,  16911,  16911,  16907,  16926,  16926,  16921,  16945,
    16945,  16936,  16939,  16943,  16943,  16970,  16959,  16969,
    16969,  16976,  16983,  16984,  17000,  17001,  17001,  17016,
    17016,  17020,  17020,  17020,  17040,  17030,  17041,  17041,
    17051,  17050,  17058,  17058,  17076,  17074,  17074,  17088,
    17088,  17098,  17090,  17105,  17105,  17116,  17125,  17125,
    17137,  17135,  17153,  17152,  17165,  17154,  17174,  17174,
    17191,  17191,  17192,  17195,  17211,  17209,  17224,  17222,
    17222,  17249,  17259,  17263,  17252,  17279,  17270,  17276,
    17284,  17293,  17307,  17307,  17317,  17330,  17334,  17339,
    17327,  17348,  17357,  17359,  17379,  17366,  17376,  17388,
    17393,  17393,  17409,  17409,  17412,  17417,  17417,  17436,
    17430,  17445,  17445,  17460,  17453,  17462,  17475,  17475,
    17488,  17488,  17502,  17495,  17509,  17516,  17516,  17526,
    17535,  17545,  17543,  17544,  17552,  17556,  17563,  17573,
    17576,  17583,  17584,  17601,  17601,  17616,  17616,  17627,
    17627,  17635,  17631,  17631,  17654,  17658,  17673,  17679,
    17692,  17686,  17687,  17692,  17720,  17713,  17724,  17748,
    17734,  17748,  17765,  17766,  17770,  17782,  17793,  17795,
    17808,  17815,  17823,  17830,  17846,  17837,  17850,  17861,
    17866,  17878,  17878,  17891,  17898,  17918,  17918,  17929,
    17934,  17951,  17948,  17958,  17956,  17965,  17980,  17980,
    18000,  18000,  18014,  18020,  18030,  18034,  18041,  18044,
    18055,  18062,  18070,  18078,  18081,  18096,  18103,  18113,
    18127,  18138,  18127,  18147,  18145,  18165,  18180,  18166,
    18180,  18191,  18214,  18201,  18210,  18236,  18246,  18243,
    18256,  18275,  18284,  18300,  18293,  18313,  18324,  18325,
    18331,  18348,  18357,  18372,  18375,  18405,  18398,  18405,
    18413,  18441,  18436,  18459,  18472,  18472,  18480,  18491,
    18501,  18512,  18537,  18531,  18542,  18553,  18567,  18573,
    18584,  18587,  18599,  18598,  18614,  18624,  18636,  18649,
    18655,  18649,  18666,  18671,  18683,  18695,  18714,  18717,
    18737,  18737,  18733,  18752,  18772,  18761,  18786,  18792,
```

```
    18801,  18813,  18815,  18837,  18839,  18853,  18853,  18864,
    18874,  18877,  18901,  18905,  18909,  18941,  18941,  18938,
    18955,  18963,  18966,  18982,  19000,  19000,  19013,  19027,
    19034,  19041,  19051,  19066,  19084,  19076,  19095,  19119,
    19108,  19126,  19126,  19147,  19147,  19161,  19148,  19170,
    19171,  19202,  19208,  19202,  19214,  19239,  19225,  19245,
    19255,  19255,  19280,  19309,  19290,  19304,  19308,  19330,
    19332,  19358,  19345,  19376,  19376,  19392,  19402,  19413,
    19409,  19429,  19432,  19458,  19482,  19484,  19489,  19509,
    19511,  19516,  19538,  19540,  19554,  19559,  19573,  19589,
    19612,  19615,  19619,  19627,  19648,  19652,  19659,  19670,
    19684,  19702,  19710,  19729,  19720,  19741,  19747,  19765,
    19785,  19795,  19795,  19802,  19823,  19836,  19836,  19842,
    19853,  19866,  19880,  19895,  19911,  19919,  19945,  19952,
    19955,  19963,  19996,  19982,  19991,  20017,  20020,  20034,
    20053,  20067,  20085,  20092,  20086,  20102,  20108,  20127,
    20143,  20152,  20146,  20181,  20187,  20195,  20191,  20213,
    20242,  20266,  20255,  20277,  20285,  20286,  20316,  20334,
    20354,  20325,  20377,  20396,  20412,  20403,  20441,  20455,
    20466,  20466,  20481,  20502,  20534,  20527,  20552,  20574,
    20600,  20606,  20618,  20623,  20634,  20669,  20670,  20684,
    20705,  20724,  20727,  20760,  20762,  20773,  20779,  20812,
    20842,  20839,  20867,  20897,  20909,  20927,  20955,  20977,
    21000,  21013,  21024,  21058,  21082,  21095,  21109,  21144,
    21170,  21163,  21180,  21210,  21240,  21265,  21275,  21286,
    21344,  21323,  21372,  21429,  21441,  21438,  21497,  21488,
    21530,  21589,  21608,  21664,  21635,  21688,  21730,  21755,
    21788,  21794,  21809,  21851,  21856,  21939,  21934,  22021,
    21991,  22063,  22116,  22148,  22198,  22212,  22291,  22293,
    22338,  22378,  22423,  22462,  22505,  22509,  22559,  22594,
    22631,  22630,  22702,  22738,  22778,  22863,  22891,  22898,
    22985,  23006,  23045,  23116,  23125,  23197,  23228,  23270,
    23333,  23366,  23477,  23537,  23504,  23645,  23604,  23674,
    23738,  23801,  23836,  23879,  23922,  23984,  24067,  24123,
    24044,  24130,  24226,  24209,  24263,  24347,  24437,  24390,
    24608,  24527,  24601,  24712,  24754,  24812,  24866,  24994,
    25034,  25004,  25122,  25263,  25211,  25374,  25402,  25445,
    25606,  25639,  25616,  25767,  25787,  25853,  25934,  25912,
    26116,  26177,  26211,  26317,  26400,  26444,  26497,  26584,
    26840,  26870,  26970,  27009,  27064,  27246,  27293,  27368,
    27487,  27475,  27577,  27649,  27762,  27854,  28013,  27986,
    28176,  28374,  28318,  28359,  28523,  28730,  28846,  28927,
    28927,  29014,  29249,  29375,  29465,  29620,  29665,  29808,
    29809,  29818,  29818,  29818,  29818,  29818,  29818,  29818,
    29818,  29818,  29818,  29818,  29818,  29818,  29818,  29818,
    29818,  29818,  29818,  29818,  29818,  29818,  29818,  29818,
    29818,  29818,  29818,  29818,  29818,  29818,  29818,  29818
    };
#endif
```

```
1  /*******************************************************************************/
2  /*  Autor:          Robert Ostermann                                           */
3  /*  Datum:          01.09.2011 - 24.02.2012                                    */
4  /*  Controller:     MSP430f1611 => "Regelcontroller"                           */
5  /*                                                                             */
6  /*  Datei:          offset_sin.h                                               */
7  /*  Beschreibung:   Lookup-Tabelle der Sinusfunktion                           */
8  /*                                                                             */
9  /*******************************************************************************/
10
11 #ifndef OFFSET_SIN_H
12 #define OFFSET_SIN_H
13
14 const unsigned int offset_sin[65] = {
15     32768,  32758,  32729,  32679,  32610,
16     32522,  32413,  32286,  32138,  31972,
17     31786,  31581,  31357,  31114,  30853,
18     30572,  30274,  29957,  29622,  29269,
19     28899,  28511,  28106,  27684,  27246,
20     26791,  26320,  25833,  25330,  24812,
21     24279,  23732,  23170,  22595,  22006,
22     21403,  20788,  20160,  19520,  18868,
23     18205,  17531,  16846,  16151,  15447,
24     14733,  14010,  13279,  12540,  11793,
25     11039,  10279,   9512,   8740,   7962,
26      7180,   6393,   5602,   4808,   4011,
27      3212,   2411,   1608,    804,      0
28     };
29 #endif
```

## D.2 Matlab

### D.2.1 Simulation stückweise lineare Approximation

```
%-------------------------------------------------------------------------
% Skript zur Simulation der approximierten logarithmischen Funktion
%
% Datei: log_approximation.m
%
% Erstellt von: Robert Ostermann
% Datum: 01.09.2011 - 24.02.2012
%-------------------------------------------------------------------------

clear all;
clc;

% Sytemparameter
Ulimit = 1;    % Sättigung der Verstärker
N = 8;         % Anzahl Stufen
v = 2;         % Verstärkung der einzelnen Stufen

% Berechnung der Eingangssignale bei denen die einzelnen
% Verstärker gerade in die Begrenzung gehen
k = 1:1:N;
Ut_k = Ulimit./((v-1).*v.^(k-1));

% Berechnung der Ausgangssignale in den Knickpunkten
Ua_k = Ulimit/(v-1) + N*Ulimit - Ulimit*log(Ulimit)/log(v) ...
        + Ulimit*log(v-1)/log(v);
Ua = Ua_k + Ulimit/log(v)*log(Ut_k);

% Darstellung der "logarithmischen Kennline"
figure(1);
hold off;
semilogx(Ut_k,Ua,'-r*');
grid on;
hold on;

% Stückweise lineare Approximation
Ue = 0.000:0.0001:Ut_k(1);
Ua = Ue;
Ua = Ua + (Ue*(v-1)>=Ulimit)*Ulimit ...
        + (Ue*(v-1)<Ulimit).*(Ue*(v-1));
for i=1:1:N-1
    Ua = Ua + (Ue*(v-1)*v^i>=Ulimit)*Ulimit ...
            + (Ue*(v-1)*v^i<Ulimit).*(Ue*(v-1)*v^i);
end;

figure(1);
semilogx(Ue,Ua);
set(gca,'fontsize',16);
title('Stückweise lineare Approximation');
xlabel('Ue/V');
ylabel('Ua/V');
legend('ideal','approximiert','Location','NorthWest');
```

### D.2.2 Simulation Kennlinie

```
%-------------------------------------------------------------------------
% Skript zur Simulation der Kennlinie des Logarithmierers
%
% Datei: funktion_logamp.m
%
% Erstellt von: Robert Ostermann
% Datum: 01.09.2011 - 24.02.2012
%-------------------------------------------------------------------------

clear all;
clc;

figure(1);
hold off;

% Funktionsgraph zeichnen
in = -1:0.00001:+1;
plot(in,logamp_fnc(in),'k');
hold on;
grid on;

% Eingangssignal
t = 0:0.001:2*pi;
Uin = 0.6*sin(t);
plot(Uin,-t-3*pi,'k');

% Ausgangssignal (logarithmiert)
signal = logamp_fnc(Uin);
plot(t/7+1,signal,'k');

% Achsenkreuz
plot([-1 2],[0 0],'k');     % x
plot([0 0],[-20 10],'k');   % y

% Horizontal Linien
plot([min(Uin) 2],[min(signal) min(signal)],'--k');
plot([max(Uin) 2],[max(signal) max(signal)],'--k');

% Vertikale Linien
plot([min(Uin) min(Uin)],[min(signal) -20],'--k');
plot([max(Uin) max(Uin)],[max(signal) -20],'--k');

% Achsenbeschriftung
set(gca,'fontsize',16);
title('verschobener Logarithmierer');
xlabel('Ue/V');
ylabel('Ua/V');
```

```
%-------------------------------------------------------------------------
% Funktion zur Simulation des logarithmischen Verstärkers
%
% Datei: logamp_fnc.m
%
% Erstellt von: Robert Ostermann
% Datum: 01.09.2011 - 24.02.2012
%-------------------------------------------------------------------------

function [ Uout ] = logamp_fnc( Uin )

    Ulimit = [1 1 1 1 1 1 1];

    prefix = sign(Uin); % vorzeichen speichern
    Uin = abs(Uin);

    Uout = 2*Uin;

    for i=1:1:7
        Uout = Uout + (((Uin*2^i)>=Ulimit(i))*Ulimit(i))...
                    + (((Uin*2^i)<Ulimit(i)).*Uin*2^i);
    end;

    Uout = prefix.*Uout;
end
```

### D.2.3 Simulation Phasenverschiebung

```
%-------------------------------------------------------------------------
% Simulation zur Addition phasenverschoben und begrenzter Signale
%
% Datei: phase.m
%
% Erstellt von: Robert Ostermann
% Datum: 01.09.2011 - 24.02.2012
%-------------------------------------------------------------------------

clc;
clear all;

limit = 1;
gain = 2;

t = 0:0.001:1.5;
a = gain.^(0:1:5);

for i=1:1:6
    in(:,i) = a(i)*sin(2*pi*t);
    out(:,i) = a(i)*sin(2*pi*t-log2(a(i))*pi/180*5);  % 5° verschiebung pro stufe
end;

% limitieren des Signals
out = out.*(out<limit) + limit*(out>=limit);
out = out.*(out>-limit) - limit*(out<=-limit);
in = in.*(in<limit) + limit*(in>=limit);
in = in.*(in>-limit) - limit*(in<=-limit);

set(gca,'fontsize',16);
hold off;
plot(t,[sum(in'); sum(out')]);
grid on;

title(['Vergleich Addition, mit und ohne Phasenverschiebung'...
        ' / 5° Verschiebung pro Stufe (5 Stufen)']);
xlabel('t');
ylabel('Uout/V');
legend('nicht Phasenverschoben','Phasenverschoben','Location','SouthEast');
export_fig(gcf,'Vergleich_Phasenverschiebung.pdf','A4L');
```

### D.2.4 Lookup-Tabelle Delogarithmierung

```
%-------------------------------------------------------------------------
% Skript zum erstellen der Lookup-Tabelle für die Delogarithmierung
% der gesampelten Werte auf dem Regelcontroller
%
% Datei: delog_lookup.m
%
% Erstellt von: Robert Ostermann
% Datum: 01.09.2011 - 24.02.2012
%-------------------------------------------------------------------------

clear all;
clc;

D = 0:1:4095;
Uin = 0:2.5/length(D):2.5-2.5/length(D);

% Messergebnisse laden
load('messung_logamp_in_out_2.mat');

for i=1:1:length(Uin)
    temp = abs(ch4 - Uin(i));   % differenz zwischen real und optimal bilden
    [C,I] = min(temp);          % Minima bestimmen
    table(i) = ch3(I);          % realwert speichern
end;

fp = fopen('delog_table.h','w');
fprintf(fp,['#ifndef DELOG_TABLE_H\n'...
            '#define DELOG_TABLE_H\n\n'...
            'const unsigned int delog_table[4096] = {']);

for i=1:1:length(D)-1
    if mod(i-1,8) == 0
        fprintf(fp,'\n');
    end;
    fprintf(fp,'\t%5.0f,',table(i)*10000);
end;

fprintf(fp,'\t%5.0f\n',table(length(D))*10000);
fprintf(fp,'\t};\n');
fprintf(fp,'#endif\n');

fclose(fp);
```

### D.2.5 Lookup-Tabelle Sinusfunktion

```
%-------------------------------------------------------------------------
% Skript zum erstellen der Sinus-Lookup-Tabelle für die Offsetberechnung
%
% Datei: sin_lookup.m
%
% Erstellt von: Robert Ostermann
% Datum: 01.09.2011 - 24.02.2012
%-------------------------------------------------------------------------

clear all;
clc;

Duty = 0:1:64;

sin_offset = round(-32768*sin(pi.*Duty./128-pi/2));

fp = fopen('offset_sin.h','w');
fprintf(fp,['#ifndef OFFSET_SIN_H\n'...
            '#define OFFSET_SIN_H\n\n'...
            'const unsigned int offset_sin['...
            num2str(length(Duty)) '] = {']);

for i=1:1:length(Duty)-1
    if mod(i-1,5) == 0
        fprintf(fp,'\n');
    end;
    fprintf(fp,'\t%5.0f,',sin_offset(i));
end;

fprintf(fp,'\t%5.0f\n',sin_offset(length(Duty)));
fprintf(fp,'\t};\n');
fprintf(fp,'#endif\n');

fclose(fp);
```

# Abkürzungsverzeichnis

**ABS** . . . . . Antiblockiersystem

**ADC** . . . . . Analog-Digital-Umsetzer

**AMR** . . . . . Anisotrope Magnetoresistive

**DAC** . . . . . Digital-Analog-Umsetzer

**DVGA** . . . . . Digital Variable Gain Amplifier

**GWBP** . . . . . Verstärkungs-Bandbreite-Produkt

**ISR** . . . . . Interrupt Service Routine

**OP** . . . . . Operationsverstärker

**THD** . . . . . Total Harmonic Distortion

# Tabellenverzeichnis

# Abbildungsverzeichnis

# Versicherung über die Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §16(5) APSO-TI-BM ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen habe ich unter Angabe der Quellen kenntlich gemacht.

| Hamburg, 24. Februar 2012 | |
|---|---|
| Ort, Datum | Unterschrift |