TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

ATOMINSTITUT

DIPLOMARBEIT

Aufbau, Test und Automatisierung einer Kontrollelektronik für magnetische Wanderwellenresonatoren zur Manipulation polarisierter Neutronenstrahlen

ausgeführt unter der Leitung von

Em.Univ.-Prof. Dipl.-Ing. Dr.techn. Gerald Badurek
Ass.Prof. Dipl.-Ing. Dr.techn. Erwin Jericha
Atominstitut, e141

eingereicht an der Technischen Universität Wien
Fakultät für Physik

von

Patrick Reisinger, BSc
Matrikelnummer 01607394
patrick.reisinger@gmx.at

Wien, am 20. Februar 2023

(Unterschrift Verfasser) (Unterschrift Betreuer)

Kurzfassung

Polarisierte Neutronenstrahlen können durch spezielle räumliche Magnetfeldverteilungen nach ihren spektralen Eigenschaften manipuliert und in ihrer zeitlichen Struktur beeinflusst werden. Auf dieser Tatsache beruht das Konzept von magnetischen Spinresonatoren für Neutronen, in denen für eine Resonanzwellenlänge eine Inversion der Neutronenpolarisation erreicht werden kann. Durch die Verwendung von individuell steuerbaren Resonatorelementen kann dabei eine größtmögliche Flexibilität in der Formung von Neutronenstrahlen erreicht werden. Eine aktuelle Entwicklung ist eng in das Neutronenzerfallsprojekt PERC integriert, das sich zur Zeit an der Forschungsneutronenquelle FRM II, in Garching bei München, in Entwicklung befindet.

Das Kernstück des Neutronenresonators ist eine eigens am Atominstitut entwickelte, schnell schaltbare Hochstromquelle, die für jedes Resonatorelement den erforderlichen Strom für die Erzeugung des Magnetfeldes im erforderlichen Zeitraum, der von Mikrosekunden bis zu Tagen dauern kann, zur Verfügung stellt. In dieser Diplomarbeit wurden solche Stromquellen in ein Gesamtsystem für bis zu 48 Resonatorelemente integriert. Dies beinhaltete, neben der Beseitigung bestehender Hardwareprobleme, die Entwicklung einer Software zur individuellen Kommunikation mit jeder Stromquelle, wodurch der Resonator automatisiert und flexibel kontrolliert werden kann.

Mittelfristiges Ziel, über diese Arbeit hinaus, sind Experimente mit dem Resonator an einem weißen thermischen Neutronenstrahl am TRIGA-Reaktor des Atominstituts der TU Wien in seinen verschiedenen Betriebsarten, wie stationärem Betrieb oder Wanderwellenmodus. Die experimentellen Parameter werden sich dabei an den Charakteristika thermischer Neutronenstrahlen sowie an den speziellen Anforderungen des PERC-Projekts mit kalten Neutronenstrahlen orientieren.

Abstract

Polarized neutron beams can be manipulated and influenced in their temporal structure based on their spectral properties using spatial magnetic field distributions. This concept is the basis for magnetic spin resonators for neutrons, in which an inversion of the neutron polarization can be achieved for a certain resonance wavelength. The use of individually controllable resonator elements allows for maximum flexibility in shaping the neutron beams. A current development is closely integrated into the neutron decay project PERC, which is currently under development at the Research Neutron Source FRM II in Garching near Munich.

The core of the neutron resonator is a high-current source developed specifically at the Atominstitut of the TU Wien, which can quickly switch and provide the necessary current for generating the magnetic fields. The achievable time frame of the source can vary between microseconds up to days for each resonator element. In this thesis, such current sources were integrated into a system for up to 48 resonator elements. This included resolving existing hardware problems as well as developing a software for individual communication with each current source, allowing the resonator to be controlled flexibly.

The medium-term goal beyond this work is to conduct experiments with the resonator on a white thermal neutron beam at the TRIGA reactor of the Atominstitut in its various operating modes, such as stationary operation or traveling wave mode. The experimental parameters will be based on the characteristics of thermal neutron beams and the specific requirements of the PERC project with cold neutron beams.

Inhaltsverzeichnis

1 Einleitung

Diese Arbeit beschäftigt sich mit der Steuerungselektronik des MONOPOLs, ein magnetischer Badurek-Wanderwellenresonator zur Erzeugung monochromatischer Neutronenstrahlen. Strahlen dieser Art kommen in vielen physikalischen Experimenten zum Einsatz, zum Beispiel dem PERC-Experiment (Proton Electron Radiation Channel) [1–3].Der Wanderwellenresonator bietet im Vergleich zu herkömmlichen Methoden viele Vorteile. So kann die Wellenlängenselektion im Vergleich zu Choppern und kristallinen Monochromatoren, welche kurz vorgestellt werden, ohne jegliche mechanischen Eingriffe rein durch Ändern von Magnetfeldern vorgenommen werden.

Ein Chopper besteht aus einer rotierenden Scheibe, welche aus einem Neutronenabsorber gefertigt ist und eine oder mehrere Öffnungen aufweist. Diese Öffnungen führen dazu, dass ein auf den Chopper gerichteter Neutronenstrahl durch periodisches Unterbrechen in Pakete unterteilt wird. Ein inhärenter Nachteil von Choppern ist der Neutronenverlust, da die meiste Zeit der absorbierende Teil der Scheibe im Strahl steht.
Die Wellenlängenselektion mittels Chopper kann auf zwei verschiedene Arten realisiert werden. Einerseits kann man statt der Scheibe eine rotierende Walze verwenden. Die zuvor beschriebenen Öffnungen sind nun ein in die Walze eingebrachter Kanal, welcher im Ruhesystem der Neutronen mit gewünschter Geschwindigkeit (Selektion durch Rotationsfrequenz) ein passierbarer Tunnel ist. Andererseits können zwei drehzahlidente Scheiben-Chopper mit einem definierten Abstand hintereinander platziert werden, wodurch ebenfalls nach Geschwindigkeit gefiltert werden kann [4, Abschnitt 13.3].
Kristalline Monochromatore nutzen die Bragg-Bedingung aus, um eine Wellenlängenselektion zu erreichen. Will man die Wellenlänge ändern muss jedoch der gesamte Aufbau gedreht und gegebenenfalls der verwendete Kristall getauscht werden [5, Abschnitt 11.3].

Erstmals wurde die Wellenlängenselektion mittels magnetischer Felder von Drabkin [6] im Jahre 1963 vorgeschlagen und von Badurek und Jericha [7] in das im MONOPOL zur Anwendung kommende Konzept ausgereift.
Der MONOPOL-Resonator bietet drei unterschiedliche Betriebsmodi, welche individuell konfiguriert werden können. Dabei kann neben der Wellenlänge auch die Pulsdauer angepasst werden. Ihr unteres Limit ist durch die Schaltzeiten der Resonatorspulen begrenzt.

Die vorliegende Diplomarbeit beschäftigt sich mit der softwareseitigen Implementierung der drei Betriebsmodi, welche durch die Neutronengeschwindigkeiten strenge zeitkritische Anforderungen erfüllen muss. Zusätzlich zur physikalischen Funktionalität wird ein Interface zur Verfügung gestellt, um mit der Resonator-Ablaufsteuerung über einen PC zu kommunizieren. Neben der Softwareerstellung wurden diverse Hardwareprobleme detektiert und weitestgehend auch behoben. Details dazu sind in Kapitel 6 ausgeführt.

1.1 Aufbau- und Funktionsbeschreibung

Der Badurek-Wanderwellenresonator nutzt zur Wellenlängenselektion das anomale magnetische Moment der Neutronen. Der prinzipielle Aufbau ist in Abb. 1.1 zu sehen.
Ein unpolarisierter polychromatischer Neutronenstrahl wird zunächst durch einen Superspiegel polarisiert. Anschließend durchqueren die Neutronen den Resonator, bestehend aus 48 Aluminium-Spulen. In diesem Resonator herrscht einerseits das Führungs-Magnetfeld B_0, auch Selektorfeld genannt, andererseits erzeugen die Spulen ein Magnetfeld $B_1(x)$. Beide Felder stehen üblicherweise normal aufeinander und es gilt $B_0 >> B_1$. Die Neutronen führen aufgrund des resultierenden Magnetfelds, welches wie B_0 und B_1 transversal zur Strahlrichtung steht (Abb. 1.1), eine Larmorpräzession durch. Wird $B_1(x)$ so gewählt, dass die Frequenz des Magnetfelds im Ruhesystem der Neutronen mit der Larmorfrequenz übereinstimmt und die Amplitudenbedingung

$$\frac{B_1}{B_0} \cdot \frac{L}{2a} = (2k+1)\frac{\pi}{4}, \qquad k = 0, 1, 2, ... \tag{1.1}$$

erfüllt ist ($2k+1$ ... Anzahl der Spinflips), kann die Spinflipwahrscheinlichkeit

$$W(\lambda) = \frac{\xi^2}{(\Delta\lambda/\lambda)^2 + \xi^2} \cdot \sin^2\left[\frac{\pi L\lambda}{2a\lambda_0}\sqrt{(\Delta\lambda/\lambda)^2 + \xi^2}\right], \tag{1.2}$$

mit

$$\xi = \frac{2B_1}{\pi B_0}, \tag{1.3}$$

$$\Delta\lambda = \lambda - \lambda_0. \tag{1.4}$$

angegeben werden (L ... Länge, $2a$ ... Periode des Resonators) [8].

$$\lambda_0 = \frac{\pi h}{am|\gamma|B_0} \tag{1.5}$$

ist dabei die Resonanzwellenlänge, welche von Naturkonstanten (h ... Plank´sches Wirkungsquantum, m ... Neutronenmasse, γ ... gyromagnetisches Verhältnis), dem Selektorfeld B_0 und via a von der Geometrie des Resonators abhängt.

Die maximale Wellenlängenauflösung erhält man, wenn $k = 0$ gilt und die Anzahl der Resonatorperioden $N = \frac{L}{2a}$ maximiert wird

$$\frac{\Delta\lambda_{1/2}}{\lambda_0} \simeq \frac{0.8}{N} \tag{1.6}$$

mit der Halbwertsbreite (FWHM) $\Delta\lambda_{1/2}$ des globalen Maximums der Spinflipwahrscheinlichkeit Gl. (1.2).
Nach Durchquerung des Resonators werden die Neutronenspins durch einen breitbandigen Stromblatt-Flipper umgekehrt. Ein darauffolgender Analyzer lässt nur jene Neutronen passieren, welche zuvor im Resonator einen Spinflip erfahren haben, also jene Neutronen im Bereich der Resonanzwellenlänge λ_0 [9].

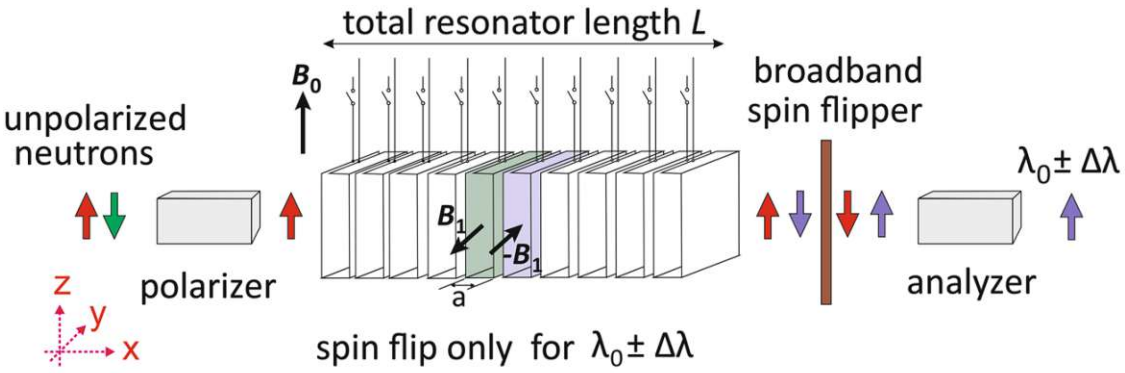

Abbildung 1.1: Schematische Darstellung eines Badurek-Wanderwellenresonators (Bild aus [9])

Betriebsmodi

Die drei unterschiedlichen Betriebsmodi des Wanderwellenresonators sind der Static Mode (SM), der Conventional Mode (CM) und der Traveling Wave Mode (TWM). Im Vergleich zum SM können die beiden anderen Modi zusätzlich zur Wellenlängenselektion auch zur Pulsgenerierung verwendet werden. Dies wird durch gezieltes Ein- und Ausschalten der Resonatorspulen erreicht. Der Unterschied zwischen CM und TWM besteht darin, dass im TWM alle 48 im MONOPOL vorkommenden Spulen individuell geschaltet werden können. Die Neutronen werden quasi von den Magnetfeldern durch den Resonator „begleitet“. Diese technische Erweiterung erlaubt dem Badurek-Resonator im optimalen Fall die Reduzierung der minimalen Puls-Anstiegs- und -Fallzeit von $\Delta t_{CM,min} = L/v_0$ auf $\Delta t_{TWM,min} = a/v_0$ (veranschaulicht in Abb. 1.2), also in diesem Fall $\frac{1}{48}$-stel eines herkömmlichen Drabkin-Resonators.

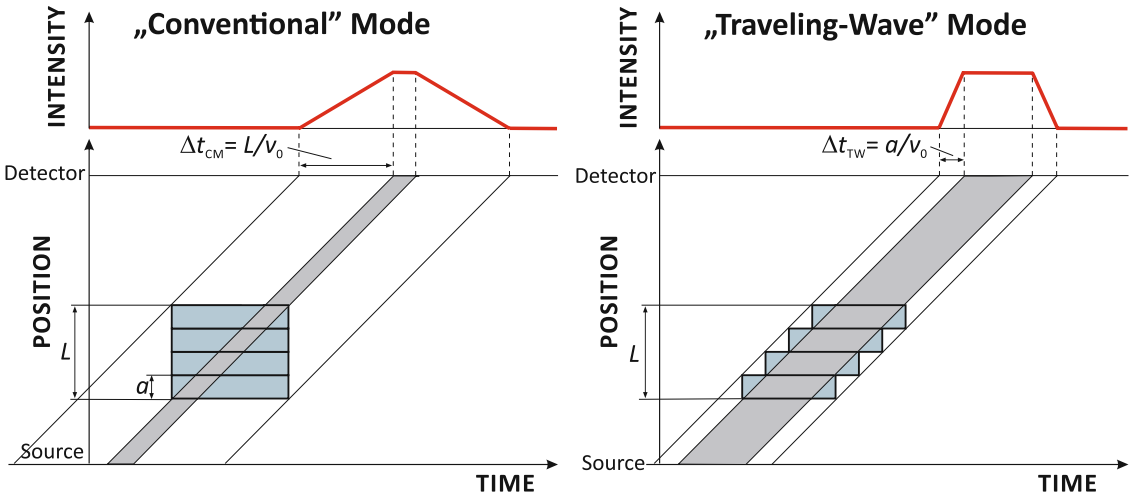

Abbildung 1.2: Vergleich des Zeitverhaltens von CM und TWM (Bild aus [9])

Ansteuerungskonzept

Die elektronische Steuerung hat zwei Hauptaufgaben zu erfüllen. Einerseits muss die Stromstärke der einzelnen Resonatorspulen und damit die Amplitude der Magnetfelder eingestellt werden. Andererseits müssen die Ströme gezielt ein- und ausgeschaltet werden.
Um diese Aufgaben zu erfüllen, wurde in [10] ein Konzept entwickelt, welches hier kurz beschrieben wird. Der Benutzer stellt über eine GUI die gewünschte Wellenlänge, die Pulsdauer sowie weitere Parameter ein. Die PC-Software ermittelt daraus die Betriebsparameter der Ablaufsteuerung und übermittelt diese an einen Mikrocontroller via UART (Abschnitt 2.1.2). Der Controller konfiguriert daraufhin die Stromquellen mit den gewünschten Stromstärken über DACs.
Nun kann der Einschaltvorgang begonnen werden. Um Spannungsspitzen zu vermeiden, wurde das Konzept einer Artificial- und Real-Last eingeführt, welche beide in etwa den gleichen ohmschen Widerstand aufweisen und pro Stromquelle ausgeführt sind. Dabei wird zuerst der Strom über die Artificial-Last geführt bis sich Stabilität einstellt. Anschließend wird der Strom über die Real-Last geleitet, wobei kurzzeitig beide Lasten aktiv sind (=Glitch). Der Ausschaltvorgang erfolgt analog in umgekehrter Reihenfolge. Die Schaltvorgänge werden über Schieberegister so definiert, dass sich das vom Benutzer gewünschte Zeitverhalten einstellt und der Neutronenstrahl dadurch monochromatisiert und gegebenenfalls gepulst wird.

2 Grundlagen

In diesem Kapitel werden die notwendigen Grundlagen der MONOPOL-Elektronik dargelegt. Dies beinhaltet sowohl technische Details wie Übertragungsprotokolle, als auch relevante physikalische und messtechnische Grundlagen und Konzepte.

2.1 Übertragungsprotokolle

2.1.1 I²C-Bus

Der I²C-Bus (Inter-Integrated Circuit) [11] wurde von Philips entwickelt und ist ein serieller Bus. Er basiert auf dem Master-Slave Prinzip und kann von einem oder mehreren Mastern gesteuert werden.
Jedem Gerät im Busnetzwerk, beziehungsweise Slave in Singlemaster-Systemen, muss eine eindeutige Adresse zugeordnet sein. Sie kann entweder aus 8 bit (7 bit + R/$\overline{\text{W}}$) oder 10 bit bestehen. Meist setzt sich die Adresse aus einem vom Chiphersteller und einem über Pins konfigurierbaren Teil zusammen. So wird sichergestellt, dass mehrere baugleiche ICs (=Integrated Circuit) in einem Netzwerk adressiert werden können.
Die theoretische Obergrenze für die Anzahl der angeschlossenen Bausteine ergibt sich durch die limitierte Anzahl an Adressen, in der Praxis muss jedoch auf die kapazitive Last geachtet werden. Spezielle I²C-Erweiterungen (Repeater, Hubs, Multiplexer und Switches) können das Netzwerk in mehrere Subnetze unterteilen und gezielt ansprechen. Dies verhindert einerseits die kapazitive Überbelastung, andererseits können so auch Adresskonflikte aufgelöst werden. Die Busteilnehmer merken von dieser Unterteilung nichts, die Adressierung erfolgt wie gehabt.
Der I²C-Bus kann je nach angeschlossener Peripherie mit unterschiedlichen Übertragungsraten betrieben werden. Eine Zusammenfassung ist in Tab. 2.1 zu finden, wobei die jeweiligen Obergrenzen angegeben sind. Sollte die Taktfrequenz zu hoch für die adressierten Bausteine sein, können diese den Takt verlangsamen und dadurch eine korrekte Übertragung sicherstellen.

Bezeichnung	*Übertragungsrate*
Standard Mode	$< 100\,\text{kbit/s}$
Fast Mode	$< 400\,\text{kbit/s}$
Fast Mode Plus	$< 1\,\text{Mbit/s}$
High Speed Mode	$< 3.4\,\text{Mbit/s}$
Ultra Fast Mode[1]	$< 5\,\text{Mbit/s}$

[1] nur unidirektional verwendbar

Tabelle 2.1: I²C-Bus Übertragungsmodi [11]

Aufbau und Übertragungsablauf

Der physikalische Aufbau des Bussystems besteht aus zwei Leitungen, der Taktleitung SCL und der Datenleitung SDA. Alle Teilnehmer, sowohl alle Master als auch alle Slaves müssen an diese angeschlossen werden (Abb. 2.1).

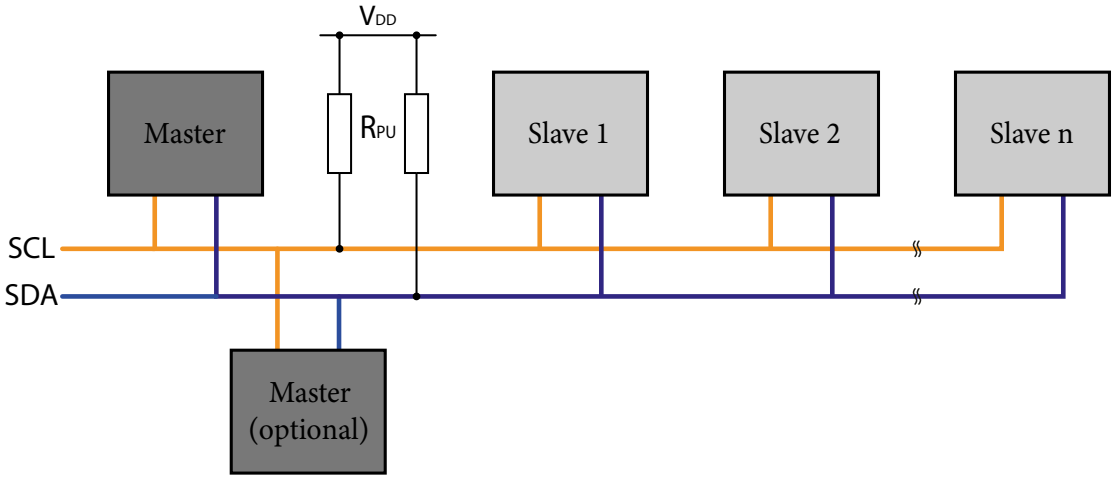

Abbildung 2.1: Schematische Darstellung eines I²C-Bussystems mit R_{PU} ... Pull-Up Widerstände, V_{DD} ... Versorgungsspannung

In Abb. 2.2 ist eine Veranschaulichung des I²C-Übertragungsprotokolls zu sehen. Findet keine Übertragung statt ist der Bus frei und befindet sich im *Idle*-Zustand (SCL = SDA = High). Dieser besteht bis ein Master den Beginn einer Übertragung durch das Einleiten einer Start-Bedingung signalisiert.
Die Start-Bedingung besteht aus einem High-Low-Übergang der Datenleitung SDA während die Taktleitung SCL auf High bleibt. Nun legt der Master das Adressbyte auf den Bus, wobei das Least Significant Bit (LSB) angibt, ob der Master Daten empfangen ($R/\overline{W}$ = Low) oder senden ($R/\overline{W}$ = High) will. Das Adressbyte wird von allen Teilnehmern mit ihrer individuellen Adresse abgeglichen und der Adressat bestätigt im 9. Taktzyklus, indem er die Datenleitung SDA auf Low zieht

(= ACK ... Acknowledge).
Jetzt kann der eigentliche Datenaustausch stattfinden. Der Master gibt den Takt auf SCL vor, der Slave kann den Low-Pegel des Takts verlängern und dadurch die Übertragungsrate anpassen [12, Abschnitt 8.6]. Auf der Datenleitung überträgt je nach zuvor adressiertem Modus entweder der Master oder der Slave. Der Empfänger bestätigt nach Erhalt des Bytes mit einem ACK. Byteweises Senden inklusive anschließender Bestätigung wiederholt sich nun, bis die gesamte Nachricht übermittelt wurde.
Das Ende der laufenden Kommunikation wird durch die Stop-Bedingung eingeleitet, welche der Start-Bedingung gleicht und den Bus wieder frei gibt. Die Start/Stop-Bedingungen zeichnen sich insofern aus, dass hier eine Änderung des Datensignals auftritt, während die Taktleitung auf High ist. Bis auf diese Ausnahmen darf der Pegel auf SDA nur im Low-Zustand von SCL geändert werden[13, Abschnitt 15.3.2].

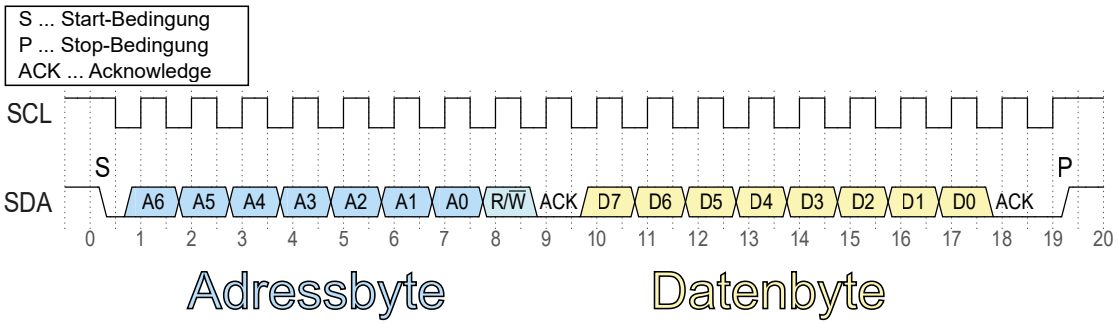

Abbildung 2.2: Datenübertragung mittels I²C-Bus

2.1.2 UART

UART steht für Universal Asynchronous Receiver / Transmitter und stellt eine Schnittstelle zum Peer-to-Peer Datenaustausch dar. Die Übermittlung erfolgt dem Namen nach asynchron, es gibt also kein gemeinsames Taktsignal. Stattdessen wird auf die Baudrate zurückgegriffen, um die einzelnen Bits zu differenzieren. Sie muss für eine korrekte Übertragung bei Sender und Empfänger übereinstimmen und kann entweder fest eingestellt oder aber aus der Nachricht ermittelt werden. Weiters bietet UART die Möglichkeit unterschiedliche Frames (= Übertragungsrahmen) zu verwenden. So kann die Anzahl der Daten- (5-9 bit) und Stopbits (1 oder 2) sowie der Paritätsbitmodus (gleich oder ungleich) eingestellt werden.

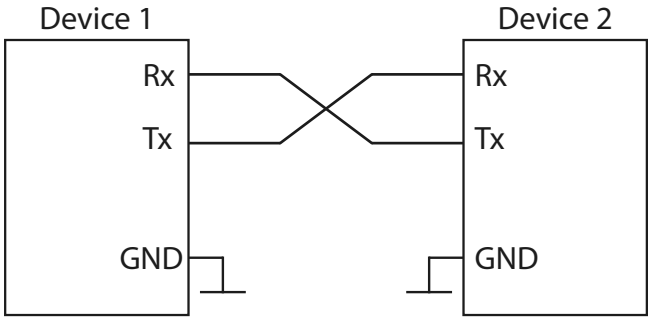

Abbildung 2.3: Schematische Darstellung der UART-Beschaltung

Der Aufbau eines UART-Systems besteht im Normalfall aus zwei Datenleitungen (Tx und Rx, Abb. 2.3). Zusätzlich wird oftmals auf zwei weitere Leitungen zwecks Handshake-Verfahren zurückgegriffen (RTS ... Request to Send und CTS ... Clear to Send). Die UART-Schnittstelle selbst trifft aber noch keinerlei Aussagen über die physikalische Schicht und unterstützt so die unterschiedlichsten Protokolle. Dabei können sowohl vollduplex (= gleichzeitiges Senden und Empfangen, Beispiel: *RS232*) als auch halbduplex (= kein gleichzeitiges Senden und Empfangen, Beispiel: *RS485*) Protokolle gewählt werden [14, Abschnitt 7.1].

2.2 Signalverarbeitung und -übertragung

2.2.1 A/D- und D/A-Wandler

Analog-Digital Konverter (ADC) wandeln ein kontinuierliches analoges in ein diskretes digitales Signal um. Dabei werden abhängig von der Bandbreite (je größer, desto genauer) bestimmten Spannungspegelbereichen Zahlenwerte zugewiesen. Durch die limitierte Auflösung entsteht bei der Konversion der sogenannte Quantisierungsfehler.

Digital-Analog Konverter (DAC) sind das Gegenstück zum ADC und machen die Quantisierungsprozedur rückgängig. Sie erzeugen aus dem digitalen Zahlenwert eine analoge physikalische Größe. Meist werden Spannungen erzeugt, welche wiederum über einen Aktor in die gewünschte physikalische Größe umgewandelt werden können.

Beispiel: 12-Bit ADC (0 V bis 12 V) mit analoger Eingangsspannung von 3.3 V

Analoger Wertebereich:	0 V bis 12 V
Digitaler Wertebereich (12-Bit):	0 bis $2^{12} - 1$
Volt pro Stufe:	$x = \frac{12}{2^{12}} = 2.93\,\frac{\text{mV}}{\text{bit}}$: $\rightarrow$ max. Quantisierungsfehler von ± 2.93 mV
Analoge Eingangsspannung:	3.3 V
Digitalisierter Wert:	$\frac{3.3\,\text{V}}{x} = 1126.40 \approx 1126 = z$
Fehler der analogen Ausgangsspannung:	$3.3\,\text{V} - z \cdot x = 1.17\,\text{mV}$

Heutzutage werden Datenwandler (ADC und DAC) meist seriell an die verarbeitende Elektronik gekoppelt [13, Abschnitt 9.2]. Beim MONOPOL-Aufbau findet die Kommunikation zwischen ADC/DAC und Mikrocontroller über den I^2C-Bus statt.

2.2.2 Hall-Sensor

Der Hall-Sensor ist ein mit Strom durchflossenes Plättchen, welches den Hall-Effekt zur Messung des magnetischen Feldes nutzt.
Durch das Plättchen fließt der Strom I_1, der durch eine Stromquelle eingespeist wird (Abb. 2.4). Auf die Elektronen wirkt bei äußeren Feldern die Lorentzkraft

$$\vec{F_L} = q \cdot (\vec{E} + \vec{v} \times \vec{B}). \tag{2.1}$$

Liegt kein elektrisches Feld vor ($\vec{E} = 0$) verbleibt lediglich der magnetische Anteil. Dieser bewirkt eine Kraft auf die Elektronen, wodurch die direkt zum gemessenen Feld proportionale Hall-Spannung

$$U_H = \frac{R_H}{d} \cdot B \cdot I_1. \tag{2.2}$$

entsteht. Bei bekannten Hallkoeffizient R_H und Dicke der Hall-Platte d lässt sich so der Betrag des magnetischen Feldes ermitteln [15, Abschnitt 3.3.5].

In der vorliegenden Arbeit wird je ein Hall-Sensor pro Resonatorelement zwecks Strommessung verbaut. Genauere Details der realisierten Stromermittlung werden in den Abschnitten 3.4.4 und 6.2.1 beschrieben.

2.2.3 Vierleitermessmethode

Die Vierleitermessmethode entspricht näherungsweise der idealen spannungsrichtigen Messschaltung (Abb. 2.5.a) und kann zur Widerstands-, sowie Leistungsermittlung herangezogen werden. Der große Vorteil dieser Art von Messschaltung im Vergleich zur Zweileitermessmethode ist die hohe Genauigkeit, da der systematische Fehler durch Leitungs- und Kontaktierungswiderstände praktisch keinerlei

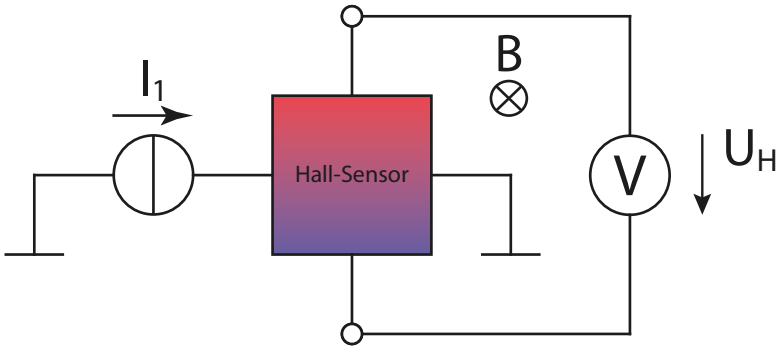

Abbildung 2.4: Funktionsprinzip eines Hall-Sensors mit I_1 ... Sensorstrom, B ... magnetisches Feld in die Zeichenebene, U_H ... Hall-Spannung

Einfluss auf die Messung hat [16]. Erreicht wird dies, indem für Stromquelle und Voltmeter jeweils separate Leitungen zum zu messenden Verbraucher geführt werden. Dadurch ist es möglich die Spannung direkt am Widerstand abzugreifen und etwaige Spannungsabfälle der Strom-Zuleitung auszublenden.

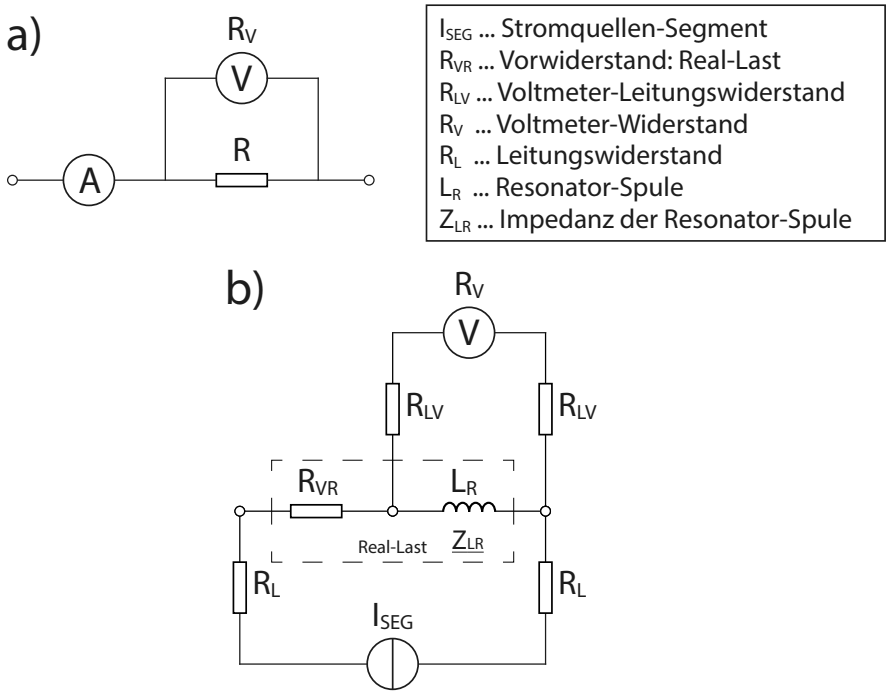

Abbildung 2.5: Zum Vergleich: **a** Ideale spannungsrichtige Messschaltung; **b** Vierleitermessmethode im MONOPOL

Die Kabelwiderstände der Spannungsmessung sind prinzipiell von geringer Bedeutung. Der vergleichsweise große Innenwiderstand des Voltmeters ($R_{LV} << R_V$)

bewirkt, dass der Messstrom durch das Spannungsmessgerät und die Messleitungen vernachlässigbar gering ist:

$$R_P = (2R_{LV} + R_V) \parallel \underline{Z_{LR}} \approx \frac{R_V \cdot Z_{LR}}{R_V + Z_{LR}} \approx Z_{LR} \tag{2.3}$$

$$U_{LR} = I_{SEG} \cdot R_P \approx I_{SEG} \cdot Z_{LR} \tag{2.4}$$

Nach Einstellen der Stromquelle und Ablesen der resultierenden Spannung muss man nur, je nach zu bestimmender Größe, eine entsprechende Gleichung anwenden. Im Falle einer Leistungsermittlung kommt folgende Gleichung zur Anwendung:

$$P = U \cdot I. \tag{2.5}$$

Soll der Widerstandswert errechnet werden, so greift man auf das Ohmsche Gesetz zurück:

$$R = \frac{U}{I}. \tag{2.6}$$

Letzteres wird im MONOPOL verwendet (siehe Abschnitt 3.4.4), um den ordnungsgemäßen Zustand der einzelnen Resonatorspulen sicherzustellen. Sollte eine Spule beschädigt sein, so steigt ihr Widerstand Z_{LR} (Abb. 2.5.b).

2.2.4 Koaxialkabel

Koaxialkabel (Abb. 2.6) werden in der Mess- und Übertragungstechnik verwendet, um Signale möglichst störungsfrei zu übermitteln. Im vorliegenden Projekt wird in Abschnitt 3.5.3 ein Aufbau zur Stromversorgung, der auf dem Koaxialkabel-Prinzip beruht, vorgestellt. Dieser soll mögliche Einflüsse auf den Neutronenstrahl minimieren.

Koaxialkabel bestehen aus einem Innenleiter mit Radius R_1, welcher von einem konzentrischen Außenleiter (R_3) durch eine Isolation getrennt ist. Dieser Aufbau erzeugt keinerlei elektrische und magnetische Felder außerhalb des Kabels, falls beide Leiter gegengleiche Ströme führen.
Für das elektrische Feld ist dies offensichtlich, da die Feldlinien alle innerhalb des Kabels verlaufen. Die Feldfreiheit für das magnetische Feld kann ausgehend vom Ampèreschen Gesetz gezeigt werden

$$\oint_S \vec{B} \cdot d\vec{s} = \mu_0 I, \tag{2.7}$$

wobei S eine Schleife um den stromführenden Leiter ist. Im dargestellten Fall bietet sich ein Kreis an, dessen Fläche normal auf die Stromrichtung und somit auf die Feldlinien des entstehenden B-Feldes steht.
Betrachtet man zuerst nur das Feld des Innenleiters mit Strom I_I, so ergibt sich nach Integration von Gl. (2.7)

$$2\pi r B = \mu_0 I_I, \tag{2.8}$$

$$\rightarrow B = \frac{\mu_0 I_I}{2\pi r}. \tag{2.9}$$

Für das Feld des Außenleiters mit Strom I_A erhält man analog

$$B = \frac{\mu_0 I_A}{2\pi r}. \tag{2.10}$$

Durch Superposition beider Ergebnisse erhält man das magnetische Feld außerhalb eines Koaxialkabels

$$B = \frac{\mu_0 (I_I + I_A)}{2\pi r}, \tag{2.11}$$

wobei für $I_I = -I_A$ das B-Feld 0 wird.

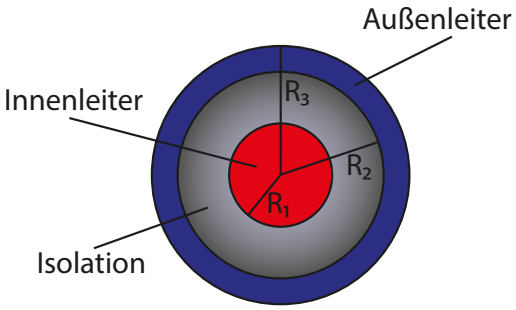

Abbildung 2.6: Querschnitt eines Koaxialkabels

Die Kapazität C und die Induktion L des Koaxialkabels hängen neben den Kabeldimensionen (Länge l) vom Dielektrikum (ε_r) und der Permeabilität (μ_r) ab

$$C = 0,2 \cdot \mu_r \cdot l \cdot \ln \frac{R_3}{R_1}, \tag{2.12}$$

$$L = \frac{55,3 \cdot \varepsilon_r \cdot l}{\ln \frac{R_3}{R_1}}. \tag{2.13}$$

Der Wellenwiderstand Z_L, welcher auch als Abschlusswiderstand zur Vermeidung von Reflexionen verwendet wird, kann durch

$$Z_L = \frac{60}{\sqrt{\varepsilon_r}} \cdot \ln \frac{R_3}{R_1} \tag{2.14}$$

berechnet werden. In der Praxis übliche Werte des Wellenwiderstands von Koaxialkabeln sind 50 Ω beziehungsweise 75 Ω, je nach Anwendungsfall können aber auch andere Werte verwendet werden [17].

2.3 Elektronische Schaltwerke

2.3.1 Schieberegister (SR)

Ein Schieberegister ist ein taktgesteuerter Baustein, der aus mehreren Speicherzellen (D-Flip-Flops) besteht. Es besitzt je nach Ausführung zwei Eingänge, einen Daten- (D_{IN}) und einen Takt-Eingang (CLK). Die Taktung erfolgt meist über die positive Taktflanke.
In Abb. 2.7.a ist der schematische Aufbau eines Schieberegisters zu sehen. Erfolgt eine positive Taktflanke auf der CLK-Leitung wird der logische Pegel des Eingangs D_{IN} im D-Flip-Flop der Zelle S0 (2.7.b) gespeichert. Mit jedem weiteren Takt wird der Wert von jeder Zelle in die nächste weitergereicht (S0→S1, S1→S2, etc.). Der Wert, der in der Zelle S7 gespeichert war, wird aus dem Register geschoben. Je nach Realisierung des Registers können alle Zellen parallel ausgelesen werden.
Schieberegister können anwendungsspezifisch mehrere verschiedene Aufgaben erfüllen. Sie können einerseits als FIFO-Speicher (First-In-First-Out), andererseits zur Parallelisierung serieller Daten verwendet werden. Im Wanderwellenresonator wird letztere Funktion angewendet. [18, Abschnitt 8.5]

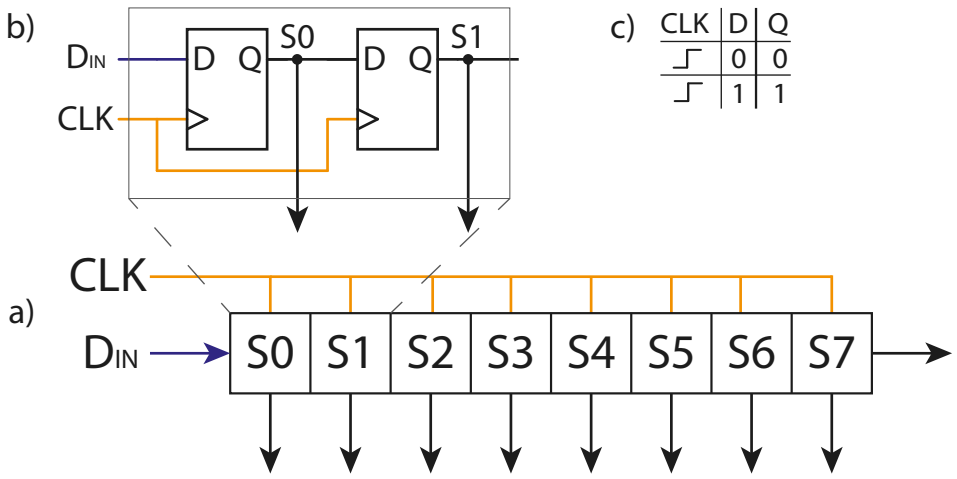

Abbildung 2.7: Schematische Darstellung eines 8 bit-Schieberegisters. **a** Darstellung mit Speicherzellen; **b** Detailansicht der Zellen; **c** Wahrheitstabelle eines D-Flip-Flops

3 Hardware

In diesem Kapitel werden alle beteiligten Hardware-Komponenten, sowie die Stromversorgung beschrieben. In Abb. 3.1.a ist eine schematische Darstellung des Aufbaus zu sehen. Auf die einzelnen Komponenten und Subkomponenten wird in den entsprechenden Unterkapiteln detailliert eingegangen.

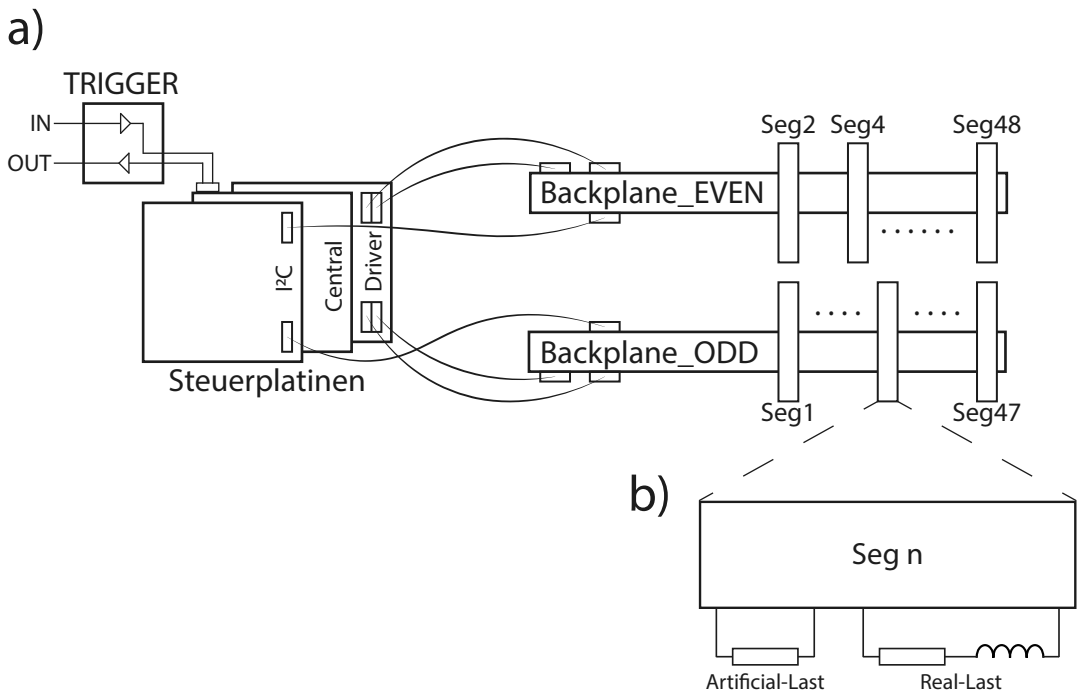

Abbildung 3.1: Skizzierter Plan der beteiligten Hardwarekomponenten. **a** Übersicht; **b** Anschlussplan von Artificial- und Real-Last

3.1 Platinen

Die Automatisierung des MONOPOL-Projekts besteht neben anderen Komponenten aus mehreren Platinen, welche von Andrzej Pelczar entwickelt wurden. Die so entstandene Hardware wurde im Laufe mehrerer Projekte [10, 19–22] in den bestehenden Aufbau eingebunden und getestet.

3.1.1 Steuerplatinen (SP)

Die Steuerplatinen sind das Herzstück der Automatisierung. Sie bestehen aus drei miteinander verdrahteten und übereinander gestapelten Platinen (Abb. 3.2).

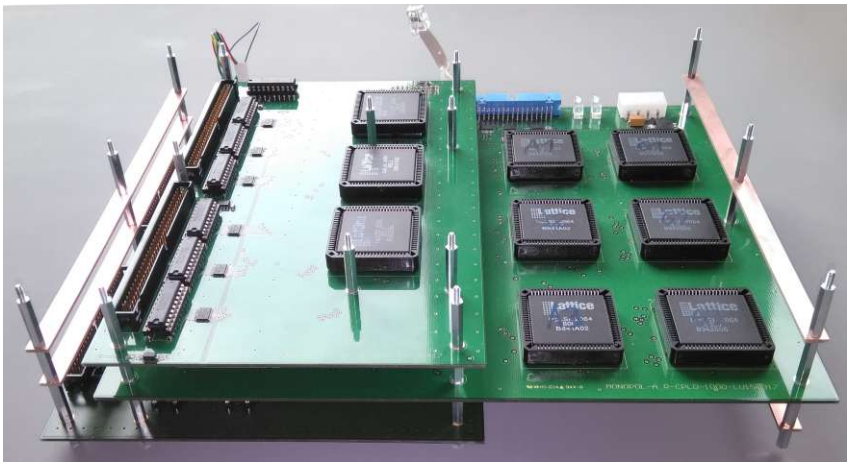

Abbildung 3.2: Foto der übereinander gestapelten Steuerplatinen (Bild aus [10])

I^2C-Steuerplatine

Ganz oben befindet sich die I^2C-Steuerplatine. Auf ihr befindet sich das Schieberegister zur Auswahl der Stromquelle, mit der über I^2C-Bus kommuniziert werden kann. Zu beachten ist, dass immer nur ein einziges High-Bit im gesamten Schieberegister vorhanden sein darf, da andererseits mehrere Bausteine gleichzeitig adressiert werden. Dies würde den Bus überlasten und die Kommunikation fehlschlagen lassen. Die I^2C-Steuerplatine ist mit der Zentralen-Steuerplatine (Schieberegister-Steuerpins), sowie mit den Backplanes (SDA, SCL und Schieberegister-Ausgangpins) verbunden.

Zentrale-Steuerplatine (CP)

In der Mitte ist die eigentliche Steuerplatine. Auf ihr befindet sich neben den Schieberegistern (Artificial-, Real-Last und Glitch) zur Spulenansteuerung auch der Mikrocontroller (µC) der den gesamten Ablauf steuert.
Neben der zuvor oben erwähnten Verbindung mit der I^2C-Steuerplatine, bestehen auch noch Verdrahtungen zur Treiber-Platine sowie zu einem PC über RS485 (UART).

Treiber-Platine

Die Treiber-Platine verstärkt die Ausgangspins der Artificial-Last-, Real-Last- und Glitch-Register und stellt die Konnektoren für die Verbindung zu den Backplanes zur Verfügung.

3.1.2 Backplane (BP)

Die Backplane (Abb. 3.3) besteht aus zwei miteinander verschraubten Platinen und ist das Verbindungsstück zwischen Steuerplatinen und Stromquellen-Segmenten. Alle Schieberegister-Ausgänge sowie, I^2C-Busleitungen werden hier zu den jeweiligen Segmenten entflechtet. Weiters stellt die Backplane ein I^2C-Erweiterungsinterface für zusätzliche Sensorik und Aktorik bereit.

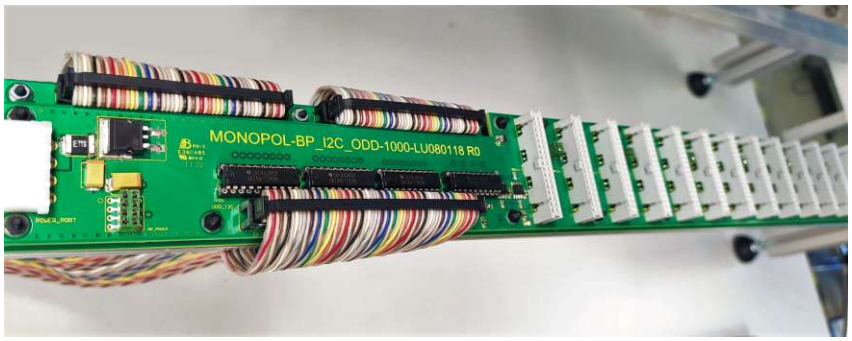

Abbildung 3.3: Foto der Backplane-Platinen

3.1.3 Stromquellen-Segmente

Das Stromquellen-Segment (Abb. 3.4) ist eine über I^2C-Kommunikation steuerbare und auslesbare Stromquelle (siehe Abschnitt 3.4), welche in 48-facher Ausführung im MONOPOL-Projekt zum Einsatz kommt. Sie kann Ausgangsströme bis zu 25 A liefern. Dazu besitzt die Platine zwei DACs (DAC_A: 0 A - 5 A und DAC_B: 0 A - 25 A), die jeweils einen Feldeffekt-Transistor ansteuern. Diese Transistoren erhitzen sich abhängig vom eingestellten Strom und müssen über Temperatursensoren (TMP101) überwacht werden.
Weiters kann die Funktion des Segmentes sowie der angeschlossenen Peripherie über zwei ADCs kontrolliert werden. Dabei wird der Strom über einen Hall-Sensor in eine direkt proportionale Spannung umgesetzt und gemeinsam mit der Spannung an der Spule digitalisiert (Details in Abschnitt 3.4.4). So kann festgestellt werden, ob die zum Segment dazugehörige Spule intakt ist und die festgelegten Ströme eingehalten werden.
Neben der Bereitstellung des Spulenstromes ist das Segment auch für das Umschalten zwischen Artificial- und Real-Last, durch Steuerung über die entsprechenden Schieberegister, verantwortlich. Dieser Vorgang muss möglichst schnell (etwa 4 µs) vonstattengehen, darf aber gleichzeitig nur geringe Überschwinger produzieren. Um das zu gewährleisten wird der Strom für etwa 600 ns über beide Lasten geleitet (= *Glitch*) [10]. Die stromführende Last ist durch LEDs ersichtlich, was zur Schnellüberprüfung dient. Die grüne LED steht für die Artificial- die rote LED für die Real-Last.

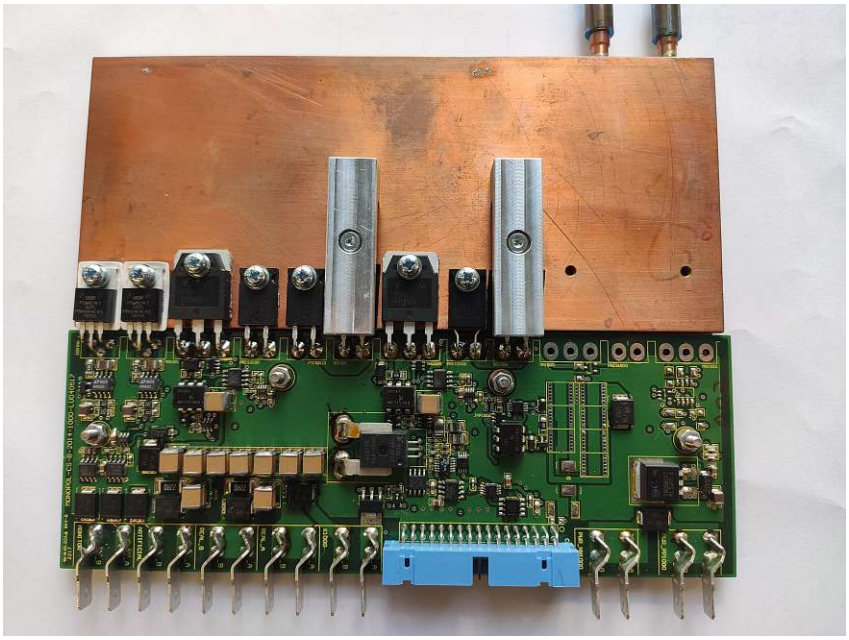

Abbildung 3.4: Stromquellen-Segment

Pinbelegung

In Abb. 3.5 ist der Anschlussplan eines Segmentes zu sehen, die dazugehörige Erklärung liefert Tab. 3.1.

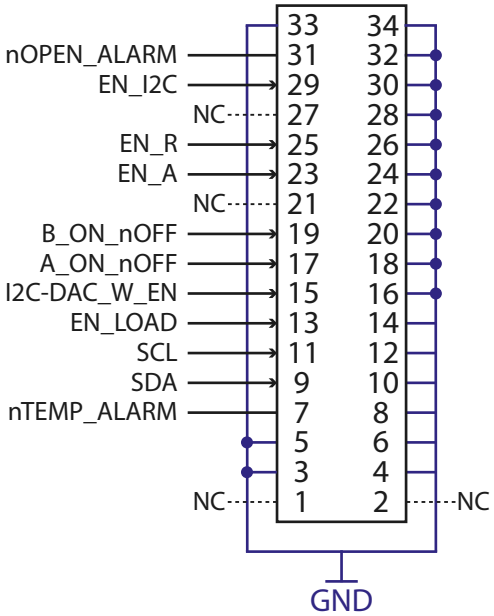

Abbildung 3.5: Anschlussplan eines Stromquellen-Segmentes

Pin#	*Bezeichnung*	*Beschreibung*	*Verbindung zu*
7	nTEMP_ALARM	LOW: Temperatur-Alarm von TMP101 liegt vor	µC, I/O-Extender
9	SDA	I^2C-Datenleitung	µC
11	SCL	I^2C-Clockleitung	µC
13	EN_LOAD	Ermöglicht das Ein- und Ausschalten der Stromquelle; Oder-Verknüpfung zwischen *EN_I_SRC* und *IO_EN_LOAD* (invertiert)	µC, I/O-Extender
15	I2C-DAC_W_EN	Ermöglicht die Kommunikation mit den DACs	µC
17	A_ON_nOFF	Aktiviert den DAC für den niederen Strombereich	I/O-Extender (invertiert)
19	B_ON_nOFF	Aktiviert den DAC für den höheren Strombereich	I/O-Extender (invertiert)
23	EN_A	Schaltet den Strom über die Artificial-Last	SR_A
25	EN_R	Schaltet den Strom über die Real-Last	SR_R
29	EN_I2C	Ermöglicht die I^2C-Kommunikation mit dem jeweiligen Segment	SR_I2C
31	nOPEN_ALARM	LOW: Spule ist nicht korrekt angeschlossen	µC, I/O-Extender

Tabelle 3.1: Pinbeschreibung der Stromquellen-Segmente mit µC ... Mikrocontroller (Abschnitt 3.2), SR ... Schieberegister (Abschnitt 3.3.1) und I/O-Extender (Abschnitt 3.4.5)

3.1.4 Platinen der Artificial- und Real-Last

Diese Platinen werden pro Stromquellen-Segment einmal benötigt und bestehen lediglich aus zwei Widerständen. Einer fungiert als Artificial-Last und der andere als Vorwiderstand für eine Resonatorspule. Die Serienschaltung aus Resonatorspule und Vorwiderstand ergeben die Real-Last.
Um die beim Umschaltvorgang zwischen den Lasten auftretenden Spannungsspitzen so klein wie möglich zu halten, sollten die Real- und die Artificial-Last in etwa den gleichen elektrischen Widerstand aufweisen. In [10] wurde gezeigt, dass $0.22\,\Omega$ Widerstände ein guter Kompromiss zwischen Leistungsaufnahme und Stabilität sind.

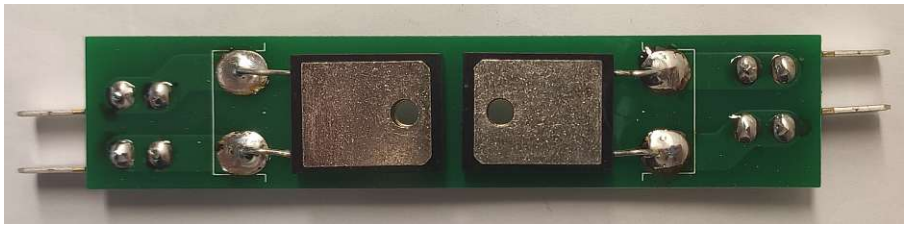

Abbildung 3.6: Platine der Artificial- und Real-Last

3.1.5 Trigger-Platine

Der MONOPOL-Wanderwellenresonator soll im Zusammenspiel mit externen Geräten verwendbar sein. Dazu besitzt er ein Trigger-Interface in Form der Trigger-Platine (Abb. 3.7). Auf ihr befindet sich jeweils ein Signal-Eingang und -Ausgang, beide direkt mit dem Mikrocontroller verbunden. Befindet sich der Mikrocontroller im Trigger-Mode und detektiert ein Signal am dazugehörigen Eingang, wird ein Interrupt ausgelöst und ein Neutronenpaket mit der definierten Geschwindigkeit gefiltert. Sobald der Schiebevorgang abgeschlossen ist und das Paket den Resonator verlässt, wird ein Signal am Trigger-Ausgang zur Synchronisation der nachfolgenden Instrumente ausgegeben. Zum Zeitpunkt dieser Arbeit wurde die Trigger-Platine noch nicht installiert.

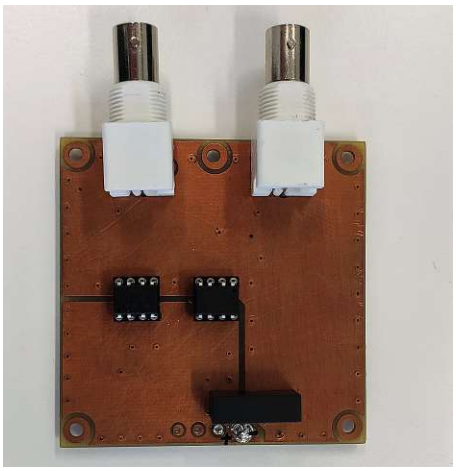

Abbildung 3.7: Trigger-Platine

3.2 Mikrocontroller (µC)

Im MONOPOL-Projekt kommt ein Mikrocontroller des Typs *PIC24FV32KA304* (Anschlussplan in Abb. 3.8, Pinbeschreibung in Tab. 3.2) der Firma *Microchip* als zentrales Steuergerät zum Einsatz.
Der Speicher des Controllers ist gemäß Harvard-Architektur in Programm- (bis zu 4 M Instruktionen) und Daten-Speicher (64 kB) aufgeteilt. Die Adressierung erfolgt

individuell für beide Speicher. Der Programm-Speicher wird über einen 24 bit-Wert, meist abgeleitet vom 23 bit Program Counter (PC), angesteuert und beheimatet das erstellte Programm, bestehend aus den Befehlen der Instruktionen-Liste (zu finden in [23, Kapitel 28]). Der Daten-Speicher, in dem sich auch alle *Working-Registers* befinden, ist 16 bit breit, kann aber byteweise angesprochen werden. Die Adressierung erfolgt dabei über eine 16 bit-Adressleitung.
Weiters stellt der Mikrocontroller einen internen Takt von 8 MHz zur Verfügung, lässt sich aber über einen externen Oszillator auf bis zu 32 MHz takten. Im MONOPOL-Projekt kommt ein 29.4912 MHz Quarz als Taktquelle zum Einsatz, wodurch sich eine Periodendauer von 33.91 ns ergibt. Da der µC zwei solcher Perioden zur Abarbeitung einer Instruktion benötigt, ergibt sich die effektive Instruktionen-Periode $T_I = 67.82\,\text{ns}$. Dieser Wert ist maßgeblich für die Berechnung der Schiebeperiode und ermöglicht somit die Ermittlung der oberen Neutronen-Grenzgeschwindigkeit des Resonators (siehe Abschnitt 6.2.2).
Neben der Verwendung eines externen Clock-Signals bietet der Controller auch benutzerfreundliche Module für unterschiedliche geläufige Schnittstellen. Im MONOPOL werden davon ein UART- und beide I^2C-Module eingesetzt.
Die Versorgung des Mikrocontrollers wird, wie bei der restlichen 5 V-Hardware, über einen durch 12 V gespeisten DC/DC-Konverter sichergestellt.

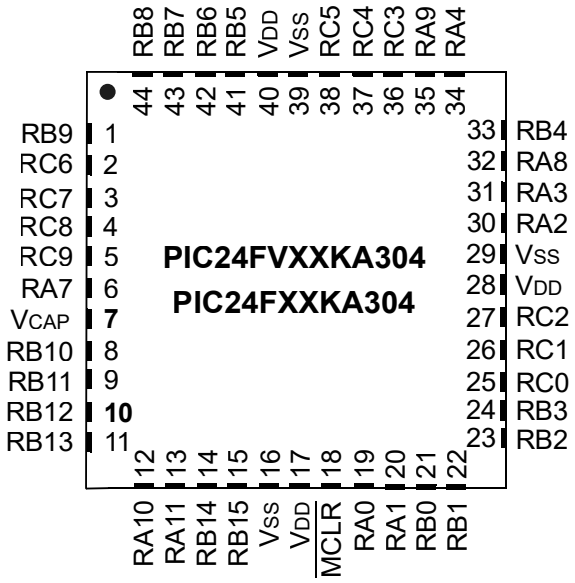

Abbildung 3.8: Anschlussplan des Mikrocontrollers

Funktion	*Port#*	*Pin#*	*Beschreibung*	*Gruppe*
DIN_A	A1	20	Artificial-SR: Dateneingang	SR_A
CLK_A	A0	19	Artificial-SR: Takteingang	SR_A
LAT_A	C1	26	Artificial-SR: Eingang um Muster zu latchen	SR_A
EN_A	C0	25	Artificial-SR: Aktiviere transparenten Modus	SR_A
DLB_A	B10	8	Artificial-SR: Feedback (letztes Bit im SR)	SR_A
DIN_R	A9	35	Real-SR: Dateneingang	SR_R
CLK_R	A8	32	Real-SR: Takteingang	SR_R
LAT_R	C3	36	Real-SR: Eingang um Muster zu latchen	SR_R
EN_R	C2	27	Real-SR: Aktiviere transparenten Modus	SR_R
DLB_R	A4	34	Real-SR: Feedback (letztes Bit im SR)	SR_R
DIN_G	A11	13	Glitch-SR: Dateneingang	SR_G
CLK_G	A10	12	Glitch-SR: Takteingang	SR_G
LAT_G	C5	38	Glitch-SR: Eingang um Muster zu latchen	SR_G
CLR_OUT_G	C4	37	Glitch-SR: Muster am Ausgang wird auf LOW gesetzt	SR_G
DLB_G	A7	6	Glitch-SR: Feedback (letztes Bit im SR)	SR_G
DIN_I2C	C9	5	I2C-SR: Dateneingang	SR_I2C
CLK_I2C	B13	11	I2C-SR: Takteingang	SR_I2C
LAT_I2C	C8	4	I2C-SR: Eingang um Muster zu latchen	SR_I2C
EN_I2C	B12	10	I2C-SR: Aktiviere transparenten Modus	SR_I2C
DLB_I2C	A3	31	I2C-SR: Feedback (letztes Bit im SR)	SR_I2C
ITR_I2C_CLK	B8	44	I2C-Modul 1: Taktleitung	I2C1

Fortsetzung auf der nächsten Seite

Tabelle 3.2: Pinbeschreibung des Mikrocontrollers

Funktion	*Port#*	*Pin#*	*Beschreibung*	*Gruppe*
ITR_I2C_DTA	B9	1	I2C-Modul 1: Datenleitung	I2C1
EXT_I2C_CLK	B3	24	I2C-Modul 2: Taktleitung	I2C2
EXT_I2C_DTA	B2	23	I2C-Modul 2: Datenleitung	I2C2
I2C-DAC_W_EN	B7	43	High: Aktiviere Kommunikation mit DACs	I2C2
RS_nRD	B5	41	LOW: Empfang möglich	UART
RS_pDE	B6	42	HIGH: Senden möglich	UART
RS_MRX	C6	2	Empfangsleitung	UART
RS_MTX	C7	3	Sendeleitung	UART
EN_I_SRC	B14	14	Aktiviert die Stromquellen	Control
ALARM_IN	B11	9	Empfangspin der Alarmfunktion	Control
PS_RTS	B4	33	LOW: Aktiviert den Drucksensor	Control
MCLR	-	18	Masterclear	Control
SYSCLK	-	30	Anschluss des externen Oszillators	Control
Reserved	B0	21	-	-
Reserved	B1	22	Zur Programmierung des µC	Control
AIN_2V048	B15	15	Referenzspannung	Control

Tabelle 3.2: Pinbeschreibung des Mikrocontrollers (Fortsetzung)

3.3 CPLD

CPLD steht für Complex Programmable Logic Device und beschreibt programmierbare Digitaltechnikbausteine. Einmal programmiert, bleibt das Verhalten auch nach Aus- und wieder Einschalten erhalten, ist also nicht flüchtig. Ein weiterer Vorteil gegenüber FPGAs (Field Programmable Gate Array) ist ihre Einfachheit und Benutzerfreundlichkeit. So bieten einige Hersteller die Möglichkeit digitale Schaltungen in einer grafischen Oberfläche nachzubilden und daraus das Programm des CPLD zu generieren. Das zeitliche Verhalten lässt sich dabei oft direkt aus dem Datenblatt der CPLDs und der programmierten Schaltung abschätzen.
Nachteil von CPLDs ist, dass sie aufgrund ihrer begrenzten Hardware-Ressourcen Anwendungen nur bis zu einem gewissen Komplexitätsmaß abdecken können. Ab dann überwiegen die Vorteile des FPGA, vor allem falls ein Produkt in die Massenfertigung gehen soll. Aus dem Code der bei FPGAs verwendeten Hardwarebeschreibungssprachen (VHDL, Verilog) kann direkt eine Chip-Maske abgeleitet werden.
Im folgenden Teil wird kurz auf die Anwendungsgebiete und die Besonderheiten

der CPLDs im Projekt eingegangen. Für detaillierte Ausführungen ist [10] hinzuzuziehen.

3.3.1 CPLDs im MONOPOL

Die im MONOPOL zum Einsatz kommenden CPLDs des Modells *ISP LSI2064 CPLD* stammen von der Firma *Lattice Semiconductor Corporation* und werden alle als 16 bit-Schieberegister eingesetzt. Sie werden in Gruppen von jeweils drei Chips verdrahtet, um ein resultierendes 48 bit-Schieberegister zu erhalten (Abb. 3.9). In Summe werden vier solcher Register verwendet, es werden also 12 CPLDs benötigt. Um Beschädigungen vorzubeugen, ist bei der Handhabung dieser Chips unbedingt auf elektrostatische Aufladung zu achten.

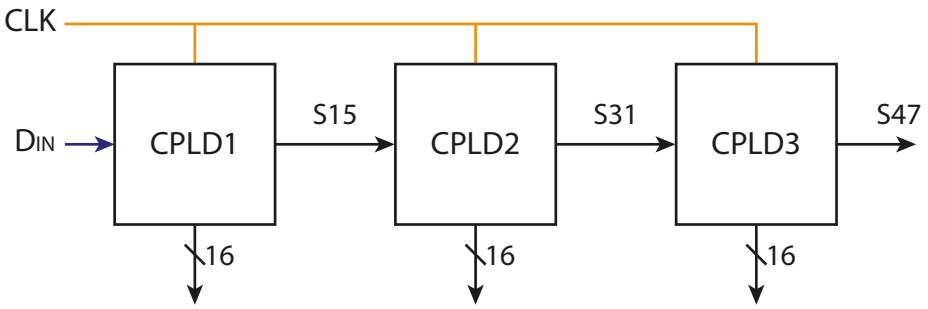

Abbildung 3.9: 48 bit-Schieberegister durch Zusammenschaltung von drei CPLDs; Sxx ... letztes Bit des jeweiligen Registers

Der Anspruch an die Register ist, möglichst schnell zu sein, die entsprechenden Setup- (t_{SU}), Clock-to-Output-Delay- (t_{CO}) und Hold-Zeiten (t_H) sollten also so kurz wie möglich sein. Mithilfe des Datenblatts [24] lassen sich dazu die Werte aus Tab. 3.3 ermitteln. Es zeigt sich, dass zwischen zwei Schaltvorgängen mindestens 93 ns ($= 3 \cdot 31$ ns; drei CPLDs ergeben ein Register) liegen sollten.

Anmerkung: Durch Vergleich mit der Periodendauer eines Instruktionen-Zyklus des Mikrocontrollers (67.82 ns) erkennt man, dass zwischen aufeinanderfolgenden Schaltvorgängen zwei Zyklen vergehen sollten. In der Software wurde dies durch Einfügen zweier NOP-Befehle nach jedem Schaltvorgang sichergestellt.

t_{SU}:	4.6 ns
t_{CO}:	19 ns
t_H:	7.4 ns
GESAMT:	31 ns

Tabelle 3.3: Schaltzeiten der verwendeten CPLDs

Die Schieberegister werden im MONOPOL-Projekt verwendet, um die begrenzte Anzahl an Mikrocontroller-Pins zu erweitern. So kann mit wenigen Steuerleitungen auf 48 parallele Leitungen skaliert werden.
Wie vorhergehend erwähnt, werden vier dieser Register benötigt, welche in zwei unterschiedlichen Ausführungen realisiert werden (Abb. 3.10). Die Beschreibung der einzelnen Ein- und Ausgänge befindet sich in Tab. 3.4.

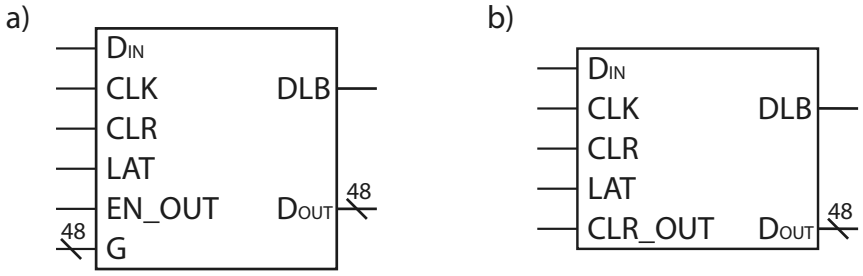

Abbildung 3.10: Varianten der eingesetzten 48 bit-Schieberegister. **a** Real- und Artificial-Lastregister **b** I2C-Control- und Glitch-Register

Bezeichnung	*Beschreibung*	*Variante*
D_{IN}	Daten-Eingangsbit	beide
CLK	Clock-Eingang	beide
CLR	Clear: Reset aller D-Flip-Flops	beide
LAT	Latch: Übernehmen der in den D-FF gespeicherten Werte an den Ausgang	beide
DLB	Bit des letzten D-FF (muss nicht gelatched sein)	beide
D_{OUT}	48 bit-Parallelausgang des Registers	beide
EN_OUT	Aktiviert den transparenten Modus: eingeschobene Bits werden sofort gelatched	a
G	Glitch: High-Pegel von G werden sofort an die Ausgänge weitergegeben, der Wert im D-FF bleibt unverändert	a
CLR_OUT	Alle Ausgänge werden auf LOW gesetzt, der Wert im D-FF bleibt unverändert	b

Tabelle 3.4: Ein- und Ausgänge der Schieberegister

Variante a: Wird bei den Registern zur Steuerung des Stroms über die Real- (SR_R) und Artificial-Last (SR_A) verwendet. Die 48 G-Eingänge sind mit den 48 Pins von D_{OUT} des Glitch-Registers verbunden. Diese Variante kommt außerdem beim I2C-Control-Register (SR_I2C) zum Einsatz, wobei die G-Eingänge nicht verbunden sind.

Variante b: Wird beim Glitch-Register (SR_G) verwendet.

Die Control-Pins (1 bit-Eingänge) aller vier Schieberegister sind genau wie die jeweiligen DLB-Pins direkt mit dem Mikrocontroller verbunden. Einzige Ausnahme sind die CLR-Eingänge, sie sind fest mit GND (0 V) verdrahtet und somit auch nicht verwendbar.

3.4 I²C-Bausteine

In diesem Abschnitt werden die I²C-fähigen Chips des MONOPOL-Projektes behandelt. In Tab. 3.5 findet sich eine Auflistung aller vorhandenen Bausteine samt dazugehöriger Adresse und verbautem Ort. Zusätzlich ist das zuständige I²C-Modul des Mikrocontrollers gelistet.

Bezeichnung	*I²C-Modul*	*Adresse*	*Platine*
PIC24FV32KA304 (µC)	I2C1, I2C2	Master	Zentrale-Steuerplatine
FM24W256-G (FRAM)	I2C1	0b10100000	Zentrale-Steuerplatine
TMP101 (Temperatursensor)	I2C1	0b10010000	Zentrale-Steuerplatine
TCA9554A (I/O-Extender)	I2C2	0b01110000	Backplane ODD
TCA9554A (I/O-Extender)	I2C2	0b01111000	Backplane EVEN
FM24W256-G (FRAM)	I2C2	0b10100000	Stromquellen-Segmente
TMP101 (Temperatursensor)	I2C2	0b10010000	Stromquellen-Segmente
TMP101 (Temperatursensor)	I2C2	0b10010100	Stromquellen-Segmente
MCP3421 (Spannungs-ADC)	I2C2	0b11010010	Stromquellen-Segmente
LTC2461 (Strom-ADC)	I2C2	0b00101000	Stromquellen-Segmente
LTC2631-LM12 (DAC A)	I2C2	0b00100100	Stromquellen-Segmente
LTC2631-LM12 (DAC B)	I2C2	0b00100000	Stromquellen-Segmente

Tabelle 3.5: Übersicht der I²C-Adressen

3.4.1 FRAM

Der FRAM-Chip *FM24W256-G* (Ferroelectric Random Access Memory) der Firma Infineon ist ein nichtflüchtiger 256 kbit Speicher, der über I^2C beschrieben und ausgelesen werden kann.

Der Baustein ist auf der Zentralen-Steuerplatine verbaut und ermöglicht das Speichern verschiedenster Parameter. So könnte der MONOPOL, nach vorheriger Konfiguration, auch ohne direkt verbundenen PC betrieben werden.
Weiters ist der Baustein auf allen Stromquellen-Segmenten verbaut. Hier können PCB-spezifische Parameter gespeichert werden, um etwa die verschiedenen Bauteiltoleranzen auszugleichen und höhere Präzisionen zu erzielen. Eine weitere Anwendung wäre die Hinterlegung der jeweiligen Segment-ID.

3.4.2 Temperatursensor (TMP101)

Der Temperatursensor *TMP101* der Firma *Texas Instruments Incorporated* ist ein über I^2C konfigurierbarer und auslesbarer Chip, der eine Temperaturüberwachungsfunktion bereitstellt. Der Sensor bietet eine einstellbare Auflösung von 9 bit bis 12 bit und deckt einen Temperaturbereich von −55 °C bis 125 °C ab.

Im MONOPOL-Projekt sind 97 Stück der TMP101 verbaut, einer auf der Zentralen-Steuerplatine zum Messen der Umgebungstemperatur, die anderen auf den Stromquellen-Segmenten zur MOSFET-Überwachung.
Die Konfiguration wird nach dem Einschalten des Systems vom Mikrocontroller durchgeführt. Dabei wird der 12 bit Modus für maximale Genauigkeit gewählt. Weiters wird die Alarmtemperatur, soweit nicht anders vom Benutzer festgelegt, auf 90 °C eingestellt. Der Alarm wird ausgelöst, sobald vier aufeinanderfolgende Messwerte den Grenzwert überschreiten und besteht solange bis ebenfalls vier Werte wieder darunter liegen. Genaue Informationen zu Betrieb und Konfiguration finden sich in [25].

3.4.3 Digital-Analog Konverter (LTH2631-LM12)

Der Baustein *LTH2631-LM12* der Firma *LINEAR TECHNOLOGY CORPORATION* ist ein 12 bit DAC und wandelt einen digitalen Zahlenwert in eine analoge Spannung um. Details zur Handhabung sind dem entsprechenden Datenblatt [26] zu entnehmen.

Im MONOPOL kommen zwei dieser Komponenten pro Stromquellen-Segment vor. Jeweils einmal für den Strombereich von 0 A bis 5 A (DAC_A) und den Bereich von 0 A bis 25 A (DAC_B). Geplant war, dass durch diese Unterteilung der Stromwert genauer geregelt werden kann.
Die Übersetzung von Spannung zu Strom erfolgt mittels in Abb. 3.11 zu sehender

Beschaltung. Die Ausgangsspannung U_{DAC} kann von 0 V bis 2.5 V variiert werden und wird über einen OPV (*AD8276*) direkt an den Widerstand R_X angelegt. Gemäß dem ohmschen Gesetz ergibt sich der resultierende Strom zu

$$I_X = \frac{U_{DAC}}{R_X}. \tag{3.1}$$

Da die Eingangswiderstände des OPV sehr groß sind, fließt praktisch der gesamte Strom I_X über die niederohmige Last. Verwendet man (aus dem Datenblatt)

$$U_{DAC} = U_{REF} \cdot \frac{z}{2^{12}}, \tag{3.2}$$

mit $U_{REF} = 2.5\,\mathrm{V}$ ergibt sich der Zusammenhang zwischen übertragenen digitalen Zahlenwert z und dem resultierenden Strom I_X mit

$$I_X(z) = \frac{U_{REF}}{R_X} \cdot \frac{z}{2^{12}}. \tag{3.3}$$

Da pro Stromquellen-Segment zwei DACs verbaut sind, ist die in der Abb. 3.11 gezeigte Schaltung ebenfalls in doppelter Ausführung vorhanden. Der Unterschied besteht in den jeweiligen Widerstandswerten von R_X. Für DAC_A gilt

$$R_{X,DAC\_A} = R_A = 0.5\,\Omega, \tag{3.4}$$

und für DAC_B

$$R_{X,DAC\_B} = R_B = 0.1\,\Omega. \tag{3.5}$$

Durch Division von U_{REF} mit den jeweiligen Widerstandswerten erhält man die maximal einstellbare Stromstärke:

$$I_{A,max} = \frac{U_{REF}}{R_A} = 5\,\mathrm{A}, \tag{3.6}$$

$$I_{B,max} = \frac{U_{REF}}{R_B} = 25\,\mathrm{A}. \tag{3.7}$$

Anmerkung: Da sich im Falle unterschiedlicher Ströme die Ausgangsspannungen der beiden DAC-Stromquellen voneinander unterscheiden, können die Ströme nicht einfach aufaddiert werden, um die resultierende Stromstärke zu ermitteln. Will man dennoch die beiden Stromquellen simultan betreiben, kann der resultierende Strom mittels Thévenin-Theorem beziehungsweise zuvor bestimmter Übersetzungstabelle bestimmt werden.

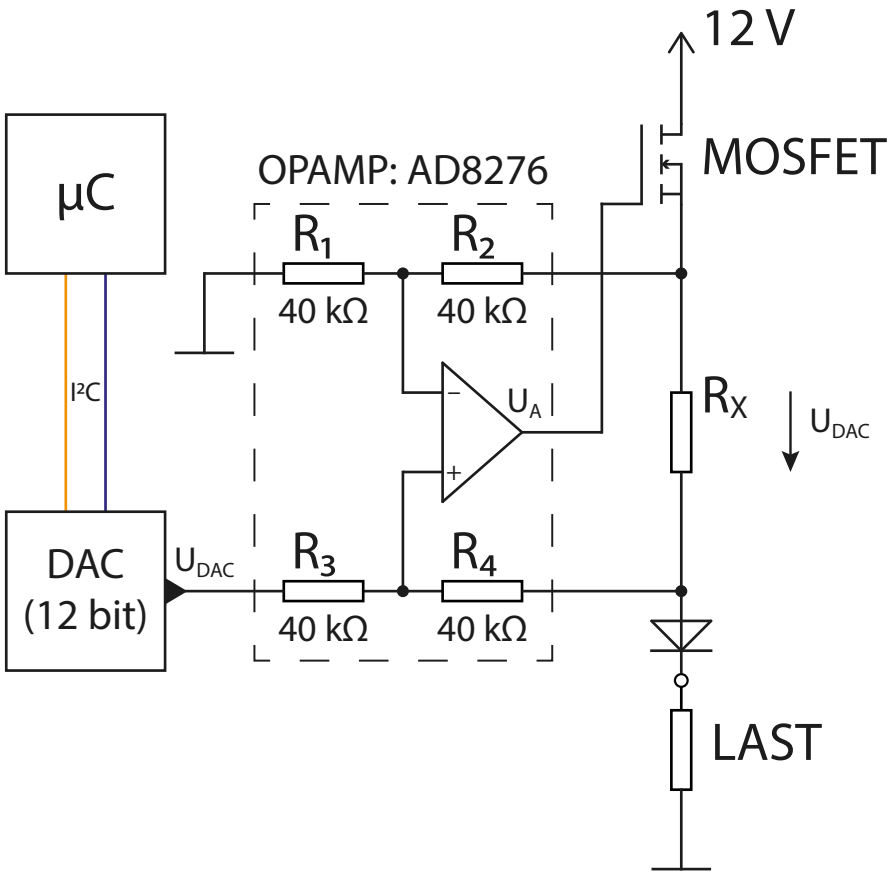

Abbildung 3.11: Stromquellensteuerung über Digital-Analog Konverter

3.4.4 ADCs

Um die Funktion des Resonator-Systems zu überprüfen, sind je Stromquellen-Segment zwei ADCs, jeweils einer zur Digitalisierung von Spulenspannung (MCP3421) und Spulenstrom (LTC2461), verbaut. In Abb. 3.12 sind alle relevanten Teile der Spannungs- und Stromwertdigitalisierung einsehbar.

Spulenspannung: MCP3421 (12 bit ADC)

Der ADC *MCP3421* [27] wird von *Microchip Technology Inc.* hergestellt und bietet Auflösungen bis zu 18 bit. Im MONOPOL-Projekt wird jedoch die Standardeinstellung von 12 bit verwendet, wodurch der Chip sofort nach einem Power-on-Reset einsatzbereit ist und nicht extra konfiguriert werden muss. Die zu dieser Einstellung dazugehörige Konversionszeit beträgt 4.2 ms. Das Eingangssignal des ADC ist die aufbereitete Spulenspannung U_{LR}. Sie wird mittels Vierleitermessmethode abgegriffen und anschließend durch einen Differenzverstäker-OPV [28] verstärkt. So kann festgestellt werden, ob die jeweiligen Spulen intakt sind und ausreichenden elektrischen Kontakt zur Ablaufsteuerung haben. Ist dies nicht der Fall, ist der

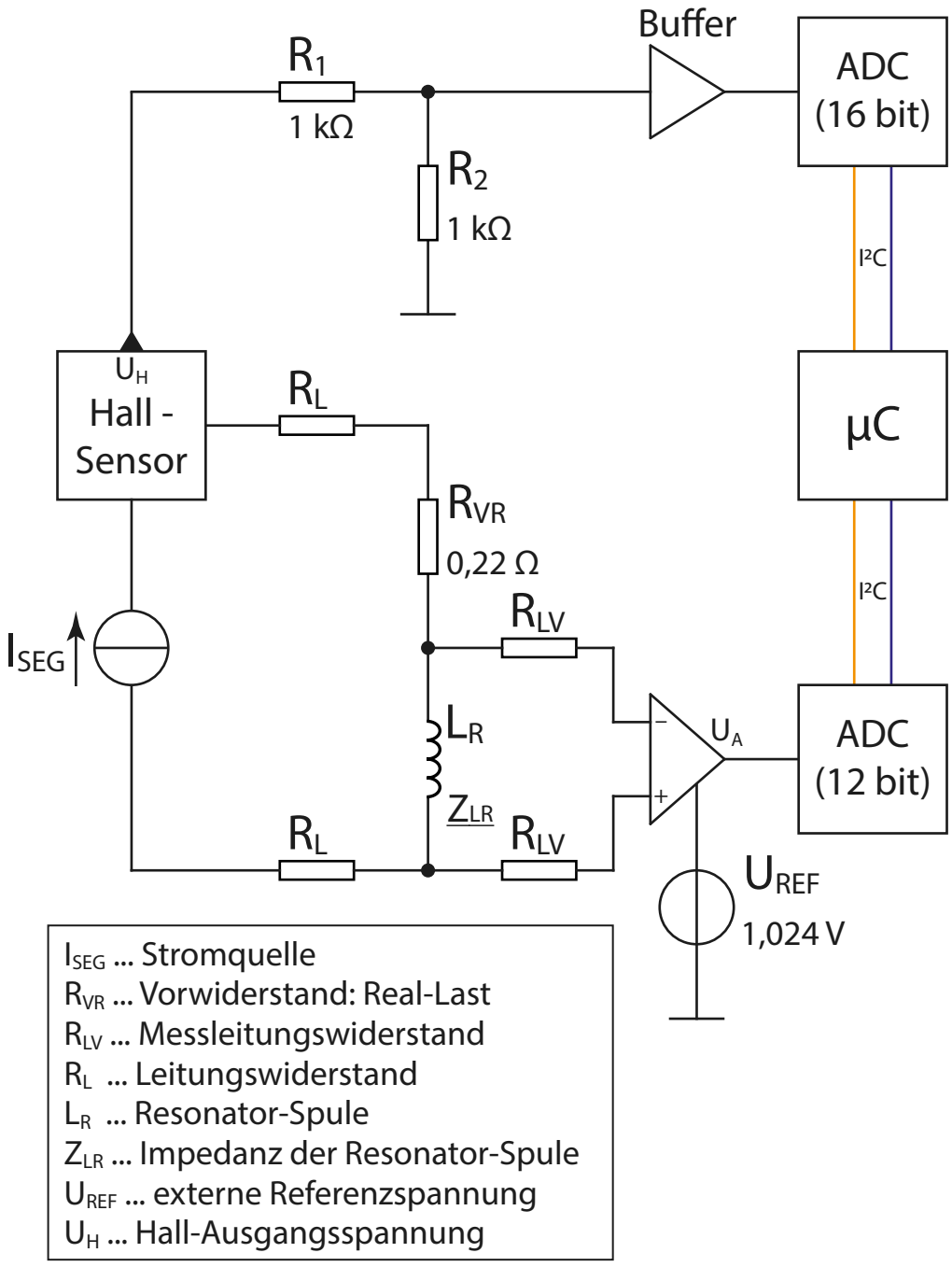

Abbildung 3.12: Schaltung zur Messung von Spulenspannung und -strom

Spannungsabfall U_{LR} über der Norm und somit detektierbar.

Durch die sehr großen Eingangswiderstände des Verstärkers wird die Spule praktisch nicht belastet und es gilt näherungsweise für die Ausgangsspannung U_A des Verstärkers

$$U_A = \frac{U_+ - U_-}{10} + U_{REF} \approx \frac{U_{LR}}{10} + U_{REF}. \tag{3.8}$$

Da der ADC im 12 bit Modus betrieben wird, ergibt sich der digitalisierte Spannungswert z_U (ganzzahlig gerundet) zu

$$z_U \approx 2^{11} \cdot \frac{U_A}{V_{REF}}, \tag{3.9}$$

wobei die interne Referenzspannung V_{REF} des Chips 2.048 V beträgt. Durch Umformen und Einsetzen von Gl. (3.8) erhält man

$$U_{LR}(z_U) \approx 10 \cdot (\frac{z_U}{2^{11}} \cdot V_{REF} - U_{REF}). \tag{3.10}$$

Setzt man nun die Grenzwerte von z_U ein erhält man den abgedeckten Spulenspannungsbereich

$$U_{LR}(0) = -10.24\,\mathrm{V}, \tag{3.11}$$
$$U_{LR}(2^{11}) = 10.24\,\mathrm{V}. \tag{3.12}$$

Spulenstrom: LTC2461 (16 bit ADC)

Der zweite ADC ist der Chip *LTC2461* [29] von *LINEAR TECHNOLOGY CORPORATION*. Er misst den eingestellten Strom eines Segmentes, der über einen Hall-Sensor in eine direkt proportionale Spannung übersetzt wird. So kann durch Vergleich von Soll- und Ist-Wert die Funktionstüchtigkeit der DAC-Stromquellen sichergestellt werden.
Dieser ADC kann nicht kontinuierlich betrieben werden und beginnt erst nach Lesen des alten Wertes eine neue Konversion, was bis zu 23 ms in Anspruch nehmen kann.

Aus dem Datenblatt des Hall-Sensors *ACS756SCA-050B-PFF-T* [30] ist folgende Relation zwischen Eingangsstrom I_{SEG} und Ausgangsspannung U_H zu entnehmen

$$U_H(I_{SEG}) = U_H(0) - 0.04\,\frac{\mathrm{V}}{\mathrm{A}} \cdot I_{SEG}, \tag{3.13}$$

wobei $U_H(0) \approx \frac{3.3\,\mathrm{V}}{2} = 1.65\,\mathrm{V}$, die von der Versorgungsspannung (3.3 V) abhängige Ausgangsspannung im stromlosen Zustand ist. Der ADC greift U_H über einen

Spannungsteiler mit darauffolgenden Buffer ab

$$U_{ADC} = \frac{1\,\mathrm{k\Omega}}{2\,\mathrm{k\Omega}} \cdot U_H. \tag{3.14}$$

Die Konversion von U_{ADC} erfolgt relativ zur ADC-internen Referenzspannung $V_{REF} = 1.25\,\mathrm{V}$ und liefert den digitalen Zahlenwert z_I (ganzzahlig gerundet)

$$z_I \approx (2^{16} - 1) \cdot \frac{U_{ADC}}{V_{REF}} \tag{3.15}$$

Durch Einsetzen und Umformen der Gl. (3.13) und (3.14) ergibt sich der gemessene Strom I_{SEG} zu

$$I_{SEG}(z_I) \approx 25 \cdot (U_H(0) - \frac{2\,\mathrm{k}}{1\,\mathrm{k}} \cdot \frac{z_I}{2^{16} - 1} \cdot V_{REF}). \tag{3.16}$$

Da der Hall-Sensor in negativer Zählrichtung verbaut ist, muss die gesamte Gleichung noch mit „-1“ multipliziert werden

$$I_{SEG}(z_I) \approx -25 \cdot (U_H(0) - \frac{2\,\mathrm{k}}{1\,\mathrm{k}} \cdot \frac{z_I}{2^{16} - 1} \cdot V_{REF}). \tag{3.17}$$

Zusätzliche Anpassungen, welche im Abschnitt 6.2.1 „Anpassung der Strommessung über ADC“ diskutiert werden führen zur finalen Gleichung der Spulenstrommessung mittels ADC

$$I_{SEG}(z_I) \approx -25 \cdot (U_H(0) - \frac{2\,\mathrm{k}}{1\,\mathrm{k}} \cdot \frac{z_I}{2^{16} - 1} \cdot V_{REF}) \cdot \frac{5}{3.18} - 0.7. \tag{6.5}$$

Den abgedeckten Strombereich kann man durch Einsetzen der z_I-Grenzwerte bestimmen

$$I_{SEG}(0) = -65.56\,\mathrm{A}, \tag{3.18}$$
$$I_{SEG}(2^{16} - 1) = 32.71\,\mathrm{A}. \tag{3.19}$$

Da die Ströme der Quellen nur in positive Zählrichtung eingestellt werden können, gilt

$$I_{SEG}\ \epsilon\ [0,\ 32.71]\mathrm{A} \tag{3.20}$$

Anmerkungen: Der Wert z_I ist indirekt proportional zum Strom I_{SEG}. Der gemessene Wert ist durch diverse Bauteiltoleranzen ungenau und sollte deswegen nur zur groben Bestimmung der Funktionstüchtigkeit herangezogen werden. Die Messung des Hall-Sensors kann möglicherweise durch die magnetischen Felder im Resonator beeinflusst werden.

3.4.5 I/O-Extender

Der I/O-Extender *TCA9554A* der Firma *Texas Instruments Incorporated* ist, wie der Temperatursensor TMP101, ein über I^2C konfigurierbarer und auslesbarer Chip. Seine Hauptfunktion ist das Erweitern der GPIO-Pins.

Der Bauteil ist in Summe zweimal verbaut und befindet sich auf den Backplanes. In Abb. 3.13 ist der logische Anschlussplan der TCA9554A zu sehen, die dazugehörigen Erklärungen finden sich in Tab. 3.6. Genaue Informationen zu Betrieb und Konfiguration können [31] entnommen werden.

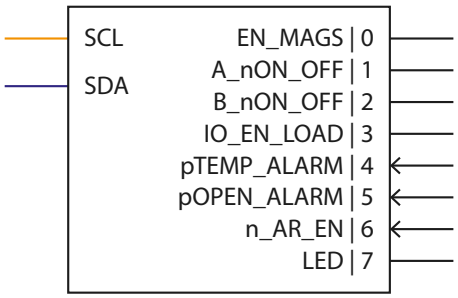

Abbildung 3.13: Anschlussplan des TCA9554A

Port#	*Bezeichnung*	*Beschreibung*	*Modus*
0	EN_MAGS	LOW: Aktiviert den Magnetsensor	Output (O)
1	A_nON_OFF	LOW: Aktiviert den DAC für den niederen Strombereich	O
2	B_nON_OFF	LOW: Aktiviert den DAC für den höheren Strombereich	O
3	IO_EN_LOAD	HIGH: Aktiviert die Stromquellen, wenn *IO_EN_LOAD* (µC) aktiv	O
4	pTEMP_ALARM	HIGH: Einer der Temperatursensoren schlägt Alarm	Input (I)
5	pOPEN_ALARM	HIGH: Eine der Resonatorspulen hat keinen geschlossenen Kontakt	I
6	n_AR_EN	LOW: Die Backplanes sind korrekt miteinander verbunden	I
7	LED	HIGH: Die LED auf der jeweiligen Backplane leuchtet	O
-	SCL	I^2C-Clockleitung	I/O
-	SDA	I^2C-Datenleitung	I/O

Tabelle 3.6: Pinbeschreibung des TCA9554A

3.5 Stromversorgung

Die Ablaufsteuerung des Resonators wird mit 12 V Gleichspannung betrieben und benötigt unter Volllast einen Gesamtstrom von bis zu 1200 A. Um diesen Energiebedarf sicherzustellen, wurde das Konzept aus Abb. 3.14 entwickelt.
Da die Ausgangsspannung der Netzgeräte nicht den Anforderungen entsprach, wird ein Einweggleichrichter mit Glättungskondensatoren dazwischengeschaltet.

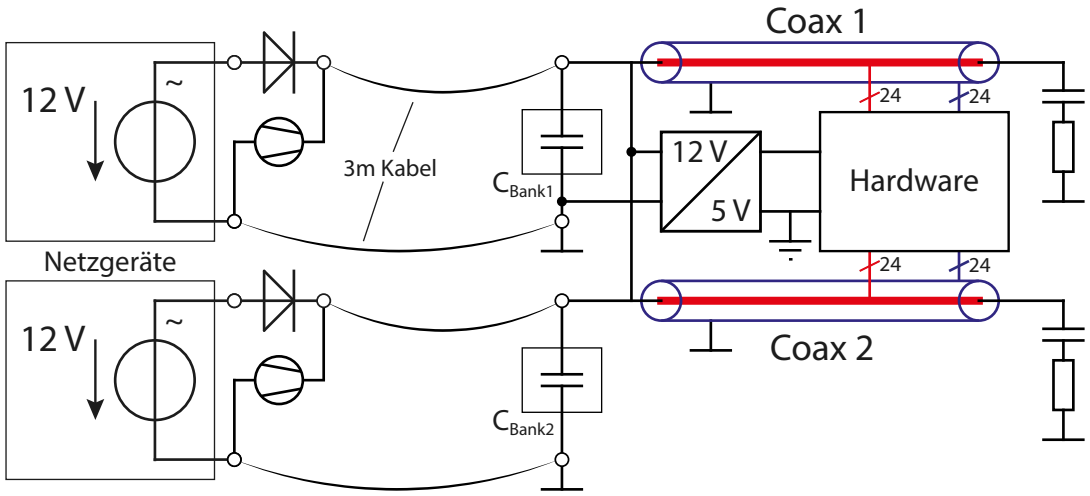

Abbildung 3.14: Übersicht der Stromversorgung

3.5.1 Netzgeräte (12V)

Insgesamt wurden drei Netzgeräte mit einer Nennleistung von je 18 kW angeschafft. Ursprünglich sollten die 12 V Netzgeräte direkt an die koaxialen Versorgungsleitungen angeschlossen werden. Leider zeigte sich nach Messungen von Andrzej Pelczar und Arno Frank [32], dass die Qualität der Ausgangsspannung der Netzgeräte, entgegen der Herstellerangaben, keinesfalls ausreichend für den geplanten Einsatz war (siehe Abb. 3.15). Um die erworbenen Netzgeräte dennoch verwenden zu können, wurde der Einsatz eines Einweggleichrichters beschlossen.

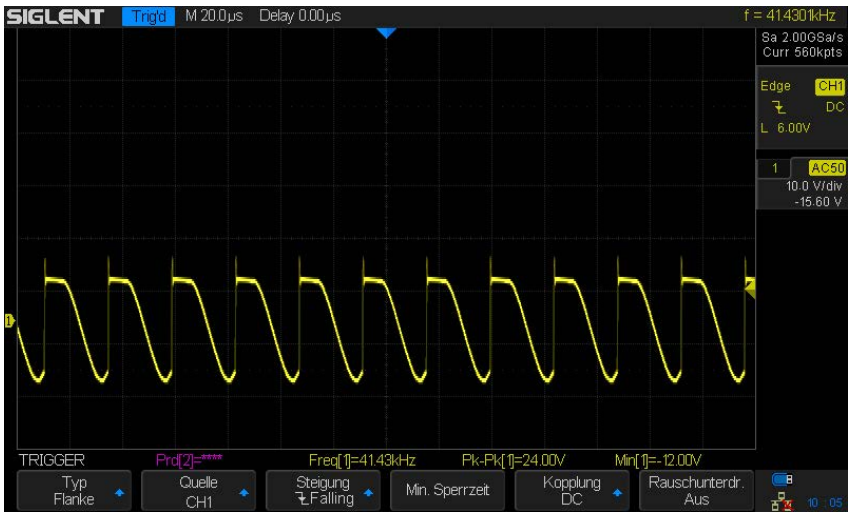

Abbildung 3.15: Ausgangsspannung der 12 V Netzgeräte ohne Last (Bild aus [32])

3.5.2 Einweggleichrichter

Der Einweggleichrichter mit angeschlossenem Kondensator wird verwendet, um eine Wechselspannung in eine möglichst konstante Gleichspannung umzuwandeln. Dabei blockt die Diode die negative Halbwelle und lässt nur die positive Spannung (abzüglich der Flussspannung) passieren. Letztere lädt einen Kondensator, welcher den Zeitraum der negativen Halbwelle überbrückt. Will man beide Halbwellen nutzen, so ist eine Brückengleichrichter zu verwenden.

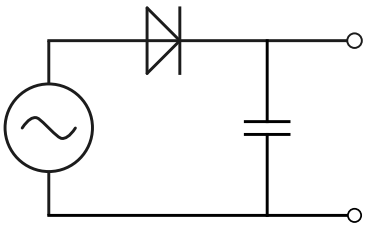

Abbildung 3.16: Schaltung eines Einweggleichrichters mit Glättungskondensator

Schutzdiode

Ursprünglich sollte eine Schottkydiode verwendet werden, da diese eine niedrigere Flussspannung und somit auch eine deutlich geringere Hitzeentwicklung mit sich bringt. Nach langer Recherche stellte sich jedoch heraus, dass keine Schottkydiode mit ausreichenden Spezifikationen lieferbar war, weswegen die Wahl auf das Modell *W0944WC150* von *IXYS UK Westcode* fiel. Hierbei handelt es sich um eine Gleichrichterdiode, welche unter der erwarteten maximalen Last von 600 A eine Verlustleistung von 650 W aufweist. Um diese Diode zu befestigen und zu kühlen wurde gemeinsam mit dem Hersteller eine Lösung erarbeitet, die bis zu 800 A betreibbar ist (zu sehen in Abb. 3.17).

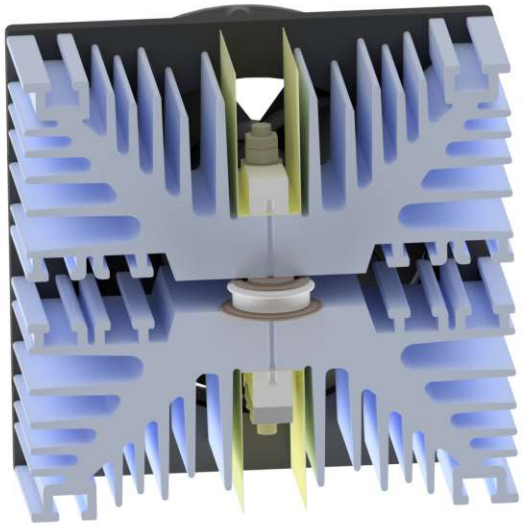

Abbildung 3.17: Rendering der Diode inklusive Halterung und Kühlkörper (Maße in mm: $200 \times 211 \times 190$, Bild von Roman Gergen)

Kondensatorbank

Der Glättungskondensator besteht nicht aus einem einzigen Kondensator, sondern aus einer Kondensatorbank, welche 192 parallel geschaltete und individuell tauschbare Kondensatoren (je 680 µF) beheimatet. Die Gesamtkapazität beläuft sich auf 130.56 mF. Diese Anzahl ist notwendig, um Beschädigungen durch hohe Einschaltströme zu verhindern.
Hauptaufgabe der Kondensatoren ist das Stützen der Betriebsspannung gegen Schwankungen, ausgelöst durch die Netzgeräte oder aber auch die plötzliche Laständerung.

3.5.3 Koaxial-Versorgungsleitung

Die parallelen Versorgungsleitungen des Resonators (Abb. 3.18) führen unter Volllast Ströme bis zu 600 A, was gemäß dem Ampèreschen Gesetz (Gl. (2.7)) ein korrespondierendes magnetisches Feld verursacht. Dieses würde mit dem Führungsfeld des Resonators interferieren und die Funktion beeinträchtigen. Um dem entgegenzuwirken, wurden Hin- und Rückleiter in Form eines Koaxialkabels realisiert. Aufgrund der hohen Ströme benötigen die Leiter in koaxialer Bauform entsprechende Querschnitte, weswegen sie aus einem Kupferstab und einem Kupferrohr gefertigt werden.
Wie in Abschnitt 2.2.4 beschrieben, müssen Koaxialkabel immer über einen Widerstand, der Reflexionen verhindert, abgeschlossen werden. In der vorliegenden Anordnung besteht dieser aus einer Reihenschaltung eines Kondensators mit einem Widerstand.

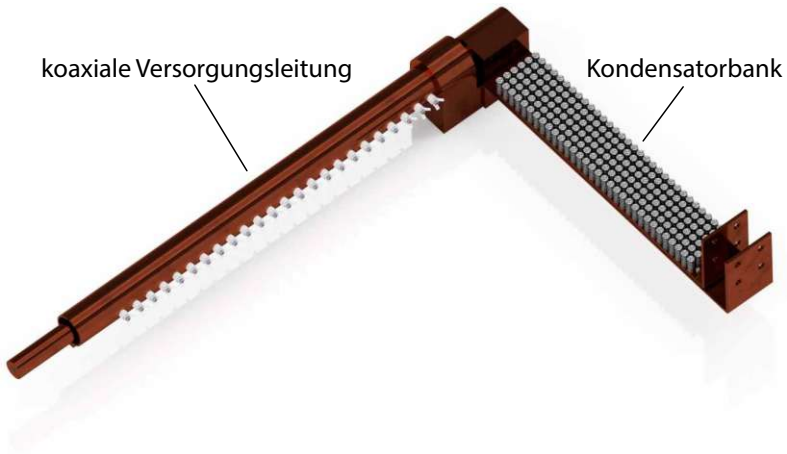

Abbildung 3.18: Rendering der koaxialen Versorgungsleitung (Länge: $\sim$890 mm) inklusive Kondensatorbank (Länge: $\sim$515 mm, Bild und Design von Roman Gergen)

3.5.4 DC/DC-Konverter (5V)

Um den MONOPOL zu betreiben, benötigt man neben der 12 V Versorgungsspannung auch noch eine 5 V Spannung. Diese wird durch einen DC/DC-Konverter (RSD-30) bereitgestellt, der bis zu 6 A Ausgangsstrom liefern kann.
Wichtig ist, dass der Konverter keine interne Verbindung zwischen Eingangs- und Ausgangsspannung GND aufweist. Dadurch wird die 5 V-Elektronik von den großen Strömen der Stromquellen-Segmente isoliert und geschützt.

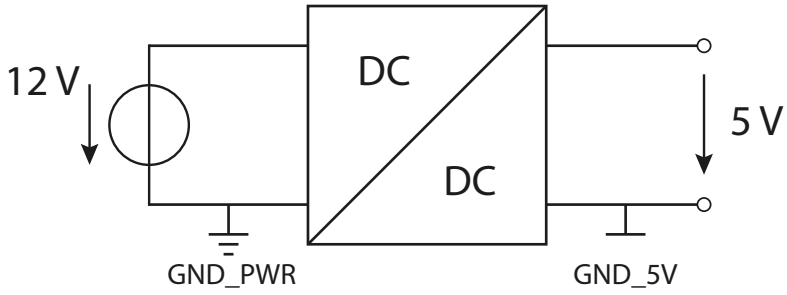

Abbildung 3.19: Eingang- und Ausgangsspannung des DC/DC-Konverters

4 Softwareübersicht

Die Software des MONOPOL-Projekts unterteilt sich in zwei miteinander kommunizierende Teile. Einmal die grafische Oberfläche, über die der Benutzer mit der Ablaufsteuerung interagiert, und einmal die Ablaufsteuerung selbst.
Der Softwareanteil der vorliegenden Arbeit befasst sich lediglich mit der Ablaufsteuerung, weswegen nur kurz auf das User-Interface eingegangen wird.

4.1 User-Interface

Das grafische User-Interface (GUI) ist die Schnittstelle zwischen Benutzer und Ablaufsteuerung. Über sie werden alle relevanten Einstellungen und Auswertungen getätigt und mittels UART über ein definiertes Protokoll (siehe Abschnitt 5.2.1) an den Mikrocontroller übermittelt.
Bisher existiert eine erste Version der GUI, welche von Maya Sajatovic [33] erstellt wurde. Diese wird derzeit im Zuge einer Bachelorarbeit überarbeitet, um weitere Funktionalitäten zu integrieren.

4.2 Ablaufsteuerung

Die Software der Ablaufsteuerung muss eine ganze Reihe von Aufgaben bewältigen. Sie muss mit der angeschlossenen Peripherie kommunizieren, sei es über logische Pegel oder Kommunikationsprotokolle wie UART beziehungsweise I^2C. Befehle, welche so vom Nutzer an die Steuerung übermittelt werden, müssen weiters dekodiert, interpretiert und ausgeführt werden.

Die Ablaufsteuerung soll in drei verschiedenen Modi betrieben werden können. Den Static Mode (SM), den Conventional Mode (CM) und den Traveling Wave Mode (TWM). Die anfallenden Schiebezeiten sind dabei so gering wie möglich zu halten. So kann die einstellbare Geschwindigkeit des erhaltenen Neutronenpakets maximiert werden.
Zusätzlich dazu, sollen CM und TWM im Trigger-Modus betrieben werden können. Bei diesem wird erst dann ein Neutronenpaket erzeugt, wenn ein externes Signal registriert wird.

Um Fehlern und infolgedessen Beschädigungen vorzubeugen, soll das System eine Selbstüberprüfung vornehmen und deren Ergebnisse an den Benutzer kommunizieren. Sollten überwachte Parameter wie die Temperatur oder der Spulenkontakt

außerhalb definierter Normen liegen, so muss die Software in der Lage sein, das System herunterzufahren.

4.3 Mikrocontroller-Toolchain

Wie in Abschnitt 3.2 Mikrocontroller beschrieben, wird ein Prozessor vom Typ *PIC24FV32KA304* von *Microchip* zur Kontrolle der Ablaufsteuerung verwendet. Die Erstellung des Quelltexts erfolgt über die vom Hersteller bereitgestellte Entwicklungsumgebung *MPLAB X IDE v3.50*. Die Übersetzung wird mit dem ebenfalls von Microchip stammenden Compiler *XC16* durchgeführt. Dieses Toolchain-Setup ermöglicht Programmierungen in *C* und *Assembler* und bietet für letzteres auch die Unterstützung von Makros und Präprozessoranweisungen. Zusätzlich kann der Code auch im Simulator oder durch einen In-Circuit Emulator getestet werden.
Da die hier beschriebene Anwendung höchst zeitkritisch ist, wird die Programmierung in der programmiertechnisch aufwendigeren Assemblersprache abgewickelt. Sie bietet, durch den Minimalismus und die vordefinierte Abarbeitungsdauer der einzelnen Befehle, entscheidende Vorteile gegenüber höheren Sprachen und ist somit schneller sowie zeitlich determiniert.
Ein Nachteil von Assemblersprachen ist, dass sie sich je nach verwendetem Compiler und programmierten Prozessor unterscheiden, wodurch je nach Anwendungsfall auf die entsprechende Dokumentation zurückgegriffen werden muss. Für dieses Projekt wurden folgende vier Dokumente herangezogen:

- PIC24FV32KA304 FAMILY-Datasheet [23]
- PIC24F Family Reference Manual [34]
- 16-bit MCU and DSC Programmer's Reference Manual [35]
- MPLAB® XC16 ASSEMBLER, LINKER AND UTILITIES User's Guide [36]

5 Software der Ablaufsteuerung

Die Ablaufsteuerungssoftware (Abb. 5.1) des Mikrocontrollers startet, wenn die Spannungsversorgung eingeschaltet ist. Bevor das System betriebsfähig ist, muss die Initialisierung mit anschließendem Systemtest (Abschnitt 5.1) erfolgen. Über weite Teile dieser Phase kann der Controller keine Daten mit dem PC austauschen und ist somit auch nicht steuerbar.
Sobald die Testphase abgeschlossen ist, wird die GUI-Software über den aktuellen Zustand der Hardware informiert. Sollten keine Probleme vorliegen, geht die Ablaufsteuerung in den Standby-Modus über und wartet auf neue Kommandos des User-Interfaces. Diese werden über die UART-Schnittstelle übertragen und durch einen Interrupt (Abschnitt 5.2.1) controller-seitig verarbeitet. Durch Setzen der *NEW_CMD*-Flag wird dabei dem Hauptprogramm das Vorliegen neuer Instruktionen vermittelt.
Die Ablaufsteuerung überprüft regelmäßig die *NEW_CMD*-Flag (auch innerhalb der Betriebsmodi) und springt im Falle eines neuen Befehls in die Kommandodekodierung. Diese löscht die Flag und leitet die Abarbeitung des Kommandos ein. Für genauere Details der einzelnen Funktionen sind die jeweiligen Unterkapitel zu konsultieren. Sofern nicht der Start-Befehl erfolgt ist, geht das System nach Beendigung der angeordneten Funktion wieder in den Standby und wartet auf neue Befehle.
Ist jedoch der Start-Befehl erfolgt, wechselt die Software in den gewählten Betriebsmodus (Abschnitt 5.3) und beginnt mit der Erzeugung der Neutronenpakete.

5.1 Initialisierung und Systemtest

Wird das System mit Spannung versorgt, beginnt die Mikrocontroller-Software mit der Initialisierung. Zuerst werden Zahlenkonstanten, wie Temperaturgrenzwerte und Konversionszeiten der ADCs in den Datenspeicher geladen. Danach werden die Arbeitsregister geleert und der Stackpointer initialisiert.
Anschließend werden die einzelnen GPIO-Pins (General Purpose Input Output) nach Verwendungszweck entweder als digitaler Eingang oder Ausgang definiert. Nun kann der Mikrocontroller mit der angeschlossenen Peripherie über TTL (Transistor-Transistor Logik) interagieren, was zur Initialisierung der Schieberegister (SR_A, SR_R, SR_G und SR_I2C) genutzt wird.
Um Kommunikationen durchzuführen, die über TTL hinaus gehen, werden als nächstes die I^2C-Module des Controllers initialisiert (Details in Unterkapitel 5.1.1). Modul 1 dient zur Kommunikation mit der Elektronik auf der Zentralen-Steuerplatine,

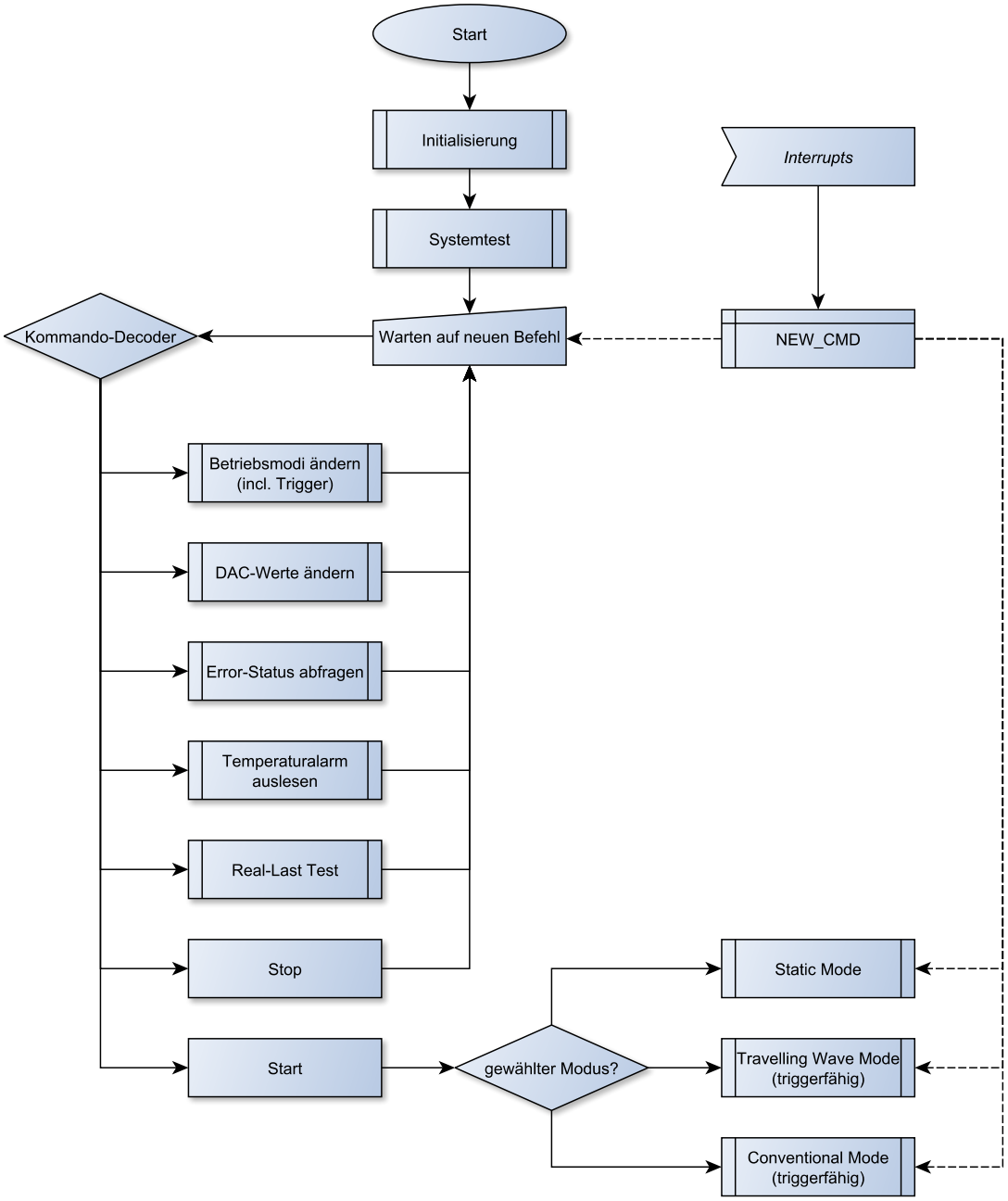

Abbildung 5.1: Flussdiagramm der Ablaufsteuerung

die auch den Mikrocontroller beheimatet. Modul 2 ist mit den über externe Leitungen verbundenen Platinen verknüpft (siehe Tab. 3.5). Es wird nach Abschluss der Konfiguration zur Anpassung der I/O-Extender (Unterkapitel 3.4.5) und der Initialisierung der DACs mit dem Wert 0 verwendet. Letzteres ist nötig, da die DACs nach Reset einen internen Referenzwert ungleich 0 aufweisen und somit einen undefinierten Stromwert hervorrufen würden.
Zum Beenden der Einschaltinitialisierung fehlt jetzt einzig die Konfiguration des UART-Moduls, beschrieben in Unterkapitel 5.1.1.

Jetzt sollte das System einsatzbereit sein. Um dies sicherzustellen, werden noch diverse Tests durchgeführt. Der Testabschnitt beginnt mit der Übertragung des Versionsstrings via UART an den PC. So kann der Benutzer überprüfen, ob die Ablaufsteuerung mit der gewünschten Softwareversion bespielt ist.
Der Systemtest unterteilt sich in drei aufeinander aufbauende Überprüfungen. So sind die Funktionalitäten des vorherigen Tests Voraussetzungen für den nachfolgenden Test. Der erste Test ist die Überprüfung der Schieberegister. Diese werden mit einem vordefinierten Muster geladen. Das Muster wird anschließend wieder aus den Registern herausgeschoben und über die jeweiligen Feedback-Leitungen abgeglichen. Läuft alles problemlos ab, kann mit den nächsten beiden Tests fortgesetzt werden. Der I2C-Test 5.1.2 und der Real-Last Test 5.1.3 sind in ihren eigenen Unterkapiteln beschrieben.
Nach Beendigung des Systemtests wird der ermittelte Fehlercode wieder via UART an den PC versandt und die Software geht in das eigentliche Hauptprogramm über.

Anmerkung: Nach geplanter Implementation der Trigger-Funktionalität stehen die Feedback-Leitungen (DLB_R, DLB_G) der Schieberegister SR_R und SR_G nicht mehr für den Schieberegistertest zur Verfügung.

5.1.1 Initialisierung der Kommunikationsschnittstellen

UART-Interface

Das UART-Modul (UART1), für die serielle Kommunikation über RS485, wird im *Standard-Mode* (kein Parity-Bit, ein Stop-Bit) betrieben und soll auf eine Baudrate von $19\,200\,\mathrm{bit\,s^{-1}}$ eingestellt werden. Um den dazugehörigen Wert des Baud-Rate-Generator Registers (U1BRG) zu berechnen, bedient man sich der Gl. (18-1) aus [23]

$$U1BRG = \frac{F_{CY}}{16 \cdot BaudRate} - 1, \tag{5.1}$$

mit der Instruktionen-Frequenz

$$F_{CY} = \frac{1}{T_I}. \tag{5.2}$$

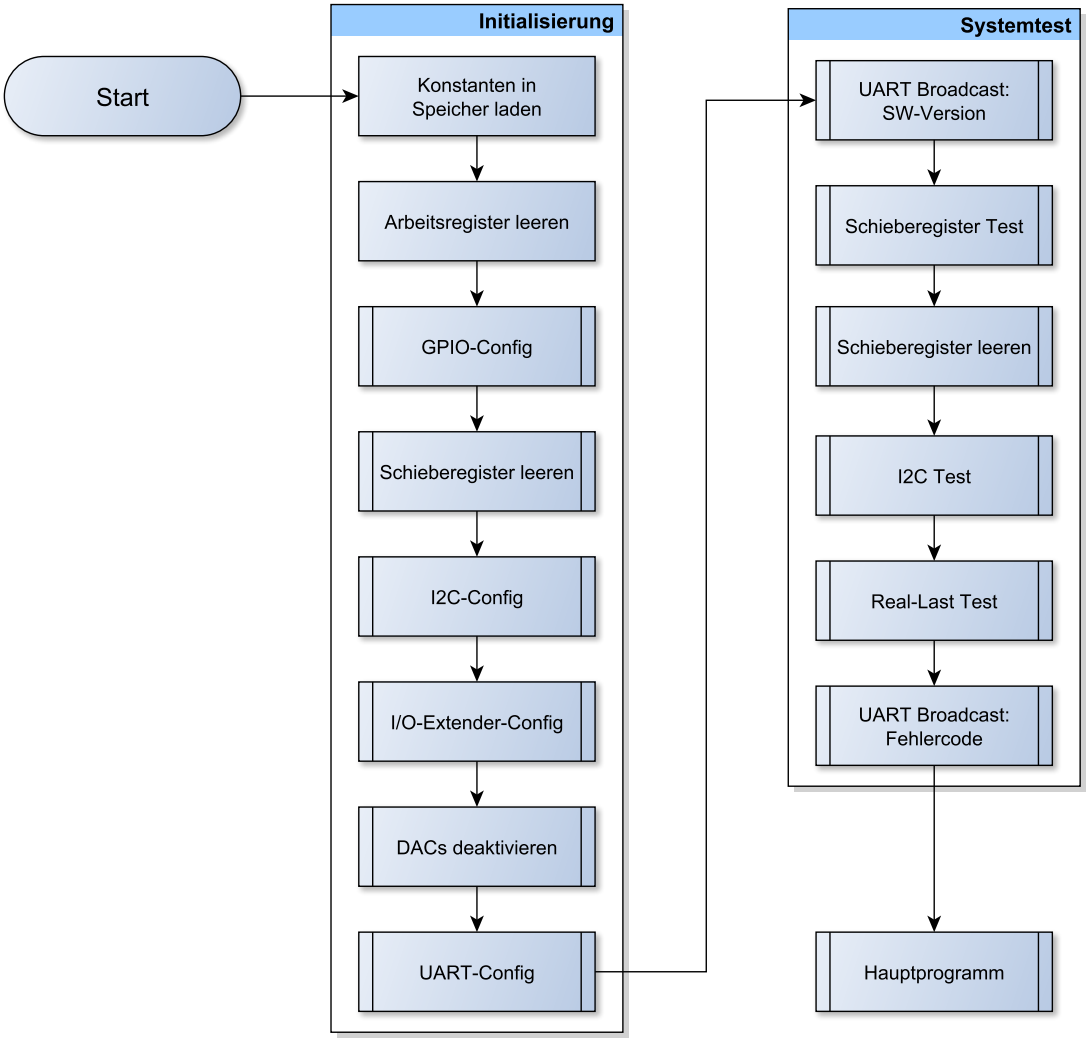

Abbildung 5.2: Ablaufdiagramm von Initialisierung und Systemtest

Setzt man die oben genannte Baudrate, sowie eine Instruktionen-Periode T_I von 67.82 ns ein, ergibt sich der Wert des Baud-Rate-Generator Registers zu

$$U1BRG = \frac{1}{16 \cdot 19\,200\,\mathrm{bit\,s^{-1}} \cdot 67.82\,\mathrm{ns}} - 1 = 47. \tag{5.3}$$

I²C-Interface

Beide I²C-Module (I2C1, I2C2) des Mikrocontroller werden im MONOPOL verwendet und im *Standard-Mode* (bis zu 100 kbit $\mathrm{s^{-1}}$) betrieben. Um die dazugehörige Baudrate zu ermitteln ist Gl. (17-1) aus [23] heranzuziehen

$$I2CxBRG = \left[\frac{F_{CY}}{F_{SCL}} - \frac{F_{CY}}{10^7}\right] - 1, \tag{5.4}$$

wobei F_{SCL} die $100\,\mathrm{kbit\,s^{-1}}$ des *Standard-Mode* sind und F_{CY} wie in Gl. (5.2) definiert ist. Nach Einsetzen ergibt sich für beide Module eine Baudrate von

$$I2CxBRG = \left(\frac{1}{100\,\mathrm{kbit\,s^{-1}} \cdot 67.82\,\mathrm{ns}} - \frac{1}{10^7 \cdot 67.82\,\mathrm{ns}}\right) - 1 = 145. \tag{5.5}$$

In Tab. 3.5 ist noch einmal die Adressauflistung samt den dazugehörigen Modulen aus Abschnitt 3.4 zu sehen. Für genaue Informationen zur Ansteuerung der einzelnen Bauelemente sind die jeweiligen Datenblätter beziehungsweise der entsprechende Assemblercode einzusehen.
Besonders zu beachten ist die Rolle des Schieberegisters SR_I2C bei der Kommunikation mit Bauteilen auf den Stromquellen-Segmenten. In diesem Register darf immer nur ein einziger High-Pegel vorhanden sein. Der Index des High-Pegels ist gleichzeitig der Index des angesprochenen Segmentes (nach Backplane-Slot, nicht Segment-ID).

5.1.2 I2C Test

Der I2C Test unterteilt sich in zwei Abschnitte. Zuerst werden die I^2C-fähigen Chips der allgemeinen Hardware getestet, anschließend die der installierten Stromquellen-Segmente (Ablaufdiagramm in Abb. 5.3). Letztere müssen über das Schieberegister SR_I2C adressiert werden und können dadurch nur nacheinander angesprochen werden.
Der Test der einzelnen Komponenten läuft im Grunde immer gleich ab. Die Adresse des getesteten Bausteins wird auf den Bus gelegt, dieser quittiert mit einem *ACK*. Nun bricht der Busmaster die Kommunikation mit einer *Stop*-Indikation ab und prüft das Acknowledge-Bit. Liegt ein Fehler vor, so wird ein entsprechendes Flag gesetzt, welches über die GUI ausgelesen werden kann.
Die Überprüfung zeigt nicht im Detail an, welches Problem vorliegt, sondern liefert nur einen Überblick. Die so ermittelten fehlerhaften Stromquellen-Segmente können zur genaueren Analyse mit der in Abschnitt 5.4 dargelegten Software getestet werden.

5.1.3 Real-Last Test

Der Real-Last Test (Ablaufdiagramm in Abb. 5.4) dient hauptsächlich zur Überprüfung der Resonatorspulen und deren Kontakte. Zusätzlich wird die Ansteuerung beider Stromquellen (DAC_A, DAC_B), die Messung der Spulenspannung, sowie der Umschaltvorgang zwischen Real- und Artificial-Last getestet.

Bevor der Test beginnen kann, muss das System erst initialisiert werden. Dazu werden beide Stromquellen aller Stromquellen-Segmente via I^2C auf 3.125 A (Übermittlung von $z_{DAC\_A} = 2560$ und $z_{DAC\_B} = 512$ an die DACs) eingestellt. Dadurch,

Bezeichnung	*I²C-Modul*	*Adresse*	*Ort*
PIC24FV32KA304 (µC)	I2C1, I2C2	Master	Zentrale-Steuerplatine
FM24W256-G (FRAM)	I2C1	0b10100000	Zentrale-Steuerplatine
TMP101 (Temperatursensor)	I2C1	0b10010000	Zentrale-Steuerplatine
TCA9554A (I/O-Extender)	I2C2	0b01110000	Backplane ODD
TCA9554A (I/O-Extender)	I2C2	0b01111000	Backplane EVEN
FM24W256-G (FRAM)	I2C2	0b10100000	Stromquellen-Segmente
TMP101 (Temperatursensor)	I2C2	0b10010000	Stromquellen-Segmente
TMP101 (Temperatursensor)	I2C2	0b10010100	Stromquellen-Segmente
MCP3421 (Spannungs-ADC)	I2C2	0b11010010	Stromquellen-Segmente
LTC2461 (Strom-ADC)	I2C2	0b00101000	Stromquellen-Segmente
LTC2631-LM12 (DAC A)	I2C2	0b00100100	Stromquellen-Segmente
LTC2631-LM12 (DAC B)	I2C2	0b00100000	Stromquellen-Segmente

Tabelle 3.5: Übersicht der I²C-Adressen (aus Abschnitt 3.4)

dass die Ströme bis auf tolerierbare Abweichungen gleich sind, addiert sich der Gesamtstrom gemäß der Kirchhoffschen Knotenregel auf 6.25 A (±Toleranzen).
Nachdem alle Stromquellen initialisiert sind, werden die Ströme nacheinander auf ihren jeweiligen Artificial-Lasten aktiviert, was Stromspitzen entgegenwirken soll. Ist dieser Vorgang abgeschlossen, wird der Einschwingvorgang abgewartet und auf die zu messende Real-Last umgeschaltet.
Jetzt beginnt der eigentliche Test. Die ADCs zur Messung der Spulenspannung der einzelnen Segmente werden nacheinander angesteuert und ausgelesen. Sollte der erhaltene Wert außerhalb eines definierten Intervalls liegen, liegt ein Fehler vor und das Segment wird vorgemerkt.
Wurden alle Werte überprüft, werden die Stromquellen von der Last getrennt und wieder mit ihren ursprünglichen DAC-Werten initialisiert. Will der Benutzer das Ergebnis des Tests erfahren, muss er den Error-Status über die GUI-Software abfragen.

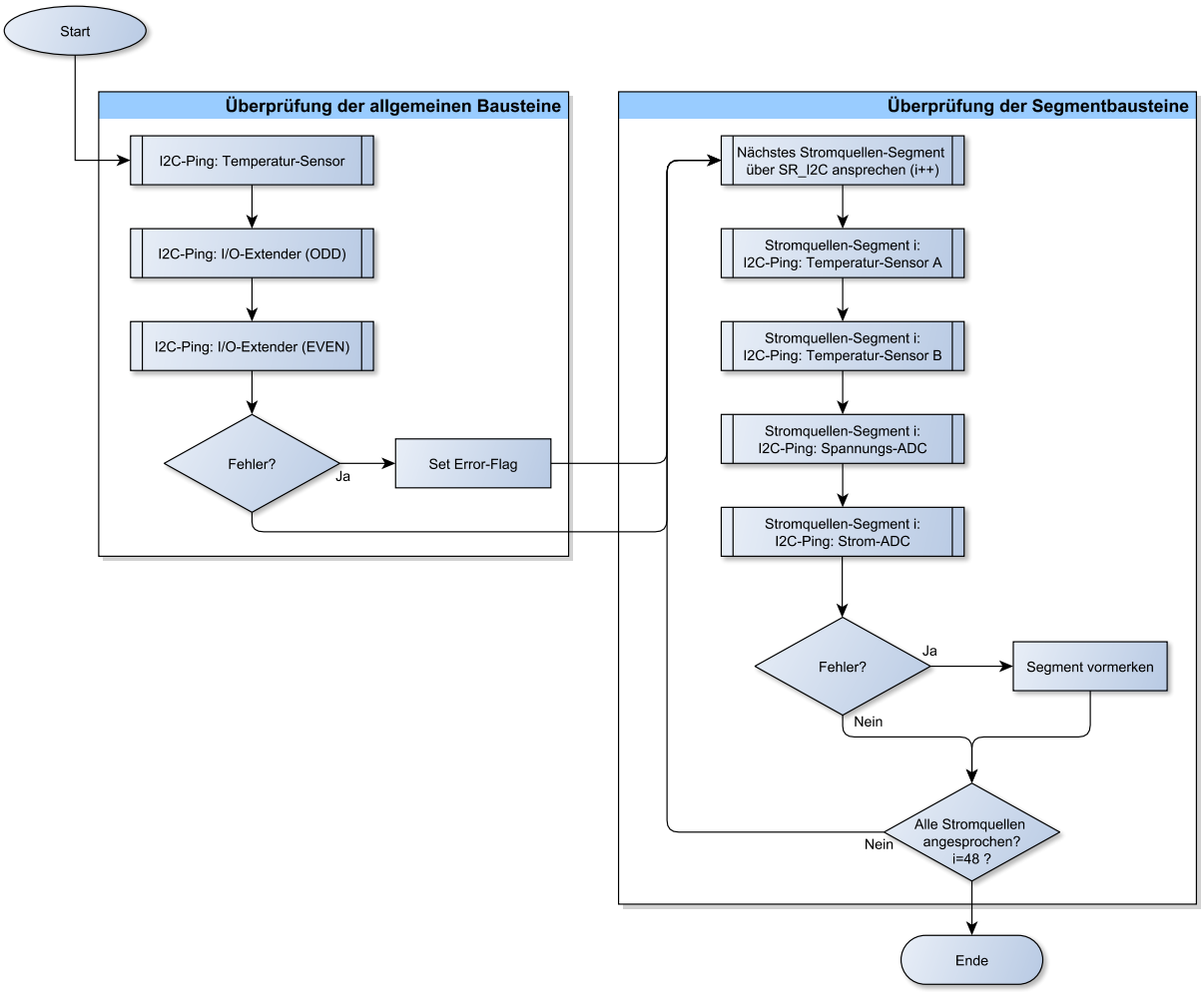

Abbildung 5.3: Ablaufdiagramm für den I2C Test

5.2 Interrupts

Interrupts sind Unterbrechungen des ursprünglichen Programmablaufs und können zum Reagieren auf Ereignisse (von „Außen“) eingesetzt werden. Heutzutage bieten so gut wie alle Mikrocontroller die Möglichkeit die unterschiedlichsten Interrupts zu verwenden. Diese reichen von Fehlerbehandlung (zum Beispiel Notabschaltung bei zu hoher Temperatur) bis zur Abwicklung von Kommunikation. Dabei können die Interrupts nach Priorität geordnet und durch den Interrupt-Controller des Prozessors koordiniert werden.
Wird ein Interrupt ausgelöst, so wird die zum Interrupt dazugehörige Flag im Interrupt-Flagregister gesetzt. Der Mikroprozessor speichert daraufhin sein Steuerregister auf den Stack und beginnt mit der Abarbeitung der Interrupt-Serviceroutine (ISR). Gleich zu Beginn der ISR sollte die Interrupt-Flag gelöscht werden. Wird auf das Zurücksetzen der Flag vergessen, löst der Interrupt nach Beendigung der

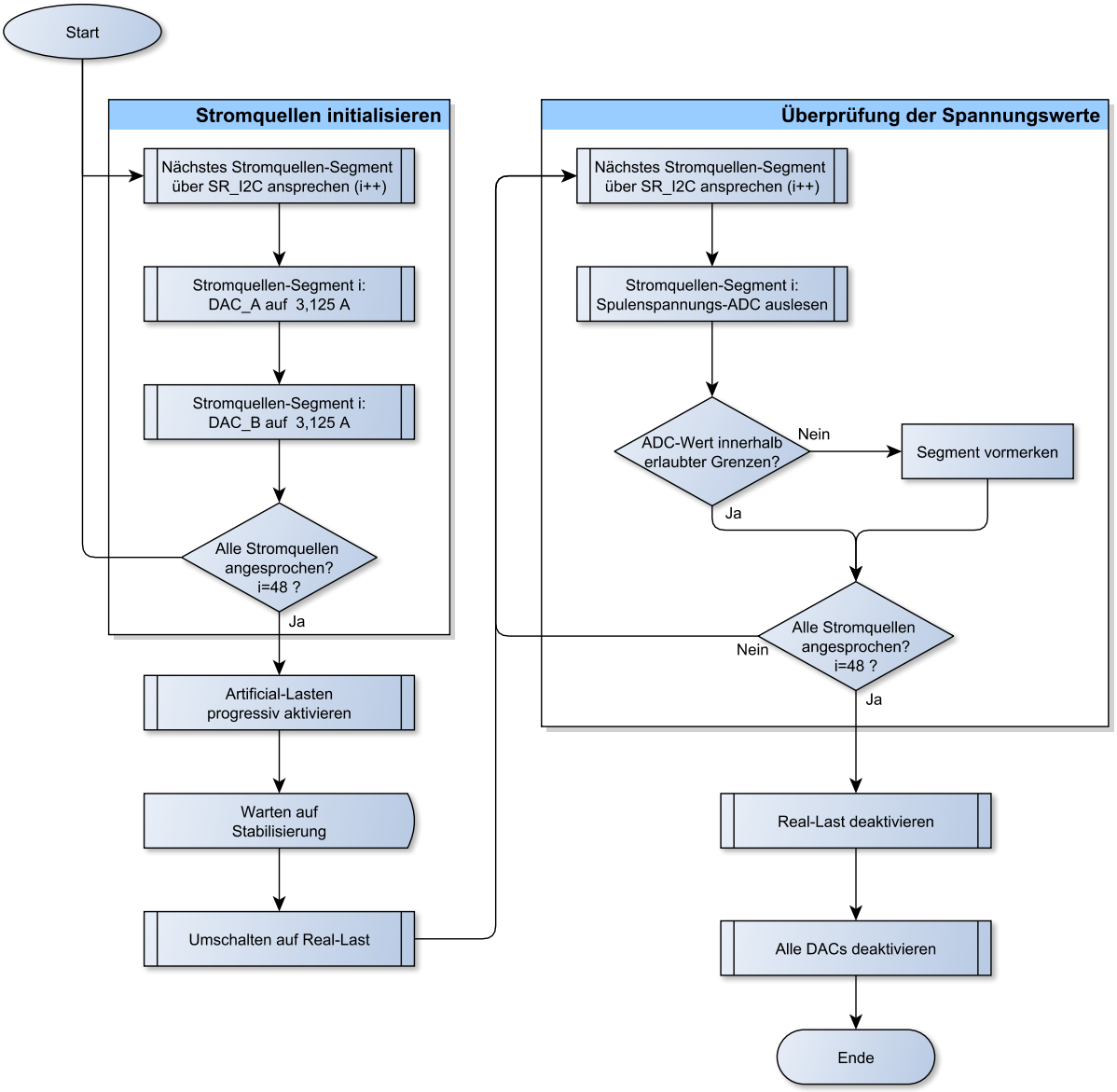

Abbildung 5.4: Ablaufdiagramm für den Real-Last Test

ISR sofort wieder aus. Nach Abschluss der ISR wird das alte Steuerregister vom Stack geladen und der Ablauf des Hauptprogramms wird fortgesetzt. Das Hauptprogramm weiß grundsätzlich nichts von der Unterbrechung durch den Interrupt, kann aber durch Manipulation von Registern und Speicher durch die ISR beeinflusst werden.
In diesem Projekt gibt es drei Interruptquellen (UART-Empfang, Alarm, Trigger), welche auf zwei unterschiedliche Interrupts (UART-Empfangsinterrupt, ICN Interrupt) verteilt sind. Der ICN Interrupt hat durch seine Funktion als Wächter die höhere Priorität und kann je nach Konfiguration den UART-Empfangsinterrupt

unterbrechen.

5.2.1 UART-Empfangsinterrupt

Der UART-Empfangsinterrupt (Abb. 5.5) wird ausgelöst, sobald ein Byte über die UART-Schnittstelle empfangen wurde. Dieses Byte muss entsprechend dem unten beschriebenen Protokoll mit dem Code für das gewählte Kommando beginnen. Nachdem das Kommandobyte im Hauptspeicher abgelegt wurde, wird es gemäß Tab. 5.1 dekodiert. Je nach Instruktion kann weitere Kommunikation mit dem PC notwendig sein. Sollte der Fehlerstatus des Systems abgefragt werden, fungiert der Controller als Sender. In allen anderen Fällen ist er der Empfänger. Die so erhaltenen Daten werden dabei zur späteren Verarbeitung im Speicher hinterlegt.
Die Kommunikation wird, sofern kein valides Kommando erhalten wurde, durch das Senden der negativen Bestätigungsnachricht ($NACK$) terminiert. Liegt kein Fehler vor, wird dies durch die positive Bestätigungsnachricht (ACK) signalisiert und die NEW-CMD-Flag wird gesetzt.
Nach Löschen der Interrupt-Flag wird die ISR verlassen und das Hauptprogramm setzt an der Stelle fort, an der es unterbrochen wurde.

Befehlsliste

Code	*Bezeichnung*	*Beschreibung*	*Nachricht, Antwort in Byte*
0x01	START	Startet den zuvor gewählten Betriebsmodus.	2|2
0x02	STOP	Beendet den laufenden Betriebsmodus.	2|2
0x03	DAC-Values	Ändert die Stromwerte der einzelnen Spulen.	194|2
0x04	SM	Static Mode wird ausgewählt.	8|2
0x05	CM	Conventional Mode wird ausgewählt.	24|2
0x06	TWM	Traveling Wave Mode wird ausgewählt.	24|2
0x07	Trigger	Trigger Mode wird getoggled. (default: off)	2|2
0x08	Real-Last Test	Real-Last Test wird durchgeführt.	2|2
0x09	Check TMP101	Alarmzustand der Stromquellen-Segment Temperatursensoren wird ausgelesen.	2|2
0x0A	Error-Status	Fehlerstatus wird an die GUI übertragen.	2|14

Tabelle 5.1: Übersicht der UART-Befehle

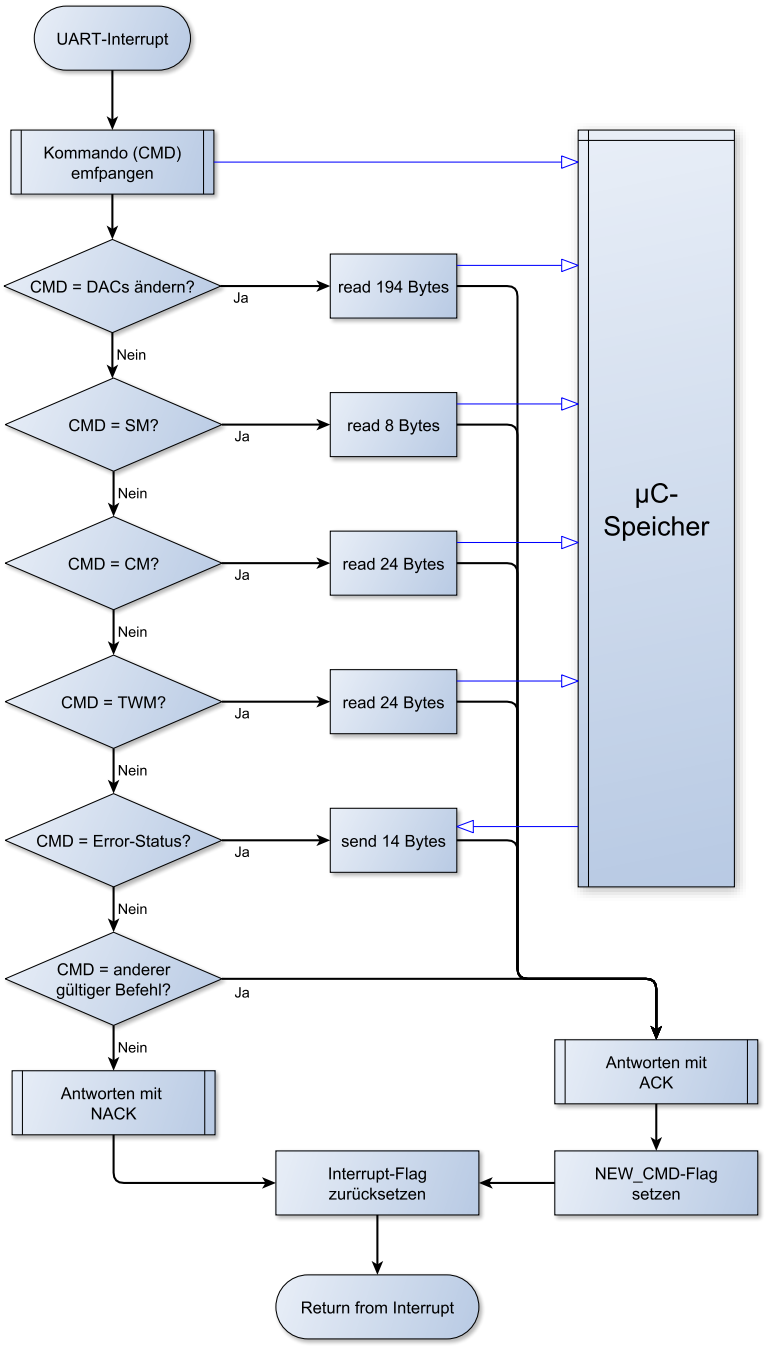

Abbildung 5.5: Ablaufdiagramm für den UART-Empfangsinterrupt

Protokoll

Das Protokoll für die UART-Kommunikation zwischen PC und Mikrocontroller besteht immer aus einer Nachricht (*MSG*), die die Kommunikation einleitet, und einer darauffolgenden Antwort (*ANS*). Bis auf die nach dem Systemstart gesendeten Version- und Fehlerstrings gehen alle Nachrichten von der GUI-Software aus, wobei immer mit dem höchstwertigen Bit (MSB) gestartet wird. Sowohl jede Nachricht als auch Antwort beginnt mit dem Kommando (*CMD*) in Form des Codes aus Tab. 5.1 und endet mit der Checksum. Die Checksum dient zur Sicherung der Datenintegrität und besteht aus einem Byte, welches mittels *XOR*-Verknüpfung aller in der Nachricht enthaltenen Bytes (exklusive Checksum-Byte) konstruiert wird. Einzige Ausnahme dieser Regel ist die Checksum der Antwort auf die Error-Statusabfrage. Da hier die Information in der Antwort steckt, werden ihre Bytes XOR-verknüpft (*µC-Checksum*).
Im folgenden Abschnitt findet sich eine Auflistung der Bytemuster von Nachrichten und Antworten, wobei auf der x-Achse die Bytes aufgetragen sind. In den Abbildungen vorkommende eckige Klammern beinhalten den jeweiligen Bit-Abschnitt. Alle 2 Byte-Nachrichten und -Antworten (in Tab. 5.1) werden durch das jeweilige Standardkommando-Muster abgedeckt. Eine Erklärung der in den Abb. 5.6 - 5.10 vorkommenden Byte-Blöcke ist in Tab. 5.2 zu finden.

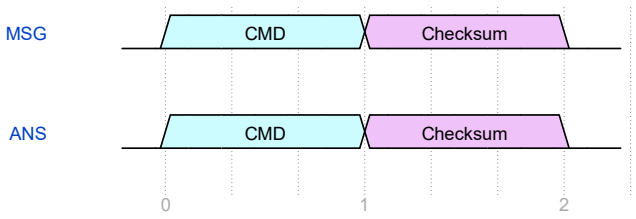

Abbildung 5.6: Bytemuster: Standardkommandos

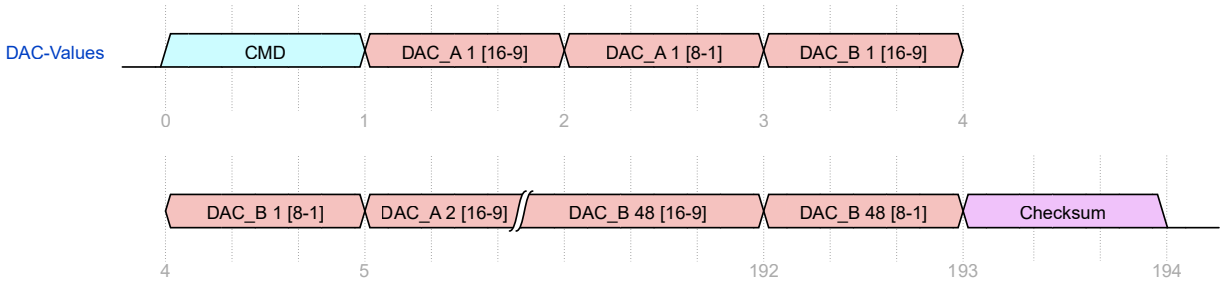

Abbildung 5.7: Bytemuster: Nachricht zur Veränderung der DAC-Werte

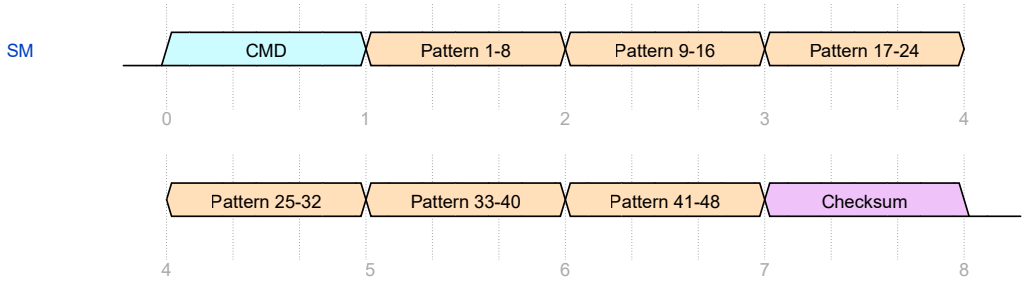

Abbildung 5.8: Bytemuster: Nachricht zur Konfiguration des Static Mode

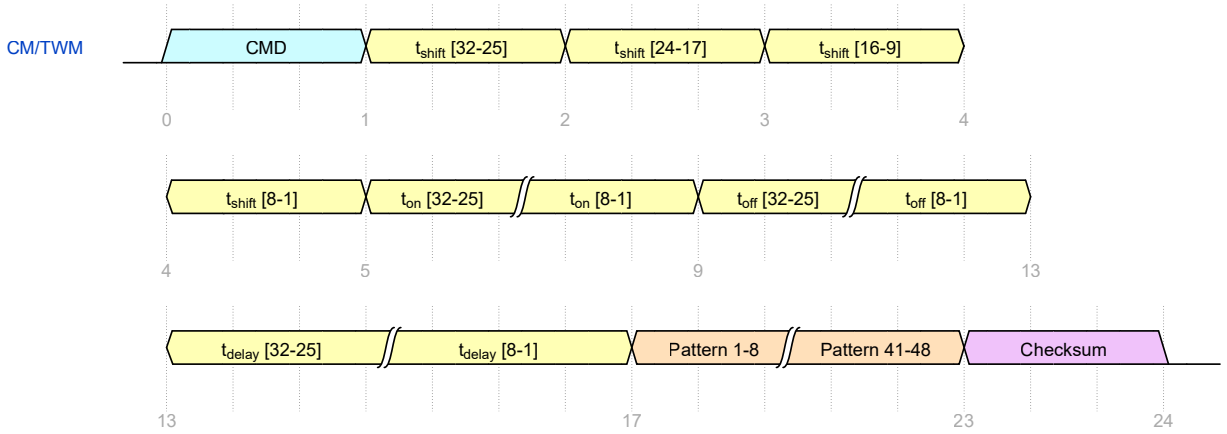

Abbildung 5.9: Bytemuster: Nachricht zur Konfiguration des Conventional Modes oder des Traveling Wave Modes, kodiert im jeweiligen Kommando-Byte *CMD*

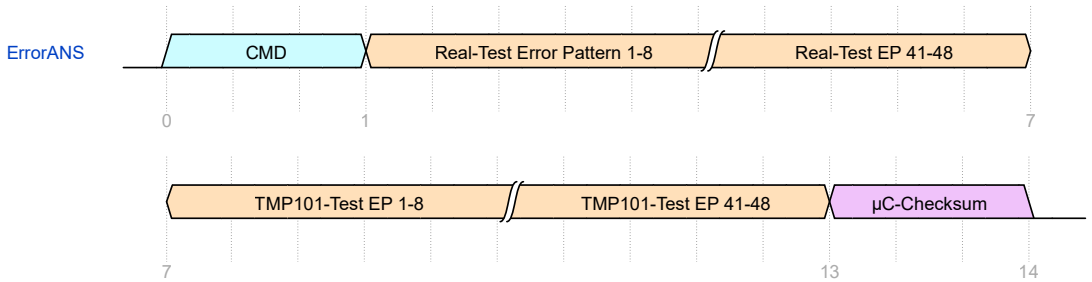

Abbildung 5.10: Bytemuster: Antwort der Fehlerstatusabfrage

Bezeichnung	*Erklärung*
CMD	Code des Befehls (siehe Tab. 5.1)
Checksum	XOR-Verknüpfung aller Bytes
DAC_X	Digitaler Wert, der an den DAC geschrieben werden soll.
Pattern	Beinhaltet das Muster für die Betriebsmodi, beginnend mit dem ersten ins Register zu schiebenden Bit.
t_{shift}	Schiebeperiode in Instruktionen-Zyklen T_I
t_{on}	Zeit (in Instruktionen-Zyklen T_I) in der das Muster auf die Spulen geschaltet wird. (Offset von +6, siehe Abschnitt 5.3.2, Bits 15 und 16 werden nicht verwendet $\rightarrow$ 30 bit)
t_{off}	Zeit (in Instruktionen-Zyklen T_I), die vergehen muss nachdem das Muster aus den Registern geschoben wurde und die Spulen stromlos sind. (Offset von +6, Bits 15 und 16 werden nicht verwendet $\rightarrow$ 30 bit)
t_{delay}	Zeit (in Instruktionen-Zyklen T_I) bevor der nächste Schiebevorgang eingeleitet wird. (Offset von +6, Bits 15 und 16 werden nicht verwendet $\rightarrow$ 30 bit)
Real-Test Error Pattern	Elemente, die beim Real-Test vorgemerkt wurden, werden durch ein HIGH im Error Pattern repräsentiert.
TMP101-Test EP	Elemente, die einen Temperaturalarm ausgelöst haben, werden durch ein HIGH im Error Pattern repräsentiert.
μC-Checksum	XOR-Verknüpfung aller vom Mikrocontroller gesendeten Bytes.

Tabelle 5.2: Erklärung der Byte-Blöcke

5.2.2 Input Change Notification (ICN) Interrupts

ICN Interrupts lösen aus, sobald sich der Pegel (TTL) am dazugehörigen Pin ändert. Der Interrupt unterscheidet dabei nicht, ob ein High-to-Low oder ein Low-to-High Übergang vorliegt. Dies muss also in der ISR vom Programmierer selbst behandelt werden. Im MONOPOL-Projekt kommen zwei unterschiedliche ICN Interrupts vor, auf die im Folgenden eingegangen wird.

Trigger-Interrupt

Der Trigger-Interrupt wird ausgelöst, wenn ein High-Pegel am *Trigger_IN*-Pin registriert wird und der Trigger Mode (Unterkapitel 5.3.4) aktiviert sowie ein zulässiger Modus (CM oder TWM) selektiert ist.
Die ISR beginnt sofort mit der Erzeugung des gewählten Neutronenpakets und signalisiert den Abschluss durch einen High-Pegel am Output-Pin. Nun geht das System wieder in den Standby Modus über und wartet auf einen neuen Trigger-

Impuls beziehungsweise ein Kommando von der PC-Software. Details werden in Abschnitt 5.3.4 Trigger Mode dargelegt.

Alarm-Interrupt

Die Aufgabe des Alarm-Interrupts ist die Abschaltung des Resonators, sollte ein mit dem sicheren oder ordnungsgemäßen Betrieb in Konflikt stehendes Ereignis auftreten. Mit den aktuell verfügbaren Installationen kann so die Temperatur der Segmente, sowie der korrekte Anschluss der Resonatorspulen während der Laufzeit überwacht werden. Wichtig in der praktischen Anwendung ist, dass der Interrupt nur während des Resonatorbetriebs aktiviert wird, da die Spulenüberwachungselektronik auch bei deaktivierter Stromquelle (im Standby Modus) Alarm schlagen kann.
Der Alarm-Interrupt wird ausgelöst, sollte sich der logische Pegel am *ALARM_IN*-Pin ändern. Im Normalfall, ist der Zustand ein HIGH-Pegel, da die Spannung über Pullup-Widerstände auf 5 V gehalten wird. Durch dieses Design ist es einfach, weitere Interrupt-Quellen zu ergänzen. Geplant ist zum Beispiel die Installation eines Drucksensors, der die Dichtheit des Resonatorgefäßes überwachen soll, wenn der Resonator in einer speziellen Gasatmosphäre (zum Beispiel Helium zur Reduktion der Neutronenabsorption) betrieben wird. Ein Nachteil dieser Methode ist, dass nicht nach Grund des Interrupts differenziert werden kann. Um diese Informationen zu erhalten, ist zusätzliche Hardware notwendig. Im Falle der oben angeführten Überwachung (Temperatur und Anschluss) läuft die Auswertung über den I/O-Extender.
Tritt ein Interrupt auf, holt der Mikrocontroller die vorhandenen Informationen in der ISR ein und überführt die Ablaufsteuerung in den Standby-Modus. Nun kann der Benutzer die erhobenen Parameter über die GUI-Software auslesen und mit der Fehlerbehebung beginnen.

5.3 Betriebsmodi

Es stehen drei unterschiedliche Betriebsmodi des Resonators zur Verfügung, der *Static Mode*, der *Conventional Mode* und der *Traveling Wave Mode*. Letztere zwei können zusätzlich zum herkömmlichen im Trigger-Modus betrieben werden. Die Auswahl zwischen den unterschiedlichen Modi erfolgt über UART mittels der GUI-Software.

5.3.1 Static Mode (SM)

Der Static Mode ist der simpelste Modus und gleicht in der Funktionalität dem traditionellen Drabkin-Resonator. Er benötigt nach erfolgtem Start, sollten keine Fehler auftreten, keinerlei Einschreiten des Mikrocontrollers.

Sobald der Static Mode (Abb. 5.11) ausgewählt ist und das Start-Signal via UART registriert wird beginnt die Ablaufsteuerung die einzelnen Stromquellen nacheinander zu aktivieren. Sind alle Quellen eingeschaltet, wird der Einschwingvorgang abgewartet (etwa 680 µs).
Nun ist das Setup abgeschlossen und der eigentliche Start kann erfolgen. Dazu wird erst das gewünschte Muster ins Schieberegister der Real-Last Ansteuerung (SR_R) geladen und gelatched. Jetzt wird der Strom sowohl über die Artificial- als auch über die Real-Last geleitet (=Glitchvorgang). Nach etwa 600 ns haben sich die Ströme stabilisiert und die Artificial-Lasten können deaktiviert werden. Von jetzt an ist der Modus aktiv und der Mikrocontroller überwacht den korrekten Ablauf und wartet auf neue Befehle des PCs. Sobald ein neues Kommando registriert wird, beendet sich der SM und die Befehlsdekodierung beginnt.

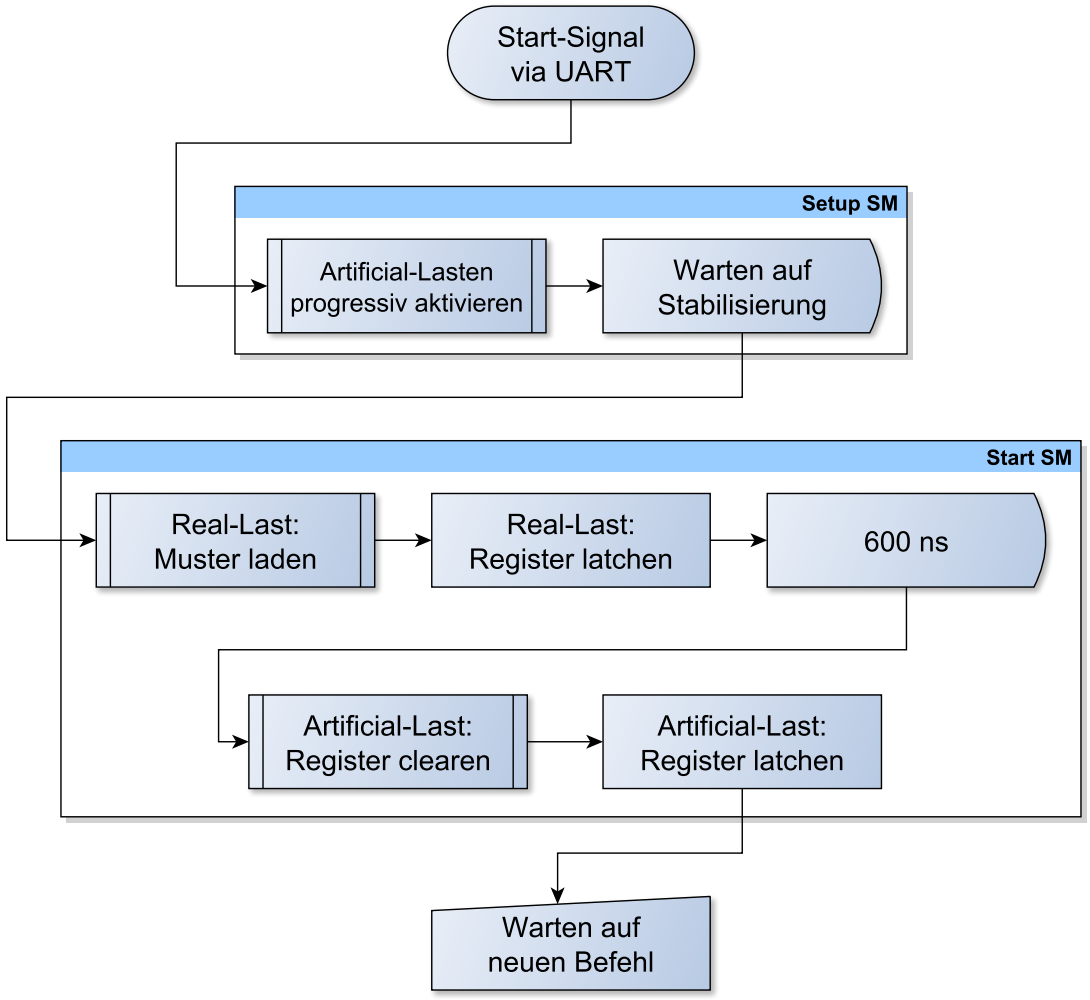

Abbildung 5.11: Ablaufdiagramm für den Static Mode

5.3.2 Conventional Mode (CM)

Im Conventional Mode (Ablaufdiagramm in Abb. 5.12) werden Neutronenpakete erzeugt, indem alle Resonatorspulen gleichzeitig ein- und ausgeschaltet werden. Die Pulsbreite des Pakets wird dabei durch die Dauer der stromführenden Phase vorgegeben.

Erfolgt ein Start-Signal nach einer vorhergehenden Auswahl des Conventional Mode, so werden die einzelnen Resonatorspulen nacheinander mit den eingestellten Strömen gespeist. Wie beim Static Mode wird erneut pausiert bis das System eingeschwungen ist. Nun wird das Glitch-Register mit dem gewünschten Muster geladen. Da im CM immer dieselben Spulen aktiviert und deaktiviert werden, ist nur ein einmaliges Laden des Glitch-Registers notwendig.
Jetzt wo der Setup-Prozess abgeschlossen ist, wird geprüft ob eine neue Übertragung vom PC angekommen ist. Falls ja, wird der CM beendet und die Befehlsdekodierung wird gestartet. Im anderen Fall wird die Hauptschleife, welche zwischen Artificial- und Real-Last umschaltet, betreten.
Umschaltvorgänge erfolgen dabei immer nach dem gleichen Schema. Das Muster wird in das mit der zu aktivierenden Last assoziierten Schieberegister geladen. Danach wird das derzeit aktive Schieberegister geleert und ein Latch-Signal wird an alle drei involvierten Register gesendet. Ist der Glitch-Vorgang abgeschlossen (600 ns), werden alle Outputs des Glitch-Registers auf Low gesetzt. Das Muster im Register bleibt dabei erhalten.
Zu Beginn der Schleife wartet das Programm die Zeit t_{delay} ab, welche die Verzögerung zwischen zwei aufeinanderfolgende Schaltvorgänge beschreibt. Anschließend wird von der Artificial- auf die Real-Last umgeschaltet. Dieser Zustand bleibt bestehen, bis die Zeit t_{on} verstrichen ist. Dann wird der Resonator durch Umschalten auf die Artificial-Last wieder deaktiviert. Ist die Zeit t_{off} vergangen, wird sofern kein neuer UART-Befehlt übermittelt wurde, mit der erneuten Abarbeitung der Schleife begonnen.

Delay-Funktion

Die *Delay-Funktion* ist ein Makro, welches Wartezeiten des Mikrocontroller von bis zu 72.82 s ermöglicht. Dies ist notwendig, um einerseits die Konversionszeit der ADCs zu überbrücken, andererseits um die unterschiedlichen Wartezeiten des CM und des TWM zu realisieren.

```
1 REPEAT #16383
2     NOP           ;1+16383 NOPs
```

Listing 5.1: Assemblercode: Wiederhole NOP-Befehl 2^{14} mal.

Der *REPEAT*-Befehl führt die nachfolgende Instruktion $N + 1$ mal aus, wobei N eine 14 bit-Zahl ist. Das größtmögliche Argument ergibt sich demnach zu $2^{14} - 1 =$

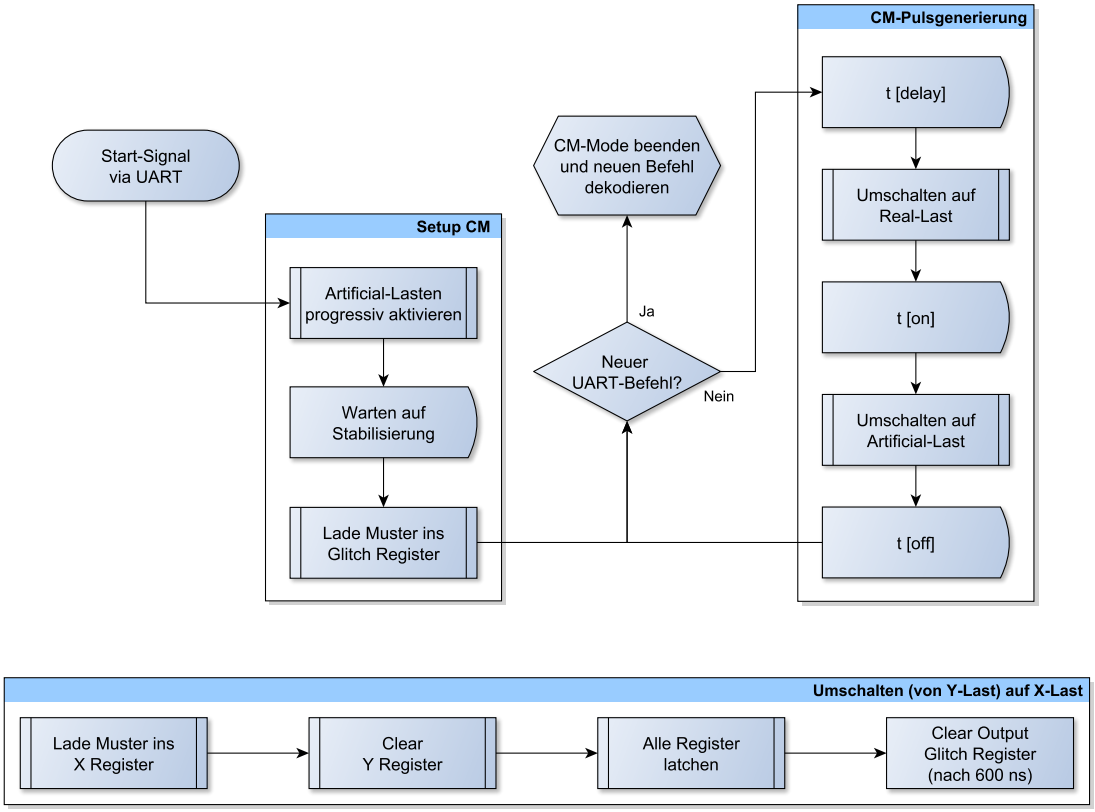

Abbildung 5.12: Ablaufdiagramm für den Conventional Mode

16383 (Listing 5.1), wodurch sich die dadurch maximal überbrückbare Zeit T_{SLEEP} (mit einer Instruktionen-Periode T_I von 67.82 ns) zu

$$T_{SLEEP} = T_I \cdot 2^{14} = 1.111\,\text{ms} \tag{5.6}$$

ergibt. Da jedoch schon alleine für die Konversionszeit der ADCs in etwa 23 ms benötigt werden, wurde das Makro *_DELAY* erstellt, zu sehen in Listing 5.2. Die Zahlen in den Kommentaren stehen für die benötigten T_I um den entsprechenden Befehl durchzuführen. Die Zahl in runden Klammern (Zeilen 9 und 18 in Listing 5.2) fällt nur an, wenn der dazugehörige *BRA*-Befehl ausgeführt wird.

```
1  .macro _DELAY K
2      ;K[eff] = K + 6, K[mind] = 6
3      MOV \K+2, MISC_REG                          ;1
4      REPEAT  MISC_REG                            ;1
5          NOP                                     ;\K+2 + 1
6
7      ;single-cycle offset: +2 if no branch else +3
8      MOV \K                                      ;1
9      BRA Z, $+6*2                                ;1 or (2)
10     ;$ = current address in program memory
```

```
11
12      NOP                                          ;1 for constant
            offset of 6
13      MOV \K, MISC_REG                             ;1
14      ;single-cycle offset until this point is +7
15      REPEAT #0x3FFB                               ;1
16          NOP                                      ;0x3FFA + 1
17      DEC MISC_REG, MISC_REG                       ;1
18      BRA NZ, $-3*2                                ;1(2)
19  .endm
```

Listing 5.2: Assemblercode des Delay-Makros

Durch die deterministische Natur von Assembler und die Kenntnis der Instruktionen-Periode T_I kann der Code nun auf unterschiedliche 30 bit-Argumente (=16 bit Arbeitsregister + 14 bit REPEAT-Befehl) für K untersucht werden. Der Parameter K steht für die zu wartenden Instruktionen-Perioden.

Übergibt man einen Wert von 0 ergibt sich dennoch eine Verzögerung von $6 \cdot T_I$, da die Instruktionen bis Zeile 9 (Listing 5.2) unabhängig vom Argument abgearbeitet werden müssen.
Ist der 30 bit-Wert ungleich 0, wartet das Makro $(K + 6 \cdot T_I)$ Sekunden. In Zeile 13 wird mit $(2^{14} - 1) - 4$ (dargestellt in Hexadezimal) initialisiert, um die tatsächliche Wartezeit möglichst nahe an „$K \cdot T_I$“ heranzubringen.

$$T\_{DELAY}(0) = 6 \cdot T_I = 406.92\,\text{ns}, \tag{5.7}$$

$$T\_{DELAY}(K) = (K + 6) \cdot T_I, \text{ mit } K \,\epsilon\, [1 \,..\, 2^{30} - 1], \tag{5.8}$$

$$T\_{DELAY}(2^{30} - 1) = (2^{30} - 1 + 6) \cdot T_I = 72.82\,\text{s}. \tag{5.9}$$

5.3.3 Traveling Wave Mode (TWM)

Im Traveling Wave Mode erfolgt ähnlich wie im Conventional Mode ein periodisches Ein- und Ausschalten der Resonatorspulen, wobei hier nicht alle Elemente kollektiv, sondern individuell geschaltet werden. Dadurch können die Anstiegs- und Fallzeiten der Neutronenpulse deutlich reduziert werden (bei 48 Resonatorspulen im Idealfall auf $\frac{1}{48}$-stel, Vergleich in Abb. 1.2).

Der Betriebsmodus (Abb. 5.13) wird gestartet, sobald ein Start-Signal empfangen wird. Neben dem schrittweisen Aktivieren der Spulenströme, was allen Modi gemein ist, wird im TWM zusätzlich eine Variable (R_{OLD}) angelegt sowie ein 32 bit-Timer initialisiert. Die Variable speichert den vorhergegangen Wert, der ins SR_R eingeschoben wurde. Dies ist notwendig, um das Glitchregister ordnungsgemäß zu schalten. Ein Glitch tritt immer nur dann auf, wenn der alte Wert ungleich dem neuen Wert ist:

$$G = R_{OLD} \oplus R. \tag{5.10}$$

Der Timer wird mit der Schiebeperiode t_{shift} (in Instruktionenzyklen) initialisiert und steuert den zeitlichen Ablauf der Schiebevorgänge. Die 32 bit ermöglichen eine Schiebeperiode von bis zu 291.28 s.
Nach durchgeführtem Setup wird die Abfrage neuer UART-Befehle identisch wie im CM abgehandelt. Sollte kein neues Kommando vorliegen, wird die Hauptschleife des TWM gestartet. Sobald die Wartezeit t_{delay} abgelaufen ist, wird der Timer gestartet. Nun wird das gewünschte Muster nach und nach, abhängig von der Schiebeperiode, eingeschoben. Ist dieser Schritt abgeschlossen wird die Zeit t_{on} gewartet. Anschließend wird der Timer zurückgesetzt und das Muster wird aus den Registern geschoben. Bevor die Schleife von neu beginnen kann, wird noch t_{off} abgewartet.

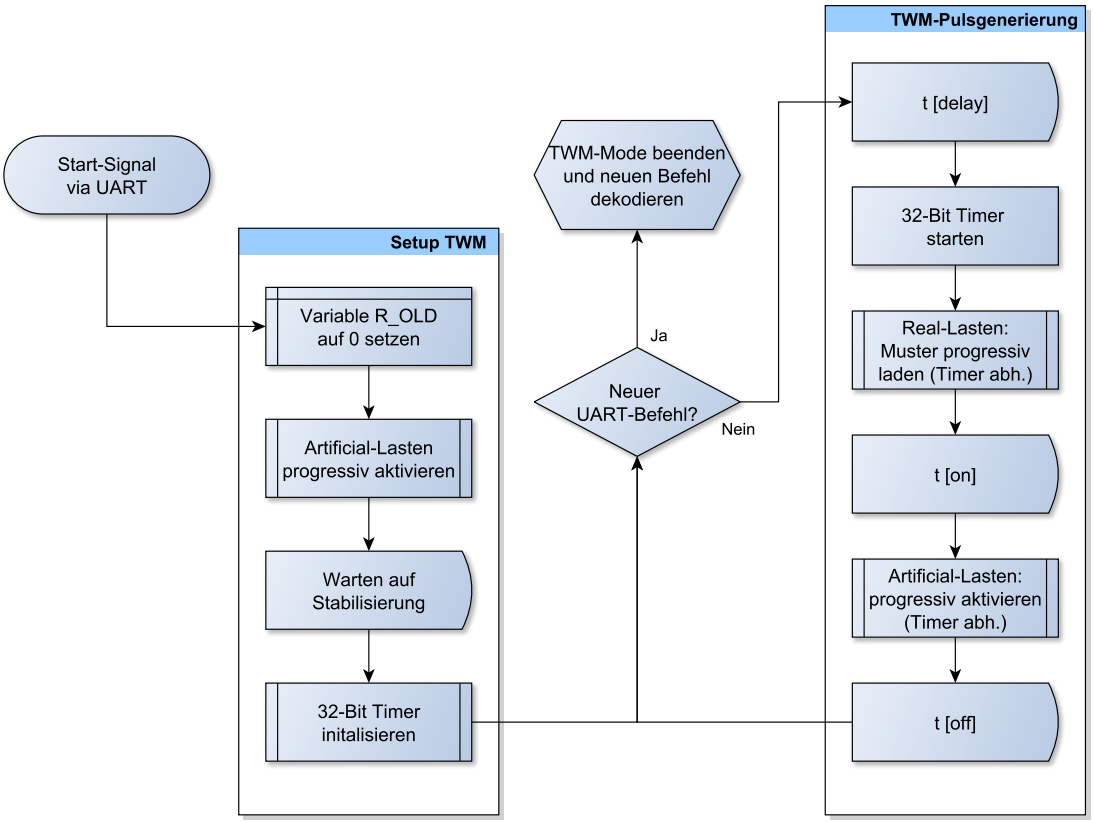

Abbildung 5.13: Ablaufdiagramm für den Traveling Wave Mode

Schiebeperiode

Die minimale Schiebeperiode begrenzt die minimale Pulsbreite und die Geschwindigkeit der gefilterten Neutronen. Um ein möglichst großes Spektrum abzudecken, ist die Schiebeperiode kurz zu halten. Durch Analyse des verwendeten Codes (zu sehen in Listing 5.3) kann man bei Kenntnis der Instruktionen-Periode T_I die theoretische Dauer eines Schiebevorganges berechnen. Dabei benötigt jeder Befehl (im Listing dunkelblau und fett gedruckt) eine Periode, mit Ausnahme der

BRA-Befehle. Diese benötigen, sollte der Sprung im Programmcode stattfinden, zwei Instruktionen-Perioden. Die notwendigen T_I um die Makros (gekennzeichnet durch„_“ am Beginn des Namens) abzuarbeiten, stehen in Zahlenform als Kommentar neben dem Makroaufruf.
Nach Abzählen kommt man auf $39 \cdot T_I$ für die innere, und $44 \cdot T_I$ für die äußere Schleife. Dieser Unterschied kann durch eine extra Sprungmarke und 3 NOP-Befehle kompensiert werden, sodass sich die minimale Schiebeperiode T_S zu

$$T_S = 44 \cdot T_I = 2.984\,\mu s. \tag{5.11}$$

ergibt. Längere Schiebeperioden können durch den verwendeten 32 bit-Timer in T_I-Schritten bis zu einer maximalen Schiebeperiode von

$$T_{S,max} = (2^{32} - 1) \cdot T_I = 291.28\,s, \tag{5.12}$$

realisiert werden.

```
1  TWM_SHIFT_IN_OUTERLOOP:
2
3      MOV #16, MISC_REG
4      MOV MISC_REG, COUNTER_16
5      MOV [PTR_REG++], PATTERN_REG
6      TWM_SHIFT_IN_INNERLOOP:
7
8          1: BTST IFS0, #T3IF      ;wait until 32 bit timer interrupt
               flag is set (K1)
9              BRA Z, 1b
10         BCLR IFS0, #T3IF
11
12         BTSS PATTERN_REG, #15
13             BRA TWM_SHIFT_BIT_LOW
14
15     TWM_SHIFT_BIT_HIGH:      ;jumpmark only for better visibility
16         ;DIN_G <= R XOR R_OLD
17         BTSS R_OLD, #0 ; if R_OLD == HIGH then NOP
18             BSET DIN_G ; else: set DIN_G = HIGH
19
20         BTSC R_OLD, #0 ; if R_OLD == LOW then NOP
21             BCLR DIN_G ; else: set DIN_G = LOW
22
23         BSET R_OLD, #0
24
25         ;set DINs for R = HIGH
26         BSET DIN_R  ;do DIN_X sets with a mask to save 1 single-cycle
27         BCLR DIN_A
28
29         BRA TWM_DO_SHIFT
30
31     TWM_SHIFT_BIT_LOW:
32         BTSS R_OLD, #0 ; if R_OLD == HIGH then NOP
```

```
33              BCLR DIN_G  ;else:  set  DIN_G = LOW
34
35          BTSC R_OLD, #0  ;if R_OLD == LOW then NOP
36              BSET DIN_G  ;else:  set  DIN_G = HIGH
37
38          BCLR R_OLD, #0
39
40          ;set DINs for R = LOW
41          BCLR DIN_R
42          BSET DIN_A
43
44      TWM_DO_SHIFT:
45          _CLK_ALL_wW0     ;6
46          _LATCH_ALL_wW0   ;6 (4 for 600 ns delay)
47
48          _SLEEP_CYCLES 5 ;wait 600 ns until CLROUT_G (8.8 SC - 1 _SLEEP
                )
49
50          _CLROUT_G        ;5
51
52          RLNC PATTERN_REG, PATTERN_REG              ;rotate right
53      DEC COUNTER_16
54      BRA NZ, TWM_EXTRA_JMP    ;BRA TWM_SHIFT_IN_INNERLOOP
55      ;39
56      ;difference beteween inner and outer shift is 5 SC - can be
            compensated with extra jump and NOPs
57  DEC COUNTER_3
58  BRA NZ, TWM_SHIFT_IN_OUTERLOOP
59  ;44
```

Listing 5.3: Assemblercode um das Muster in die Register einzuspielen.

5.3.4 Trigger Mode

Der Trigger Mode (Abb. 5.14) ist eine Erweiterung von CM und TWM. Er ermöglicht das Erzeugen von einzelnen Neutronenpaketen, abhängig von einem externen Spannungssignal. Die Erzeugung erfolgt je nach gewählten Einstellungen entweder durch das CM- oder das TWM-Schema.

Der Modus teilt sich in zwei voneinander getrennte Teile (mit UART-Empfang sogar drei) ein. Anders als bei den anderen Betriebsmodi läuft das Setup nicht unmittelbar vor der Pulsgeneration ab, sondern wird eingeleitet, sobald die Trigger-Flag durch einen UART-Empfangsinterrupt (Deaktivierung durch erneutes Senden) gesetzt wurde. Dabei werden die einzelnen Ströme über die Artificial-Lasten aktiviert. Danach geht das System in einen Idle-Zustand über und wartet auf neue Anweisungen.
Wird ein Signal von der externen Trigger-Elektronik registriert, wird ein ICN-Interrupt (Abschnitt 5.2.2) ausgelöst. Da der ICN-Interrupt nur eine Servicerouti-

ne für die unterschiedlichen Quellen anbietet, muss zunächst überprüft werden, ob wirklich ein Trigger-Interrupt vorliegt. Ist das der Fall wird vom Speicher das zu verwendende Erzeugungsschema abgefragt. Bevor das Paket generiert wird, wird die Peripherie über ein Trigger-Out Signal verständigt. Jetzt wird, je nach Einstellung, die Paketgenerierung, wie in Abschnitt 5.3.2 CM beziehungsweise Abschnitt 5.3.3 TWM beschrieben, durchgeführt. Danach geht das System erneut in einen Idle-Zustand und wartet auf neue Instruktionen, entweder durch den Benutzer oder den Trigger.

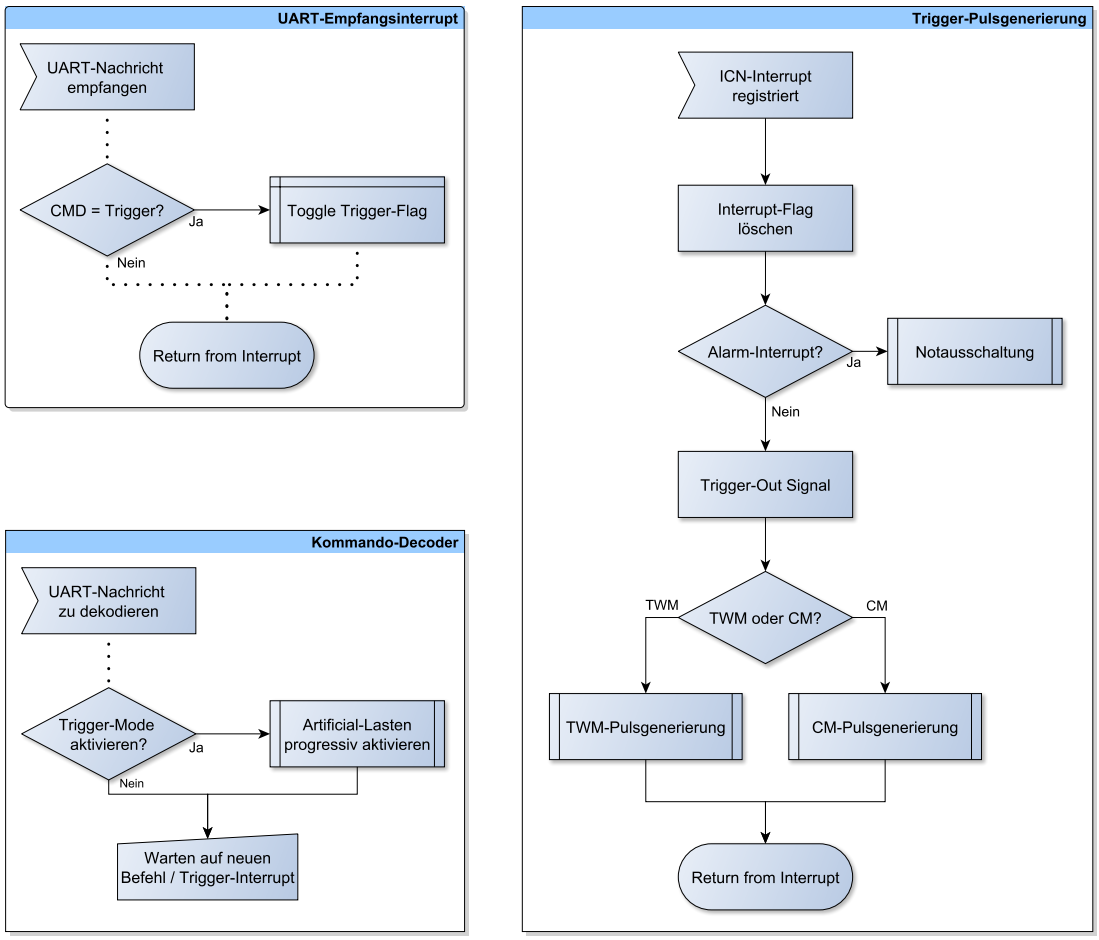

Abbildung 5.14: Ablaufdiagramme für den Trigger Mode

5.4 Software zur Überprüfung der Stromquellen-Segmente

Um die vollständige Funktion der einzelnen Stromquellen-Segmente zu gewährleisten, wurde eine eigenständige Testsoftware entwickelt, deren Ablaufdiagramm in

Abb. 5.15 zu sehen ist. Die blauen Pfeile symbolisieren immer das Ablegen der erhaltenen Messwerte im Arbeitsspeicher des Mikrocontrollers. Mit dieser Software können folgende Funktionen getestet werden:

- Einstellen der Stromstärke über DAC_A (0 A bis 5 A)
- Einstellen der Stromstärke über DAC_B (0 A bis 25 A)
- Ermittlung der Temperatur über TMP101 im Bereich des MOSFET für DAC_A
- Ermittlung der Temperatur über TMP101 im Bereich des MOSFET für DAC_B
- Ermittlung des Gesamtstromes mittels Hall-Sensor und ADC
- Ermittlung der Resonatorspulenspannung mittels ADC
- Aktivieren und Umschalten der Artificial- und Real-Last
- Alarm-Signal bei fehlender Verbindung zu den Lasten

Ablaufbeschreibung: Stromquellen-Test

Nachdem der Mikrocontroller mit der Testsoftware programmiert wurde, beginnt die Initialisierung des Systems. Sie läuft wie in Abb. 5.2 dargestellt ab.

Die Testschleife beginnt mit dem Deaktivieren beider Stromquellen, sowie dem Auslesen beider Temperatursensoren. Danach werden die ADCs im stromlosen Zustand ausgewertet, wobei immer auf die Einhaltung der dazugehörigen Konversionszeiten geachtet werden muss.
Nun wird DAC_A für den Strombereich von 0 A bis 5 A auf der Artificial-Last getestet. Dazu wird der z-Wert 2560 übertragen, welcher gemäß Gl. (3.3) in einen Strom von 3.125 A übersetzt wird. Da jetzt eine Stromquelle auf einer Last aktiv ist, kann die Hardware der Alarmfunktion überprüft werden. Danach wird der Strom über die ADCs ausgewertet. Falls wie vorgesehen ein Breakpoint gesetzt wurde, pausiert die Testsoftware, bis sie vom Benutzer fortgesetzt wird.

Nach dem Fortsetzen wird die zweite Stromquelle (DAC_B) auf Funktion getestet. Es wird der digitale Wert $z = 512$ übertragen, welcher ebenfalls gemäß Gl. (3.3) in eine Stromstärke von 3.125 A übersetzt wird. Nachdem auf die Real-Last umgeschaltet wird, wird erneut ein Alarmtest durchgeführt. Zum Abschluss dieses Testabschnitts werden noch die ADCs ausgelesen und deren Werte im Hauptspeicher hinterlegt. Sofern gesetzt, pausiert das Programm beim Breakpoint.

Danach werden alle Stromquellen ausgeschaltet und der dazugehörige Alarmwert ermittelt. Der letzte Breakpoint dient, falls weitere Segmente getestet werden sollen, zum Austauschen der Hardware oder zum Beenden des Testablaufs.

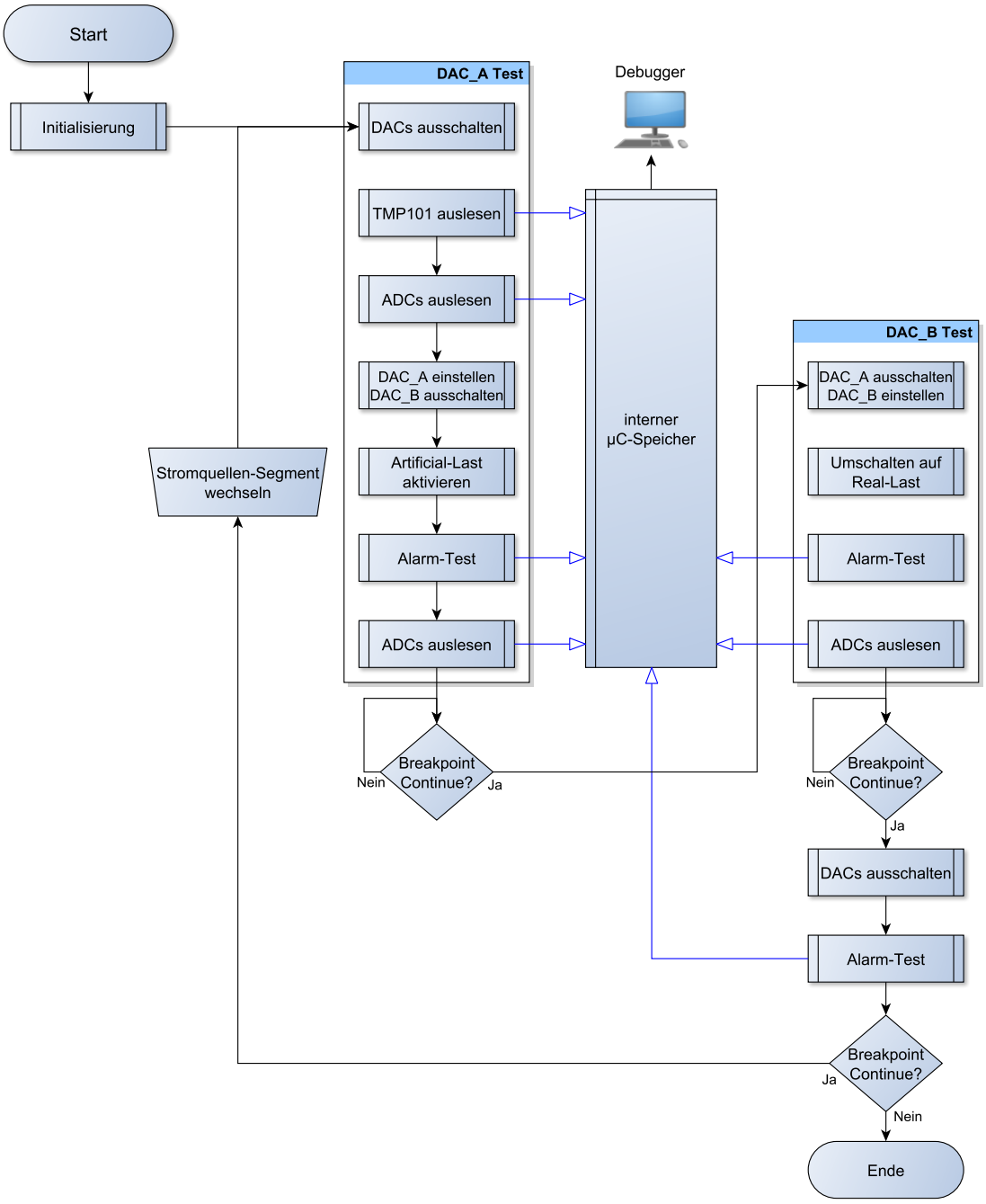

Abbildung 5.15: Ablaufdiagramm des Funktionstests eines Stromquellen-Segments

6 Tests, Experimente und Fazit

Diese Kapitel beginnt mit einer Auflistung der ermittelten Hardwaredefekte samt Behebungsstatus. Anschließend folgt eine Charakterisierung der verwendeten Stromquellen, sowie eine Beschreibung der durchgeführten Tests inklusive Interpretation der Messergebnisse.

6.1 Allgemeine Hardwareüberprüfung

Die MONOPOL-Hardware ist umfangreich, wodurch sich Fehler in der Fertigung und Handhabung nicht immer vermeiden lassen. Um die Hardware kennenzulernen und den Ist-stand zu ermitteln, wurde eine allgemeine Hardwareüberprüfung durchgeführt. Tab. 6.1 zeigt eine Auflistung, samt Behebungsstatus, der so ermittelten Fehler.

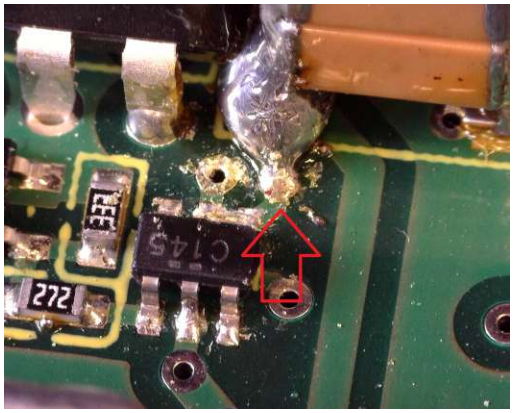

Abbildung 6.1: Segment-ID 4: Kurzschluss zwischen 12 V Versorgung und 5 V TTL-Baustein (Bild von Andrzej Pelczar)

6.2 Überprüfung der Stromquellen-Segmente

6.2.1 Charakterisierung der z/I-Kurve der DACs

Dieser Abschnitt behandelt die Kennlinien der Stromquellen, welche über die DACs gesteuert werden. Dazu wurde mittels der Messschaltung in Abb. 6.2 (Amperemeter: *FLUKE 179*) die Beziehung zwischen eingestelltem digitalen Wert z und daraus resultierendem Strom I für drei zufällig ausgewählte Stromquellen-Segmente (IDs:

Beschreibung	*Ort*	*Status*
Beim Übergang von der Zentralen-Steuerplatine zur Treiber-Platine wurden die Schieberegister-Pins von Artificial und Real Register vertauscht. Dies konnte behoben werden, indem die jeweiligen Control-Pins über die Software ebenfalls vertauscht wurden.	Zentrale-Steuerplatine, Treiber-Platine	behoben
Die Feedbackleitung des SR_I2C (DBL_I2C) ist auf der Zentralen-Steuerplatine unterbrochen. Vermutlicher Auslöser ist ein Widerstand, der entweder fehlerhaft verlötet oder vergessen wurde.	Zentrale-Steuerplatine	ausständig
Zwei der Ausgänge eines Bufferbausteins auf der Treiber-Platine waren über einen Kurzschluss verbunden. Dadurch waren Artificial- und Real-Last eines der Segmente immer gleichzeitig aktiviert.	Treiber-Platine	behoben
Auf beiden I2C-Backplanes (obere BP) wurden beim Bufferausgang der Leitung A_ON_nOFF entweder falsche oder gar keine Widerstände verlötet. Da diese Leitungen redundant ausgeführt sind, ist eine Behebung nicht unbedingt notwendig.	I2C-Backplane	ausständig
Auf beiden I2C-Backplanes wurden Inverter-ICs verkehrt eingelötet. Dadurch konnten die Stromquellengruppen (DAC_A, DAC_B) nicht individuell de- und aktiviert werden.	I2C-Backplane	behoben
Einer der Kondensatoren auf einem Stromquellen-Segment (ID: 4) wurde falsch eingelötet (Abb. 6.1), wodurch ein Kurzschluss zwischen Inverter-Baustein und 12 V Versorgungsspannung entstand. Infolge dieser Überspannung kam es zur Beschädigung von 20 Segmenten.	Stromquellen-Segment	behoben
Nach Ausmessen konnte festgestellt werden, dass 20 Schutzdioden defekt sind und getauscht werden müssen, bevor die Stromquellen wieder verwendet werden können.	Stromquellen-Segmente	behoben
Weitere Tests zeigten, dass 20 Inverter-ICs, zuständig für die Notausschaltung der Stromquellen, durch die 12 V Überspannung beschädigt wurden.	Stromquellen-Segmente	behoben
Bei beiden ADCs, welche Strom und Spannung auf den Segmenten messen, waren falsche Widerstände verbaut, weswegen diese immer in eine Richtung ausschlugen. Nach Neuberechnung der Widerstandswerte wurden diese auf allen 48 Stromquellen-Segmenten getauscht.	Stromquellen-Segmente	behoben

Tabelle 6.1: Ermittelte Hardwaredefekte

Beschreibung	*Segment-ID*	*Status*
I^2C-Kommunikation war nicht möglich, da ein Kurzschluss zwischen SCL und GND auf einem I^2C-Buffer vorhanden war.	1	behoben
Der ADC zum Messen des Stromes konnte nicht verwendet werden, da am vorgeschaltenen Operationsverstärker ein Kurzschluss bestand.	2	behoben
Die Detektion zur Ermittlung einer defekten Spule funktionierte nicht, da eine Diode in der Detektions-Elektronik fehlerhaft gelötet war.	5	behoben
Der Jumper-Widerstand, der die separaten GND-Leitungen (GND_PWR, GND_5V) verbindet, war beschädigt. Dieser Schaden entstand vermutlich, als die Verbindung zu GND_PWR während des Betriebs unterbrochen wurde. Künftig sollte dies kein Problem darstellen, da ein DC/DC-Konverter (Abschnitt 3.5.4) installiert wurde.	14	behoben
Ein Stützkondensator fehlte. Dies beeinträchtigte die Funktion durch größeren Einfluss von Temperaturschwankungen.	15	behoben
Ein Widerstand der Hall-Sensor Auswertung war nicht ordnungsgemäß gelötet. Dadurch lieferte der dazugehörige ADC immer den maximalen Ausgangswert.	19	behoben
Der DAC-Baustein für die 25 A-Stromquelle war verkehrt verbaut.	20	behoben
Der MOSFET, der den Strom über die Real-Last aktiviert, hatte einen Kurzschluss zwischen *Drain* und *Source*.	21	behoben
Ein Inverter-Baustein, der für die Ansteuerung der LEDs benötigt wird, war durchgebrannt.	22	behoben
Die Spannungsmessung mittels ADC funktioniert nicht. Vermutlich liegt das Problem beim ADC-Baustein oder dem dazugehörigen Operationsverstärker.	26	ausständig
Ein Kurzschluss am DAC_B führte dazu, dass die Stromquelle immer den maximal möglichen Strom von 25 A lieferte.	33	behoben
Ein Kurzschluss an einem OPV der Spulen-Spannungsmessung verhinderte die ordnungsgemäße Funktion.	33	behoben
Die Stromquelle von DAC_B funktionierte nicht, da ein Widerstand nicht richtig verlötet war. Dadurch war die Ansteuerung des MOSFET unterbrochen.	36	behoben
Gleich wie bei Segment-ID 14.	41	behoben
Ein I^2C-Buffer war nicht ordentlich verlötet, wodurch keinerlei Kommunikation mit dem Stromquellen-Segment möglich war.	46	behoben

Tabelle 6.2: Ermittelte Hardwaredefekte der Stromquellen-Segmente

3, 19, 44) gemessen. Da zum Messzeitpunkt noch keine ausreichende Kühlung installiert war, wurde, um Beschädigungen durch Überhitzen zu verhindern, der Maximalstrom auf 8 A begrenzt. Die so erhaltenen Messwerte sind in den Tab. 6.3 und 6.4 angeführt. Zusätzlich zur Messung mittels Amperemeter wurden die digitalen Werte z_I über den jeweiligen ADC samt Hall-Sensor erhoben.

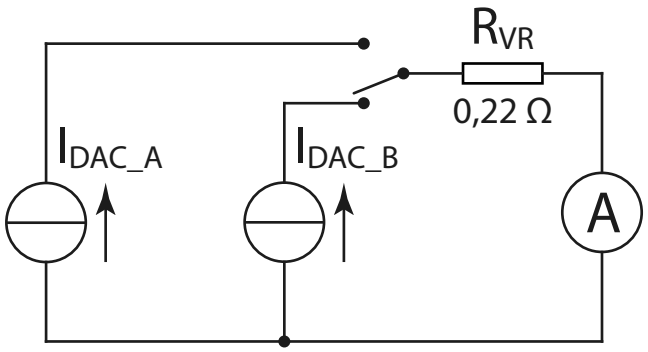

Abbildung 6.2: Messschaltung zur Überprüfung der z/I-Kurve

z	$I_{calc}(z)$ in A	$I^{(3)}_{measured}$ in A	$I^{(19)}_{measured}$ in A	$I^{(46)}_{measured}$ in A	$z^{(3)}_I$	$z^{(19)}_I$	$z^{(46)}_I$
410	0.5	0.528	0.524	0.519	43451	42371	43011
820	1.0	1.034	1.028	1.024	43788	42715	43350
1230	1.5	1.539	1.532	1.529	44123	43052	43685
1640	2.0	2.044	2.036	2.033	44460	43394	44019
2050	2.5	2.548	2.539	2.537	44796	43732	44352
2460	3.0	3.052	3.043	3.042	45131	44075	44692
2870	3.5	3.556	3.546	3.545	45466	44416	45024
3280	4.0	4.059	4.049	4.049	45801	44757	45361
3690	4.5	4.561	4.553	4.552	46138	45106	45693
4095	5.0	5.057	5.050	5.049	46469	45440	46031

Tabelle 6.3: Messwerte der DAC_A-Stromquelle (Segment-ID im hochgestellten Index, z ... an DAC gesendeter Wert, I_{calc} ... berechneter Strom, $I_{measured}$ ... gemessener Strom, z_I ... von ADC gemessener Wert)

Vergleich der Stromquellen: DAC_A / DAC_B

Die oben angeführten Messwerte erlauben es, die beiden verfügbaren Stromquellen miteinander zu vergleichen. So kann eruiert werden, ob die Unterteilung in DAC_A- (geplanter Bereich: 0.2 A bis 5 A) und DAC_B-Quelle (geplanter Bereich: 3 A bis 25 A) in der Praxis notwendig ist.

z	$I_{calc}(z)$ in A	$I^{(3)}_{measured}$ in A	$I^{(19)}_{measured}$ in A	$I^{(46)}_{measured}$ in A	$z_I^{(3)}$	$z_I^{(19)}$	$z_I^{(46)}$
82	0.5	0.525	0.525	0.516	43528	42458	43090
164	1.0	1.024	1.025	1.019	43780	42741	43351
246	1.5	1.522	1.526	1.522	44114	43078	43686
328	2.0	2.019	2.027	2.023	44444	43416	44022
410	2.5	2.517	2.527	2.526	44773	43754	44357
492	3.0	3.012	3.026	3.025	45104	44091	44686
574	3.5	3.507	3.524	3.525	45435	44430	45019
656	4.0	4.002	4.021	4.023	45765	44770	45354
738	4.5	4.495	4.519	4.520	46093	45108	45678
820	5.0	4.987	5.015	5.018	46424	45444	46010
902	5.5	5.478	5.509	5.514	46749	45777	46344
984	6.0	5.967	6.004	6.009	47075	46115	46672
1066	6.5	6.457	6.496	6.504	47399	46449	47005
1148	7.0	6.94	6.99	6.99	47728	46788	47337
1230	7.5	7.43	7.48	7.49	48049	47121	47659
1312	8.0	7.92	7.97	7.98	48374	47458	47989

Tabelle 6.4: Messwerte der DAC_B-Stromquelle (Segment-ID im hochgestellten Index, z ... an DAC gesendeter Wert, I_{calc} ... berechneter Strom, $I_{measured}$ ... gemessener Strom, z_I ... von ADC gemessener Wert)

Als Kenngröße zur Bewertung der Quellen wird die *relative Regelabweichung* verwendet

$$relative\ Regelabweichung = \frac{Sollwert - Istwert}{Sollwert} \cdot 100\%, \tag{6.1}$$

wobei der Sollwert durch I_{calc} (berechnet aus z mittels Gl. (3.3)) und der Istwert durch den gemessenen Strom $I_{measured}$ gegeben sind

$$relative\ Regelabweichung = \frac{I_{calc} - I_{measured}}{I_{calc}} \cdot 100\%. \tag{6.2}$$

Zu beachten ist das Vorzeichen der relativen Regelabweichung. Ist der Istwert größer als der Sollwert, muss zurückgeregelt werden und die Regelabweichung ist negativ.

Berechnet man nun die relative Regelabweichung für alle Werte von $I_{measured}$ (Tab. 6.3 und 6.4) und trägt das Ergebnis abhängig vom vorgegebenen Strom ein, erhält man folgende Kurven in den Abb. 6.3 - 6.5. Die Graphen vergleichen immer die Stromquellen (DAC_A, DAC_B) auf einem Stromquellen-Segment.
Weiters wurden beide Quellen einmal bei niederer und einmal bei höherer Temperatur vermessen. Da die Quellen direkt hintereinander gemessen werden, verändert

die Temperaturentwicklung der ersten Quelle den Verlauf der Zweiten. Um diesen Effekt zu kompensieren, wurde die Reihenfolge beim zweiten Durchlauf vertauscht.

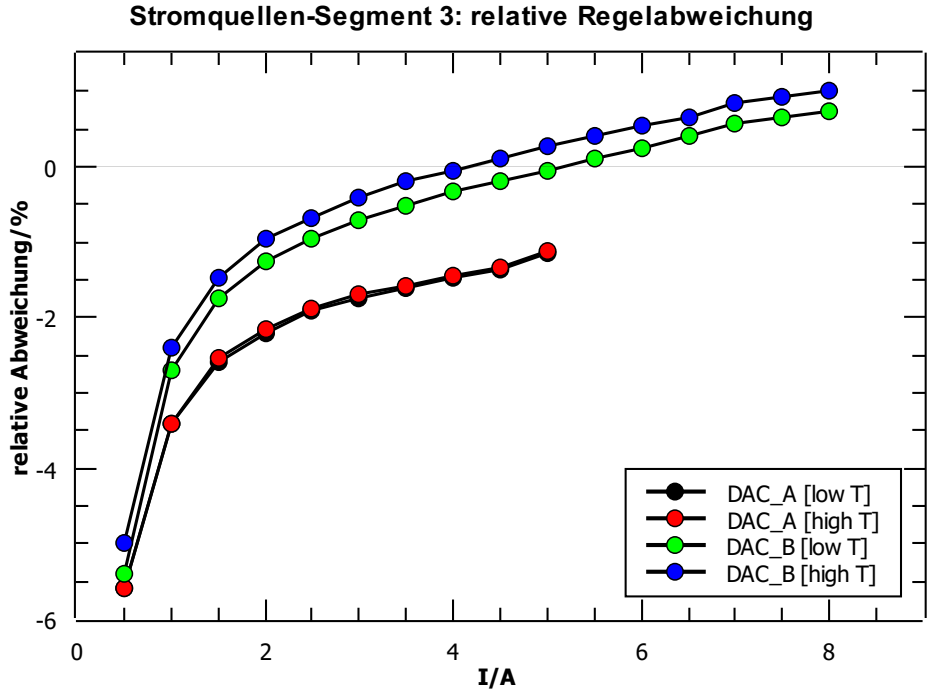

Abbildung 6.3: Segment-ID 3: Relative Regelabweichung in Abhängigkeit vom eingestellten Strom

Die einzelnen Kurven aller drei Graphen folgen einem ähnlichen Verlauf. Unterschiede, wie zum Beispiel der Offset von DAC_B in Abb. 6.3 im Vergleich zu den anderen Plots, sind wahrscheinlich auf Bauteiltoleranzen zurückzuführen.
Nach Analyse der Diagramme zeigt sich, dass die Stromquelle von DAC_A wesentlich stabiler gegen Temperaturänderungen ist, gleichzeitig aber eine (betragsmäßige) größere relative Regelabweichung im gesamten betrachteten Bereich aufweist. Die verhältnismäßig große Temperaturabhängigkeit der DAC_B-Quelle rührt vermutlich vom dazugehörigen Widerstand $R_{X,DAC\_B} = 0.1\,\Omega$ her. Dieser ist durch zwei einzelne $0.05\,\Omega$ Widerstände realisiert, wodurch die Wärme über eine größere Fläche an den Kühlkörper abgeführt, aber auch im Falle einer aktiven DAC_A-Quelle aufgenommen werden kann. Die Quelle von DAC_B ist also empfindlicher gegenüber Fremdbeheizung. Kombiniert man diesen Umstand mit der an den Widerständen anfallenden Leistung

$$P = I^2 \cdot R, \tag{6.3}$$

zeigt sich, dass bei gleichbleibendem I der Widerstand $R_{X,DAC\_A} = 0.5\,\Omega$ deutlich mehr Wärme erzeugt als der Widerstand $R_{X,DAC\_B} = 0.1\,\Omega$.
Das Heizen der DAC_A-Quelle hat somit deutlich mehr Einfluss auf die DAC_B-

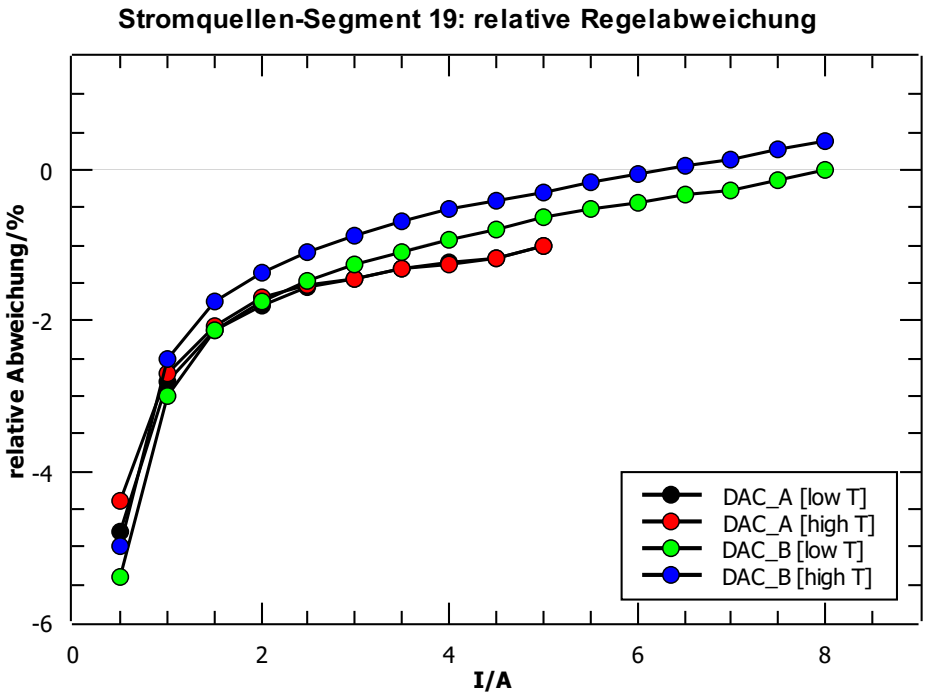

Abbildung 6.4: Segment-ID 19: Relative Regelabweichung in Abhängigkeit vom eingestellten Strom

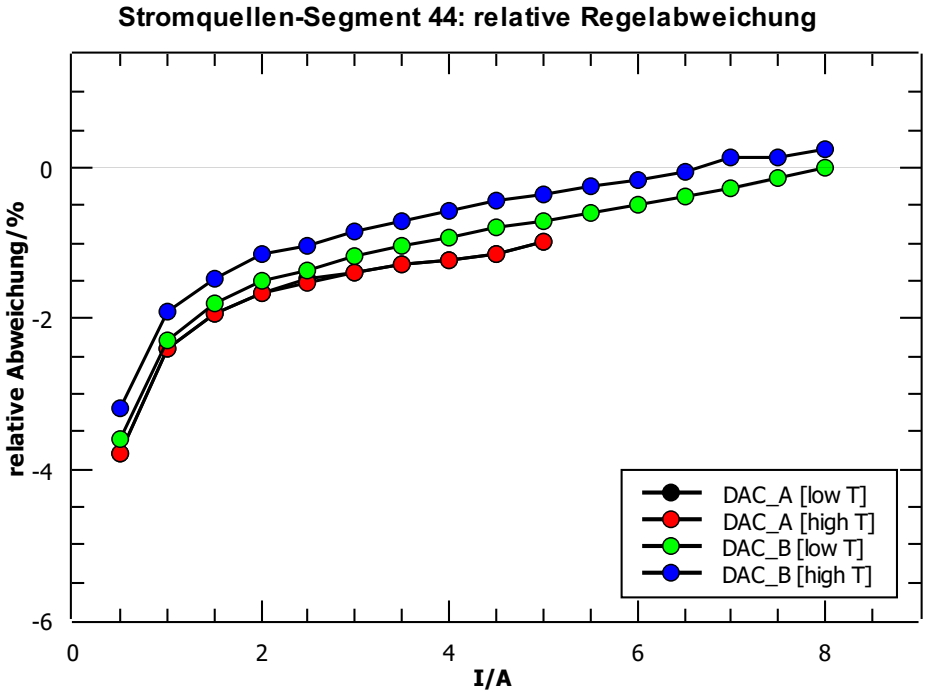

Abbildung 6.5: Segment-ID 44: Relative Regelabweichung in Abhängigkeit vom eingestellten Strom

Quelle, wodurch sich der Unterschied erklären lässt. Der positive Temperaturkoeffizient der Widerstände führt zu einem Anstieg des Widerstandswertes bei Temperaturerhöhung, wodurch der Strom abnimmt und die relative Regelabweichung somit Richtung positive y-Werte verschoben wird.

Einschränkungen

Die oben durchgeführte Messung unterliegt bestimmten Einschränkungen, die zu beachten sind. So befindet sich in der Messschaltung keine Resonatorspule, es wurde also die Artificial-Last vermessen. Zusätzlich beeinflusst der Innenwiderstand des Amperemeters die Messung. Dieser ist in etwa in der Größenordnung des Vorwiderstandes R_{VR}. Außerdem ist die Ungenauigkeit des Messgerätes zu beachten. Laut Hersteller liegt diese bei 1 %.
Ein wesentlicher Unterschied zum geplanten Betrieb ist die fehlende Kühlung. Sollte sich nach erfolgter Installation zeigen, dass der Betrag der relativen Regelabweichung für die DAC_B-Quelle geringer und über den Bereich von 0 A bis 5 A ausreichend stabil ist, könnte auf die DAC_A-Quelle verzichtet werden.

Anpassung der Strommessung über ADC

Verwendet man Gl. (3.17) um die z_I-Werte der Tab. 6.3 und 6.4 in Ströme zu übersetzen wird ersichtlich, dass die Messung durch den Hall-Sensor nicht mit der Messung des Amperemeters vereinbar ist:

z	$I_{calc}(z)$ in A	$I^{(3)}_{measured}$ in A	$z_I^{(3)}$	$I_{SEG}(z_I^{(3)})^1$ in A
1230	1.5	1.539	44123	0.505
1640	2.0	2.044	44460	0.826
2050	2.5	2.548	44796	1.146
2460	3.0	3.052	45131	1.466

[1] für höhere Genauigkeit wurde $U_H(0)$ zu 1.663 V gemessen

Tabelle 6.5: Messwerte aus Tab. 6.3 mit Verwendung von Gl. (3.17)

Es zeigt sich, dass weder der Stromwert noch die Differenz von einem zum nächsten Wert übereinstimmt. Die Schrittweite von einem $I_{SEG}(z_I)$-Wert zum folgenden beträgt im Schnitt aller erhobenen Messungen etwa 318 mA. Dies liegt an der Charakteristik des Hall-Sensors. Die Relation aus Gl. (3.13) gilt nur, falls der Chip mit einer Versorgungsspannung von 5 V betrieben wird, im vorliegenden Fall wird jedoch über eine 3.3 V Referenzspannungsquelle versorgt.
Die Differenzanpassung erfolgt durch Multiplikation der Gl. (3.17) mit dem Korrekturfaktor

$$I_{SEG}(z_I) \approx -25 \cdot (U_H(0) - \frac{2\,\mathrm{k}}{1\,\mathrm{k}} \cdot \frac{z_I}{2^{16}-1} \cdot V_{REF}) \cdot \frac{5}{3.18}. \tag{6.4}$$

Wendet man nun die erhaltene Gleichung auf die z_I-Werte (Tab. 6.5) an erhält man:

z	$I_{calc}(z)$ in A	$I^{(3)}_{measured}$ in A	$z_I^{(3)}$	$I_{SEG}(z_I^{(3)})$[1] in A
1230	1.5	1.539	44123	0.793
1640	2.0	2.044	44460	1.299
2050	2.5	2.548	44796	1.803
2460	3.0	3.052	45131	2.305

[1] für höhere Genauigkeit wurde $U_H(0)$ zu 1.663 V gemessen

Tabelle 6.6: Messwerte aus Tab. 6.3 mit Verwendung von Gl. (6.4)

Betrachtet man die Werte $I_{SEG}(z_I)$, wird ersichtlich, dass die Differenzmessung gute Werte liefert, einzig bei den absoluten Werten ist eine deutliche systematische Abweichung erkennbar. Diese kann teilweise durch eine additive Konstante (etwa 700 mA) kompensiert werden, bleibt aber durch die unterschiedlichen $U_H(0)$ der Stromquellen-Segmente (bedingt durch Bauteiltoleranzen) bestehen. Dies führt zur finalen Gleichung des ADCs zur Strommessung über den Hall-Sensor:

$$I_{SEG}(z_I) \approx -25 \cdot (U_H(0) - \frac{2\,\mathrm{k}}{1\,\mathrm{k}} \cdot \frac{z_I}{2^{16}-1} \cdot V_{REF}) \cdot \frac{5}{3.18} - 0.7. \tag{6.5}$$

6.2.2 Minimale Schiebeperiode

In Abschnitt 5.3.3 „Schiebeperiode“ wurde die theoretische minimale Schiebeperiode T_S des TW-Modes zu 2.984 µs ($= 44 \cdot T_I$) bestimmt. Nun sollen diese Angaben durch Messungen mittels Oszilloskop (*PicoScope 5444D MSO*) verifiziert werden, wobei als Muster eine High-Low-High („101“) Folge gewählt wurde. Bei der Messung werden die EN_R-Eingänge zweier benachbarter Segmente (Abb. 6.6), sowie auch der EN_A-Eingang vom zweiten Segment (Abb. 6.7), gemessen. Die ermittelten Werte für die jeweiligen Periodenzeiten sind aus den Angaben unterhalb der Messgraphen zu entnehmen.
In Abb. 6.6 ist die gewählte „101“Folge klar ersichtlich, auffällig ist jedoch der Unterschied zwischen High- und Low-Periode, sowie die Überschneidung der High-Pegel beider Kurven. Durch Vergleich mit Abb. 6.7 stellt man fest, dass der Bereich der Überschneidung gleich dem des Glitchs ist.
Es zeigt sich also, dass die ON-Zeit um etwa zwei Glitchzeiten im Vergleich zur OFF-Zeit verlängert wird. Dadurch, dass der Glitchvorgang bei beiden Übergängen (High-to-Low und Low-to-High) vorkommt, wird der High-Pegel beim Übergang High-to-Low verlängert. Dies erklärt auch die Überschneidung im High-Pegel zweier benachbarter Segmente.

Um die gemessenen mit den gerechneten Werten zu vergleichen, muss nun die reale Schiebeperiode aus den Messergebnissen bestimmt werden. Dazu gibt es zwei valide Wege.

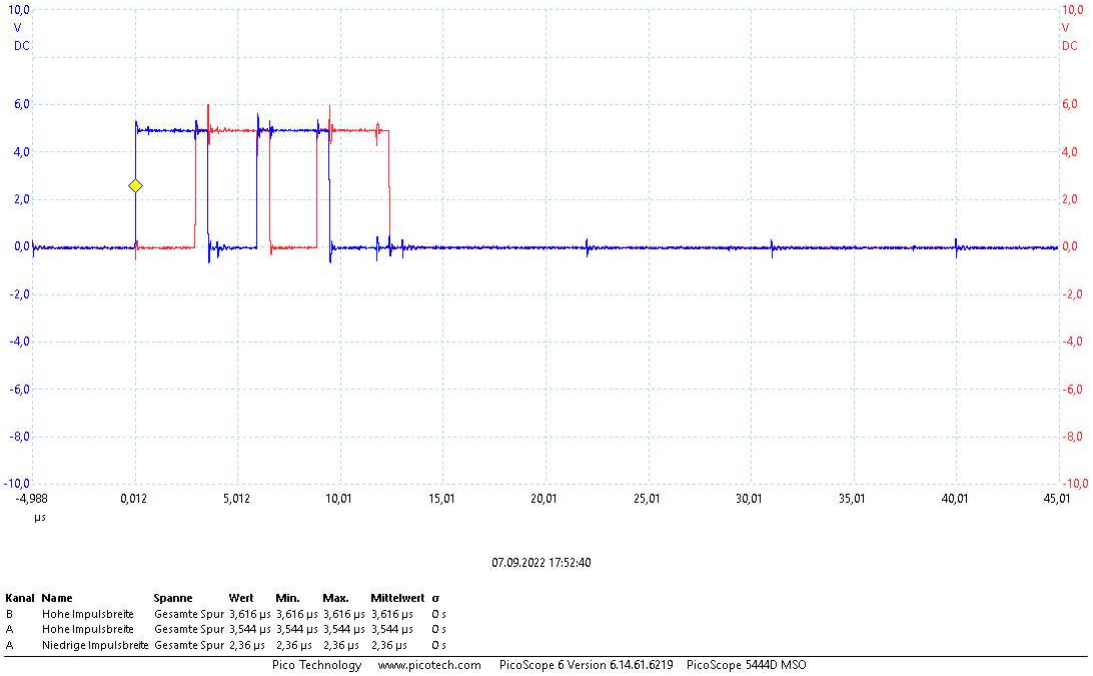

Abbildung 6.6: EN_R zweier benachbarter Segmente (EN_R1 in blau, EN_R1 in rot, siehe Abb. 3.5: Anschlussplan)

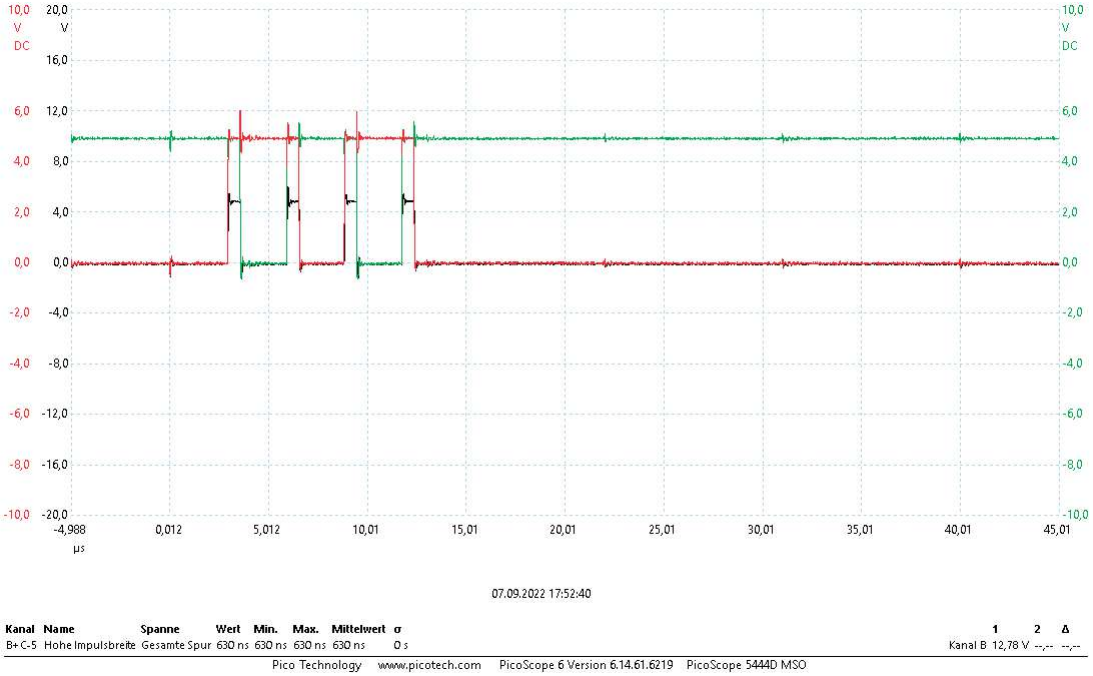

Abbildung 6.7: EN_R und EN_A eines Segmentes (EN_R in rot, EN_A in grün, Glitch in schwarz, siehe Abb. 3.5: Anschlussplan)

Einerseits kann die Glitchzeit von der On-Zeit subtrahiert, beziehungsweise zur Off-Zeit addiert werden

$$T_{ON,KanalA} = 3.544\,\mu s, \tag{6.6}$$
$$T_{OFF,KanalA} = 2.36\,\mu s, \tag{6.7}$$
$$T_{Glitch} = 0.63\,\mu s, \tag{6.8}$$
$$\rightarrow T_{ON,KanalA} - T_{Glitch} = 2.914\,\mu s, \tag{6.9}$$
$$\rightarrow T_{OFF,KanalA} + T_{Glitch} = 2.99\,\mu s. \tag{6.10}$$

Der Unterschied der beiden Ergebnisse rührt einerseits von Messfehlern her, andererseits von den Definitionen des High- ($> 70\%$) und Low-Zustandes ($< 30\%$).

Die zweite Methode ist die Mittelwertbildung von T_{ON} und T_{OFF}. Dadurch, dass die On-Zeit zwei und die Off-Zeit kein T_{Glitch} enthält, kann so die Schiebeperiode, welche eine Glitchzeit enthält, ermittelt werden:

$$T_{S,measured} = \frac{T_{ON,KanalA} + T_{OFF,KanalA}}{2} = 2.952\,\mu s. \tag{6.11}$$

Vergleicht man die Werte mit dem theoretisch ermittelten $T_S = 2.984\,\mu s$, zeigt sich eine ausreichend gute Übereinstimmung. Bis auf Gl. (6.9) liegen alle Werte innerhalb einer Instruktionenperiode T_I vom theoretisch bestimmten Wert. Auch die Glitchzeit T_{Glitch} liegt im Bereich der angestrebten 600 ns.

Nimmt man die Breite einer Resonatorspule a mit 1 cm an [10], so ergibt sich für die obere Neutronen-Grenzgeschwindigkeit des Resonators:

$$v_{0,max} = \frac{a}{T_S} = 3351\,\mathrm{m\,s^{-1}} \tag{6.12}$$

6.3 Gesamtsystemtests

Ein Gesamtsystemtest, im Sinne einer kompletten funktionalen Überprüfung, konnte aufgrund fehlender Hardware leider nicht durchgeführt werden. Um so einen Test zu ermöglichen, muss der aktuelle Aufbau durch die koaxialen Versorgungsleitungen (Abschnitt 3.5.3), sowie die Vorwiderstandsplatinen (Abschnitt 3.1.4) ergänzt werden. Letztere wurden im Zuge dieser Arbeit in 50-facher Ausführung gefertigt und stehen für die Montage bereit.

Aus oben genannten Gründen wurde eine simplifizierte Version des Gesamtsystemtests durchgeführt.
Zuerst wurden alle installierten Stromquellen-Segmente über I^2C-Kommunikation angesprochen. Es zeigte sich, dass mit der Platine in Slot 2 keine Verbindung zustande kam. Durch Austauschen der Segmente konnte der Fehler auf die Backplane

beziehungsweise die I^2C-Steuerplatine eingegrenzt werden.
Im Laufe der Tests konnte ein weiteres Problem identifiziert werden. Die Kommunikation mit den Platinen funktioniert bei 46 installierten Platinen problemlos. Fügt man nun jedoch ein weiteres Segment hinzu, fällt die Verbindung zu mehreren zuvor erfolgreich adressierten Platinen aus. Eine Verringerung der Datenrate schaffte hierbei keinerlei Abhilfe. Um diesen Fehler zu beheben sind weitere Untersuchungen notwendig. Sollte keine Ursache gefunden werden, kann das Problem durch Limitierung auf 46 installierte Segmente umgangen werden.

Der nächste Testabschnitt war das Einschieben eines Musters zur Schaltung der Stromquellen. Überprüft wurde dieser Vorgang einerseits durch Kontrolle der optischen Indikatoren (LEDs) auf den Stromquellen-Segmenten, andererseits durch Messen der Schaltvorgänge dreier Slots mittels Oszilloskop. Der Testlauf mit den LEDs verlief wie vorgegeben und auch die Messung (Abb. 6.6 und 6.7) lieferte die erwarteten Ergebnisse (genauere Analyse in Abschnitt 6.2.2).

6.4 TWM-Parameter für eine Wellenlänge von 5Å

Im folgenden Abschnitt werden die Parameter, die zu einer Wellenlängenselektion von 5 Å, also einer Geschwindigkeit von etwa $792\,\mathrm{m\,s^{-1}}$, bestimmt. Diese Wellenlänge ist charakteristisch für das in der Einleitung (Kapitel 1) erwähnte PERC-Experiment.
Die Parameterberechnungen wurden unter folgenden Annahmen getroffen:

i) Wellenlänge $\lambda = 5\,\text{Å}$,

ii) Breite einer Resonatorspule $a = 1\,\mathrm{cm}$,

iii) Pulsdauer $t_{Puls} = 2\,\mathrm{ms}$,

iv) Selektorfeld $B_0 = 1.356\,\mathrm{mT}$,

v) Spulenfeld $B_1 = 0.0222\,\mathrm{mT}$.

Zuerst muss man die Zeit t_{Puls} in Millisekunden zur Zeit t_{on} in Instruktionen-Zyklen (nur ganzzahlig) konvertieren

$$\widetilde{t}_{on} = \frac{t_{Puls}}{T_I} \approx 29490. \tag{6.13}$$

Da die Bits 15 und 16 nicht verwendet werden (siehe Tab. 5.2), müssen alle Bits mit Stelle > 14 um zwei Positionen nach links geschoben werden und es ergibt sich

$$\widetilde{t}_{on} \approx 29490_{dec} = 01|11\ 0011\ 0011\ 0010_{bin}, \tag{6.14}$$

$$\rightarrow t_{on} = 0001\ 00|11\ 0011\ 0011\ 0010_{bin} = 78642_{dec}. \tag{6.15}$$

Als nächstes wird das Spulenfeld in den dazugehörigen DAC-Wert übersetzt, wobei erst der Strom I mithilfe des Konversionsfaktors C_{Spule} von $0.009\,45\,\mathrm{mT\,A^{-1}}$ der Resonatorspulen berechnet wird

$$I = \frac{B_1}{C_{Spule}} = 2.349\,\mathrm{A}. \tag{6.16}$$

Es zeigt sich, dass der Strom im Arbeitsbereich beider DAC-Quellen liegt, weswegen sich zwei mögliche Werte (z_{DAC_A} und z_{DAC_B}, beide nur ganzzahlig) ergeben. Durch Umformen der Gl. (3.3) zu

$$z = \frac{I_X \cdot R_X}{U_{REF}} \cdot 2^{12} \tag{6.17}$$

mit $U_{REF} = 2.5\,\mathrm{V}$, $R_A = 0.5\,\Omega$ und $R_B = 0.1\,\Omega$, erfolgt die Berechnung:

$$z_{DAC\_A} = \frac{2.349\,\mathrm{A} \cdot 0.5\,\Omega}{2.5\,\mathrm{V}} \cdot 2^{12} \approx 1924, \tag{6.18}$$

$$z_{DAC\_B} = \frac{2.349\,\mathrm{A} \cdot 0.1\,\Omega}{2.5\,\mathrm{V}} \cdot 2^{12} \approx 385. \tag{6.19}$$

Nun kann der Benutzer entscheiden, welche der beiden Quellen verwendet werden soll. Um den Resonator mit der gewünschten Wellenlänge in Betrieb zu nehmen, müssen die oben bestimmten Werte über das Protokoll in Abschnitt 5.2.1 an die Ablaufsteuerung übertragen werden.

7 Zusammenfassung und Ausblick

7.1 Durchgeführte Arbeiten

Hardware

Wie in Kapitel 6 dargelegt wurde, war die Hardware durch mehrere Defekte beeinträchtigt. Mittels Schaltungsanalyse durch Soft- und Hardware konnten diese aufgespürt und so weit behoben werden, dass die Basisfunktionalität gewährleistet ist. So wiesen von den 48 Stromquellen-Segmenten 28 Stück Beschädigungen auf. Nach Abschluss der vorliegenden Arbeit konnte die Zahl auf eine Platine (Segment-ID 26) gesenkt werden.

Weiters wurde in der Auswertung der z-I-Kurve (Abschnitt 6.2.1) gezeigt, dass möglicherweise der gewünschte Stromstärkenbereich von 0 A bis 25 A vollständig durch die Stromquelle, welche durch DAC_B gesteuert wird, abgedeckt wird. Sollten in Zukunft Printplatten ersetzt werden müssen, so könnte es ausreichen, wenn die Bauteile für die DAC_A-Stromquelle eingespart werden.

Im Bereich der Stromversorgung wurden die Pläne weiter konkretisiert. So wurde ein Konzept zur Fertigung der Koaxial-Versorgungsleitungen entwickelt, welches aufgrund von Lieferengpässen noch nicht umgesetzt wurde. Die Dioden (inklusive Halterung und Kühlung) zur Einbindung der angeschafften Hochleistungs-Netzteile wurden angeschafft und es wurde ein DC/DC-Konverter zum Schutz der empfindlichen Elektronik installiert.

Software

Es wurde eine Assembler-Software entwickelt, welche durch den Benutzer via UART konfiguriert werden kann und die drei angeführten Betriebsmodi abdeckt. Die Schiebeperiode des TWM konnte dabei unter 3 µs gehalten werden, was für thermische und kalte Neutronen und somit die geplante Anwendung ausreichend ist.
Neben den drei Modi wurden weitere Features wie Temperatur-, Spannungs- und Stromüberwachung implementiert. Dazu wurden die chipspezifischen I^2C-Protokolle in Assembler programmiert und ausgetestet.

Zusätzlich wurde eine Testsoftware erstellt, welche detailliertes Testen der einzelnen Stromquellen-Segmente erlaubt. Sie ermöglichte das Finden und Beheben der beschriebenen Hardwaredefekte.

7.2 Geplante Erweiterungen und Arbeiten

In diesem Abschnitt werden die notwendigen Schritte aufgelistet, welche bis zur Einsatzfähigkeit des Neutronenspinresonators noch zu erledigen sind.

- Wie in Kapitel 6 dargelegt, liegen nach ausführlichen Hardwaretests und Fehlerbehebungen immer noch einige Defekte vor. Das Gros davon hat jedoch durch Redundanzen keinerlei Effekt auf die Funktionalität. Das Ziel sollte dennoch sein, die Hardware in einen möglichst fehlerfreien Zustand zu bringen.

- Die in Abschnitt 3.1.5 vorgestellte Platine, welche für die Trigger-Funktion nötig ist, wurde bisher lediglich gefertigt, aber noch nicht ins Gesamtsystem integriert. Ein Konzept zur Installation liegt in groben Zügen vor, es wird voraussichtlich auf die Pins A4 (TRI_IN) und A7 (TRI_SYN) des Mikrocontrollers zurückgegriffen.

- Aktuell liegt die empfindliche Steuerungselektronik frei. Um Verstauben und mögliche Beschädigung durch elektrostatische Entladung zu vermeiden, wurde ein passendes Gehäuse von Roman Gergen entworfen. Dieses sollte zeitnah gefertigt und montiert werden.

- Die koaxialen Versorgungsleitungen (Abschnitt 3.5.3) wurden zum Zeitpunkt dieser Arbeit noch nicht gefertigt und installiert, jedoch wurde ein Konzept zur Realisierung erarbeitet.

- In Abschnitt 3.1.4 wurden die Vorwiderstandsplatinen für Artificial- und Real-Last vorgestellt. Im Verlauf dieser Arbeit wurden 50 Stück, sowie ein wassergekühltes Formrohr (zur Halterung und Kühlung, entworfen von R. Gergen) erstellt, jedoch noch nicht montiert.

- Nachdem die Vorwiderstandsplatinen und die koaxialen Versorgungsleitungen installiert sind, kann die Verkabelung der einzelnen Komponenten durchgeführt werden. Zwecks besserer Übersicht sollte auf farbkodierte Kabel zurückgegriffen werden.

- Um den MONOPOL-Resonator unter Volllast zu betreiben (14.4 kW), wird eine Wasserkühlung benötigt. Diese wurde noch nicht installiert und getestet.

- Die bestellte Halterung der Gleichrichterdiode muss nach erfolgter Lieferung montiert werden.

- Die angeschafften 12 V DC-Netzteile müssen vor der finalen Installation, zwecks Schadensprävention, ausgiebig mit Gleichrichterdiode und Kondensatorbank getestet werden. Sollten hier keine Probleme auftreten, können die Netzteile in das Gesamtsystem integriert werden.

- Das grafische Benutzerinterface befindet sich aktuell noch in einer Entwicklungsversion. Derzeit wird im Rahmen einer Bachelorarbeit an der Finalisierung der Software gearbeitet. Nach Vollendung sollte das Zusammenspiel von Controllersoftware und GUI umfangreich getestet werden.

- Der finale Schritt zum Abschluss des MONOPOL-Projektes ist die Vereinigung von Resonator und Elektronik. Sobald alle dazugehörigen Tests abgeschlossen sind, kann der Neutronenspinresonator in der aktiven Forschung eingesetzt werden.

7.3 Verbesserungsmöglichkeiten

Neben den geplanten Erweiterungen bestehen noch weitere, optionale Verbesserungsmöglichkeiten, welche die Systemperformance beziehungsweise die Benutzerfreundlichkeit erhöhen, sowie weitere Funktionalitäten hinzufügen.

- Im Zuge dieser Arbeit wurden zwei Softwarepakete erstellt, eines für den Betrieb, das andere zu Testzwecken (Abschnitt 5.4). Es bietet sich an, beide zu einer Gesamtlösung zu kombinieren und den normalen Betrieb durch einen Debug-Mode zu erweitern. In diesem sollten die einzelnen Systemkomponenten individuell adressier- und somit überprüfbar sein. Die Steuerung könnte über eine eigene PC-Software, welche vorzugsweise in die GUI-Software integriert ist, via UART-Schnittstelle erfolgen.

- In der aktuellen Version wird nicht überprüft, ob die DACs konfiguriert wurden, bevor mit dem Startvorgang begonnen werden kann. Dies sollte zur Erhöhung der Sicherheit entweder controller- oder pc-seitig implementiert werden.

- Beim Real-Last Test (Abschnitt 5.1.3) werden derzeit zuerst die Ströme über alle Spulen aktiviert und anschließend gemessen. Ändert man die Software dahingehend, dass die Spulen einzeln unter Strom gesetzt, gemessen und direkt danach wieder deaktiviert werden, sinkt der Spitzenleistungsbedarf erheblich.

- Sobald im Trigger Mode der Strom über die Artificial-Lasten aktiviert ist, sollte ein Timer gestartet werden. Trifft innerhalb einer zuvor festgelegten Zeit kein Trigger-Signal ein, wird ein Abschaltvorgang eingeleitet.

- Auf der zentralen Steuerplatine ist ein nichtflüchtiger Speicher in der Form eines FRAM verbaut. Dieser kann genutzt werden, um System- und Betriebsparameter zu speichern. So könnten getätigte Einstellungen auch nach einem Neustart ausgelesen und verwendet werden.

- Neben dem FRAM-Baustein auf der Steuerplatine befinden sich weitere Speicherzellen auf den Stromquellen-Segmenten. Diese können zur Speicherung von platinenspezifischen Parametern verwendet werden. So kann zum Beispiel

die jeweilige ID oder aber auch der $U_H(0)$-Spannungswert des Hall-Sensors (zur besseren Ermittlung des Stromwertes) vermerkt werden.

- Die Delay-Funktion (Abschnitt 5.4) hat bei einem Argument ungleich 0 einen Offset von 5 Befehlszyklen. Diese Zahl kann in der GUI-Software kompensiert werden, wodurch die Genauigkeit erhöht werden würde.

- Im derzeitigen Aufbau wird der Mikrocontroller mit einem externen Oszillator von 29.4912 MHz getaktet, jedoch unterstützt der Prozessor Frequenzen bis zu 32 MHz. Bei einem Tausch würde sich die Abarbeitungsgeschwindigkeit um etwa 8.5 % erhöhen. Diese würde sich in der Praxis aber nicht vollständig umsetzen lassen, da die zeitlichen Anforderungen aller angeschlossenen Bausteine, vor allem die der CPLDs, berücksichtigt werden müssen.

- Die verwendete Toolchain, bestehend aus XC16 Compiler und MPLAB Entwicklungsumgebung, unterstützt für den eingesetzten Prozessor die Programmierung durch *C* und *Assembler*. So können die zeitkritischen Teile, namentlich der Schiebevorgang, in Assembler und die restlichen Teile in C programmiert werden. Dadurch wäre das Programm deutlich lesbarer und einfacher zu erweitern. Nachteil wäre allerdings der anfallende Overhead. Bei der Verwendung von C wird ein deutlich größerer Teil des Programmspeichers benötigt, was die Programmlänge und damit den Raum für Funktionalitäten begrenzen könnte.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] D. Dubbers, H. Abele, S. Baeßler, B. Märkisch, M. Schumann, T. Soldner, and O. Zimmer. *A clean, bright, and versatile source of neutron decay products.* Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **596**, 238 (2008).

[2] G. Konrad *et al. Neutron Decay with PERC: a Progress Report.* J. Phys. Conf. Ser. **340**, 012048 (2012).

[3] X. Wang, C. Ziener, H. Abele, S. Bodmaier, D. Dubbers, J. Erhart, A. Hollering, E. Jericha, J. Klenke, H. Fillunger *et al. Design of the magnet system of the neutron decay facility PERC.* In *EPJ Web of Conferences*, vol. 219, p. 04007. EDP Sciences (2019).

[4] H. Krieger. *Strahlungsquellen für Physik, Technik und Medizin.* Springer Berlin Heidelberg, Berlin, Heidelberg (2022).

[5] W. Demtröder. *Experimental-Physik 3.* Springer Spektrum, Berlin, 5., neu bearbeitete und aktualisierte Auflage edn. (2016).

[6] V. R. G. M. Drabkin, V.A. Trunov. *Static magnetic field analysis of a polarized neutron spectrum.* JETP **27**, 194 (1963).

[7] E. J. G. Badurek. *Upon the versatility of spatial neutron magnetic spin resonance.* Physica B **335**, 215 (2003).

[8] M. Agamalyan, G. Drabkin, and V. I. Sbitnev. *Spatial spin resonance of polarized neutrons. A tunable slow neutron filter.* Physics Reports **168**, 265 (1988).

[9] E. Jericha, C. Gösselsberger, H. Abele, S. Baumgartner, B. M. Berger, P. Geltenbort, M. Hino, T. Oda, R. Raab, and G. Badurek. *MONOPOL - A traveling-wave magnetic neutron spin resonator for tailoring polarized neutron beams.* Scientific Reports **10**, 5815 (2020).

[10] A. Frank. *Konzept und Test einer Steuerung für einen magnetischen Wanderwellenresonator zur Wellenlängenselektion von langsamen Neutronen.* Diplomarbeit, Technische Universität Wien (2019).

[11] N. Semiconductors. *I2C-bus specification and user manual.* Spezifikation, NXP Semiconductors (2021).

[12] H. Bähring. *Anwendungsorientierte Mikroprozessoren*. Springer Vieweg, Berlin, 4., vollständig überarbeitete Auflage edn. (2010).

[13] J. G. E. Hering and K. Bressler. *Elektronik für Ingenieure und Naturwissenschaftler*. Springer Vieweg, Berlin, 7., aktualisierte und verbesserte Auflage edn. (2017).

[14] A. Meroth and P. Sora. *Sensornetzwerke in Theorie und Praxis*. Springer Vieweg, 2. Auflage edn. (2021).

[15] W. Demtröder. *Experimental-Physik 2*. Springer Spektrum, Berlin, 6., überarbeitete und aktualisierte Auflage edn. (2013).

[16] T. Mühl. *Elektrische Messtechnik*. Springer Vieweg, Wiesbaden, 5., überarbeitete und erweiterte auflage edn. (2017).

[17] H. Bernstein. *NF- und HF-Messtechnik*. Springer Vieweg, 1. Auflage edn. (2015).

[18] E. G. U. Tietze, C. Schenk. *Halbleiter-Schaltungstechnik*. Springer Vieweg, 16 edn. (2019).

[19] A. Frank. *Software-Entwicklung und Test der Steuerungselektronik für den Neutronenresonator Monopol*. Bachelorarbeit, Technische Universität Wien (2015).

[20] A. Frank. *Test der Steuerungselektronik für den Neutronenresonator Monopol,*. Projektarbeit, Technische Universität Wien (2017).

[21] E. Moghadas. *Test and characterization of the control electronics for a neutron resonator*. Bachelorarbeit, Technische Universität Wien (2020).

[22] C. Hofbauer. *Monopol - Ansteuerung der Stromversorgung für einen Neutronen Wellenlängenselektor*. Projektarbeit, Technische Universität Wien (2021).

[23] M. T. Inc. *PIC24FV32KA304 FAMILY-Datasheet*. `https://www.microchip.com/doclisting/TechDoc.aspx?type=ReferenceManuals` (2011-2017).

[24] L. S. Corp. *ispLSI® 2064/A*. `https://www.latticesemi.com/-/media/LatticeSemi/Documents/DataSheets/ispLSI/ispLSI2064ADataSheet.ashx?la=en` (2006).

[25] T. I. Incorporated. *TMP10x Temperature Sensor With I2C and SMBus Interface with Alert Function in SOT-23 Package*. `https://www.ti.com/product/TMP101#tech-docs` (2002-2015).

[26] L. T. CORPORATION. *LTC2631 Single 12-/10-/8-Bit I2C VOUT DACs with 10ppm/°C Reference*. `https://www.analog.com/en/products/ltc2631.html#product-documentation` (2008).

[27] M. T. Inc. *MCP3421 18-Bit Analog-to-Digital Converter with I2C Interface and On-Board Reference*. http://ww1.microchip.com/downloads/en/devicedoc/22003e.pdf (2009).

[28] BURR-BROWN. *INA146 High-Voltage, Programmable Gain DIFFERENCE AMPLIFIER*. https://www.ti.com/lit/ds/symlink/ina146.pdf?ts=1656350320871&ref_url=https%253A%252F%252Fwww.ti.com%252F (1999).

[29] L. T. CORPORATION. *LTC2631 Single 12-/10-/8-Bit I2C VOUT DACs with 10ppm/°C Reference*. https://www.analog.com/media/en/technical-documentation/data-sheets/24613fa.pdf (2008).

[30] A. MicroSystems. *ACS756xCB Fully Integrated, Hall Effect-Based Linear Current Sensor IC with 3 kVRMS Voltage Isolation and a Low-Resistance Current Conductor*. https://www.ti.com/product/TMP101#tech-docs (2017).

[31] T. I. Incorporated. *TCA9554A Low Voltage 8-Bit I2C and SMBus Low-Power I/O Expander With Interrupt Output and Configuration Registers*. https://www.ti.com/product/TCA9554A?dcmp=dsproject&hqs=pf#tech-docs (2010-2017).

[32] F. Pelczar. *MONOPOL PSU Test Report LU050218*. Report, Technische Universität Wien (2018).

[33] M. Sajatovic. *Parametrization Software for a Badurek Travelling Wave Mode Resonator*. Projektarbeit, Technische Universität Wien (2018).

[34] M. T. Inc. *PIC24F Family Reference Manual*. https://www.microchip.com/doclisting/TechDoc.aspx?type=ReferenceManuals (2006).

[35] M. T. Inc. *16-bit MCU and DSC Programmer's Reference Manual*. https://www.microchip.com/doclisting/TechDoc.aspx?type=ReferenceManuals (2009).

[36] M. T. Inc. *MPLAB® XC16 ASSEMBLER, LINKER AND UTILITIES User's Guide*. https://www.microchip.com/doclisting/TechDoc.aspx?type=ReferenceManuals (2013-2016).