TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

Diplomarbeit

Entwicklung einer Motorsteuerung für ein mehrachsiges Aktorsystem

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines

Diplom-Ingenieurs

unter der Leitung von

Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Johann Wassermann
(E325 - Institut für Mechanik und Mechatronik, Bereich: Messtechnik und Aktorik)

eingereicht an der Technischen Universität Wien

Fakultät für Maschinenwesen und Betriebswissenschaften

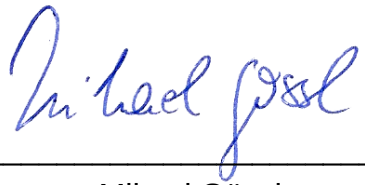
von

Mikael Gössl, BSc.

0925994 (066 482)

Gutraterplatz 3/13

1160 Wien



Wien, im August 2015

Mikael Gössl

TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

Ich habe zur Kenntnis genommen, dass ich zur Drucklegung meiner Arbeit unter der Bezeichnung

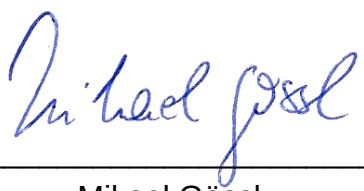
Diplomarbeit

nur mit Bewilligung der Prüfungskommission berechtigt bin.

Ich erkläre weiters Eides statt, dass ich meine Diplomarbeit nach den anerkannten Grundsätzen für wissenschaftliche Abhandlungen selbstständig ausgeführt habe und alle verwendeten Hilfsmittel, insbesondere die zugrunde gelegte Literatur, genannt habe.

Weiters erkläre ich, dass ich dieses Diplomarbeitsthema bisher weder im In- noch Ausland (einer Beurteilerin/einem Beurteiler zur Begutachtung) in irgendeiner Form als Prüfungsarbeit vorgelegt habe und dass diese Arbeit mit der vom Begutachter beurteilten Arbeit übereinstimmt.

Wien, im August 2015



Mikael Gössl

Vorwort

Die vorliegende Diplomarbeit wurde im Zeitraum von Oktober 2014 bis Juli 2015 verfasst. Die Arbeit wurde zum Abschluss meines Studiums Wirtschafts-ingenieurwesen-Maschinenbau an der TU Wien am Institut für Mechanik und Mechatronik angefertigt. Die Leitung übernahm Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Johann Wassermann, dem ich für die Vergabe des interessanten Themas und für hilfreiche Erklärungen während der Diplomarbeit als auch für sehr lehrreiche Vorlesungen aus dem Bereich der Mikroelektronik und Messtechnik im Verlauf meines Masterstudiums besonders danken möchte.
Hervorheben möchte ich auch meinen Dank an Dipl.-Ing. Klaus Bergkirchner: Er war mein Ansprechpartner bei allen kleineren und größeren Problemen während der Entwicklung und konnte mir mit seinem Fachwissen und seiner praktischen Erfahrung im Bereich der Mikroelektronik und Schaltungstechnik oft weiterhelfen. Weiters gilt mein Dank auch allen anderen Institutsmitarbeitern, die stets für ein gutes Arbeitsklima im Labor gesorgt haben.

Kurzfassung

Diese Diplomarbeit behandelt die Entwicklung einer neuen Motorsteuerung für ein mehrachsiges Aktorsystem, das seine Verwendung in einem Massagebett für den Einsatz im medizinischen Bereich findet. Genauer handelt es sich dabei um einen sechs-achsigen Roboterarm, der auf, durch Servomotoren angetriebenen, translatorischen und rotatorischen Achsen einer 3D-Bewegungsbahn folgt.

Diese Arbeit beschreibt in erster Linie die Entwicklung der Motorsteuerung, die sich im Wesentlichen aus dem Schaltplan- und Leiterplatten-Design sowie der Software-Entwicklung dieser modularen Steuerungs-Einheit zusammensetzt. Grund für die Neuentwicklung der bestehenden Motorsteuerung war der Bedarf nach einem leistungsstärkeren Mikrocontroller, sowie einem robusteren Kommunikationssystem, was durch den Einsatz eines ARM-Cortex-M4-Prozessors und ein CAN-Bus-System umgesetzt wurde.

Die Vorgänge und Ergebnisse der weiteren Entwicklungsschritte des Massagebetts werden in anderen Arbeiten des Instituts für Mechanik und Mechatronik an der TU Wien präsentiert.

Abstract

The following master’s thesis focuses on the development of a new motion control system for a multiaxial actuator system, which is used in a massage bed for medical applications. The system consists of a six-axis robotic arm, which is driven by servo motors across translational and rotatory axis to follow a 3D path of motion.

Primarily this thesis describes the development of the motor controller, i. e. the schematic- and PCB-design, as well as the software-development of this motion controller. The reason for the redevelopment of the existing motion controller was the need for a more powerful microprocessor and a more robust communication system. This was accomplished by usage of an ARM-Cortex-M4 processor and a CAN bus system.

The processes and results of further developments regarding the massage bed are addressed by other theses at the Institute for Mechanics and Mechatronics at the TU Wien.

Inhaltsverzeichnis

1 Einleitung und Aufgabenstellung

Die Entwicklung eines Massagebetts für Verwendung im medizinischen Bereich ist seit einigen Jahren Thema wissenschaftlicher Arbeiten am Institut für Mechanik und Mechatronik der TU Wien [1], [2]. Das Projekt umfasst Arbeiten in Bereichen der Mechanik und Konstruktion des Massagebetts, über die mess- und elektrotechnische Entwicklung des Sensor- und Aktorsystems bis hin zur informationstechnischen Implementierung der Steuerungs- und Regelungstechnik des Systems.

Zu Beginn dieser Diplomarbeit befand sich das Massagebett bereits in einer fortgeschrittenen Entwicklungsphase. Im Zuge von Dissertationen um die Entwicklung des Massagebetts traten Probleme mit den verwendeten Controller-Einheiten, die für die Motoransteuerung und Informationsverarbeitung von Sensordaten zuständig sind, auf. Die Fehlerursache konnte auf eine unzureichende Rechenleistung des verwendeten Mikrocontrollers sowie eine störungsanfällige Informationsübertragung am Main-Bus zurückgeführt werden. Das Hauptziel dieser Diplomarbeit ist demnach die Neuentwicklung einer Motorsteuerung mit einem leistungsstärkeren Mikrocontroller. Ein weiteres Ziel ist die Umstellung des Main-Bus-Systems von RS-485 auf einen CAN-Bus, welcher wesentliche Vorteile insbesonders gegenüber elektromagnetischen Störungen bietet.

Die Entwicklung einer neuen Motorsteuerung beinhaltet die Erstellung einer neuen Controllerplatine. Die auszuführenden Entwicklungsschritte sind:

- Schaltplanentwurf, sowie Bauteileauswahl und Simulation bestimmter Komponenten,
- Entwurf des Leiterplatten-Layouts (Routing),
- Schaltungsaufbau eines Prototypen und
- Programmierung des verwendeten Mikrocontrollers.

Kapitel 2 befasst sich mit dem grundlegenden Aufbau des Steuerungssystems des Massagebetts sowie der Controller-Einheit und gibt eine Einführung in das CAN-Bus-Protokoll. Ein Eignungstest des neuen Mikrocontrollers im Vergleich zum Mikrocontroller der Vorgängerversion folgt in **Kapitel 3**. Das **Kapitel 4** beschreibt die Umsetzung des Schaltplanentwurfs und Überlegungen zur Bauteilauswahl sowie Simulationsergebnisse bestimmter Funktionsgruppen. Der Aufbau der Platine und der Entwurf des Layouts werden in **Kapitel 5** beschrieben. Die softwaretechnische Implementierung und Programmierung des Mikrocontrollers folgt in **Kapitel 6.**

2 Grundlagen

2.1 Massagebettprojekt

Abbildung 2.1 gibt einen Überblick über die wesentlichen Komponenten, die zum Betrieb des Massagebetts notwendig sind. Im aktuellen Versuchsaufbau verfügt das Massagebett über einen sechs-achsigen Massagearm, es soll im Endprodukt um einen weiteren Massagearm erweitert werden. Die Benutzer-Interaktion erfolgt über ein LabVIEW-Terminal. Über den Main-Bus kommuniziert diese Haupt-Steuerungseinheit den Massagepfad sowie sämtliche Regelparameter mit allen Controller-Einheiten. Die Aufgabe der Controller-Einheiten besteht in der Positions-, und Geschwindigkeitsregelung der Achsen und der damit verbundenen Stromregelung der Motor-Phasenströme. Die Z-Achse verfügt zusätzlich über einen DMS-Kraftsensor.

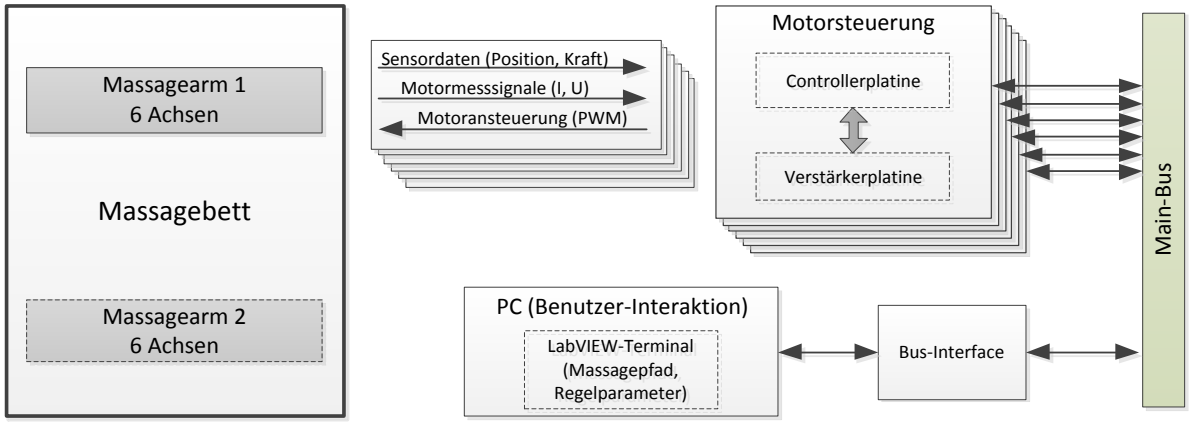


Abbildung 2.1 - Übersicht Massagebettprojekt

2.2 Motorsteuerung

Das Hauptziel dieser Diplomarbeit ist die Entwicklung einer neuen Motorsteuerung. Diese ist für alle verwendeten Motoren identisch und besteht im Wesentlichen aus einer Steuereinheit (Controllerplatine) und einer Verstärkereinheit (in weiterer Folge: Verstärkerplatine). Die beiden Platinen befinden sich im selben Gehäuse und bilden eine modulare Einheit im Aufbau des Massagebetts, siehe Abbildung 2.1. Pro Motor bzw. Achse wird eine solche Einheit benötigt, wobei die Zuweisung zu einer Achse durch eine hardwaremäßig codierbare Adresse erfolgt – dies gewährt eine vollständige Austauschbarkeit der Controller-Einheiten untereinander und führt zu einer erheblichen Verringerung der Komplexität. Für den aktuellen sechs-achsigen Versuchsaufbau werden demnach 6 Einheiten benötigt.

Da sich die bisher verwendete Verstärkerplatine im Versuch bewährt hat, soll im Zuge dieser Diplomarbeit nur die Controllerplatine neu entwickelt werden. Dabei

bleibt die Schnittstelle zur Verstärkerplatine bestehen. Nach außen hin, d. h. zu Komponenten außerhalb der Motorsteuerung, bleiben die Schnittstellen bis auf den Main-Bus bestehen. Die Umstellung auf einen CAN Main-Bus war eine der grundlegenden Aufgabenstellungen. In Kapitel 2.3 wird näher auf den CAN-Bus eingegangen. Abbildung 2.2 zeigt den funktionellen Aufbau der Controllerplatine sowie deren Schnittstellen zur Verstärkerplatine und Umgebung. Von der Verstärkerplatine werden nur die direkt mit der Controllerplatine in Verbindung stehenden Komponenten aufgezeigt.

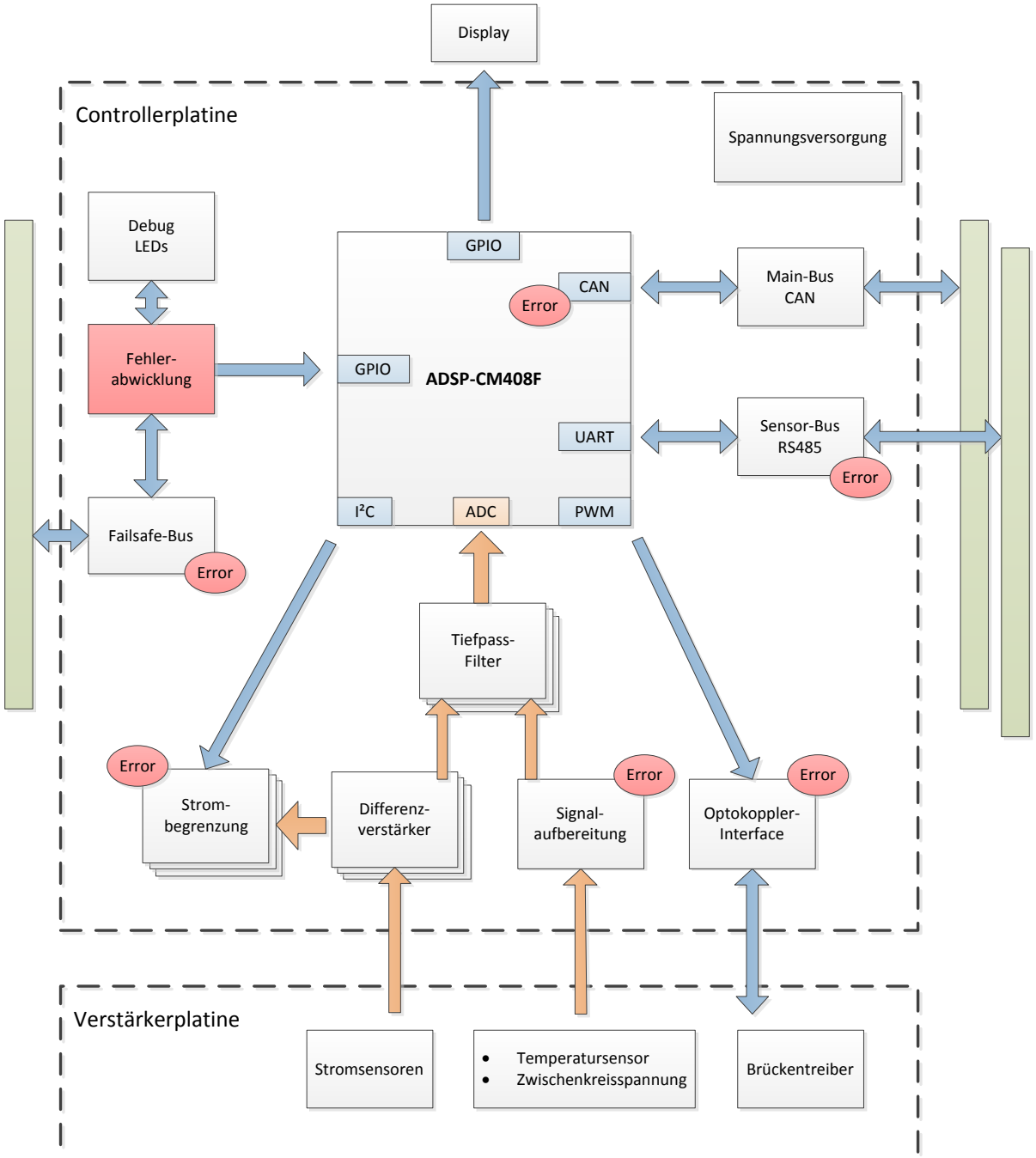


Abbildung 2.2 - Blockdiagramm Controllerplatine

Es soll nun kurz auf die einzelnen funktionalen Blöcke eingegangen werden. Eine detailliertere Beschreibung der Blöcke sowie der enthaltenen Komponenten folgt im Kapitel 4.

2.2.1 Mikrocontroller ADSP-CM408F

Für das Projekt wurde der Mikrocontroller *CM408F* des Chipherstellers *Analog Devices* ausgewählt. Es handelt sich um einen ARM-Prozessor der Cortex-M4 Familie. Das britische Unternehmen ARM liefert Prozessor-Architekturen und Standards, welche mittlerweile von allen namhaften Chipherstellern lizensiert wurden und aus dem Embedded-Bereich nicht mehr wegzudenken sind. Cortex-M4 ist eine Familie von 32-bit Mikroprozessoren, die auf der ARMv7-ME Architektur [3] basieren. Der *ADSP-CM408F* ist ein Prozessor, erweitert durch DSP Funktionen, der dem neuesten Stand der Technik entspricht.

Die für die Entwicklung der Controllerplatine wichtigsten Eckdaten und Funktionen des Prozessors sind:

- 240 MHz Core-Taktfrequenz
- 2 ADCs (SAR, 16-bit, 8 Kanal)
- 384 kB SRAM
- 2048 kB Flash
- 12 PWM-Paare
- 8 GP Timer (32-bit)
- 2 CAN 2.0B Controller
- 3 UART Ports
- I^2C Controller
- 17 DMA Kanäle
- 91 GPIOs (alle interruptfähig)
- FPU (einfache Genauigkeit)

Die Funktionen des *CM408F* sind umfangreich. Abbildung 2.3 zeigt ein Blockdiagramm der *ADSP-CM40x* Prozessor-Serie. Im Kapitel 6 wird genauer auf die verwendeten Module eingegangen.

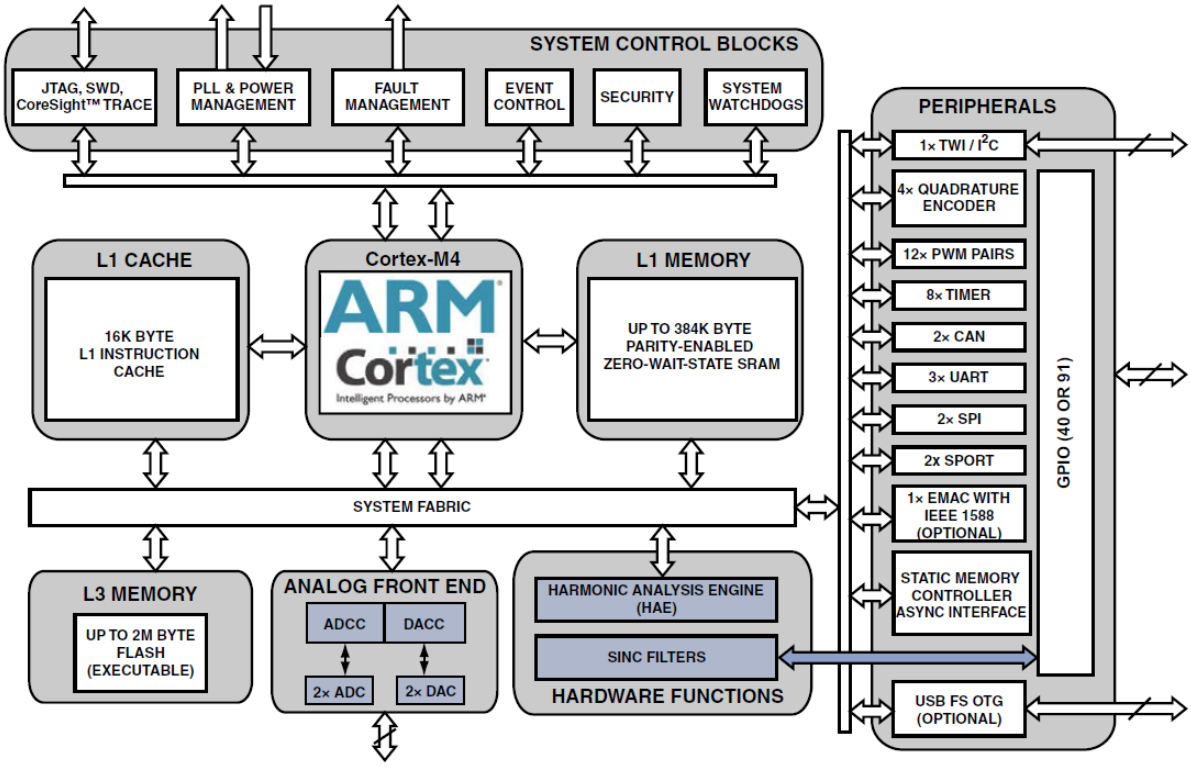


Abbildung 2.3 - CM40x Blockdiagramm [4]

Eine interessante Funktion, die nicht aus dem Blockdiagramm hervorgeht, ist die sogenannte Trigger Routing Unit (TRU), die einen Eingriff in Abläufe auf System-Level ohne Core Intervention ermöglicht. Bestimmte Peripherie-Module des Prozessors können als Trigger-Slave einem Trigger-Master zugewiesen werden. Beispiele für die Anwendung der TRU wären:

- Starten einer ADC-Messung nach jeder PWM-Periode. PWM triggert den ADC-Controller.
- Starten einer DMA-Sequenz nach dem Empfang eines Pakets am UART-Port.

Wie die TRU speziell in diesem Projekt eingesetzt wird, wird in Kapitel 6.3.4 (Programmierung des ADC-Controllers) erläutert.

2.2.2 Optokoppler-Interface

Die Motoranregung erfolgt auf der Verstärkerplatine durch einen 3-Phasen-Wechselrichter. Die Ansteuerung der einzelnen Phasen erfolgt durch PWM-Paare über jeweils einen Halb-Brücken-Treiber. Das Optokoppler-Interface dient der galvanischen Trennung der Verstärker- von der Controllerplatine. Die Brückentreiber haben Fehler-Ausgänge, welche ebenfalls über das Optokoppler-Interface auf die Controllerplatine geführt werden.

2.2.3 Strommessung und Strombegrenzung

Die Messung der Motor-Phasenströme erfolgt auf der Verstärkerplatine durch potentialtrennende Stromwandler bis 25 A. Das Strom-Messsignal wird durch einen OPV in Differenzverstärker-Schaltung verstärkt und anschließend gefiltert. Durch eine Komparatorschaltung am Ausgang des OPVs erfolgt eine Fehlergenerierung im Falle eines zu hohen Phasenstroms. Dieser Fehler führt zu einer hardwaretechnischen Abschaltung der PWM-Ausgänge. Der maximal zugelassene Phasenstrom wird über ein digitales Potentiometer vorgegeben.

2.2.4 Verstärker-Messwerte

Weiters wird die Zwischenkreisspannung des Wechselrichters gemessen und auf die Controllerplatine geführt, das Signal verstärkt und ebenfalls gefiltert. Eine hardwaretechnische Fehlergenerierung erfolgt hier ebenfalls durch eine Komparatorschaltung im Falle einer zu hohen bzw. zu niedrigen Zwischenkreisspannung.

Außerdem wird die Temperatur des Kühlkörpers, durch den die Wärme der Wechselrichter-MOSFETs abgeleitet wird, erfasst, um eine Abschaltung im Falle einer zu hohen Temperatur zu ermöglichen.

2.2.5 Fehlerabwicklung

Sämtliche Fehler-Ausgangssignale werden durch Logik-Gatter verknüpft, was eine Ausschaltung der PWM-Ausgänge herbeiführt und ebenfalls die anderen Controller-Einheiten über das Failsafe-Bussystem über den Fehler informiert. So wird eine Abschaltung aller Achsen bei Ausfall einer einzelnen Einheit ermöglicht.

2.2.6 Bus

Auf der Controllerplatine wurden drei Bus-Systeme realisiert:

Main-Bus
Der Main-Bus dient der Kommunikation mit der Haupt-Steuereinheit des Massagebetts. Über den Main-Bus werden die Bewegungsabläufe an die Controller-Einheiten sowie sämtliche Regel-Parameter übertragen. Technisch wurde dieser durch einen CAN-Bus realisiert. Im Kapitel 2.3 wird näher auf die CAN 2.0 Spezifikation eingegangen.

Sensor-Bus
Der Sensor-Bus dient der Positions- und Geschwindigkeitsregelung der Achsen. Auf der Z-Achse erfolgt zusätzlich eine Kraftregelung. Die Umsetzung des Sensor-Systems selbst ist Teil einer anderen wissenschaftlichen Arbeit um die Entwicklung des Massagebetts [1]. Da jede Achse über einen Sensor verfügt, der nur mit der

jeweiligen Controller-Einheit kommuniziert, handelt es sich um ein Bus-System mit nur zwei Busteilnehmern, wobei der Sensor kontinuierlich und ununterbrochen die Messdaten an die Controller-Einheit sendet. Technisch wurde der Sensor-Bus durch einen RS485-Bus im Halbduplex-Betrieb realisiert.

Failsafe-Bus
Der Failsafe-Bus dient der Fehler-Kommunikation und in weiterer Folge einer Not-Abschaltung aller am System angeschlossenen Verstärker. Die technische Realisierung wird im Kapitel 4.2.5 erläutert.

2.2.7 Debugging, Systeminformationen

Systemfehler können durch LEDs angezeigt werden. Eine genauere Angabe von Fehlern sowie weitere Systeminformationen können durch ein 16x2-LCD-Display ausgegeben werden.

2.3 Controller Area Network (CAN)

Die Umstellung des Main-Bus von RS485 auf CAN war eine grundlegende Anforderung dieser Diplomarbeit. Für die technische Umsetzung ist ein Verständnis des CAN-Protokolls Voraussetzung. Es soll nun auf die Grundlagen der Datenübertragung mit einem CAN-Bus eingegangen werden. Für nähere Informationen wird auf die CAN 2.0b Spezifikation der Robert Bosch GmbH verwiesen [5].

Der CAN-Bus wurde 1983 von Bosch entwickelt und ist ein, der Familie der Feldbusse angehöriges, serielles Bussystem. Es wurde ursprünglich für die Vernetzung von Steuergeräten in Automobilen entwickelt, findet heute aber auch in Anwendungen der Medizin-, Automatisierungs- oder Schienenfahrzeugtechnik Verwendung. Der CAN-Bus unterscheidet sich von anderen Bus- bzw. generell Vernetzungssystemen vor allem durch die ausgeprägte Fehlerbehandlung und Echtzeitfähigkeit.

Die maximal mögliche Anzahl der Netzwerk-Knoten ist theoretisch unbegrenzt, wird aber praktisch durch die niedrigere Übertragungsgeschwindigkeit bei längeren Leitungen und durch die verwendeten Bustreiberbausteine (Transceiver) limitiert. Bei einem Highspeed-CAN-Bus beträgt die maximale Datenübertragungsrate 1 Mbit/s. Die (theoretische) Leitungslänge darf, um diese Übertragungsrate zu erreichen, nicht länger als 40 m sein.

2.3.1 Grundlegende Eigenschaften

Priorisierung von Nachrichten
Durch einen *Identifier* im Datenpaket wird eine statische Nachrichtenpriorität während des Buszugriffs definiert. Sollten zwei Knoten zur gleichen Zeit eine Übertragung starten wollen, entscheidet eine bitweise Arbitrierung des Nachrichten *Identifiers* über den Konflikt.

Multicast-Empfang
Durch das Konzept der Nachrichten-Filterung kann jeder Knoten eine gesendete Nachricht empfangen und alle Knoten können simultan auf die gleiche Nachricht reagieren.

Systemweite Datenkonsistenz
In einem CAN-Netzwerk ist es garantiert, dass eine Nachricht entweder von allen oder von keinem Knoten empfangen wird.

Multimaster-fähig
Wenn der Bus frei ist, kann jeder Knoten eine Übertragung starten. Der Knoten mit der Nachricht der höheren Priorität erhält den Buszugang.

Fehlererkennung und –signalisierung
Fehlerüberwachung durch alle Busteilnehmer durch CRC, Bit-Stuffing und Datenpaket-Überprüfung. Korrupte Nachrichten werden von einem Knoten, welcher den Fehler erkennt, signalisiert. Der übertragende Netzwerkknoten überprüft zusätzlich die tatsächlichen Bit-Pegel am Bus mit den zu übertragenden Pegeln.

Automatische, erneute Übertragung von fehlerhaften Nachrichten
Nachdem ein Fehler von einem Knoten signalisiert wurde, wird die Übertragung der Nachricht abgebrochen und automatisch wiederholt.

Automatisches Erkennen und Abschalten von fehlerhaften Netzwerkknoten
CAN-Netzwerkknoten können zwischen kurzfristigen Störungen und permanenten Fehlern unterscheiden – fehlerhafte Knoten werden abgeschaltet.

2.3.2 Bus-Anbindung

Abbildung 2.4 stellt die Vernetzung von CAN-Knoten dar. Jeder Knoten verfügt über einen CAN-Transceiver (Bustreiber), welcher für die Anpassung der Daten an das Übertragungsmedium zuständig ist. Der CAN-Controller ist für das Codieren bzw. Decodieren und Interpretieren der Daten zuständig. In der Praxis werden Basic-CAN-Controller, welche sich nur für Netzwerke mit niedrigen Datenraten eignen, und Full-CAN-Controller, welche sich auch für High-Speed-Netzwerke eignen, unterschieden. Es gibt sowohl eigenständige CAN-Controller sowie in Mikrocontrollern integrierte

CAN-Controller. Der für das Projekt verwendete *ADSP-CM408F* Mikrocontroller verfügt über zwei Full-CAN-Controller, worauf im Kapitel 6.3.6 im Zuge der Programmierung näher eingegangen wird.

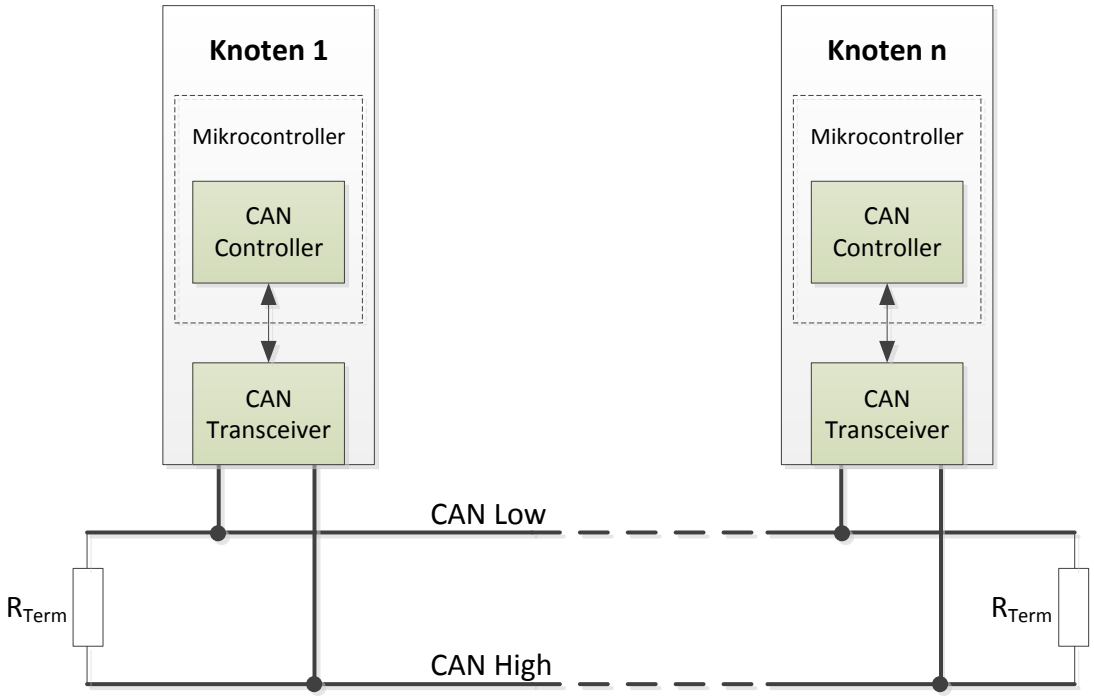


Abbildung 2.4 - Vernetzung von CAN-Knoten

Die Signalleitungen CAN-High und CAN-Low weisen bei „rezessiven“ bzw. „dominanten“ Signalen einen entsprechenden differentiellen Pegel auf.

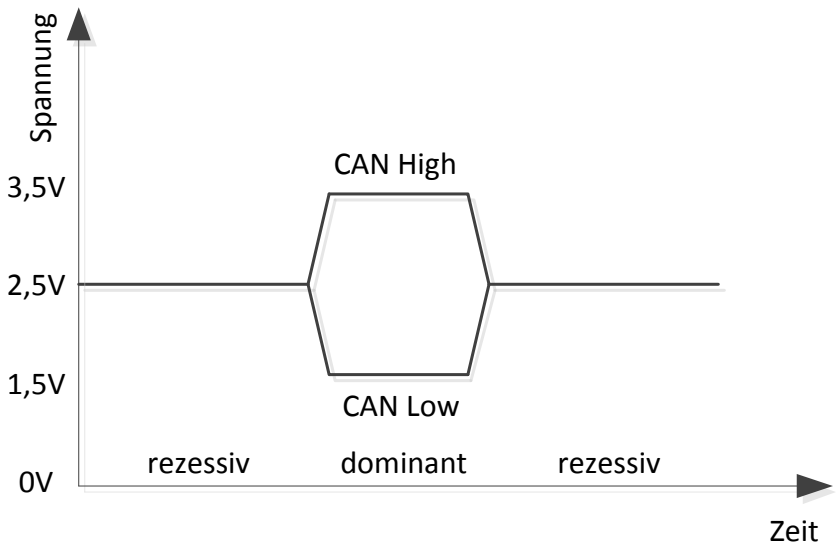


Abbildung 2.5 - CAN-Pegel

Abbildung 2.5 zeigt die nach ISO 11898-2 [6] definierten Bus-Pegel in einem High-Speed CAN-Netzwerk. Der „dominante“ Pegel entspricht einer logischen „0“, der „rezessive“ Pegel einer logischen „1“. „Dominant“, da sich durch die logische UND-

Verknüpfung, der „dominante“ Pegel eines einzelnen Knoten am Bus durchsetzt und andere „rezessive“ Pegel der weiteren Busteilnehmer überschreibt.

2.3.3 CAN Frames

Prinzipiell unterscheidet man vier verschiedene Arten von CAN Frames:

Remote Frame
Mit einem solchen Frame können von einem Knoten Daten mit einem bestimmten *Identifier* angefordert werden.

Error Frame
Wird von einem Knoten übertragen, sobald dieser einen Fehler erkennt

Overload Frame
Wird von einem Knoten an den Bus gesendet, um eine zusätzliche Verzögerung zwischen dem vorigen und nächsten Frame (Daten- oder Remote-Frame) mitzuteilen.

Error und Overload Frames werden nur vom CAN-Controller erzeugt, der Entwickler hat auf die Generierung dieser Frames keinen Einfluss.

Daten Frame
Gewöhnliches Datenpaket für Nutzdaten. Der Aufbau eines Daten Frames wird in Abbildung 2.6 dargestellt. Man unterscheidet zwischen einem Standard-Data-Frame, welcher einen 11-bit *Identifier* im Arbitrationsfeld enthält, und einem Extended-Data-Frame, welcher einen 29-bit *Identifier* enthält.

Abbildung 2.6 - CAN Daten-Frame

Die einzelnen Felder eines Daten Frames setzen sich wie folgt zusammen:

Feldbezeichnung	Bit	Beschreibung
Start of Frame	1	Anfang einer Nachricht
Arbitrationsfeld	12 / 32	Länge und Zusammensetzung abhängig vom Format des Daten Frames. Standard: 11-bit Identifier und 1-bit RTR (Remote Transmission Request) Extended: 29-bit Identifier, jeweils 1-bit SRR (Substitute Remote Request), RTR und IDE (Identifier Extension).
Kontrollfeld	6	Beinhaltet die Anzahl der übertragenen Datenfelder
Datenfeld	max. 64	Nutzdaten mit einer Länge von 0 bis 8 Byte. Die Übertragung erfolgt vom MSB zum LSB.
CRC (Sicherungsfeld)	16	Der Cyclic Redundancy Check dient der Erkennung von Übertragungsfehlern.
ACK (Bestätigungsfeld)	2	Jeder Empfänger bestätigt den Erhalt des Frames indem er das rezessive Bit des Senders mit einem dominanten Bit überschreibt. Dies muss im ersten Bit (ACK Slot) des Bestätigungsfeldes geschehen.
End of Frame	7	Ende der Nachricht – besteht aus 7 rezessiven Bits.
Ruhezustand	3	Während der Ruhephase dürfen die Knoten nur einen *Overload* signalisieren, aber keine neuen Nachrichten senden oder anfordern.

Tabelle 2.1 - CAN-Frame

Mit einer Daten-Frame-Länge von höchstens 131 Bit im Extended-Format und 111 Bit im Standard-Format ergibt sich somit eine theoretische Nutzdatenrate (im Falle fehlerfreier Übertragung) von ca. 49 % respektive 58 %.

2.3.4 Akzeptanzfilterung

Da jeder CAN-Knoten alle Nachrichten am Bus empfängt, erfolgt durch den CAN-Controller eine Akzeptanzfilterung, um die vom Mikrocontroller zu bearbeitende Datenmenge zu reduzieren. Dies erfolgt durch eine entsprechende Programmierung des Filterregisters auf einen *Identifier*. So werden Nachrichten, die für einen bestimmten Knoten uninteressant sind, gar nicht erst in die Nachrichtenregister gespeichert. Abhängig vom CAN-Controller bzw. vom verwendeten Mikrocontroller kann somit ohne Core-Intervention auf den Erhalt einer bestimmten Nachricht reagiert werden. Die Umsetzung und Programmierung, des im Projekt verwendeten CAN-Controllers wird in Kapitel 6.3.6 erläutert.

3 Eignung und Vergleich der Prozessoren

Die Neuentwicklung der Controllerplatine wird unter anderem durch die fehlende Rechenleistung der Vorgängerversion begründet. Aufgrund rechenintensiver Operationen musste die Programmierung hardwarenah optimiert werden, und lässt nun so gut wie keinen Spielraum mehr in der Einbindung neuer Funktionen und anderen Abläufen zu. Zum Einsatz kommt auf der Vorgänger-Controllerplatine ein DSP der Delfino-Serie von *Texas Instruments.* Der DSP vom Typ *TMS320F28335* wurde 2007 veröffentlicht und soll in der neuen Version der Controllerplatine durch einen leistungsstärkeren Prozessor ersetzt werden. Wie bereits im vorigen Kapitel erwähnt, soll ein ARM Cortex-M4 Prozessor zum Einsatz kommen. Die Wahl fällt auf den *ADSP-CM408F* von *Analog Devices* – ein Mikrocontroller auf dem neuesten Stand der Technik.

3.1 Vergleich der Mikrocontroller

Tabelle 3.1 zeigt einen Vergleich der beiden Mikrocontroller mit den wichtigsten Funktionen und Eigenschaften.

Funktion		TMS320F28335	ADSP-CM408F
Core Takt (max.)		150 MHz	240 MHz
Flash		256 KB	2048 KB
RAM		34 KB	384 KB
ADC	Auflösung	12 Bit	16 Bit
	Samplingrate	12.5 MSPS	2.63 MSPS
	Wandlungszeit	80 ns (bei 25MHz ADC-Takt)	380 ns (bei 50Mhz ADC-Takt)
	Eingänge	16 Kanäle (1 ADC)	Insgesamt 24 Kanäle (2 ADCs)
GPIO	Anzahl	88	91
	interruptfähig	8	91
Interrupt-Priorisierung		Nein	16 Prioritäten
Floating Point Unit		Single precision	Single precision
DMA		6 Kanäle	17 Kanäle
MDMA		Nein	2 Kanäle
GP Timer (32-bit)		3	8
PWM-Modul		6 Ausgänge	4x 3-Phasen-PWM (12 PWM-Paare)
Inter-Integrated Circuit (I^2C)		1	1
UART-Ports		3	3

CAN-Controller	2	2
USB-Controller	-	USB 2.0 Device Controller
Ethernet	-	10/100 Ethernet MAC Controller
Gehäuse	176-Pin LQFP	176-Pin LQFP (mit GND-Pad)

Tabelle 3.1 - Mikrocontroller Vergleich [4], [7]

Dies sind nur die wichtigsten Funktionen der Mikrocontroller. Vor allem der *CM408F* verfügt noch über eine Vielzahl weiterer Funktionen, die dem *Preliminary Technichal Data-Sheet* der CM40x Mikrocontroller-Serie entnommen werden können [4].

Unschwer zu erkennen ist, dass durch den technischen Fortschritt in der Chipherstellung, eine größere Anzahl an Funktionen und Modulen in einem Gehäuse derselben Größe untergebracht werden können. Ein klarer Unterschied, zum Nachteil des *CM408F*, ist allerdings in der ADC-Geschwindigkeit zu vermerken. Bei höherem Takt ist der ADC des *CM408F* deutlich langsamer. Dies ist auf die ADC Architektur zurückzuführen. Der *CM408F* verwendet einen high-speed SAR (successive approximation register) ADC während der Delfino einen Konverter mit einer 12-bit vierstufigen Pipeline Architektur (Parallelverfahren) verwendet. Eine Samplingrate von 2,64 MSPS ist für das Projekt aber prinzipiell ausreichend und der *CM408F* ist mit der höheren ADC-Auflösung von 16 Bit gegenüber den 12 Bit des Delfino im Vorteil. Weiters verfügt der *CM408F* über zwei ADCs, die theoretisch eine simultane Erfassung zweier Kanäle bei voller Samplingrate ermöglichen. Wie diese Funktion praktisch umgesetzt wird, wird in Kapitel 6.3.4 ausgeführt.

3.2 Benchmark

Um Sicherzustellen, dass die Rechenleistung des *CM408F* für die Durchführung der rechenintensiven Operationen im Programmablauf ausreichend ist und in welchem Ausmaß die Steigerung gegenüber dem Delfino ausfällt, wurde ein Prozessor-Benchmark durchgeführt.

3.2.1 Implementierung

Die rechenintensivste Operation im Programmablauf der Controller-Einheit ist die digitale Filterung der Sensordaten. Für den Benchmark wurde ein FIR-Filter mit 48 Koeffizienten implementiert.

Der Ausgangswert bzw. das Ergebnis $y(n)$ des FIR-Filters ergibt sich durch Aufsummierung der Multiplikationsergebnisse der Eingangswerte $x(n), \dots, x(n-N)$ mit den $N+1$ Filterkoeffizienten b_i.

$$y(n) = \sum_{i=0}^{N} b_i \cdot x(n-i)$$

Es wird schnell deutlich, dass ein FIR-Filter eine rechenintensive Funktion ist, da für jeden neuen Eingangswert $N+1$ Multiplikationen und Additionen durchgeführt werden müssen. Der Rechenaufwand steigt mit der Anzahl der Filterkoeffizienten. Die Tatsache, dass es sich in der Praxis bei den Eingangswerten und Koeffizienten um Gleitkommazahlen handelt, ist ein weiterer Nachteil hinsichtlich der benötigten Rechenleistung. Es ist also von Vorteil, wenn der Prozessor über eine FPU verfügt.

Bei der Umsetzung des Programms sollte auch auf die Anzahl der Speicher-Schreibvorgänge geachtet werden, da es nicht das Ziel ist, die Leistung des Arbeitsspeichers zu messen. Wird ein linearer Buffer für die Eingangswerte programmiert (siehe Abbildung 3.1), so müssen alle vorangegangenen Werte im Speicher verschoben werden, sobald ein neuer Eingangswert vorliegt. Das heißt für $N+1$ Koeffizienten erfolgen ebenso viele Schreibvorgänge im Arbeitsspeicher.

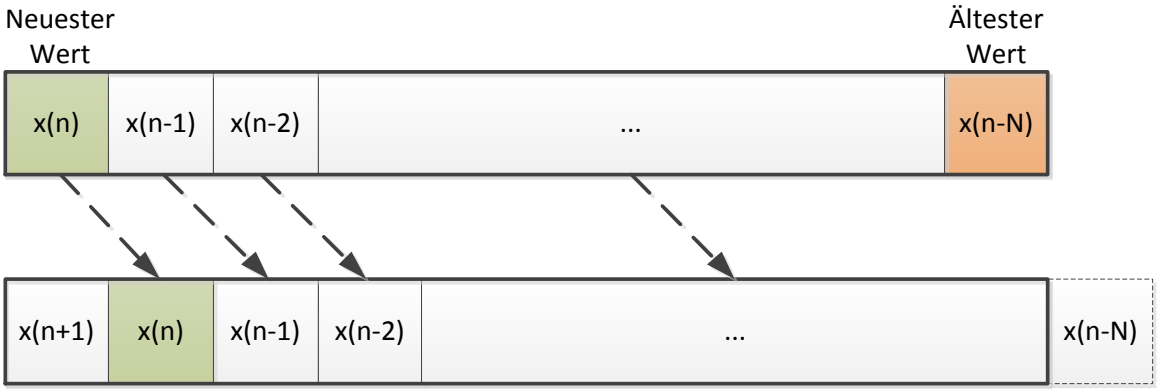


Abbildung 3.1 - Linearer Speicher

Eine deutlich effizientere Methode wird über einen Zirkular- bzw. Ringbuffer implementiert. Das Konzept eines solchen Buffers ist in Abbildung 3.2 dargestellt. Durch eine zusätzliche Variable wird die aktuelle Zeigerposition gespeichert und beim Eingang eines neuen Wertes inkrementiert. Stimmt die Anzahl der Felder des Zirkularbuffers mit der Anzahl der Filterkoeffizienten überein, wird immer nur der älteste Wert mit dem neuesten überschrieben. Die Speicherschreibvorgänge werden so auf ein Minimum reduziert. Steigt die Anzahl der Filterkoeffizienten, so steigt zwar die Anzahl der erforderlichen Speicherzellen, die Anzahl der Speicherschreib-vorgänge (nach dem ersten Durchlauf) bleibt aber gleich.

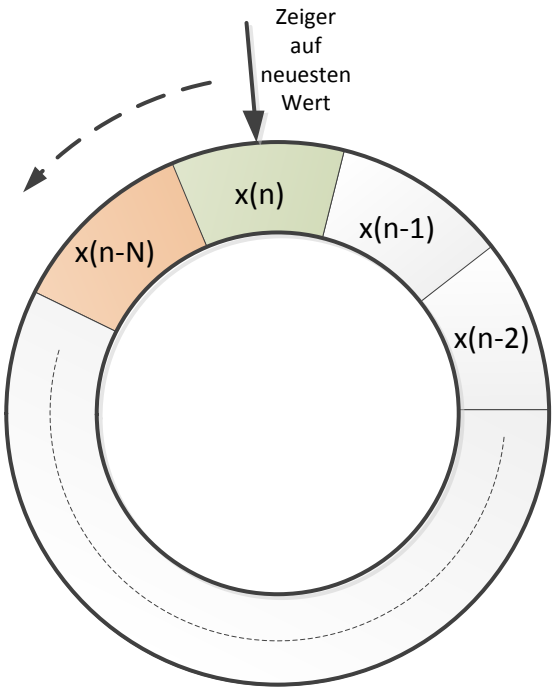


Abbildung 3.2 - Zirkularbuffer

Für den Benchmark wurde eine Testfunktion programmiert, die einen FIR-Filter mit Gleitkommazahlen einfacher Genauigkeit (*float*) und einem Zirkularbuffer implementiert. Es werden 48 Filterkoeffizienten verwendet, die zufällige Werte aufweisen. Die Filterfunktion wird 100-mal durchlaufen (dies entspricht 100 neuen Messwerten), und signalisiert den jeweiligen Durchlauf über einen GPIO-Pin. Am Oszilloskop wird die Zeit für einen Testdurchlauf gemessen und gibt somit Auskunft über die Rechenleistung des Mikrocontrollers.

3.2.2 Ergebnisse

Der Kern des *CM408F* ist mit 240 Mhz gegenüber 150 Mhz um 60 % schneller getaktet – der wahre Vorteil liegt aber in der ARM-Cortex-Architektur des Prozessors. So sollte eine Steigerung der Rechenleistung deutlich über der 60 % Marke liegen.

Im ersten Vergleich wurden die Funktionen aus dem Flash-Speicher ausgeführt. Nach jedem Durchlauf, d. h. nach der Ausführung von 4800 Multiplikationen und Additionen, wird der logische Zustand eines GPIO-Pin verändert. Demnach entspricht die Pulsweite des Signalverlaufs einem Funktionsdurchlauf. In Abbildung 3.3 und Abbildung 3.4 wird der Leistungsunterschied deutlich. Der *CM408F* benötigt für einen Funktionsdurchlauf ca. 4,78 µs während der Delfino für die Ausführung derselben Funktion 26,69 µs benötigt.

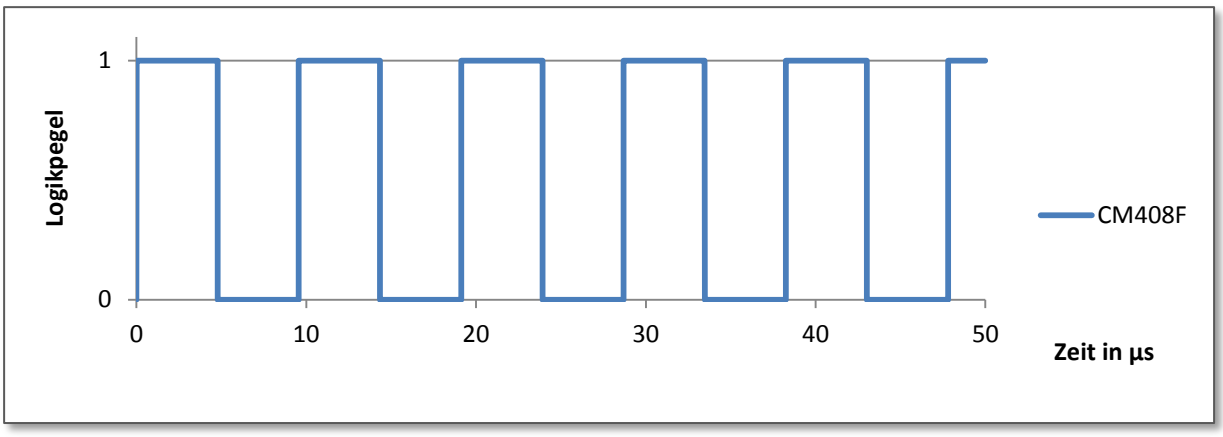


Abbildung 3.3 - Benchmark CM408F (Flash)

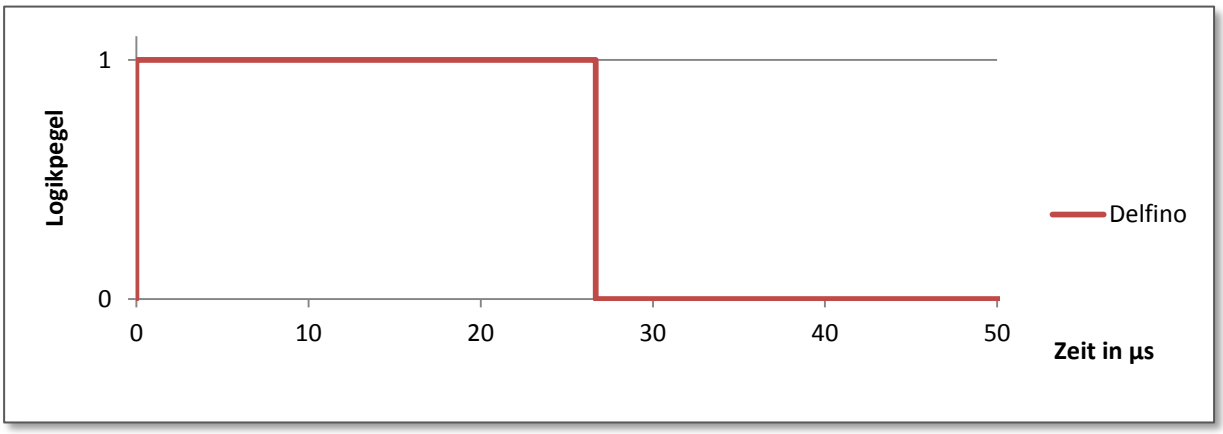


Abbildung 3.4 - Benchmark Delfino (Flash)

Der *CM408F* verfügt mit 384 KB über deutlich mehr Arbeitsspeicher als der Delfino (34KB). Daher besteht prinzipiell die Möglichkeit, wichtige und oft aufgerufene Funktionen aus dem RAM auszuführen. Dazu wird ein Adressbereich des Arbeitsspeichers für Funktionen definiert und reserviert. Beim Bootvorgang werden die als „RAM-Funktionen“ definierten Funktionen in den Arbeitsspeicher kopiert. Abbildung 3.5 und Abbildung 3.6 zeigen den Zeitgewinn, wenn die FIR-Filter-Funktionen aus dem RAM ausgeführt werden.

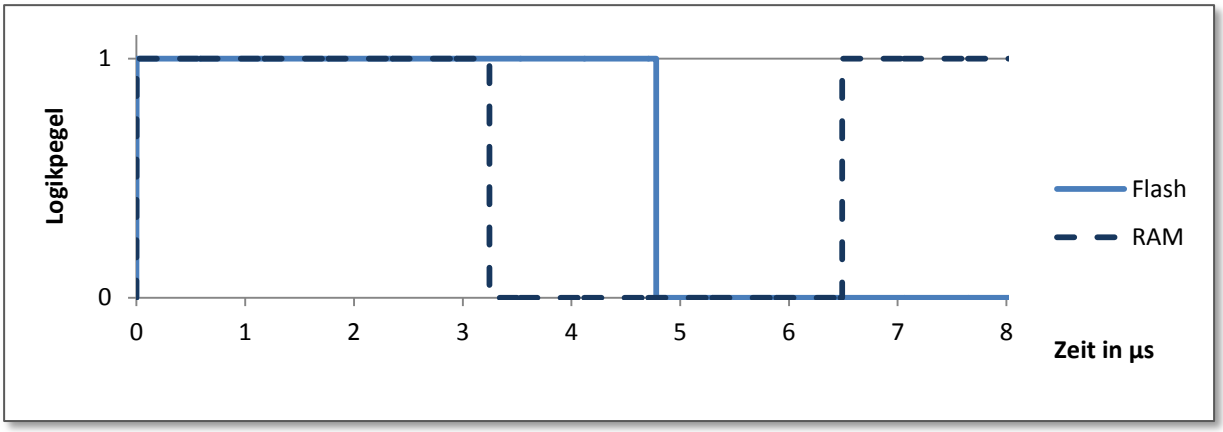


Abbildung 3.5 - Benchmark CM408F (RAM)

Für Vergleiche wurde der Benchmark auch am Delfino aus dem Arbeitsspeicher ausgeführt:

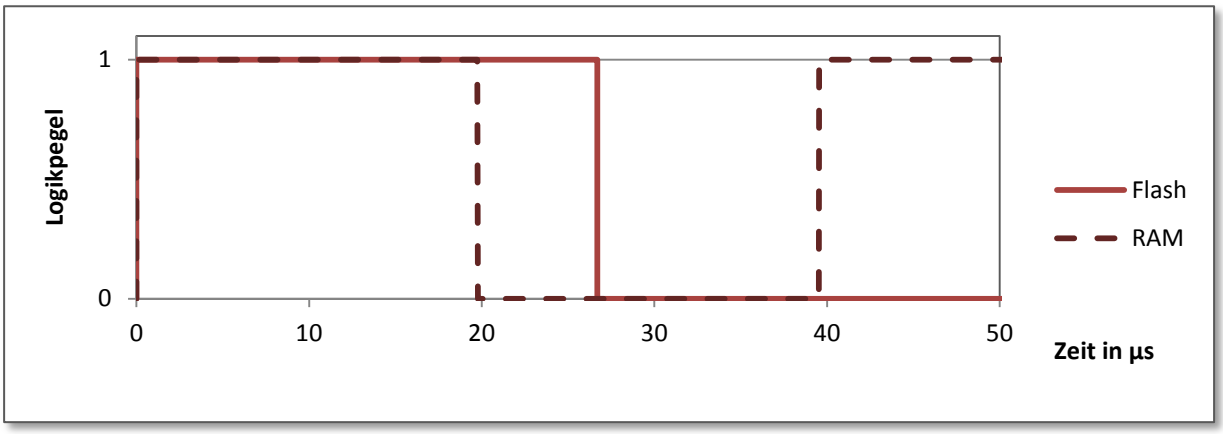


Abbildung 3.6 - Benchmark Delfino (RAM)

Der Funktionsdurchlauf wird bei beiden Prozessoren deutlich beschleunigt, wobei auch hier der *CM408F* mit einer Zeitersparnis von etwa 32 % gegenüber 26 % beim Delfino eine geringere Rechenzeit ermöglicht.

Quantitativ spielt hier die benötigte Zeit für einen Funktionsdurchlauf eine untergeordnete Rolle, da die Anzahl der Filterdurchläufe für den Benchmark willkürlich gewählt wurde, vielmehr war das Ziel einen qualitativen Vergleich der beiden Prozessoren zu ermöglichen. Tabelle 3.2 fasst den Benchmark der beiden Mikrocontroller zusammen und zeigt ein sehr erfreuliches Ergebnis hinsichtlich der gesteigerten Rechenleistung.

Beschreibung	TMS320F28335	ADSP-CM408F
Zeit für Funktionsdurchlauf (Flash)	26,69 µs	4,778 µs
Zeit für Funktionsdurchlauf (RAM)	19,74 µs	3,246 µs
Zeitersparnis durch RAM-Funktionen	26,04 %	32,06 %
Rechenleistung (RAM-Funktionen)	100 %	608 %

Tabelle 3.2 - Benchmark-Ergebnisse

4 Entwurf des Schaltplans

In diesem Kapitel soll systematisch auf den Schaltungsaufbau der Controllerplatine (siehe Abbildung 2.2) eingegangen werden. Die Funktionen der wesentlichen Komponenten werden in Blockdiagrammen aufgezeigt, da die Beschreibung aller Bauteile den Rahmen dieser Diplomarbeit sprengen würde. Der vollständige, mit *Altium Designer* [8] erstellte, Schaltplan befindet sich im Anhang der Arbeit.

4.1 Analogteil

4.1.1 Messung der Motor-Phasenströme

Die Strommessung legt die Basis für die Regelung des Antriebs und erfolgt hier durch drei potentialtrennende Stromwandler (Typ *LEM CASR-25-NP*) auf der Verstärkerplatine. Diese ermöglichen eine Messung bis 25 A und liefern eine Referenzspannung $U_{Ref} = 2{,}5\,V$ sowie eine zum Eingangsstrom proportionale Ausgangsspannung $U_{Out} = U_{Ref} \pm 25\,mV/A$. Der Messbereich muss an den ADC angepasst werden. Die Signalaufbereitung wird in Abbildung 4.1 dargestellt.

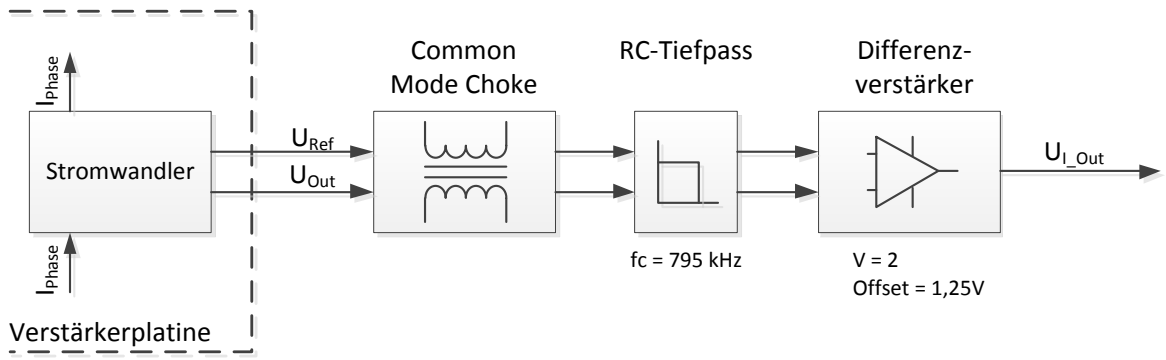


Abbildung 4.1 - Strommessung

Im ersten Schritt werden die Signale vom Ausgang des Stromwandlers durch eine stromkompensierte Drossel (*Common Mode Choke*) gefiltert. Das Funktionsprinzip einer stromkompensierten Drossel ist in Abbildung 4.2 dargestellt. Durch die gleiche Anzahl Wicklungen in die entgegengesetzte Richtung wird die gewünschte Wirkung erzeugt.

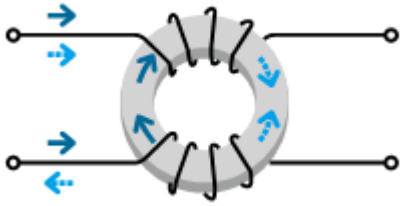


Abbildung 4.2 - Stromkompensierte Drossel

→ Da sich der magnetische Fluss von Gleichtaktströmen aufsummiert, wird in
→ der Spule eine hohe Impedanz für Gleichtaktsignale erzeugt.

→ Der magnetische Fluss von Gegentaktströmen hebt sich auf, damit bleibt
← der Widerstand für Gegentaktsignale gering.

Stromkompensierte Drosseln werden zur Unterdrückung von Gleichtaktstörungen bevorzugt eingesetzt, da Anwendungen mit hoher Impedanz leicht erzielt werden können. Abbildung 4.3 zeigt eine typische Impedanz-Charakteristik einer solchen Drossel.

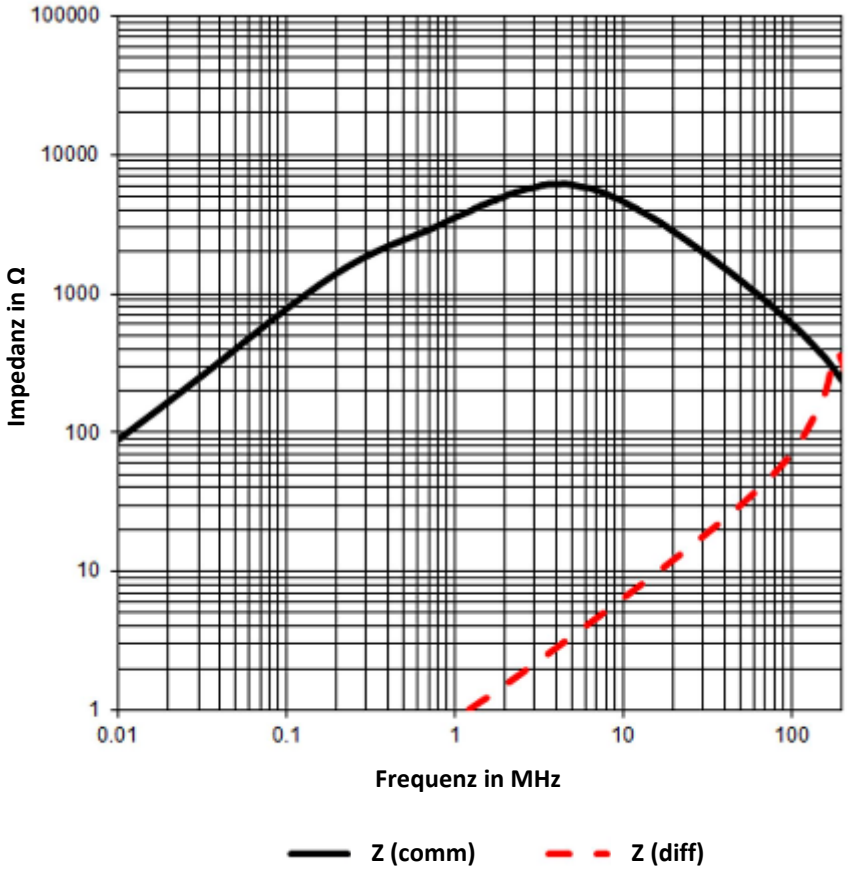


Abbildung 4.3 - Impedanzcharakteristik einer stromkompensierten Drossel [9]

Durch die hohe Gleichtakt-Impedanz der Drossel, die im MHz-Bereich einige kΩ beträgt, werden Störungen erfolgreich unterdrückt, während das differenzielle Nutzsignal im kHz-Bereich nahezu unbeeinflusst bleibt.

Vor dem Eingang des OPVs in Differenzverstärker-Schaltung wurde noch ein einfacher RC-Tiefpass mit einer Grenzfrequenz von ca. 800 kHz vorgesehen, um eine weitere Filterung der Eingangssignale zu ermöglichen. Die maximal messbare Eingangsspannung am ADC des verwendeten Mikrocontrollers beträgt 2,5 V. Um den Messbereich bestmöglich auszunutzen, wird für das Strommesssignal eine Verstärkung von 2 und ein Offset von 1,25 V gewählt. Bei einem Phasenstrom im Bereich von ±25 A ergibt sich somit eine Ausgangsspannung im messbaren Bereich von 0 V bis 2,5 V mit $U_{I\_Out} = 1{,}25\,V \pm 50\,mV/A$.

4.1.2 Überstrom Notabschaltung

Um eine softwareunabhängige Notabschaltung des Motors zu gewährleisten, wurde eine Komparatorschaltung implementiert. Abbildung 4.4 zeigt diese Schaltung für eine Motorphase.

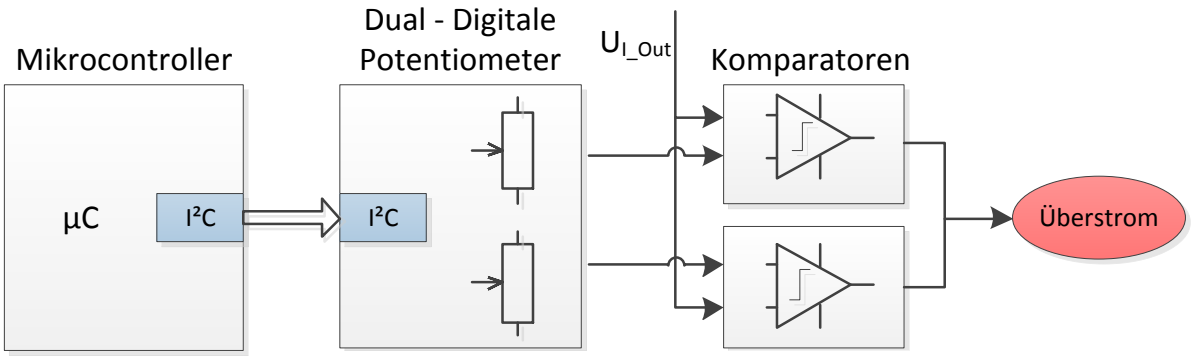


Abbildung 4.4 - Überstrom Notabschaltung

Die Ausgangsspannung der Verstärkerschaltung aus 4.1.1 wird an den nicht-invertierenden Eingang des einen bzw. an den invertierenden Eingang des anderen Komparators angelegt. Der maximal zulässige Strom wird durch, per I²C-Bus programmierbare, digitale Potentiometer vorgegeben. Wegen der „Open-Collector“ Ausgänge der Komparatoren können alle Ausgänge (6 Komparatoren für 3 Phasen) über einen Pull-Up-Widerstand zusammengeschlossen werden und bilden so eine logische UND-Verknüpfung, die ein Active-Low Fehlersignal generiert.

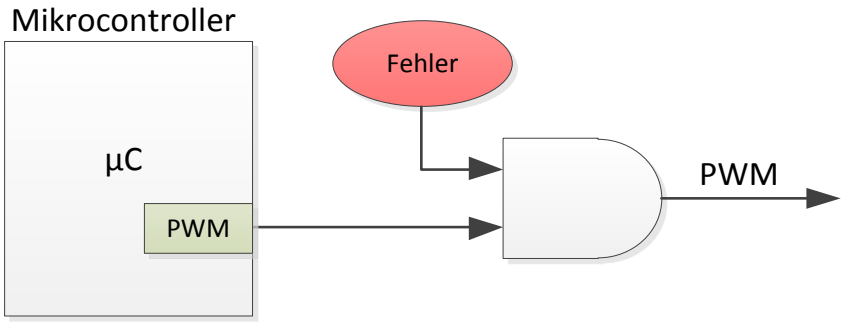


Abbildung 4.5 - PWM Abschaltung

Der Überstrom-Fehler wird mit weiteren Fehlerausgängen anderer Funktionen verknüpft (siehe Kapitel 4.2.4). Das Active-Low Fehlersignal stoppt somit, im Falle eines Fehlers, die Ansteuerung mit PWM-Signalen des Mikrocontrollers (Abbildung 4.5).

4.1.3 Wechselrichter-Zwischenkreisspannung

Während des Betriebes soll die Zwischenkreisspannung des Wechselrichters kontrolliert werden um im Falle einer zu hohen bzw. zu niedrigen Spannung eine Abschaltung zu generieren.

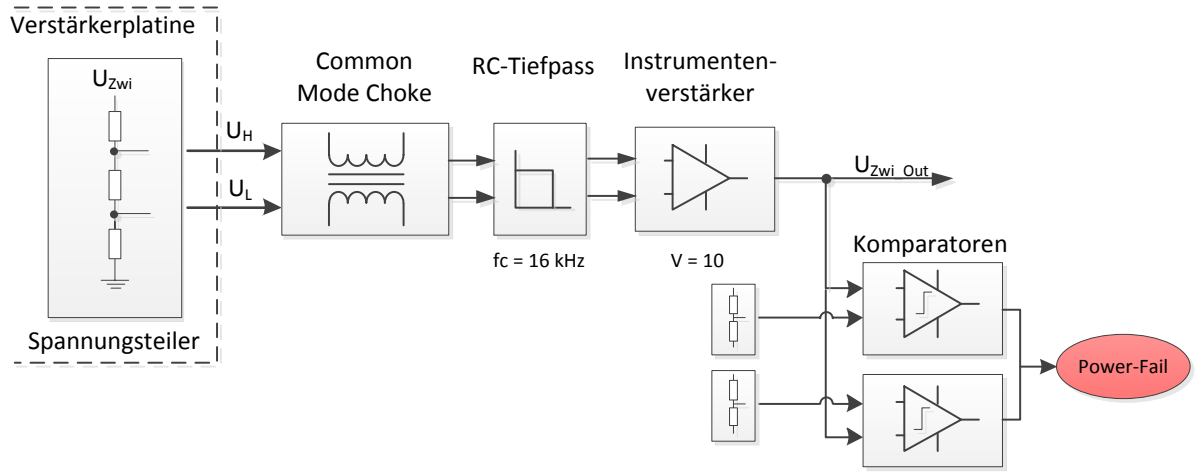


Abbildung 4.6 - Zwischenkreisspannung

Die Messung der Zwischenkreisspannung erfolgt durch einen Spannungsteiler auf der Verstärkerplatine. Durch den Spannungsabfall über einen Widerstand kann auf die Zwischenkreisspannung zurückgerechnet werden. Wie auch bei der Messung der Phasenströme wurden hier eine stromkompensierte Drossel und ein RC-Tiefpass zur Störungsunterdrückung vorgesehen. Die Differenzspannung wird durch einen Instrumentenverstärker mit einer Verstärkung von 10 verstärkt. Die Grenzen für die erlaubte Zwischenkreisspannung werden durch Spannungsteiler mit festen Widerständen eingestellt. Die maximale Zwischenkreisspannung wurde mit 50 V, die Minimale mit 24 V festgelegt. Die Open-Collector Komparatoren erzeugen ein Active-Low Fehlersignal bei Über- bzw. Unterschreiten dieser Grenzen.

4.1.4 Verstärker-Temperatur

Die MOSFETs des Wechselrichters sind, wie in Abbildung 4.7 dargestellt, an einem gemeinsamen Kühlkörper angebracht. Die Temperatur wird durch einen integrierten Temperatursensor (*LM35*) gemessen. In der implementierten Beschaltung des Sensors ist eine einfache Temperaturmessung von 2 °C bis 150 °C möglich. Der Sensor ist bei Auslieferung direkt auf die Celsius-Skala kalibriert und weist eine

lineare Kennlinie auf, wobei die Ausgangsspannung $U_{T\_Out} = 0\,V + 10\,mV/°C$ [1] beträgt.

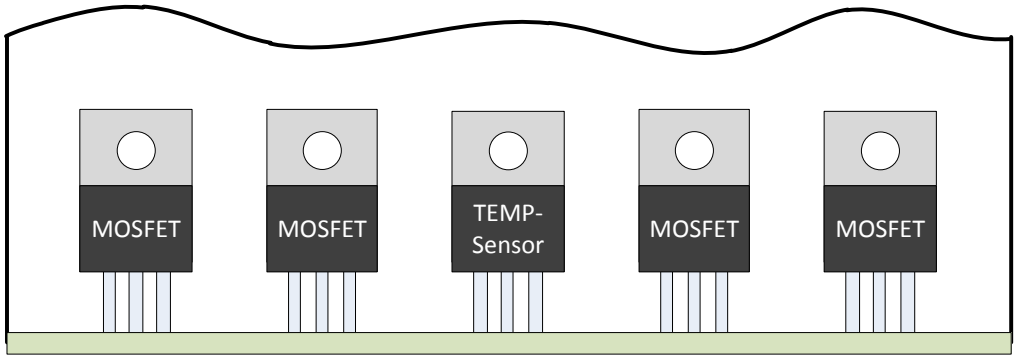


Abbildung 4.7 - Montage Temperatursensor

Bei der Temperaturmessung muss keine hohe Genauigkeit erreicht werden. Die Messung dient lediglich der Notabschaltung des Verstärkers bei zu hoher Temperatur. Im ununterbrochenen Betrieb des Versuchsaufbaus über mehrere Stunden wurden keine nennenswerten Temperaturerhöhungen des Verstärkers festgestellt. Die maximal erlaubte Temperatur des Kühlkörpers wird auf 70 °C ausgelegt.

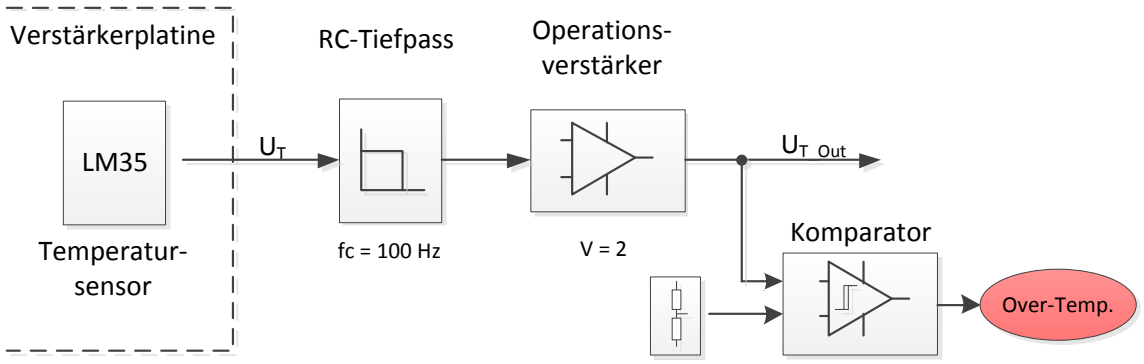


Abbildung 4.8 - Temperaturmessung

Da die Temperaturveränderung ein verhältnismäßig langsamer Prozess ist, wurde der RC-Tiefpass mit einer Grenzfrequenz von 100 Hz sehr tief abgestimmt und weiters der Komparator mit einer Hysterese ausgelegt, um im Grenzbereich ein mehrfaches Triggern des Fehlerausgangs zu verhindern.

[1] Theoretisch beträgt die Ausgangsspannung 0V bei einer Temperatur von 0°C. Praktisch ist die minimale Ausgangsspannung laut Datenblatt aber bei 2°C erreicht.

4.1.5 Tiefpass-Filter

Zur Filterung der analogen Signale, die ihren Ursprung auf der Verstärkerplatine haben, kommen aktive Tiefpassfilter 4. Ordnung vom Typ Linear Phase 0.05° mit einer Grenzfrequenz von 110 kHz zum Einsatz. Die Filterauslegung war Teil bisheriger Arbeiten am Massagebett und hat sich im Versuchsaufbau bewährt – daher wurde die Filterschaltung für den neuen Schaltplan übernommen (siehe Anhang). Der Aufbau erfolgt mit zwei integrierten Dual-OPV-Bausteinen, da die Pinbelegung der Bauteile so kürzere Leiterbahnen und einen symmetrischen Aufbau der Filter ermöglichen. Da die meisten integrierten Operationsverstärker in der gleichen Gehäusebauform pinkompatibel sind, wird ein nachträglicher Austausch der Bauteile ermöglicht, falls die gewählten Operationsverstärker sich in der Praxis als ungeeignet herausstellen.

Um die Signalqualität im Vergleich zur alten Controllerplatine zu verbessern, wurde ein Tausch der Operationsverstärker des Tiefpassfilters in Erwägung gezogen. Durch die höhere Auflösung des ADCs werden auch höhere Anforderungen an die Signalaufbereitung gestellt. Das LSB des 16-bit ADCs mit einem Messbereich von 0 V bis 2,5 V beträgt nur etwa 38 µV. Es wurden mehrere OPVs verglichen – die Wahl fiel auf einen „Auto-Zeroed“ Operationsverstärker. Im Vergleich zu anderen OPVs hat ein „Auto-Zeroed“ OPV eine wesentlich geringere „Input Offset Voltage“. Der ausgewählte *MCP6V27* von *Microchip* hat im Vergleich zum *TLV2732* von *Texas Instruments*, der auf der alten Controllerplatine zum Einsatz kommt, mit einer typischen Offsetspannung von ±2 µV, eine wesentlich geringere Input Offset Voltage[2] [10], [11].

Dieser OPV wurde im Single-Channel Gehäuse auch für den Differenzverstärker in der Strommessschaltung gewählt, siehe Kapitel 4.1.1.

4.1.6 ADC-Eingang

Der interne Aufbau des ADC-Eingangs ist vereinfacht in Abbildung 4.9 dargestellt. Vor allem die Notwendigkeit eines RC-Glieds am Eingang muss genauer untersucht werden. Prinzipiell ist durch den aktiven Tiefpassfilter eine Signalquelle mit niedriger Ausgangsimpedanz gegeben. Durch den gemultiplexten Aufbau des ADCs, ist es prinzipiell nicht vorhersagbar, welche Spannung vom vorherigen Sampledurchgang am internen Sample-Hold-Kondensator (C_{SH}) anliegt. Schließt der „Schalter“ (vereinfachte Darstellung, S1) so muss C_{SH} innerhalb der Samplezeit (Acquisition time) der Sample-Hold-Schaltung auf 0,5 LSB der Eingangsspannung geladen werden, um die maximale Genauigkeit des ADCs zu erreichen. Im Worst-Case-Szenario müsste der Kondensator von der minimalen auf die maximale Eingangs-

[2] Die maximalen Input Offset Voltage beträgt 3 mV beim TLV2732.

spannung innerhalb der Samplezeit von 150 ns geladen werden. Bei einem Messbereich von 0 V bis 2,5 V wären das also 99,99924 % von 2,5 V um innerhalb der 16-Bit Genauigkeit zu liegen.

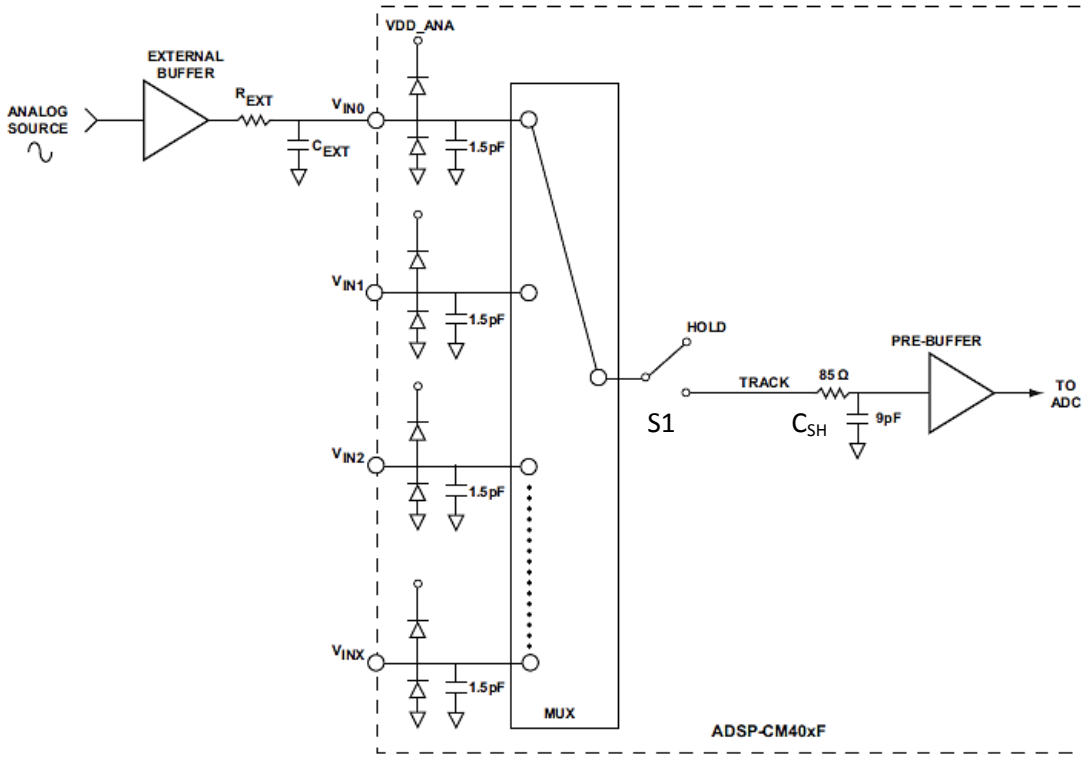


Abbildung 4.9 - CM408F ADC-Eingang [4]

Der Vorteil eines externen Puffer-Kondensators (C_{EXT}) liegt nun darin, den internen Sample-Hold-Kondensator schnell laden zu können. Weiters kann C_{EXT} eine Ladung von C_{SH} auffangen, für den Fall, dass die Eingangsspannung niedriger ist, als die am Sample-Hold-Kondensator vom vorherigen Sampledurchgang. Die Kapazität von C_{SH} beträgt beim verwendeten ADC 9 pF. Es soll nun die ideale Kapazität des externen Kondensators ermittelt werden.

Ladungstransfer zwischen C_{EXT} und C_{SH} bei vollem Messbereich von 0 V auf 2,5 V:

$$Q = C \cdot V$$

$$Q_{SH} = C_{SH} \cdot V_{FSR}$$

$$Q_{SH} = 9\,pF \cdot 2{,}5\,V = 22{,}5\,pC$$

Idealerweise kann nun C_{EXT} diese Ladung bereitstellen während der Spannungsabfall nur 0,5 LSB des vollen Messbereichs beträgt.

$$Q_{EXT} = Q_{SH}$$

$$Q_{EXT} = C_{EXT} \cdot V_{1/2\,LSB}$$

$$C_{EXT} = \frac{Q_{EXT}}{V_{1/2\ LSB}} = \frac{22{,}5\ pC}{19{,}073\ \mu V} = 1{,}180\ \mu F$$

Ein Kondensator mit ca. 1µF ist allerdings aus mehreren Gründen problematisch. Dies ist eine zu hohe kapazitive Last für den OPV (des Tiefpassfilters) und kann zu Stabilitätsproblemen führen. Laut Datenblatt [10] des verwendeten OPVs wird ein Isolationswiderstand (hier R_{EXT}) bei einer solchen kapazitiven Last zwingend notwendig, um die Phasenreserve der Rückkopplung zu verbessern. Ein empfohlener Widerstandwert von R_{EXT} liegt bei dieser kapazitiven Last bei etwas weniger als 100 Ω (siehe Abbildung 4.10 mit $G_N = 2$). Dies würde zu einer zu großen Zeitkonstante von R_{EXT} | C_{EXT} führen und somit einem RC-Tiefpass mit einer viel zu tiefen Grenzfrequenz von etwa 1600 Hz.

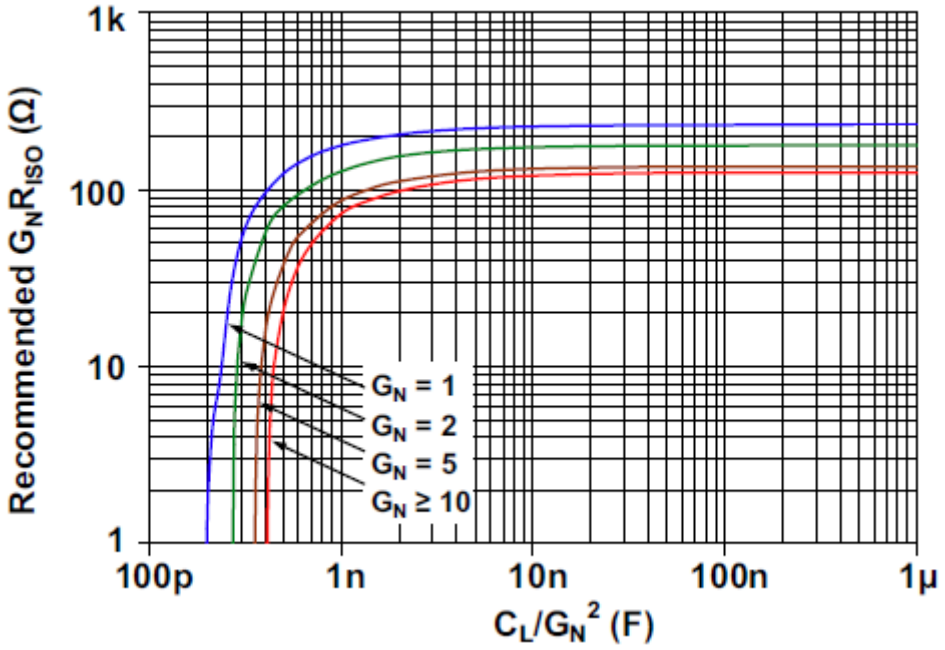


Abbildung 4.10 - MCP6Vx, empfohlener R_{ISO}

In der Praxis wird der interne Sample-Hold-Kondensator nicht allein durch den Puffer-Kondensator geladen, sondern der OPV wird auch einen Teil der Ladung bereitstellen. Das Ausgangsverhalten des OPVs bei einer transienten Last ist hierfür ausschlaggebend. Leider ist dies eine Eigenschaft, die sich nur selten in den jeweiligen Datenblättern dokumentiert ist. Eine Auslegung erfolgt nach der maximal vertretbaren kapazitiven Last unter der Beachtung der Faustregel $C_{EXT} \geq 20 \times C_{SH}$. Typische Werte für C_{EXT} liegen im Bereich von 220 pF bis 3 nF. Dieser Kondensator sollte möglichst kleine Spannungs- und Frequenzkoeffizienten haben. Hier bieten sich Keramikkondensatoren vom Typ C0G (Dielektrikum) an.

Mit der gewählten Kapazität kann nun der passende Widerstandswert für R_{EXT} ermittelt werden. Ausschlaggebend hierfür ist die Zeitkonstante des RC-Glieds.

Durch die Ladekurve des Kondensators wird ermittelt, welcher Faktor k der Zeitkonstante benötigt wird, damit die Kondensatorspannung innerhalb der 0,5 LSB Genauigkeit liegt.

$$U_c(t) = U_0 \cdot \left(1 - e^{-\frac{t}{\tau}}\right)$$

$$V_{FSR} - V_{1/2\ LSB} = V_{FSR} \cdot \left(1 - e^{-\frac{\tau \cdot k}{\tau}}\right)$$

Tabelle 4.1 gibt für verschiedene ADC-Auflösungen an, welcher Faktor k der Zeitkonstante benötigt wird, um den Kondensator auf 0,5 LSB der Eingangsspannung zu laden.

ADC-Auflösung (Bit)	0,5 LSB	Zeitkonstanten Faktor k
10	0,0488281%	7,62
12	0,0122070%	9,01
14	0,0030518%	10,40
16	0,0007629%	11,78

Tabelle 4.1 - Zeitkonstanten Faktor k

Für die Zeitkonstante des externen RC-Glieds gilt also:

$$\tau_{EXT} = R_{EXT} \cdot C_{EXT}$$

$$t_{ACQ} \geq k \cdot \tau_{EXT}$$

$$\tau_{EXT} \leq \frac{150\ ns}{11{,}78}$$

$$\tau_{EXT} \leq 12{,}73\ ns$$

Um das transiente Lastverhalten und die Kleinsignal-Einschwingzeit des OPVs zu berücksichtigen, wird eine Toleranz von 20 % angenommen. R_{EXT} wird somit wie folgt berechnet:

$$R_{EXT} \leq 0{,}8\ \cdot \frac{\tau_{EXT}}{C_{EXT}}$$

Für unterschiedliche Größen von C_{EXT} ergeben sich somit die folgenden Widerstandswerte:

C_{EXT}	R_{EXT}	Erfüllt R_{ISO} Empfehlung
0,5 nF	20 Ω	Ja
0,68 nF	15 Ω	Ja
1 nF	10 Ω	Ja
2,2 nF	4 Ω	Nein
3,3 nF	3 Ω	Nein

Tabelle 4.2 - Auswahl von R_{EXT}

Die größte Kapazität, die noch die Empfehlung für den Isolationswiderstand (siehe Abbildung 4.10) erfüllt, ist 1 nF.

4.2 Digitalteil

4.2.1 Optokoppler-Interface

Zur Potentialtrennung der digitalen Signale der Controllerplatine von der Verstärkerplatine werden bidirektionale Optokoppler eingesetzt. Übertragen werden die drei PWM-Paare sowie die Fehlerausgänge der Brückentreiber (des Wechselrichters), siehe Abbildung 4.11. Die Dioden der Optokoppler werden von beiden Seiten über MOSFET geschaltet. So kann der optimale Strom für die Leuchtdioden der Optokoppler durch einen Vorwiderstand eingestellt werden.

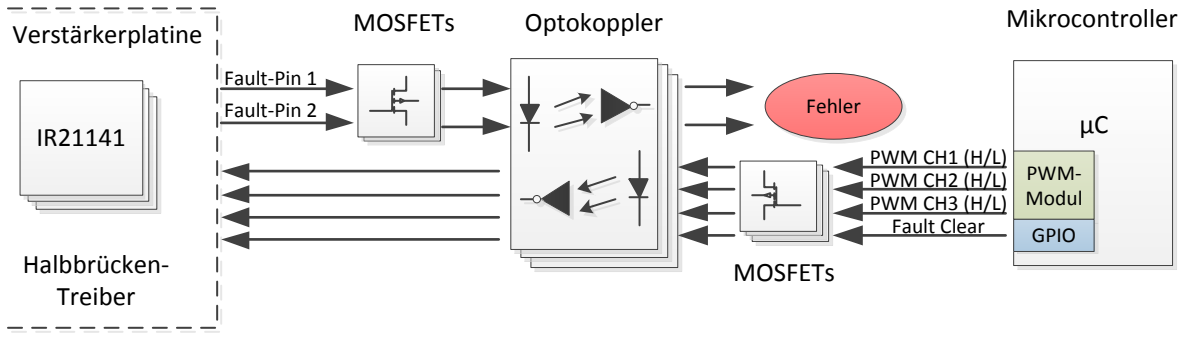


Abbildung 4.11 - Optokoppler Interface

Die drei Halbbrückentreiber des Wechselrichters haben jeweils zwei Fehlerausgänge, die einerseits der Kommunikation der Treiber untereinander und andererseits der Fehlererkennung und -abfrage externer Komponenten dienen. Die Active-Low Fehlerausgänge der drei Brückentreiber werden jeweils zu einem Netzwerk zusammengeschlossen und über Optokoppler auf die Controllerplatine geleitet, damit eine Fehlerabwicklung erfolgen kann. Ein Fehlerausgang dient der

Kommunikation von Phase-zu-Phase-Kurzschlüssen, die von den Brückentreibern durch eine Entsättigung der MOSFET erkannt werden können. Der zweite Fehlerausgang kann unter mehreren Umständen aktiv werden:

- Soft-Shutdown nach einer MOSFET Entsättigung ist abgeschlossen.
- Zu niedrige Versorgungsspannung V_{CC} der Brückentreiber.
- Wird extern auf Low-Pegel gezogen (z.B. von einem anderen Brückentreiber)

Die Fehler können nach erfolgter Abwicklung vom Mikrocontroller durch den *Fault Clear* Eingang quittiert werden.

4.2.2 Main-Bus

Wie bereits erwähnt wurde der Main-Bus von RS-485 auf CAN umgestellt, um eine zuverlässigere Datenübertragung zu gewährleisten. Weiters wurden Maßnahmen zur Filterung der Eingangssignale umgesetzt. Abbildung 4.12 zeigt ein Blockdiagramm der Schaltung.

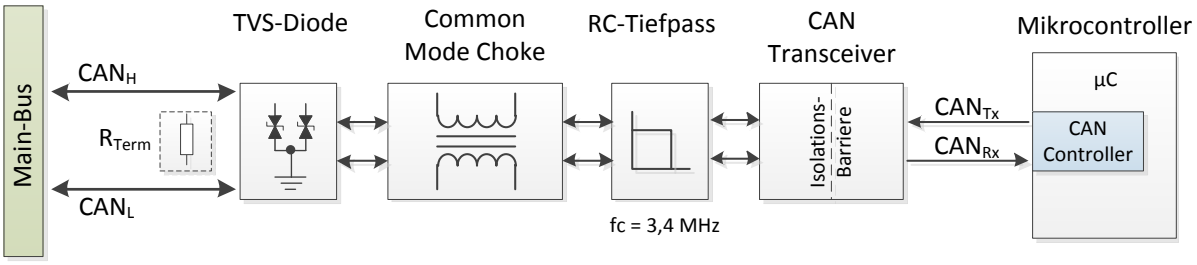


Abbildung 4.12 - Main-Bus

Um Reflexionen in den Bus-Leitungen zu verhindern, werden beim ersten und letzten Bus-Teilnehmer Abschlusswiderstände zur Terminierung bestückt. Abbildung 4.13 demonstriert die Wichtigkeit einer korrekten Terminierung der Bus-Leitungen. Die linke Aufnahme zeigt deutlich wie sich die reflektierten Wellen zum Ausgangssignal hinzuaddieren, bis die Endpegel erreicht sind.

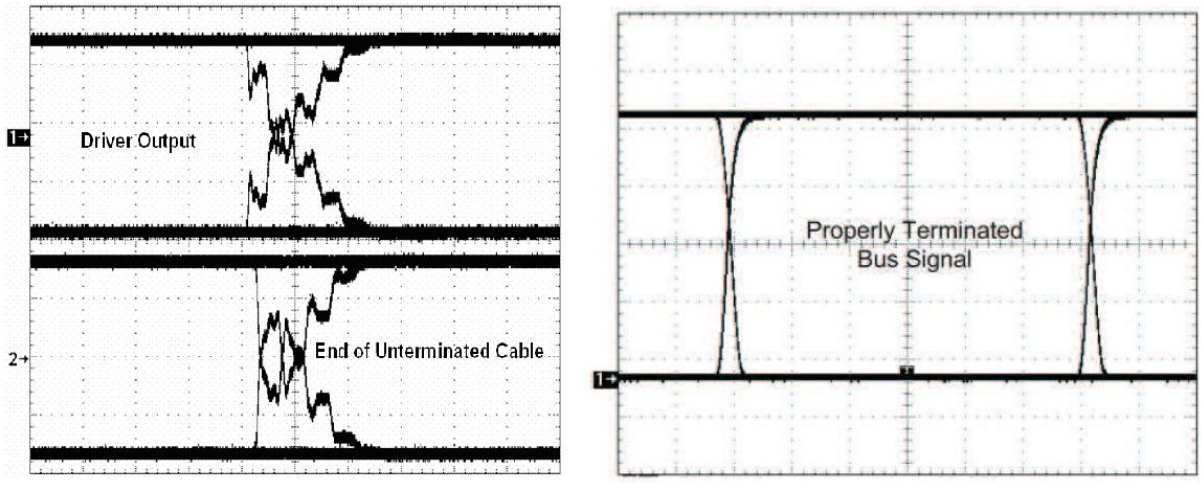


Abbildung 4.13 - CAN Terminierung [12]

Standardmäßig wird für die Terminierung ein 120 Ω Abschlusswiderstand (R_T) an den Leitungsenden eingesetzt. Um die Signalqualität zu verbessern, wird in High-Speed-CAN-Netzwerken eine sogenannte „Split-Termination“ empfohlen (siehe Abbildung 4.14).

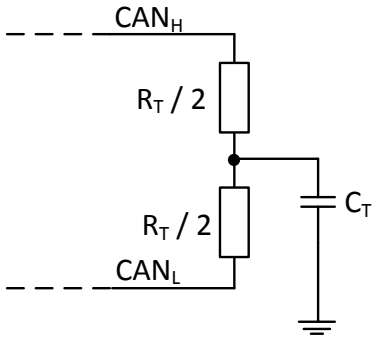


Abbildung 4.14 - Split-Termination

Die „Split-Termination“ dient der Filterung von hochfrequenten Gleichtaktstörungen. Dies wird durch einen Koppelkondensator zwischen zwei 60 Ω Abschluss-widerständen an Masse realisiert. Als Standardwert für den Kondensator bietet sich eine Kapazität von 4,7 nF an. Mit

$$f_c = \frac{1}{2\pi \left(\frac{R_T}{2} \,||\, \frac{R_T}{2}\right) C_T}$$

ergibt sich somit eine Grenzfrequenz von 1,1 MHz.

Um die CAN-Transceiver vor Spannungsimpulsen zu schützen, werden *Transient Voltage Supressor* (TVS) Dioden eingesetzt. Der gewählte Bauteil ist speziell für den CAN-Bus ausgelegt und hat zwei solcher Dioden pro Gehäuse. Die TVS-Dioden bieten einen einfachen und billigen Schutz gegen elektromagnetisch eingekoppelte Spannnungssüberhöhungen und elektrostatische Entladungen (ESD).

Die Signalfilterung erfolgt durch eine stromkompensierte Drossel zur effektiven Unterdrückung von Gleichtaktstörungen. Hier wurde eine, speziell für den CAN-Bus ausgelegte, Drossel mit der in Abbildung 4.15 dargestellten Impedanz-Charakteristik ausgewählt.

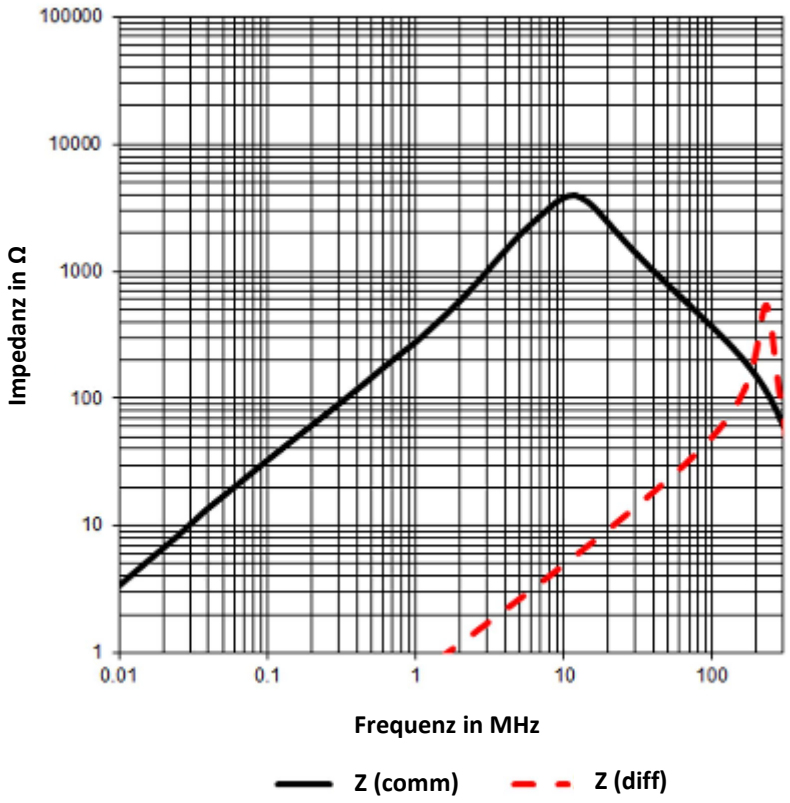


Abbildung 4.15 - Stromkompensierte Drossel (CAN-Bus) [13]

Während die differentiellen Nutzsignale nahezu unbeeinflusst bleiben, weist die Drossel für Gleichtaktstörungen bereits im niedrigen MHz-Bereich eine sehr hohe Impedanz auf.

Beide CAN-Bus-Leitungen werden vor dem Transceiver noch mit jeweils einem RC-Tiefpass mit einer Grenzfrequenz von 3,4 MHz gefiltert. Die stromkompensierten Drosseln, die RC-Tiefpassfilter sowie die korrekte Terminierung an den Bus-Enden sollten, in Verbindung mit einem geschirmten Twisted-Pair-Kabel für die Bus-Leitungen, eine Datenübertragung mit geringer Fehlerrate garantieren.

4.2.3 Sensor-Bus

Der Sensor-Bus dient der Positions- und Geschwindigkeitsregelung der Achsen. Auf der Z-Achse erfolgt zusätzlich eine Kraftregelung. Die Sensor-Entwicklung ist Teil einer anderen wissenschaftlichen Arbeit am Massagebett. Im Vergleich zur alten Controllerplatine wurde die RS-485-Schnittstelle mit einigen Komponenten erweitert, um die Signalqualität zu verbessern. Abbildung 4.16 zeigt den Aufbau der RS-485-Schnittstelle. Der Sensor-Bus hat lediglich zwei Bus-Teilnehmer – der Sensor, der als Master kontinuierlich die Positionsdaten (bei der Z-Achse zusätzlich Kraftdaten) sendet und der Controllereinheit, die als Slave diese Daten empfängt und verarbeitet. Technisch ist eine bidirektionale Kommunikation im Halbduplexmodus möglich, wird aber nicht realisiert, um die maximale Übertragung von Positions- bzw. Kraftwerten vom Sensor zum Controller zu ermöglichen.

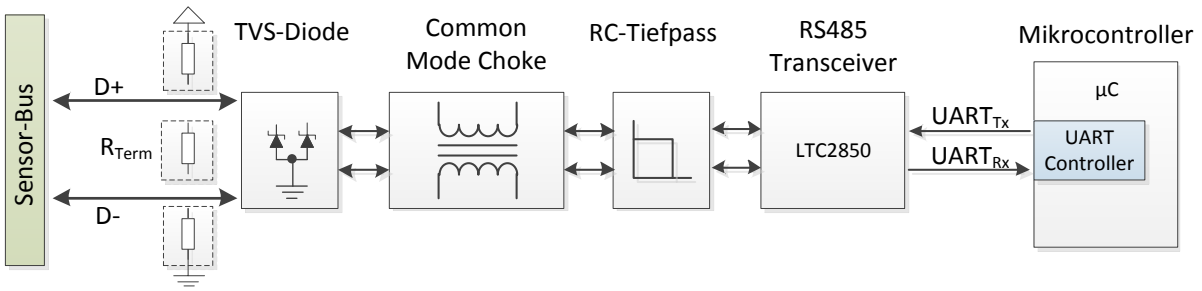


Abbildung 4.16 - Sensor-Bus

Ebenfalls muss hier der RS-485-Bus durch einen Abschlusswiderstand terminiert werden. Zusätzlich kommen zwei Bias-Widerstände zum Einsatz, welche im inaktiven Zustand für definierte Bus-Pegel sorgen.

Wie auch beim Main-Bus soll eine TVS-Diode die Elektronik vor elektromagnetisch eingekoppelten Spannungsüberhöhungen und elektrostatischen Entladungen schützen. Eine stromkompensierte Drossel, sowie ein RC-Tiefpass sollen Störungen unterdrücken und so eine möglichst fehlerfreie Datenübertragung gewährleisten. Das Übertragungsprotokoll und die Zusammensetzung des Datenpakets werden in Kapitel 6.3.5 erläutert.

4.2.4 Fehlerabwicklung

Um möglichst verzögerungsfrei auf sämtliche Systemfehler reagieren zu können, wurde eine hardwaremäßige Fehlerabwicklung implementiert. Alle „Active-Low“ Fehlersignale werden logisch UND-verknüpft. Die wichtigste Funktion wurde schon durch Abbildung 4.5 aufgezeigt – der allgemeine Fehlerausgang wird mit den PWM-Signalen verknüpft und ermöglicht so eine Motor-Abschaltung im Fehlerfall. Mögliche System-Fehler sind:

- Überstrom: Der maximal erlaubte Phasenstrom würd überschritten.
- Verstärkertemperatur: Wird der Kühlkörper der Controllereinheit zu warm, wird ein Fehler ausgegeben.
- Mikrocontroller System-Fehler: Der Mikrocontroller verfügt über eine integrierte *Fault Management Unit*, welche Systemfehler über einen Pin anzeigt.
- DSC-Fehler: Im Programmablauf wird jede PWM-Periode der logische Pegel eines GPIO-Pins geändert, um einen Hardware-Watchdog zu triggern. Geschieht das aufgrund eines Software-Fehlers nicht rechtzeitig, wird ein Fehler generiert.
- Sensor-Bus-Fehler: Wie auch beim DSC-Fehler, wurde für den Sensor-Bus ein Hardware-Watchdog implementiert, der bei Erhalt von gültigen Sensordaten zurückgesetzt wird. Bei einem Sensor-Fehler, wird dieser Fehler aktiv.
- Zwischenkreisspannung: Wie in Kapitel 4.1.3 erläutert, wird bei einer Wechselrichter-Zwischenkreisspannung außerhalb eines definierten Bereichs ein Fehler generiert.
- Brückentreiber-Fehler: Bei zu niedriger Brückentreiber-Versorgungsspannung oder Erkennung eines Phasenkurzschlusses.
- Failsafe-Bus: Wird einer der oben genannten Fehler aktiv, wird das über den Failsafe-Bus übertragen und generiert so einen Fehler auf allen am System beteiligten Controllereinheiten.

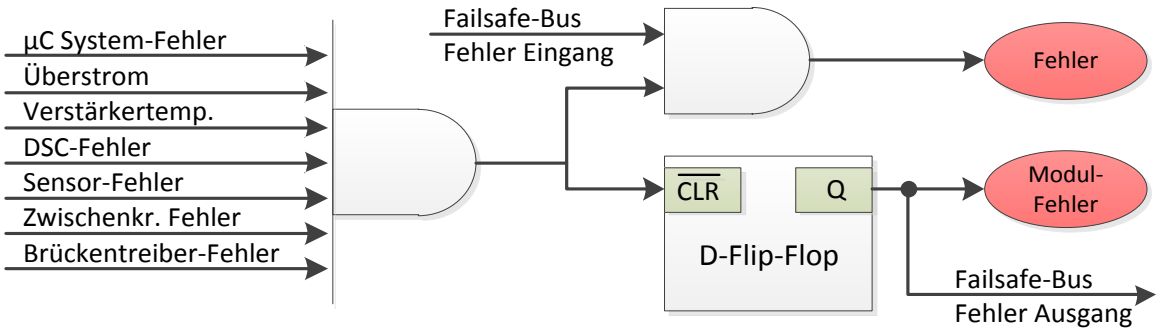


Abbildung 4.17 - Fehlerabwicklung

Abbildung 4.17 zeigt die Verknüpfung der Fehler mit UND-Gattern. Alle Fehler-Signale sind „Active-Low“ und werden mit dem Eingangssignal des Failsafe-Bus zu einem allgemeinen Fehlersignal verknüpft, durch das die PWM-Abschaltung (siehe Abbildung 4.5) herbeigeführt wird. Um den Fehlerausgang für den Failsafe-Bus zu erzeugen, wird ein D-Flip-Flop eingesetzt. Die Beschaltung des Flip-Flops ist in Abbildung 4.18 dargestellt. Durch ein Fehlersignal mit Low-Pegel am $\overline{CLR}$ Eingang nimmt der Ausgang Q des Flip-Flops einen Low-Pegel an und speichert diesen unabhängig vom aktuellen Pegel am CLK Eingang. Erst mit einer steigenden Flanke am CLK Eingang wird der Ausgang wieder „High“ und somit der Fehler am Failsafe-Bus zurückgesetzt (siehe Tabelle 4.3).

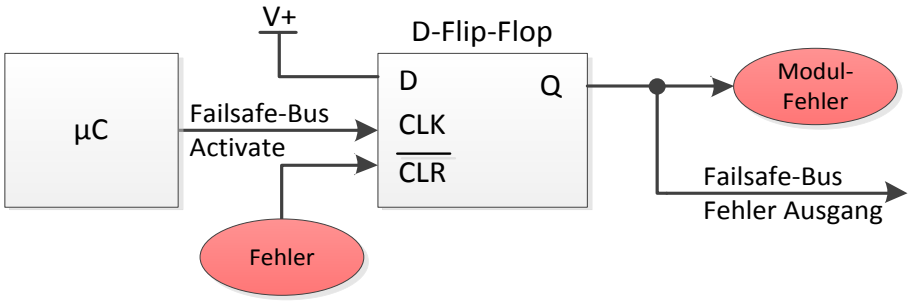


Abbildung 4.18 - D-Flip-Flop

Eingänge			Ausgang
$\overline{CLR}$	CLK	D	Q
H	↑	L	L
H	↑	H	H
H	H oder L	X	Q_0
L	X	X	L

Tabelle 4.3 - Wahrheitstabelle D-Flip-Flop

Für den DSC-Fehler und den Sensor-Fehler wurden Hardware-Watchdogs mittels integrierten monostabilen Multivibratoren (Mono-Flop) implementiert (siehe Anhang). Der Mikrocontroller muss diese innerhalb eines, mittels externem Widerstand und Kondensator festgelegten, Intervalls zurücksetzen. Damit der DSC-Fehler nicht aktiv wird, muss das Reset-Intervall kleiner als 100 µs sein. Der Reset wird dabei in der Interrupt-Service-Routine nach jeder PWM-Periode ausgeführt. Beim Sensor-Fehler wurde das Intervall auf 200 µs festgelegt. Der Reset erfolgt durch den Mikrocontroller, nachdem gültige Sensordaten empfangen wurden.

4.2.5 Failsafe-Bus

Um nun alle anderen Controllereinheiten über einen Fehler zu informieren bzw. über einen Fehlerzustand von anderen Controllereinheiten informiert zu werden, wurde der Failsafe-Bus implementiert. Im Wesentlichen besteht der Failsafe-Bus aus einer spannungsführenden Leitung, die im Fehlerfall mit Hilfe eines MOSFET von der Controllereinheit unterbrochen wird. Abbildung 4.19 zeigt ein Blockdiagramm der Failsafe-Bus Schnittstelle.

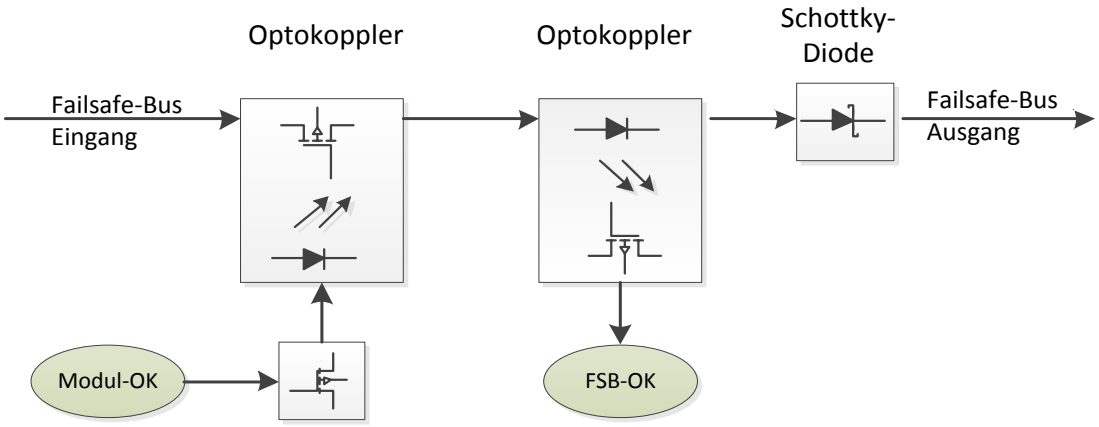


Abbildung 4.19 - Failsafe-Bus Schnittstelle

4.2.6 Debugging

Um mögliche Fehlerursachen zu finden wurden LEDs, die direkt mit den Fehlersignalen verknüpft sind, vorgesehen. So lässt sich auch bei einem Mikrocontroller-Fehler die auslösende Schnittstelle feststellen.

Für eine genauere Fehlerinformation bzw. allgemeine Systeminformationen wurde eine Schnittstelle für ein externes LCD-Display implementiert. Dafür wurden bidirektionale Bus-Transceiver eingesetzt, um den 3,3 V-Logik-Pegel des Mikrocontrollers in einen 5 V-Pegel zu wandeln. Die meisten LCD-Module arbeiten nach dem HD44780-Standard oder sind zu diesem kompatibel. Daher wurde eine Buchse

mit der 16-Pin Standardbelegung für eine 8-Bit Ansteuerung des Displays auf der Controllerplatine vorgesehen. Auf dem Display wird eine Stiftleiste angelötet, für die Fehlersuche kann das Display dann einfach auf die Controllerplatine gesteckt werden.

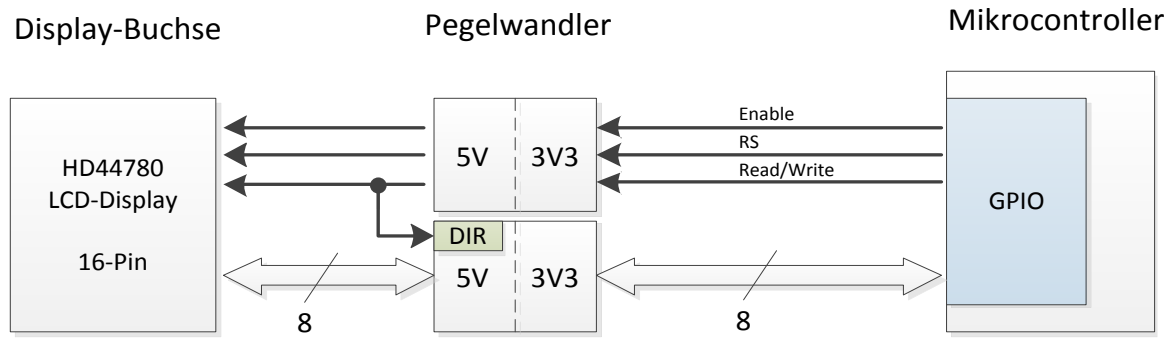


Abbildung 4.20 - Display

Abbildung 4.20 zeigt die Display-Verbindung. Die Schnittstelle kann für eine 8-Bit oder 4-Bit Display-Ansteuerung verwendet werden. Die Geschwindigkeit, mit der das Display die Befehle abarbeitet sind modellabhängig, daher ist eine bi-direktionale Kommunikation mit dem Display von Vorteil. Durch lesen der Datenleitungen kann festgestellt werden, ob das Display noch im „Busy“-Zustand ist, oder ob es für den nächsten Befehlssatz bereit ist. So kann die Schreibgeschwindigkeit deutlich gesteigert werden, da keine großzügig ausgelegte Wartezeit eingerechnet werden muss. Die „Read/Write“-Steuerleitung gibt den Befehl an das Display und schaltet gleichzeitig den zweiten Pegelwandler in die gewünschte Signal-Richtung.

4.3 Spannungsversorgung

Da die analogen und digitalen Schaltungsteile strikt getrennt sein sollten, und unterschiedliche Spannungspegel nötig sind, wurden verschiedene Spannungsregler eingesetzt. Abbildung 4.21 zeigt eine Übersicht der, auf der Controllerplatine, vorkommenden Spannungen.

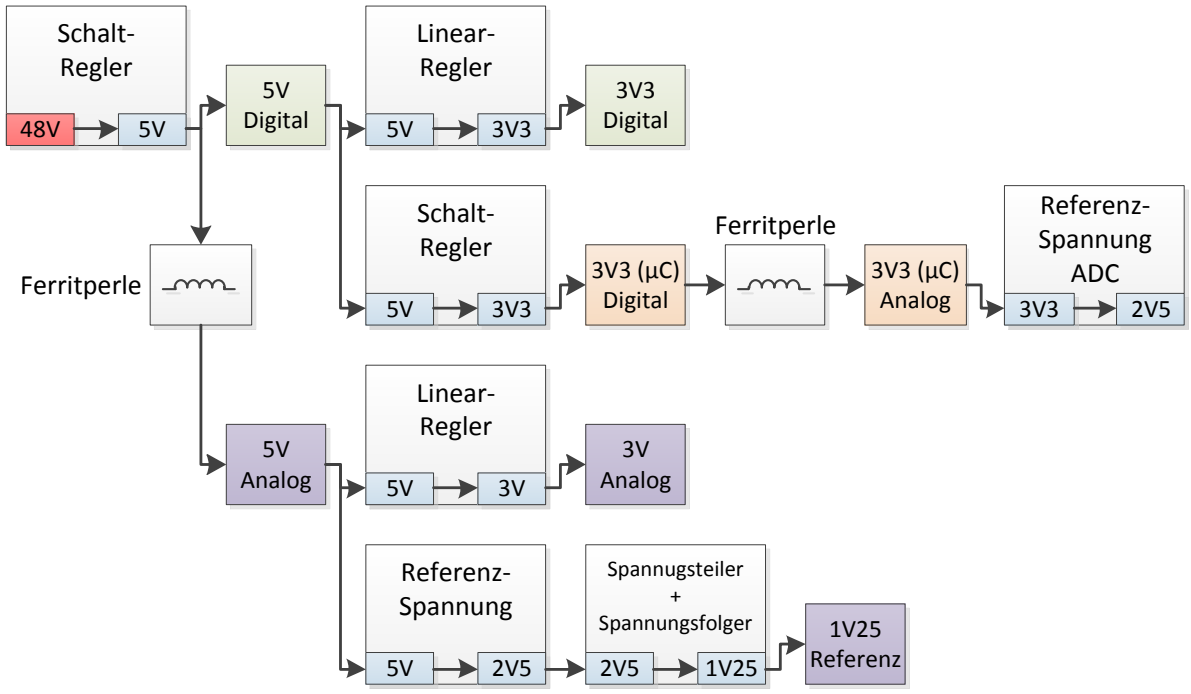


Abbildung 4.21 - Spannungsversorgung

Im ersten Schritt wird die 48 V-Versorgungsspannung der Controllereinheit auf 5 V gewandelt. Aufgrund der hohen Spannungsdifferenz und der Tatsache, dass der gesamte Strombedarf der Controllerplatine über diesen Regler bereitgestellt wird, wurde hier ein Schaltregler eingesetzt, um die Verlustleistung möglichst gering zu halten. Abhängig vom Strombedarf liegt die Effizienz der Wandlung bei ca. 80 %.

Es folgt ein Linearregler zur Erzeugung der 3,3 V Versorgungsspannung für den digitalen Schaltungsteil. Zur dynamischen Trennung des analogen vom digitalen Schaltungsteil werden Ferritperlen eingesetzt. Diese eignen sich gut zur Störungsunterdrückung bei Versorgungsspannungen, da sie einen geringen Gleichstromwiderstand aufweisen. Für höhere Frequenzanteile steigt auch die induktive Impedanz (siehe Abbildung 4.22).

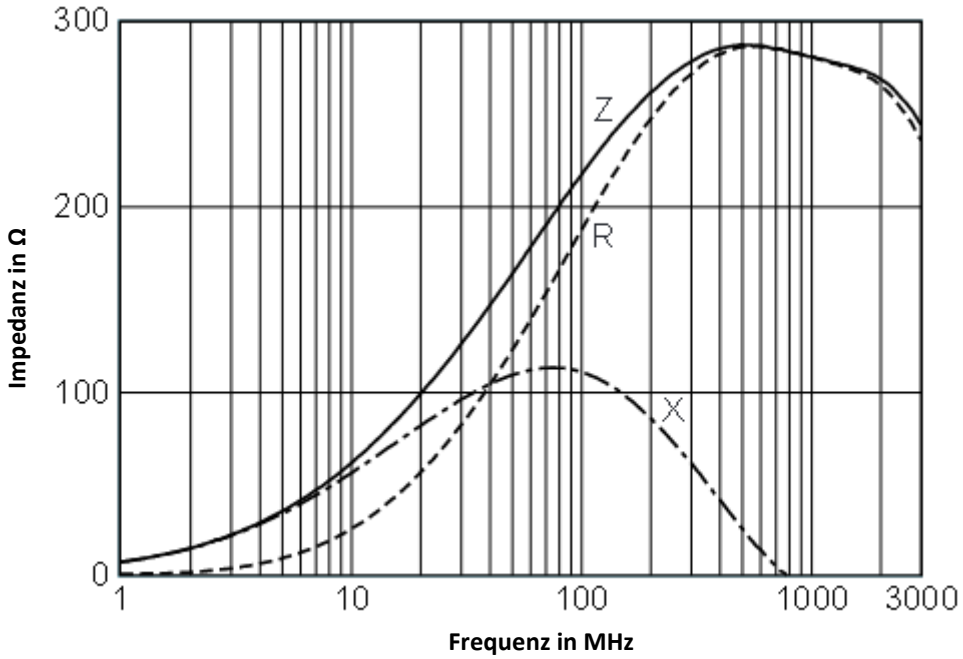


Abbildung 4.22 - Impedanzcharakteristik Ferritperle

Von der analogen 5 V Spannung wird dann die 3 V Versorgungsspannung für den analogen Schaltungsteil mit einem Linearregler erzeugt. Die 1,25 V Referenzspannung wird als Mitte des 2,5 V ADC-Messbereichs für den Offset des Differenzverstärkers (Strommessung, siehe Kapitel 4.1.1) und die Referenzierung der Eingangsspannung der unipolar betriebenen OPVs der Tiefpassfilter (siehe Kapitel 4.1.5) benötigt. Dafür kommen ein 2,5 V-Präzisions-Linearregler und ein Spannungsteiler mit einem nachgeschaltenen Impedanzwandler zum Einsatz (siehe Abbildung 4.23).

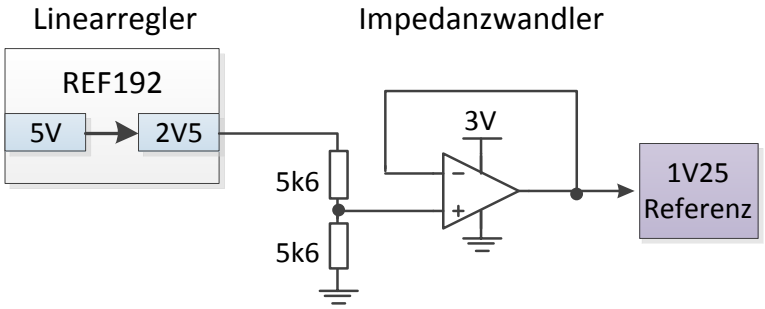


Abbildung 4.23 - 1,25V Referenzspannung

Für die Mikrocontroller-Versorgungsspannung wird ein Schaltregler eingesetzt, um die Verlustleistung gering zu halten. Auch hier wird eine Ferritperle vorgesehen, um den digitalen und analogen Schaltungsteil (ADC) dynamisch zu trennen. Der *CM408F* hat eine interne 2,5 V ADC-Referenzspannung, die durch eine Externe überschrieben werden kann. Es werden zwei 2,5 V Präzisions-Linearregler[3] eingesetzt, die eine höhere Genauigkeit erzielen können, als die interne Referenzspannungsquelle.

[3] Auf dem Evaluation-Board des CM408F von Analog Devices wird für beide ADCs ein eigener Spannungsregler eingesetzt. Da das Design als Empfehlung angesehen werden kann, wurden auch hier zwei Spannungsregler verwendet.

5 PCB-Layout Design

Der nächste Schritt in der Entwicklung der Controllerplatine war der Entwurf des PCB-Layouts. Die prinzipiellen Anforderungen an die Platine waren:

- Gleicher Formfaktor (Euro-Platine mit 160 x 100 mm) wie die Vorgängerversion.
- Flachbandbuchse zur Verbindung mit der Verstärkerplatine soll erhalten bleiben, damit die gleichen Verstärkerplatinen verwendet werden können.
- Einfache Anschlussmöglichkeit eines LCD-Displays.
- Zugänglichkeit der JTAG-Schnittstelle bei geschlossenem Gehäuse.
- Gleiche Position der Bohrungen für die Befestigungsschrauben.

Das „Routing“ der 160 mm x 100 mm großen Platine mit vier Schichten und ca. 600 Bauteilen war ein sehr zeitintensiver Prozess. Da die Qualität des Designs einen großen Einfluss auf die Funktion der Schaltung hat, wurden, wo vorhanden, die Layout-Empfehlungen der Hersteller beachtet, um eine hohe Signalqualität zu gewährleisten. Das Designergebnis befindet sich mit einem 1:1 Maßstab im Anhang der Diplomarbeit.

5.1 Aufbau der Platine

Beachtet man die Anzahl der Bauteile, die auf der Platine untergebracht werden sollen, wird schnell klar, dass ein zweischichtiges Platinendesign nicht ausreichen wird. Da die Vorgängerversion als eine vierschichtige Platine realisiert wurde, soll auch die neue Version der Controllerplatine mit einer Power- und Ground-Plane entwickelt werden. Dadurch, dass die Versorgungsleitungen als Flächen auf der Power-Plane realisiert werden, ist die Platzierung der Bauteile viel variabler, als wenn lange Versorgungsleitungen zu jedem Bauteil gelegt werden müssen. Ein wesentlicher Vorteil des Multi-Layer-Designs ist auch die bessere Unterdrückung von hochfrequenten Störungen und eine Reduktion des Übersprechens zwischen Leiterbahnen.

Abbildung 5.1 zeigt den Aufbau der Multi-Layer-Platine mit vier Schichten. Außen (Top und Bottom Layer) befinden sich die Signalleitungen. Getrennt durch Schichten von Dielektrikum befinden sich die Masse- und Versorgungslayer im Inneren der Platine. Durch spezielle Ätztechnik ist es möglich per Multi-Layer-Durchkontaktierungen eine Verbindung zu den Kupferflächen auf einem inneren Layer herzustellen.

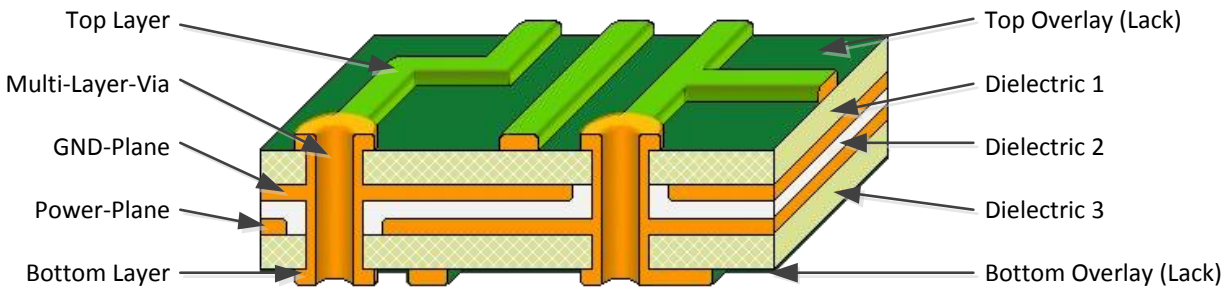


Abbildung 5.1 - PCB Layerstack

Zum Korrosionsschutz und zur Isolierung befinden sich Lackschichten an den Außenseiten. Durch eine Maske wird verhindert, dass der Lack auf die Lötpads aufgetragen wird.

5.2 Umsetzung

Zuerst wurden die Bauteile zusammengehörender Schaltungsteile in Gruppen sortiert und die Verbindungen so kurz wie möglich geroutet. Anschließend wurden die Bauteile auf der Platine platziert, so dass sich schlussendlich die in Abbildung 5.2 dargestellte Anordnung entwickelte.

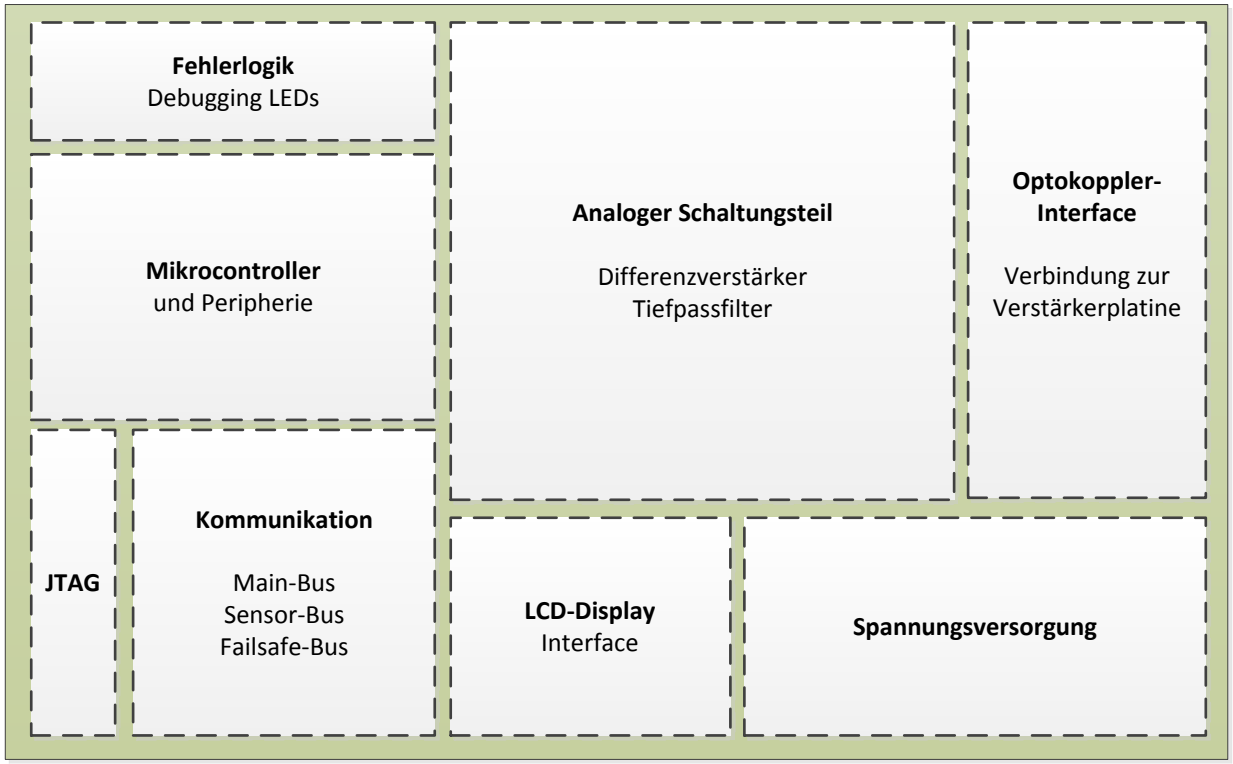


Abbildung 5.2 - Schaltungsbereiche

Während der Layout-Entwicklung mussten einige Details zum Schaltungsaufbau genauer untersucht werden. Es soll nun auf die Lösung von Problemstellungen hinsichtlich der Umsetzung bestimmter Schaltungsteile eingegangen werden.

5.2.1 Störungen durch PWM-Schaltvorgänge

Ein Problem, das untersucht werden musste, ist die Auswirkung der PWM-Schaltvorgänge auf andere, aber vor allem die analogen Schaltungsteile. Die sechs PWM-Leitungen haben ihren Ursprung am Mikrocontroller und verlaufen bis zum Optokoppler-Interface, wo die Verbindung zur Verstärkerplatine hergestellt wird. Somit gehören diese Leiterbahnen zu den Längsten (ca. 100 mm) auf der Platine, die auch am analogen Schaltungsteil vorbeigeführt werden müssen. Das Problem verursacht hier die Übertragung von HF-Anteilen auf diese Schaltungsteile durch die Flankensteilheit der PWM-Signale. Aufgrund der Stromänderungen durch die Schaltvorgänge wird ein Schwingkreis aus Leitungsinduktivität und parasitären Kapazitäten angeregt.

Die Induktivität einer Leiterbahn mit der Länge l, Breite b und Dicke d in cm lässt sich durch folgende Beziehung [14] abschätzen:

$$L = 2 \cdot l \cdot \left[ln\left(\frac{2l}{b+d}\right) + 0{,}5 + 0{,}2235\left(\frac{b+d}{l}\right)\right] \; nH$$

Mit einer durchschnittlichen Leitungslänge von ca. 100 mm, einer Leiterbahnbreite von 15 mil (0,381 mm) und einer Kupferstärke von 35 µm, ergibt sich somit eine Leiterbahninduktivität von:

$$L = 134 \; nH$$

Der ohmsche Widerstand der Leiterbahn berechnet sich mit dem spezifischen elektrischen Widerstand von Kupfer $\rho = 17{,}86 \; m\Omega \frac{mm^2}{m}$ und der Beziehung:

$$R = \frac{\rho \cdot l}{A} = 134 \; m\Omega$$

In der SPICE Software *TINA-TI* von *Texas Instruments* wurde die in Abbildung 5.3 dargestellte Schaltung mit dem zu schaltenden MOSFET modelliert. Hierfür wurde das SPICE-Modell des MOSFET-Herstellers verwendet, das die parasitären Kapazitäten und den Schaltvorgang des Bauteils gut simuliert. Es soll die AC-Übertragungscharakteristik untersucht werden, um mögliche Störquellen zuordnen zu können, die durch Anteile hoher Frequenzen in den Schaltflanken verursacht werden.

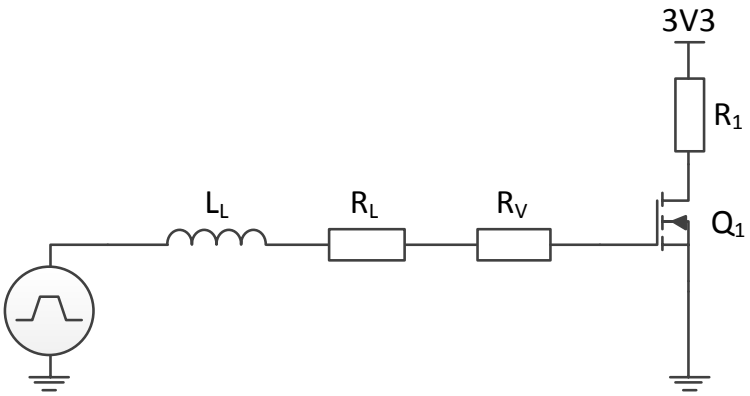


Abbildung 5.3 - Simulation der PWM-Schaltvorgänge

Das Simulationsergebnis ist in Abbildung 5.4 dargestellt. Ohne Vorwiderstand (R_V) erfährt das Signal eine deutliche Verstärkung im Frequenzbereich von 20 bis 60 MHz mit einer Überhöhung von 6,3 dB bei einer Frequenz von 40,7 MHz. Dies stellt ein erhebliches Problem dar, da es in diesem Frequenzbereich zu kapazitiver Einkopplung in benachbarte Leiterbahnen kommen kann, was die Signalqualität des analogen Schaltungsteils beeinflussen könnte. Die Lösung ist einfach und effektiv - durch einen Vorwiderstand am Gate des MOSFETs wird der Schaltvorgang verlangsamt. Dabei muss darauf geachtet werden, dass dadurch die Schaltverluste des MOSFETs ansteigen. Aus diesem Grund und weil der Verlauf der PWM-Signale erhalten bleiben soll, darf der Schaltvorgang nicht beliebig verlangsamt werden.

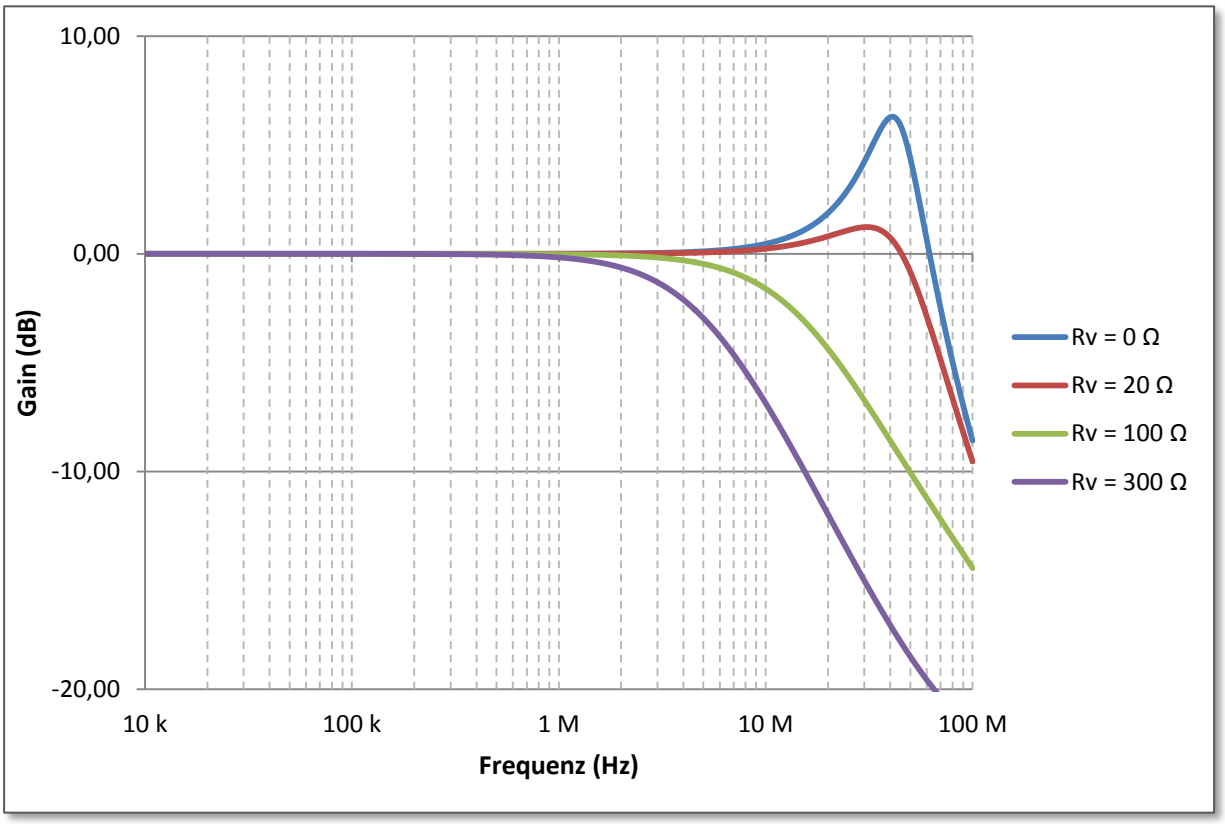


Abbildung 5.4 - Übertragungscharakteristik der PWM-Leitung

Da das PWM-Signal möglichst wenig beeinflusst und ein zu langsamer Schaltvorgang des MOSFETs vermieden werden sollte, wird für eine experimentelle Ermittlung des Vorwiderstands das Modell auch in der Zeitebene mit einer Rechteck-Spannungsquelle simuliert.

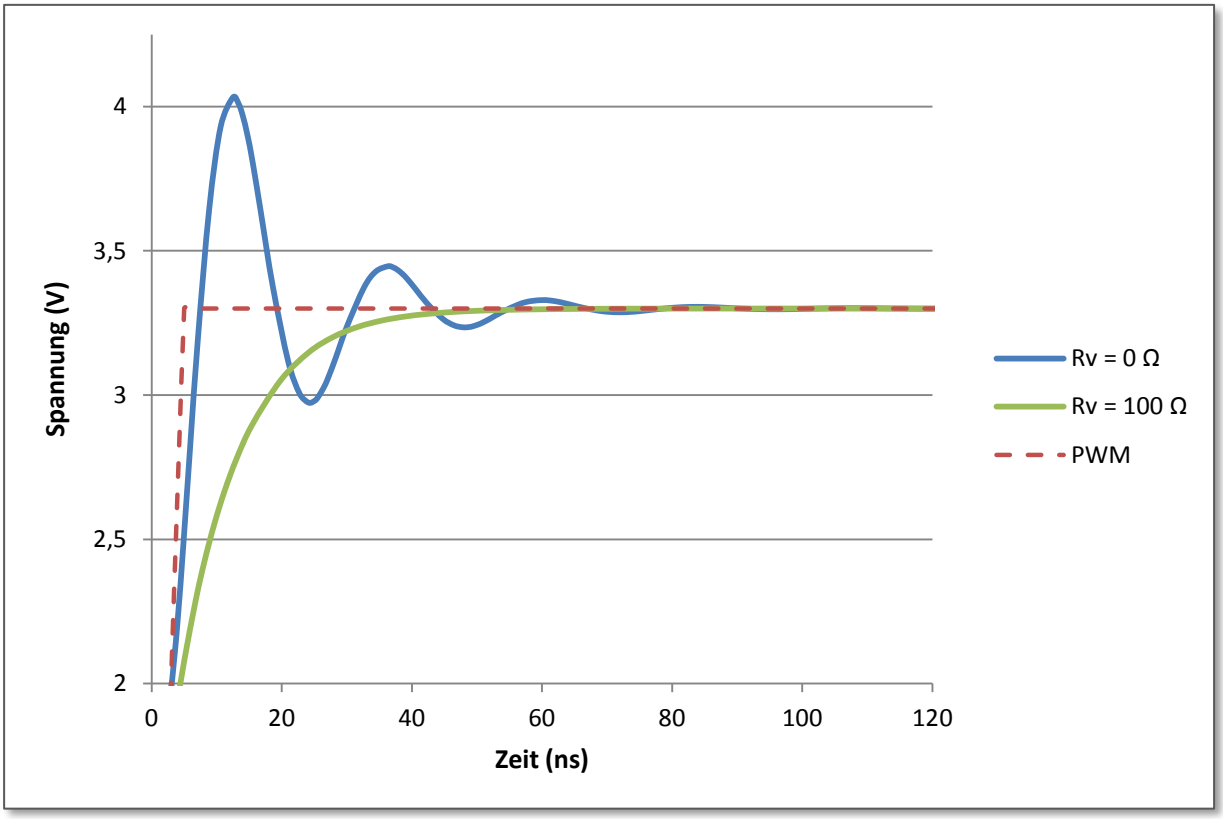


Abbildung 5.5 - PWM Flanke

Abbildung 5.5 zeigt die ersten 120 ns der steigenden PWM-Flanke. Ohne Vorwiderstand ist ein deutliches Überschwingen sichtbar. Durch Verwendung eines Vorwiderstandes von 100 Ω wird die Flankensteilheit reduziert und der Einschwingvorgang gedämpft. Der Signalausschnitt ist in der Abbildung stark vergrößert. Bei einer PWM-Frequenz von 50 kHz stellt eine Verzögerung von wenigen Nanosekunden kein Problem für den Signalverlauf und die Schaltverluste des MOSFETs dar.

5.2.2 Bauteile mit Kühlflächen

Durch immer kleinere Bauteilgrößen bei gleichen oder sogar größeren Schaltleistungen kommt die Wärmeabfuhr durch Thermal-Pads vermehrt zum Einsatz. Durch Thermal-Pads auf der Unterseite der Bauteilgehäuse erfolgt die Kühlung über Kupferflächen auf der Platine anstatt über Kühlkörper. Abbildung 5.6 zeigt einen Querschnitt durch einen solchen Gehäuseaufbau.

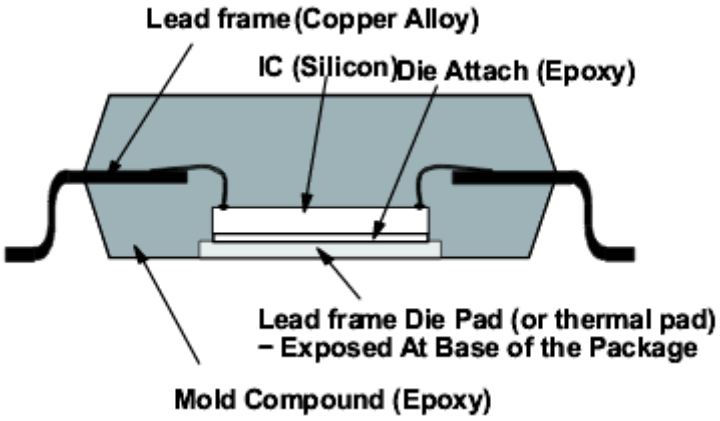


Abbildung 5.6 - Thermal Pad [15]

Das Problem für dieses Projekt ist die Herstellbarkeit der Platine, genauer das Löten dieser Bauteile. Da es sich um einen Prototypen handelt, sollen die Platinen im Labor von Hand bestückt werden. Für das Löten von Hand sind Bauteile mit Thermal-Pads gänzlich ungeeignet. Die empfohlene Herstellungsmethode ist das Reflow-Lötverfahren, bei dem die Platine mit allen Bauteilen in einem Ofen mit einem definierten zeitlichen Temperaturverlauf erhitzt wird. Dadurch, dass die Löt-Pads auf der Platine mit großen Massenflächen verbunden sind, wird auch das Heißluftlöten erschwert, da die Wärme an der Lötstelle abgeführt wird. Wird der Bauteil zu lange dieser Hitze ausgesetzt, könnte das zu thermischen Schäden führen.

Zwei Bauteile auf der Controllerplatine haben ein Thermal-Pad und sind nur in diesem Gehäuse verfügbar (der Mikrocontroller und ein Schaltwandler). Um diese Bauteile von Hand mit dem Lötkolben löten zu können, wurde auf den Löt-Pads eine relativ große Durchkontaktierung platziert. Die Durchkontaktierung wird von der Platinenunterseite mit Lötzinn aufgefüllt, sodass eine Verbindung mit dem Thermal-Pad hergestellt wird. Die Multi-Layer-Durchkontaktierung ist außerdem im inneren Layer mit der Massefläche auf der Ground-Plane verbunden. Abbildung 5.7 zeigt das am Beispiel des Schaltreglers. Der Innendurchmesser der Durchkontaktierung ist mit 1,5 mm groß genug damit das Pad mit einer passenden Lötspitze angelötet werden kann.

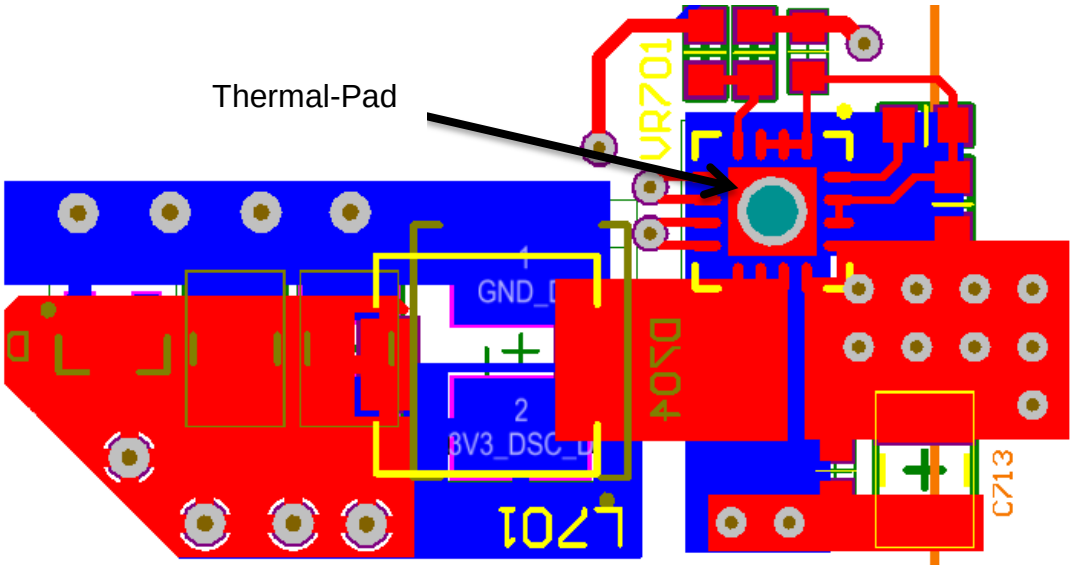


Abbildung 5.7 - Thermal Pad Schaltregler

6 Programmierung

In diesem Kapitel soll näher auf die Software-Entwicklung des Projekts eingegangen werden, wobei der Schwerpunkt hier auf dem Programm-Ablauf und der Funktion der Komponenten liegt. Anmerkungen zum Programmcode wurden direkt in den Source-Dateien kommentiert.

Aufgrund der Tatsache, dass sich der Mikrocontroller, der für das Projekt gewählt wurde, auch in der Endphase dieser Arbeit im „Pre-Release“ Stadium befand, und die Verfügbarkeit des Mikrocontrollers selber sowie die von Treibern und generellen Programmierungsrichtlinien leider stark reduziert war, mussten einige Einbußen hinsichtlich der ursprünglichen Zielsetzung dieser Diplomarbeit hingenommen werden. Da *Analog Devices* keine Notwendigkeit in der Bereitstellung von Startup-Routinen und HAL-Treibern (Hardware Abstraction Layer) sieht, die mit dem frei verfügbaren GCC-Compiler des GNU-Projekts kompatibel sind, musste ein großer Teil der Entwicklungszeit in diesen Abschnitt des Projekts investiert werden.

Bis zum Abschluss dieser Diplomarbeit war der Mikrocontroller nicht einzeln erwerblich und leider auch nicht als Musterexemplar verfügbar. Das heißt, die Software-Entwicklung musste mit dem Evaluations-Kit ausgeführt und getestet werden. Da es keine Möglichkeit gibt, die Motoransteuerung und Regelung durch die Controllerplatine am Massagebett ohne den Mikrocontroller zu testen, wurde das Ziel der Software-Entwicklung auf die Programmierung der HAL-Treiber mit einer, für das Projekt spezifischen, Konfigurierung der Prozessor-Komponenten eingeschränkt. Die Implementierung der Regelung soll in einer weiteren wissenschaftlichen Arbeit am Massagebettprojekt umgesetzt werden.

6.1 CMSIS

Die grundlegende Anforderung an die Software-Entwicklung, war die Programmierung eines GCC-Compiler kompatiblen Codes, da die von *Analog Devices* unterstützten Entwicklungsumgebungen (*Keil MDK* und *IAR Embedded Workbench*) leider nur in stark eingeschränkten Versionen (32kB-Limit) frei verfügbar sind. Durch die ARM-Cortex-Architektur wird prinzipiell ein herstellerunabhängiges Software-Grundgerüst mit dem *ARM Cortex Microcontroller Software Interface Standard* (CMSIS) geschaffen (siehe Abbildung 6.1). Durch diesen Standard ist, bis zu einem gewissen Grad, eine Programmkompatibilität zwischen Prozessoren (der gleichen Architektur) verschiedener Hersteller gegeben. Da die Prozessor-Peripherie (CAN-Controller, UART-Controller, ADC, PWM-Modul, usw.) herstellerabhängig sind, werden die Treiber (HAL-Treiber) dafür üblicherweise vom Hersteller bereitgestellt.

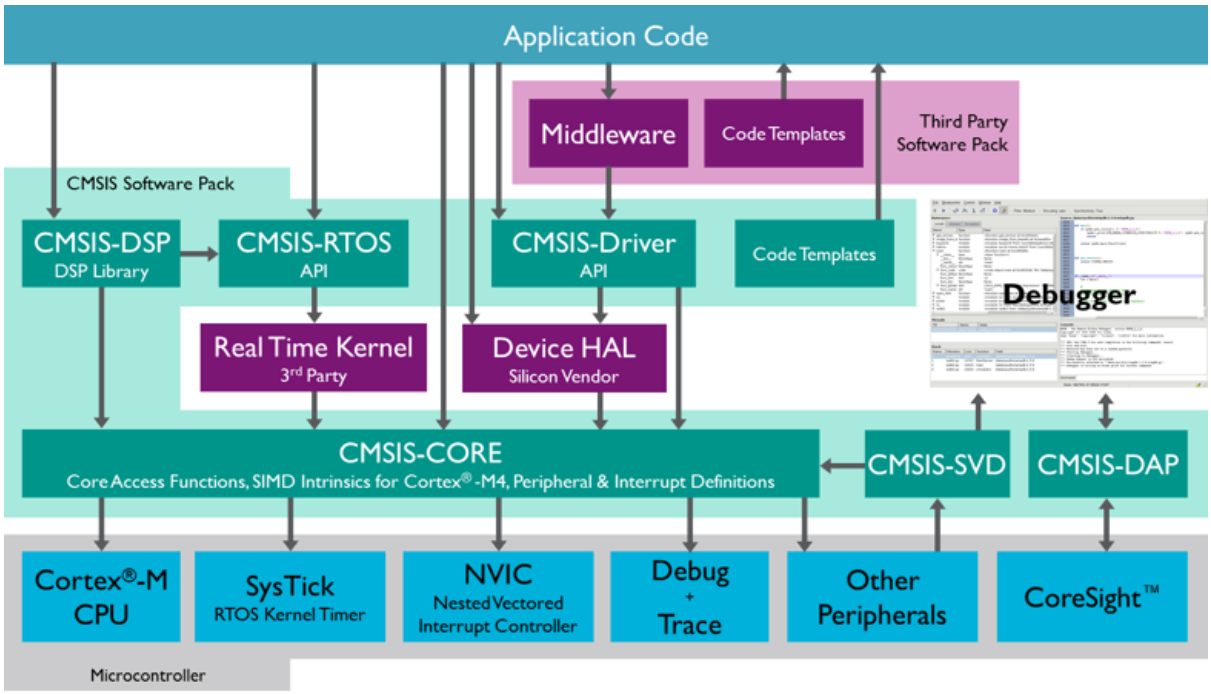


Abbildung 6.1 - CMSIS Übersicht [16]

6.2 Programmablauf

Obwohl die Umsetzung des Regelalgorithmus, sowie auch die Datenverarbeitung der Bus-Kommunikation nicht mehr Teil dieser Diplomarbeit sind, besteht ein Konzept für den grundlegenden Programmablauf für den Betrieb der Controller-Einheit. Aufgrund der zur Verfügung stehenden Rechenleistung des Mikrocontrollers und der autonom arbeitenden Modul-Controller (siehe nachfolgende Kapitel) wird eine ereignis-orientierte Programmierung bevorzugt, wodurch der Regelalgorithmus sowie die Datenverarbeitung von Sensor- und Main-Bus in den jeweiligen Event-Handlern stattfindet (siehe Abbildung 6.2).

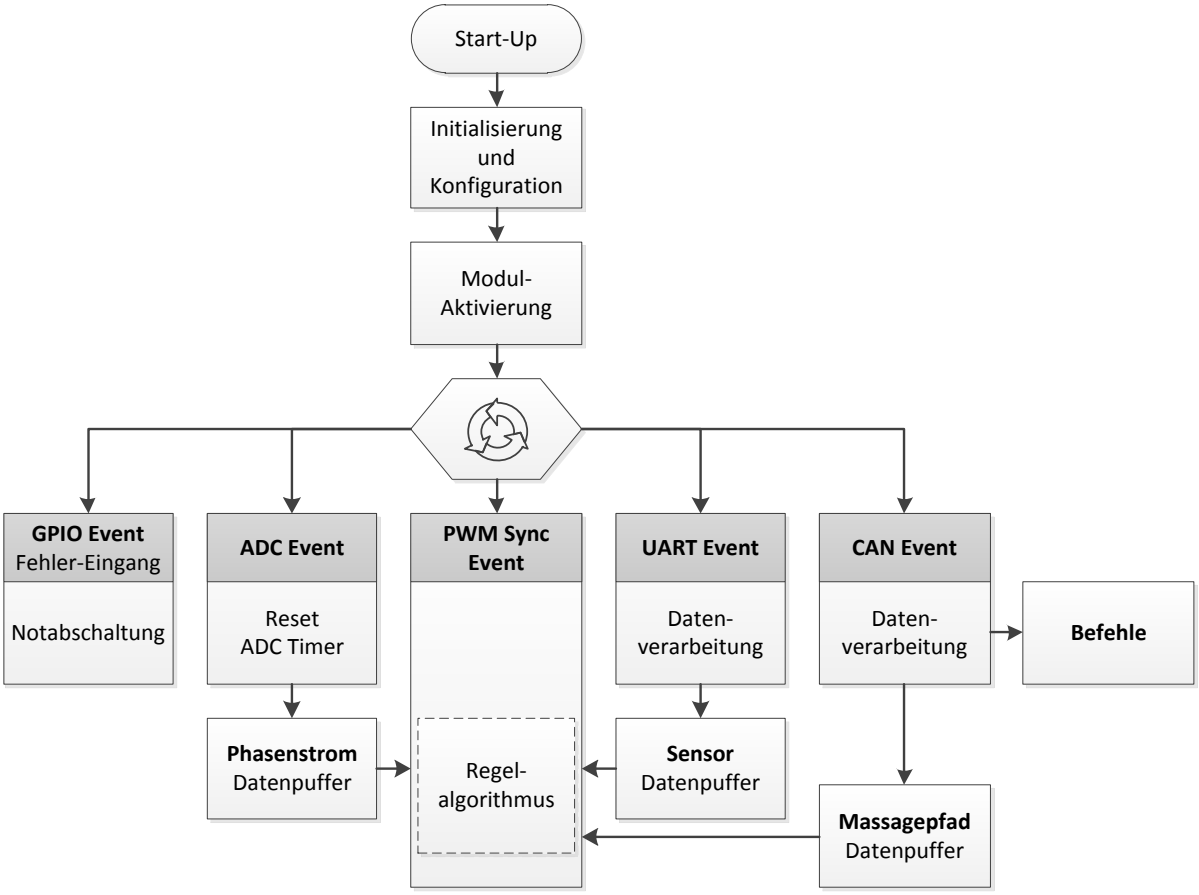


Abbildung 6.2 - Programmablauf

6.2.1 Interrupt-Prioritäten

Da der gesamte Softwareablauf eventbasierend in den Interrupt-Service-Routinen (ISR) der Module programmiert werden soll, muss darauf geachtet werden, dass die entsprechend Ereignisse zeitgerecht abgearbeitet werden. Die ARM Cortex Architektur bietet mit dem Konzept der Interrupt-Prioritäten eine entscheidende Eigenschaft für die Umsetzung dieses Programmablaufs. Die Interrupt-Requests (IRQ) der einzelnen Module bekommen eine Priorität zugewiesen. IRQ mit einer höheren Priorität können während der Ausführung einer ISR eines Interrupts mit niedrigerer Priorität ausgeführt werden. D. h., die ISR niedriger Priorität kann für einen Interrupt höherer Priorität unterbrochen und nach deren Ausführung wieder fortgesetzt werden. Dadurch können auch zeitintensivere Rechenoperationen in den Interrupt Handlern ausgeführt werden, ohne Gefahr zu laufen, einen zeitkritischen Interrupt zu verpassen.

Die Interrupt-Prioritäten können durch Routinen aus dem CMSIS-Paket für den Nested Vectored Interrupt Controller (NVIC) gesetzt werden. Zuerst muss allerdings die Anzahl der Prioritätsstufen, die nach dem Cortex-Standard konfigurierbar ist,

festgelegt werden. Im *CM408F* sind vier Bit für diese Konfiguration implementiert, d. h. der Mikrocontroller verfügt über 16 Prioritätsstufen. Prinzipiell wird dabei in „Preempt Priority“ und „Subpriority“ unterschieden. Unter Preempt Priority ist die erwähnte Möglichkeit zur Unterbrechung eines anderen Interrupt-Handlers zu verstehen. Die Subpriority bestimmt, welcher Interrupt mit gleicher Preempt Priority bei einem zeitgleichen Request zuerst ausgeführt wird. Die Prioritätsstufen wurden für dieses Projekt so konfiguriert, dass acht Preempt Priority Stufen und zwei Subpriority Stufen zur Verfügung stehen. Abbildung 6.3 zeigt den Programmablauf zur Konfiguration der Interrupt-Prioritäten. Bei der Umsetzung konnte auf Routinen aus dem CMSIS-Paket zurückgegriffen werden.

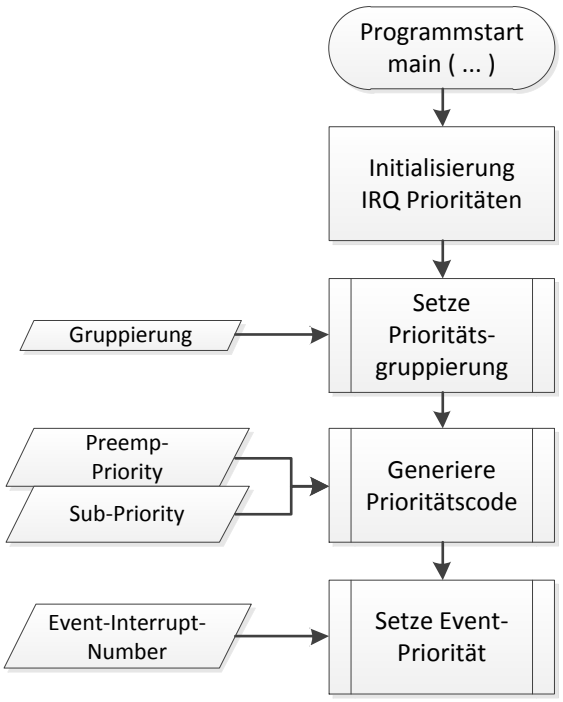


Abbildung 6.3 - Konfiguration der Interrupt Prioritäten

Abbildung 6.4 zeigt, wie die Prioritätsstufen der verwendeten Module definiert werden. Von außen nach innen weisen die Interrupts der Module eine höhere Preempt Priorität auf, sodass sie alle äußeren Interrupt-Handler unterbrechen können. Der GPIO Interrupt tritt prinzipiell nur im Fehlerfall ein und soll eine Notabschaltung auslösen, daher erhält dieser die höchste Priorität. Die ADC-Timer müssen nach jedem Frame zurückgesetzt werden (siehe Kapitel 6.3.4). Dadurch tritt dieser Interrupt häufig ein und muss auch rechtzeitig abgearbeitet werden. Die Interrupts von Sensor- und Main-Bus weisen prinzipiell nur darauf hin, dass neue Daten zur Verfügung stehen, da sowohl der UART- als auch der CAN-Controller autonom Daten empfangen und im Arbeitsspeicher ablegen kann (siehe Kapitel 6.3.5 und 6.3.6).

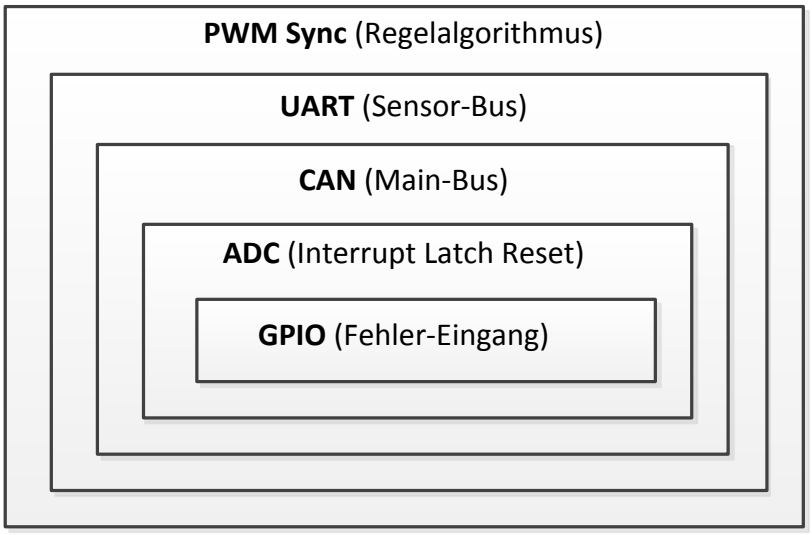


Abbildung 6.4 - Interrupt Prioritäten

6.3 Initialisierung und Konfiguration der Peripherie

Da die Funktionen und Betriebsmodi der verwendeten Peripherie sehr umfangreich sind, würden HAL-Treiber, die den gesamten Funktionsumfang abdecken, einen unverhältnismäßigen Entwicklungsaufwand benötigen. Es wurden also Initialisierungs- und Konfigurations-Routinen entwickelt, die speziell auf die benötigten Funktionen zugeschnitten sind, aber trotzdem eine übersichtliche Struktur behalten, wodurch eine spätere Anpassung der Parameter leicht zu bewerkstelligen ist.

In den folgenden Punkten, soll näher auf die verwendeten Komponenten und deren Aufbau eingegangen werden.

6.3.1 Pinmultiplexing

Nachdem alle Mikrocontroller mit einer begrenzten Pin-Anzahl auskommen müssen, die nicht für alle Ein- und Ausgänge der enthaltenen Module ausreichend sind, werden einige Signale auf einem Pin zusammengefasst. Damit nun die gewünschte Funktion eines bestimmten Pins verfügbar ist, muss diese im *Multiplexer Control Register* des Ports eingestellt werden. Analog Devices liefert eine benutzerfreundliche Variante zur Erstellung eines Programmcodes mit den richtigen Registerwerten anhand eines Java-Tools mit grafischer Benutzeroberfläche (siehe Abbildung 6.5). Der generierte Code wird zu den Initialisierungs-Routinen hinzugefügt.

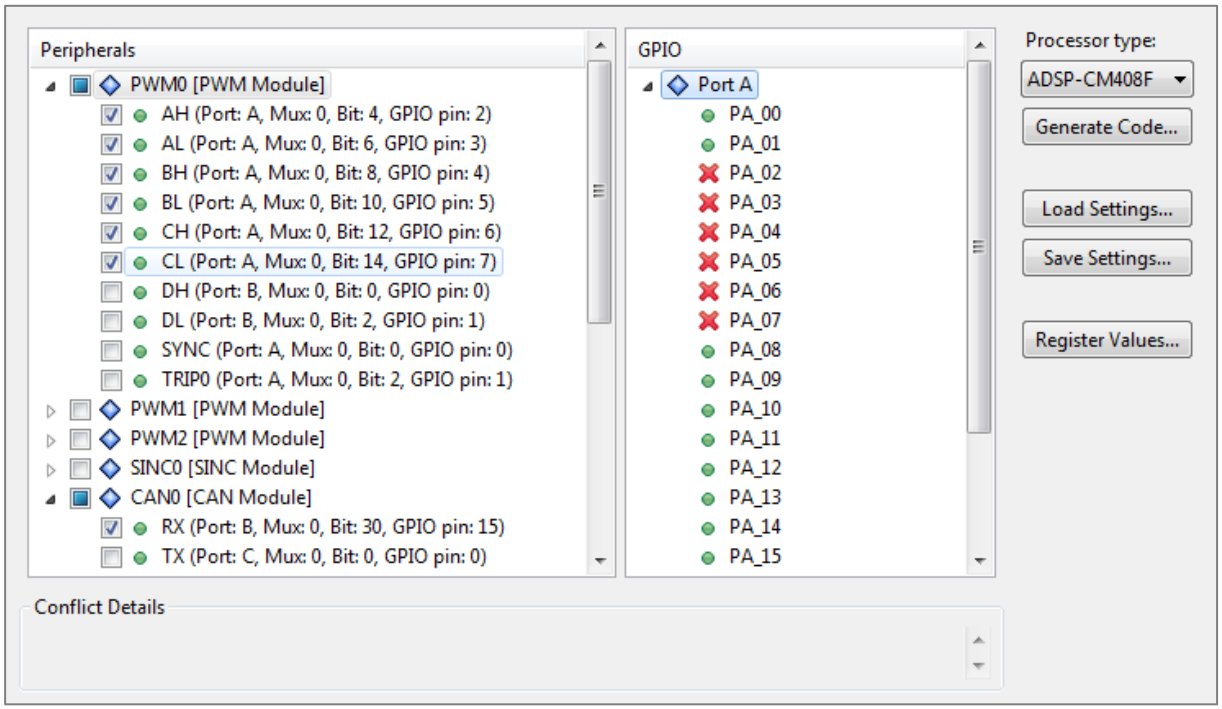


Abbildung 6.5 - Pinmultiplexing Tool

6.3.2 GPIO Ports

Der Mikrocontroller verfügt in der eingesetzten Version über 6 Ports (Port A bis F) mit insgesamt 91 General-purpose Input/Output Pins. Wie bereits in Kapitel 6.3.1 erwähnt, sind die Pins intern mit verschiedenen Funktionen belegt. Die Auswahl der verwendeten Pins ist in erster Linie davon abhängig, ob eine benötigte Funktion (z. B. UART-Rx oder PWM-Modul) damit verbunden ist. Ein großer Vorteil der ARM-Cortex-Architektur ist, dass alle GPIO-Pins interruptfähig sind. Dabei werden die GPIO-Ports wahlweise einem Interrupt zugewiesen (siehe Abbildung 6.6). Dadurch verfügt nicht jeder Pin über einen eigenen Interrupt, sondern ein ganzer Port (16 Pins pro Port mit Ausnahme von Port F) löst einen Interrupt aus. Daher muss in der Interrupt Service Routine (ISR) überprüft werden, welcher Pin den Interrupt generiert hat. Lösen mehrere Pins desselben Ports einen Interrupt aus, wird die ISR wiederholt durchlaufen, bis alle Bits im Latch-Register zurückgesetzt wurden. Dies verhindert, dass ein Interrupt verpasst wird. Das Pin-Interrupt Modul verfügt über ein Masken-Register für jeden Port, wodurch verhindert wird, dass ein Nicht-Interrupt-Pin, der auf dem gleichen Port mit einem Interrupt-Pin liegt, einen IRQ generiert.

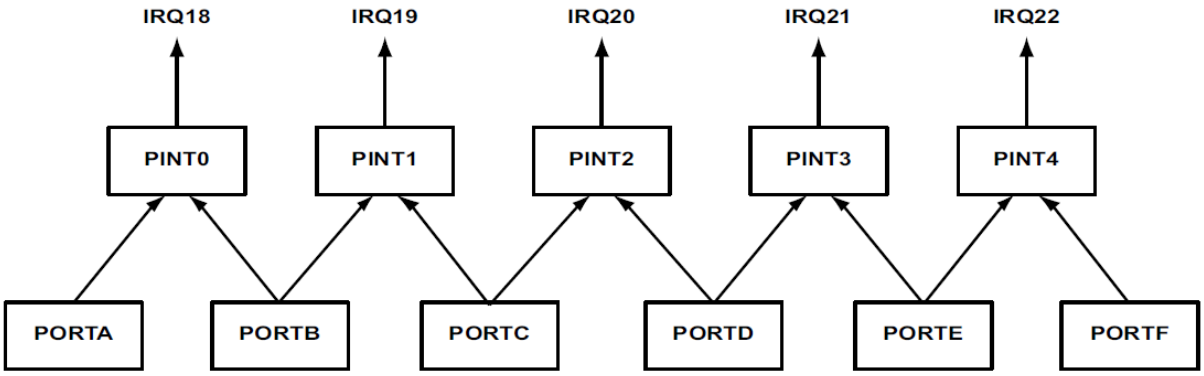


Abbildung 6.6 - Pin Interrupt [17]

Vor der Interrupt-Konfiguration erfolgt die Definition der Pin-Modi. Jeder Pin kann entweder im Input, Output oder Open-Drain Modus konfiguriert werden. Standardmäßig befinden sich die Pins nach dem Reset im Input-Modus, wobei die Eingangstreiber nicht aktiv sind. Dadurch werden unnötige Stromflüsse unterbunden und keine externen Pull-Widerstände an den unbelegten Pins benötigt.

Der umgesetzte Programmablauf der Konfiguration wird in Abbildung 6.7 dargestellt. Für Input-Pins mit Interruptzuweisung erfolgt die in Abbildung 6.8 dargestellte, erweiterte Konfiguration.

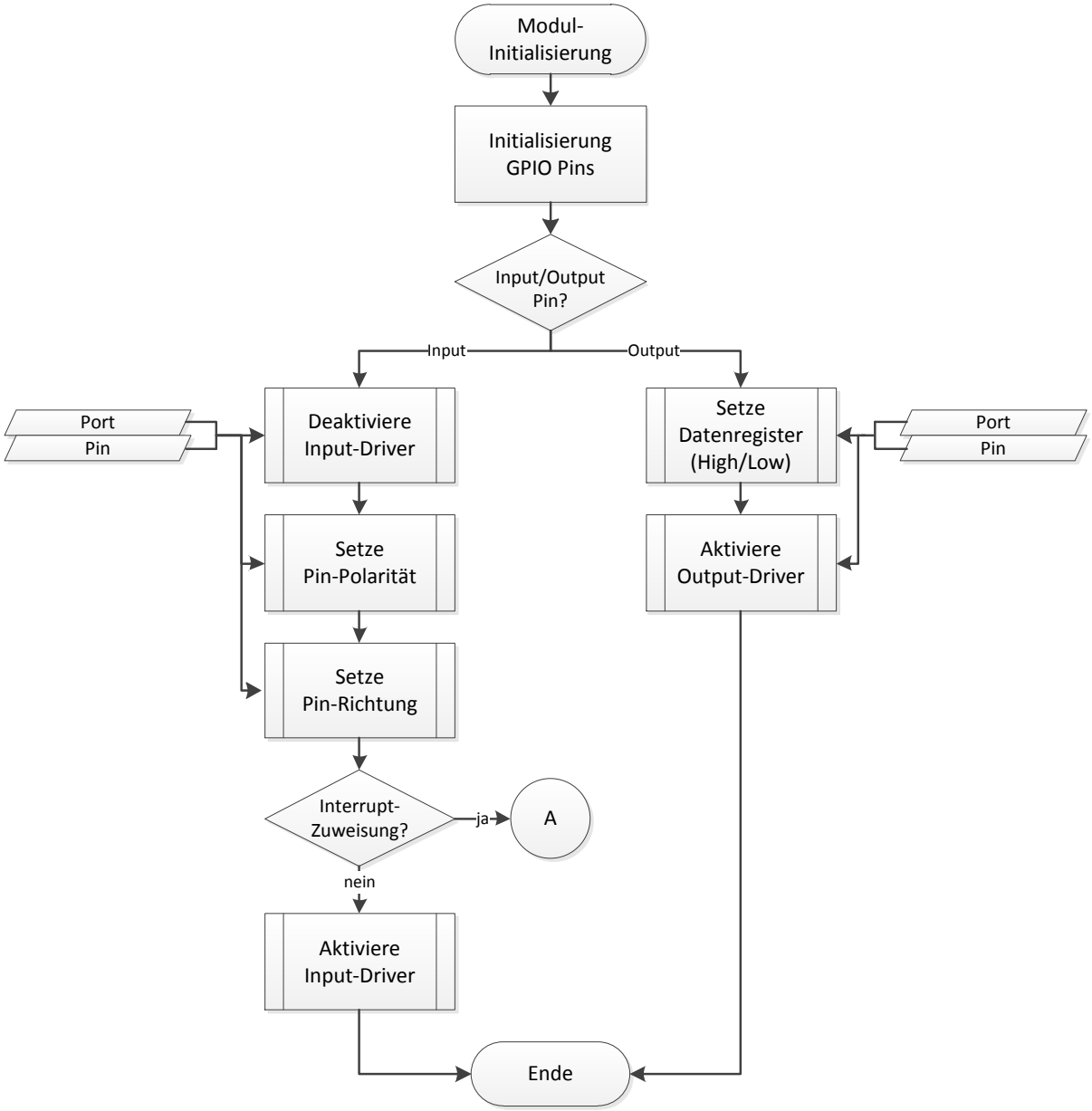


Abbildung 6.7 - GPIO Konfiguration

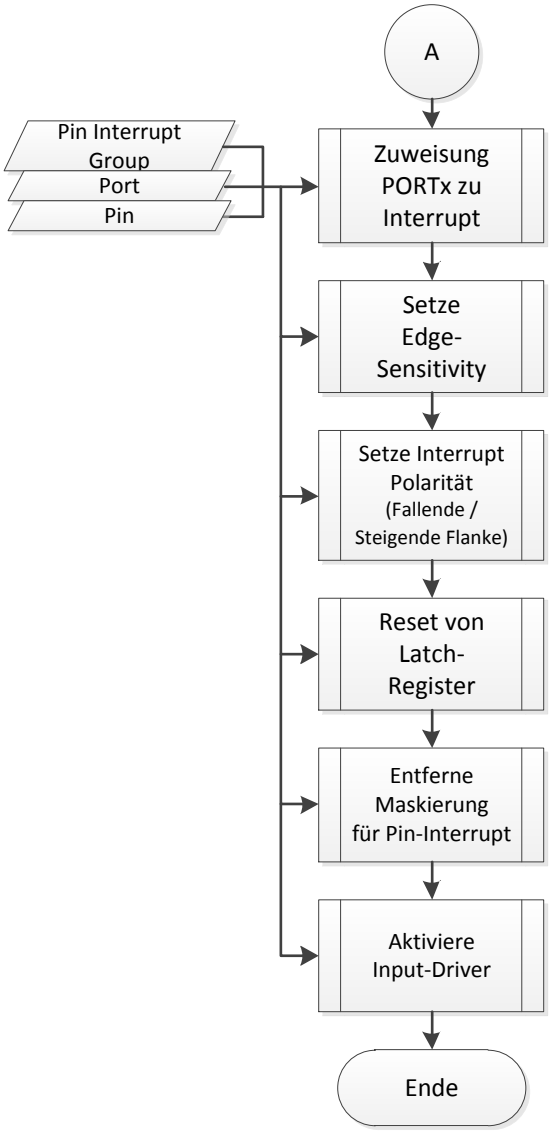


Abbildung 6.8 - GPIO Pin Interrupt

6.3.3 PWM-Modul

Der Mikrocontroller verfügt über ein fortgeschrittenes PWM-Modul, womit es möglich ist, PWM-Paare mit komplexen Signalverläufen ohne CPU-Intervention zu generieren. Es sollen drei PWM-Paare mit einer Frequenz von 50 kHz für die Motoransteuerung ausgegeben werden, deren Duty-Cycle in jeder PWM-Periode durch den Regelalgorithmus zeitgleich angepasst werden kann. Die PWM-Signale sollten symmetrisch sein, damit ein zeitgleiches Schalten der Wechselrichter-MOSFETs der drei Phasen verhindert wird.

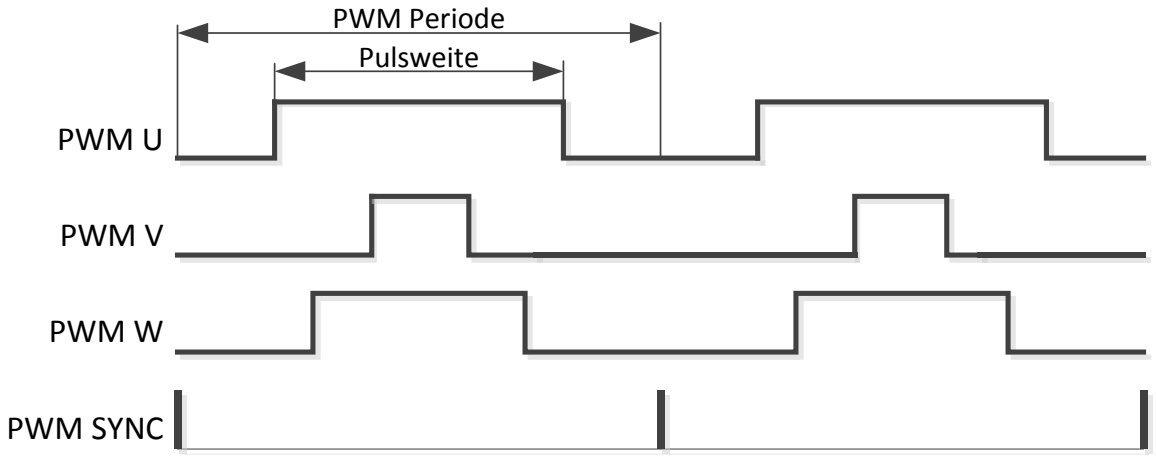


Abbildung 6.9 - PWM-Paare

Nachdem das PWM-Modul konfiguriert und die PWM-Ausgabe gestartet wurde, kann der Duty-Cycle jedes PWM-Paares (PWM U, V, W) durch eine Funktion mit Wertübergabe zwischen 0 und 100 % eingestellt werden. Nach jeder PWM-Periode wird der PWM-Sync Interrupt getriggert, in dessen ISR der Regelalgorithmus ausgeführt wird und die neu berechneten Duty-Cycles gesetzt werden.

Jedes PWM-Paar besteht aus einem Low- und High-Side-Kanal zur Ansteuerung der Halbbrücken. Aufgrund der realen Anstiegs- und Abfallzeiten der Halbbrücken-MOSFETs muss eine Totzeit zwischen dem Ausschalten des High-Side-MOSFETs und dem Einschalten des Low-Side-MOSFETs und umgekehrt vorgesehen werden, damit zu keinem Zeitpunkt beide MOSFETs gleichzeitig durchgeschaltet sind.

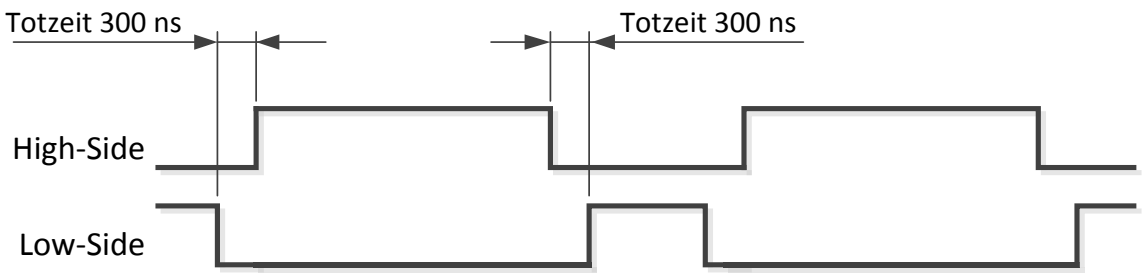


Abbildung 6.10 - PWM-Totzeit

Das PWM-Modul unterstützt diese Einstellungsmöglichkeiten durch Programmierung der entsprechenden Register, so ist nach der einmaligen Initialisierung keine Software-Intervention (bis auf die Pulsweitenveränderung) mehr nötig.

6.3.4 ADC-Controller

Der Mikrocontroller verfügt über einen ADC-Controller, der die Zugriffe auf die beiden 16-Bit ADCs verwaltet und den Sampling-Prozess automatisiert. Der Vorteil in der Verwendung dieses Controllers liegt darin, dass der komplette Sampling-Vorgang und die Übertragung der Daten in den Arbeitsspeicher durch den DMA automatisiert ablaufen und keine Kern-Intervention notwendig ist. Nachteilig ist eine relativ komplexe Konfiguration und ein intransparenter Sampling-Vorgang zu erwähnen.

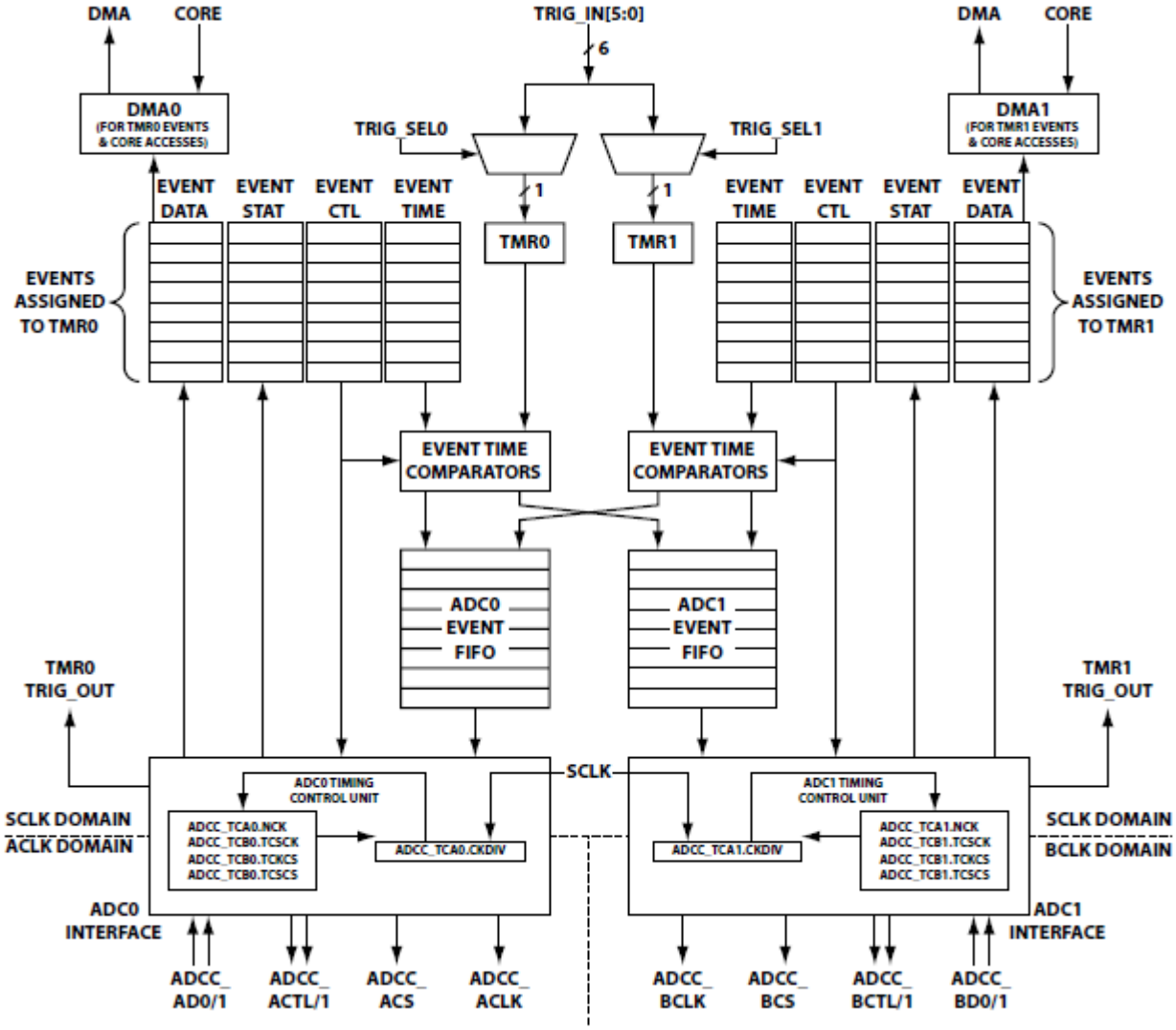


Abbildung 6.11 - ADC-Controller Blockdiagramm [17]

Abbildung 6.11 zeigt das Blockdiagramm des ADC-Controllers. Der Controller hat zwei interne Timer *TMR0* und *TMR1*, welche exklusiv für die ADC-Steuerung zur Verfügung stehen und mit dem System-Clock getaktet sind. Des Weiteren können

die Daten aus den Datenregistern mit den DMA-Modulen *DMA0* und *DMA1* in den Arbeitsspeicher übertragen werden. Es besteht die Möglichkeit einen Zirkularbuffer im Arbeitsspeicher umzusetzen. In den insgesamt 24 Eventregistern werden die Sampling-Zeitpunkte in Relation zum Timer-Start und die Zuweisung zu *TMR0* bzw. *TMR1* und einem ADC-Kanal festgelegt. Der Start der beiden Timer wird durch die TRU ausgelöst. Dazu muss ein Trigger-Master, dem Trigger-Slave-Eingang des ADC-Controllers zugewiesen werden. Als Trigger-Master bieten sich zwei General-Purpose-Timer an, die bei Ablauf Ihrer Perioden jeweils den Start der ADC-Timer auslösen. Der ADC-Timer wird automatisch gestoppt, sobald alle Events, die dem Timer zugewiesen sind, ausgeführt wurden. Abbildung 6.12 zeigt diesen Ablauf an einem Beispiel.

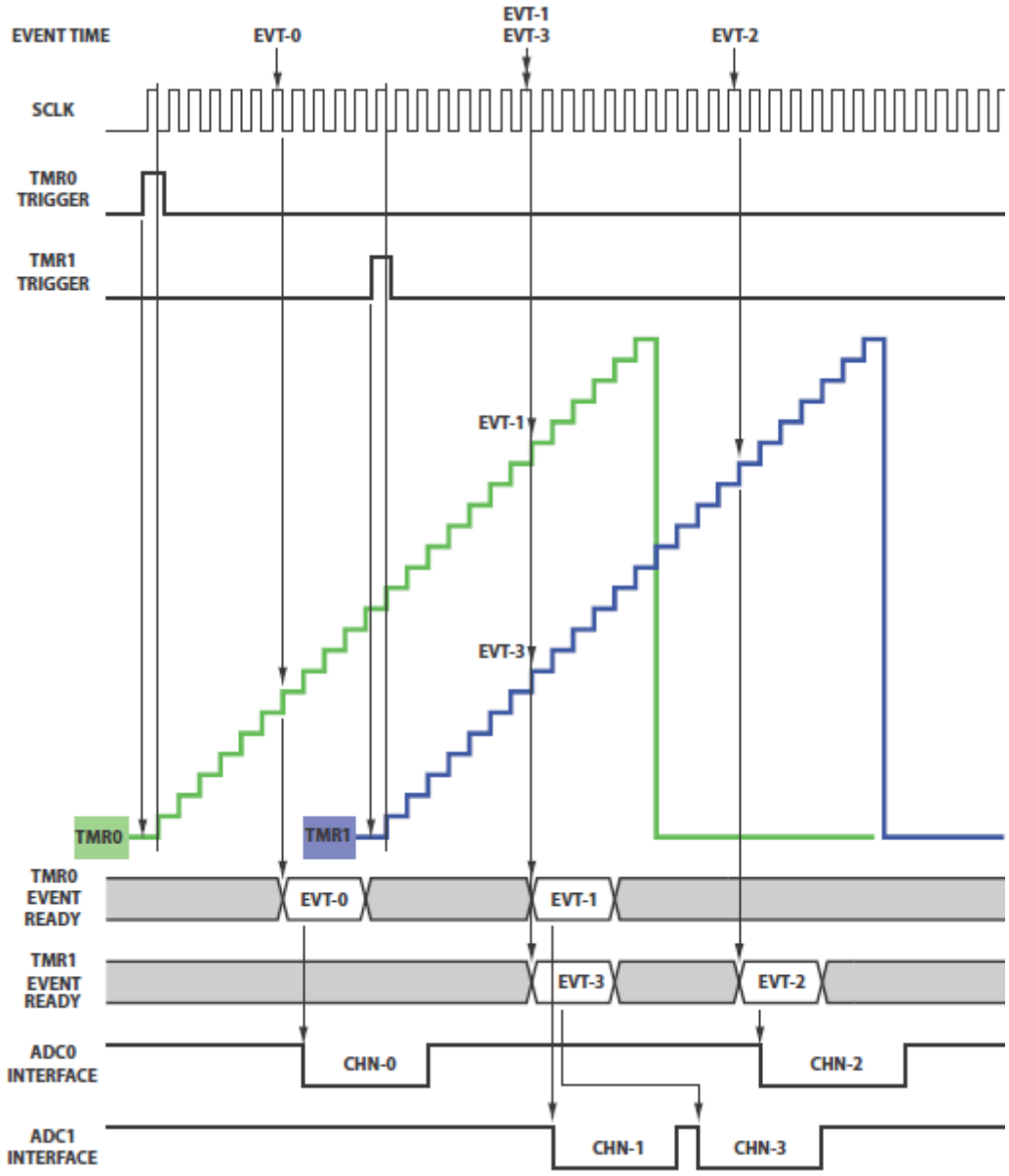


Abbildung 6.12 - ADC Event Timing [17]

Eine wichtige Funktion des Controllers wird hier mit Event 1 und 3 aufgezeigt. Die beiden Events haben denselben Startzeitpunkt und sind dem ADC1 zugewiesen. Der Controller verschiebt nun den Samplingzeitpunkt von Event 3 auf den frühest Möglichen und gibt einen Event-Collision-Fehler aus. Dies ist eine potentielle Fehlerquelle, die der Benutzer beachten muss, um eine fehlerhafte Daten-aufzeichnung zu vermeiden. Eine korrekte Event-Konfiguration hat also oberste Priorität.

In der Programmierung für dieses Projekt soll der ADC-Samplingvorgang maximiert werden. Das heißt, beide ADCs sollen bei maximalen Samplingraten simultan und kontinuierlich Daten aufzeichnen und diese per DMA in den Arbeitsspeicher übertragen, sodass die Stromwerte von zwei Motorphasen bei Ausführung des Regelalgorithmus in der PWM-ISR zur Verfügung stehen. Die Größe des Zirkularbuffers wird so ausgelegt, dass über eine PWM-Periode neue Werte aufgezeichnet werden. Durch Konfiguration der Timer, Trigger und Events, wie in Abbildung 6.13 dargestellt, lässt sich der gewünschte Samplingvorgang umsetzen.

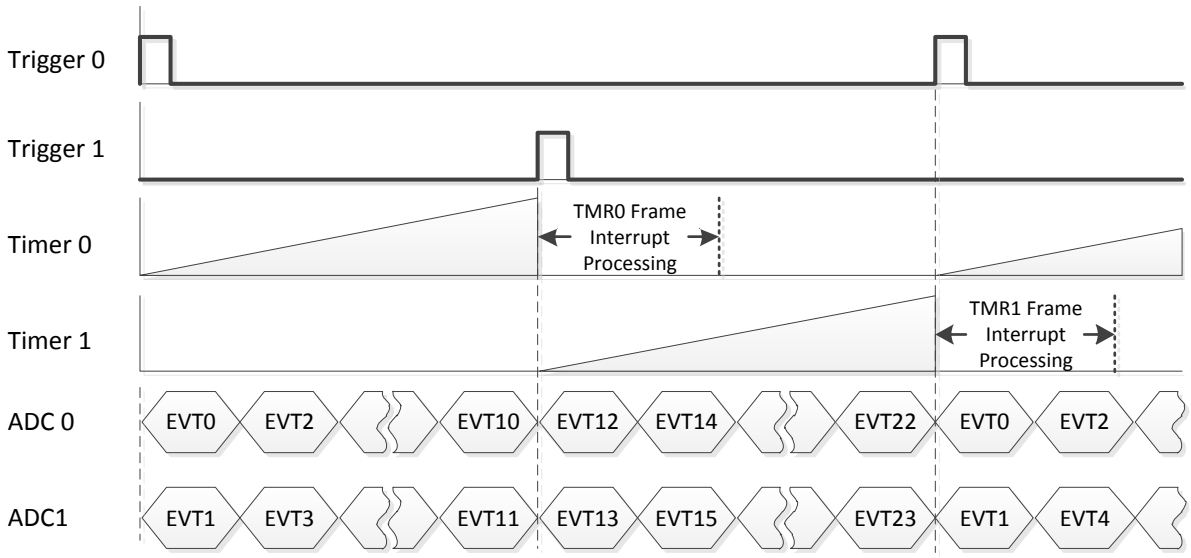


Abbildung 6.13 - ADC Event Programmierung

Es werden immer zwei Events simultan an den ADCs ausgeführt. Um einen kontinuierlichen Vorgang (über die 24 Events hinaus) zu gewährleisten, müssen beide ADC Timer mit einem Versatz der halben Periode zueinander ausgeführt werden, da sonst die Kern-Intervention nach jedem „Frame Complete“ Interrupt den Samplingvorgang unterbrechen würde. Die Events 0 bis 11 sind dabei dem *TMR0* zugewiesen, die Events 12 bis 23 dem *TMR1*. Der Frame Complete Interrupt dient einzig dem Zurücksetzen des „Frame Completion“-Bits, da andernfalls kein neuer

Frame vom ADC-Controller (ADCC) gestartet wird. Der Programmablauf zur Umsetzung der beschriebenen Konfiguration wird in Abbildung 6.14 dargestellt.

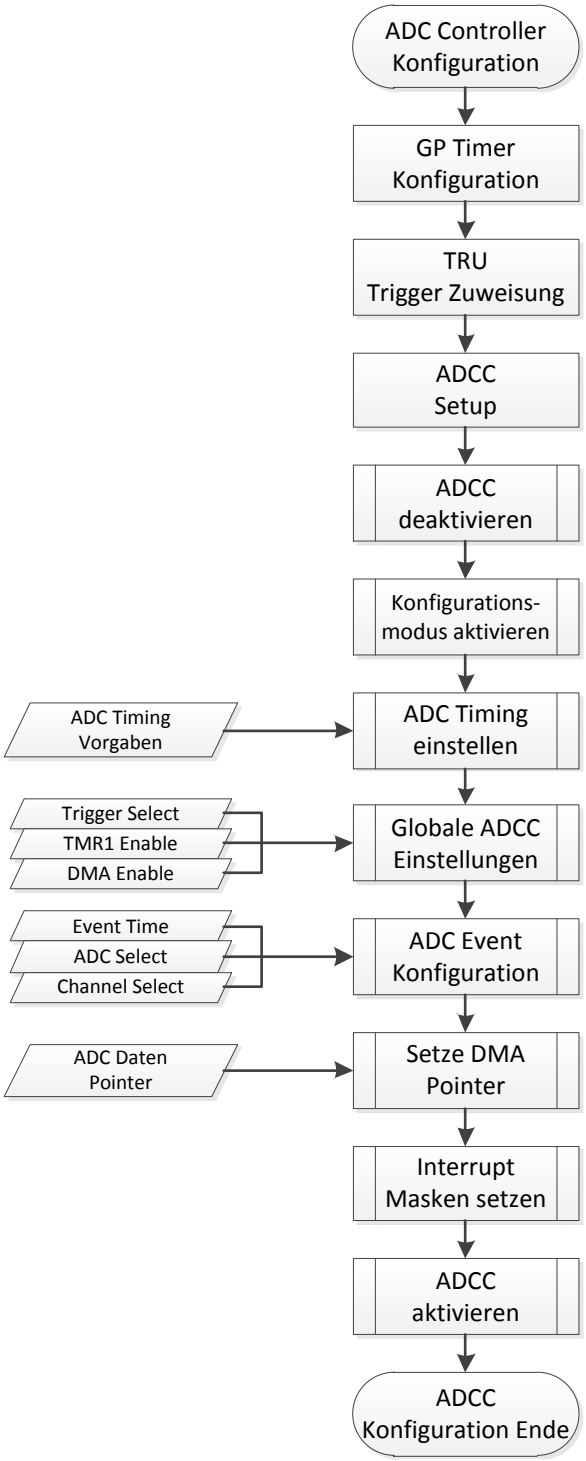


Abbildung 6.14 - ADC-Controller Konfiguration

Da der ADC-Controller in der Lage ist, kollidierende Events eigenständig zu verschieben, musste diese Konfiguration, die auf den maximalen Datendurchsatz ausgelegt ist (kein Abstand zwischen Events), auf Fehler in der Ausführung überprüft werden. Um den tatsächlichen Samplingzeitpunkt der ADCs festzustellen, wurde der in Abbildung 6.15 dargestellte Versuchsaufbau mit dem Evaluations-Board durchgeführt.

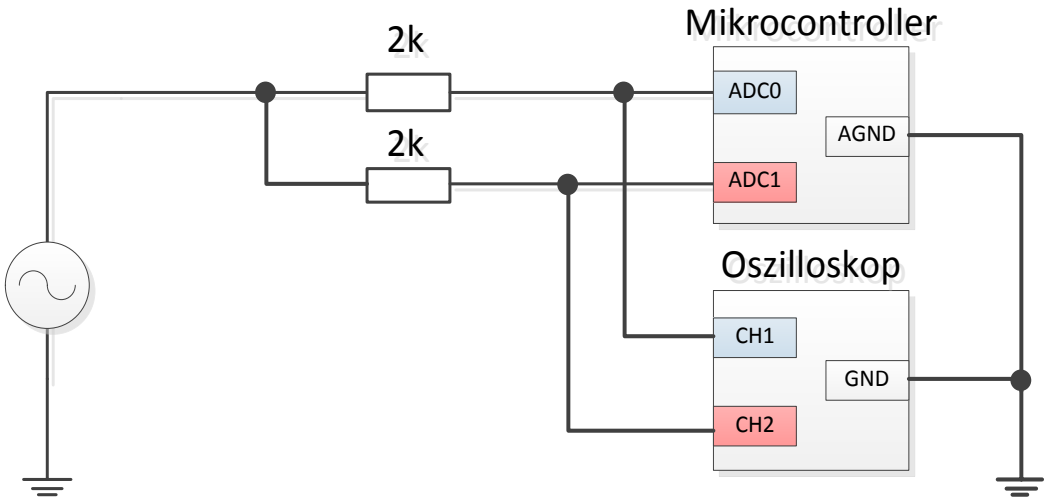


Abbildung 6.15 - ADC Versuchsaufbau

Wie in Kapitel 4.1.6 beschrieben, kann hier der Ladevorgang des internen Sample-Hold-Kondensators (siehe Abbildung 4.9) für den Zweck des Versuchs ausgenutzt werden. Mit einem relativ großen Widerstand am Eingang des ADCs wird die Spannung durch den Kondensator im Samplezeitpunkt einige mV auf einen niedrigeren Pegel gezogen. Mit einer hohen Auflösung am Oszilloskop, kann so sehr gut der tatsächliche Samplezeitpunkt der ADCs bestimmt werden.

Abbildung 6.16 zeigt die Oszilloskop-Aufzeichnung des kontinuierlichen Samplingvorgangs an den beiden ADCs. Zu Beginn der ADC-Aquisitionszeit bricht die Spannung durch den Ladevorgang des Sample-Hold-Kondensators um ca. 8 mV ein. In dieser Konfiguration wurde pro Kanal eine Samplingrate von 2,50 MSPS gemessen. Dies entspricht der erwarteten Samplingrate, da der mit 45 MHz getaktete ADC 18 Takte für eine Wandlung benötigt. Es ist auch ersichtlich, dass der Übergang zwischen den Frames (alle 12 Samples) ohne Verzögerung abläuft und somit einen kontinuierlichen Datenfluss gewährleistet.

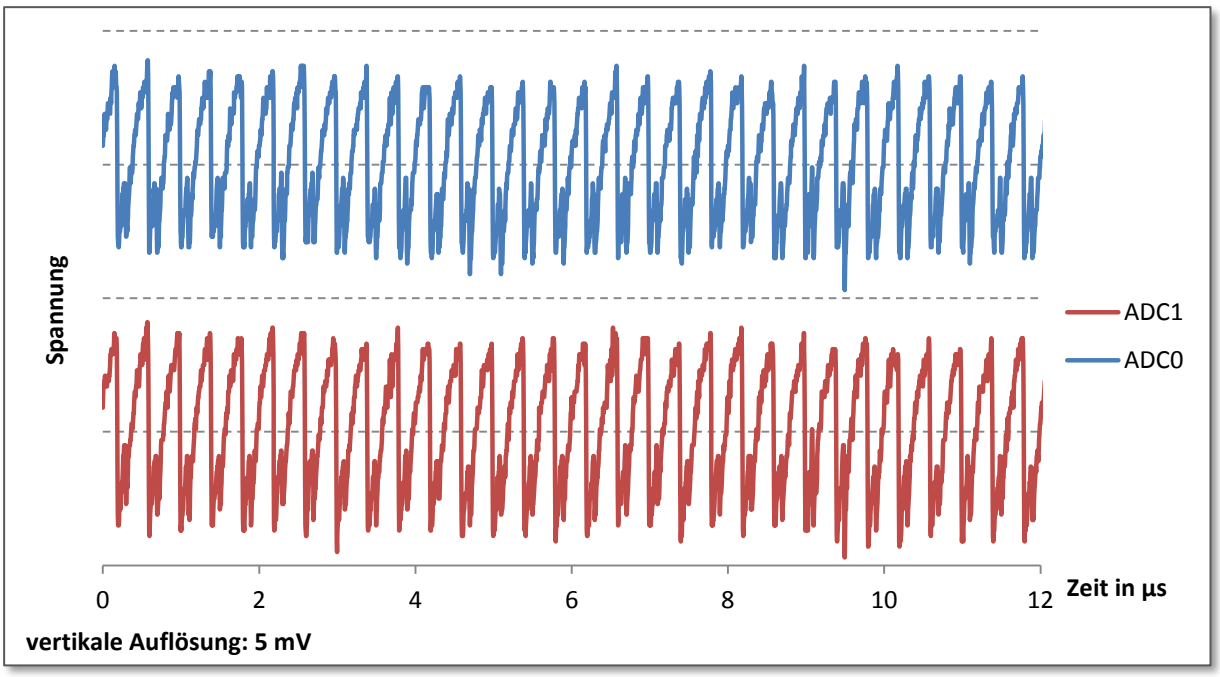


Abbildung 6.16 - ADC simultanes Sampling

Abbildung 6.17 zeigt die Aufzeichnung mit einer höheren zeitlichen Auflösung und bestätigt, dass beide ADCs den Samplingvorgang simultan ausführen.

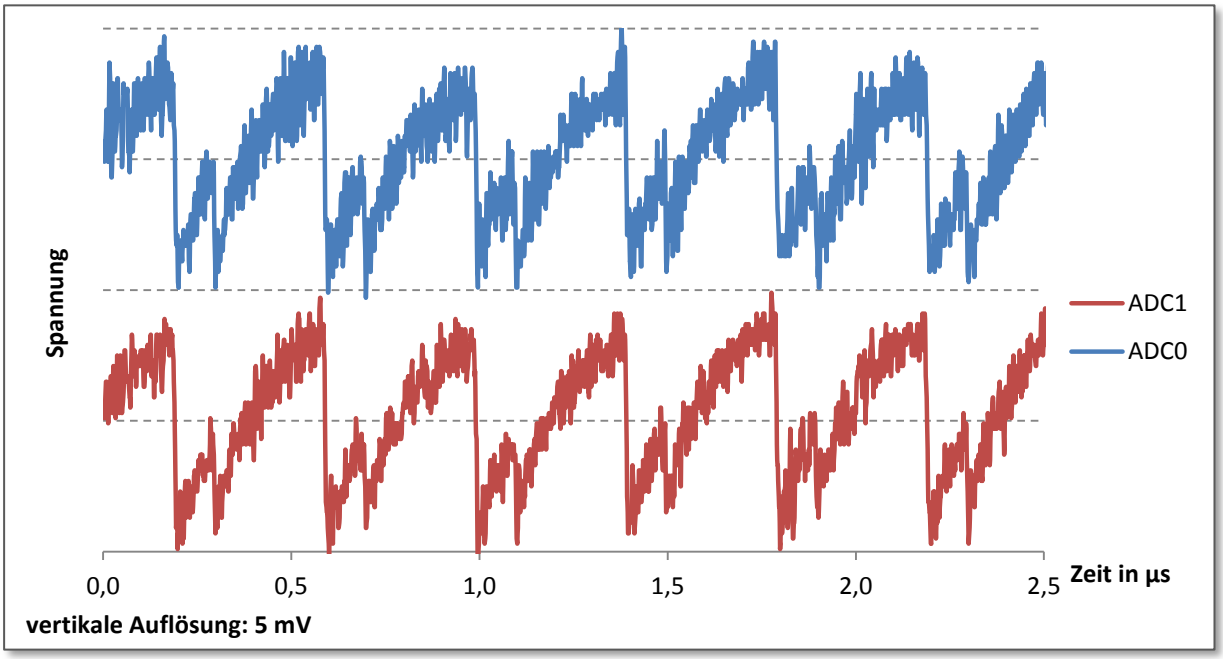


Abbildung 6.17 - ADC simultanes Sampling (Zoom)

Diese optimale Konfiguration konnte nur durch eine genaue Abstimmung der Eventzeiten, Timer und Trigger erzielt werden. Ist der ADC-Controller einmal konfiguriert und gestartet, erfolgt im normalen Programmablauf kein Zugriff mehr darauf, sondern es müssen lediglich die aktuellen Messdaten aus dem Arbeitsspeicher gelesen werden.

Abbildung 6.18 zeigt die Aufzeichnung einer weniger erfolgreichen Konfiguration. Der erste Frame wird noch korrekt ausgeführt, es kommt dann zu einer Eventkollision aufgrund eines falsch programmierten Trigger-Intervalls und der ADC-Controller verschiebt die restlichen Events. Wie sich diese Wartezeiten zwischen den Events ergeben, ist nicht nachvollziehbar. Dieses Verhalten zeigt einen wesentlichen Nachteil des ADC-Controllers auf, da es keine Rückmeldung gibt, wann ein Event tatsächlich ausgeführt wurde und der komplette Samplingvorgang dadurch intransparent wird. Ist der Controller allerdings korrekt konfiguriert, kann davon ausgegangen werden, dass es zu keinen Verzögerungen der Samplingzeitpunkte kommt.

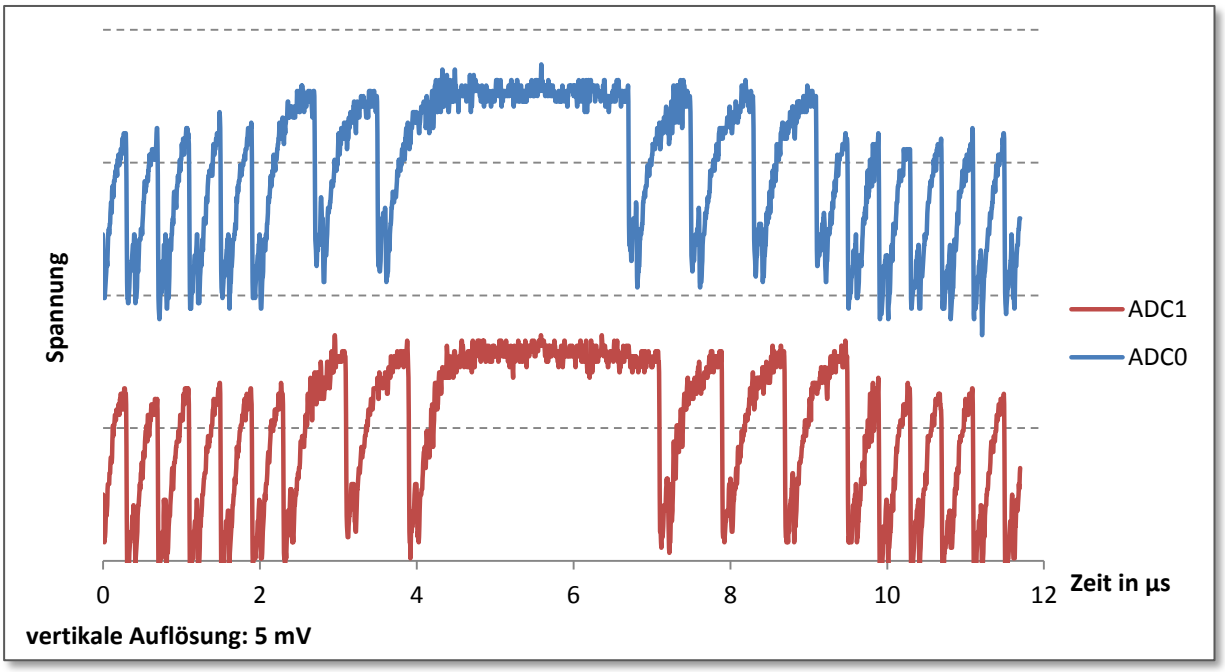


Abbildung 6.18 - ADC Sampling bei inkorrekter Konfiguration

6.3.5 UART-Controller

Der Mikrocontroller verfügt über einen UART-Controller mit drei UART-Ports, wovon einer für den Empfang der Sensordaten am Sensorbus verwendet wird. Im Verlauf der Entwicklung des Massagebetts wurden auch die Sensoren weiterentwickelt. Für genauere Informationen hinsichtlich der Umsetzung und Funktion der Sensoren wird auf die Arbeit aus diesem Bereich verwiesen [1].

Die Datenpakete haben eine Länge von 7 Byte und werden in einem Intervall von 100 µs (≙ 10 kHz) mit einer Bitrate von 937,5 kHz vom Sensor an den Mikrocontroller gesendet. Abbildung 6.19 zeigt das Sensorprotokoll für ein Datenpaket der aktuell neuesten Sensorversion.

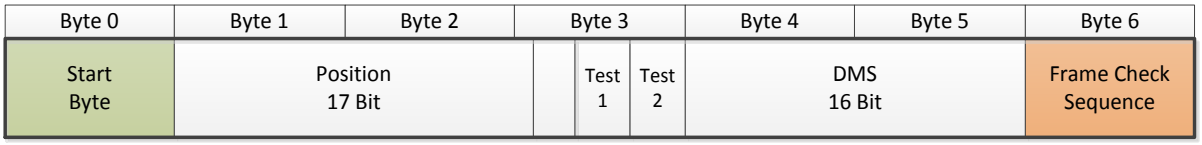


Abbildung 6.19 - Sensor Datenpaket

Der UART-Controller verfügt über einen DMA-Transfermodus, der genutzt werden kann, um die Kern-Intervention beim Datenempfang auf ein Minimum zu reduzieren. Da nicht ausgeschlossen werden kann, dass der erste Empfang am Mikrocontroller inmitten eines Datenpakets beginnt, muss der UART-Controller zu Beginn im *Core-Modus* betrieben werden. Der UART-Controller triggert nach dem Empfang eines Bytes den *Receive-Interrupt* und die Software überprüft in der ISR, ob es sich bei diesem Byte um das Start-Byte des Datenpakets handelt. Nach dem Empfang eines gültigen Start-Bytes und sechs weiteren Bytes, wird der UART-Controller in den *DMA-Modus* geschaltet. Im *DMA-Modus* wird eine definierte Anzahl Bytes aus dem Empfangsbuffer in den Arbeitsspeicher transferiert und erst dann ein Interrupt getriggert. So muss die Software erst nach dem Empfang eines kompletten Datenpakets eingreifen und die Sensordaten verarbeiten. Sollten die Daten fehlerhaft sein, das heißt, es liegt beispielsweise ein falsches Start-Byte vor, oder der CRC liefert ein negatives Ergebnis, so wird der UART-Controller wieder in den *Core-Modus* gewechselt und auf den Empfang eines gültigen Datenpakets gewartet. Der beschriebene Programmablauf des Interrupt-Handlers wird in Abbildung 6.20 dargestellt.

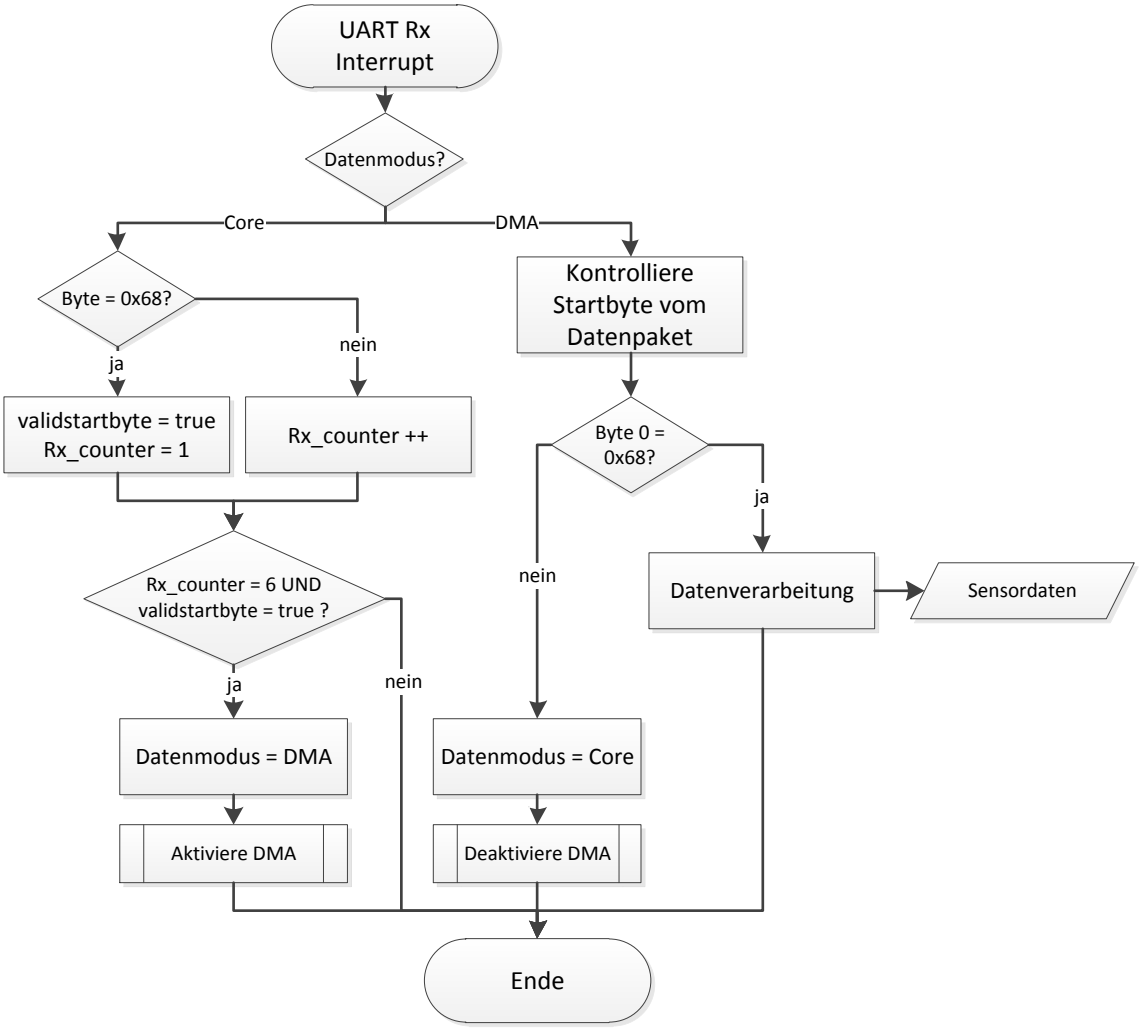


Abbildung 6.20 - UART Interrupt-Handler

6.3.6 CAN-Controller

Der Mikrocontroller verfügt über ein CAN-Modul mit zwei Interfaces, die dem CAN 2.0B Standard entsprechen. Für die Kommunikation am Main-Bus wird ein CAN-Port benötigt. Da die Umsetzung des Main-Bus CAN-Interface (siehe Abbildung 2.1) und die damit verbundene Umstellung des gesamten Bus auf den CAN-Standard in einer nachfolgenden Arbeit umgesetzt werden soll, wurde in der Programmierung des CAN-Controllers eine empfangs- und sendebereite Konfiguration erstellt, deren Code zu einem späteren Zeitpunkt durch entsprechende datenverarbeitende Routinen ergänzt werden soll.

Der CAN-Controller verfügt über 32 Nachrichten-Puffer, die als „Mailboxes“ bezeichnet werden und sich aus den, in Abbildung 6.21 dargestellten, Registern zusammensetzen.

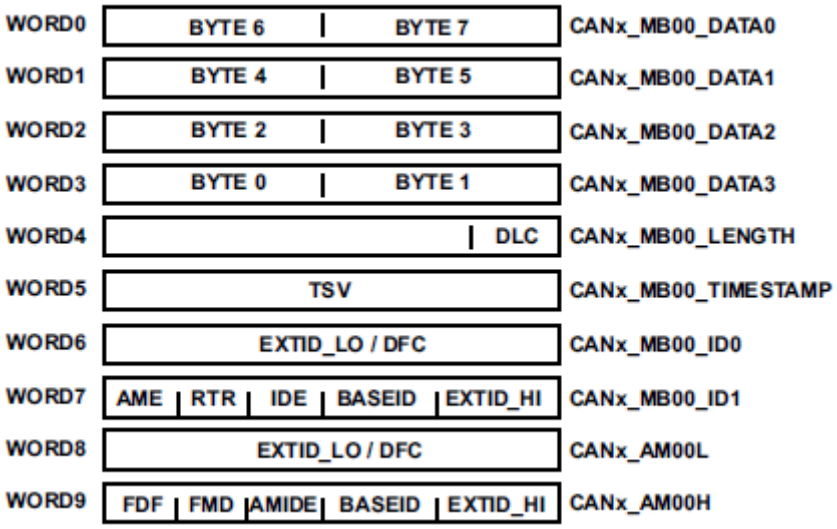


Abbildung 6.21 - CAN Mailbox [17]

Jede Mailbox besteht aus acht 16-Bit Kontroll- und Datenregistern sowie zwei optionalen „Acceptance Mask“ Registern, die zur Akzeptanzfilterung der eingehenden Nachrichten verwendet werden können. Jeweils acht Mailboxes sind fix für den Empfangs- und Sendevorgang voreingestellt. Die Richtung der übrigen 16 Mailboxes kann einzeln programmiert werden.

Der CAN-Controller empfängt autonom und missachtet ungültige Nachrichten. Geht eine gültige Nachricht ein, überprüft die Empfangslogik auf eine Übereinstimmung der Mailbox-Akzeptanzfilterung. Ist eine Nachricht nicht für das Modul bestimmt, wird sie ignoriert. Der Controller-interne Empfangsprozess wird in Abbildung 6.22 verdeutlicht.

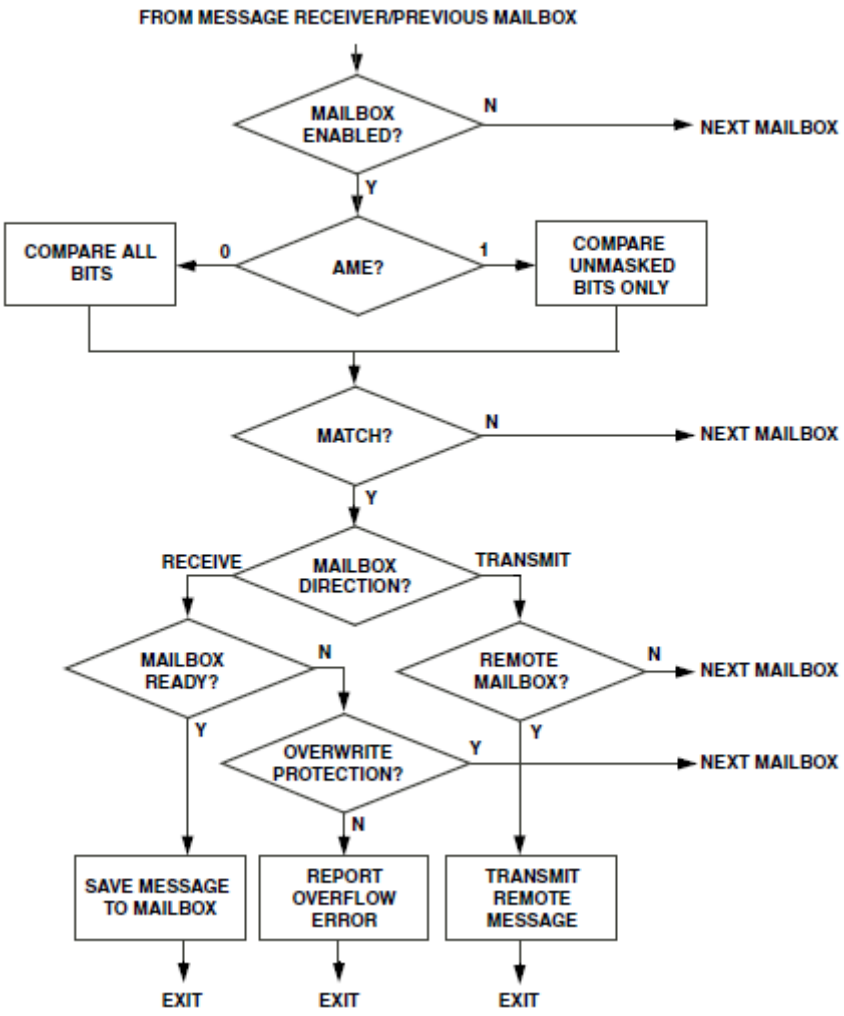


Abbildung 6.22 - CAN Nachrichtenempfang [17]

Jede Mailbox kann bei entsprechend programmierter Interrupt-Maske den *Receive-Interrupt* auslösen sobald eine Nachricht in der Mailbox abgelegt wurde. Das hat zur Folge, dass der Kern bzw. die Software erst beim Eingang einer gültigen Nachricht intervenieren muss, um die empfangenen Daten zu verarbeiten und zu interpretieren. Es besteht außerdem die Möglichkeit, den Watchdog-Modus des CAN-Controllers zu verwenden. Dabei muss innerhalb einer definierten Periode eine gültige Nachricht in Mailbox 4 abgelegt werden, andernfalls wird ein Fehler-Interrupt ausgelöst. Dieser Modus eignet sich dazu, die Funktion des Main-Bus zu kontrollieren und eine Notabschaltung im Fehlerfall auszulösen.

7 Zusammenfassung und Ausblick

Ziel dieser Diplomarbeit war die Entwicklung einer Motorsteuerung für das mehrachsige Antriebssystem des Massagebetts am Institut für Mechanik und Mechatronik. Die entwickelte Steuerplatine soll als Ersatz für die bisher eingesetzte Version, mit einer klaren Aufwertung der Rechenleistung sowie einem robusteren Kommunikationssystem, dienen.

Zuerst wurde ein neuer Schaltplan entwickelt, der die Anforderungen für einen leistungsstärkeren Mikrocontroller und einem Datennetz nach dem CAN-Standard erfüllt. Zahlreiche Verbesserungen wurden auch in allen anderen Schaltungsbereichen vorgenommen. Hauptaugenmerk lag dabei in der Behebung von bekannten Problemen, die im Betrieb der Vorgängerversion festgestellt wurden. So wurden Maßnahmen für eine Verbesserung der Signalqualität durch neue Komponenten und Änderungen im Schaltungs-Design vorgenommen.

Es folgte der Prozess der PCB-Layout Entwicklung durch ein vierschichtiges Platinen-Design einer 160 mm x 100 mm Leiterplatte. Nach der professionellen Anfertigung einer Anzahl an Platinen-Prototypen durch einen Kleinserien-Hersteller konnte der erste Prototyp bestückt werden.

Die Programmierung der Software wurde über den gesamten Verlauf der Entwicklungszeit auf einem Evaluations-Kit des Mikrocontroller-Herstellers durchgeführt. Zu Beginn der Entwicklung beschränkte sich die Lieferbarkeit des Mikrocontrollers auf das Evaluations-Kit mit einer voll bestückten Platine um alle Module testen zu können. Wie sich im Laufe der Diplomarbeit herausstellte, sollte sich die Marktveröffentlichung des Mikrocontrollers auf das Frühjahr 2016[4] verzögern, wodurch auch davor keine Musterstücke des Mikrocontrollers verfügbar waren. Eine Demontage des Evaluations-Kits wurde in Erwägung gezogen. Da aber das Mikrocontrollergehäuse mit einer verlöteten Massenfläche auf der Gehäuseunterseite (thermal Pad) ausgestattet ist, konnte ein thermischer Schaden dieser und auch anderer Komponenten auf der Platine nicht ausgeschlossen werden. Die erstellte Software beschränkt sich aus diesem Grund auf die mit dem Evaluations-Kit prüfbaren Bereiche. Es wurden alle benötigten Module und Komponenten konfiguriert und ein grundlegender Programmablauf gestaltet. In einem weiteren Entwicklungsschritt und Thema einer wissenschaftlichen Arbeit soll die Programmierung des Regelalgorithmus für die Motorsteuerung sowie die Umstellung des Kommunikations-Interface auf den CAN-Standard umgesetzt werden.

[4] Stand zum Verfassungszeitpunkt, der tatsächliche Veröffentlichungstermin kann sich ändern.

8 Literaturverzeichnis

[1] K. Bergkirchner, „Dissertation: Entwicklung eines mechatronischen therapeutischen Massagesystems,“ TU Wien, Wien: Institut für Mechanik und Mechatronik, in Ausführung.

[2] P. Finsterwalder, „Dissertation: Analyse und mechanisches Konzept therapeutisch wirksamer Massagebewegungen,“ TU Wien, Wien: Institut für Mechanik und Mechatronik, 2015.

[3] ARM Holdings, „ARMv7-M Architecture Reference Manual,“ Cambridge, 2010.

[4] Analog Devices Inc., „ADSP-CM402F/CM403F/CM407F/CM408F/CM409F: Mixed-Signal Control Processor with ARM Cortex-M4,“ August 2014. [Online]. Available: http://www.analog.com/media/en/technical-documentation/data-sheets/ADSP-CM402F_CM403F_CM407F_CM408F_CM409F.pdf. [Zugriff am 22 April 2015].

[5] Robert Bosch GmbH, „CAN Specification 2.0,“ September 1991. [Online]. Available: http://www.bosch-semiconductors.de/media/pdf_1/canliteratur/can2spec.pdf. [Zugriff am 23 April 2015].

[6] ISO 11898-2:2003, Part 2: High-speed medium access unit.

[7] Texas Instruments Inc., „Delfino Microcontrollers TMS320F28335 (Data Manual),“ Juni 2007. [Online]. Available: http://www.ti.com/lit/gpn/tms320f28335. [Zugriff am 24 April 2015].

[8] Altium Limited, „Altium Designer,“ [Online]. Available: http://www.altium.com/. [Zugriff am 29 April 2015].

[9] Würth Elektronik, „Würth Common Mode Line Filter,“ [Online]. Available: http://katalog.we-online.de/pbs/datasheet/744222.pdf. [Zugriff am 29 April 2015].

[10] Microchip Technology, „MCP6V26/7/8 Datasheet,“ 2011. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/25007B.pdf. [Zugriff am 04 Mai 2015].

[11] Texas Instruments Inc., „TLV2372 Datasheet,“ März 2001. [Online]. Available:

http://www.ti.com/lit/ds/symlink/tlv2371.pdf. [Zugriff am 04 Mai 2015].

[12] Texas Instruments Inc., „Application Report SLLA270 - Controller Area Network Physical Layer Requirements," Januar 2008. [Online]. Available: http://www.ti.com/lit/an/slla270/slla270.pdf. [Zugriff am 05 Mai 2015].

[13] Würth Elektronik, „Common Mode Line Filter 744227," Juli 2014. [Online]. Available: http://katalog.we-online.de/pbs/datasheet/744227.pdf. [Zugriff am 06 Mai 2015].

[14] F. Terman, Radio Engineers Handbook, New York: McGraw-Hill, 1945.

[15] Texas Instruments Inc., „PowerPAD™ Layout Guidelines Application Report SLOA120," Mai 2006. [Online]. Available: http://www.ti.com/lit/an/sloa120/sloa120.pdf. [Zugriff am 14 05 2015].

[16] ARM Ltd., „CMSIS - Cortex Microcontroller Software Interface Standard," [Online]. Available: http://www.arm.com/products/processors/cortex-m/cortex-microcontroller-software-interface-standard.php. [Zugriff am 18 Mai 2015].

[17] Analog Devices In.c, „ADSP-CM40x Mixed-Signal Control Processor with ARM Cortex-M4 Hardware Reference Rev. 0.2," September 2013. [Online]. Available: http://www.analog.com/media/en/dsp-documentation/processor-manuals/ADSP-CM40x_hrm_rev.0.2.pdf. [Zugriff am 19 Mai 2015].

9 Abbildungsverzeichnis

10 Tabellenverzeichnis

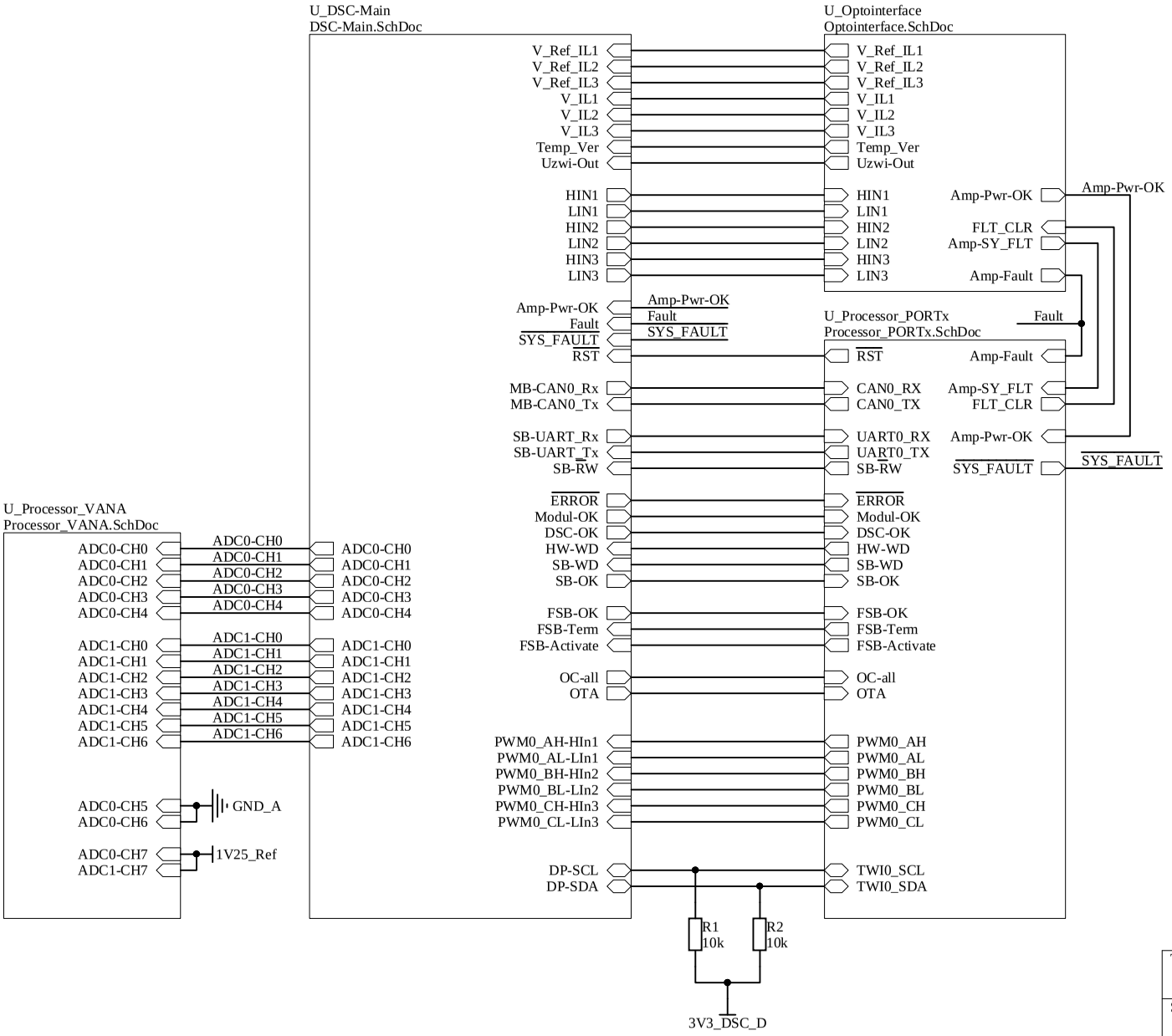
11 Abkürzungsverzeichnis

ADC ..... *Analog-to-Digital-Converter*
ADCC ..... *ADC-Controller*
CAN ..... *Controller Area Network*
CMSIS ..... *Cortex Microcontroller Software Interface Standard*
CRC ..... *Cyclic Redundancy Check*
DMA ..... *Direct Memory Access*
DMS ..... *Dehnungsmessstreifen*
DSC ..... *Digital Signal Controller*
DSP ..... *Digital Signal Processor*
ESD ..... *Electrostatic Discharge*
FIR ..... *Finite Impulse Response*
FPU ..... *Floating Point Unit*
GCC ..... *GNU Compiler Collection*
HAL ..... *Hardware Abstraction Layer*
I²C ..... *Inter-Integrated Circuit*
IRQ ..... *Interrupt Request*
ISR ..... *Interrupt Service Routine*
MOSFET ..... *Metall-Oxid-Halbleiter-Feldeffekttransistor*
NVIC ..... *Nested Vectored Interrupt Controller*
OPV ..... *Operationsverstärker*
PCB ..... *Printed Circuit Board*
PWM ..... *Pulsweitenmodulation*
RAM ..... *Random-Access Memory*
SPICE ..... *Simulation Program with Integrated Circuit Emphasis*
TRU ..... *Trigger Routing Unit*
TVS ..... *Transient Voltage Surpressor*
UART ..... *Universal Asynchronous Receiver Transmitter*

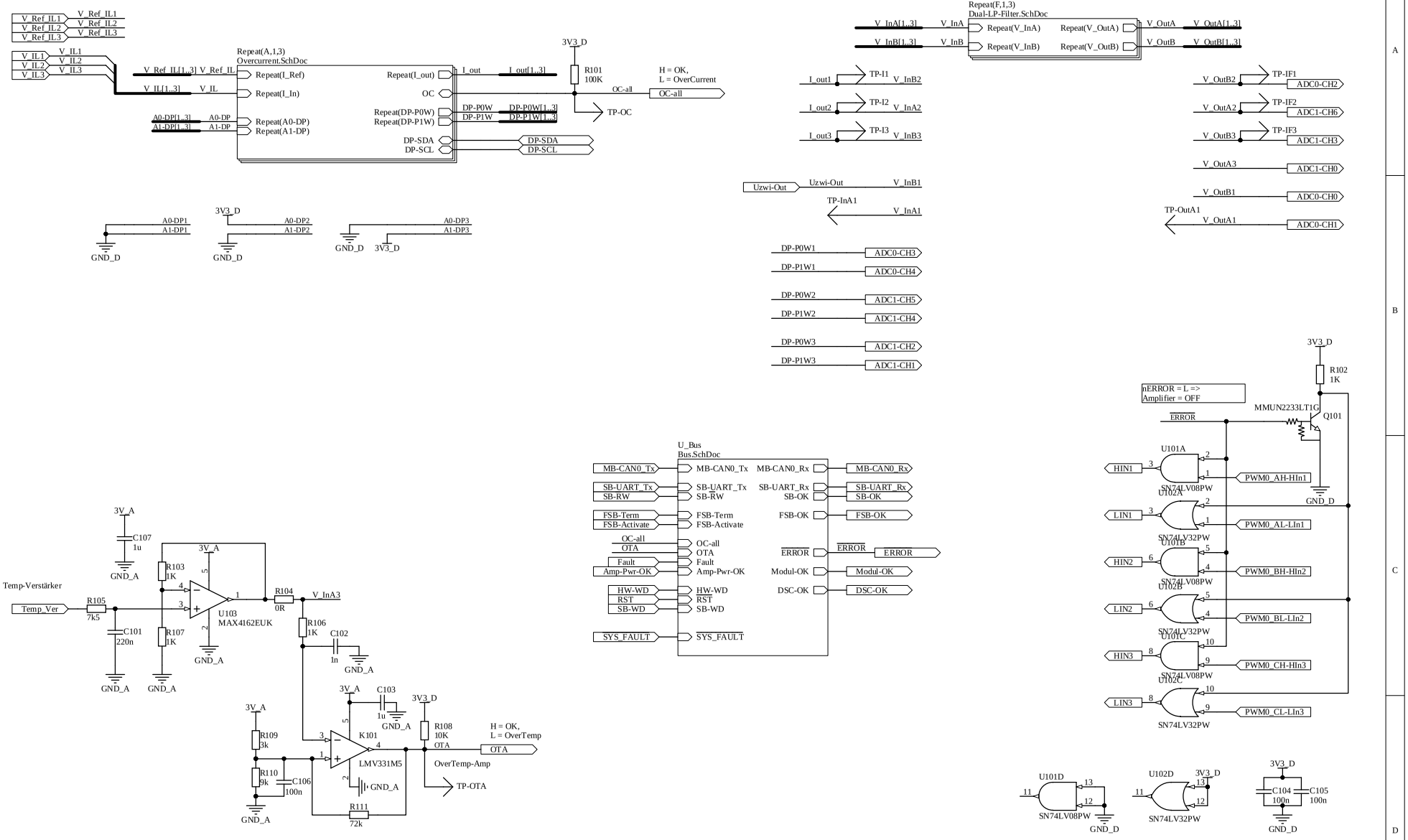
12 Anhang

12.1 Schaltplan

Anschließend folgt der vollständige, mit *Altium Designer* erstellte, Schaltplan der Controllerplatine.

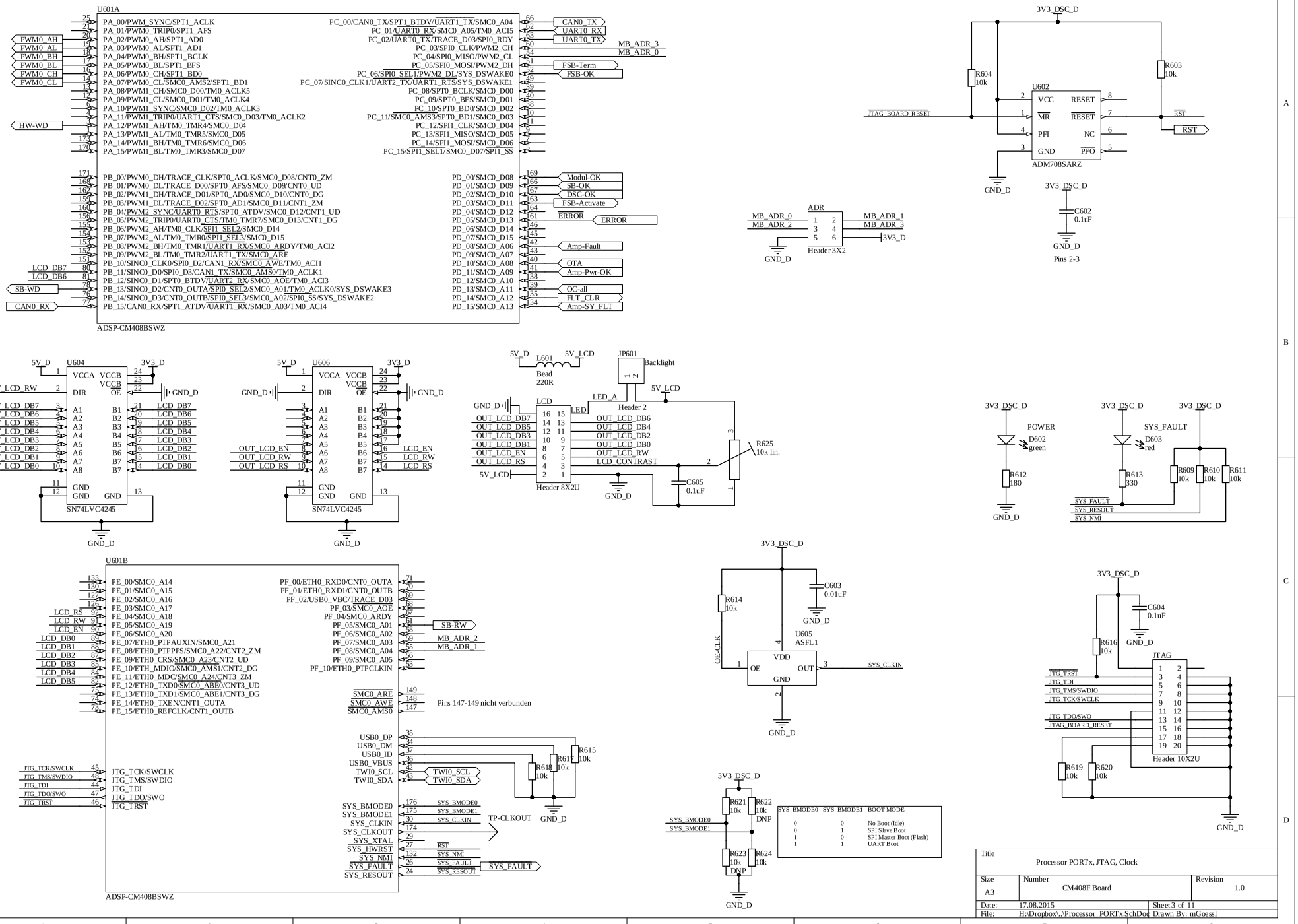


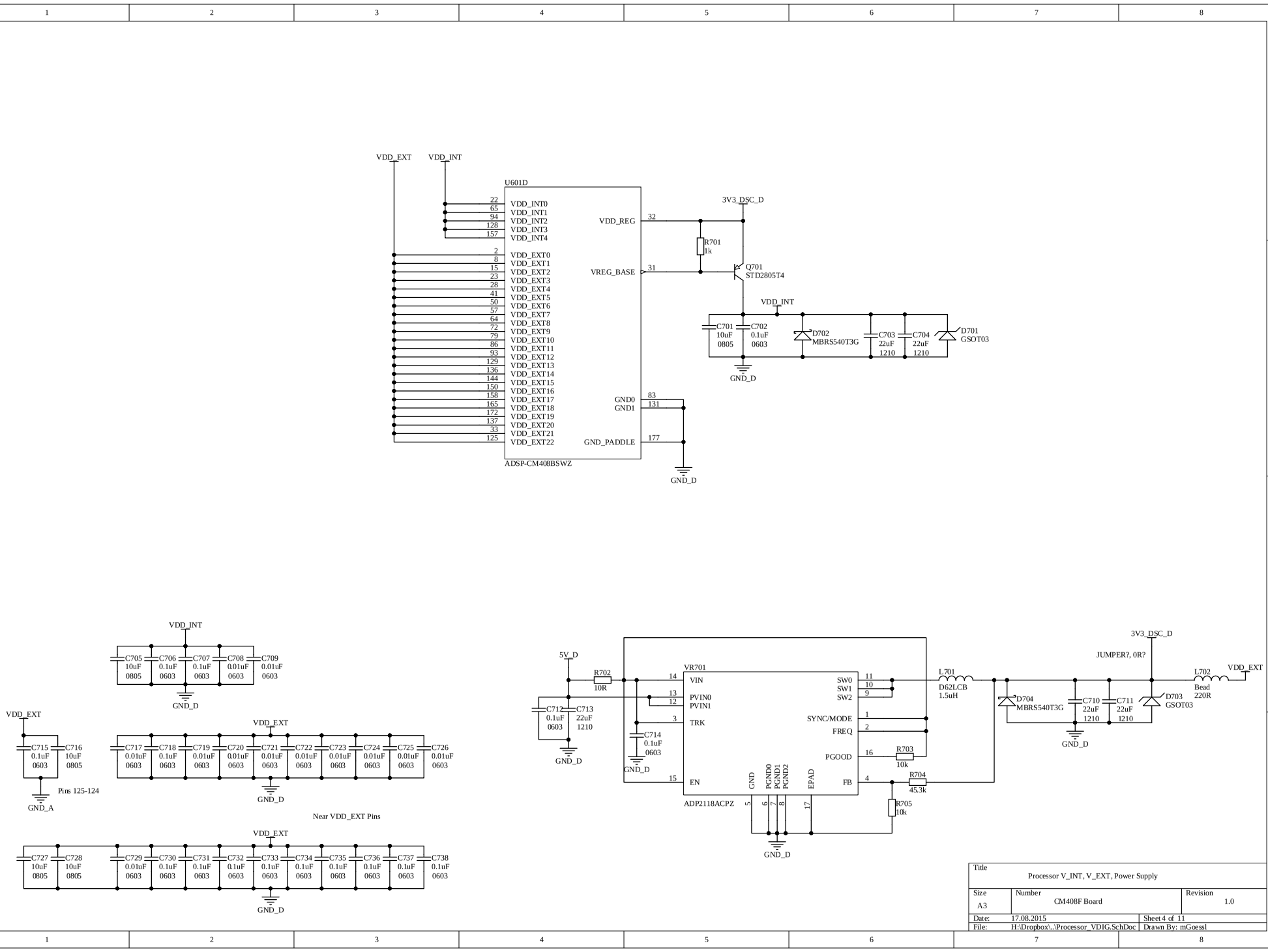
Title	TopSheet	
Size: A4	Number: CM408F Board	Revision: 1.0
Date: 17.08.2015		Sheet 1 of 11
File: H:\Dropbox\..\Topsheet.SchDoc		Drawn By: mGoessl

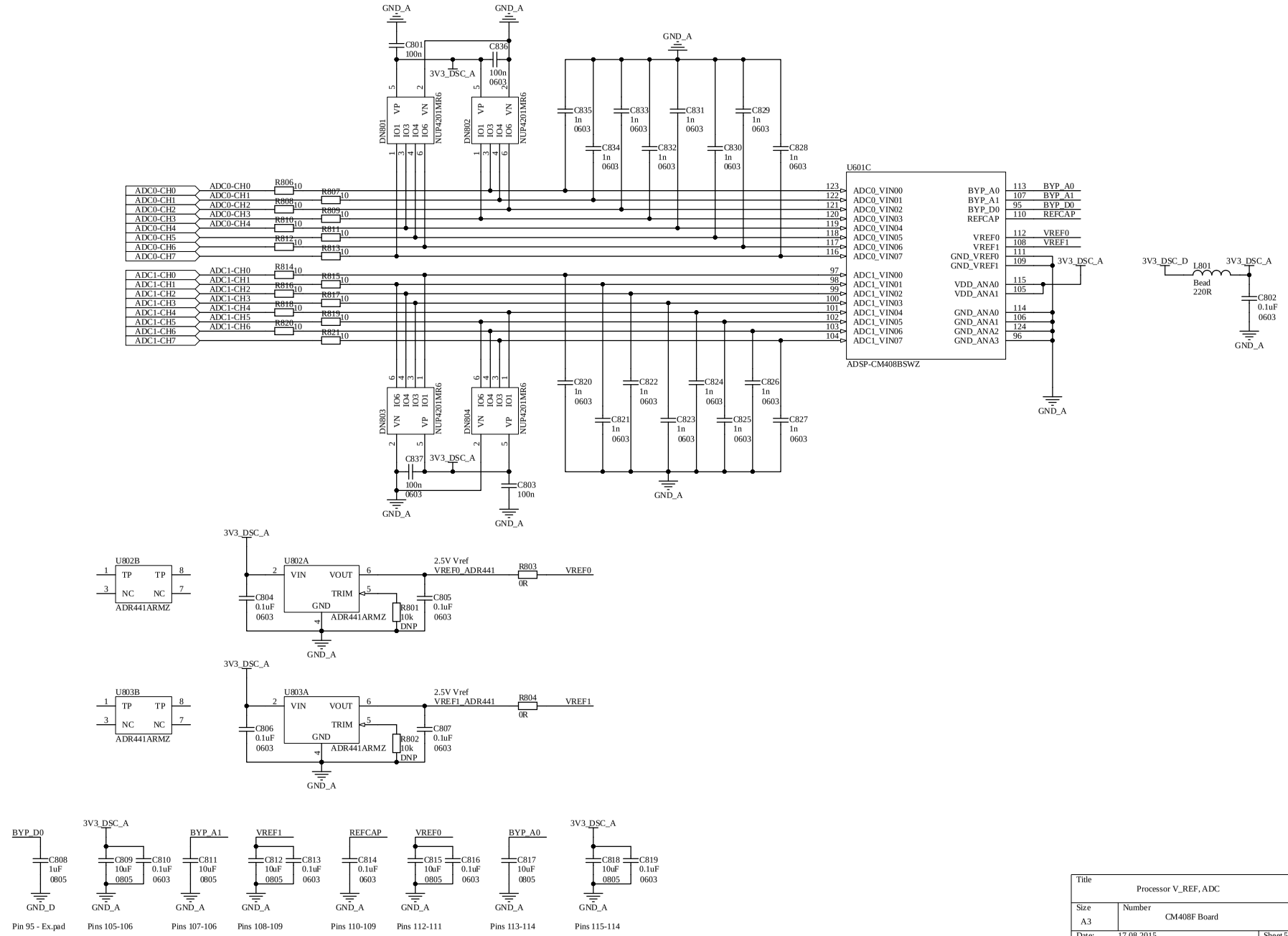


Werte für R109, R110 und R111 müssen angepasst werden

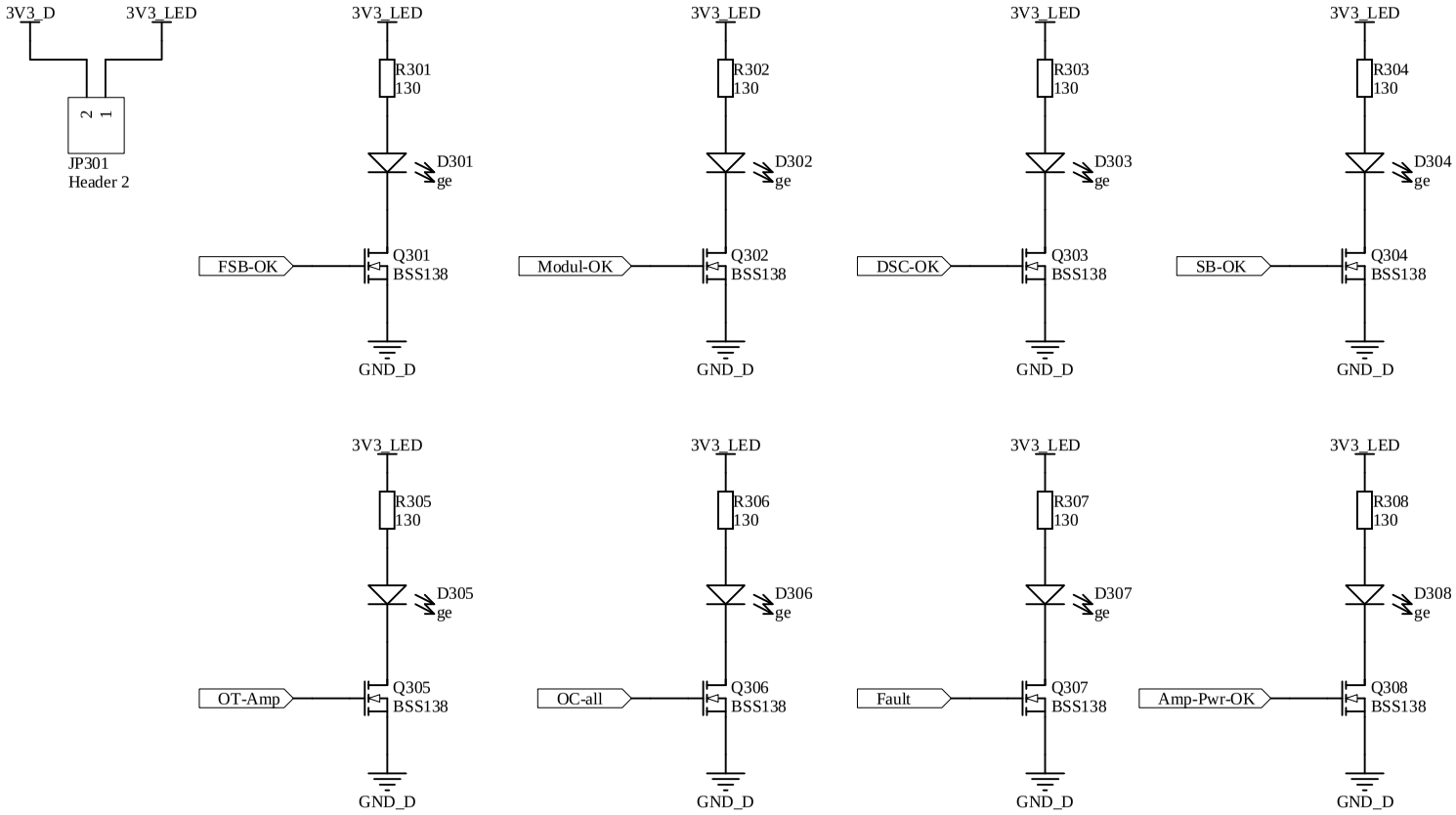
Title: DSC-Main		
Size: A3	Number: CM408F Board	Revision: 1.0
Date: 17.08.2015		Sheet 2 of 11
File: H:\Dropbox\..\DSC-Main.SchDoc		Drawn By: mGoessl


U601A
ADSP-CM408BSWZ
PWM0_AH
PWM0_AL
PWM0_BH
PWM0_BL
PWM0_CH
PWM0_CL
HW-WD
LCD_DB7
LCD_DB6
SB-WD
CAN0_RX
CAN0_TX
UART0_RX
UART0_TX
MB_ADR_3
MB_ADR_0
FSB-Term
FSB-OK
Modul-OK
SB-OK
DSC-OK
FSB-Activate
ERROR
Amp-Fault
OTA
Amp-Pwr-OK
OC-all
FLT_CLR
Amp-SY_FLT
3V3_DSC_D
R604
10k
R603
10k
U602
JTAG_BOARD_RESET
RST
ADM708SARZ
GND_D
C602
0.1uF
Pins 2-3
ADR
MB_ADR_0
MB_ADR_2
MB_ADR_1
MB_ADR_3
Header 3X2
5V_D
U604
U606
SN74LVC4245
OUT_LCD_RW
OUT_LCD_DB7
OUT_LCD_DB6
OUT_LCD_DB5
OUT_LCD_DB4
OUT_LCD_DB3
OUT_LCD_DB2
OUT_LCD_DB1
OUT_LCD_DB0
OUT_LCD_EN
OUT_LCD_RW
OUT_LCD_RS
L601
Bead
220R
5V_LCD
JP601
Backlight
Header 2
LCD
LED_A
LED
OUT_LCD_CONTRAST
R625
10k lin.
C605
0.1uF
Header 8X2U
POWER
D602
green
R612
180
SYS_FAULT
D603
red
R613
330
R609
10k
R610
10k
R611
10k
SYS_RESOUT
SYS_NMI
U601B
PE_00/SMC0_A14
PE_01/SMC0_A15
PE_02/SMC0_A16
PE_03/SMC0_A17
PE_04/SMC0_A18
PE_05/SMC0_A19
PE_06/SMC0_A20
PE_07/ETH0_PTPAUXIN/SMC0_A21
PE_08/ETH0_PTPPPS/SMC0_A22/CNT2_ZM
PE_09/ETH0_CRS/SMC0_A23/CNT2_UD
PE_10/ETH_MDIO/SMC0_AMS1/CNT2_DG
PE_11/ETH0_MDC/SMC0_A24/CNT3_ZM
PE_12/ETH0_TXD0/SMC0_ABE0/CNT3_UD
PE_13/ETH0_TXD1/SMC0_ABE1/CNT3_DG
PE_14/ETH0_TXEN/CNT1_OUTA
PE_15/ETH0_REFCLK/CNT1_OUTB
PF_00/ETH0_RXD0/CNT0_OUTA
PF_01/ETH0_RXD1/CNT0_OUTB
PF_02/USB0_VBC/TRACE_D03
PF_03/SMC0_AOE
PF_04/SMC0_ARDY
PF_05/SMC0_A01
PF_06/SMC0_A02
PF_07/SMC0_A03
PF_08/SMC0_A04
PF_09/SMC0_A05
PF_10/ETH0_PTPCLKIN
SB-RW
MB_ADR_2
MB_ADR_1
SMC0_ARE
SMC0_AWE
SMC0_AMS0
Pins 147-149 nicht verbunden
USB0_DP
USB0_DM
USB0_ID
USB0_VBUS
TWI0_SCL
TWI0_SDA
R615
10k
R616
10k
R617
10k
SYS_BMODE0
SYS_BMODE1
SYS_CLKIN
SYS_CLKOUT
SYS_XTAL
SYS_HWRST
SYS_NMI
SYS_FAULT
SYS_RESOUT
TP-CLKOUT
JTG_TCK/SWCLK
JTG_TMS/SWDIO
JTG_TDI
JTG_TDO/SWO
JTG_TRST
3V3_DSC_D
R614
10k
C603
0.01uF
U605
ASFL1
OE-CLK
VDD
OE
OUT
GND
SYS_CLKIN
C604
0.1uF
R616
10k
JTAG
Header 10X2U
R619
10k
R620
10k
R621
10k
R622
10k
DNP
R623
10k
R624
10k
SYS_BMODE0 SYS_BMODE1 BOOT MODE
0 0 No Boot (Idle)
0 1 SPI Slave Boot
1 0 SPI Master Boot (Flash)
1 1 UART Boot
Title
Processor PORTx, JTAG, Clock
Size
A3
Number
CM408F Board
Revision
1.0
Date: 17.08.2015
Sheet 3 of 11
File: H:\Dropbox\..\Processor_PORTx.SchDoc
Drawn By: mGoessl

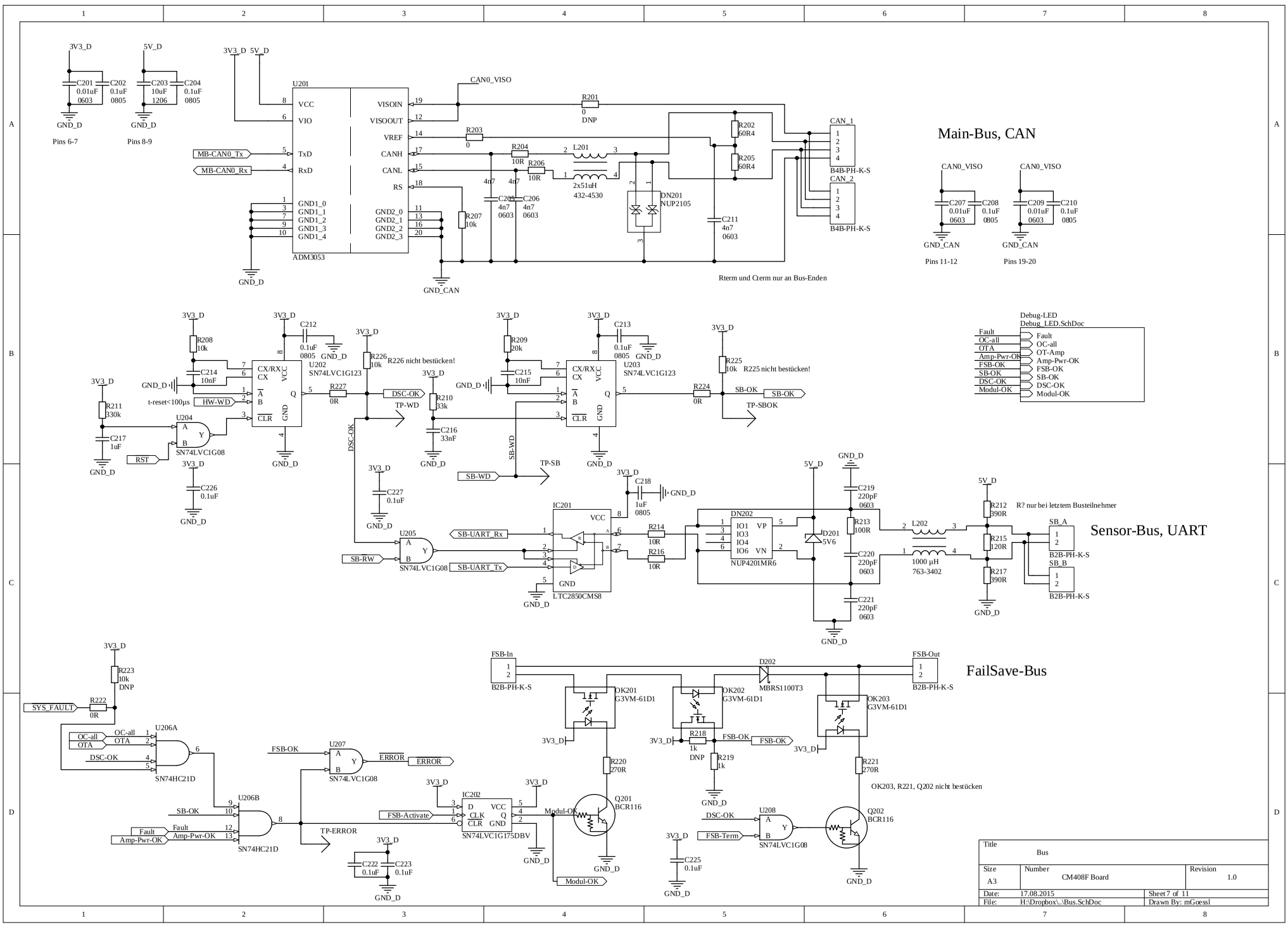

VDD_EXT
VDD_INT
U601D
VDD_INT0
VDD_INT1
VDD_INT2
VDD_INT3
VDD_INT4
VDD_EXT0
VDD_EXT1
VDD_EXT2
VDD_EXT3
VDD_EXT4
VDD_EXT5
VDD_EXT6
VDD_EXT7
VDD_EXT8
VDD_EXT9
VDD_EXT10
VDD_EXT11
VDD_EXT12
VDD_EXT13
VDD_EXT14
VDD_EXT15
VDD_EXT16
VDD_EXT17
VDD_EXT18
VDD_EXT19
VDD_EXT20
VDD_EXT21
VDD_EXT22
VDD_REG
VREG_BASE
GND0
GND1
GND_PADDLE
ADSP-CM408BSWZ
3V3_DSC_D
R701
1k
Q701
STD2805T4
C701
10uF
0805
C702
0.1uF
0603
D702
MBRS540T3G
C703
22uF
1210
C704
22uF
1210
D701
GSOT03
GND_D
C705
10uF
0805
C706
0.1uF
0603
C707
0.1uF
0603
C708
0.01uF
0603
C709
0.01uF
0603
VDD_EXT
C715
0.1uF
0603
C716
10uF
0805
Pins 125-124
GND_A
C717
0.01uF
0603
C718
0.1uF
0603
C719
0.01uF
0603
C720
0.01uF
0603
C721
0.01uF
0603
C722
0.01uF
0603
C723
0.01uF
0603
C724
0.01uF
0603
C725
0.01uF
0603
C726
0.01uF
0603
Near VDD_EXT Pins
C727
10uF
0805
C728
10uF
0805
C729
0.01uF
0603
C730
0.1uF
0603
C731
0.1uF
0603
C732
0.1uF
0603
C733
0.1uF
0603
C734
0.1uF
0603
C735
0.1uF
0603
C736
0.1uF
0603
C737
0.1uF
0603
C738
0.1uF
0603
5V_D
R702
10R
C712
0.1uF
0603
C713
22uF
1210
VR701
VIN
PVIN0
PVIN1
TRK
EN
SW0
SW1
SW2
SYNC/MODE
FREQ
PGOOD
FB
GND
PGND0
PGND1
PGND2
EPAD
ADP2118ACPZ
C714
0.1uF
0603
R703
10k
R704
45.3k
R705
10k
L701
D62LCB
1.5uH
D704
MBRS540T3G
C710
22uF
1210
C711
22uF
1210
D703
GSOT03
JUMPER?, 0R?
L702
Bead
220R
Title
Processor V_INT, V_EXT, Power Supply
Size
A3
Number
CM408F Board
Revision
1.0
Date:
17.08.2015
Sheet 4 of 11
File:
H:\Dropbox\..\Processor_VDIG.SchDoc
Drawn By: mGoessl

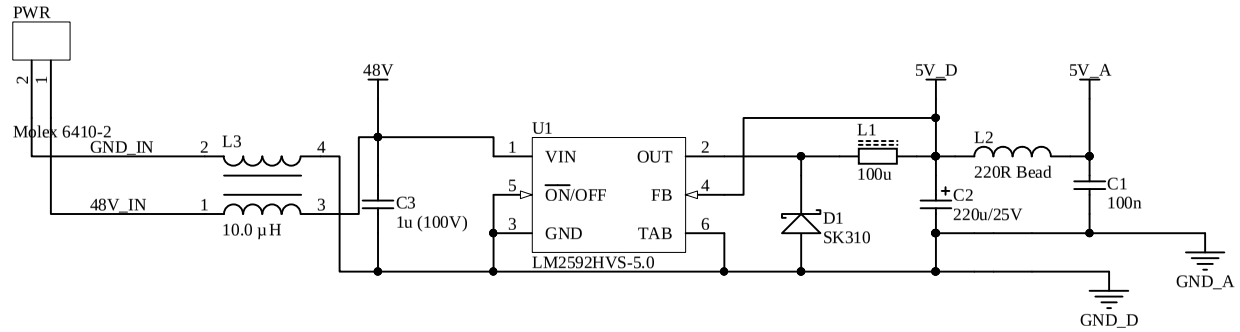


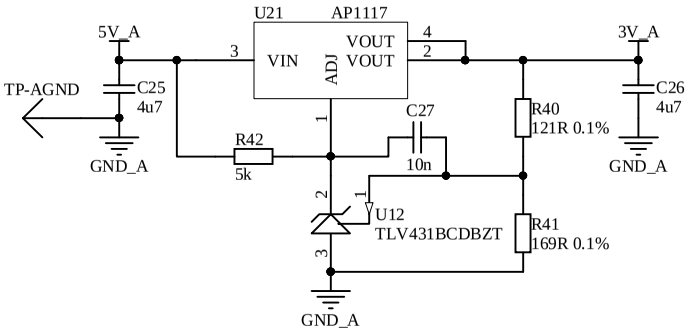
Title: Processor V_REF, ADC		
Size: A3	Number: CM408F Board	Revision: 1.0
Date: 17.08.2015	Sheet 5 of 11	
File: H:\Dropbox\..\Processor_VANA.SchDoc	Drawn By: mGoessl	

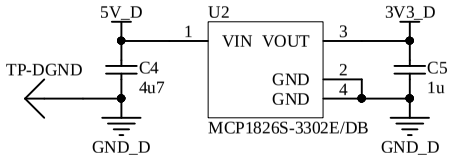


Title: Debug LED		
Size: A4	Number: CM408F Board	Revision: 1.0
Date: 17.08.2015		Sheet 6 of 11
File: H:\Dropbox\..\Debug_LED.SchDoc		Drawn By: mGoessl

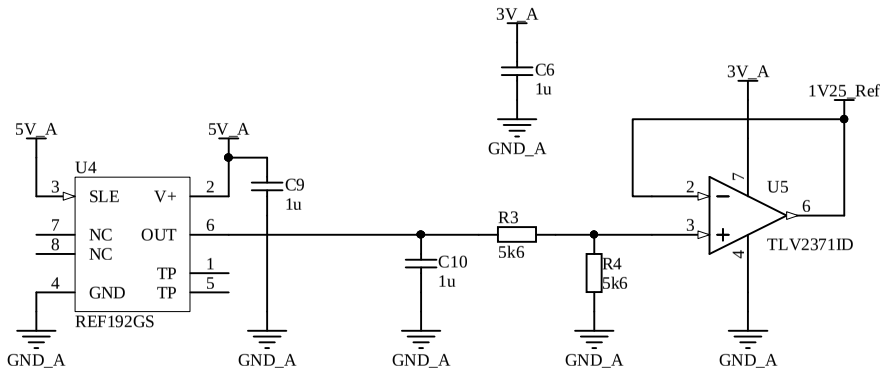

Main-Bus, CAN
Sensor-Bus, UART
FailSave-Bus
Debug-LED
Debug_LED.SchDoc
ADM3053
SN74LVC1G123
SN74LVC1G08
SN74HC21D
SN74LVC1G175DBV
LTC2850CMS8
NUP4201MR6
NUP2105
G3VM-61D1
MBRS1100T3
B4B-PH-K-S
B2B-PH-K-S
Rterm und Cterm nur an Bus-Enden
R226 nicht bestücken!
R225 nicht bestücken!
R? nur bei letztem Busteilnehmer
OK203, R221, Q202 nicht bestücken
Pins 6-7
Pins 8-9
Pins 11-12
Pins 19-20
Title
Bus
Size
A3
Number
CM408F Board
Revision
1.0
Date:
17.08.2015
Sheet 7 of 11
File:
H:\Dropbox\..\Bus.SchDoc
Drawn By: mGoessl



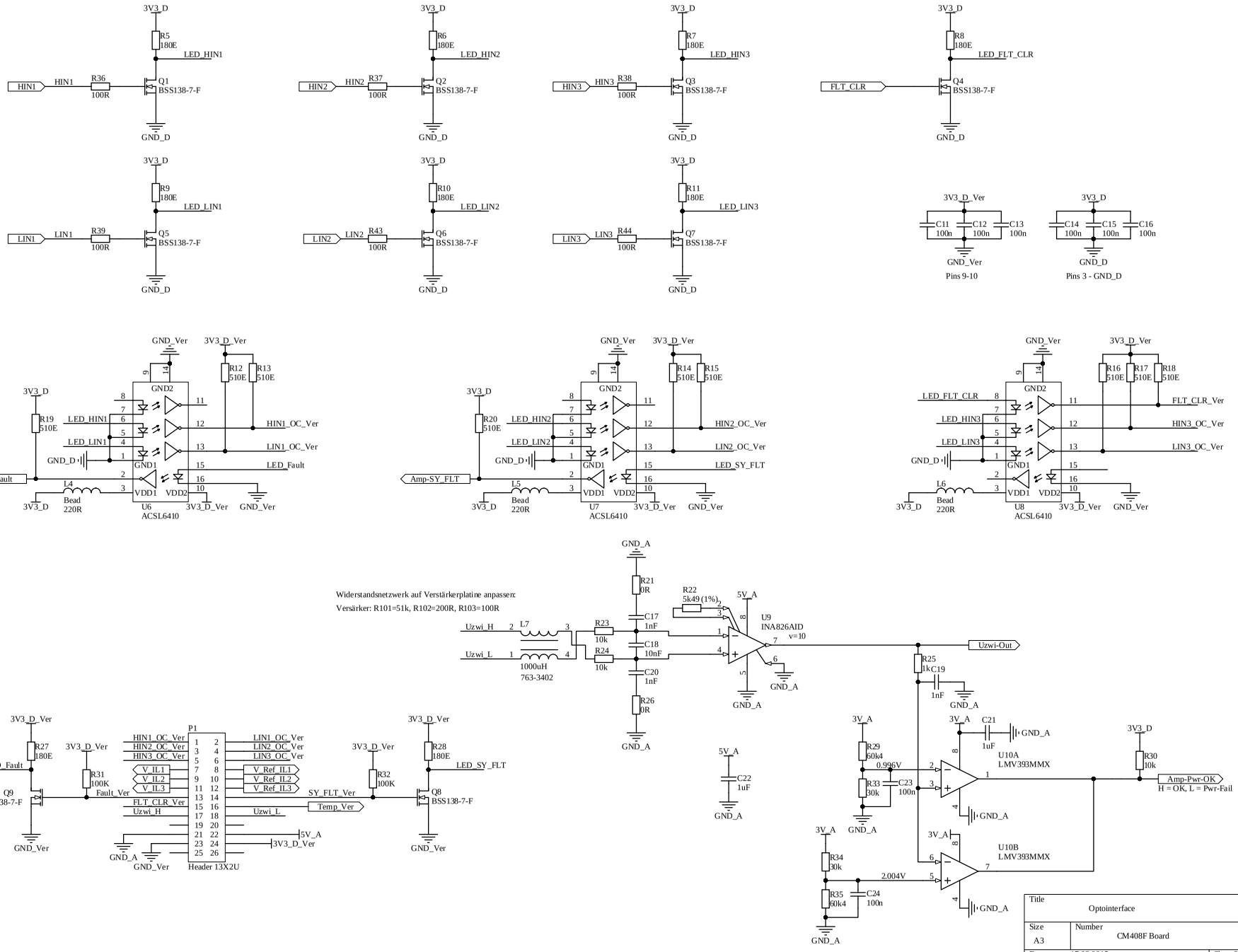


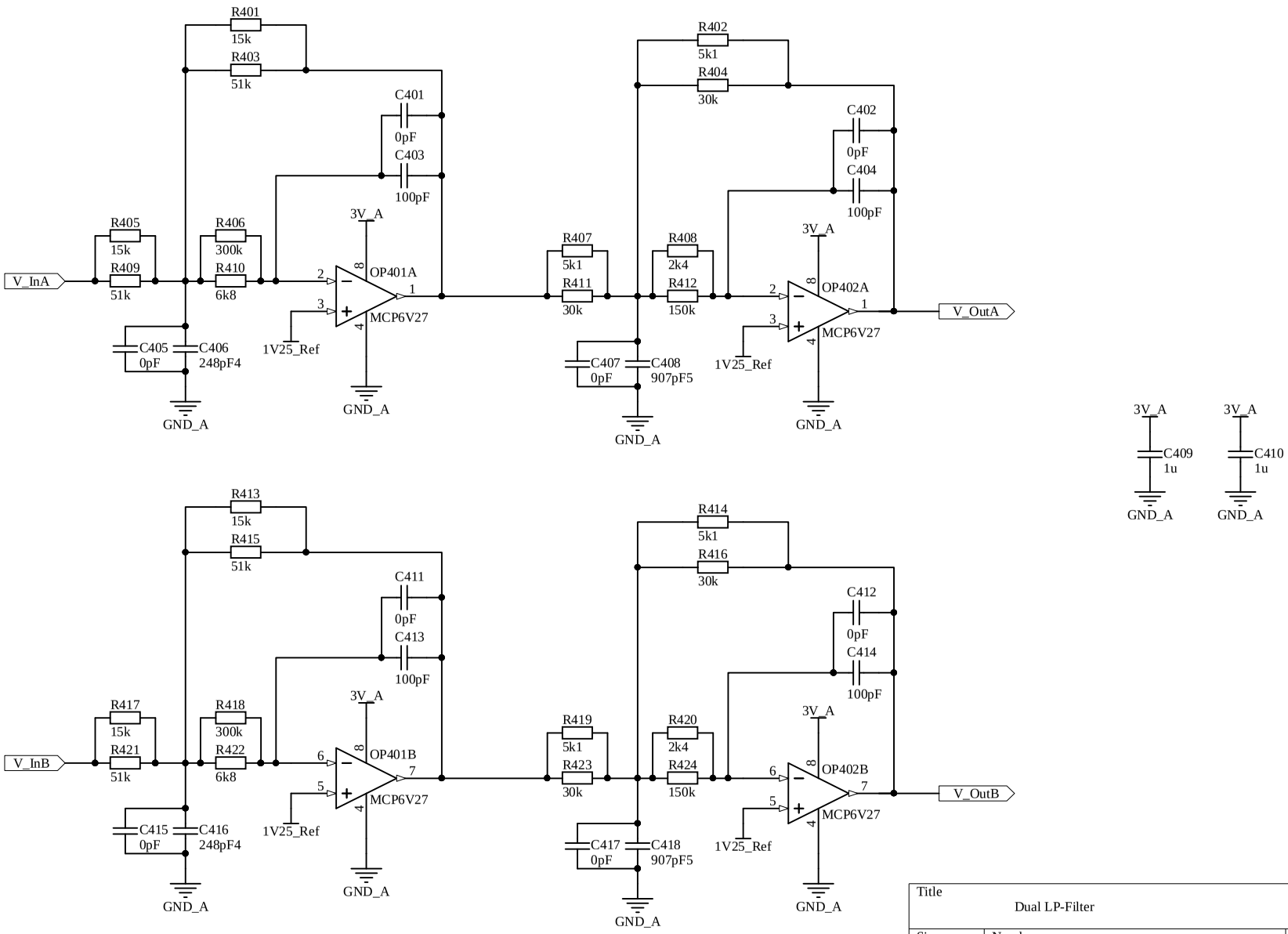


Stützkos für U404 in U_Ref-Schaltung

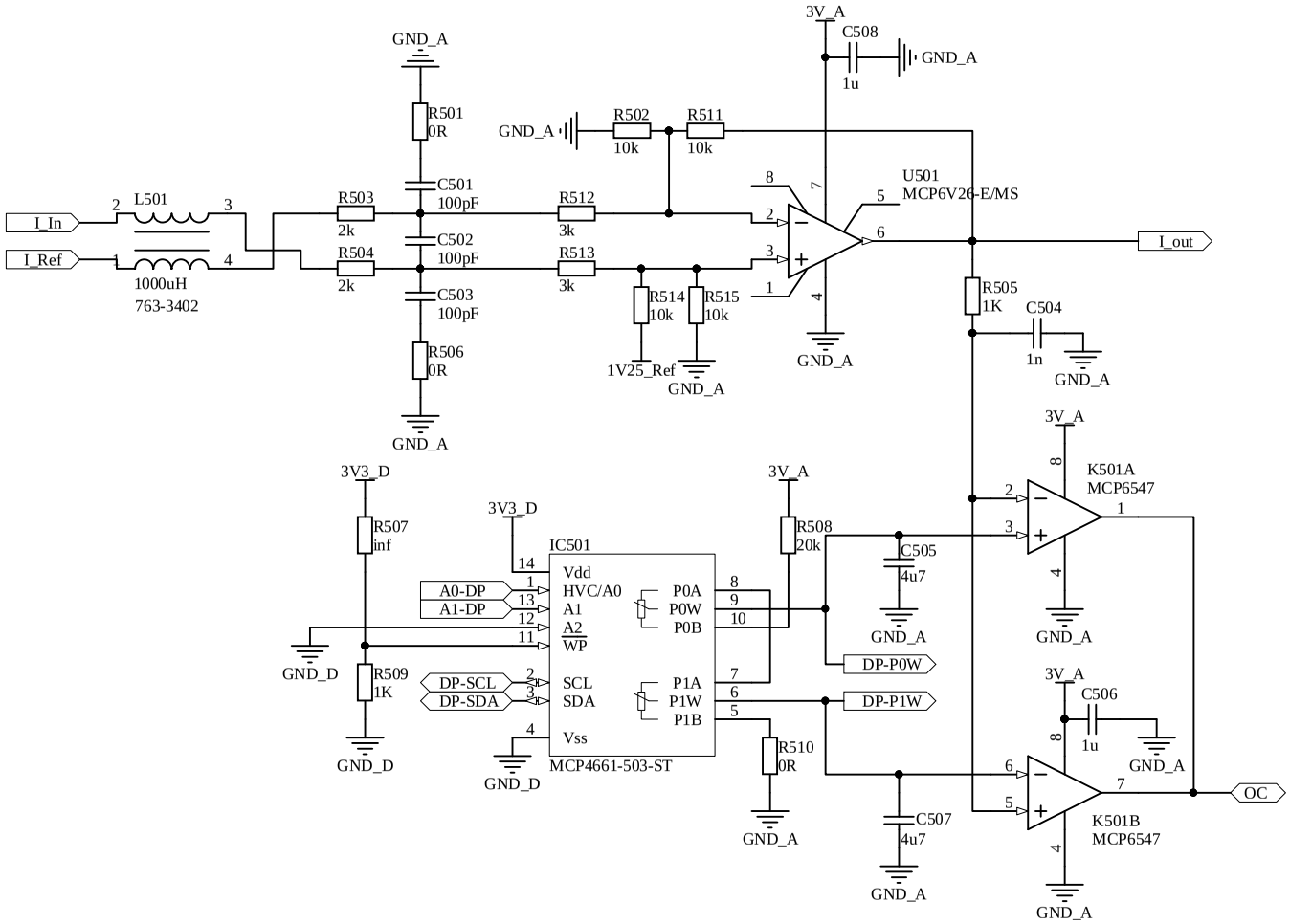


Title: Power-Supply		
Size: A4	Number: CM408F Board	Revision: 1.0
Date: 17.08.2015		Sheet 8 of 11
File: H:\Dropbox\..\Power.SchDoc		Drawn By: mGoessl


3V3_D
R5 180E
LED_HIN1
HIN1
R36 100R
Q1 BSS138-7-F
GND_D
R6 180E
LED_HIN2
HIN2
R37 100R
Q2 BSS138-7-F
R7 180E
LED_HIN3
HIN3
R38 100R
Q3 BSS138-7-F
R8 180E
LED_FLT_CLR
FLT_CLR
Q4 BSS138-7-F
R9 180E
LED_LIN1
LIN1
R39 100R
Q5 BSS138-7-F
R10 180E
LED_LIN2
LIN2
R43 100R
Q6 BSS138-7-F
R11 180E
LED_LIN3
LIN3
R44 100R
Q7 BSS138-7-F
3V3_D_Ver
C11 100n
C12 100n
C13 100n
GND_Ver
Pins 9-10
C14 100n
C15 100n
C16 100n
Pins 3 - GND_D
GND_Ver
R19 510E
R12 510E
R13 510E
GND2
GND1
VDD1
VDD2
U6 ACSL6410
Amp-Fault
L4 Bead 220R
HIN1_OC_Ver
LIN1_OC_Ver
LED_Fault
R20 510E
R14 510E
R15 510E
Amp-SY_FLT
L5 Bead 220R
HIN2_OC_Ver
LIN2_OC_Ver
LED_SY_FLT
U7 ACSL6410
R16 510E
R17 510E
R18 510E
L6 Bead 220R
FLT_CLR_Ver
HIN3_OC_Ver
LIN3_OC_Ver
U8 ACSL6410
Widerstandsnetzwerk auf Verstärkerplatine anpassen:
Versärker: R101=51k, R102=200R, R103=100R
R21 0R
R22 5k49 (1%)
C17 1nF
C18 10nF
C20 1nF
R26 0R
Uzwi_H
Uzwi_L
L7 1000uH 763-3402
R23 10k
R24 10k
U9 INA826AID v=10
Uzwi-Out
R25 1k
C19 1nF
R27 180E
R31 100K
Fault_Ver
Q9 BSS138-7-F
P1
HIN1_OC_Ver
HIN2_OC_Ver
HIN3_OC_Ver
V_IL1
V_IL2
V_IL3
FLT_CLR_Ver
Uzwi_H
V_Ref_IL1
V_Ref_IL2
V_Ref_IL3
Uzwi_L
5V_A
3V3_D_Ver
Header 13X2U
LIN1_OC_Ver
LIN2_OC_Ver
LIN3_OC_Ver
SY_FLT_Ver
Temp_Ver
R28 180E
R32 100K
LED_SY_FLT
Q8 BSS138-7-F
C22 1uF
R29 60k4
0.996V
R33 30k
C23 100n
C21 1uF
U10A LMV393MMX
R30 10k
Amp-Pwr-OK
H = OK, L = Pwr-Fail
R34 30k
2.004V
R35 60k4
C24 100n
U10B LMV393MMX
Title
Optointerface
Size
A3
Number
CM408F Board
Revision
1.0
Date:
17.08.2015
Sheet 9 of 11
File:
H:\Dropbox\..\Optointerface.SchDoc
Drawn By: mGoessl



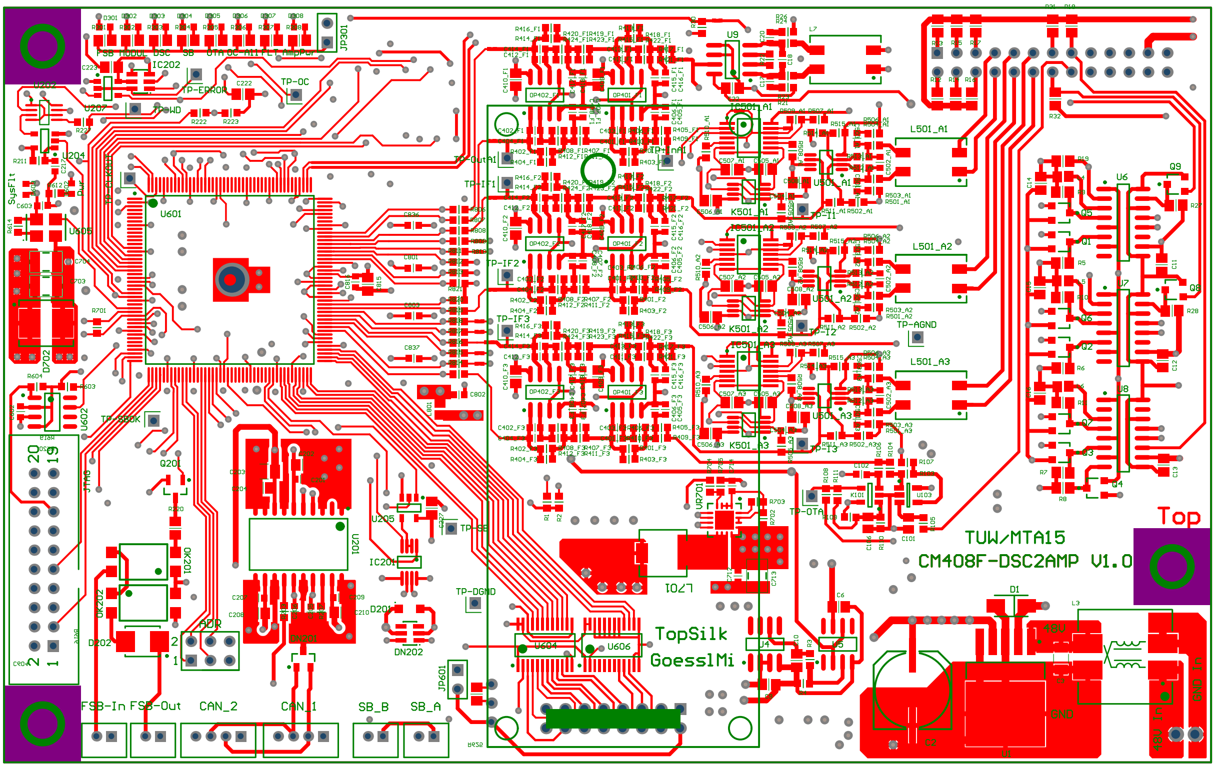
Title	Dual LP-Filter	
Size: A4	Number: CM408F Board	Revision: 1.0
Date: 17.08.2015		Sheet10of 11
File: H:\Dropbox\..\Dual-LP-Filter.SchDoc		Drawn By: mGoessl

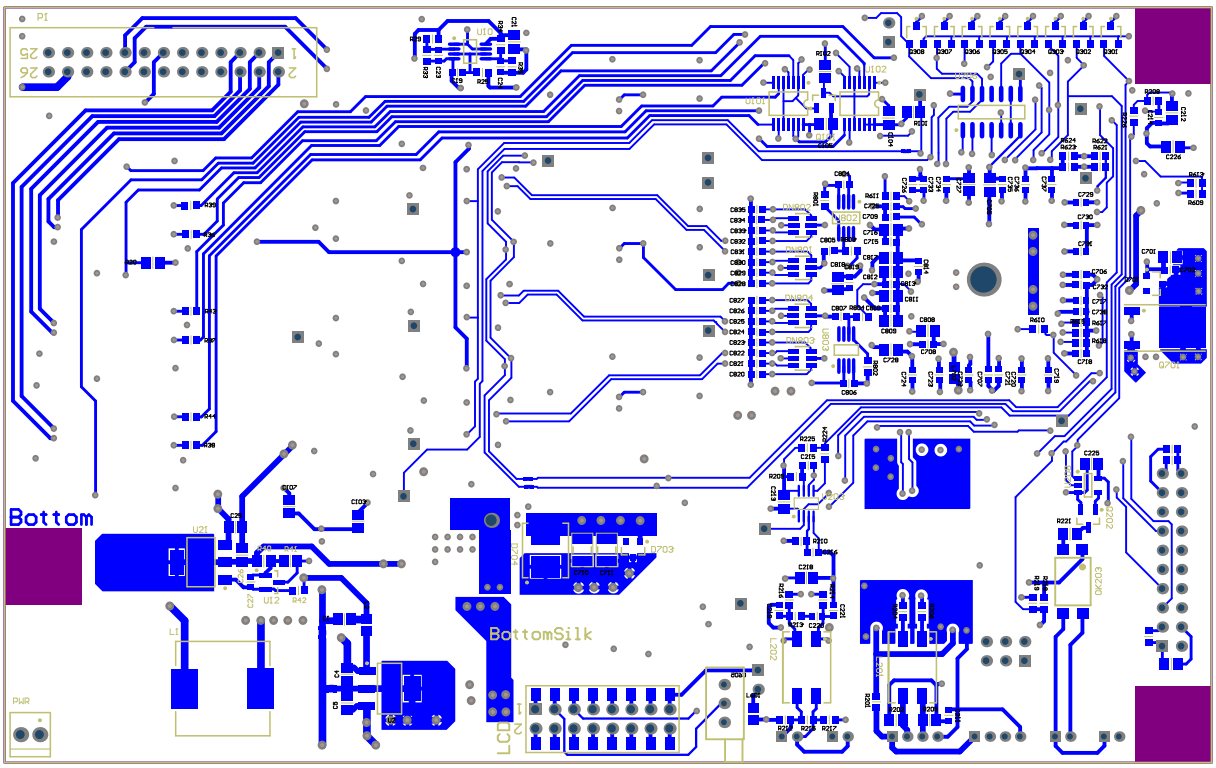


Title	Overcurrent		
Size: A4	Number: CM408F Board		Revision: 1.0
Date: 17.08.2015		Sheet11of 11	
File: H:\Dropbox\..\Overcurrent.SchDoc		Drawn By: mGoessl	

12.2 Bestückungsplan

Es folgen die Bestückungspläne der beiden PCB Seiten im Maßstab 1:1 zur Illustration des Platinenlayout-Designs der äußeren Layer.


Top
TUW/MTA15
CM408F-DSC2AMP V1.0
TopSilk
GoesslMi
FSB-In
FSB-Out
CAN_2
CAN_1
SB_B
SB_A
U601
U201
K501_A1
K501_A2
K501_A3
L501_A1
L501_A2
L501_A3
JTAG


Bottom
BottomSilk
LCD
CN1
PWR