Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Simon Langhof

Entwicklung einer mikrocontrollergesteuerten Spannungsregelung auf Basis von Step-Up- und Step-Down-Wandlern

Simon Langhof

Entwicklung einer mikrocontrollergesteuerten Spannungsregelung auf Basis von Step-Up- und Step-Down-Wandlern

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Technische Informatik
am Fachbereich Elektrotechnik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Klemke
Zweitgutachter: Prof. Dr. Canzler

Abgegeben am 29.02.2008

Simon Langhof

Thema der Bachelorarbeit

Entwicklung einer mikrocontrollergesteuerten Spannungsregelung auf Basis von Step-Up- und Step-Down-Wandlern

Stichworte

Step-Up-Wandler, Step-Down-Wandler, Schaltregler, AVR Mikrocontroller

Kurzzusammenfassung

Es wird ein Zweifach-Schaltregler entworfen, der durch einen Mikrocontroller geregelt aus einer Eingangsspannung von 24 V mit einem Step-Up-Wandler eine Ausgangsspannung von 48 V sowie mit einem Step-Down-Wandler eine Ausgangsspannung von 12 V erzeugt. Desweiteren wird die Software, die auf dem Mikrocontroller läuft entwickelt und das Verhalten der Schaltung in verschiedenen Betriebszuständen analysiert.

Simon Langhof

Title of the paper

Development of a microcontroller regulated power supply based on Step-Up- and Step-Down-Converters

Keywords

Step-Up-Converter, Step-Down-Converter, Switching regulator, AVR Microcontroller

Abstract

A switching regulator with two output voltages, regulated by a microcontroller is designed, which converts an input voltage of 24 V into 48 V using a step-up-converter and into 12 V using a step-down-converter. The software running on the microcontroller is developed and the circuit's behavior is analysed.

Inhaltsverzeichnis

1 Einleitung

In vielen Bereichen ist es nötig, aus einer vorhandenen Spannung eine andere Spannung zu erzeugen. Dabei kann grob unterschieden werden, ob es sich jeweils bei Ein- und Ausgangsspannung um Gleich- oder Wechselspannung handelt, sowie das Verhältnis von Eingangsspannung zu Ausgangsspannung, also ob die Ausgangsspannung kleiner oder größer als die Eingangsspannung sein soll. Für jede dieser Umwandlungen haben sich Schaltungen durchgesetzt. Beispielsweise sind Spannungswandler von 12 V Gleichspannung auf 230 V Wechselspannung zum Einsatz in Kraftfahrzeugen in der Größe einer Getränkedose erhältlich, die den Betrieb eines Notebooks ohne KFZ-Netzteil ermöglichen. Der Wirkungsgrad einer solchen zweifachen Wandlung ($12\,\mathrm{V} = \rightarrow 230\,\mathrm{V} \sim \rightarrow \mathrm{z.B.}\,19\,\mathrm{V} =$) ist aber schlechter als bei nur einer Spannungswandlung.
Für die Ansteuerung von Schrittmotoren ist eine hohe Betriebsspannung von Vorteil, da dadurch eine höhrere Drehzahl erreicht werden kann. In den meisten industriellen Anlagen steht eine Versorgungs(gleich)spannung von 24 V zur Verfügung, während einige Schrittmotortreiber auch mit deutlich höheren Spannungen betrieben werden können. Es wäre also wünschenswert, aus der vorhandenen Gleichspannung eine höhere zu erzeugen, um die Drehgeschwindigkeit der Motoren zu erhöhen.
Aufgrund der Nachteile von herkömmlichen Netzteilen sind für diesen Anwendungsbereich Schaltregler sinnvoll. Schaltregler mit der nötigen Ausgangsleistung sind aber nicht als komplett integrierte Schaltungen verfügbar, was den Einsatz eines speziellen Reglers oder die komplette Entwicklung eines Reglers erforderlich macht.
Aufgabe dieser Bachelorarbeit ist die Dimensionierung und Umsetzung eines Step-Up-Wandlers (DC/DC-Wandler) mit Mikrocontrollerregelung um eine Ausgangsspannung von 48 V aus der vorhandenen Versorgungsspannung von 24 V zu erzeugen. Es soll dann am Wandler die Leistungsfähigkeit untersucht werden. Desweiteren soll untersucht werden, ob es möglich ist, zusätzlich mit Step-Down-Wandlern Spannungen von 12 V sowie 3,3 V zu erzeugen. Diese Wandler sollen dann nach Möglichkeit ebenfalls umgesetzt und untersucht werden.

1.1 Gliederung

Zunächst werden in Kapitel 2.1 einige Möglichkeiten der Spannungswandlung aufgezeigt. Kapitel 2.2 gibt eine Einführung in die verschiedenen Arten von Schaltreglern. Die Funktionsweise eines Step-Up-Wandlers sowie die Auswirkungen durch Veränderung einzelner Parameter an der Schaltung wird in Kapitel 2.3 durch Simulationen mit Spice aufgezeigt.
Für die Regelung der Wandler bestehen verschiedene Möglichkeiten, die in Kapitel 2.4 kurz aufgezeigt werden. Für diese Arbeit ist die Regelung mit einem Mikrocontroller vorgegeben.
Die umgesetzte Schaltung, die in Kapitel 3.1 beschrieben wird, besteht neben dem Step-Up-Wandler mit 48 V Ausgangsspannung noch aus einem Step-Down-Wandler, der aus der Eingangsspannung eine Ausgangsspannung von 12 V erzeugt. Ein weiterer geplanter Step-Down-Wandler mit einer Ausgangsspannung von 3,3 V für die Versorgung der Schrittmotorcontrollerschaltung wurde nicht realisiert.
In Kapitel 3.2 werden die Grundlagen zur Programmierung der Mikrocontrollerkomponenten gezeigt und in Kapitel 3.3 der komplette umgesetzte Mikrocontrollercode erläutert. Beide Wandler der Schaltung werden in Kapitel 4 auf das Verhalten im Betrieb, beim Ein- bzw. Ausschalten und bei Lastwechseln untersucht. Weiterhin wird der zulässige Eingangsspannungsbereich ermittelt. Abschließend wird noch kurz die EMV-Problematik betrachtet.

2 Grundlagen

2.1 Möglichkeiten der Spannungswandlung

Die in dieser Arbeit behandelten Step-Up- und Step-Down-Wandler sind sogenannte sekundärgetaktete Schaltregler, bei denen in der Regel die komplette Wandlung auf der Sekundärseite eines Netztransformators geschieht. Entgegen der Herkunft des Namens ist auch eine Versorgung aus Batterien oder Solarzellen, also ohne Primär- und Sekundärseite eines Transformators denkbar. Es handelt sich bei diesen Schaltreglern um Gleichspannungswandler mit einem relativ geringen Schaltungsaufwand und einem hohen Wirkungsgrad, der mehr als 90 % erreichen kann. Grundsätzlich bestehen sie aus einem steuerbaren Wechselschalter, einer Induktivität und einem Kondensator, wobei der Wechselschalter in der Regel durch einen MOSFET und eine Diode ersetzt wird. Zusätzlich ist noch eine Regelschaltung erforderlich, die den Schalter bzw. Transistor in Abhängigkeit von der Soll- und der Ist-Ausgangsspannung ansteuert.
Außerdem gibt es noch primärgetaktete Schaltregler, bei denen die Netzspannung gleichgerichtet und durch einen Leistungsschalter getaktet auf einen Hochfrequenztransformator geführt wird. Primärgetaktete Schaltregler kommen beispielsweise in Computernetzteilen zum Einsatz.
Für kleine Leistungen und kompakte Baugrößen, bei denen keine galvanische Isolierung zwischen Eingang und Ausgang benötigt wird, sind sekundärgetaktete Schaltregler von Vorteil. In „Schaltnetzteile in der Praxis“ (Kilgenstein, 1986) werden als Grenzen für den Einsatz sekundärgetakteter Schaltregler (Durchflußwandler) 50 V und 50 W genannt. Höhere Spannungen sind der Einsatzbereich für sogenannte Sperrwandler, höhere Leistungen für sogennante Eintaktdurchfluß- oder Gegentaktdurchflußwandler bzw. Halb- oder Vollbbrücken-Gegentaktflußwandler.
Herkömmliche Netzteile mit Transformatoren besitzen verschiedene Nachteile gegenüber Schaltreglern bzw. Schaltnetzteilen mit gleicher Leistung:

- Die benötigten Transformatoren sind größer und schwerer
- Der Wirkungsgrad ist schlechter
- Es wird eine Wechselspannung benötigt, die unter Umständen erst erzeugt werden muss

Auch Schaltregler haben Nachteile gegenüber herkömmlichen Trafonetzteilen:

- Die Schaltungskomplexität ist höher
- Der Preis ist in der Regel höher, da für große Ausgangsleistungen spezielle leistungsfähige Bauteile nötig sind
- „They used to have a bad reputation for reliability, with occasional spectacular pyrotechnic displays during episodes of catastrophic failure.“ (Horowitz und Hill, 1998, S. 356)

Als reine DC/DC-Wandler lassen sich auch Linearregler einsetzen, die allerdings nicht die Möglichkeit bieten, die Spannung zu erhöhen. Wenn die Spannung verringert werden soll, können sie zwar verwendet werden, ihr Nachteil ist aber ihr schlechter Wirkungsgrad, da die überschüssige Leistung in Wärme umgesetzt wird. Daher benötigen Linearregler für höhere Leistungen auch eine gute Kühlung. Ein Linearregler der mit 24 V versorgt wird und eine Ausgangsspannung von 12 V bei einem Strom von 1 A liefert, hat einen rechnerischen Wirkungsgrad von 50 %, es werden also neben den 12 W Ausgangsleistung nochmal 12 W in Wärme umgesetzt, die vom Regler abgeführt werden muss.

2.2 Grundschaltungen sekundärgetakteter Schaltregler

Dieser Abschnitt behandelt die verschiedenen Arten der sekundärgetakteten Schaltregler bzw. deren Wandlerteil ohne Regelung. Der Step-Up- und der Step-Down-Wandler werden auch in der später umgesetzten Schaltung verwendet.

2.2.1 Step-Up-Wandler

Ein Step-Up-Wandler (Abb. 2.1) wird verwendet, um aus einer gegebenen Gleichspannung eine höhere Gleichspannung zu erzeugen. Ohne Ansteuerung des MOSFETs liegt die Ausgangsspannung aufgrund des Spannungsabfalls an der Diode etwas unter der Eingangsspannung.
Bei leitendem MOSFET fließt ein hoher Strom durch die Induktivität. Dabei wird Energie in einem Magnetfeld gespeichert. Wenn dann der MOSFET sperrt, versucht die Induktivität durch Selbstinduktion diesen hohen Strom zu erhalten, wodurch die Induktivität wie eine Spannungsquelle wirkt, die mit dem Eingang in Reihe geschaltet ist. Daher addiert sich die induzierte Spannung zur Versorgungsspannung.

Das Verhältnis der Ausgangs- zur Eingangsspannung entspricht im theoretischen Idealfall dem Verhältnis der Periodendauer T der Ansteuerung zum Zeitraum, in dem der MOSFET T sperrt. Alternativ lässt sich das Verhältnis auch über das Tastverhältnis $D = \frac{t_{ein}}{T}$ berechnen.

$$\frac{U_a}{U_e} = \frac{t_{aus} + t_{ein}}{t_{aus}} = \frac{T}{t_{aus}} = \frac{1}{1 - D}$$

$$U_a = U_e \cdot \frac{T}{t_{aus}} = U_e \cdot \frac{1}{1 - D}$$

Wenn dem Speicherkondensator weniger Energie entnommen wird, als ihm durch die Schaltvorgänge zugeführt wird, steigt die Ausgangsspannung an und die Einschaltdauer muss verkürzt werden. Dieser Effekt tritt auch beim Step-Down-Wandler (siehe Abschnitt [2.2.2]) auf.
Wenn das Magnetfeld in der Induktivität vor dem Beginn des nächsten Ladevorgangs komplett abgebaut ist, kann eine Schaltperiode in drei Abschnitte unterteilt werden. Wie im Normalbetrieb wird die Induktivität erst geladen und anschließend wieder entladen, dann schließt sich noch ein dritter Abschnitt an, in dem kein Strom durch die Induktivität fließt. Dieser sogenannte lückende Stromfluss (englisch: Discontinuous Conduction Mode) ist oft durch ein hörbares Pfeifen der Induktivität wahrnehmbar.

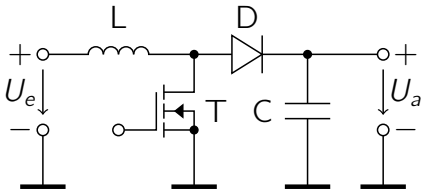


Abbildung 2.1: Prinzipschaltung eines Step-Up-Wandlers

2.2.2 Step-Down-Wandler

Mit einem Step-Down-Wandler (Abb. 2.2) wird aus einer Gleichspannung am Eingang eine niedrigere Gleichspannung am Ausgang erzeugt.
Leitet der MOSFET, fließt der Strom vom Eingang über die Induktivität und lädt den Kondensator auf bzw. steht einem Verbraucher am Ausgang zur Verfügung. Die Induktivität baut aufgrund des Stromflusses ein Magnetfeld auf, das beim Übergang des MOSFETs in den sperrenden Zustand durch Selbstinduktion einen Stromfluss zum Ausgang erzeugt, wobei die Spannung abnimmt. Durch den Kondensator werden die Spannungsschwankungen geglättet. Leitet der MOSFET dauerhaft, entspricht die Ausgangsspannung etwa der Eingangsspannung, nachdem der Kondensator C komplett geladen ist.

Das Tastverhältnis zu einem gegebenen Spannungsverhältnis bzw. das Spannungsverhältnis zu einem bekanntem Tastverhältnis lassen sich ähnlich wie beim Step-Up-Wandler berechnen:

$$\frac{U_a}{U_e} = \frac{t_{ein}}{t_{aus} + t_{ein}} = \frac{t_{ein}}{T} = D$$

$$U_a = U_e \cdot \frac{t_{ein}}{T} = U_e \cdot D$$

Bei unbelastetem Ausgang steigt auch hier die Spannung des Speicherkondensators an, so dass das Tastverhältnis verringert werden muss und der Wandler im lückenden Betrieb läuft.

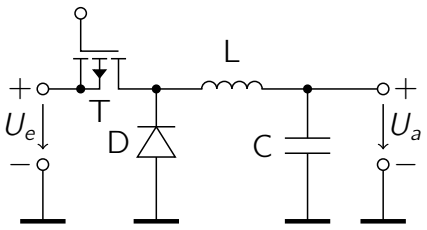


Abbildung 2.2: Prinzipschaltung eines Step-Down-Wandlers

2.2.3 Invertierender Wandler

Ein Invertierender Wandler (Abb. 2.3) erzeugt aus einer gegebenen Gleichspannung eine Gleichspannung, deren Betrag theoretisch zwischen $0\,\mathrm{V}$ und $\infty\,\mathrm{V}$ liegen kann, wobei die Polarität der Ausgangsspannung zur Eingangsspannung invertiert ist. In der Praxis wird die maximale Spannung durch die elektrischen Grenzwerte der verwendeten Bauteile begrenzt.
Das Verhältnis aus Ausgangs- und Eingangsspannung entspricht dem Verhältnis von Ein- und Ausschaltdauer des Schalters bzw. MOSFETs:

$$\frac{U_a}{U_e} = \frac{t_{ein}}{t_{aus}} = \frac{D}{1 - D}$$

Durch Umstellen erhält man die Formel zum Berechnen der Ausgangsspannung bei gegebenem Tastverhältnis und Eingangsspannung:

$$U_a = U_e \cdot \frac{t_{ein}}{t_{aus}} = U_e \cdot \frac{D}{1 - D}$$

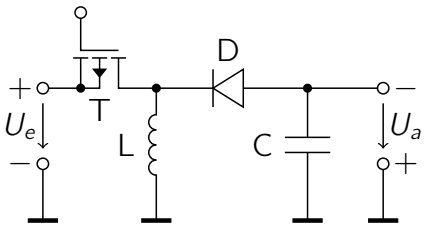


Abbildung 2.3: Prinzipschaltung eines invertierenden Wandlers

2.2.4 Alternative Bezeichnungen

In der deutsch- und englischsprachigen Literatur trifft man auf verschiedene Begriffe für die drei Typen sekundärgetakteter Schaltregler. Alle drei Typen werden unter dem Begriff Durchfluß- oder Drosselwandler zusammengefasst. Alternative Bezeichnungen sind in der folgenden Tabelle aufgeführt.

Bezeichnung des Wandlers	Alternative Bezeichnung	Englische Bezeichnung
Step-Up-Wandler	Aufwärtswandler, Hochsetzsteller	Step-Up-Converter Boost converter
Step-Down-Wandler	Abwärtswandler, Tiefsetzsteller	Step-Down-Converter Buck converter
Invertierender Wandler	Sperrsteller, Tief-Hochsetzsteller	Buck-boost-converter

Tabelle 2.1: Alternative Bezeichnungen

Ein „Buck-Boost converter“ kann auch aus einem Step-Down- und einem Step-Up-Wandler bestehen, die hintereinander geschaltet sind, wodurch eine nicht invertierte Ausgangsspannung erreicht werden kann, die größer oder kleiner der Eingangsspannung ist. Es müssen aber beide Wandler angesteuert werden, wobei je nach Ausgangsspannung (größer oder kleiner als die Eingangsspannung) nur jeweils einer mit einem PWM-Signal angesteuert wird.

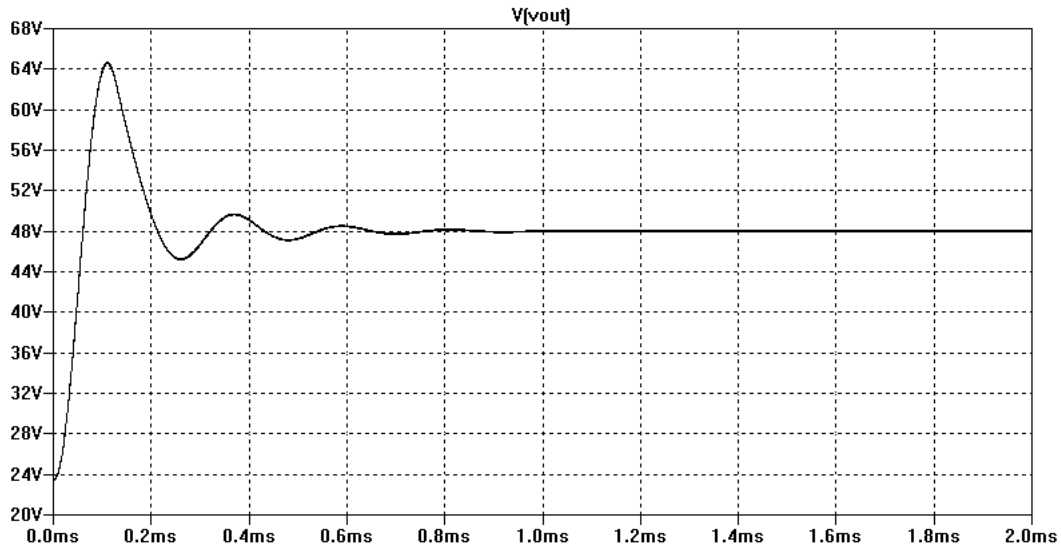
Die Begriffe Induktivität, Spule und Drossel bezeichnen alle ein Bauteil, das aus einer oder mehreren Windungen eines Leiters besteht, der ein Magnetfeld erzeugt, wenn ein Strom durch ihn fließt. Die Bezeichnungen Spule und Drossel werden häufig genutzt, um Induktivitäten für verschiedene Anwendungsbereiche zu unterscheiden. In dieser Arbeit werden diese Bauteile generell als Induktivitäten bezeichnet.

2.3 Simulation eines Step-Up-Wandlers

Anhand verschiedener Simulationen eines Step-Up-Wandlers soll hier das Verhalten im Normalbetrieb und bei der Änderung verschiedener Parameter gezeigt werden.
Die Simulation erfolgt mit dem Programm LTSpice (siehe Anhang E). Die Schaltungsbeschreibungen für die Spice-Simulation sind im Anhang A aufgeführt. Auf die Benutzung des grafischen Schaltplaneditors SwitcherCAD, der ebenfalls zum Programmpaket gehört wurde verzichtet. Die Modelle bzw. Subcircuits der verwendeten Bauteile sind im Anhang A.4 zu finden. Für die Simulation müssen diese in die Datei `models.lib` im gleichen Verzeichnis wie die Schaltungsbeschreibung oder in die Beschreibung selbst eingefügt werden, wobei im zweiten Fall die Zeile `.lib models.lib` aus der Beschreibung entfernt werden muss.
Auf die Erläuterung der Simulationen des Step-Down-Wandlers wird aufgrund des ähnlichen Verhaltens verzichtet, die Schaltungsbeschreibungen sind aber ebenfalls im Anhang zu finden.

2.3.1 Simulation der Grundschaltung

Die Grundschaltung (Beschreibung im Anhang A.1) in der Simulation entspricht von den simulierten Bauteilen etwa der umgesetzten Schaltung. Der MOSFET wird aber mit einem konstanten Tastverhältnis angesteuert, es findet also keine Regelung der Ausgangsspannung statt. Als Last wird ein Widerstand mit $32\,\Omega$ simuliert, durch den bei einer Ausgangsspannung von 48 V ein Strom 1,5 A fließt. Es wäre bei der realen Schaltung also ein Leistungswiderstand mit einer Nennleistung von mindestens 72 W nötig.
Das Diagramm 2.4 zeigt nach dem Einschalten ein Einschwingen auf den Sollwert von 48 V innerhalb von etwa 1,5 ms. Dieses Überschwingen tritt auf, da der Kondensator zunächst nicht geladen ist und der Strom nur durch die Bauteile im Stromkreis begrenzt ist. Wenn die Spannung am Kondensator dann den gewünschten Wert erreicht hat, versucht die Induktivität diesen hohen Strom zu erhalten, wodurch der Kondensator weiter geladen wird. In der Praxis wird dieses starke Überschwingen durch eine passende Regelung vermieden, indem nicht sofort das als optimal angenommene Tastverhältnis – für eine beabsichtigte Spannungsverdoppelung also 50 % – angelegt wird, sondern der passene Wert etwas langsamer eingestellt wird.



X-Achse: t_{sim}; Y-Achse: U_{Out}

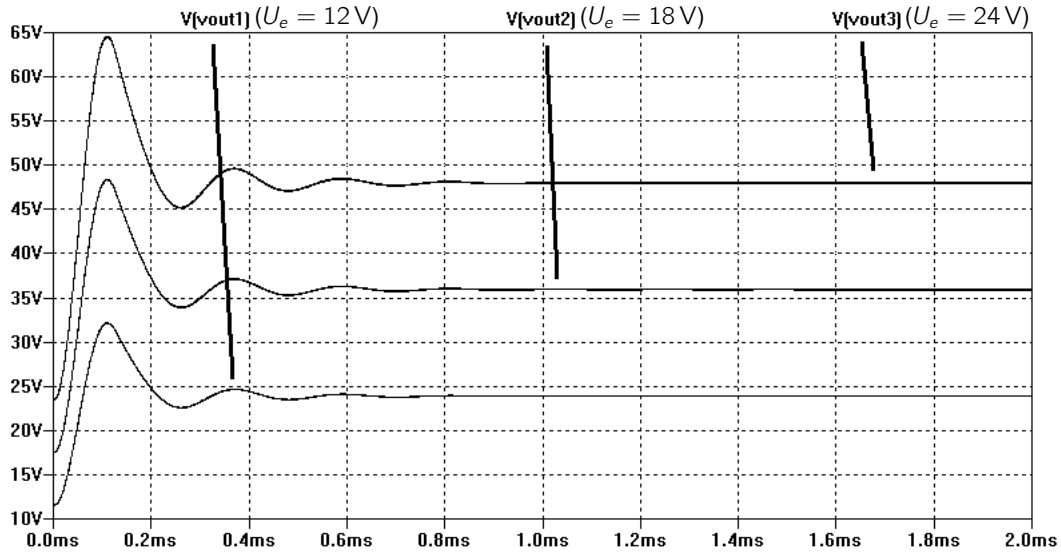
Abbildung 2.4: Simulation des Step-Up-Wandlers

2.3.2 Betrieb mit verschiedenen Eingangsspannungen

Eine Variation der Eingangsspannung verursacht eine proportionale Änderung der Ausgangsspannung (Abb. 2.5). Aus dem gewählten Tastverhältnis von 50 % folgt jeweils eine Verdoppelung der Eingangsspannung. Für die simulierten Eingangsspannungen von 12 V, 18 V sowie 24 V (siehe Anhang A.2) ergeben sich demnach Ausgangsspannungen von 24 V, 36 V, sowie 48 V. Das Einschwingen auf eine konstante Ausgangsspannung dauert in allen Fällen etwa gleich lange.

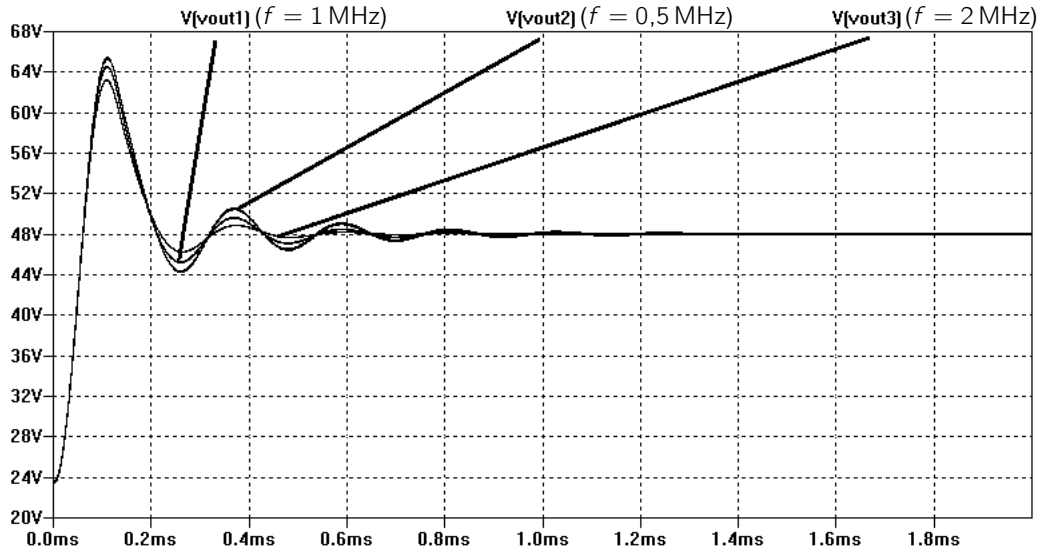
2.3.3 Betrieb mit verschiedenen Schaltfrequenzen

Wird die Schaltfrequenz verändert (Abb. 2.6, Simulationscode im Anhang A.3, die Frequenzen in der Simulation sind 500 kHz, 1 MHz und 2 MHz), ändert sich die Amplitude des Über- und Unterschwingens. Je höher die Frequenz ist, desto geringer ist die Amplitude. Hervorgerufen wird das durch den Blindwiderstand des Kondensators. Durch die beim Einschwingen mit der Zeit abnehmende Amplitude wird bei höherer Frequenz auch schneller eine als konstant anzusehende Ausgangsspannung erreicht.



X-Achse: t_{sim}; Y-Achse: U_{Out}

Abbildung 2.5: Simulation des Step-Up-Wandlers, Variation der Eingangsspannung



X-Achse: t_{sim}; Y-Achse: U_{Out}

Abbildung 2.6: Simulation des Step-Up-Wandlers, Variation der Schaltfrequenz

2.3.4 Betrieb mit verschiedenen Tastverhältnissen

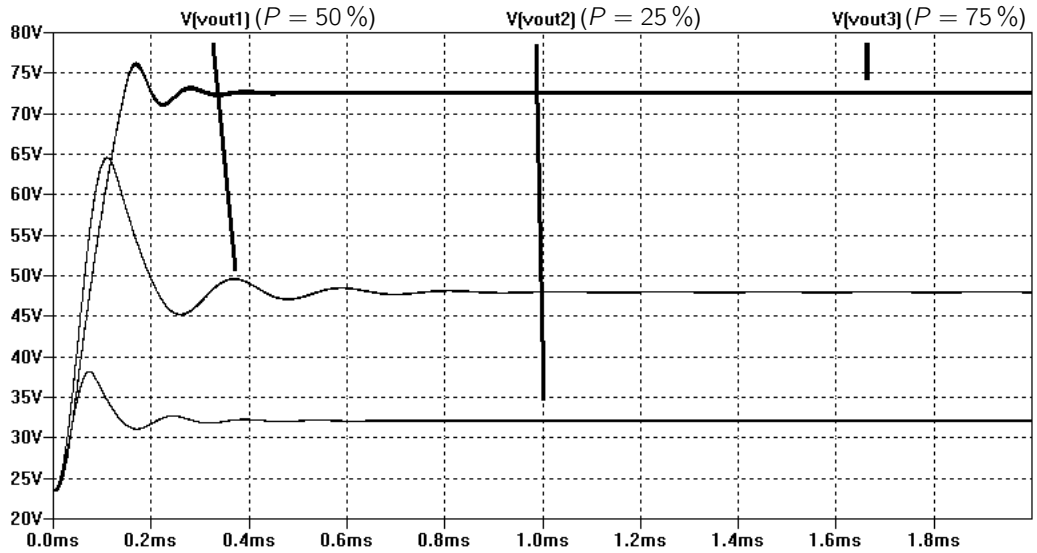
Für die Regelung grundlegend ist die Variation des Tastverhältnisses (in der Simulation 25 %, 50 % sowie 75 %, siehe Anhang A.4). Je größer das Tastverhältnis ist, desto größer ist auch die Ausgangsspannung (Abb. 2.7). Anhand der Gleichung in 2.2.1 lässt sich aus dem Tastverhältnis und der Eingangsspannung direkt die Ausgangsspannung berechnen. Für das Tastverhältnis von 75 % ergibt sich theoretisch allerdings nicht wie in der Simulation eine Ausgangsspannung von etwa 72 V sondern eine Spannung von $U_a = 24\,\mathrm{V} \cdot \frac{1}{1-75\,\%} = 96\,\mathrm{V}$. Diese Differenz beruht auf Grenzwerten der Bauteile, die zum Teil in der Simulation berücksichtigt werden und auch in ähnlichem Maße in der Realität auftreten würden, sofern die Bauteile nicht vorher schon zerstört werden.

2.3.5 Betrieb mit verschiedenen Induktivitäten

Wird als einzige Größe die Induktivität verändert (in der Simulation 10 µH, 33 µH sowie 100 µH, siehe Anhang A.5), ändert sich die Einschwingzeit, da sich der Stromfluss durch eine größere Induktivität langsamer ändert als durch eine kleine Induktivität. Desweiteren ändert sich noch die Frequenz des Einschwingens (Abb. 2.8). Die Ausgangsspannung liegt unabhängig von der Induktivität nach dem Einschwingen immer bei 48 V. Ein Effekt, der in dieser Simulation nicht erkennbar ist, tritt bei einer geringen Last am Ausgang auf. Der Strom, der dem Wandler minimal entnommen werden muss, um den Bezug zwischen Tastverhältnis und Ausgangsspannung zu erhalten, ist umgekehrt proportional zur verwendeten Induktivität. Dieser Effekt wird in Abschnitt 2.3.8 genauer betrachtet.

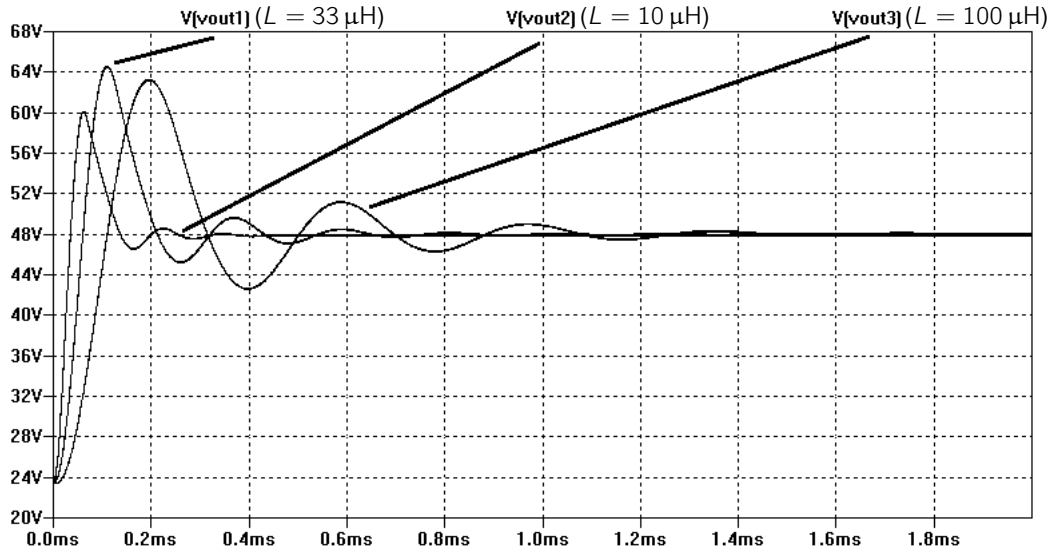
2.3.6 Betrieb mit verschiedenen Kapazitäten

Bei einer Veränderung der Kapazität am Ausgang (simuliert mit 4,5 µF, 8,9 µF und 17,7 µF, siehe Anhang A.6) verändert sich augenscheinlich nur die Frequenz des Einschwingens (Abb. 2.9). Bei genauer Betrachtung (Abb. 2.10) der Ausgangsspannung nach dem Einschwingen zeigt sich außerdem, dass sich die Restwelligkeit umgekehrt proportional zur gewählten Kapazität verhält. Verursacht wird das wiederum durch den Blindwiderstand X_C des Kondensators.



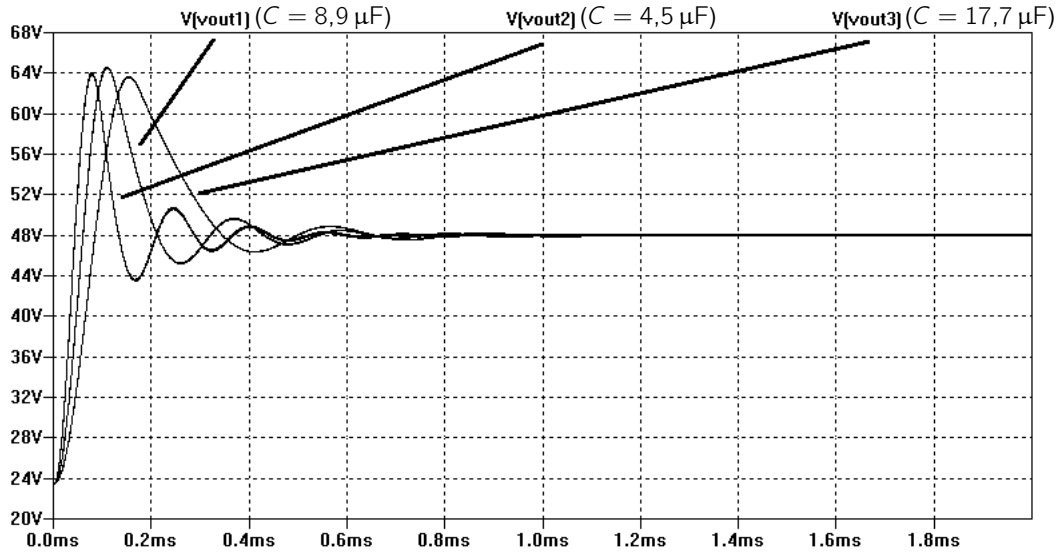
X-Achse: t_{sim}; Y-Achse: U_{Out}

Abbildung 2.7: Simulation des Step-Up-Wandlers, Variation des Tastverhältnisses



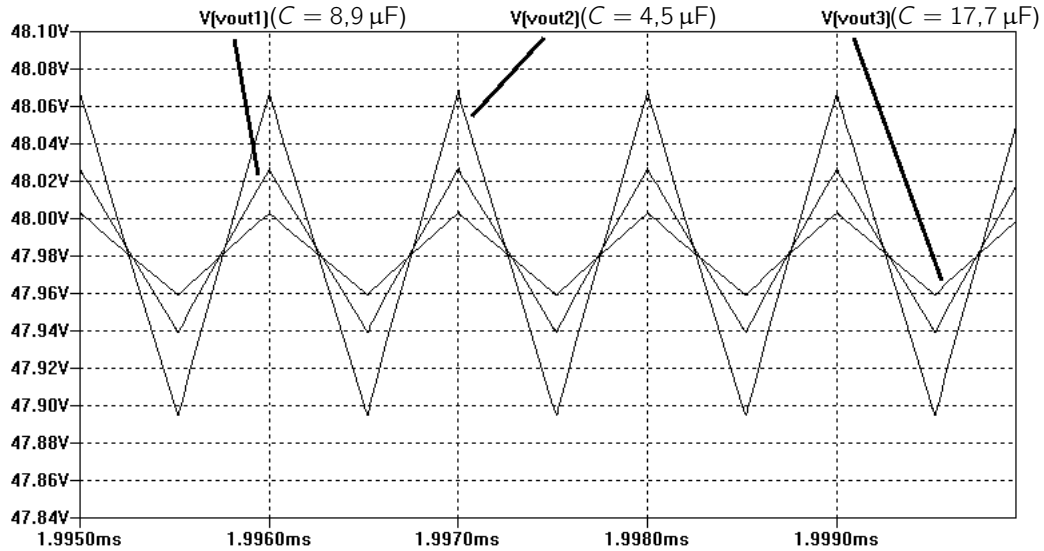
X-Achse: t_{sim}; Y-Achse: U_{Out}

Abbildung 2.8: Simulation des Step-Up-Wandlers, Variation der Induktivität



X-Achse: t_{sim}; Y-Achse: U_{Out}

Abbildung 2.9: Simulation des Step-Up-Wandlers, Variation der Kapazität



X-Achse: t_{sim}; Y-Achse: U_{Out}

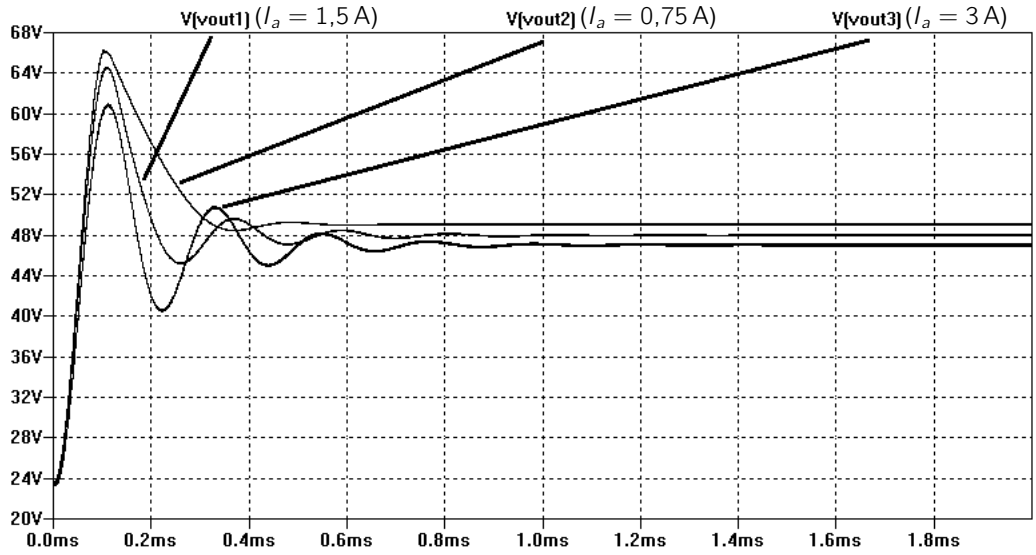
Abbildung 2.10: Ausschnittsvergrößerung von Abb. 2.9

2.3.7 Betrieb mit verschiedenen Lasten

Bei verschiedenen Lasten am Ausgang, hier durch Widerstände mit $16\,\Omega$, $32\,\Omega$ und $64\,\Omega$ simuliert (siehe Anhang A.7), wodurch sich Ströme von 3 A, 1,5 A sowie 0,75 A ergeben, werden unterschiedliche Einschwingdauern sowie -frequenzen hervorgerufen, die jeweils umso größer sind, je höher die Last am Ausgang ist. Außerdem ändert sich die Ausgangsspannung um einen kleinen Betrag umgekehrt proportional zur Last (Abb. 2.5). Die Ursache hierfür ist der Innenwiderstand des Wandlers und ist so auch bei anderen Spannungsquellen wie z. B. Batterien vorhanden.

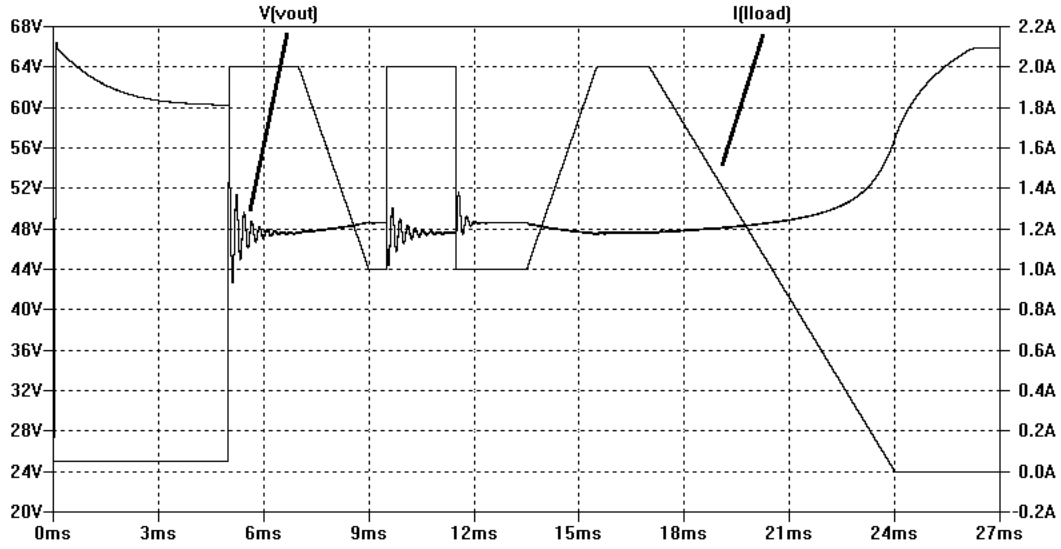
2.3.8 Lastwechsel im Betrieb

Werden am Ausgang Lastwechsel durchgeführt, während alle anderen Parameter konstant bleiben – Code siehe Anhang A.8 – lassen sich im Diagramm 2.12 weitere Effekte erkennen.
Während der ersten 5 ms werden 50 mA als Last am Ausgang angeschlossen. Dabei steigt die Ausgangsspannung erst auf den durch die Diode begrenzten Maximalwert von 66 V und sinkt dann auf etwa 60 V ab. Dieser Effekt einer zu hohen Ausgangsspannung für das angelegte Tastverhältnis tritt auf, wenn der Strom am Ausgang zu niedrig ist und ein deutlich geringeres Tastverhältnis erforderlich macht.
Anschließend wird innerhalb einer Mikrosekunde die Last erhöht, so dass ein Strom von 2 A fließt. Die Spannung sinkt erst auf ca. 43 V ab und schwingt dann auf knapp unter 48 V ein. Es werden nicht genau 48 V erreicht, da das angelegte Tastverhältnis auf eine Ausgangsspannung von 48 V bei 1,5 A angepasst ist. Wie im Abschnitt 2.3.7 gezeigt, weicht die Ausgangsspannung bei wechselnder Last um einen geringen Betrag umgekehrt proportional vom Idealwert ab.
Zwischen 7 ms und 9 ms sinkt die Last mit $-500\,{}^{\mathrm{mA}}/_{\mathrm{ms}}$ auf 1 A ab. Dabei steigt die Ausgangsspannung nahezu linear auf einen Wert knapp über 48 V.
Bei den folgenden Lastsprüngen mit $\pm 1\,{}^{\mathrm{A}}/_{\mu\mathrm{s}}$ schwingt die Ausgangsspannung ebenfalls auf Werte knapp unter bzw. über 48 V ein. Ebenso verursacht die langsame Steigerung der Last, wiederum mit $500\,{}^{\mathrm{mA}}/_{\mathrm{ms}}$ eine nahezu lineare Abnahme der Spannung.
Abschließend wird die Last abgesenkt, wobei die Ausgangsspannung mit niedriger werdender Last immer stärker ansteigt. Wenn 0 A erreicht sind, steigt die Spannung weiter an, die Kurve wird aber wieder flacher, da die Spannung gegen einen Grenzwert von 66 V konvergiert.



X-Achse: t_{sim}; Y-Achse: U_{Out}

Abbildung 2.11: Simulation des Step-Up-Wandlers, Variation der Last



X-Achse: t_{sim}; Y-Achse links: U_{Out}; Y-Achse rechts: I_{load}

Abbildung 2.12: Simulation des Step-Up-Wandlers, Lastwechsel und unbelasteter Ausgang

2.4 Mögliche Regelungsarten

Das Tastverhältnis, das mit den Gleichungen für die Step-Up- und Step-Down-Wandler berechnet werden kann, stellt nur den theoretischen Optimalfall dar, der in der Praxis nicht eintreten wird. Im Betrieb unter Last treten, wie in Abschnitt 2.3.7 gezeigt, je nach Last kleine Abweichungen vom optimalen Tastverhältnis auf.
Größere Probleme treten beim Betrieb mit niedriger bzw. ohne Last auf, da dem Speicherkondensator weniger Energie entnommen wird, als ihm zugeführt wird. Dadurch steigt die Ausgangsspannung sowohl beim Step-Up- als auch beim Step-Down-Wandler beim optimalen Tastverhältnis stark an. Es muss ein deutlich niedrigeres Tastverhältnis eingestellt werden, um Beschädigungen der Elektronik zu verhindern.
Um die gewünschte Ausgangsspannung zu erreichen, wird eine Regelung benötigt, die die tatsächliche Ausgangsspannung (Ist-Spannung U_{Ist}) und die einzustellende Ausgangsspannung (Soll-Spannung U_{Soll}) vergleicht und das Tastverhältnis dynamisch anpasst.
Als Möglichkeit zur Regelung existieren verschiedene integrierte Schaltkreise. Beispielsweise könnte der LM3478 von National Instruments eingesetzt werden, dieser ist mit einem achtpoligen Gehäuse sehr klein und benötigt nur wenige zusätzliche Bauteile. Diese Regler haben aber teilweise ungeeignete Grenzen in der Schaltfrequenz, der möglichen Ausgangsspannung sowie der Ausgangsleistung.
Alternativ kann der Regler auch komplett entwickelt werden. Die einfachste Lösung für einen solchen Regeler besteht in einem analogen Regelkreis. Ein möglicher Aufbau, der auch zusammen mit einem Schalt-MOSFET in einem IC verfügbar ist, ist im Buch „Halbleiter-Schaltungstechnik“ (Tietze und Schenk, 1999, Kapitel 16.6.2, Abb. 16.39) dargestellt.
Mehr Freiheiten bei der Regelung, die Fähigkeit mehrere Ausgänge zu reglen sowie die Möglichkeit der Abfrage und Veränderung verschiedener Parameter bietet die hier umgesetzte Lösung auf Basis eines Mikrocontrollers. Es kann ein optimierter Regelalgorithmus umgesetzt werden, der schneller auf Störungen reagiert als ein analoger Regelkreis. Desweiteren liegen die Werte der Ausgangsspannung vor und können auch extern – beispielsweise von einem Steuerrechner – abgefragt werden. Zusätzlich kann eine Funktion zum Ändern der Soll-Spannung implementiert werden, die ermöglicht, die Ausgangsspannung nachträglich zu ändern. Sofern der Mikrocontroller ausreichend leistungsfähig ist, könnten die Parameter des Regelalgorithmus veränderlich ausgelegt werden, so dass nachträglich eine Optimierung der Regelung für das Lastprofil der Endanwendung möglich ist. Für diese Arbeit wird aufgrund von Vorkenntnissen der Architektur ein Mikrocontroller der AVR-Familie von Atmel verwendet. Es gibt vermutlich auch andere Mikrocontroller, die aufgrund ihrer Rechenleistung noch besser für die Regelung geeignet wären. Ebenso kann auch ein FPGA oder bei großen Stückzahlen sogar ein speziell für die Anwendung entwickelter Chip (ASIC) eingesetzt werden.

3 Entwurf und Umsetzung

Aufgabenstellung dieser Bachelorarbeit war die konkrete Planung und die Umsetzung eines Schaltungsentwurfs für einen Step-Up-Wandler und die Untersuchung von dessen Leistungsfähigkeit. Dabei gibt es verschiedene Anforderungen an den Step-Up-Wandler beziehungsweise an die komplette Schaltung.

Grundlegende Anforderungen

- Kleine Baugröße
  Damit der Step-Up-Wandler preisgünstig als Teil einer umfangreicheren Schaltung eingesetzt werden kann, sollte die Schaltung möglichst klein ausfallen. Das wird duch mehrere Faktoren erreicht.
  $\Rightarrow$ Einsatz von SMD-Bauelementen – Diese Bauelemente finden sich in nahezu jedem elektrischen Gerät und tragen zur Miniaturisierung in allen Bereichen bei.
  $\Rightarrow$ Hohe Schaltfrequenz – Durch eine Vergrößerung der Schaltfrequenz kann die Induktivität verkleinert werden, die schon in der umgesetzten Schaltung mit etwa 144 mm^2 Fläche das größte Bauteil darstellt.

- Stabile Ausgangsspannung
  Die Ausgangsspannung sollte möglichst in allen Betriebsbedingungen stabil sein und möglichst wenig schwanken. Dafür gibt es wieder unterschiedliche Ansätze.
  $\Rightarrow$ Gute Regelung bei Lastschwankungen – Der Regelalgorithmus muss schnell reagieren, wenn die Ausgangsspannung bei Lastschwankungen vom Sollwert abweicht.
  $\Rightarrow$ Einsatz von Glättungskondensatoren – Große Kapazitäten am Ausgang glätten einerseits leichte Spannungsschwankungen, die durch die Schaltvorgänge unweigerlich auftreten (den sogenannten Ripple), andererseits können sie auch die spontanen Änderungen bei Lastschwankungen abschwächen.

- Schnelles Erreichen der Ausgangsspannung
  Nach dem Einschalten der Regelung soll die Ausgangsspannung möglichst schnell den Sollwert erreichen. Am stärksten kann man darauf über den Regelalgorithmus Einfluß nehmen. Auch die Kapazität am Ausgang beeinflusst diese Zeit, je größer die Kapazität ist, desto länger dauert es, bis diese auf die Sollspannung geladen ist.

Es muss also ein Kompromiss aus Glättungsqualität und Einregelgeschwindigkeit gefunden werden.

- Möglichst hohe Ausgangsleistung
  Durch den Einsatz leistungsfähiger Komponenten kann eine hohe Ausgangsleistung erreicht werden. Beispielsweise gehört die Induktivität mit einem maximalen Strom von 4 A zu den leistungsfähigsten in der Baugröße.

Ergänzende Anforderungen

- Anschluss eines zuschaltbaren Lastwiderstands für den Fall einer Rückspeisung durch Induktion
- Zusätzliche Integration von ein oder zwei Step-Down-Wandlern mit den gleichen grundlegenden Anforderungen
- Möglichkeit verschiedene Betriebsdaten, beispielsweise die Ausgangsspannungen sowie deren Sollwerte, auszulesen und ggf. zu verändern

Eine Alternative zu der kompletten Entwicklung wäre der Einsatz eines Reglerbausteins, der für die geforderten Ausgangsleistungen trotzdem eine weitere externe Beschaltung erfordert. Neben der entfallenden Möglichkeit, dynamisch auf Betriebsdaten Einfluss zu nehmen, müsste für jeden Schaltregler ein solcher Baustein eigesetzt werden, wodurch die Herstellungskosten höher ausfallen als bei der Verwendung eines Mikrocontrollers.

3.1 Hardware – Schaltungsentwurf

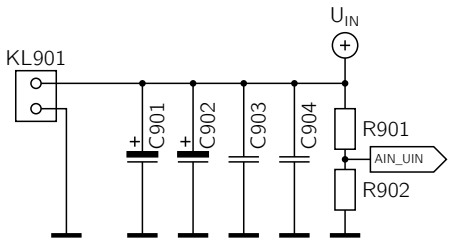
Für die Dimensionierung der Schaltung stand ein Entwurf zur Verfügung, der mit den Induktivitäten und Schalttransistoren einige vorgegebene Bauteile enthielt. Die Schaltung basiert auf einem ATtiny861V, der über einen internen 8 MHz Oszillator verfügt und zusätzlich eine PLL-Einheit zur Taktvervielfachung enthält. Diese erzeugt einen 64 MHz Takt, der für den Zähler zur PWM-Signalerzeugung genutzt werden kann. Damit ergibt sich bei einer Auflösung des Zählers von 8 Bit eine PWM-Frequenz von 250 kHz. Durch eine Beschränkung auf 6 Bit wird die vierfache Frequenz von 1 MHz erreicht, durch die bei den gegebenen Induktivitäten nur kleine Bauteile nötig sind, um gute elektrische Eigenschaften bei nicht zu geringer Auflösung zu erhalten. Statt des ATtiny861V kann auch der ATtiny861 verwendet werden. Dieser entspricht dem ATtiny861V, ist aber für einen Takt bis 20 MHz ausgelegt, während der ATtiny861V nur bis zu einem Takt von 10 MHz getestet ist.

Beim ATtiny861 hat man daher auch die Möglichkeit, die CPU mit einem über die PLL intern erzeugten Takt von 16 MHz zu betreiben. Außerdem kann der ATtiny861V ab einer Eingangsspannung von 1,8 V betrieben werden, der ATtiny861 erst ab 2,7 V. Bei dieser Schaltung ist das aber unwichtig, da der Mikrocontroller ohnehin mit 5 V versorgt wird.
Die Formeln zur Dimensionierung der Schaltregler wurden dem Buch „Halbleiter-Schaltungstechnik“ (Tietze und Schenk, 1999) entnommen, dabei wurden die gegebenen Induktivitäten berücksichtigt. Die genauen Berechnungen werden jeweils in den Abschnitten 3.1.2, 3.1.3 und 3.1.4 gezeigt.
Die Bauteilnumerierungen in den Schaltplänen sind nach den Komponenten der Schaltung gruppiert:

- **1xx** Mikrocontroller mit Beschaltung
- **2xx** Step-Up-Wandler
  - **25x** Überspannungsschutzschaltung (Lastwiderstand)
- **3xx** 12 V-Step-Down-Wandler
- **4xx** 3,3 V-Step-Down-Wandler
- **9xx** Spannungsversorgung der Schaltung

3.1.1 Spannungsversorgung und Beschaltung des Mikrocontrollers

An den Anschluss KL901 des Versorgungsteils (Abb. 3.1) wird die Versorgungsspannung angeschlossen. Ausgelegt ist diese Schaltung für eine Betriebsspannung von 24 V, sie lässt sich aber auch in einem gewissen Rahmen sowohl mit niedrigerer als auch mit höherer Spannung betreiben.
Die Kondensatoren C901 bis C904 dienen dem Ausgleich von Spannungsschwankungen, die sowohl von der Versorgungsspannungsquelle eingespeist werden können, als auch durch Lastschwankungen in der Schaltung entstehen können.
Der aus den Widerständen R901 und R902 aufgebaute Spannungsteiler wird für die Messung der Eingangsspannung über den AD-Wandler des Mikrocontrollers genutzt.



C901, C902	100 µF
C903, C904	100 nF
R901	4,7 kΩ
R902	470 Ω

Abbildung 3.1: Versorgungsteil

Für die Dimensionierung von R901 und R902 ist die Referenzspannung des AD-Wandlers, hier 5 V, sowie der mögliche Eingangsspannungsbereich zu betrachten. Die Eingangsspannung muss über 12 V liegen, da sonst der 12 V-Step-Down-Wandler seine Ausgangsspannung nicht erreichen kann. Desweiteren muss sie unter 48 V liegen, da der 48 V-Step-Up-Wandler sonst ebenfalls nicht korrekt arbeiten kann. In der Realität ist der mögliche Spannungsbereich sogar noch kleiner, die genauen Grenzwerte werden in Kapitel 4.4 ermittelt. Bei einer Eingangsspannung von 48 V darf am Ausgang des Spannungsteilers die Referenzspannung von 5 V nicht überschritten werden, da sonst keine Messung mehr möglich ist. Desweiteren sollte der Gesamtwiderstand einerseits natürlich groß sein, damit nur ein möglichst geringer Strom durch die zwei Widerstände fließt, andererseits sollte er geringer als etwa 10 kΩ sein, da das Messergebnis sonst verfälscht werden kann. Mit den eingesetzten Widerstandswerten ergibt sich folgende Spannung bei einer Versorgungsspannung von 48 V:

$$U_{AIN\_UIN[max]} = U_{IN} \cdot \frac{R2}{R1 + R2} = 48\,\mathrm{V} \cdot \frac{470\,\Omega}{4{,}7\,\mathrm{k}\Omega + 470\,\Omega} = 48\,\mathrm{V} \cdot \frac{470\,\Omega}{5170\,\Omega} = 4{,}36\,\mathrm{V}$$

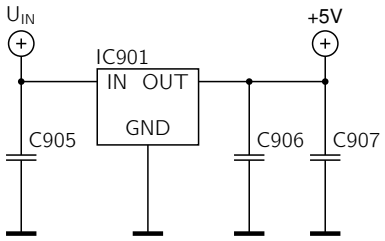
Es bleibt also ein Toleranzbereich gegenüber einer zu hohen Versorgungsspannung. Für die minimale Versorgungsspannung von 12 V ergibt sich folgende Spannung am Ausgang des Spannungsteilers:

$$U_{AIN\_UIN[min]} = 12\,\mathrm{V} \cdot \frac{470\,\Omega}{5170\,\Omega} = 1{,}09\,\mathrm{V}$$

Der 5 V Regler IC901 (Abb. 3.2) dient der Versorgung des Mikrocontrollers sowie des Treiber-ICs IC201 für den MOSFET des Step-Up-Wandlers im Versuchsaufbau der Schaltung. Es handelt sich beim LM2936HVMA um einen integrierten 5 V Spannungsregler, der einen großen Eingangsspannungsbereich abdeckt. Durch die geringe Dropoutspannung von etwa 0.2 V lässt er sich ab ca. 5.2 V bis zu seiner maximalen Eingangsspannung von 60 V betreiben. Somit deckt er auch den möglichen Eingangsspannungsbereich der Schaltung ab. Wegen eines Defekts am LM2936HVMA wurde für die weitere Entwicklung und die Messungen ein 78L05 eingesetzt, der pinkompatibel ist, aber nur mit maximal 30 V Eingangsspannung betrieben werden darf.

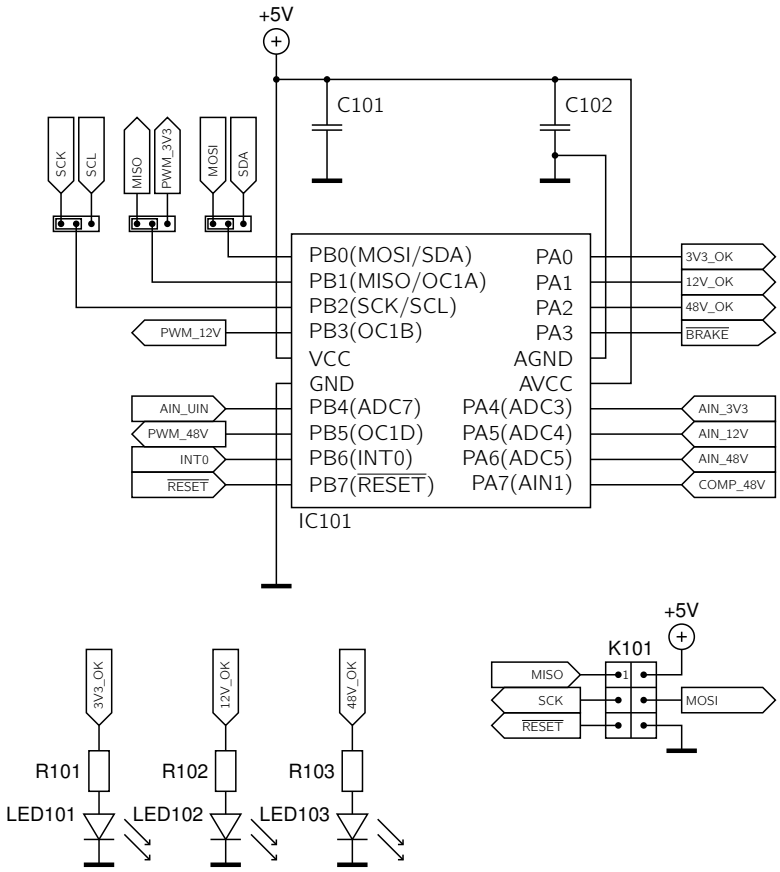
Abb. 3.3 zeigt den Mikrocontroller mit den angeschlossenen Signalen sowie den Glättungskondensatoren C101 und C102. Außerdem sind noch die Statusleuchtdionen LED101 bis LED103 mit deren Vorwiderständen R101 bis R103 und der Steckverbinder K101 mit dem ISP-Anschluss dargestellt, der der Programmierung des Mikrocontrollers dient.

Die Bedeutung der einzelnen Signalnamen ist in der Tabelle 3.1 (Seite 28) aufgeführt. Die Signale SCK und SCL, MISO und PWM_3V3 sowie MOSI und SDA sind über Jumper umschaltbar ausgelegt, da nach der ursprünglichen Planung die ISP- (und SPI-)Signale nur zur Programmierung und die PWM- und I^2C-Signale für den Betrieb benötigt werden. Für die nun realisierte Kommunikation über SPI sind diese Jumper nicht nötig, da alle ISP-Datenleitungen und außerdem die vorher ungenutzte INT0-Leitung dafür benötigt werden.



C905, C906	100 nF
C907	4,7 µF
IC901	LM2936HVMA

Abbildung 3.2: 5 V Regler



C101, C102	100 nF
IC101	Atmel ATtiny861V
R101, R102, R103	1,5 kΩ
LED101, LED102, LED103	Low Current (2 mA) Leuchtdioden, grün
K101	ISP-Anschluss

Abbildung 3.3: Mikrocontroller, Statusleuchtdioden und ISP-Schnittstelle

3V3_OK 12V_OK 48V_OK	Ausgänge für den Anschluss von Statusleuchtdioden, die anzeigen, ob die jeweilige Spannung ihren Sollwert erreicht hat. Die 3V3_OK LED wird als allgemeine Statusanzeige genutzt.
$\overline{\text{BRAKE}}$	Steuerleitung für einen Lastwiderstand, der bei einer Überspannung auf der Ausgangsseite des Step-Up-Wandlers zugeschaltet werden kann.
AIN_3V3 AIN_12V AIN_48V AIN_UIN	Eingänge des AD-Wandlers für die Messung der jeweiligen Ausgangsspannungen sowie der Versorgungsspannung.
COMP_48V	Eingang des Analog-Komparators zur Überwachung der Ausgangsspannung des Step-Up-Wandlers, um im Falle einer Überspannung schnell gegensteuern zu können.
$\overline{\text{RESET}}$	Der Reset-Eingang des Mikrocontrollers, dieser wird zur Programmierung benötigt.
INT0	Ein unbelegter I/O-Pin, der auch als externer Interrupteingang genutzt werden kann. Im Aufbau der Schaltung, der zur Bearbeitung dieser Arbeit genutzt wird, wird dieser Pin als Slave-Select-Eingang für die SPI-Kommunikation genutzt.
PWM_3V3 PWM_12V PWM_48V	Ausgänge für die Steuersignale der einzelnen Wandlerkomponenten.
SCK MOSI MISO	ISP-Leitungen zur Programmierung und SPI-Leitungen für die Kommunikation des Mikrocontrollers.
SCL SDA	I^2C Kommunikationsleitungen. Im vorhandenen Aufbau entgegen erster Planungen ungenutzt.

Tabelle 3.1: Beschreibung der Signalnamen

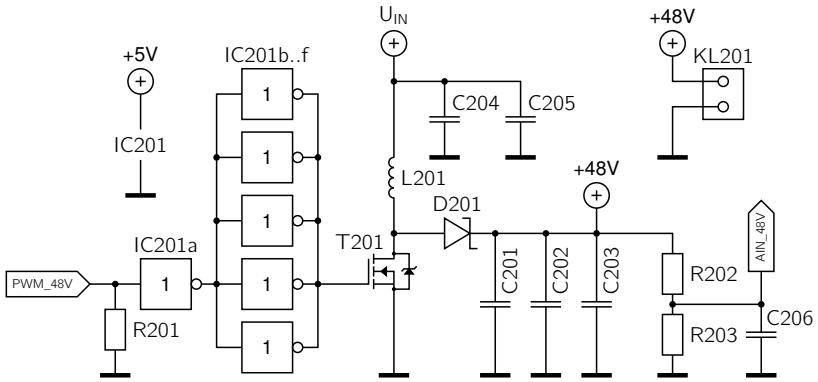
3.1.2 48 V Step-Up-Wandler

Der Step-Up-Wandler in dieser Schaltung ist für eine Verdoppelung einer Eingangsspannung von 24 V ausgelegt. Für eine Spannungsverdopplung wird ohne Berücksichtigung der Verluste an der Diode, dem Widerstand der Induktivität und am MOSFET ein PWM-Tastverhältnis von genau 50 % benötigt.
Die Basis des Step-Up-Wandlers aus Abb. 3.4 bildet die Kombination aus der Induktivität L201, dem Schalttransistor T201, der Diode D201 und der Kapazität, die aus den parallel geschalteten Kondensatoren C201, C202 und C203 gebildet wird.
Sofern der Strom am Ausgang größer als ein Minimalstrom ist, gilt zur Berechnung der Ausgangsspannung die Gleichung $U_a = U_e \cdot \frac{T}{t_{aus}}$, wobei T die Periodendauer des PWM-Signals ist und t_{aus} die Dauer, in der der Transistor sperrt. Wenn der Strom geringer als dieser Minimalstrom ist, ist für die gleiche Ausgangsspannung ein deutlich höherer Wert für t_{aus}, also eine kürzere Einschaltdauer notwendig.
Der Minimalstrom lässt sich mit folgender Formel (Tietze und Schenk, 1999) berechnen:

$$\begin{aligned} I_{a\,min} &= \frac{U_e^2}{U_a} \cdot \left(1 - \frac{U_e}{U_a}\right) \cdot \frac{T}{2 \cdot L} \\ &= \frac{24\,\mathrm{V}^2}{48\,\mathrm{V}} \cdot \left(1 - \frac{24\,\mathrm{V}}{48\,\mathrm{V}}\right) \cdot \frac{1\,\mu\mathrm{s}}{2 \cdot 33\,\mu\mathrm{H}} \\ &= \frac{1}{11}\,\frac{\mathrm{V} \cdot \mathrm{s}}{\mathrm{H}} = \frac{1}{11}\,\mathrm{A} = 90{,}9\,\mathrm{mA} \end{aligned}$$

Wird dem eingeregelten Wandler ein Strom über diesem Minimalstrom entnommen, fließt ständig Strom durch die Induktivität. Ist der entnommene Strom gleich dem Minimalstrom, erreicht der Stromfluss durch die Induktivität am tiefsten Punkt genau 0 A. Wird der Minimalstrom unterschritten, erreicht der Spulenstrom in einer kürzeren Zeit als t_{aus} die 0 A und bleibt auf diesem Wert, bis der MOSFET durchschaltet und der Strom wieder ansteigt.
Um die Ausgangsspannung zu glätten, ist der Kondensator am Ausgang passend zu dimensionieren. Die Formel um einen ungefähren Wert zu errechnen lautet $C \approx \frac{T \cdot I_{a\,min}}{\Delta U_a}$, wobei ΔU_a für die Restwelligkeit der Ausgangsspannung steht. Für eine Amplitude der Welligkeit von 0,1 % der Ausgangsspannung, also 0,048 V, ergibt sich eine benötigte Kapazität von

$$\begin{aligned} C &\approx \frac{T \cdot I_{a\,min}}{\Delta U_a} \\ &= \frac{1\,\mu\mathrm{s} \cdot 90{,}9\,\mathrm{mA}}{48\,\mathrm{mV}} \\ &= 1{,}89\,\mu\mathrm{F} \end{aligned}$$



IC201	74AC04	D201	SS36
R201	1 kΩ	C201, C204	100 nF
R202	6,8 kΩ	C202, C203	2,2 µF
R203	620 Ω	C205	4,7 µF
L201	33 µH	C206	100 pF
T201	Si7852DP		

Abbildung 3.4: Step-Up-Wandler 24 V nach 48 V

Mit den vorgesehenen Kondensatoren C201-C203 ergibt sich eine Kapazität von 4,5 µF, C202 und C203 wurden im Aufbau doppelt bestückt, so dass die Gesamtkapazität bei 8,9 µF liegt, wodurch die Welligkeit noch weiter reduziert wird.
Werden die Formeln für Restwelligkeit und Minimalstrom kombiniert, kann man gut sehen, welche Maßnahmen die beste Verringerung der Restwelligkeit verursachen.

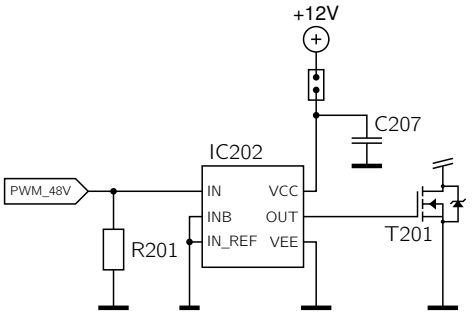
$$\begin{aligned}\Delta U_a &\approx \frac{T \cdot I_{a\,min}}{C} \\ &= \frac{T}{C} \cdot \frac{U_e^2}{U_a} \cdot \left(1 - \frac{U_e}{U_a}\right) \cdot \frac{T}{2 \cdot L} \\ &= \frac{T^2}{2 \cdot C \cdot L} \cdot U_e \left(\frac{U_e}{U_a} - \left(\frac{U_e}{U_a}\right)^2\right)\end{aligned}$$

Da die Periodendauer quadratisch eingeht, würde eine Verdoppelung der Frequenz die Restwelligkeit auf ein Viertel reduzieren. Änderungen an der Induktivität und der Kapazität wirken sich nur linear aus. Die einfachste Änderungsmöglichkeit ist aber die Erhöhung der Kapazität durch Parallelschaltung von Kondensatoren, wie es im praktischen Aufbau auch vorgenommen wurde. Die Periodendauer bzw. Frequenz lässt sich dagegen nur schwer, oft mit Einbußen an anderen Stellen, ändern.

Änderungen an der Eingangsspannung sind in manchen Anwendungsfällen auch möglich, wobei die hier verwendete Spannungsverdoppelung sogar den schlechtesten Fall für die Restwelligkeit darstellt, da der Term $\frac{U_e}{U_a} - (\frac{U_e}{U_a})^2$ der obigen Gleichung bei einem Verhältnis $\frac{U_e}{U_a} = 0{,}5$ mit $0{,}25$ sein Maximum erreicht und bei den unrealistischen Grenzen $U_e = 0\,\mathrm{V}$ sowie $U_e = U_a$ den Wert 0 hat. Desweiteren hängt die Amplitude der Welligkeit noch linear von der Eingangsspannung ab. Das theoretische Optimum wäre also eine Eingangsspannung nahe der 0 V, bei der die Restwelligkeit ebenfalls gegen 0 geht. Da die Leistung auf der Ausgangsseite aber der Leistung auf der Eingangsseite abzüglich der Verluste entspricht, muss für eine gegebene Ausgangsleistung mit sinkender Eingangsspannung ein immer höherer Eingangsstrom fließen. Eine Änderung der Ausgangsspannung ist in der Regel nicht möglich, da der Wandler gerade für diese konzipiert wurde.

Für die Glättung der Eingangsspannung werden die Kondensatoren C204 und C205 genutzt. Über den Spannungsteiler aus R202 und R203 wird die Ausgangsspannung auf einen für AD-Wandler des Mikrocontrollers zulässigen Wert unter 5 V reduziert. C206 ist ein Pufferkondensator um Einstreuungen auf der Messleitung abzuschwächen. Die erzeugte Ausgangsspannung steht am Anschluss KL201 zur Verfügung.

Der Mikrocontroller kann mit 40 mA nicht genug Strom am Ausgang liefern, um T201 schnell umzuschalten, da zum vollständigen Durchschalten eine bestimmte Ladung am Gate erreicht werden muss. Durch langsame Schaltvorgänge fällt Leistung am Transistor ab, wodurch dieser sich stark erwärmen kann. Um dies zu vermeiden wird ein MOSFET-Treiber eingesetzt, der in der Lage ist, einen deutlich höheren Strom zu liefern und die Ladung am Gate somit schneller aufzubauen. Vorgesehen war ein LM5112SD, ein integrierter Baustein, der den Transistor mit mehreren Ampere ansteuern kann. Die Beschaltung dieses Bausteins ist in Abb. 3.5 zu sehen. Die Versorgungsspannung von 12 V wurde hier zunächst über einen Jumper schaltbar ausgelegt. Problematisch bei der Verwendung des LM5112 ist dessen Undervoltage-Grenzwert. Dieser liegt laut Datenblatt (National Semiconductor, 2006) bei etwa 3 V, was dazu führt, dass der MOSFET mit dieser niedrigen Gatespannung angesteuert wird, so dass dieser zwar leitet, aber immer noch einen relevanten Drain-Source-Widerstand besitzt. Durch die abfallende Leistung kann der MOSFET zerstört werden und unter Umständen die anliegende Spannung über das Gate an den LM5112 zurückleiten, wodurch dieser ebenfalls zerstört wird. Eine solche Versorgungsspannung des Treibers (zu niedrig für den MOSFET, zu hoch für die Schutzschaltung) kann beispielsweise beim Einschalten auftreten. Daher ist sicherzustellen, dass die Versorgungsspannung ausreichend hoch ist, wenn der MOSFET angesteuert wird. Desweiteren bedeutet dies, dass die Regelung des Step-Up-Wandlers erst eingeschaltet werden darf, wenn der Step-Down-Wandler die 12V konstant erzeugt, sofern der MOSFET-Treiber durch diesen versorgt wird. Aufgrund dieser Empfindlichkeit wurde dieser für die Entwicklung und die Messungen durch einen 74AC04 ersetzt, der mit 5 V versorgt wird. Der 74AC04 (IC201, Abb. 3.4) enthält sechs Inverter, von denen einer direkt vom Mikrocontroller angesteuert wird und wiederum die anderen fünf parallel geschalteten Inverter ansteuert, die jeweils 50 mA Ausgangsstrom liefern. Der Transistor wird dadurch mit 250 mA bei 5 V angesteuert.



R201	1 kΩ
C207	100 nF
T201	Si7852DP
IC202	LM5112SD

Abbildung 3.5: Vorgesehener Treiber für den Step-Up-Wandler

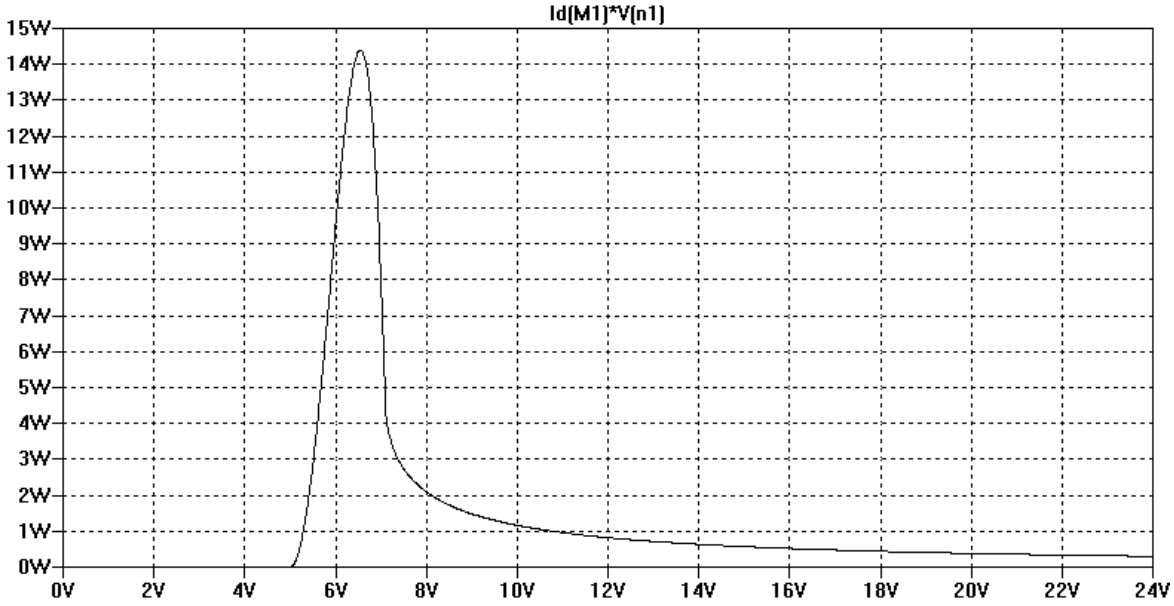
Auswirkung der Schaltzeiten an MOSFETs

An MOSFETs können während des Wechsels zwischen den Zuständen *sperrend* und *leitend* deutliche Leistungen abfallen. Abbildung 3.6 zeigt die Leistung an der Drain-Source-Strecke in Abhängigkeit der Gatespannung in einer Spice-Simulation. Die simulierte Schaltung (Spice-Beschreibung im Anhang A.17) entspricht prinzipiell der Schaltung in Abbildung 3.7.
Zwischen die Versorgungsspannung von 24 V und den Drain-Anschluss des MOSFETs ist der Source-Widerstand R_S mit $10\,\Omega$ als Last geschaltet. Bei zunehmender Gatespannung sinkt der Widerstand zwischen Drain und Source, wobei zunächst die abfallende Leistung am MOSFET auf über 14 W ansteigt. Bei noch höherer Gatespannung nimmt dann die am MOSFET abfallende Leistung wieder ab.

Einige Berechnungen sollen zeigen, wie sich unterschiedliche Schaltströme auf die Schaltzeiten und somit auf den Leistungsabfall am MOSFET auswirken. Je länger der Übergang zwischen den Zuständen *sperrend* und *leitend* dauert, desto mehr Energie wird am Transistor in Wärme umgesetzt. Die zugrundeliegende Schaltung entspricht wiederum der schon erwähnten Prinzipschaltung (Abb. 3.7), allerdings besitzen die Bauteile andere Werte als in der Simulation.
Zunächst werden die ungefähren Schaltzeiten des MOSFETs bei Verwendung verschiedener Ansteuerungen berechnet. Für die Berechnung der Schaltzeit kann die folgende Formel herangezogen werden:

$$t = \frac{Q}{I}$$

Q ist die benötigte Ladung am Gate des MOSFETs. Beim verwendeten Si7852DP zeigt das Datenblatt (Vishay, 2007, Seite 3, Gate Charge; Das Durchschalten erfolgt in dem horizontal verlaufenden Bereich), dass der Schaltvorgang bei einem Drain-Source-Strom $I_{DS} = 10\,\mathrm{A}$ zwischen etwa 7,5 nC und 20 nC abläuft, somit also 12,5 nC Ladung transportiert werden müssen.



X-Achse: U_{GS}; Y-Achse: P_{DS}

Abbildung 3.6: Schaltverhalten am MOSFET

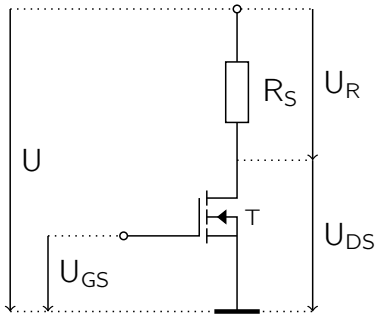


Abbildung 3.7: Ersatzschaltbild der MOSFET-Schaltung

I ist der Schaltstrom, hierfür werden die maximalen Ströme der drei oben genannten Ansteuerungsarten (Mikrocontroller, 5 $\times$ 74AC04, LM5112SD) eingesetzt.
Der LM5112SD kann 3 A liefern (Source Current) aber 7 A aufnehmen (Sink Current), daher unterscheiden sich die Ein- und Ausschaltzeit.

$$t_{ATtiny861V} = \frac{12{,}5\,\text{nC}}{40\,\text{mA}} = \frac{12{,}5 \cdot 10^{-9}\,\text{C}}{40 \cdot 10^{-3}\,\text{A}} = \frac{12{,}5}{40} \cdot 10^{-6}\,\text{s} = 0.3125\,\mu\text{s}$$

$$= 312{,}5\,\text{ns}$$

$$t_{74AC04} = \frac{12{,}5\,\text{nC}}{250\,\text{mA}} = \frac{12{,}5 \cdot 10^{-9}\,\text{C}}{250 \cdot 10^{-3}\,\text{A}} = \frac{12{,}5}{250} \cdot 10^{-6}\,\text{s} = 0.05\,\mu\text{s}$$

$$= 50\,\text{ns}$$

$$t_{ein\,LM5112SD} = \frac{12{,}5\,\text{nC}}{3\,\text{A}} = \frac{12{,}5 \cdot 10^{-9}\,\text{C}}{3\,\text{A}} = \frac{12{,}5}{3} \cdot 10^{-9}\,\text{s}$$

$$= 4{,}17\,\text{ns}$$

$$t_{aus\,LM5112SD} = \frac{12{,}5\,\text{nC}}{7\,\text{A}} = \frac{12{,}5 \cdot 10^{-9}\,\text{C}}{7\,\text{A}} = \frac{12{,}5}{7} \cdot 10^{-9}\,\text{s}$$

$$= 1{,}79\,\text{ns}$$

Im Datenblatt des Si7852DP sind keine genauen Werte für den Drain-Source-Widerstand angegeben, daher basieren die folgenden Berechnungen auf ungefähren Werten. Dem Datenblatt kann ab einer Gate-Spannung von etwa 5 V, also im leitenden Zustand, ein ungefährer Widerstand $R_{DS(on)} = 20\,\text{m}\Omega$ entnommen werden. Für den nichtleitenden Zustand findet man im Datenblatt keine Angaben, hier wird ein Widerstand von $R_{DS(off)} = 10\,\text{M}\Omega$ angenommen.
Die abfallende Leistung am leitenden MOSFET beträgt $P = R_{DS} \cdot I^2 = 20\,\text{m}\Omega \cdot (10\,\text{A})^2 = 2000\,\text{mW}$ bei einem Spannungsabfall von 200 mV. Bei einer Versorgungsspannung von 40 V ergibt sich ein Gesamtwiderstand von $R_G = \frac{U}{I} = \frac{40\,\text{V}}{10\,\text{A}} = 4\,\Omega$. Der Widerstand, der mit dem Transistor in Reihe geschaltet ist – entweder als normaler Widerstand oder beispielsweise als Serienwiderstand der Induktivität bei konstantem Strom – beträgt demzufolge $3{,}98\,\Omega$.
Wenn der MOSFET sperrt, ist der fließende Strom deutlich geringer. Der Gesamtwiderstand beträgt $R_L + R_{DS} = 3{,}98\,\Omega + 10\,\text{M}\Omega = 10000003{,}98\,\Omega$. Daraus ergibt sich ein Strom von $\frac{40\,\text{V}}{10000003{,}98\,\Omega} \approx 40\,\text{nA}$.

Am MOSFET fällt nun eine Leistung von $P = R_{DS} \cdot I^2 = 10\,\mathrm{M\Omega} \cdot (40\,\mathrm{nA})^2 = 16\,\mathrm{nW}$ ab.
Als dritter Fall wird nun betrachtet, dass der MOSFET nur teilweise durchschaltet und einen Widerstand von $R_{DS} = 3{,}98\,\Omega$ darstellt, der genau so groß ist wie der in Reihe geschaltete Widerstand. Nun fließt ein Strom von $I = \frac{U}{R} = \frac{40\,\mathrm{V}}{7{,}96\,\Omega} = 5{,}03\,\mathrm{A}$ und somit fällt eine Leistung von $P = R_{DS} \cdot I^2 = 3{,}98\,\Omega \cdot (5{,}03\,\mathrm{A})^2 = 100{,}5\,\mathrm{W}$ am Transistor ab. Laut Datenblatt darf der Transistor mit dieser Leistung nur während eines Pulses von etwa 15 ms Länge belastet werden. Um die Schaltverluste gering zu halten, sollte der Bereich möglichst schnell durchlaufen werden. Um das zu erreichen, muss das Gate mit einem hohen Strom angesteuert werden.
Um den Leistungsabfall am Transistor in Abhängigkeit des Drain-Source-Widerstands R_{DS} für einen gegebenen Serienwiderstand R_S und eine gegebene Spannung U zu berechnen ergibt sich die folgende Formel:

$$P_{[MOSFET]} = R_{DS} \cdot \left(\frac{U}{R_{DS} + R_S}\right)^2$$

Die maximale Leistung lässt sich über die Ableitung der Funktion der Leistung nach dem Drain-Source-Widerstand bestimmen (komplette Rechnung in Anhang D).

$$\begin{aligned}
f(R_{DS}) &= P_{[MOSFET]}(R_{DS}) \\
f(R_{DS}) &= R_{DS} \cdot \left(\frac{U}{R_{DS} + R_S}\right)^2 = g(R_{DS}) \cdot h(R_{DS}) \\
g(R_{DS}) &= R_{DS} \qquad\qquad h(R_{DS}) = \left(\frac{U}{R_{DS} + R_S}\right)^2 \\
f'(R_{DS}) &= g'(R_{DS}) \cdot h(R_{DS}) + g(R_{DS}) \cdot h'(R_{DS}) \\
&= 1 \cdot \left(\frac{U}{R_{DS} + R_S}\right)^2 + R_{DS} \cdot \left(\left(\frac{U}{R_{DS} + R_S}\right)^2\right)' \\
&= \frac{U^2 \cdot (R_S - R_{DS})}{(R_{DS} + R_S)^3}
\end{aligned}$$

$$f'(R_{DS}) = 0 \quad | \quad R_{DS} = R_S$$

Es ergibt sich somit ein maximaler Leistungsabfall am MOSFET bei $R_{DS} = R_S$.

3.1.3 12 V Step-Down-Wandler

Der 12 V-Step-Down-Wandler (Abb. 3.8) entspricht in der Struktur der Grundschaltung in Abb. 2.2. Für den eingesetzten MOSFET wird ebenfalls ein Treiber benötigt. Die Induktivität dieses Wandlers ist kleiner als beim Step-Up-Wandler.

Für eine Spannungshalbierung von 24 V auf 12 V wird ebenfalls ein PWM-Signal mit einem Tastverhältnis von 50 % benötigt, die Berechnung erfolgt aber etwas anders: $U_a = U_e \cdot \frac{t_{ein}}{T}$. Das Verhältnis der Spannungen entspricht also dem verwendeten Tastverhältnis. Diese Formel gilt aber ebenfalls nur ab einer Belastung mit einem minimalen Strom $I_{a\,min}$. Dieser berechnet sich wie folgt:

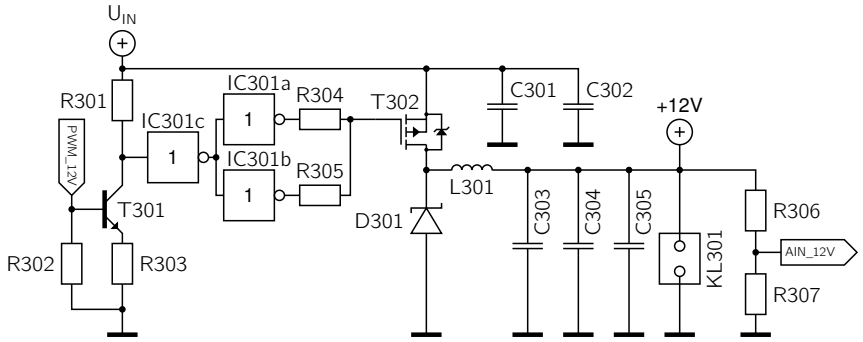
$$I_{a\,min} = \frac{T}{2 \cdot L} \cdot U_a \cdot \left(1 - \frac{U_a}{U_e}\right)$$

Für den konkreten Wandler, lassen sich Minimalstrom und die nötige Kapazität am Ausgang folgendermaßen bestimmen:

$$\begin{aligned} I_{a\,min} &= \frac{T}{2 \cdot L} \cdot U_a \cdot \left(1 - \frac{U_a}{U_e}\right) \\ &= \frac{1\,\mu\text{s}}{2 \cdot 22\,\mu\text{H}} \cdot 12\,\text{V} \cdot \left(1 - \frac{12\,\text{V}}{24\,\text{V}}\right) \\ &= \frac{12}{88}\,\frac{\text{V} \cdot \text{s}}{\text{H}} = \frac{3}{22}\,\text{A} = 136{,}36\,\text{mA} \end{aligned}$$

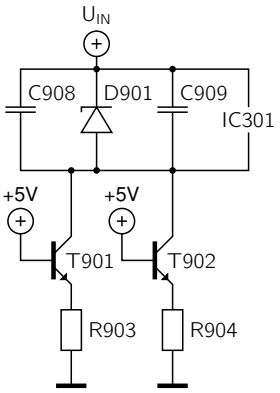
$$\begin{aligned} C &= \frac{T \cdot I_{a\,min}}{4 \cdot \Delta U_a} \\ &= \frac{1\,\mu\text{s} \cdot 136{,}36\,\text{mA}}{4 \cdot 12\,\text{mV}} \\ &= \frac{136{,}36}{48}\,\mu\text{F} = 2{,}84\,\mu\text{F} \end{aligned}$$

Durch den Einsatz einer höheren Kapazität von 4,5 µF wird hier ebenfalls die Restwelligkeit weiter reduziert.
Für die Ansteuerung des PMOS Schalttransistors T302 wird das vom Mikrocontroller gelieferte PWM-Signal mit einer Treiberschaltung (R301-R305, T301, IC301) angepasst. Der Sechsfach-Inverter 74AC04 (IC301) wird durch die Schaltung aus Abb. 3.9 mit einer Spannung versorgt, deren Masseniveau durch die Zenerdiode D901 5,1 V unter der Versorgungsspannung der Schaltung liegt. Gleichzeitig wird der Strom durch die Transistoren T901 und T902 sowie die Widerstände R903 und R904 begrenzt. Der Strom berechnet sich folgendermaßen:
Zwischen der Basis eines Transistors und Masse liegt eine Spannung von 5 V an. Im Transistor fallen zwischen Basis und Emitter etwa 0,7 V ab, am Emitterwiderstand liegen somit noch 4,3 V an. Nun muss nur noch der Strom durch den Widerstand berechnet werden. Dieser beträgt $I = \frac{U}{R} = \frac{4{,}3\,\text{V}}{2{,}2\,\text{k}\Omega} = 0{,}00195454\,\text{A} \approx 1{,}95\,\text{mA}$. Durch die doppelte Ausführung ergibt sich der doppelte Strom von ca. 3,9 mA.



C301, C305	100 nF	L301	22 µH
C302	4,7 µF	T301	BC846
C303, C304	2,2 µF	T302	Si4948BEY
R301, R302, R303, R306	2,2 kΩ	D301	SS16
R304, R305	47 Ω	IC301	74AC04
R307	1 kΩ		

Abbildung 3.8: Step-Down-Wandler nach 12 V



C908	220 nF
C909	100 nF
R903, R904	2,2 kΩ
T901, T902	BC846
D901	Zenerdiode 5,1V
IC301	74AC04

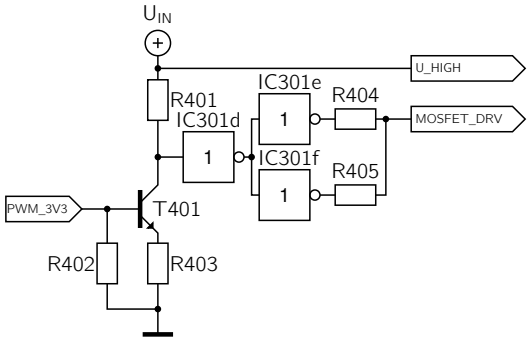
Abbildung 3.9: Versorgungsschaltung für den 74AC04 als MOSFET-Treiber

3.1.4 3,3 V Step-Down-Wandler

Durch die Verwendung der SPI-Schnittstelle, die für weitere Anwendungen bevorzugt wird, wird unter anderem der PWM-Ausgang des Mikrocontrollers benötigt, der ursprünglich für den 3,3 V Step-Down-Wandler vorgesehen war. Auf dessen Umsetzung wurde daher verzichtet, da für die geringen benötigten Ausgangsleistungen auch fertige Schaltregler oder Linearregler eingesetzt werden können. Dennoch soll hier die geplante Dimensionierung gezeigt werden.
Es waren zwei unterschiedliche Bestückungsvarianten für den auch hier nötigen MOSFET-Treiber vorgesehen. Die erste (Abb. 3.10) war vom zusätzlichen Hardwareaufwand vorteilhaft, da die beim 12 V-Step-Down-Wandler ungenutzten drei Gatter des 74AC04 (IC301) genutzt werden sollten. Der weitere Aufbau des MOSFET-Treibers ist äquivalent zu dem des 12 V Step-Down-Wandlers.

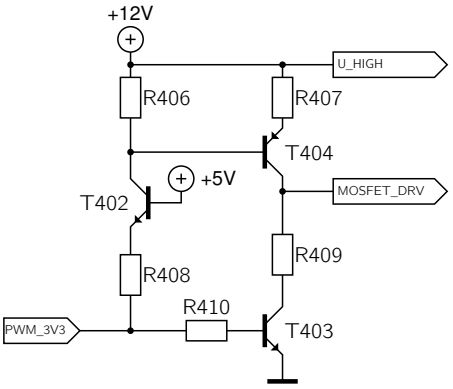
Die zweite Bestückungsvariante (Abb. 3.11) stellt einen Pegelwandler dar, der aus dem TTL-kompatiblen Signal des Mikrocontrollers ein Signal mit einem High-Pegel von 12 V erzeugt. Damit wird eine Eingangsspannung von 12 V für den Step-Down-Wandler ermöglicht, die in der Schaltung ohnehin erzeugt wird. Der Vorteil hierbei ist ein Tastverhältnis der PWM, das statt bei 13,75 % bei elektrisch günstigeren 27,5 % liegt. Der Rest des Step-Down-Wandlers ist in Abbildung 3.12 dargestellt.

Der Minimalstrom $I_{a\,min}$ sowie die nötige Kapazität C am Ausgang des Wandlers für eine Restwelligkeit von 0,1 % der Ausgangsspannung werden analog zum 12 V-Wandler berechnet. Es ergeben sich für die erste Variante des MOSFET-Treibers Werte von $I_{a\,min} = 64{,}69\,\text{mA}$ und $C = 1{,}35\,\mu\text{F}$ sowie für die zweite Variante des MOSFET-Treibers $I_{a\,min} = 54{,}38\,\text{mA}$ und $C = 1{,}13\,\mu\text{F}$. Die vorgesehene Kapazität von 2,3 µF hätte also ebenfalls für eine Welligkeit unter 0,1 % gesorgt.



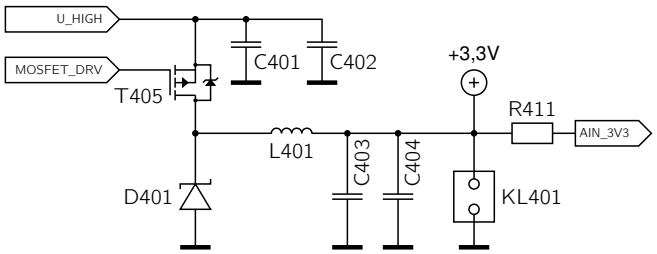
R401, R402, R403	2,2 kΩ
R404, R405	33 Ω
T401	BC846
IC301	74AC04

Abbildung 3.10: MOSFET-Treiber für den 3,3 V-Wandler, Variante 1



R406, R408, R410	2,2 kΩ
R407, R409	33 Ω
T402, T403	BC846
T404	BC856

Abbildung 3.11: MOSFET-Treiber für den 3,3 V-Wandler, Variante 2

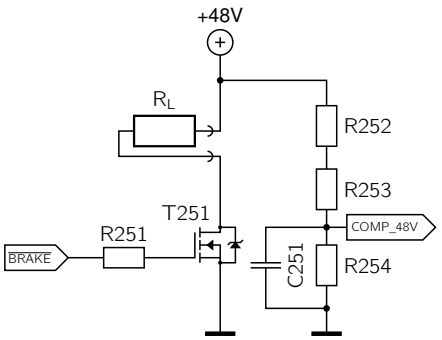


C401, C404	100 nF
C402	4,7 µF
C403	2,2 µF
R411	4,7 kΩ
L401	22 µH
T405	Si4948BEY
D401	SS16

Abbildung 3.12: Step-Down-Wandler nach 3,3 V

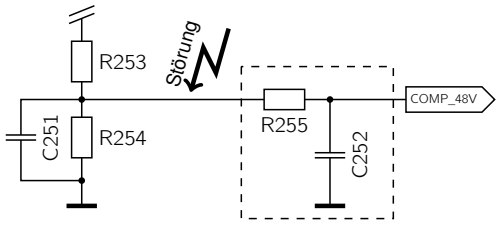
3.1.5 Schaltbarer Lastwiderstand

Die Steuerung (Abb. 3.13) für den externen Lastwiderstand R_L dient dazu, Überspannungen abzubauen. Diese können auftreten, wenn ein Motor, der mit den 48 V versorgt wird durch Induktion eine Spannung zurück in den Step-Up-Wandler speist. Außerdem können Schäden an der Elektronik verhindert werden, wenn durch einen Fehler in der Regelung die Spannung zu weit ansteigt.
Die verwendeten Glättungskondensatoren besitzen eine maximale Betriebsspannung von 63 V, mit einem zusätzlichen Sicherheitsbereich wurde eine maximale Spannung von 55 V veranschlagt. Wenn diese überschritten wird, schaltet der Mikrocontroller über den Transistor T251 einen angeschlossenen Leistungswiderstand R_L zwischen den 48 V Ausgang der Schaltung und Masse. Über diesen wird die überschüssige Energie abgeführt. Außerdem wird für die Dauer der Überschreitung der Schaltschwelle die PWM-Erzeugung für den Step-Up-Wandler ausgesetzt. Die Kontrolle der Spannung erfolgt über einen Spannungsteiler (R252-R254), der bei einer Spannung von 55 V eine Spannung von 1,1 V am Komparator-Pin des ATtiny861V liefert. Dies ist die Spannung der internen Bandgap-Referenz, so dass der Komparator-Interrupt zum schnellen Schalten des Transistors T251 genutzt werden kann.
Im Aufbau der Schaltung zeigte sich hier ein Problem, da das Signal zwar größtenteils unter dem Grenzwert liegt, bei den Schaltvorgängen aber von der Induktivität des Step-Up-Wandlers starke Störungen in die Leiterbahn eingespeist werden, so dass der Interrupt fälschlicherweise ausgelöst wird. Bedingt durch die hohe Schaltfrequenz befindet sich der Mikrocontroller dann nahezu dauerhaft in der ISR des Komparators, da der Interrupt schneller wieder auftritt als die ISR abgearbeitet wird.
Der Kondensator C251 am Spannungsteiler und ein RC-Tiefpassfilter (Abb. 3.1.5) direkt vor dem Eingang des Mikrocontrollers, bestehend aus dem $4{,}7\,\mathrm{k}\Omega$ Widerstand R255 und dem 680 pF Kondensator C252, dämpfen die Störungen ausreichend stark, lassen aber trotzdem schnelle Reaktionen auf Überspannungen zu.



R_L	Lastwiderstand
R251, R254	100 Ω
R252	200 Ω
R253	4,7 kΩ
C251	1 nF
T251	Si4946EY

Abbildung 3.13: Ansteuerung des Lastwiderstands



R253, R255	4,7 kΩ
R254	100 Ω
C251	1 nF
C252	680 pF

Abbildung 3.14: RC-Tiefpass zur Störungsfilterung

3.2 Software – Mikrocontrollerprogrammierung

Hier werden verschiedene Aspekte der Mikrocontrollerprogrammierung betrachtet, die für den Betrieb der Schaltung benötigt werden. Soweit es möglich war, wurden die Beispiele im Simulator des AVR Studios überprüft. Teilweise war dies aber nicht möglich, da beispielsweise die Timer des ATtiny861 in der aktuellen Version (4.13 Service Pack 2) nicht simuliert werden. Informationen zu den Einschränkungen des Simulators für die verschiedenen Mikrocontroller sind in der Hilfe des AVR-Studios aufgeführt (im „AVR Tools User Guide“ unter Simulator/Known Issues/Notes for . . .). Die Beispiele sollen auch nur grundlegend zeigen, wie die Komponenten zu programmieren sind. Sie basieren auf der Registerbelegung des ATtiny861 (siehe Datenblatt (Atmel, 2006)), bei anderen AVR-Mikrocontrollern können die verfügbaren bzw. benötigten Register sowie deren Belegung abweichen. Der Quellcode, der letztendlich im Mikrocontroller verwendet wird, wird im Abschnitt 3.3 erläutert und ist ist im Anhang C komplett aufgeführt.

3.2.1 Pulsweitenmodulation (PWM)

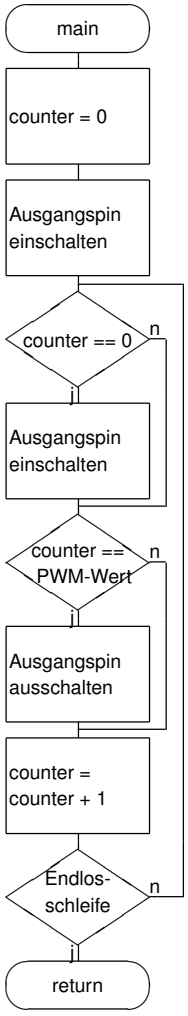
Bei einer Pulsweitenmodulation wird ein Signal erzeugt, das mit einer in der Regel festen Frequenz und einem veränderlichen Tastverhältnis zwischen HIGH- und LOW-Pegel wechselt. Mit dem PWM-Signal werden die Step-Up- bzw. Step-Down-Wandler angesteuert. In der einfachsten Form wird das Signal zu Beginn einer Periode auf HIGH und zum gewünschten Zeitpunkt wieder auf LOW gesetzt, ein anderes Verfahren arbeitet beispielsweise genau invertiert. Bei den Atmel AVR Mikrocontrollern kann ein solches Signal auf verschiedene Arten erzeugt werden.

Soft-PWM I

Ein Beispielprogramm hierzu befindet sich im Anhang B.1.
Die einfachste Variante lässt sich komplett in Software realisieren. Sie besteht aus einer Endlosschleife, die bei Mikrocontrollerprogrammen meistens ohnehin genutzt wird oder wenigstens als leere Schleife am Ende des Hauptprogramms vorhanden ist. Darin wird eine Zählvariable bei jedem Durchlauf inkrementiert und mit dem gewünschten Wert verglichen. Ist dieser Wert erreicht, wird der Ausgangspin auf LOW geschaltet. Wenn die Zählvariable überläuft oder einen festgelegten Maximalwert erreicht und vom Programm auf 0 gesetzt wird, wird der Ausgangspin wieder auf HIGH gesetzt. Der Wert für die Umschaltung muss entweder konstant sein, mit einem zusätzlichen Zähler nach einer bestimmten Zahl von Perioden geändert werden oder in einer globalen Variable gespeichert sein, damit er durch eine ISR geändert werden kann.

Die Grundfrequenz dieser Lösung ist von mehreren Faktoren abhängig:

- Der Taktfrequenz des Mikrocontrollers
- Dem vom Compiler erzeugten Assemblercode
- Der Anzahl der PWM-Kanäle und der nötigen Vergleiche
- Der Häufigkeit von Interrupts



Daher lässt sich eine genaue Vorhersage der Frequenz nicht ohne weiteres durchführen. Beim Beispiel verursacht schon eine Änderung der Compileroptimierungseinstellung Frequenzen von ca. 1,6 kHz bei abgeschalteter Optimierung und ca. 2,4 kHz bei Optimierung nach Codegröße bei gleicher Taktfrequenz von 8 MHz. Dieses einfache Verfahren kann aber sehr gut verwendet werden, wenn z. B. nur die Helligkeit von Leuchtdioden geregelt werden soll. Für Anwendungen, in denen es auf präzise oder hohe Frequenzen ankommt, ist es nicht geeignet.
Vorteile dieser Lösung sind:

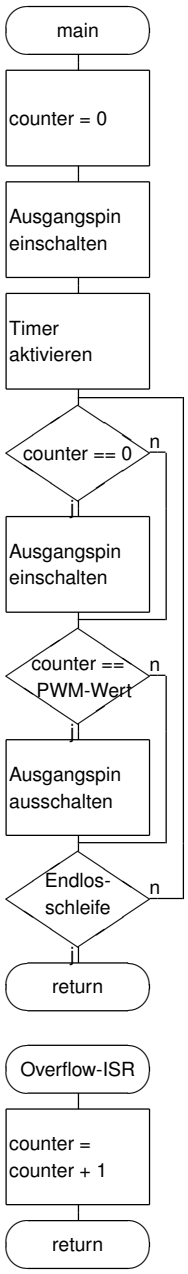
- Sie lässt sich einfach umsetzen
- Jeder I/O-Pin kann als PWM-Ausgang genutzt werden
- Für die PWM-Erzeugung sind keine ISRs nötig

Nachteile dieser Lösung sind:

- Die Frequenz kann durch andere Interrupts schwanken
- Das Tastverhältnis kann durch andere Interrupts schwanken (Jitter)
- Das Signal wird komplett von der CPU erzeugt, wodurch diese stark belastet wird

Soft-PWM II

Ein Beispielprogramm hierzu befindet sich im Anhang B.2.
Die zweite Möglichkeit, ein PWM-Signal zu erzeugen unterscheidet sich zur ersten darin, dass zum Inkrementieren des Zählers eine Timer-ISR genutzt wird, die die Variable regelmäßig inkrementiert. Dadurch wird das Schwanken der Frequenz nahezu verhindert, nur durch Überlagerung von mehreren IRQs und deren Bearbeitung kann es zu Verzögerungen kommen. Dieser Vorteil wird allerdings durch eine reduzierte Frequenz erkauft. Diese Variante funktioniert nur zuverlässig, wenn sichergestellt ist, dass die Endlosschleife mit den Vergleichen zum Ein- und Ausschalten des Ausgangspins mindestens einmal zwischen zwei Aufrufen der Timer-ISR durchläuft.
In diesem Fall lässt sich die Frequenz des erzeugten PWM-Signals aus der Taktfrequenz, der Vorteilung des Timers und der Größe des Timers sowie der Zählvariablen berechnen. Bei einer 8 Bit-Variablen und einem 8 Bit-Timer läuft die Zählvariable nach $2^8 \cdot 2^8 = 2^{16}$ Takten über. Bei einem Vorteiler von 1 und einer Taktfrequenz von 8 MHz ergibt sich so eine PWM-Frequenz von ca. 122 Hz.



Vorteile dieser Lösung sind:

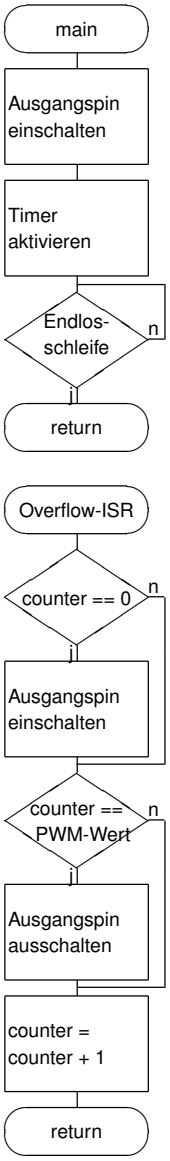
- Sie lässt sich ebenfalls einfach umsetzen
- Auch hier kann jeder I/O-Pin als PWM-Ausgang genutzt werden

Nachteile dieser Lösung sind:

- Die Frequenz ist gegenüber der ersten Lösung niedriger
- Es wird ein Timer benötigt
- Die CPU wird wie bei der ersten Lösung stark belastet
- Das Hauptprogramm kann „verhungern“ und Schaltvorgänge auslassen

Soft-PWM III

Ein Beispielprogramm hierzu befindet sich im Anhang B.3.
Diese Variante unterscheidet sich von der vorigen dadurch, dass die Schaltvorgänge zur Signalerzeugung ebenfalls in der Timer-ISR ausgeführt werden. Dadurch wird bei jedem Inkrementieren des Zählers geprüft, ob geschaltet werden muss und es werden alle Zählerstände berücksichtigt. Es kann aber noch zu Fehlern kommen, wenn der Zähler niedriger als der Abschaltwert pwm_val, der Ausgang also eingeschaltet ist und pwm_val dann auf einen Wert unter dem Zählerstand gesetzt wird. In diesem Fall wird der Ausgang erst beim nächsten Durchlauf abgeschaltet. Als Lösung für dieses Problem wäre es möglich, eine Kopie von pwm_val lokal (und static) in der ISR zu halten und mit diesem Wert zu vergleichen, Änderungen an pwm_val aber nur synchron mit dem Einschalten des Ausgangs beim Zählerstand von 0 zu übernehmen. Dadurch wirkt sich die Änderung erst in der nächsten Periode aus.
Die PWM-Frequenz berechnet sich hier genauso wie im vorigen Fall.



Vorteile dieser Lösung sind:

- Auch diese Lösung lässt sich einfach umsetzen
- Es kann ebenfalls jeder I/O-Pin als Ausgang genutzt werden

Nachteile dieser Lösung sind:

- Die Frequenz ist wie bei der zweiten Lösung niedrig
- Es wird wie bei der zweiten Lösung ein Timer benötigt

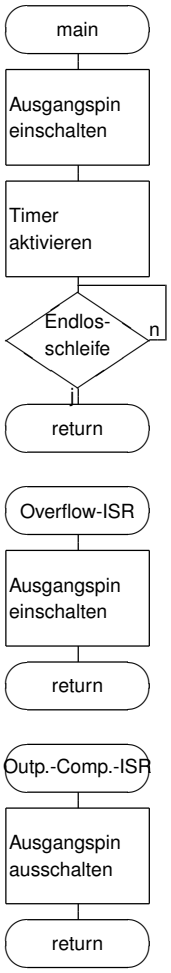
Realisierung über Counter mit Output-Compare-Interrupt

Ein Beispielprogramm hierzu befindet sich im Anhang B.4.
Die Timer/Counter der AVR-Mikrocontroller verfügen über sogenannte Output-Compare-Units. Es handelt sich dabei um Einheiten, die den Wert des Zählerregisters mit dem Wert eines weiteren Registers vergleichen und bei Gleichheit beispielsweise einen Interrupt auslösen. Diese Funktion wird hier zusammen mit dem Interrupt beim Überlauf des Zählers genutzt. Beim Überlauf wird der PWM-Ausgangspin gesetzt und beim Output-Compare-Interrupt wieder gelöscht. Das geschieht über zwei ISRs, wird also durch die CPU vorgenommen.
Hier hängt die PWM-Frequenz nur vom Takt, dem Vorteiler und der Größe des Zählers ab. Bei einem 8 Bit-Zähler und einem Vorteiler von 1 bei 8 MHz ergibt sich eine Frequenz von 31,25 kHz.
Vorteile dieser Lösung sind:

- Die Realisierung ist einfach
- Jeder I/O-Pin kann als Ausgang genutzt werden
- Die CPU-Last ist niedriger als bei den vorigen Verfahren

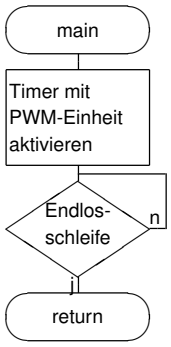
Nachteile dieser Lösung sind:

- Es wird ein Timer mit Output-Compare-Einheit benötigt
- Durch Überlagerung von mehreren IRQs kann es zu Jitter kommen



Realisierung über Counter mit PWM-Einheit (Hardware-PWM)

Ein Beispielprogramm hierzu befindet sich im Anhang B.5.
Fast alle Mikrocontroller der AVR-Familie verfügen über mindestens einen Hardware-PWM-Kanal. Hierbei wird die komplette Erzeugung des Signals nach vorheriger Konfiguration von der Hardware übernommen. Es wird für die PWM-Erzeugung keine ISR und außer für die Initialisierung keine CPU-Leistung benötigt. Allerdings muss für jeden PWM-Kanal ein bestimmter Ausgangspin genutzt werden.
Wie im vorigen Fall hängt hier die PWM-Frequenz nur vom Takt, dem Vorteiler und der Größe des Zählers ab. Es ergibt sich bei den obigen Daten ebenfalls eine Frequenz von 31,25 kHz.



Vorteile dieser Lösung sind:

- Sie lässt sich sehr einfach umsetzen
- Die CPU wird nach der Initialisierung für die Signalerzeugung nicht mehr benötigt
- Es tritt kein Jitter auf
- Es ist eine hohe Frequenz möglich

Nachteile dieser Lösung sind:

- Es wird ein Timer mit PWM-Einheit benötigt
- Es können nur bestimmte Pins (OCnx) als PWM-Ausgang verwendet werden
- Die Auswahl der möglichen Frequenzen ist bei einigen Mikrocontrollern eingeschränkt

Fazit

Aufgrund der Einfachheit und Stabilität der Hardware-PWM-Lösung (Abschnitt 3.2.1) ist diese für eine präzise Steuerung zu bevorzugen. Außerdem bietet der ATtiny861 die Möglichkeit, den Counter mit einer Frequenz von 64 MHz zu takten, wodurch im Vergleich zu anderen Bausteinen der AVR-Familie höhere Frequenzen erreicht werden können. Der Counter 1 des ATtiny861 verfügt auch über drei PWM-Ausgänge und zusätzlich ein Register, um den maximalen Zählerwert festzulegen. Für die Anwendung im Schaltregler ist er daher sehr gut geeignet.
Es wird ein maximaler Zählerwert von 63 (6-Bit Zähler) genutzt, so dass die PWM-Frequenz bei 1 MHz liegt. Dies ist ein guter Kompromiss zwischen den elektrischen Eigenschaften der Schaltung und der Feinheit, in der die PWM einstellbar ist.

Über einen zusätzlichen in Software realisierten Zähler, wird eine Erweiterung auf 8 Bit implementiert, ohne die PWM-Frequenz von 1 MHz zu verringern. Dazu werden die oberen 6 Bit des 8-Bit-PWM-Wertes als Vergleichswert des Hardwarezählers verwendet und bestimmen das Tastverhältnis, die unteren zwei Bit dienen der Entscheidung, ob in einem Durchlauf des 2-Bit-Softwarezählers der Vergleichswert keinmal, einmal, zweimal oder dreimal um eins erhöht in das Output-Compare-Register geschrieben wird und ein erhöhtes Tastverhältnis verursacht. Über den Zeitraum eines kompletten Durchlaufs des Softwarezählers ($0 \ldots 3$) ergibt sich so ein Mittelwert des Tastverhältnisses, das dem Tastverhältnis eines 8-Bit-Zählers entspricht. Der Softwarezähler wird bei jedem Aufruf der AD-Wandler-ISR inkrementiert.
Die Tabelle 3.2 stellt diese Zuordnung dar. Die Einträge sind binär, wobei das x als „don't care“ zu verstehen ist und in das Output-Compare-Register übernommen wird (ggf. um eins erhöht).

Softwarezähler PWM-Wert	00	01	10	11
xxxxxx00	xxxxxx	xxxxxx	xxxxxx	xxxxxx
xxxxxx01	xxxxxx	xxxxxx	xxxxxx + 1	xxxxxx
xxxxxx10	xxxxxx	xxxxxx + 1	xxxxxx	xxxxxx + 1
xxxxxx11	xxxxxx + 1	xxxxxx + 1	xxxxxx	xxxxxx + 1

Tabelle 3.2: PWM-Erweiterung

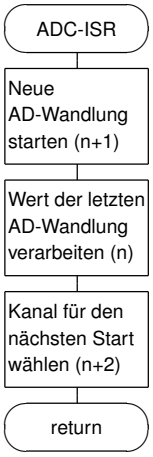
3.2.2 AD-Wandler

Ein Beispielprogramm hierzu befindet sich im Anhang B.6.
Der AD-Wandler im ATtiny861 verfügt über 11 Eingangskanäle sowie einen Kanal für eine interne Referenzspannung von 1,1 V, einen Kanal für die Analog-Masse (AGND) und einen Kanal mit einem internen Temperatursensor. Es besteht die Möglichkeit, einen dieser Kanäle direkt gegen AGND (Single Ended Input) oder die Spannungsdifferenz zwischen zwei Eingängen (Differential Input), ggf. auch mit einer Verstärkung, zu messen. Für den Schaltregler werden vier Single Ended Input Kanäle verwendet. Als Referenzspannung stehen entweder die Betriebsspannung, eine externe Referenzspannung oder eine interne Referenzspannung von 1,1 V oder 2,56 V zur Verfügung.
Der zulässige Frequenzbereich für den Takt des AD-Wandlers liegt zwischen 200 kHz und 1 MHz, wobei die Genauigkeit mit steigender Frequenz abnimmt. Das Verhalten bei höheren Frequenzen ist im Datenblatt nicht spezifiziert.

Die Auflösung des Wandlers beträgt 10 Bit, von denen aber nur die oberen 8 genutzt werden. Einerseits weil bei einer Taktung des AD-Wandlers mit 1 MHz ohnehin nur eine Genauigkeit von ± 3 LSB erreicht wird, die drei niedrigstwertigen Bits also nicht zwingend korrekt sind, andererseits um bei der Verarbeitung Zeit zu sparen, indem nur mit 8 Bit-Variablen gearbeitet wird. Mit der Funktion, die acht höchstwertigen Bits komplett in das ADCH-Register zu schieben (ADLAR: ADC Left Adjust Result), kann man diese mit einem Zugriff auslesen. Durch Verwendung des Noise-Reduction-Mode, bei dem u.a. die CPU angehalten wird, ließe sich die Genauigkeit auf $\pm 2{,}5$ LSB erhöhen. Da während der Wandlung aber auch Berechnungen durchgeführt werden müssen, kommt der Einsatz nicht in Frage.
Die Wandlungen können einzeln durch die Software oder durch Triggerereignisse gestartet werden sowie im Free-Running-Mode direkt nacheinander ausgeführt werden. Für die erste Wandlung werden 25 Taktzyklen, für jede weitere 13 Taktzyklen benötigt. Bei einem Takt von 1 MHz entspricht das 25 µs bzw. 13 µs. Im Free-Running-Mode folgt auf den letzten Takt einer Wandlung direkt der erste Takt der nächsten. Bei softwareseitig gestarteten Wandlungen vergeht eine kurze Zeit, in der beispielsweise die CPU in die ISR springt, die aufgerufen wird, wenn eine Wandlung beendet ist (ADC Conversion Complete), um dort dann die nächste Wandlung zu starten. Bei 1 MHz Takt im Free-Running-Mode wird das Signal also mit $\frac{1\,\mathrm{MHz}}{13} = 76{,}923\,\mathrm{kHz}$ abgetastet, bei Wandlungen, die in der ISR gestartet werden, liegt die Abtastfrequenz etwas niedriger.
Da die Kanäle über einen Multiplexer geschaltet sind, muss zwischen den Messungen der Kanal gewechselt werden. Dies ist im Free-Running-Mode nur schwer möglich, da zu dem Zeitpunkt, an dem man ein Wandlungsergebnis ausliest schon die nächste Wandlung gestartet wurde und nicht festgestellt werden kann, von welchem Kanal das Ergebnis stammt. Bei bekannter Reihenfolge kann natürlich der Kanal ermittelt werden, wenn aber einmal ein Ergebnis nicht ausgelesen wird, ist die korrekte Zuordnung nahezu unmöglich.

Für den Schaltregler wurde daher ein anderer Ansatz gewählt:
In der ADC Conversion Complete ISR wird zunächst die nächste Wandlung gestartet. Bei der nächsten steigenden Flanke des ADC-Taktes, also spätestens nach 1 µs wird die Kanalauswahl übernommen. Da dieser Zeitraum bei niedrigen CPU-Taktfrequenzen nicht genügt, um den Wert im Register zu ändern, scheidet die Möglichkeit aus, als erstes die Wandlung zu starten und direkt danach den Kanal auszuwählen. Gleichzeitig soll aber der Zeitraum zwischen dem Ende einer Wandlung und dem Start der nächsten möglichst gering sein, daher muss der Start der Wandlung als erste Anweisung in der ISR ausgeführt werden, die Kanalauswahl muss also vor dem Betreten der ISR getroffen sein.

Dies wird dadurch erreicht, dass sobald die Wandlung n abgeschlossen ist, die ISR aufgerufen wird und darin die Wandlung $n+1$ gestartet wird, dann das Ergebnis der Wandlung n ausgewertet wird und zuletzt (nachdem die Wandlung $n+1$ sicher gestartet wurde) die Kanalauswahl für die Wandlung $n+2$ ausgeführt wird. Durch dieses Interleaving entsteht eine möglichst geringe Pause zwischen zwei Wandlungen. In der Pause werden andere ISRs ausgeführt bis ein neuer ADC-IRQ auftritt. Sollten dann noch höher priorisierte IRQs anstehen, werden erst die zugehörigen ISRs ausgeführt. Anschließend springt die CPU in die ADC Conversion Complete ISR.
Der Befehl `asm("nop");` im Beispielprogramm dient nur dazu, in der Simulation mit Einzelschritten beim Debugging anschaulich zwischen Hauptprogramm und ISR zu wechseln.



3.2.3 Analog-Komparator

Ein Beispielprogramm hierzu befindet sich im Anhang B.7.
Der Analog-Komparator des ATtiny861 bietet eine Möglichket, einen binären Vergleich zweier Spannungen durchzuführen. Ist die Spannung am positiven Eingang größer als die Spannung am negativen Eingang, erhält man am Ausgang eine logische 1, ist sie kleiner oder gleich, eine logische 0. Als positive Eingangsspannung ist entweder eine Referenzspannung von 1,1 V oder einer von drei Pins (AIN0–AIN2) wählbar. Als negative Eingangsspannung kann ebenfalls einer der drei Pins AIN0–AIN2 oder einer der AD-Wandler-Eingänge ausgewählt werden, letzteres aber nur, wenn der AD-Wandler selbst nicht genutzt wird. Desweiteren kann eine Schalthysterese von 0 V, 20 mV oder 50 mV ausgewählt werden. Diese verhindert bei den Einstellungen 20 mV und 50 mV ein ständiges Wechseln des Ausgangsbits, wenn die beiden Spannungen nahezu gleich sind und nur um einen geringen Betrag schwanken. Wann ein IRQ ausgelöst wird, kann ebenfalls frei gewählt werden, entweder bei einer steigenden Flanke, d. h. die positive Eingangsspannung hat die negative überschritten, bei einer fallenden Flanke, d. h. die positive Eingangsspannung hat die negative unterschritten, oder bei jeder Flanke.

3.2.4 Kommunikation

Ein Beispielprogramm hierzu befindet sich im Anhang B.8.
Zur Kommunikation zwischen dem Mikrocontroller und beispielsweise einem Steuerrechner ist eine SPI-Schnittstelle vorgesehen. Bei SPI handelt es sich um ein serielles synchrones Übertragungsverfahren, wobei die Übertragung in beide Richtungen parallel abläuft. Es sind außer Masse vier Adern für die Übertragung nötig:

SCK	Serial Clock; der Übertragungstakt
MOSI	Master Out, Slave In; Die Übertragungsleitung vom Master zum Slave
MISO	Master In, Slave Out; Die Übertragungsleitung vom Slave zum Master
$\overline{SS}$	Slave Select; Wird vor Beginn der Kommunikation auf Low gezogen

Tabelle 3.3: SPI-Signalleitungen

Für die $\overline{SS}$ Leitung steht kein Anschluss am USI zur Verfügung, dieser wird stattdessen an den Eingang für einen externen Interrupt angeschlossen. Es wird ein Interrupt bei jeder Flanke ausgelöst und in der ISR geprüft, ob ein Select (fallende Flanke) oder ein Deselect (steigende Flanke) des Chips erfolgte. Nur zwischen fallender und steigender Flanke reagiert der Mikrocontroller auf IRQs des USI.
Als Kommunikationsprotokoll wurde ein Austausch von 4 Byte-Datagrammen vorgesehen. Das erste Byte kann als Adresse verstanden werden, wobei das unterste Bit bestimmt, ob von der Adresse gelesen werden soll (1) oder an die Adresse geschrieben werden soll (0). Es wird so ein „Speicher“ von 128 3-Byte-Worten repräsentiert, wobei nur wenige Adressen genutzt werden.
Das erste Byte, das zurückgeliefert wird, ist immer ein Statusbyte (das Flag-Register), die drei folgenden Bytes sind bei Schreiboperationen immer 0, bei Leseoperationen enthalten sie das 24 Bit-Wort von der gewünschten Adresse.
Da das USI des ATtiny861 nach einer Anzahl von maximal 8 empfangenen Bits einen Interrupt auslöst, muss jedes empfangene Byte einzeln behandelt werden. Dafür wird eine einfache State-Machine genutzt, die zyklisch durch ihre vier Zustände läuft. Jedes empfangene Byte verursacht einen Zustandswechsel. Wenn die $\overline{SS}$ Leitung inaktiv - also HIGH - wird, wird der Zustand (und der Index des nächsten Bytes) auf 0 (Startzustand) gesetzt.
Nachdem das erste Byte empfangen wurde wird geprüft, ob gelesen oder geschrieben werden soll und welche Adresse davon betroffen ist. Im Falle eines Lesezugriffs werden die Daten in einen Sendepuffer kopiert, der mit den nächsten drei Byteübertragungen gesendet wird. Bei Schreibvorgängen muss zunächst die komplette Übertragung abgewartet werden, danach werden die empfangenen Daten gespeichert.

3.2.5 Regelung

Für die Regelung des Step-Up-Wandlers und der Step-Down-Wandler sind Regelalgorithmen notwendig, die auf Basis des Ist-Wertes vom AD-Wandler und des Soll-Wertes möglichst schnell Differenzen durch Anpassung des PWM-Wertes ausgleichen. Sowohl für den Step-Up- als auch für den Step-Down-Wandler ist die Ausgangsspannung proportional zum positiven Tastverhältnis der PWM, d. h. je größer der eingestellte PWM-Wert ist, desto größer ist das Tastverhältnis und desto größer ist die Ausgangsspannung. Die Verwendung von klassischen Regelalgorithmen kann aufgrund der geringen Rechenleistung der Mikrocontroller-CPU nur stark vereinfacht erfolgen. Ein Arbeiten mit Fließkommazahlen für die Parameter der Regelung ist zwar möglich, die Berechnungsdauer liegt aufgrund der fehlenden Fließkommabefehle der CPU aber nicht im sinnvollen Bereich. Für die Regelung würde auch schon die Verwendung von Fixkommazahlen genügen, diese Möglichkeit wurde auch getestet, benötigte aber immer noch zu viel Rechenzeit. Mit einigen Einschränkungen und speziellen Optimierungen lässt sich dennoch eine geeignete Regelung umsetzen.

Rechenzeitoptimierung

Da eine schnelle Berechnung der Änderung nötig ist, sollten auf einem AVR-Mikrocontroller keine Berechnungen mit Fließkommazahlen durchgeführt werden. Bestimmte Faktoren können dennoch einfach duch Schiebeoperationen und Additionen erreicht werden. Faktoren der Art 2^n bzw. 2^{-n} lassen sich durch reine Schiebeoperationen um n Bit nach links bzw. rechts erreichen. Andere Faktoren lassen sich durch Addition erreichen. Beispielsweise lässt sich eine Berechnung $\frac{x}{1{,}6}$ folgendermaßen aufteilen:

$$\frac{x}{1{,}6} = x \cdot \frac{1}{1{,}6} = x \cdot \frac{10}{16} = (x \cdot \frac{2}{16}) + (x \cdot \frac{8}{16}) = (\frac{x}{8}) + (\frac{x}{2}) \approx \underbrace{(\mathrm{x} >> 3) + (\mathrm{x} >> 1)}_{\mathrm{C-Code}}$$

Mit zwei Summanden lassen sich allgemein Faktoren der Art $\frac{2^n+2^m}{2^n \cdot 2^m} = \frac{2^n+2^m}{2^{n+m}}$ darstellen. Im Falle einer Berechnung int8_t y = x/1.6; bindet der Compiler eine Fließkomma-Bibliothek ein, obwohl die Nachkommastellen abgeschnitten werden. Allein für den Sprung in eine Divisionsroutine und die Rückkehr daraus werden schon 7 Taktzyklen benötigt. Die Berechnung int8_t y = (x>>3)+(x>>1); benötigt dagegen nur wenige Assemblerbefehle. Ohne das Kopieren von x aus dem Speicher in ein Register und dem Kopieren von y aus einem Register in den Speicher werden zwei Register und 6 Taktzyklen benötigt (Listing im Anhang B.9).
Der Nachteil dieser Methode ist allerdings, dass bei jedem Summanden eventuell entstandene Nachkommastellen abgeschnitten werden, während die Division bei der

Umwandlung der Fließkommazahl in eine Integerzahl nur einmal die Nachkommastellen verwirft. Dadurch erhöht sich bei einigen Quotienten auch die Abweichung vom korrekten Fließkommaergebnis. Durch das mehrfache Abschneiden der Nachkommastellen wird allerdings teilweise die Abweichung gegenüber dem korrekt gerundeten Ergebnis sowie der Fließkommadivision mit einmaligem Abschneiden der Nachkommastellen vergrößert. Im obigen Beispiel liegt der Betrag der Abweichung für negative x zwischen 0 und 2, dabei überwiegend bei 1, und für positive x zwischen 0 und 1, überwiegend aber bei 0. In vielen Anwendungsfällen werden aber die Vorteile der deutlich höhere Berechnungsgeschwindigkeit den Nachteilen der leichten Ungenauigkeit überwiegen.

Ein weiteres Problem für die klassischen Regelalgorithmen sind die deutlichen Unterschiede des Tastverhältnisses mit und ohne Last am Ausgang. Während beim belasteten Step-Up-Wandler ein Tastverhältnis von etwa 50 % benötigt wird, liegt dieses beim unbelasteten Wandler bei etwa 5 %. Für eine stabile Regelung wären je nach Belastung unterschiedliche Regelparameter nötig, um möglichst ohne große Überschwinger die Sollspannung zu erreichen. Da sich die Belastung aber verändern kann und nicht bekannt ist, ist eine klassische Regelung schwer umsetzbar.
Die gewählte Lösung des Problems verhält sich je nach Tastverhältnis unterschiedlich. Bei niedrigen Tastverhältnissen, die im Betrieb ohne Last nötig sind und auch den Startwert nach dem Aktivieren der Regelung bilden, wird generell nur in Einerschritten nachgeregelt. Wenn der Bereich nach oben verlassen ist, setzt eine vereinfachte PI-Regelung ein. Bei Regelabweichungen über einem bestimmten Betrag wird in Viererschritten nachgeregelt, darunter in Einerschritten, solange die Regelabweichung ausreichend groß ist. Wenn die Regelabweichung relativ klein ist, wird sie aufaddiert und nur nachgeregelt, wenn diese Summe einen festgelegten Wert überschreitet. Wenn dann nachgeregelt wird, wird die Summe wieder auf 0 gesetzt.
Außerdem wird noch geprüft, ob die Ist-Spannung einen Maximalwert überschreitet, in dem Fall wird komplett neu vom kleinsten Tastverhältnis beginnend eingeregelt. Dieser Fall tritt auf, wenn die Last plötzlich entfernt wird und verhindert weitestgehend ein Einschalten des Lastwiderstands.
Für den 12 V-Step-Down-Regler wurde außerdem nur ein vereinfachter I-Regler eingesetzt, da sich keine Verbesserung durch den beim Step-Up-Wandler verwendeten vereinfachten PI-Regler ergab.

3.2.6 Kalibrierung

Die AD-Wandler des Mikrocontrollers besitzen verschiedene Ungenauigkeiten, die von Chip zu Chip unterschiedlich ausfallen, aber auf verschiedene Arten über Software herausgerechnet werden können. Desweiteren können auch durch Toleranzen der Spannungsteilerwiderstände zur Messung weitere Ungenauigkeiten auftreten.
Der einfachste Weg der Kalibrierung ist, das Ergebnis der jeweiligen AD-Wandlung für die korrekte Ausgangsspannung eines Schaltreglers fest als Soll-Wert im Programmcode zu verankern. Diese Methode ist für Einzelstücke anwendbar, da für jeden Aufbau ein eigenes Programmierfile erstellt werden muss. Es wird auch nur der Soll-Wert kalibriert, so dass die Regelung zwar den gewünschten Wert erreicht, andere Werte, die vom Sollwert abwichen, aber unter Umständen nicht korrekt umgerechnet werden können.
Eine Verbesserung dieser Methode besteht darin, die Werte nicht fest einzubinden sondern im EEPROM abzulegen, und eine Möglichkeit zu bieten, diese nachträglich zu ändern. Darüber könnte die Schaltung automatisch kalibriert werden.
Eine bessere Kalibrierung, die auch eine genauere Wiedergabe von anderen Beträgen sowie der Eingangsspannung ermöglicht, benötigt mindestens eine Bestimmung von zwei Wandlungsergebnissen zu bekannten Spannungen, aus denen dann die Zwischenwerte interpoliert werden. Die Interpolation im Programmcode des Mikrocontrollers würde allerdings viel Zeit benötigen.
Eine Variante, die mit wenig Berechnungsaufwand im Mikrocontroller trotzdem gut kalibrierte Spannungswerte auf einem PC mit Kontrollsoftware liefert, besteht darin, zwei oder mehr Wertepaare für die Kalibrierung im EEPROM zu speichern, für die Regelung aber nur die Soll-Werte zu berücksichtigen. Die anderen Werte werden von der Kontrollsoftware bei der Initialisierung ausgelesen und für die Interpolation der Spannungen auf dem PC genutzt.

3.3 Software – Erläuterungen zum Mikrocontrollercode

Der Programmcode, der letztendlich auf dem Mikrocontroller zum Einsatz kommt, besteht größtenteils aus Interrupt-Service-Routinen, die bei verschiedenen Ereignissen ausgeführt werden. Das Hauptprogramm dient fast ausschließlich der Initialisierung. Der komplette Programmcode befindet sich im Anhang C.

Definitionen und globale Variablen

Die beiden auskommentierten #define-Direktiven (`AUTOSTART_12V` bzw. `AUTOSTART_48V`) sollen ein automatisches Einregeln der jeweiligen Ausgangsspannung nach dem Einschalten aktivieren. Für die Entwicklung wurde diese Funktion nicht genutzt, da durch Fehler im Regelalgorithmus teilweise Probleme auftreten, die eine weitere Programmierung unmöglich machen, wenn die Wandler sofort selbständig anlaufen.
Die Soll- und Grenzwertdefinitionen (`#define AD_...`) geben verschiedene Zahlenwerte an, die vom AD-Wandler stammen und für die Regelung von Bedeutung sind. Diese Werte sind mikrocontrollerabhängig und müssen mit dieser Software für jeden Chip einzeln angepasst werden. Sollen mehrere Schaltungen aufgebaut und eingestellt werden, sollten diese Werte im EEPROM abgelegt und nach der Programmierung kalibriert werden.
Das Flag-Register enthält verschiedene Statusbits (Flags), die global verwendet werden. Die Verwendung des Registers `GPIOR0` hat gegenüber einer Variablen den Vorteil, dass Zugriffe auf einzelne Bits in einem Taktzyklus möglich sind.
Auf die mit `volatile register` definierten Variablen, wird in der AD-Wandler-ISR zugegriffen. Sie werden in festen Registern statt im RAM gehalten, um die Zugriffe zu beschleunigen.

Hauptprogramm

Im Hauptprogramm werden zunächst die Ein- und Ausgänge konfiguriert. Anschließend erzeugt das Programm eine Startverzögerung, um ein Neuprogrammieren zu ermöglichen, selbst wenn ein Fehler in der Regelung das Programmieren unmöglich macht. Hierzu wird über eine Timer-ISR (Timer 0 Overflow) ein Zähler vom Startwert 7 dekrementiert, bis ein Überlauf von 0 nach 255 auftritt. Danach läuft das Programm weiter.
Es folgen weitere Konfigurationen für die verschiedenen Komponenten des Mikrocontrollers. Bei der Konfiguration für den AD-Wandler wird zur Compilezeit unterschieden, ob der Mikrocontroller mit 8 MHz (ATtiny861 und ATtiny861V) oder 16 MHz (nur beim ATtiny861 möglich) betrieben wird.

Nach der Initialisierung der Komponenten werden global die Interrupts eingeschaltet und die erste AD-Wandlung gestartet. Danach läuft das Hauptprogramm in eine Endlosschleife, in der nur geprüft wird, ob der Überspannungsschutz mit dem Lastwiderstand aktiviert wurde und falls die Spannung wieder im zulässigen Bereich liegt, den Lastwiderstand abschaltet und die PWM reaktiviert. Eine nicht getestete mögliche Verbesserung besteht darin, die PWM etwas verzögert oder erst bei einer weit niedrigeren Spannung wieder zu aktivieren (also eine künstliche Hysterese einzubauen), da es im Grenzbereich oft zu einem schnellen Aus- und Einschalten und einem deutlichen Pfeifen kommt. Durch die integrierte Hysterese des Analog-Komparators kann dieser Effekt nicht kompensiert werden.

Analog-Komparator ISR (Überspannungsschutz)

Die ISR für den Interrupt des Analog-Komparators hat die Aufgabe, bei einer Überspannung die PWM abzuschalten und den Lastwiderstand einzuschalten. Die Reaktion und die Rückkehr zum Hauptprogramm erfolgen sehr schnell, da nur zwei Read-Modify-Write-Operationen in der ISR durchgeführt werden.

AD-Wandler ISR (Regelung)

In der ISR des AD-Wandlers wird die komplette Regelung der Ausgangsspannungen vorgenommen. Wie im Abschnitt 3.2.2 erläutert, wird zunächst die folgende Wandlung gestartet. Danach erfolgt die Auswahl, von welchem Kanal die aktuelle Wandlung stammt. Abhängig davon werden für die Versorgungsspannung der Wert global gespeichert und bei Unter- bzw.. Überschreitung der Grenzwerte die beiden PMW-Ausgänge abgeschaltet. Für die Ausgangsspannungen werden nach dem Abspeichern des Wertes die Regelungsalgorithmen (siehe Abschnitt 3.2.5) durchlaufen, sofern die Regelung für den Kanal aktiviert ist. Wenn bei einem Fehler ein ungültiger Zustand angenommen wurde, wird die Wandlung mit dem Kanal der Versorgungsspannung fortgesetzt. Außerdem wird in dieser ISR noch die Software-Erweiterung der PWM auf 8 Bit (siehe Abschnitt 3.2.1) für beide Ausgangskanäle realisiert.

Enterner Interrupt 0 ISR (SPI-Slave-Select)

Diese ISR wird für die SPI-Kommunikation genutzt. Die Slave-Select-Leitung, die an den Pin für den externen Interrupt 0 angeschlossen ist, hat im Ruhezustand einen HIGH-Pegel (5 V). Die ISR wird sowohl bei einer fallenden Flanke, dem Slave-Select, als auch bei einer

steigenden Flanke, also nach dem Ende der Kommunikation („Slave Deselect“) aufgerufen.

USI Overflow ISR (SPI-Kommunikation)

Dieser Interrupt wird bei einem Überlauf des 4 Bit-USI-Counters aktiviert. Der Zähler wird mit jeder Flanke des SPI-Taktsignals inkrementiert, läuft also nach 8 Bytes über. Es sind jeweils 4 Byte zu einem Datagramm zusammengefasst (siehe Abschnitt 3.2.4), das aus einem Adress-/Statusbyte und drei Datenbytes besteht. Alle Ist-Spannungen (bzw. deren AD-Wandlungsergebnisse) sowie die eingestellten Werte der Regelung können ausgelesen werden. Bei den Ausgangsspannungen der Schaltregler besteht das 3-Byte-Datenwort aus dem Ergebnis der jeweiligen AD-Wandlung in Byte 1, dem aktuellen PWM-Wert in Byte 2 und einem Byte, dessen niedrigstes Bit anzeigt, ob die jeweilige PWM aktiviert ist. Bei der Eingangsspannung entfallen der PWM-Wert und das Statusbit, die zugehörigen Bytes haben den Wert 0. Der gemessene Ist-Wert muss bei Bedarf extern in einen Spannungswert umgerechnet werden.
Einstellbar ist der PWM-Wert des jeweiligen Schaltreglers im ersten Datenbyte. Es kann die PWM für jeden Schaltregler ein- bzw. ausgeschaltet werden (LSB des zweiten Datenbytes), sowie gesteuert werden, ob die automatische Regelung aktiviert werden soll (LSB des dritten Datenbytes).
Die Zuordnung der Adressen ist in Tabelle 3.4 ersichtlich, Beispiele fur die Kommunikation zeigt Tabelle 3.5 (SB steht dabei für das Statusbyte).

Adresse	Byte 0	Zuordnung
1	2, 3	12 V Step-Down-Wandler
2	4, 5	48 V Step-Up-Wandler
3	7	Versorgungsspannung

Tabelle 3.4: Adresszuordnung

Gesendet	Empfangen	Erklärung
07 00 00 00	SB 6F 00 00	Eingangsspannung: $\frac{111 \cdot 5\,\mathrm{V} \cdot 5170\,\Omega}{256 \cdot 470\,\Omega} = 23{,}85\,\mathrm{V}$
05 00 00 00	SB CD 80 01	Step-Up-Spannung: $\frac{205 \cdot 5\,\mathrm{V} \cdot 7420\,\Omega}{256 \cdot 620\,\Omega} = 47{,}92\,\mathrm{V}$ PWM-Wert: 128, PWM aktiv
02 00 01 01	SB 00 00 00	Step-Down-Regler PWM und automatische Regelung einschalten
02 80 01 00	SB 00 00 00	Step-Down-Regler PWM auf Wert 128 setzen und einschalten

Tabelle 3.5: Kommunikationsbeispiele

Timer 0 Overflow ISR (Startverzögerung)

Hier wird die Startverzögerung abgearbeitet. Da ein Überlauf des Timers zu schnell abläuft wurde eine Variable als `static` deklariert, die von 7 abwärts zählt. Gleichzeitig wird dieser Zählerstand über die LEDs angezeigt. Wenn der Zähler von 0 weiter abwärts zählt, wird durch den Überlauf 255 erreicht. In dem Fall löscht die ISR das `START_WAIT`-Flag und signalisiert damit der Busy-Waiting-Schleife im Hauptprogramm, dass die Wartezeit abgelaufen ist.

4 Messungen

4.1 Verhalten des Step-Up-Wandlers

Betriebsverhalten

Die erste Messung (Abb. 4.1) erfolgt ohne angeschlossene Last am Step-Up-Wandler. Dadurch wird nur ein sehr niedriges Tastverhältnis benötigt, um die gewünschte Ausgangsspannung von 48 V zu erreichen. Die resultierende Ausgangsspannung ist stabil, liegt aber etwas unter der Sollspannung. Da die Spannung aber ohnehin nur an den Spannungsteilern für den AD-Wandler und den Analog-Komparator abfällt und nicht genutzt wird, ist das kein Problem. Dass die Spannung etwas zu niedrig ist liegt daran, dass der eingestellte PWM-Wert dem eigentlich optimalen Wert am nächsten kommt. Der nächsthöhere Wert würde eine deutlich zu hohe Ausgangsspannung verursachen.

Wird nun ein Lastwiderstand von $1\,\mathrm{k}\Omega$ angeschlossen, fließt ein Strom von 48 mA. Dieser Strom liegt noch unter dem berechneten Mindeststrom $I_{a\,min}$ = 90,9 mA, der Regelung gelingt es aber, die Ausgangsspannung mit einem reduzierten Tastverhältnis nahezu konstant zu halten (Abb. 4.2). Das Springen zwischen zwei Spannungen mit einer Frequenz von etwa 125 Hz stammt von der Regelung, die die Abweichung aufsummiert und dann den eingestellten PWM-Wert um eins erhöht bzw. verringert.

Bei einem kleineren Widerstand von $100\,\Omega$ ist der Strom mit 480 mA ausreichend groß, so dass nahezu das optimale Tastverhältnis von 50 % zur Spannungsverdoppelung nötig ist. Auch hier (Abb. 4.3) wird zwischen zwei PWM-Werten gewechselt, so dass die Ausgangsspannung zwischen zwei Werten springt. Das Wechseln wird hier aber von der Softwareerweiterung der PWM auf 8 Bit hervorgerufen. Das Signal ist etwa 3 ms auf einem höheren Pegel und etwa 1 ms auf einem niedrigeren Pegel, woraus man schließen kann, dass der von der Regelung eingestellten PWM-Wert gilt: $PWM \equiv 3 \mod 4$

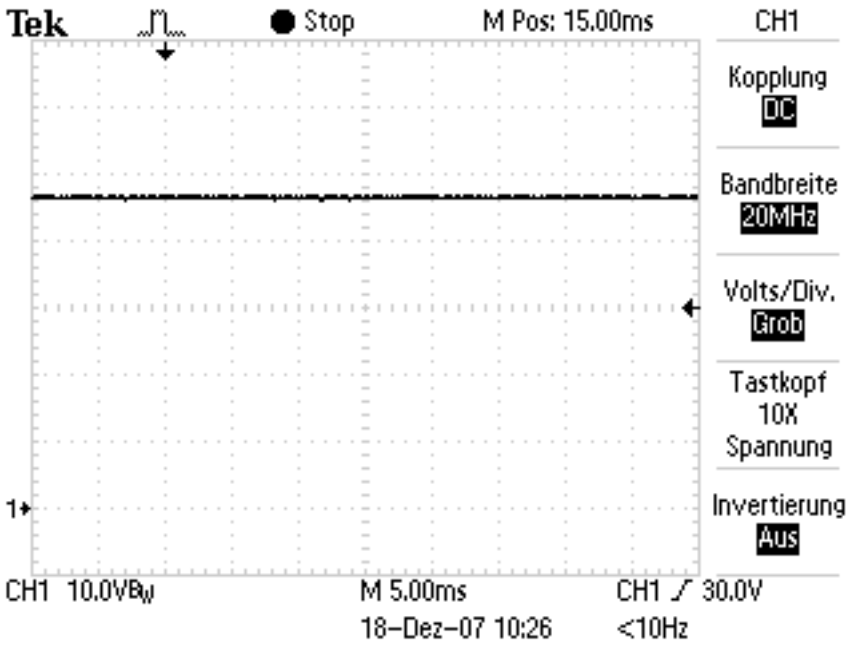


Abbildung 4.1: Step-Up-Wandler ohne Last

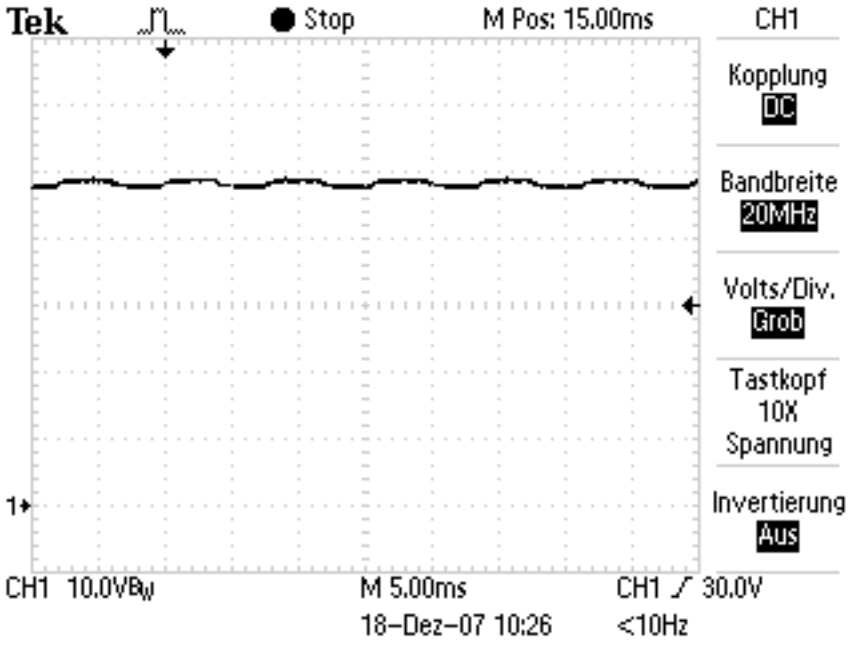


Abbildung 4.2: Step-Up-Wandler mit kleiner Last

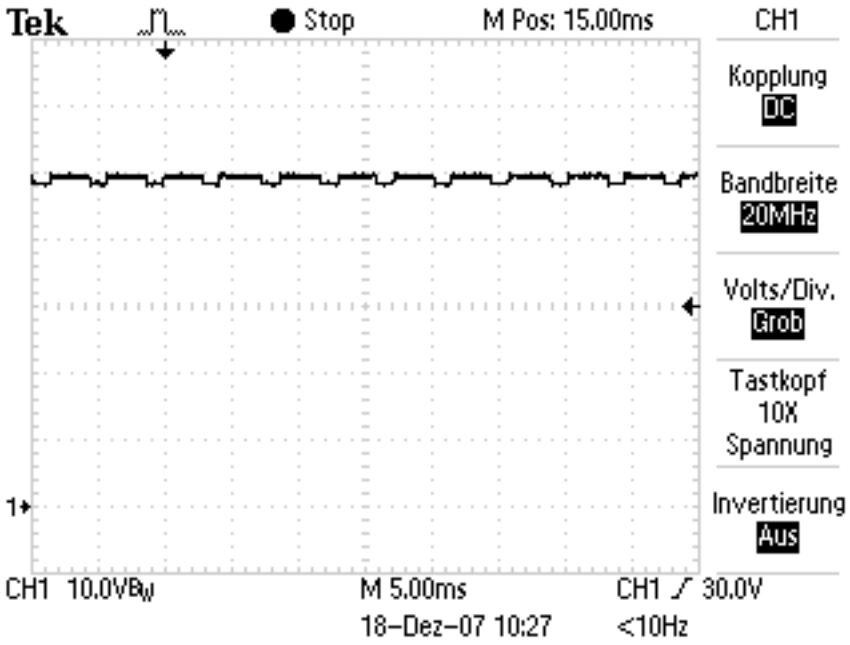


Abbildung 4.3: Step-Up-Wandler mit großer Last

Einschaltverhalten

Die Einschaltverhalten mit einem Lastwiderstand von 100 Ω sowie ohne Last zeigen nur leichte Unterschiede. Ohne Last schwingt die Ausgangsspannung auf 53,2 V über, um dann auf die etwas zu niedrige Ausgangsspannung abzusinken. Wenn der Lastwiderstand angeschlossen ist, nähert sich die Ausgangsspannung den gewünschten 48 V ohne überzuschwingen an. (Abb. 4.4 und 4.5)

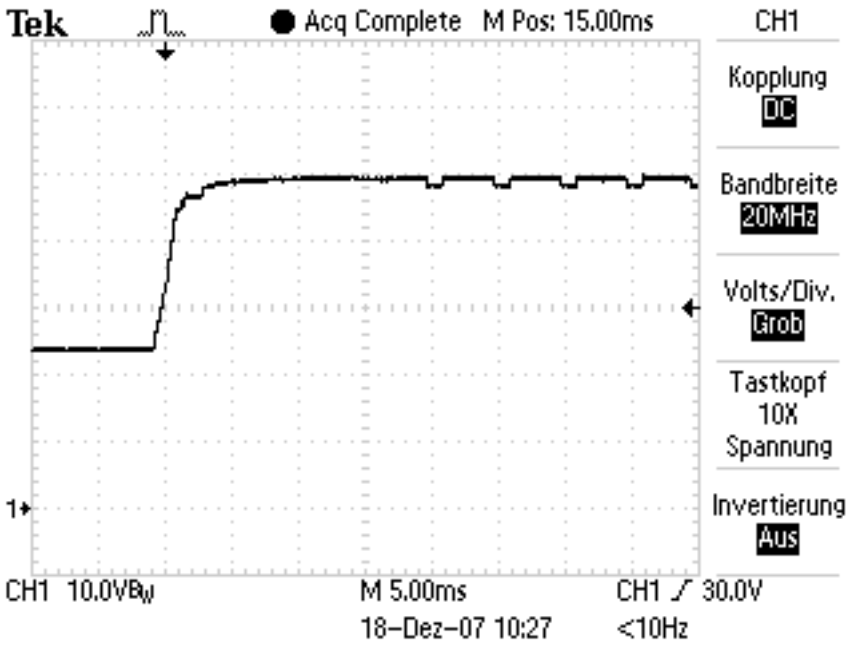


Abbildung 4.4: Einschaltverhalten des Step-Up-Wandlers mit Last

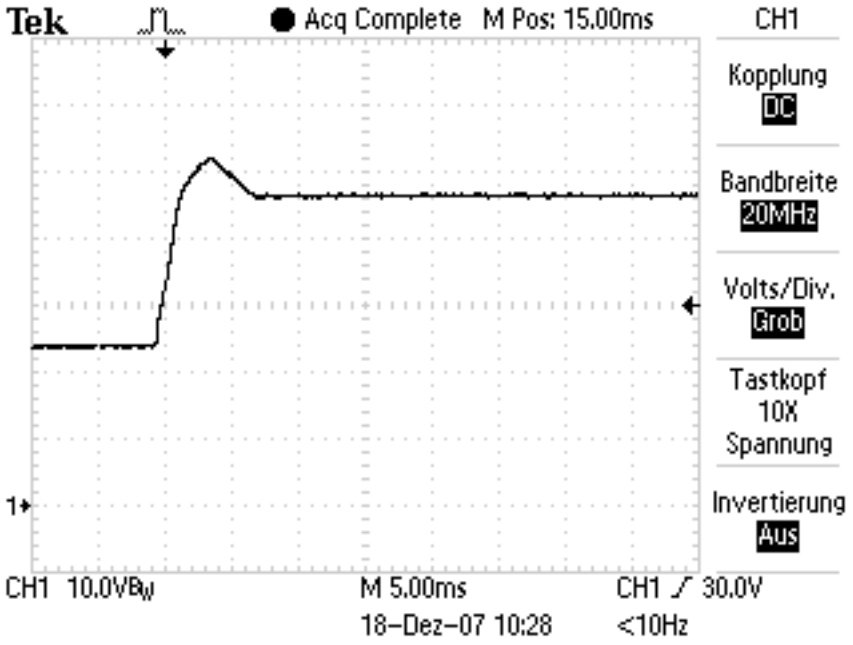


Abbildung 4.5: Einschaltverhalten des Step-Up-Wandlers ohne Last

Ausschaltverhalten

Beim Ausschalten mit und ohne Last zeigen sich deutlichere Unterschiede als beim Einschalten. An einem belasteten Ausgang fällt die Spannung innerhalb einer Millisekunde von 48 V auf 24 V ab (Abb. 4.6).

An einem unbelasteten Ausgang dauert die gleiche Spannungsänderung 18 ms (Abb. 4.7). Diese unterschiedlichen Zeiten entstehen vermutlich durch den Entladevorgang der Kondensatoren. Am unbelasteten Ausgang werden die Kondensatoren nur über die Spannungsteilerwiderstände mit einem verhältnismäßig geringen Strom entladen. Ist der Lastwiderstand von 100 Ω angeschlossen, fließt ein deutlich höherer Strom und entlädt den Kondensator schneller.
Über die Formel für den Entladevorgang eines Kondensators sollte sich anhand des Signalverlaufs beim Abschalten des unbelasteten Wandlers der Widerstand der Spannungsteiler errechnen lassen, um die Richtigkeit der Annahme zu zeigen. Es wird dennoch eine Abweichung gegenüber dem errechneten Wert der eingebauten Widerstände geben, da die Bauteile gewissen Fertigungstoleranzen unterliegen.

$$
\begin{aligned}
U(t) &= U_0 \cdot e^{-\frac{t}{R \cdot C}} \\
\ln U(t) &= \ln U_0 - \frac{t}{R \cdot C} \\
\ln U(t) - \ln U_0 &= -\frac{t}{R \cdot C} \\
R &= -\frac{t}{(\ln U(t) - \ln U_0) \cdot C} \\
&= -\frac{18\,\text{ms}}{(\ln 24\,\text{V} - \ln 48\,\text{V}) \cdot 8{,}9\,\mu\text{F}} \\
&= 2917{,}81\,\Omega
\end{aligned}
$$

Mit den verwendeten Widerständen sollte der Wert bei

$$
\frac{(6{,}8\,\text{k}\Omega + 620\,\Omega) * (200\,\Omega + 4{,}7\,\text{k}\Omega + 100\,\Omega)}{(6{,}8\,\text{k}\Omega + 620\,\Omega) + (200\,\Omega + 4{,}7\,\text{k}\Omega + 100\,\Omega)} = 2987{,}12\,\Omega
$$

liegen, also etwa 70 Ω über dem anhand der Entladekurve berechneten Wert. Das zeigt, dass die Annahme über die Ursache der unterschiedlichen Abschaltdauern richtig ist.

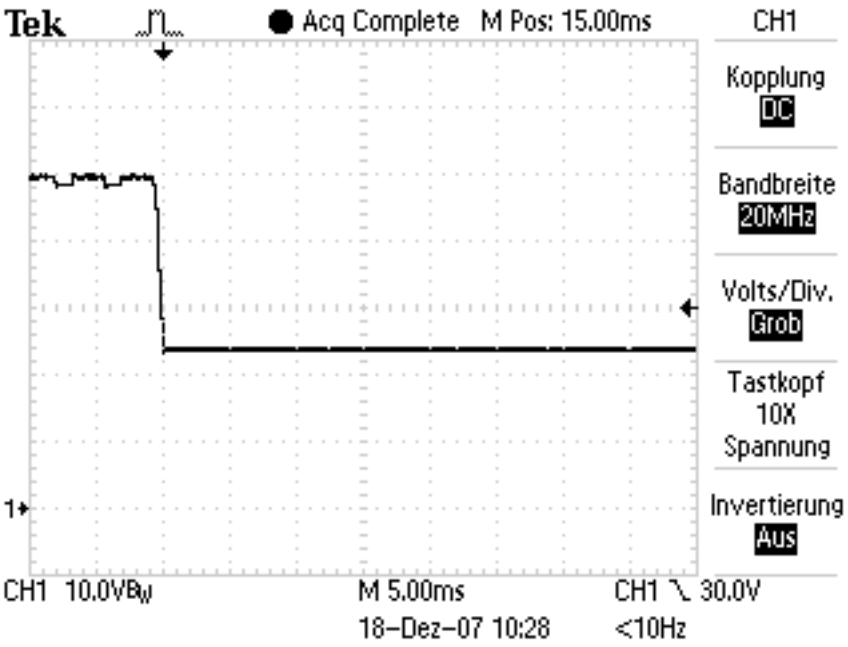


Abbildung 4.6: Ausschaltverhalten mit Last

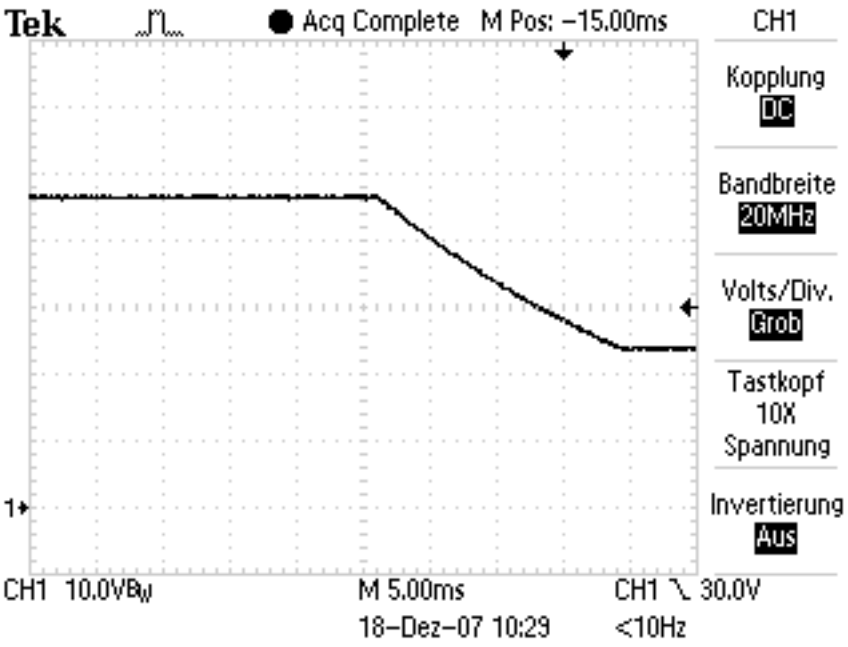


Abbildung 4.7: Ausschaltverhalten ohne Last

Verhalten bei Lastwechseln

Das Regelverhalten bei Lastwechseln ist ebenfalls wichtig. Hierfür wurde der Schaltregler zunächst ohne Last (von den zwei Spannungsteilern abgesehen) eingeschaltet und dann der Lastwiderstand im Betrieb angeschlossen (Abb. 4.8). Dabei bricht die Spannung zuerst deutlich auf ca. 28 V ein, erreicht nach 2 ms wieder 45 V und wird dann mit einem Überschwingen wieder auf ihren Sollwert eingeregelt. Im Anschluss wurde der Lastwiderstand wieder entfernt (Abb. 4.9). Dadurch steigt die Spannung kurzzeitig auf 53 V an, sinkt dann aber in 4 ms auf die Leerlaufspannung ab.

Fazit zum Step-Up-Wandler

Das Verhalten in den verschiedenen gemessenen Situationen ist im Rahmen der Regelung mit dem ATtiny861 gut. Kleinere Unzulänglichkeiten – wie die Dauer des Einregelns auf die Sollspannung sowie das leichte Schwanken um einen Ausgangsspannungswert – ließen sich mit einer höheren AD-Wandlungsrate und höherer Rechenleistung sowie einer feineren PWM-Auflösung (die tatsächliche Auflösung ohne Software-Erweiterung) beheben.
Zum Abschluss wurde wieder der LM5112 als MOSFET-Treiber eingesetzt, um einige Vergleiche anzustellen. Dabei zeigten sich keine großen Unterschiede bei der Ausgangsspannung. Durch die schnelleren Schaltzeiten erwärmte sich der MOSFET aufgrund des geringeren Leistungsabfalls aber weniger stark.

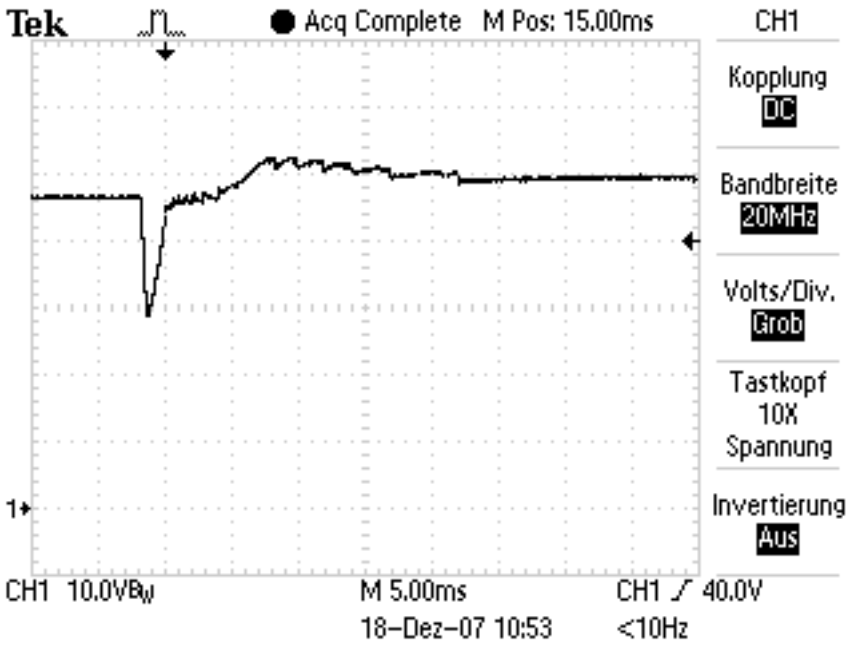


Abbildung 4.8: Verhalten des Step-Up-Wandlers beim Anschließen einer Last

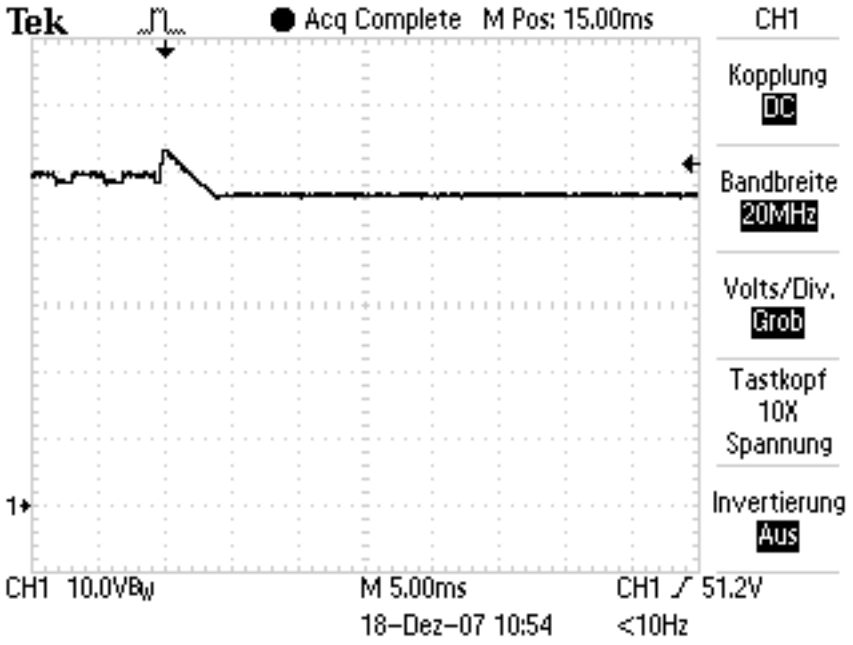


Abbildung 4.9: Verhalten des Step-Up-Wandlers beim Entfernen einer Last

4.2 Verhalten des Step-Down-Wandlers

Betriebsverhalten

Der unbelastete Step-Down-Wandler zeigt ein Schwingen zwischen 11,4 V und 12,6 V. Der Ursprung ist hier die Regelung, die den PWM-Wert stärker verändert als es notwendig wäre Abb. 4.10). Mit einer manuellen Vorgabe des PWM-Wertes ohne Regelung lässt sich eine konstante Ausgangsspannung von 48 V ohne Probleme erreichen. Da die Ausgangsspannung aber wiederum nur am Spannungsteiler abfällt ist auch dieses verhältnismäßig starke Schwingen unproblematisch.

Schon bei einer kleinen Last von 12 mA, das sind weniger als 10 % des berechneten Minimalstroms, werden die Spannungsschwankungen reduziert. Die Ausgangsspannung schwankt nur noch im Bereich von etwa 11,7 V bis 12,6 V. (Abb. 4.11)

Bei einer Last von 120 mA, einem Strom, der immer noch etwas geringer als der Minimalstrom ist, reduziert sich der Bereich in dem die Spannung schwankt auf etwa 0,3 V. Mit einer Last von 200 mA, die demnach über dem Minimalstrom liegt, ergibt sich hier auch keine weitere Verbesserung. (Abb. 4.12)

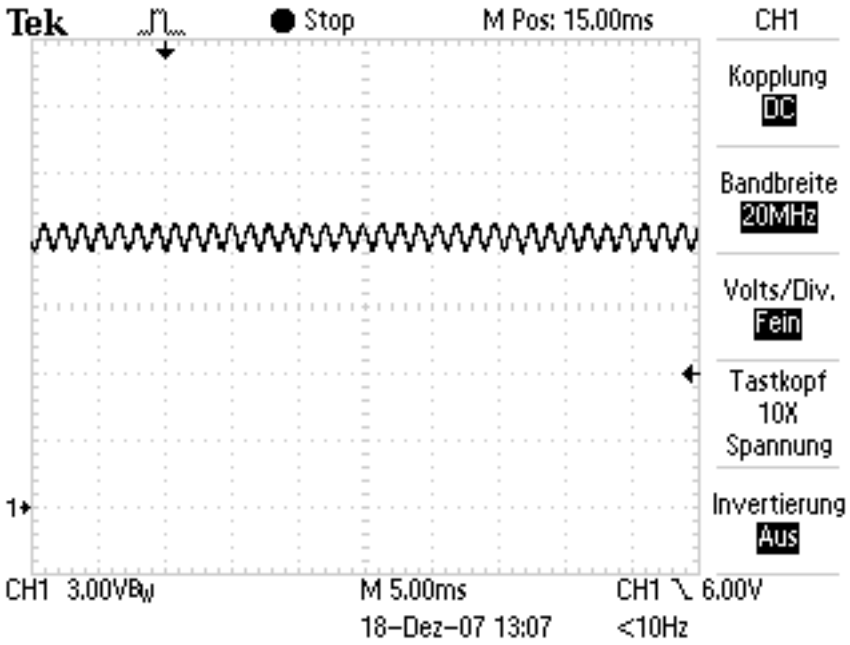


Abbildung 4.10: Step-Down-Wandler ohne Last

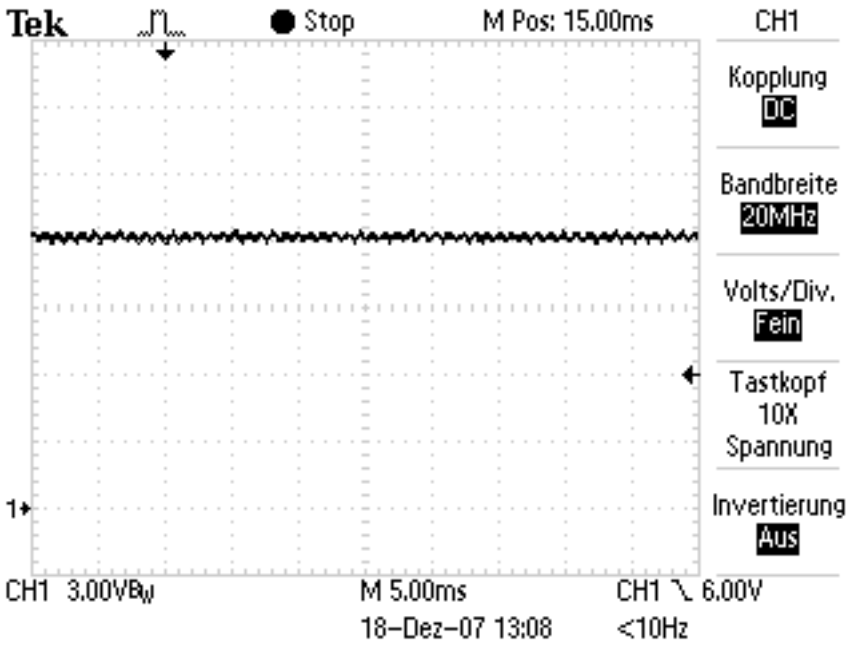


Abbildung 4.11: Step-Down-Wandler mit kleiner Last

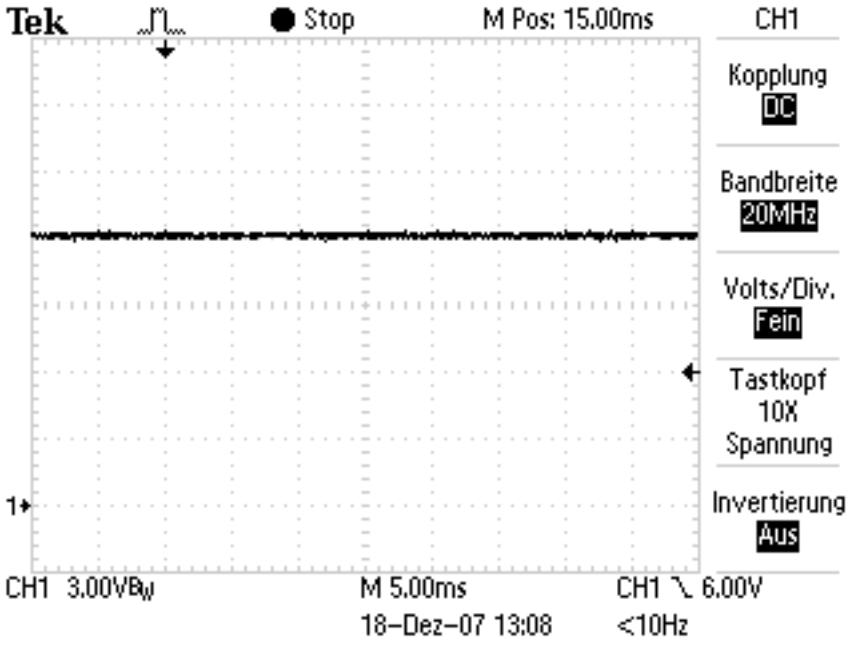


Abbildung 4.12: Step-Down-Wandler mit größerer Last

Einschaltverhalten

Beim Einschalten des belasteten Step-Down-Wandlers dauert das Einregeln auf 12 V ca. 2 ms. Die Sollspannung wird dabei ohne Überschwingen erreicht. Das ist insofern wichtig, da angeschlossene Geräte durch die Spannungsspitze beim Überschwingen eventuell beschädigt werden könnten. (Abb. 4.13)

Ohne Last wird die Sollspannung nach dem Einschalten zwar schon nach etwa 1 ms erreicht, dafür schwingt die Ausgangsspannung zunächst auf 15 V und danach dauerhaft zwischen 11,4 V und 12,9 V. Da ohne eine Last auch keine angeschlossene Elektronik beschädigt werden kann und da die maximale Spannung innerhalb des zulässigen Bereiches der im Wandler verwendeten Bauteile liegt, ist das Überschwingen hier unkritisch. (Abb. 4.14)

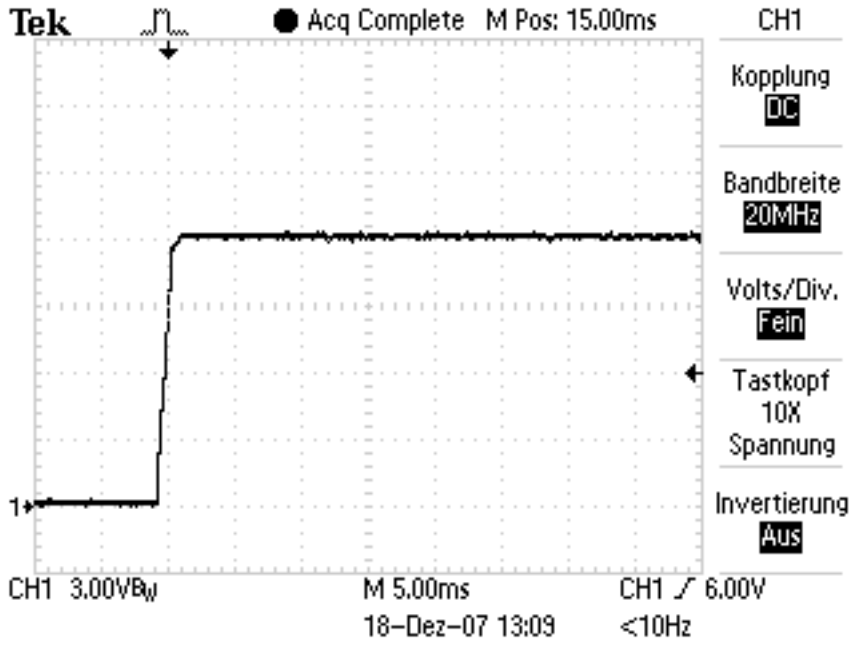


Abbildung 4.13: Einschaltverhalten des Step-Down-Wandlers mit Last

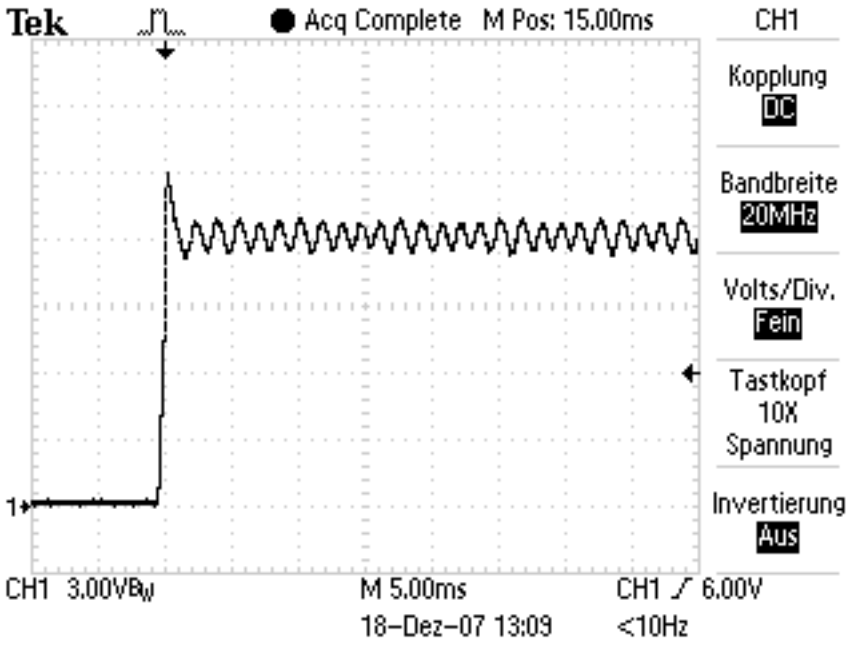


Abbildung 4.14: Einschaltverhalten des Step-Down-Wandlers ohne Last

Ausschaltverhalten

Wird der belastete Wandler abgeschaltet, sinkt die Spannung innerhalb weniger ms auf 0 V ab (Abb. 4.15), da die Ladung der Kondensatoren am Ausgang über die Last schnell abgebaut wird. Ist der Wandler unbelastet, sieht man deutlich eine typische Entladekurve eines Kondensators (Abb. 4.16). Das Verhalten ist dem Abschalten des Step-Up-Wandlers ähnlich, nur dass der Kondensator hier komplett entladen wird. Aus der Entladekurve lässt sich wiederum der Widerstand des Spannungsteilers herleiten, über den der Kondensator (bzw. die Kondensatoren mit einer Gesamtkapazität von 4.5 µF) entladen werden.

$$R = -\frac{30\,\text{ms}}{(\ln 1{,}2\,\text{V} - \ln 12{,}6\,\text{V}) \cdot 4{,}5\,\mu\text{F}}$$

$$= 2835{,}22\,\Omega$$

Zum Spannungsteiler mit 3,2 kΩ besteht eine Abweichung von 11,4 %. Diese wird von Bauteiltoleranzen und Abeleseungenauigkeiten am Oszilloskop hervorgerufen.

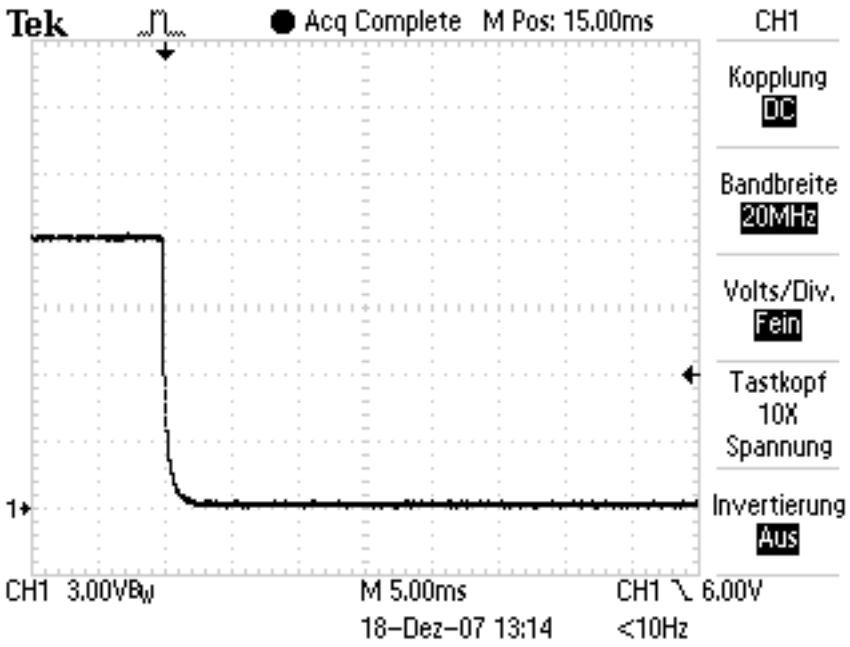


Abbildung 4.15: Ausschaltverhalten mit Last

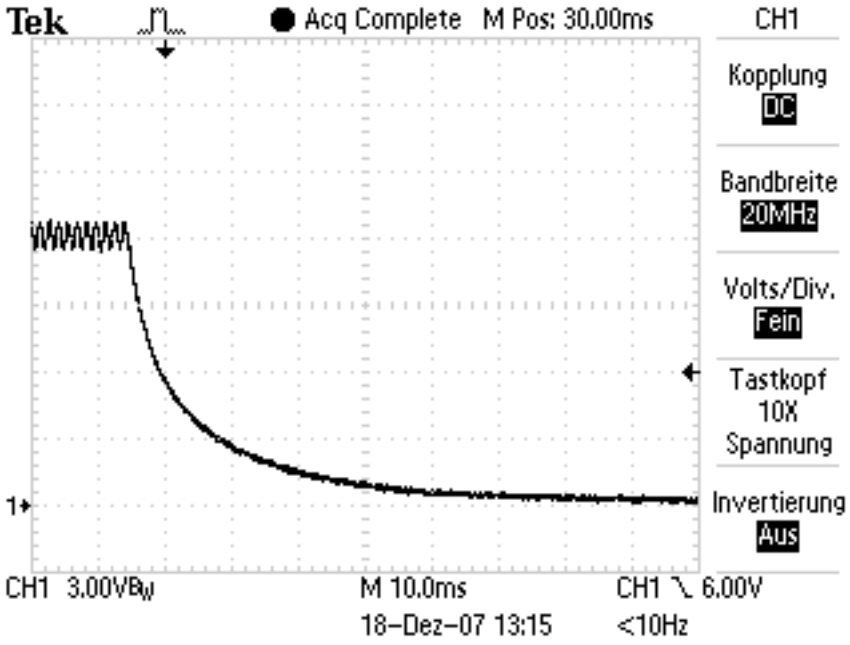


Abbildung 4.16: Ausschaltverhalten ohne Last

Verhalten bei Lastwechseln

Der Lastwechsel von etwa 0 W – durch den Spannungsteiler fließt noch ein Strom von 3,75 mA – auf 1,44 W (bei Berücksichtigung des Spannungsteilers 1,485 W) bricht die Spannung für 1,5 ms bis auf 6 V ein, nachdem sie wieder eingeregelt wurde, bleibt sie aber stabil auf 12 V (Abb. 4.17. Wird die Last wieder entfernt, entsteht eine Spanungsspitze, die für 1 ms zwischen 12 V und 16 V liegt (Abb. 4.18).

Fazit zum Step-Down-Wandler

Sofern eine Last an den Wandler angeschlossen ist, zeigt sich ebenfalls eine gute Regelung. Eine besser angepasste Regelung, die unter Umständen mehr Rechenleistung benötigt, könnte auch das starke Schwingen im unbelasteten Zustand reduzieren. Außerdem könnte dadurch das Überschwingen beim Einschalten ohne Last und beim Entfernen einer Last weiter reduziert werden.

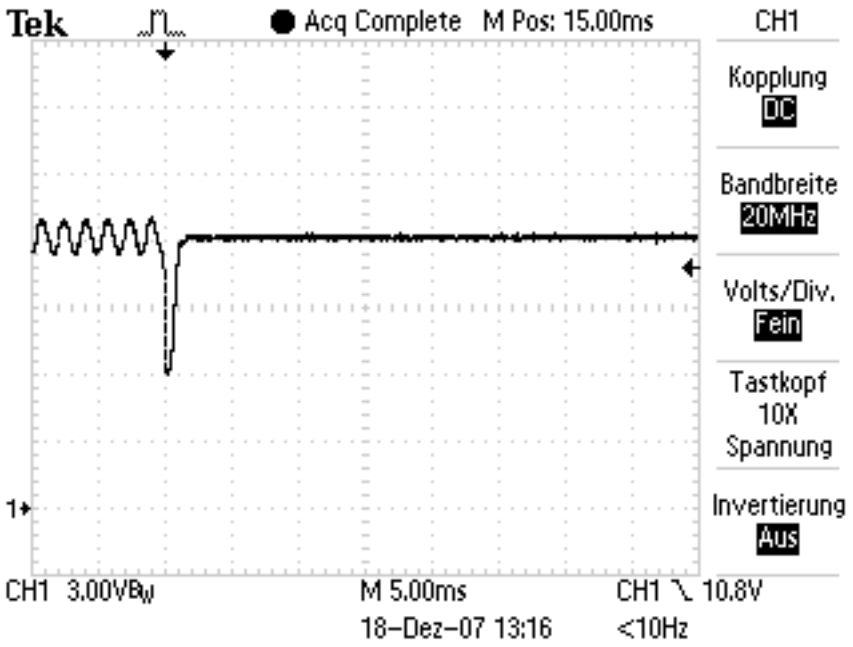


Abbildung 4.17: Verhalten des Step-Down-Wandlers beim Anschließen einer Last

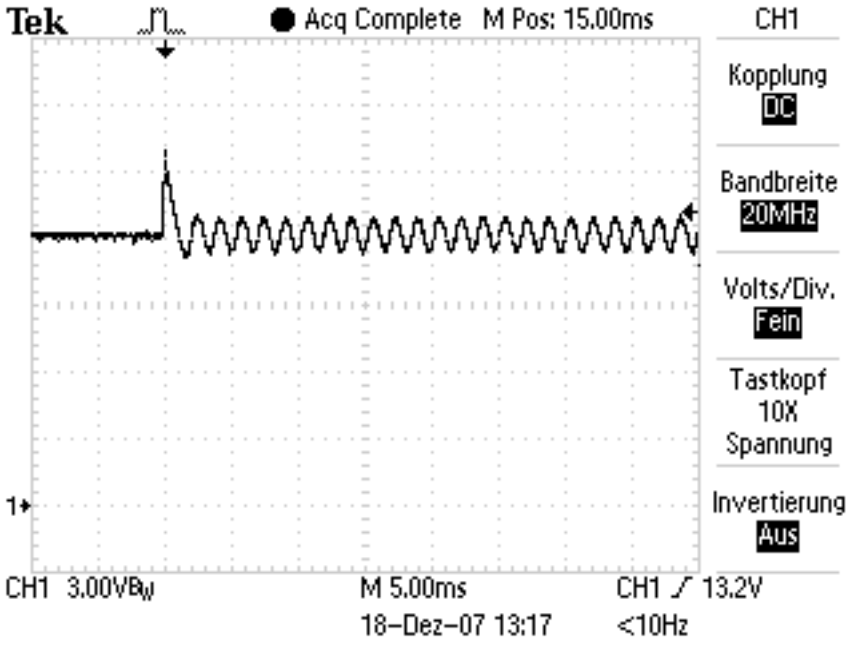


Abbildung 4.18: Verhalten des Step-Down-Wandlers beim Entfernen einer Last

4.3 Erzeugte Störungen

Die Probleme mit den eingestrahlten Störungen auf der Leiterbahn zum Komparator machen deutlich, dass bei Verwendung hoher Schaltfrequenzen, insbesondere in Verbindung mit Induktivitäten, elektromagnetische Effekte auftreten, die nicht nur die Schaltung selbst sondern auch andere Schaltungen in der Umgebung stören können. Ein ideales Rechtecksignal mit 50 % Tastverhältnis und unendlich steilen Flanken besteht aus verschiedenen überlagerten Sinusschwingungen, mit der Basisfrequenz und deren ungeradzahligen Vielfachen mit abnehmender Amplitude. Somit sind bei 1 MHz Schaltfrequenz neben einer 1 MHz Sinusschwingung auch ein 3 MHz Anteil, ein 5 MHz Anteil etc. vorhanden.
Die Störungen, die von der Induktivität des Step-Up-Wandles abgestrahlt werden, können schon mit der Tastkopfspitze eines Oszilloskops aufgenommen werden. Noch besser lassen sich die Störungen darstellen, indem die Masseklemme des Tastkopfes an dessen Spitze angeschlossen wird. Die dadurch entstandene Leiterschleife bildet eine gute Empfangsantenne, wie die Abbildung 4.19 zeigt. Für diese Messung wurde der Step-Up-Wandler mit 1 A belastet und die Schleife am Tastkopf etwa 2 cm über die Platine gehalten, wobei das Zentrum der Schleife über der Induktivität lag. Abbildung 4.20 zeigt eine Ausschnittsvergrößerung der Zeitachse, in der Schwingungen mit deutlich mehr als 1 MHz erkennbar sind. Die maximale Amplitude der gemessenen Schwingungen liegt bei etwa 1 V.
Das Verhalten der Schaltung bei externen Störungen wurde nicht untersucht.

Andere Messmethoden sowie Ursachen und Möglichkeiten der Reduzierung von Störungen werden in „EMV-gerechtes Gerätedesign“ (Durcansky, 1999) aufgezeigt. Zusätzliche Informationen, speziell zur Funkentstörung von Schaltnetzteilen, sind in „Professionelle Schaltungstechnik, Band 3“ (Klaschke, 1996, S. 639 ff.) zu finden.

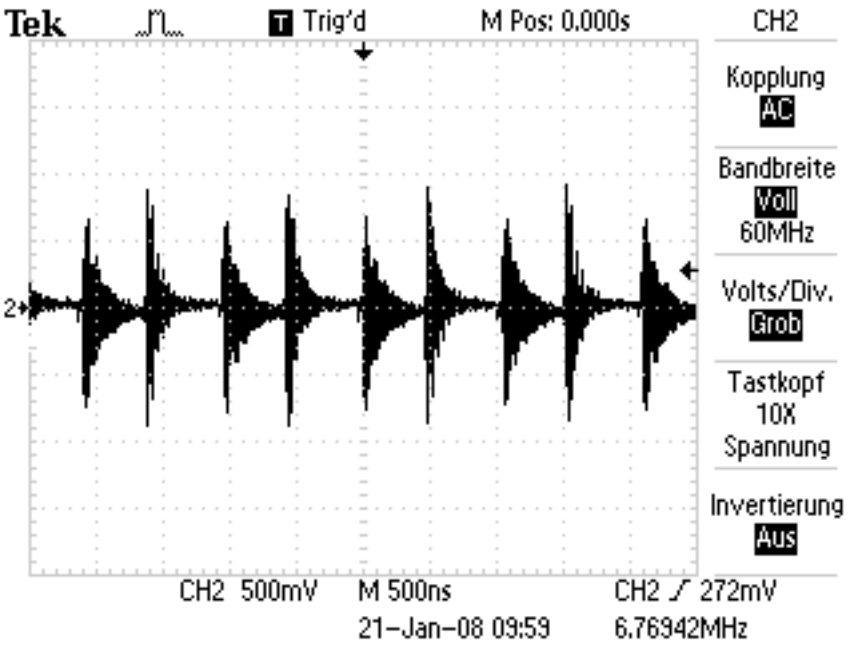


Abbildung 4.19: Abgestrahlte Störungen des Step-Up-Wandlers

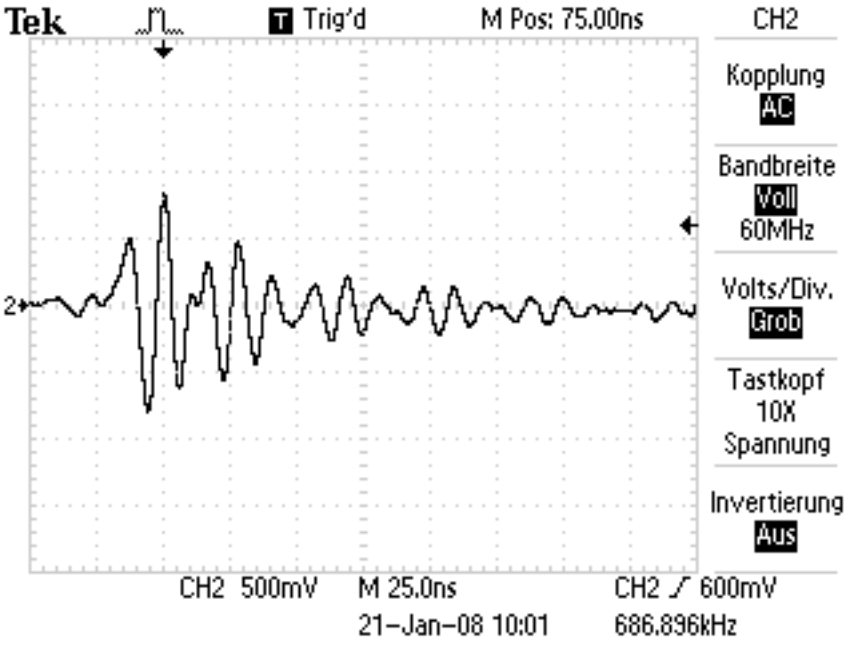


Abbildung 4.20: Ausschnittsvergrößerung der Zeitachse

4.4 Elektrische Daten

Der Step-Up-Wandler hat bei den Versuchen mit einer Eingangsspannung von 8 V noch eine Ausgangsspannung von 48 V bei ca. 500 mA geliefert. Vermutlich wäre auch ein höherer Strom möglich gewesen, dieser Versuch war aber mit dem verwendeten Labornetzteil nicht möglich, da es maximal etwa 3 A liefern kann. Allerdings hat der Step-Down-Wandler bei dieser Eingangsspannung keine Möglichkeit, eine Ausgangsspannung von 12 V zu erzeugen.
Die minimale Betriebsspannung wird also durch den Step-Down-Wandler festgelegt und liegt bei etwa 12,9 V. Eine Spannung über 48 V kann vom Step-Up-Wandler nicht auf 48 V reduziert werden, so dass dieser Wert auch die obere Grenze darstellt. Da in der Schaltung während der Messungen ein Spannungsregler vom Typ 78L05 zum Einsatz kam, wird die maximale Betriebsspannung durch diesen auf 30 V begrenzt.
Dieser Spannungsbereich wird vom Mikrocontroller überprüft und für den Fall, dass er unter- oder überschritten wird, schaltet der Mikrocontroller die PWM-Signale ab, so dass am Ausgang des Step-Up-Wandlers die Eingangsspannung und am Ausgang des Step-Down-Wandlers 0 V anliegen.
Mit dem verwendeten Treiber aus 5 Invertern des 74ACT04 konnte der Step-Up-Wandler auf längere Zeit mit maximal 1 A belastet werden, kurzzeitig waren auch ca. 1,3 A möglich. Darüber war aufgrund der langen Schaltvorgänge die Verlustleistung am MOSFET zu groß, so dass dieser zerstört wurde.
Nachdem die Entwicklung abgeschlossen war, wurde wieder ein LM5112 als Treiber eingesetzt, der allerdings mit einem externen Netzteil anstelle des 12 V-Step-Down-Wandlers versorgt wurde. Mit diesem Treiber und einem 24 V/5 A Netzteil konnte die geplante Ausgangsleistung von 96 W (48 V bei 2 A) erreicht werden.
Der Wirkungsgrad wurde aus Eingangsstrom und -spannung, sowie Ausgangsstrom und -spannung berechnet und lag ab etwa 300 mA Ausgangsstrom schon über 75 % und ab etwa 36 W Ausgangsleistung (also 750 mA Ausgangsstrom) über 90 %.

5 Fazit

Im Rückblick ist die Umsetzung sowohl auf Hardwareseite als auch auf Softwareseite ohne allzugroße Probleme abgelaufen. Die Software konnte schon vor Fertigstellung der Hardware vorbereitet werden. Nachdem die Hardware aufgebaut war, konnte dann mit kleinen Anpassungen am vorab geschriebenen Code direkt zur Entwicklung des passenden Regelungsalgorithmus übergegangen werden. An einigen Stellen musste aber die Hardware gegenüber dem ersten Entwurf noch angepasst werden.
Der Step-Up-Wandler konnte die geplante Ausgangsleistung von ca. 96 W (48 V bei 2 A) liefern. Der 12 V Step-Down-Wandler konnte die Spannung gut stabilisieren, schwingt aber bei niedrigen Belastungen etwas zu stark. Der geplante 3,3 V Step-Down-Wandler konnte nicht umgesetzt werden, das ist aber aufgrund der guten Verfügbarkeit von Standardbauteilen für diese Regelung kein großer Nachteil. Mit weitergehenden Untersuchungen können wahrscheinlich auch diese kleinen Probleme beseitigt werden und die Schaltung insgesamt noch verbessert werden.

5.1 Aufgetretene Probleme/Schwierigkeiten

Beim ATtiny861 handelt es sich um einen Mikrocontroller mit eher geringer CPU-Leistung. Deshalb musste der Regelalgorithmus sehr einfach gehalten werden, um in einem möglichst kleinen Teil der Dauer einer AD-Wandlung durchlaufen zu werden. Außerdem ist die Dauer einer AD-Wandlung mit 13 µs pro Kanal deutlich länger als die Periodendauer der PWM. Dadurch wird seltener nachgeregelt als es eigentlich möglich wäre.
Durch Fehler beim Platinenlayout traten weitere Probleme auf. Die 33 µH Induktivität des Step-Up-Wandlers strahlte durch die Schaltvorgänge deutliche Störungen auf die zu dicht daneben und teilweise darunter verlegten Leiterbahnen, unter anderem die Verbindung vom Spannungsteiler zum Analog-Komparator-Eingang des Mikrocontrollers. Dadurch traten im Normalbetrieb ab einer gewissen Ausgangsspannung Spannungsspitzen auf, die den Grenzwert überschritten, bei jedem Schalten einen Interrupt auslösten (wobei die meisten unbeachtet blieben, da das Flag noch nicht gelöscht war) und den Mikrocontroller somit ständig in die ISR zwangen. Daher musste noch ein RC-Tiefpass als Störungsfilter vor dem Eingang integriert werden. Die Nachrüstung war allerdings kein großes Problem.

Dazu musste die Leiterbahn mit einem Messer aufgetrennt werden und der Lötstopplack entfernt werden, um Kontaktflächen für den SMD-Widerstand zu bieten. Der Kondensator wurde mit Fädeldraht an den Massepin des Mikrocontrollers angeschlossen. Bei weiteren Platinendesigns sollten vor allem zwischen Induktivitäten und analogen Leiterbahnen ausreichend große Abstände und Masseflächen als Abschirmung vorgesehen werden.
Auch an anderen Stellen der Schaltung musste noch nachgebessert werden, beispielsweise wurden auf die zwei Ausgangskondensatoren des Step-Up-Wandlers zwei weitere aufgelötet um die Kapazität zu verdoppeln und damit die Welligkeit der Ausgangsspannung zu reduzieren. Im Platinenlayout wurde auch ein Glättungskondensator für die Versorgungsspannung des MOSFET-Treibers für den 12 V-Wandler in dessen unmittelbarer Nähe vergessen, der ebenfalls – fliegend verdrahtet – nachgerüstet wurde.
Durch Fehler in der Software sind in der Entwicklungsphase außerdem mehrfach die Schalttransistoren zerstört worden.
Ursprünglich war eine I^2C-Schnittstelle zur Kommunikation vorgesehen. Die Programmierung des Mikrocontrollers für diese Funktion stellte sich aber – nachdem die Platinen schon in der Fertigung waren – als sehr kompliziert heraus, da keine spezielle I^2C (bei Atmel TWI – Two Wire Interface – genannt) Einheit im Mikrocontroller vorhanden ist. Die Funktionalität muss mit dem USI abgebildet werden, wobei durch die 9-Bit-Übertragung des I^2C-Protokolls (8 Datenbits, 1 Acknowledge-Bit) doppelt so viele Aufrufe der ISR erfolgen, wie z.B. bei der implementierten SPI-Kommunikation. Abgesehen vom Programmieraufwand wäre also noch ein Teil der ohnehin knappen Rechenleistung verloren gegangen. Der augenscheinliche Nachteil des entfallenden 3,3 V-Reglers stellt sich auf den zweiten Blick sogar als Vorteil dar, da eine dritte Software-Regelung entfällt und die Abtastfrequenz (und die Regelfrequenz) der verbleibenden zwei Kanäle dadurch gesteigert wird. Der 3,3 V-Step-Down-Wandler war auch nur als optionaler Bestandteil der Schaltung vorgesehen, da an diesem Ausgang keine große Leistung benötigt wird und passende Linearregler verfügbar sind.
Für die Messungen war eine elektronische Last vorgesehen, die als Stromsenke dienen sollte und die aufgenommene Leistung über Leistungstransistoren in Wärme umsetzt. Diese wurde im Vorfeld der Bachelorarbeit entwickelt und regelt den Stromfluss ebenfalls mit einem AVR Mikrocontroller. Da diese durch ihre eigene Regelung aber auch leichte Schwankungen hervorrufen kann und in Verbindung mit den Regelungen der Wandler stärkere Schwingungen erzeugt als ein Widerstand, kam diese nur zur Bestimmung des maximalen Stroms zum Einsatz. Ansonsten wurden die Messungen mit verschiedenen Widerständen durchgeführt.

5.2 Ausblick

Angedacht ist eine weitere Umsetzung der Schaltung, in der anstelle des Mikrocontrollers ein FPGA-Baustein für die Regelung zum Einsatz kommt. Dabei sind dann allerdings noch zusätzliche externe Bauteile nötig, beispielsweise AD-Wandler um die Ausgangsspannungen zu bestimmen. Die Regelung im FPGA ist aber unter Umständen schneller als im Mikrocontroller, da die einzelnen Regelungsalgorithmen und auch die Funktionalität der anderen ISRs parallel arbeiten.

5.3 Persönliches Fazit

Diese Arbeit hat mir einige tiefere Erkenntnisse auf dem Gebiet der Elektrotechnik gebracht, mit der ich mich bisher hauptsächlich im Hobbybereich befasst habe. Auch verschiedene Tests, die ich bei der Einarbeitung in die Simulation mit Spice durchgeführt habe, konnten das Verständnis von Gleichungen verbessern, die ich bisher zwar verwendet habe, deren Grundlagen ich aber nicht weiter hinterfragt habe.
Die Fehler beim Platinenlayout, die zu den Problemen mit dem Analog-Komparator geführt haben, führten dazu, dass ich mich ein wenig mit dem vorher von mir wenig beachteten Bereich der elektromagnetischen Verträglichkeit (EMV) befasst habe. Bei weiteren zu entwickelnden Schaltungen werde ich sicher stärker darauf achten, diese von vornherein störungsarm auszulegen.
Im Bereich der Softwareentwicklung konnte ich feststellen, wie wichtig, gerade bei hardwarenaher Programmierung, ein Verständnis der zugrundeliegenden Hardwarearchitektur ist. Ich habe mir mehrfach den aus dem C-Quellcode erzeugten Assemblercode angesehen, um eventuelle Optimierungsmöglichkeiten zu finden. In den Code selber musste ich allerdings nicht eingreifen, die einzigen Optimierungen waren die Verwendung von festen Registern für globale Variablen, auf die schnell zugegriffen werden muss sowie die Verwendung eines General-Purpose-Registers als Flag-Register, in dem einzelne Bits mit einem Taktzyklus geschrieben bzw. gelesen werden können.
Die Auffrischung der LaTeX-Kenntnisse zum Schreiben dieser Arbeit hat zwar etwas Zeit in Anspruch genommen, ermöglichte aber ein stressfreies Arbeiten und wird auch über diese Arbeit hinaus nützlich sein.

Literaturverzeichnis

[Agrawal 2001] AGRAWAL, Jai P.: *Power electronic systems : theory and design*. Prentice-Hall, 2001. – ISBN 0-13-442880-3

[Atmel 2004] ATMEL: *AVR035: Efficient C Coding for AVR (Application Note)*. Rev. 1497D - 01/04. http://atmel.com/dyn/resources/prod_documents/doc1497.pdf: , 2004

[Atmel 2006] ATMEL: *ATtiny261/461/861 Datasheet*. Rev. 2588A - 11/06. http://www.atmel.com/dyn/resources/prod_documents/doc2588.pdf: , 2006

[Bredendiek 2000–2007] BREDENDIEK, Jörg: *Schaltregler und Transverter (verschiedene Webseiten und Schaltungen)*. http://www.sprut.de/electronic/switch/index.htm: , 2000–2007

[Brown 2001] BROWN, Marty: *Power supply cookbook*. Newnes, 2001. – ISBN 0-7506-7329-X

[Durcansky 1999] DURCANSKY, Georg: *EMV-gerechtes Gerätedesign*. Franzis', 1999. – ISBN 3-7723-5388-6

[Hering u. a. 2001] HERING, Ekbert ; BRESSLER, Klaus ; GUTEKUNST, Jürgen: *Elektronik für Ingenieure*. Springer, 2001. – ISBN 3-540-41738-9

[Horowitz und Hill 1998] HOROWITZ, Paul ; HILL, Winfield: *The Art of Electronics*. Cambridge University Press, 1998. – ISBN 0-521-37095-7

[Kilgenstein 1986] KILGENSTEIN, Otmar: *Schaltnetzteile in der Praxis*. Vogel, 1986. – ISBN 3-8023-0727-5

[Klaschke 1996] KLASCHKE, Günter: *Professionelle Schaltungstechnik, Band 3*. Franzis', 1996. – ISBN 3-7723-4004-0

[Lindner u. a. 2004] LINDNER, Helmut ; BAUER, Harry ; LEHMANN, Constans ; LINDNER, Hartmut: *Taschenbuch der Elektrotechnik und Elektronik*. Fachbuchverlag Leipzig, 2004. – ISBN 3-446-22546-3

[Michel 2003] MICHEL, Manfred: *Leistungselektronik*. Springer, 2003. – ISBN 3-540-02110-8

[National Semiconductor 2006] NATIONAL SEMICONDUCTOR: *LM5112SD Datasheet*. April 2006. http://www.national.com/ds.cgi/LM/LM5112.pdf: , 2006

[Ross 1997] ROSS, J. N.: *The Essence of Power Electronics*. Prentice Hall Europe, 1997. – ISBN 0-13-525643-7

[Seifart 1988] SEIFART, Manfred: *Analoge Schaltungen*. Hüthig, 1988. – ISBN 3-7785-1652-3

[Specovius 2003] SPECOVIUS, Joachim: *Grundkurs Leistungselektronik*. Vieweg, 2003. – ISBN 3-528-03963-9

[Thiel 1995] THIEL, Udo Leonhard: *Schaltnetzteile erfolgreich planen und dimensionieren*. Franzis', 1995. – ISBN 3-7723-7682-7

[Tietze und Schenk 1999] TIETZE, Ulrich ; SCHENK, Christoph: *Halbleiter-Schaltungstechnik*. Springer, 1999. – ISBN 3-540-64192-0

[Verschiedene Autoren verschiedene Jahre] VERSCHIEDENE AUTOREN: *Professionelle Schaltungstechnik, Band 1-12*. Franzis', verschiedene Jahre

[Vishay 2007] VISHAY: *Si7852DP Datasheet*. Rev. D. `http://www.vishay.com/docs/71627/71627.pdf:`, 2007

[Zantis 1994] ZANTIS, Franz-Peter: *Schaltnetzteile*. elektor, 1994. – ISBN 3-928051-75-X

Abbildungsverzeichnis

Tabellenverzeichnis

A Spice Schaltungsbeschreibungen

A.1 Step-Up-Wandler

A.1.1 Grundschaltung

Listing A.1: step-up.cir

```
* Step-Up-Wandler
* Datei: step-up.cir

Vin vin 0 24

Vpwm pwm 0 PULSE(0 5 0 1n 1n 0.499u 1u)

L1 vin n1 33u Rser=36.8m
Xt n1 pwm 0 Si7852DP
Xd n1 vout ss36
C1 vout 0 8.9u

Rload vout 0 32

.tran 2m
.lib models.lib
```

A.1.2 Verschiedene Eingangsspannungen

Listing A.2: step-up_var_Uin.cir

```
* Step-Up-Wandler
* Datei: step-up_var_Uin.cir

Vin1 vin1 0 12
Vin2 vin2 0 18
Vin3 vin3 0 24
Vpwm pwm 0 PULSE(0 5 0 1n 1n 0.499u 1u)

* Wandler 1 (12 V)
L11 vin1 n11 33u Rser=36.8m
Xt1 n11 pwm 0 Si7852DP
Xd1 n11 vout1 ss36
C11 vout1 0 8.9u

Rload1 vout1 0 32

* Wandler 2 (18 V)
L12 vin2 n12 33u Rser=36.8m
Xt2 n12 pwm 0 Si7852DP
Xd2 n12 vout2 ss36
C12 vout2 0 8.9u

Rload2 vout2 0 32

* Wandler 3 (24 V)
L13 vin3 n13 33u Rser=36.8m
Xt3 n13 pwm 0 Si7852DP
Xd3 n13 vout3 ss36
C13 vout3 0 8.9u

Rload3 vout3 0 32

.tran 2m
.lib models.lib
```

A.1.3 Verschiedene Schaltfrequenzen

Listing A.3: step-up_var_f.cir

```
* Step-Up-Wandler
* Datei: step-up_var_f.cir

Vin vin 0 24

* Wandler 1 (1 MHz)
Vpwm1 pwm1 0 PULSE(0 5 0 1n 1n 0.499u 1u)

L11 vin n11 33u Rser=36.8m
Xt1 n11 pwm1 0 Si7852DP
Xd1 n11 vout1 ss36
C11 vout1 0 8.9u

Rload1 vout1 0 32

* Wandler 2 (0,5 MHz)
Vpwm2 pwm2 0 PULSE(0 5 0 1n 1n 1.009u 2u)

L12 vin n12 33u Rser=36.8m
Xt2 n12 pwm2 0 Si7852DP
Xd2 n12 vout2 ss36
C12 vout2 0 8.9u

Rload2 vout2 0 32

* Wandler 3 (2 MHz)
Vpwm3 pwm3 0 PULSE(0 5 0 1n 1n 0.245u 0.5u)

L13 vin n13 33u Rser=36.8m
Xt3 n13 pwm3 0 Si7852DP
Xd3 n13 vout3 ss36
C13 vout3 0 8.9u

Rload3 vout3 0 32

.tran 2m
.lib models.lib
```

A.1.4 Verschiedene Tastverhältnisse

Listing A.4: step-up_var_P.cir

```
* Step-Up-Wandler
* Datei: step-up_var_P.cir

Vin vin 0 24

* Wandler 1 (50%)
Vpwm1 pwm1 0 PULSE(0 5 0 1n 1n 0.499u 1u)

L11 vin n11 33u Rser=36.8m
Xt1 n11 pwm1 0 Si7852DP
Xd1 n11 vout1 ss36
C11 vout1 0 8.9u

Rload1 vout1 0 32

* Wandler 2 (25%)
Vpwm2 pwm2 0 PULSE(0 5 0 1n 1n 0.249u 1u)

L12 vin n12 33u Rser=36.8m
Xt2 n12 pwm2 0 Si7852DP
Xd2 n12 vout2 ss36
C12 vout2 0 8.9u

Rload2 vout2 0 32

* Wandler 3 (75%)
Vpwm3 pwm3 0 PULSE(0 5 0 1n 1n 0.749u 1u)

L13 vin n13 33u Rser=36.8m
Xt3 n13 pwm3 0 Si7852DP
Xd3 n13 vout3 ss36
C13 vout3 0 8.9u

Rload3 vout3 0 32

.tran 2m
.lib models.lib
```

A.1.5 Verschiedene Induktivitäten

Listing A.5: step-up_var_L.cir

```
* Step-Up-Wandler
* Datei: step-up_var_L.cir

Vin vin 0 24
Vpwm pwm 0 PULSE(0 5 0 1n 1n 0.499u 1u)

* Wandler 1 (33 uH)
L11 vin n11 33u Rser=36.8m
Xt1 n11 pwm 0 Si7852DP
Xd1 n11 vout1 ss36
C11 vout1 0 8.9u

Rload1 vout1 0 32

* Wandler 2 (10 uH)
L12 vin n12 10u Rser=36.8m
Xt2 n12 pwm 0 Si7852DP
Xd2 n12 vout2 ss36
C12 vout2 0 8.9u

Rload2 vout2 0 32

* Wandler 3 (100 uH)
L13 vin n13 100u Rser=36.8m
Xt3 n13 pwm 0 Si7852DP
Xd3 n13 vout3 ss36
C13 vout3 0 8.9u

Rload3 vout3 0 32

.tran 2m
.lib models.lib
```

A.1.6 Verschiedene Kapazitäten

Listing A.6: step-up_var_C.cir

```
* Step-Up-Wandler
* Datei: step-up_var_C.cir

Vin vin 0 24
Vpwm pwm 0 PULSE(0 5 0 1n 1n 0.499u 1u)

* Wandler 1 (8,9 uF)
L11 vin n11 33u Rser=36.8m
Xt1 n11 pwm 0 Si7852DP
Xd1 n11 vout1 ss36
C11 vout1 0 8.9u

Rload1 vout1 0 32

* Wandler 2 (4,5 uF)
L12 vin n12 33u Rser=36.8m
Xt2 n12 pwm 0 Si7852DP
Xd2 n12 vout2 ss36
C12 vout2 0 4.5u

Rload2 vout2 0 32

* Wandler 3 (17,7 uF)
L13 vin n13 33u Rser=36.8m
Xt3 n13 pwm 0 Si7852DP
Xd3 n13 vout3 ss36
C13 vout3 0 17.7u

Rload3 vout3 0 32

.tran 2m
.lib models.lib
```

A.1.7 Verschiedene Lasten

Listing A.7: step-up_var_load.cir

```
* Step-Up-Wandler
* Datei: step-up_var_load.cir

Vin vin 0 24
Vpwm pwm 0 PULSE(0 5 0 1n 1n 0.499u 1u)

* Wandler 1 (1,5 A)
L11 vin n11 33u Rser=36.8m
Xt1 n11 pwm 0 Si7852DP
Xd1 n11 vout1 ss36
C11 vout1 0 8.9u

Rload1 vout1 0 32

* Wandler 2 (0,75 A)
L12 vin n12 33u Rser=36.8m
Xt2 n12 pwm 0 Si7852DP
Xd2 n12 vout2 ss36
C12 vout2 0 8.9u

Rload2 vout2 0 64

* Wandler 3 (3 A)
L13 vin n13 33u Rser=36.8m
Xt3 n13 pwm 0 Si7852DP
Xd3 n13 vout3 ss36
C13 vout3 0 8.9u

Rload3 vout3 0 16

.tran 2m
.lib models.lib
```

A.1.8 Lastwechsel

Listing A.8: step-up_loadchange.cir

```
* Step-Up-Wandler
* Datei: step-up_loadchange.cir

Vin vin 0 24
Vpwm pwm 0 PULSE(0 5 0 1n 1n 0.499u 1u)

L1 vin n1 33u Rser=36.8m
Xt n1 pwm 0 Si7852DP
Xd n1 vout ss36
C1 vout 0 8.9u

Iload vout 0 PWL(0 50m 5m 50m 5.001m 2 7m 2 9m 1 9.5m 1 9.501m 2
+11.5m 2 11.501m 1 13.5m 1 15.5m 2 17m 2 24m 0) load

.tran 27m
.lib models.lib
```

A.2 Step-Down-Wandler

A.2.1 Grundschaltung

Listing A.9: step-down.cir

```
* Step-Down-Wandler
* Datei: step-down.cir

Vin vin 0 24

Vpwm pwm 0 PULSE(19 24 0 1n 1n 0.486u 1u)

Xt n1 pwm vin Si4948BEY
Xd 0 n1 ss16
L1 n1 vout 22u Rser=0.21m
C1 vout 0 4.5u

Rload vout 0 8

.tran 0.5m
.lib models.lib
```

A.2.2 Verschiedene Eingangsspannungen

Listing A.10: step-down_var_Uin.cir

```
* Step-Down-Wandler
* Datei: step-down_var_Uin.cir

Vin1 vin1 0 18
Vin2 vin2 0 24
Vin3 vin3 0 30

Vpwm1 pwm1 0 PULSE(13 18 0 1n 1n 0.486u 1u)
Vpwm2 pwm2 0 PULSE(19 24 0 1n 1n 0.486u 1u)
Vpwm3 pwm3 0 PULSE(25 30 0 1n 1n 0.486u 1u)

* Wandler 1 (18 V)
Xt1 n11 pwm1 vin1 Si4948BEY
Xd1 0 n11 ss16
L11 n11 vout1 22u Rser=0.21m
C11 vout1 0 4.5u

Rload1 vout1 0 8

* Wandler 2 (24 V)
Xt2 n12 pwm2 vin2 Si4948BEY
Xd2 0 n12 ss16
L12 n12 vout2 22u Rser=0.21m
C12 vout2 0 4.5u

Rload2 vout2 0 8

* Wandler 3 (30 V)
Xt3 n13 pwm3 vin3 Si4948BEY
Xd3 0 n13 ss16
L13 n13 vout3 22u Rser=0.21m
C13 vout3 0 4.5u

Rload3 vout3 0 8

.tran 0.5m
.lib models.lib
```

A.2.3 Verschiedene Schaltfrequenzen

Listing A.11: step-down_var_f.cir

```
* Step-Down-Wandler
* Datei: step-down_var_f.cir

Vin vin 0 24

Vpwm1 pwm1 0 PULSE(19 24 0 1n 1n 0.486u 1u)
Vpwm2 pwm2 0 PULSE(19 24 0 1n 1n 0.972u 2u)
Vpwm3 pwm3 0 PULSE(19 24 0 1n 1n 0.243u 0.5u)

* Wandler 1 (1 MHz)
Xt1 n11 pwm1 vin Si4948BEY
Xd1 0 n11 ss16
L11 n11 vout1 22u Rser=0.21m
C11 vout1 0 4.5u

Rload1 vout1 0 8

* Wandler 2 (500 kHz)
Xt2 n12 pwm2 vin Si4948BEY
Xd2 0 n12 ss16
L12 n12 vout2 22u Rser=0.21m
C12 vout2 0 4.5u

Rload2 vout2 0 8

* Wandler 3 (2 MHz)
Xt3 n13 pwm3 vin Si4948BEY
Xd3 0 n13 ss16
L13 n13 vout3 22u Rser=0.21m
C13 vout3 0 4.5u

Rload3 vout3 0 8

.tran 0.5m
.lib models.lib
```

A.2.4 Verschiedene Tastverhältnisse

Listing A.12: step-down_var_P.cir

```
* Step-Down-Wandler
* Datei: step-down_var_P.cir

Vin vin 0 24

Vpwm1 pwm1 0 PULSE(19 24 0 1n 1n 0.499u 1u)
Vpwm2 pwm2 0 PULSE(19 24 0 1n 1n 0.749u 1u)
Vpwm3 pwm3 0 PULSE(19 24 0 1n 1n 0.249u 1u)

* Wandler 1 (50%)
Xt1 n11 pwm1 vin Si4948BEY
Xd1 0 n11 ss16
L11 n11 vout1 22u Rser=0.21m
C11 vout1 0 4.5u

Rload1 vout1 0 8

* Wandler 2 (75%)
Xt2 n12 pwm2 vin Si4948BEY
Xd2 0 n12 ss16
L12 n12 vout2 22u Rser=0.21m
C12 vout2 0 4.5u

Rload2 vout2 0 8

* Wandler 3 (25%)
Xt3 n13 pwm3 vin Si4948BEY
Xd3 0 n13 ss16
L13 n13 vout3 22u Rser=0.21m
C13 vout3 0 4.5u

Rload3 vout3 0 8

.tran 0.5m
.lib models.lib
```

A.2.5 Verschiedene Induktivitäten

Listing A.13: step-down_var_L.cir

```
* Step-Down-Wandler
* Datei: step-down_var_L.cir

Vin vin 0 24

Vpwm pwm 0 PULSE(19 24 0 1n 1n 0.486u 1u)

* Wandler 1 (22 uH)
Xt1 n11 pwm vin Si4948BEY
Xd1 0 n11 ss16
L11 n11 vout1 22u Rser=0.21m
C11 vout1 0 4.5u

Rload1 vout1 0 8

* Wandler 2 (10 uH)
Xt2 n12 pwm vin Si4948BEY
Xd2 0 n12 ss16
L12 n12 vout2 10u Rser=0.21m
C12 vout2 0 4.5u

Rload2 vout2 0 8

* Wandler 3 (100 uH)
Xt3 n13 pwm vin Si4948BEY
Xd3 0 n13 ss16
L13 n13 vout3 100u Rser=0.21m
C13 vout3 0 4.5u

Rload3 vout3 0 8

.tran 0.5m
.lib models.lib
```

A.2.6 Verschiedene Kapazitäten

Listing A.14: step-down_var_C.cir

```
* Step-Down-Wandler
* Datei: step-down_var_C.cir

Vin vin 0 24

Vpwm pwm 0 PULSE(19 24 0 1n 1n 0.486u 1u)

* Wandler 1 (4.5 uF)
Xt1 n11 pwm vin Si4948BEY
Xd1 0 n11 ss16
L11 n11 vout1 22u Rser=0.21m
C11 vout1 0 4.5u

Rload1 vout1 0 8

* Wandler 2 (10 uF)
Xt2 n12 pwm vin Si4948BEY
Xd2 0 n12 ss16
L12 n12 vout2 22u Rser=0.21m
C12 vout2 0 10u

Rload2 vout2 0 8

* Wandler 3 (100 nF)
Xt3 n13 pwm vin Si4948BEY
Xd3 0 n13 ss16
L13 n13 vout3 22u Rser=0.21m
C13 vout3 0 0.1u

Rload3 vout3 0 8

.tran 0.5m
.lib models.lib
```

A.2.7 Verschiedene Lasten

Listing A.15: step-down_var_load.cir

```
* Step-Down-Wandler
* Datei: step-down_var_load.cir

Vin vin 0 24

Vpwm pwm 0 PULSE(19 24 0 1n 1n 0.486u 1u)

* Wandler 1 (1,5 A)
Xt1 n11 pwm vin Si4948BEY
Xd1 0 n11 ss16
L11 n11 vout1 22u Rser=0.21m
C11 vout1 0 4.5u

Rload1 vout1 0 8

* Wandler 2 (750 mA)
Xt2 n12 pwm vin Si4948BEY
Xd2 0 n12 ss16
L12 n12 vout2 22u Rser=0.21m
C12 vout2 0 4.5u

Rload2 vout2 0 16

* Wandler 3 (3 A)
Xt3 n13 pwm vin Si4948BEY
Xd3 0 n13 ss16
L13 n13 vout3 22u Rser=0.21m
C13 vout3 0 4.5u

Rload3 vout3 0 4

.tran 0.5m
.lib models.lib
```

A.2.8 Lastwechsel

Listing A.16: step-down_loadchange.cir

```
* Step-Down-Wandler
* Datei: step-down_loadchange.cir

Vin vin 0 24

Vpwm pwm 0 PULSE(19 24 0 1n 1n 0.486u 1u)

Xt n1 pwm vin Si4948BEY
Xd 0 n1 ss16
L1 n1 vout 22u Rser=0.21m
C1 vout 0 4.5u

Iload vout 0 PWL(0 50m 5m 50m 5.001m 2 7m 2 9m 1 9.5m 1 9.501m 2
+11.5m 2 11.501m 1 13.5m 1 15.5m 2 17m 2 24m 0) load

.tran 27m
.lib models.lib
```

A.3 MOSFET-Schaltverhalten

Listing A.17: mosfet_sim.cir

```
* MOSFET-Schaltverhalten
* Datei: mosfet_sim.cir

Vin vin 0 24
Vctrl ctrl 0 0
R1 vin n1 10
M1 n1 ctrl 0 nmosfet

.model nmosfet NMOS(Vto=5 Kp=1 Rds=10Meg)
.DC Vctrl 0 24 0.01
```

A.4 Subcircuits (models.lib)

Listing A.18: models.lib; Si4948BEY

```
*Nov 21, 2005
*Doc. ID: 78263, S-52399, Rev. B
*File Name: Si4948BEY_PS.txt and Si4948BEY_PS.lib
.SUBCKT Si4948BEY 4 1 2
M1  3 1 2 2 PMOS W=1050913u L=0.25u
M2  2 1 2 4 NMOS W=1050913u L=0.50u
R1  4 3     RTEMP 50E-3
CGS 1 2     470E-12
DBD 4 2     DBD
************************************************************
.MODEL  PMOS       PMOS  ( LEVEL  = 3           TOX    = 5E-8
+ RS     = 40E-3           RD     = 0           NSUB   = 1.3E17
+ KP     = 4.5E-6          UO     = 400
+ VMAX   = 0               XJ     = 5E-7        KAPPA  = 4E-2
+ ETA    = 1E-4            TPG    = -1
+ IS     = 0               LD     = 0
+ CGSO   = 0               CGDO   = 0           CGBO   = 0
+ NFS    = 0.8E12          DELTA  = 0.1)
************************************************************
.MODEL  NMOS       NMOS  ( LEVEL  = 3           TOX    = 5E-8
+NSUB    = 3E16            TPG    = -1)
************************************************************
.MODEL DBD D (CJO=80E-12 VJ=0.38 M=0.32
+RS=0.1 FC=0.5 IS=1E-12 TT=6E-8 N=1 BV=61)
************************************************************
.MODEL RTEMP RES (TC1=10E-3 TC2=5.5E-6)
************************************************************
.ENDS
```

Quelle: `http://www.vishay.com/docs/78263/si4948be.lib`

Listing A.19: models.lib; Si7852DP

```
*February 20, 2006
*Doc. ID: 79124, S-60245, Rev. B
*File Name: Si7852DP_PS.txt and Si7852DP_PS.lib
*This document is intended as a SPICE modeling guideline and does not
*constitute a commercial product data sheet.  Designers should refer to
*the appropriate data sheet of the same number for guaranteed
*specification limits.
.SUBCKT Si7852DP 4 1 2
M1  3 1 2 2 NMOS W=3862950u L=0.50u
M2  2 1 2 4 PMOS W=3862950u L=0.75u
R1  4 3     RTEMP 8E-3
CGS 1 2     900E-12
DBD 2 4     DBD
*************************************************************************
.MODEL  NMOS       NMOS  (LEVEL  = 3               TOX    = 10E-8
+ RS     = 2.3E-3         RD     = 0               NSUB   = 1.15E17
+ KP     = 0.9E-5         UO     = 650
+ VMAX   = 0              XJ     = 5E-7            KAPPA  = 1E-1
+ ETA    = 1.0E-4         TPG    = 1
+ IS     = 0              LD     = 0
+ CGSO   = 0              CGDO   = 0               CGBO   = 0
+ NFS    = 0.8E12         DELTA  = 0.1)
*************************************************************************
.MODEL  PMOS       PMOS (LEVEL   = 3               TOX    = 10E-8
+NSUB    = 1.5E16         TPG    = -1)
*************************************************************************
.MODEL DBD D (CJO=600E-12  VJ=0.38  M=0.33
+RS=1 FC=0.5 IS=1E-12 TT=7E-8 N=1 BV=80.2)
*************************************************************************
.MODEL RTEMP RES (TC1=8E-3  TC2=5.5E-6)
*************************************************************************
.ENDS
```

Quelle: `http://www.vishay.com/docs/79124/si7852dp.lib`
Zeilenlänge wurde angepasst

Listing A.20: models.lib; SS16

```
**********************************
* Copyright:                     *
*   Thomatronik GmbH, Germany    *
*   info@thomatronik.de          *
**********************************
*   SPICE3
.subckt ss16 1 2
ddio 1 2 legd
dgr 1 2 grd
.model legd d is = 9.5727E-008 n = 1.14007 rs = 0.0887941
+ eg = 0.849688 xti = 2.99996
+ cjo = 2.35293E-010 vj = 0.610616 m = 0.540837 fc = 0.5
+ tt = 1.4427E-009 bv = 66 ibv = 5 af = 1 kf = 0
.model grd d is = 9.6926E-009 n = 1.83874 rs = 0.015442
+ eg = 1.49044 xti = 2.01516
.ends
```

Quelle: http://www.vishay.com/docs/88206/ss16.txt

Listing A.21: models.lib; SS36

```
**********************************
* Model created by                *
*   Uni.Dipl.-Ing. Arpad Buermen *
*   arpad.burmen@ieee.org         *
* Copyright:                      *
*   Thomatronik GmbH, Germany     *
*   info@thomatronik.de           *
**********************************
*   SPICE3
.subckt ss36 1 2
ddio 1 2 legd
dgr 1 2 grd
.model legd d is = 1.83591E-007 n = 1.85181 rs = 0.00755407
+ eg = 1.47332 xti = 0.526784
+ cjo = 6.58521E-010 vj = 1.00748 m = 0.547755 fc = 0.5
+ tt = 1.4427E-009 bv = 66 ibv = 4.0 af = 1 kf = 0
.model grd d is = 5.67518E-009 n = 0.775515 rs = 0.0452019
+ eg = 0.692385 xti = 1.42251
.ends
```

Quelle: http://www.vishay.com/docs/88171/ss36.txt

B Quellcodebeispiele

B.1 Soft-PWM I

Listing B.1: Beispielprogramm zur Soft-PWM I

```
#include <avr/io.h>

volatile uint8_t pwm_val;

int main(void) {
  uint8_t counter = 0;
  pwm_val = 0x42;   // konstanter Wert als Beispiel
  DDRA = (1<<PA0);  // Port A, Pin 0 als Ausgang
  PORTA = (1<<PA0); // Port A, Pin 0 auf 1 setzen
  for (;;) {
    if (counter == 0) {
      // Port A, Pin 0 auf 1 setzen
      PORTA |= (1<<PA0);
    }
    if (counter == pwm_val) {
      // Port A, Pin 0 auf 0 setzen
      PORTA &= ~(1<<PA0);
    }
    counter++;
  }
  return 0;
}
```

B.2 Soft-PWM II

Listing B.2: Beispielprogramm zur Soft-PWM II

```
#include <avr/io.h>
#include <avr/interrupt.h>

volatile uint8_t pwm_val;
volatile uint8_t counter = 0;

int main(void) {
  pwm_val = 0x42;   // konstanter Wert als Beispiel
  PORTA = 0;
  DDRA = (1<<PA0);  // Port A, Pin 0 als Ausgang
  TIMSK = (1<<TOIE0); // Overflow Interrupt
  sei(); // Interrupts einschalten
  TCCR0B = (0b001<<CS00); // Counter starten
  for (;;) {
    if (counter == 0) {
      // Port A, Pin 0 auf 1 setzen
      PORTA |= (1<<PA0);
    }
    if (counter == pwm_val) {
      // Port A, Pin 0 auf 0 setzen
      PORTA &= ~(1<<PA0);
    }
  }
  return 0;
}

ISR(TIMER0_OVF_vect) {
  counter++;
}
```

B.3 Soft-PWM III

Listing B.3: Beispielprogramm zur Soft-PWM III

```
#include <avr/io.h>
#include <avr/interrupt.h>

volatile uint8_t pwm_val;

int main(void) {
  pwm_val = 0x42;   // konstanter Wert als Beispiel
  PORTA = 0;
  DDRA = (1<<PA0);  // Port A, Pin 0 als Ausgang
  TIMSK = (1<<TOIE0); // Overflow Interrupt
  sei(); // Interrupts einschalten
  TCCR0B = (0b001<<CS00); // Counter starten
  for (;;);
  return 0;
}

ISR(TIMER0_OVF_vect) {
  static uint8_t counter = 0;
  if (counter == 0) {
    // Port A, Pin 0 auf 1 setzen
    PORTA |= (1<<PA0);
  }
  if (counter == pwm_val) {
    // Port A, Pin 0 auf 0 setzen
    PORTA &= ~(1<<PA0);
  }
  counter++;
}
```

B.4 PWM mit Output-Compare-Interrupt

Listing B.4: Beispielprogramm zur PWM-Erzeugung mit Output-Compare-Interrupt

```
#include <avr/io.h>
#include <avr/interrupt.h>

int main(void) {
  PORTA = 0;
  DDRA = (1<<PA0);  // Port A, Pin 0 als Ausgang
  TIMSK = (1<<TOIE0)|(1<<OCIE0A); // Overflow Interrupt
  OCR0A = 0x42;   // konstanter Wert als Beispiel
  sei(); // Interrupts einschalten
  TCCR0B = (0b001<<CS00); // Counter starten
  for (;;); // leere Endlosschleife
  return 0;
}

ISR(TIMER0_OVF_vect) {
  // Port A, Pin 0 auf 1 setzen
  PORTA |= (1<<PA0);
}

ISR(TIMER0_COMPA_vect) {
  // Port A, Pin 0 auf 0 setzen
  PORTA &= ~(1<<PA0);
}
```

B.5 PWM mit PWM-Einheit

Listing B.5: Beispielprogramm zur PWM-Erzeugung mit PWM-Einheit

```
#include <avr/io.h>

int main(void) {
  PORTB = 0;
  DDRB = (1<<PB1);  // Port B, Pin 1 als Ausgang (OC1A)
  OCR1A = 0x42; // konstanter Wert als Beispiel
  OCR1C = 0xFF; // maximaler Zählerstand
  TCCR1A = (0b10<<COM1A0)|(1<<PWM1A);
  TCCR1B = (0b0001<<CS10);
  for (;;); // leere Endlosschleife
  return 0;
}
```

B.6 AD-Wandler

Listing B.6: Beispielprogramm für das Interleaving am AD-Wandler

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

/*
 * Grundeinstellungen für das ADMUX-Register:
 * - VCC als Referenz,
 * - "Left Adjust Result", die 8 höchstwertigen
 *   Bits des Ergebnisses liegen im Register ADCH
 */
#define ADMUX_BASE ((0<<REFS1)|(0<<REFS0)|(1<<ADLAR))

// Speicher für die Ergebnisse der vier Kanäle
uint8_t messwerte[4];

int main(void) {
  ADMUX = ADMUX_BASE | 0; // Neuer Kanal: ADC0
  /*
   * - AD-Wandler einschalten,
   * - Interrupt-Flag löschen,
   * - Interrupt einschalten,
   * - Taktteilung F_CPU/8
   */
  ADCSRA = (1<<ADEN)|(0<<ADSC)|(0<<ADATE)|(1<<ADIF)
         |(1<<ADIE)|(0b011<<ADPS0);
  sei();
  ADCSRA |= (1<<ADSC); // Wandlung starten
  _delay_us(2); // Sicherstellen, dass die Wandlung läuft
  ADMUX = ADMUX_BASE | 1; // Neuer Kanal: ADC1
  for (;;) {
    asm("nop"); // nichts tun
  }
}

ISR(ADC_vect) {
  static uint8_t state = 0;
  ADCSRA |= (1<<ADSC); // Wandlung starten
  _delay_us(2); // Sicherstellen, dass die Wandlung läuft
  /*
   * Der switch-Block lässt sich hier auf wenige Zeilen
```

```
   * zusammenfassen, vereinfacht aber in dieser Form die
   * unterschiedliche Verarbeitung der Ergebnisse der Kanäle.
   */
  switch(state) {
    case 0:
      messwerte[0] = ADCH;
      state = 1;
      ADMUX = ADMUX_BASE | 2; // Neuer Kanal: ADC2
      break;
    case 1:
      messwerte[1] = ADCH;
      state = 2;
      ADMUX = ADMUX_BASE | 3; // Neuer Kanal: ADC3
      break;
    case 2:
      messwerte[2] = ADCH;
      state = 3;
      ADMUX = ADMUX_BASE | 0; // Neuer Kanal: ADC0
      break;
    case 3:
      messwerte[3] = ADCH;
      state = 0;
      ADMUX = ADMUX_BASE | 1; // Neuer Kanal: ADC1
      break;
    default: // Unbekannter Zustand
      ADMUX = ADMUX_BASE | 0; // Neuer Kanal: ADC0
      _delay_us(2);
      ADCSRA |= (1<<ADSC);
      state = 0;
  }
}
```

B.7 Analog-Komparator

Listing B.7: Beispielprogramm für den Analog-Komparator

```
#include <avr/io.h>
#include <avr/interrupt.h>

int main(void) {
  DDRA = (1<<PA0);
  PORTA = 0;
  // Analog-Komparator initialisieren
  ACSRA =  (0<<ACD)|(1<<ACBG)|(0<<ACIE)|(0<<ACME)
         |(1<<ACIS1)|(0<<ACIS0)|(1<<ACIE);
  ACSRB = 0;
  sei();
  for (;;);
}

ISR(ANA_COMP_vect) {
  // Ausgang invertieren
  PORTA ^= (1<<PA0);
}
```

B.8 SPI-Kommunnikation (USI)

Listing B.8: Beispielprogramm für die SPI-Kommunikation mit dem USI

```
#include <avr/io.h>
#include <avr/interrupt.h>

volatile uint8_t rxidx;
volatile uint8_t rxbuf[4];
volatile uint8_t txbuf[4];

// Kompletter Speicher für das Beispiel, auf einem ATtiny861
// wären hiermit 3/4 des verfügbaren Speichers belegt.
volatile uint8_t memory[128][3];

int main(void) {
  // Jede Flanke erzeugt einen IRQ
  MCUCR = (0<<ISC01)|(1<<ISC00);
  // Externen Interrupt 0 einschalten
  GIMSK = (1<<INT0);
  // USI auf PORTB mappen
  USIPP = (0<<USIPOS);
  // 3-wire-mode, externer Takt
  USICR = (0<<USIOIE)|(0b01<<USIWM0)|(0b10<<USICS0)|(0<<USICLK);
  // Overflow-Interrupt-Flag löschen
  USISR |= (1<<USIOIF);
  sei();
  for (;;);
}

ISR(INT0_vect) {
  if (PINB&(1<<PB6)) { // Deselect
    // Overflow Interrupt disable
    USICR &= ~(1<<USIOIE);
    // SPI-State-Machine in den Startzustand setzen
    rxidx = 0;
    // 0xff als erstes Byte senden
    USIDR = 0xff;
  } else {  // Select
    // Overflow Interrupt enable
    USICR |= (1<<USIOIE);
    // USI-Zähler zurücksetzen
    USISR &= ~(0x0f<<USICNT0);
    // Overflow-Interrupt-Flag löschen
```

```
    USISR |= (1<<USIOIF);
  }
}

ISR(USI_OVF_vect) { // SPI-Byte empfangen
  static uint8_t write_state = 0;

  // empfangenes Byte speichern
  rxbuf[rxidx] = USIDR;
  // Zustand/Empfangspufferindex ändern
  rxidx = (rxidx+1)&3;
  if (rxidx == 1) { // Adressbyte wurde empfangen
    // Sendepuffer initialisieren
    txbuf[0] = 0xff;
    txbuf[1] = 0;
    txbuf[2] = 0;
    txbuf[3] = 0;
    switch (rxbuf[0]) {
      case 0:   // Adresse 0 schreiben
        write_state = 1;
        break;
      case 1:   // Adresse 0 lesen
        txbuf[1] = memory[0][0];
        txbuf[2] = memory[0][1];
        txbuf[3] = memory[0][2];
        break;
      case 2:   // Adresse 1 schreiben
        write_state = 2;
        break;
      case 3:   // Adresse 1 lesen
        txbuf[1] = memory[1][0];
        txbuf[2] = memory[1][1];
        txbuf[3] = memory[1][2];
        break;
      // Weitere Adressen nach dem gleichen Schema
    }
  }
  // nächstes Byte zum Senden in das Datenregister legen
  USIDR = txbuf[rxidx];
  if (rxidx == 0) { // 4. Byte wurde empfangen
    switch (write_state) {
      case 1:
        memory[0][0] = txbuf[1];
        memory[0][1] = txbuf[2];
```

```
        memory[0][2] = txbuf[3];
        write_state = 0;
        break;
      case 2:
        memory[1][0] = txbuf[1];
        memory[1][1] = txbuf[2];
        memory[1][2] = txbuf[3];
        write_state = 0;
        break;
      // Weitere Adressen nach dem gleichen Schema
      default:
        write_state = 0;
    }
  }
  // Overflow-Interrupt-Flag löschen
  USISR |= (1<<USIOIF);
}
```

B.9 Rechenzeitoptimierung

Listing B.9: Assemblercode für die Division eines 8-Bit-Integers durch 1,6

```
; y = x/1.6;
; x liegt in r16, y soll in r17 liegen
MOV r17, r16  ; y = x;
ASR r16       ; x >>= 1;
ASR r17       ; y >>= 1;
ASR r17       ; y >>= 1;
ASR r17       ; y >>= 1;
ADD r17, r16  ; y += x;
```

C Komplettes Mikrocontrollerprogramm

Listing C.1: Verwendeter Quellcode für den Schaltregler

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

// Autostart der einzelnen Regler
//#define AUTOSTART_12V
//#define AUTOSTART_48V

// LED-Steuerung
#define LED_STATUS 0
#define LED_12V 1
#define LED_48V 2
#define LED_EIN(x)  PORTA |= (1<<x)
#define LED_AUS(x)  PORTA &= ~(1<<x)

// Lastwiderstand
#define LAST_EIN()  PORTA |= (1<<3)
#define LAST_AUS()  PORTA &= ~(1<<3)

// Grundeinstellungen für das ADMUX-Register
#define ADMUX_BASE ((0<<REFS1)|(0<<REFS0)|(1<<ADLAR))

// Soll- und Grenzwerte des AD-Wandlers
#define AD_MAX_IN 132
#define AD_MIN_IN 60

#define AD_SOLL_48V 198
#define AD_MAX_48V 216

#define AD_SOLL_12V 185
#define AD_MAX_12V 200

// Flag-Register Definitionen
#define FLAG_REG GPIOR0
```

```
#define AUTO_12V 0
#define AUTO_48V 1
#define PWM_ACT_12V 2
#define PWM_ACT_48V 3
#define START_WAIT 7

// TOP-Wert für den PWM-Counter -- f_{PWM} = 1MHz
#define MAX_COUNT 0x3f

// Variablen für die Sollwerte
// Änderungsmöglichkeit nicht implementiert
volatile uint8_t soll_48V;
volatile uint8_t soll_12V;

volatile register uint8_t val_Uin asm("r2");
volatile register uint8_t val_48V asm("r3");
volatile register uint8_t val_12V asm("r4");

volatile register uint8_t pwm_48V asm("r5");
volatile register uint8_t pwm_12V asm("r6");

volatile uint8_t rxidx;
volatile uint8_t rxbuf[4];
volatile uint8_t txbuf[4];

int main(void) {
  soll_48V = AD_SOLL_48V;
  soll_12V = AD_SOLL_12V;

  PORTA = 0b00000000;
  DDRA = 0b00001111;
  PORTB = (1<<PB6);
  DDRB =  (0<<PB7)|0<<PB6)|(1<<PB5)|(0<<PB4)|
          (1<<PB3)|(0<<PB2)|(1<<PB1)|(0<<PB0);

  // Startverzögerung beim Einschalten
  FLAG_REG |= (1<<START_WAIT);
  TCCR0A = (1<<TCW0)|(0<<ICEN0)|(0<<WGM00);
  TIMSK = (1<<TOIE0);
  sei();
  TCCR0B = (0b011<<CS00);
  while (FLAG_REG & (1<<START_WAIT));
  cli();
  TIMSK = 0;
```

```
  PORTA = 0;

  // INT0 für SPI ~SS einstellen
  MCUCR = (0<<ISC01)|(1<<ISC00);
  GIMSK = (1<<INT0);

  // USI einstellen
  USIPP = (0<<USIPOS); // USI auf PORTB gemappt
  USICR = (0<<USIOIE)|(0<<USIWM1)|(1<<USIWM0)|
          (1<<USICS1)|(0<<USICS0)|(0<<USICLK);
  USISR = (1<<USIOIF);

  // AD-Wandler initialisieren
  ADMUX = ADMUX_BASE + 7; // Neuer Kanal: Uin
#if (F_CPU == 8000000)
  ADCSRA =  (1<<ADEN)|(0<<ADSC)|(0<<ADATE)|
            (1<<ADIF)|(1<<ADIE)|(0b011<<ADPS0); // 8MHz
#elif (F_CPU == 16000000)
  ADCSRA =  (1<<ADEN)|(0<<ADSC)|(0<<ADATE)|
            (1<<ADIF)|(1<<ADIE)|(0b100<<ADPS0); // 16MHz
#else
#error "F_CPU muss 8000000 oder 16000000 sein"
#endif
  ADCSRB = (0<<BIN)|(0<<GSEL)|(0<<REFS2)|(0<<MUX5)|(0b000<<ADTS0);

  // PLL starten
  PLLCSR = (0<<LSM)|(1<<PCKE)|(1<<PLLE);
  _delay_us(40); // Wartezeit zur PLL-Stabilisierung
  _delay_us(40);
  _delay_us(40);

  // Timer 1 (PWM) initialisieren
  TCCR1A =  (1<<COM1A1)|(0<<COM1A0)|(1<<COM1B1)|(0<<COM1B0)|
            (0<<FOC1A)|(0<<FOC1B)|(0<<PWM1A)|(0<<PWM1B);
  TCCR1B =  (0<<PWM1X)|(0<<PSR1)|(0<<DTPS11)|
            (0<<DTPS10)|(0b0000<<CS10);
  TCCR1C |= (1<<COM1D1)|(0<<COM1D0)|(0<<FOC1D)|(0<<PWM1D);
  TCCR1D =  (0<<FPIE1)|(0<<FPEN1)|(0<<FPNC1)|(0<<FPES1)|
            (0<<FPAC1)|(0<<FPF1)|(0b00<<WGM10);

  // TOP setzen
  TC1H = 0x00;
  OCR1C = MAX_COUNT;
```

```
  // PWM mit 0 starten
  pwm_12V = 0;
  pwm_48V = 0;
  OCR1B = pwm_12V;
  OCR1D = pwm_48V;

  // Automatische Regelung beim Einschalten
#ifdef AUTOSTART_48V
  FLAG_REG |= (1<<AUTO_48V);
  TCCR1C |= (1<<PWM1D); // 48V  PWM
  FLAG_REG |= (1<<PWM_ACT_48V);
  PORTA &= ~(1<<PA3); // Lastwiderstand abschalten
#endif
#ifdef AUTOSTART_12V
  FLAG_REG |= (1<<AUTO_12V);
  TCCR1A |= (1<<PWM1B); // 12V  PWM
  FLAG_REG |= (1<<PWM_ACT_12V);
#endif

  // warten, bis PLL stabil läuft, dann Timer starten
  while(!(PLLCSR&0x01));
  TCCR1B |= (0b0001<<CS10);

  // Analog-Komparator initialisieren
  ACSRA = (0<<ACD)|(1<<ACBG)|(0<<ACIE)|
          (0<<ACME)|(1<<ACIS1)|(0<<ACIS0);
  ACSRB = (1<<HSEL)|(1<<HLEV)|(0<<ACM2)|(0<<ACM1)|(0<<ACM0);
  ACSRA |= (1<<ACIE);

  // Interrupts einschalten
  sei();

  // erste AD-Wandlung starten
  ADCSRA |= (1<<ADSC);
  _delay_us(2);
  ADMUX = ADMUX_BASE + 4; // Neuer Kanal: 48V

  for(;;) {
    if ((PORTA & (1<<PA3))&&(ACSRA & (1<<ACO))) {
      LAST_AUS();
      TCCR1C |= (1<<PWM1D); // 48V PWM einschalten
    }
  }
  return 0;
```

```
}

// Überspannung auf der 48V-Leitung
ISR(ANA_COMP_vect) {
  TCCR1C &= ~(1<<PWM1D);  // 48V PWM abschalten
  LAST_EIN();
}

// AD-Wandlung abgeschlossen ->
// neue Wandlung starten, Wert speichern,
// übernächsten Kanal wählen
// ------------------------------------------------------
// Sicherstellen, dass vor der neuen
// Auswahl die neue Wandlung läuft:
// ADCSRA |= (1<<ADSC); --- 1us --- ADMUX = ADMUX_BASE + x;
// ------------------------------------------------------
// ADC(H) bleibt erhalten, bis die nächste Wandlung abgeschlossen
// ist (13-14us nach ADCSRA |= (1<<ADSC)). ADMUX kann 1us nach
// ADCSRA |= (1<<ADSC) auf den nächsten Wert gesetzt werden.
ISR(ADC_vect) {
  static uint8_t state;

  static int16_t diff;

  static int16_t diffsumme_48V;
  static int16_t diffsumme_12V;

  static uint8_t count;


  ADCSRA |= (1<<ADSC);

  LED_EIN(LED_STATUS);
  switch(state) {
    case 0: // Uin Wandlung beendet
      val_Uin = ADCH;
      if ((val_Uin < AD_MIN_IN)||(val_Uin > AD_MAX_IN)) {
        // Außerhalb des Eingangsspannungsbereiches
        DDRB &= ~((1<<PB3)|(1<<PB5));
      } else {
        // Innerhalb des Eingangsspannungsbereiches
        DDRB |= (1<<PB3)|(1<<PB5);
      }
      state++;
```

```
    ADMUX = ADMUX_BASE + 4; // Neuer Kanal: 12V
    break;

  case 1: // 48V Wandlung beendet
    val_48V = ADCH;
    // Regelung aktiv?
    if ((FLAG_REG & (1<<AUTO_48V)) &&
        (FLAG_REG & (1<<PWM_ACT_48V))) {
      diff = (soll_48V - val_48V);

      diffsumme_48V += diff;
      // Abweichung vom Sollwert positiv?
      if (((pwm_48V < 50) && (diff > 0)) ||
          (diff > 12) || (diffsumme_48V > (5*12)) ) {
        diffsumme_48V = 0;
        // Nachregeln
        if ((diff > 24) && (pwm_48V < 252)) {
            pwm_48V += 4;
        } else if (pwm_48V < 255) {
            pwm_48V++;
        }
        LED_AUS(LED_48V);
      } else if ( ((pwm_48V < 50) && (diff < 0)) ||
                  (diff < -12) || (diffsumme_48V < -(5*12)) ) {
        diffsumme_48V = 0;
        // Spannung viel zu hoch?
        if (val_48V > AD_MAX_48V) {
          // Neu einregeln
          pwm_48V = 1;
        } else {
          // Nachregeln
          if ((diff < -24) && (pwm_48V > 3)) {
            pwm_48V -= 4;
          } else if (pwm_48V > 0) {
            pwm_48V--;
          }
        }
        LED_AUS(LED_48V);
      } else {
        LED_EIN(LED_48V);
      }

    } else {
      // inaktive Regelung
```

```
      LED_AUS(LED_48V);
    }
    state++;
    ADMUX = ADMUX_BASE + 7; // Neuer Kanal: Uin
    break;

  case 2: // 12V Wandlung beendet
    val_12V = ADCH;

    // Regelung aktiv?
    if ((FLAG_REG & (1<<AUTO_12V)) &&
        (FLAG_REG & (1<<PWM_ACT_12V))) {
      diff = (soll_12V - val_12V);

      diffsumme_12V += diff;
      // Abweichung vom Sollwert positiv?
      if (((pwm_12V < 50) && (diff > 0)) ||
          (diffsumme_12V > 4) ) {
        diffsumme_12V = 0;
        // Nachregeln
        if ((diff > 16) && (pwm_12V < 248)) {
            pwm_12V += 8;
        } else if (pwm_12V < 255) {
            pwm_12V++;
        }
        LED_AUS(LED_12V);
      } else if ( ((pwm_12V < 50) && (diff < 0)) ||
                  (diffsumme_12V < -4) ) {
        diffsumme_12V = 0;
        // Spannung viel zu hoch?
        if (val_12V > AD_MAX_12V) {
          // Neu einregeln
          pwm_12V = 1;
        } else {
          // Nachregeln
          if ((diff < -16) && (pwm_12V > 7)) {
            pwm_12V += 8;
          } else if (pwm_12V > 0) {
            pwm_12V--;
          }
        }
        LED_AUS(LED_12V);
      } else {
        LED_EIN(LED_12V);
```

```
      }

    } else {
      // inaktive Regelung
      LED_AUS(LED_12V);
    }


    state = 0;
    ADMUX = ADMUX_BASE + 5; // Neuer Kanal: 48V
    break;

  default: // Unbekannter Zustand
    ADMUX = ADMUX_BASE + 7; // Neuer Kanal: Uin
    ADCSRA |= (1<<ADSC);
    state = 0;
}

switch (pwm_48V & 3) {
  case 0:
      OCR1D = (pwm_48V)>>2;
    break;
  case 1:
    if (count == 2) {
      OCR1D = (pwm_48V + 1)>>2;
    } else {
      OCR1D = (pwm_48V)>>2;
    }
    break;
  case 2:
    if (count&1) {
      OCR1D = (pwm_48V + 1)>>2;
    } else {
      OCR1D = (pwm_48V)>>2;
    }
    break;
  case 3:
    if (count == 2) {
      OCR1D = (pwm_48V)>>2;
    } else {
      OCR1D = (pwm_48V + 1)>>2;
    }
    break;
}
```

```
  switch (pwm_12V & 3) {
    case 0:
        OCR1B = (pwm_12V)>>2;
      break;
    case 1:
      if (count == 2) {
        OCR1B = (pwm_12V + 1)>>2;
      } else {
        OCR1B = (pwm_12V)>>2;
      }
      break;
    case 2:
      if (count&1) {
        OCR1B = (pwm_12V + 1)>>2;
      } else {
        OCR1B = (pwm_12V)>>2;
      }
      break;
    case 3:
      if (count == 2) {
        OCR1B = (pwm_12V)>>2;
      } else {
        OCR1B = (pwm_12V + 1)>>2;
      }
      break;
  }

  count++;
  count &= 3;

  LED_AUS(LED_STATUS);
}

ISR(INT0_vect) { // SPI Slave Select -- beide Flanken
  if (PINB&(1<<PB6)) { // Deselected
    USICR &= ~(1<<USIOIE);
    rxidx = 0;
    USIDR = FLAG_REG;
  } else {  // Selected
    USICR |= (1<<USIOIE);
    USISR &= ~(0x0f<<USICNT0);
    USISR |= (1<<USIOIF);
  }
```

```
}

ISR(USI_OVF_vect) { // SPI-Byte empfangen
  static uint8_t write_state = 255;

  rxbuf[rxidx] = USIDR;
  rxidx = (rxidx+1)&3;

  if (rxidx == 1) { // Befehlsbyte empfangen
    txbuf[0] = FLAG_REG;
    txbuf[1] = 0;
    txbuf[2] = 0;
    txbuf[3] = 0;
    switch (rxbuf[0]) {
      case 2:   // 12V schreiben
        write_state = 2;
        break;
      case 3:   // 12V lesen
        txbuf[1] = val_12V;
        txbuf[2] = pwm_12V;
        txbuf[3] = (TCCR1A&(1<<PWM1B))?1:0;
        break;
      case 4:   // 48V schreiben
        write_state = 4;
        break;
      case 5:   // 48V lesen
        txbuf[1] = val_48V;
        txbuf[2] = pwm_48V;
        txbuf[3] = (TCCR1C&(1<<PWM1D))?1:0;
        break;
      case 7:   // Uin lesen
        txbuf[1] = val_Uin;
        break;
    }
  }
  USIDR = txbuf[rxidx];

  if (rxidx == 0) { // 4. Byte empfangen
    USIDR = FLAG_REG;
    switch (write_state) {

      case 2:
        if (rxbuf[2]) {
          if (rxbuf[3]) {
```

```
            FLAG_REG |= (1<<AUTO_12V);
            pwm_12V = 0;
          } else {
            FLAG_REG &= ~(1<<AUTO_12V);
            pwm_12V = rxbuf[1];
          }
          OCR1B = pwm_12V;
          TCCR1A |= (1<<PWM1B); // 12V  PWM
          FLAG_REG |= (1<<PWM_ACT_12V);
        } else {
          TCCR1A &= ~(1<<PWM1B);  // 12V  PWM
          FLAG_REG &= ~(1<<PWM_ACT_12V);
        }
        write_state = 255;
        break;
      case 4:
        if (rxbuf[2]) {
          if (rxbuf[3]) {
            FLAG_REG |= (1<<AUTO_48V);
            pwm_48V = 0;
          } else {
            FLAG_REG &= ~(1<<AUTO_48V);
            pwm_48V = rxbuf[1];
          }
          OCR1D = pwm_48V;
          TCCR1C |= (1<<PWM1D); // 48V  PWM
          FLAG_REG |= (1<<PWM_ACT_48V);
          PORTA &= ~(1<<PA3); // Lastwiderstand abschalten
        } else {
          TCCR1C &= ~(1<<PWM1D);  // 48V  PWM
          FLAG_REG &= ~(1<<PWM_ACT_48V);
        }
        write_state = 255;
        break;

      case 255:
        break;
      default:
        write_state = 255;
    }
  }
  USISR |= (1<<USIOIF);
}
```

```
// ISR für die Startverzögerung
ISR(TIMER0_OVF_vect) {
  static uint8_t count = 7;

  PORTA = count;
  count--;
  if (count == 255) {
    FLAG_REG &= ~(1<<START_WAIT);
  }
}
```

D Berechnung der Ableitung von Seite 35

$$
\begin{aligned}
f'(R_{DS}) &= g'(R_{DS}) \cdot h(R_{DS}) + g(R_{DS}) \cdot h'(R_{DS}) \\
&= 1 \cdot \left(\frac{U}{R_{DS}+R_S}\right)^2 + R_{DS} \cdot \left(\left(\frac{U}{R_{DS}+R_S}\right)^2\right)' \\
&= \left(\frac{U}{R_{DS}+R_S}\right)^2 + R_{DS} \cdot U^2 \left(\left(\frac{1}{R_{DS}+R_S}\right)^2\right)' \\
&= \left(\frac{U}{R_{DS}+R_S}\right)^2 + R_{DS} \cdot U^2 \left(\frac{1}{(R_{DS}+R_S)^2}\right)' \\
&= \left(\frac{U}{R_{DS}+R_S}\right)^2 + R_{DS} \cdot U^2 \left(\frac{-2}{(R_{DS}+R_S)^3}\right) \\
&= \left(\frac{U}{R_{DS}+R_S}\right)^2 + \left(\frac{-2 \cdot R_{DS} \cdot U^2}{(R_{DS}+R_S)^3}\right) \\
&= \frac{U^2 \cdot (R_{DS}+R_S)}{(R_{DS}+R_S)^3} + \left(\frac{-2 \cdot R_{DS} \cdot U^2}{(R_{DS}+R_S)^3}\right) \\
&= \frac{U^2 \cdot (R_{DS}+R_S)}{(R_{DS}+R_S)^3} - \left(\frac{2 \cdot R_{DS} \cdot U^2}{(R_{DS}+R_S)^3}\right) \\
&= \frac{U^2 \cdot (R_{DS}+R_S) - 2 \cdot U^2 \cdot R_{DS}}{(R_{DS}+R_S)^3} \\
&= \frac{U^2 \cdot R_{DS} + U^2 \cdot R_S - 2 \cdot U^2 \cdot R_{DS}}{(R_{DS}+R_S)^3} \\
&= \frac{U^2 \cdot R_S - U^2 \cdot R_{DS}}{(R_{DS}+R_S)^3} \\
&= \frac{U^2 \cdot (R_S - R_{DS})}{(R_{DS}+R_S)^3}
\end{aligned}
$$

E Verwendete Software

- Simulation
  - LTspice/SwitcherCAD III
    `http://www.linear.com/designtools/software/`
- Schaltungs- und Platinenentwurf
  - Target 3001!
    `http://www.ibfriedrich.com/home.htm`
- Softwareentwicklung
  - WinAVR (AVR-GCC)
    `http://winavr.sourceforge.net/`
  - AVR-Studio
    `http://atmel.com/dyn/products/tools_card.asp?tool_id=2725`
- Sonstige
  - proTEXt 2.0
    `http://www.tug.org/protext/`

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §22(4) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 29.02.2008	
Ort, Datum	Unterschrift