

Entwicklung einer Entry-Level Audioeffektbox mit einem ARM Cortex-M4F DSP

FACHBERICHT: PROJEKT 5 - BURKHARDT SIMON, STUDER MISCHA
29. November 2019

Betreuung:

Prof. Dr. Markus Hufschmid

Team:

Simon Burkhardt
Mischa Studer

Studiengang:

Elektro- und Informationstechnik

Semester:

Herbstsemester 2019

Abstract

Keywords

Inhaltsverzeichnis

1	Einleitung	1
2	Analyse und Konzept	2
2.1	Produktbeschreibung	2
2.2	Lösungskonzept	2
2.2.1	Anforderungen an Microcontroller	2
2.2.2	Konzept USB Akkuladeregler IC	2
2.2.3	Kaskadierung mehrerer Boards	2
3	Hardware	3
3.1	Blockschaltbild	3
3.2	Pegeldiagramm	4
3.3	Schema	5
3.3.1	Schema Speisung	5
3.3.2	Schema DSP	6
3.3.3	Schema Codec	8
3.4	PCB	10
3.5	Kosten	10
4	C Software	11
4.1	Entwicklungsumgebung Keil uVision 5	11
4.1.1	Installieren und einrichten von Keil	11
4.2	Projektstruktur	11
4.3	Libraries	12
4.3.1	SSD1306 C Library	12
4.3.2	TLV320 C Library	13
4.3.3	CMSIS / DSP	13
4.4	Konfiguration mit STM32CubeMX	15
4.4.1	Encoder Mode mit Hardware Timer	15
4.4.2	Inter Integrated Sound (I ² S)	17
4.4.3	Inter Integrated Circuit (I2C)	17
4.4.4	Asynchron UART (Debug Interface)	18
4.4.5	Analog Input GPIO (ADC)	19
4.4.6	Direct Memory Access (DMA)	19

4.4.7	Interrupt Funktionen (NVIC)	20
4.4.8	Clock Konfiguration	21
4.5	Digitaler Datenfluss	23
5	Validierung	23
5.1	Spannungsversorgung	23
5.2	Akkumulator	23
5.2.1	ADC Messung der Batteriespannung	23
6	Status und Verbesserungen	24
7	Todo-Notes	25

1 Einleitung

In den Bereichen Amateurfunk und Hobbymusik gibt es viele Situationen in denen ein einfaches, DSP-basiertes Effektgerät zur Anwendung gebracht werden kann. So soll beispielsweise ein Notchfilter einen Störton unterdrücken, oder auf Knopfdruck ein Reverb-Effekt eingeschaltet werden können.

Das derzeit verwendete DSP Board für den Unterricht im MicroCom Labor basiert auf einem dsPIC33 mit Fixed-Point-Recheneinheit. Die neuen ARM Prozessoren bieten ab der Cortex-M4 Serie eine Floating-Point-Unit (FPU) und ermöglichen dadurch eine schnellere Verarbeitung von Signalen.

Aus diesem Grund wird die Hardware des DSP Boards überarbeitet und soll mit einem ARM Cortex-M4 Microcontroller ausgestattet werden. Der Schaltungsaufwand beschränkt sich auf die wesentlichen Funktionen. Diese beinhalten die MCU, einen Codec für die AD/DA Wandlung, die Audio-Steckverbinder und die Bedienelemente des HMI.

Im Bereich Amateurfunk und Hobbymusik besteht oft ein Bedürfnis nach einer einfachen Möglichkeit, ein Audiosignal mit einem Effekt zu verändern. So kann es sein, dass ein Amateurfunker mit einem Notch-Filter einen Störton unterdrücken möchte. Als Musiker möchte man mit einer Effektbox einen Reverbeffekt erzeugen. Effektgeräte und Filter am Markt sind oft zu einem Premiumpreis erhältlich. Dieses Projekt hat zum Ziel, eine günstige Alternative zu diesen Geräten zu bieten.

Heute bieten die DSP Funktionen in der ARM Cortex-M4 Architektur eine günstige Möglichkeit Signalverarbeitung auf Microcontrollerebene zu betreiben. Der Rahmen dieses Projektes umfasst die Entwicklung der Hard- und Firmware eines DSP Boards mit ARM Cortex-M4 Microcontroller. Das Gerät wird mit Bedienelementen wie 2 Dreh

2 Analyse und Konzept

2.1 Produktbeschreibung

2.2 Lösungskonzept

In diesem Abschnitt werden die Anforderungen and die einzelnen Teilaspekte aufgelistet und die Spezifikationen mehrerer Varianten verglichen.

2.2.1 Anforderungen an Microcontroller

Die an den Prozessor gestellten Anforderungen sind ein ARM-Cortex M4 Core mit DSP und FPU sowie Schnittstelle(n) zur Kommunikation mit dem Audio Codec. Dabei wird aufgrund der genaueren Samplingrate der Codec als Master betrieben und der DSP als Slave. Der DSP muss also keine genaue Clock zur Verfügung stellen. Auf eine Cortex-M7 Architektur wird verzichtet, weil ab diesem Punkt auch ein Single-Board Computer (vgl. Raspberry Pi) eingesetzt werden kann. Eine Tacktfrequenz von 200MHz ist wünschenswert, jedoch befinden sich die Cortex-M4 Prozessoren mit 200MHz auf dem selben Preisniveau von Cortex-M7 Microcontrollern.

2.2.2 Konzept USB Akkuladeregler IC

Ein weiches Ziel ist die Autonomie ohne externe Energieversorgung. Dazu soll ein Akkumulator genügend Energie liefern, um die Schaltung während einiger weniger Stunden (live-Konzert) zu betreiben. Der Ladestrom soll nicht grösser als die über USB-2.0 zugelassenen 2.0A betragen.

2.2.3 Kaskadierung mehrerer Boards

Mehrere Boards sollen mit Gehäuse neben einander kaskadierbar sein. Das Audio Signal wird von einem Board zum nächsten jeweils analog weitergereicht. Der Steckverbinder soll kleiner als D-Sub sein. Auf eine digitale Schnittstelle wird wegen der aufwändigen Clock-synchronisation und der Kosten für Steckverbinder (vgl. Optisch Toslink) verzichtet.

3 Hardware

Das nachfolgende Kapitel Hardware beschreibt die verbauten Komponenten. Berechnungen und wichtige Details im Schema sind in den jeweiligen Unterkapiteln beschrieben.

3.1 Blockschaltbild

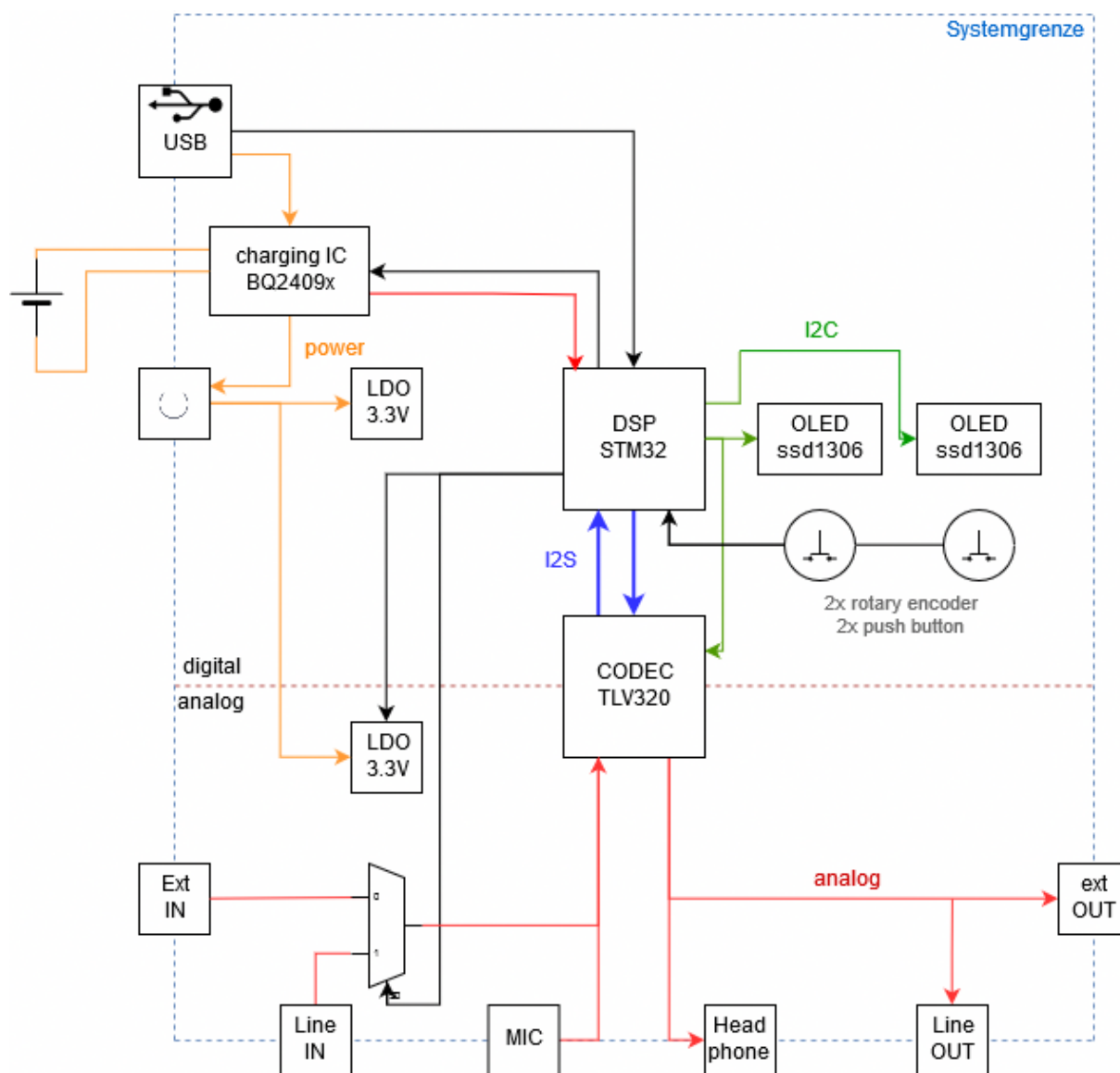


Abbildung 3.1: Blockschaltbild des DSP Boards

Die Abbildung 3.1 oben zeigt das Blockschaltbild mit den Komponenten um den DSP.

3.2 Pegeldiagramm

Nachfolgend sind in den Abbildungen 3.2 und 3.3 die Pegeldiagramme des Signalpfades dargestellt. Von Line IN können Signale mit bis zu +6 dBV ankommen. Mit einem 1:1 Spannungsteiler wird das Signal auf 0 dBV abgeschwächt.

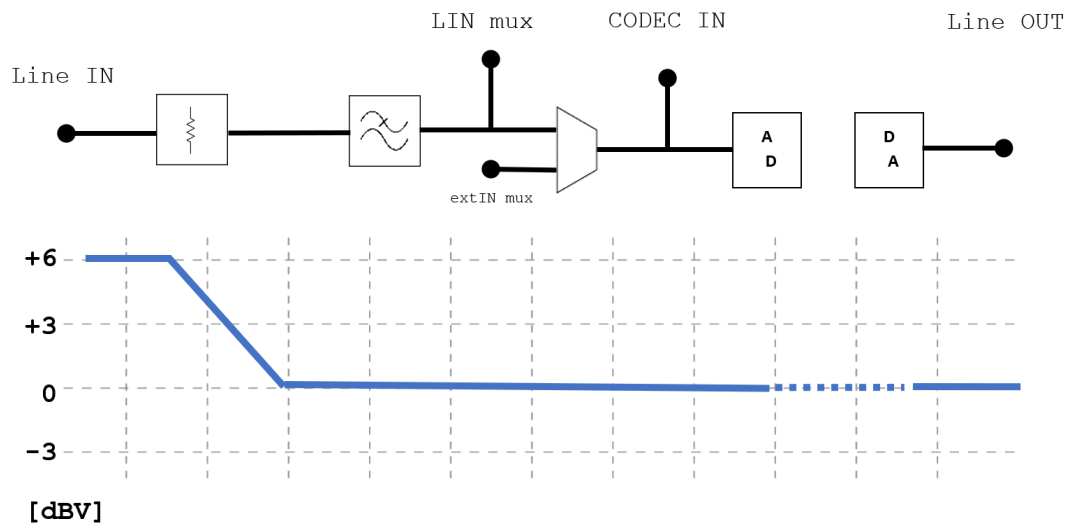


Abbildung 3.2: Pegeldiagramm des Audiopfades von Line IN nach Line OUT

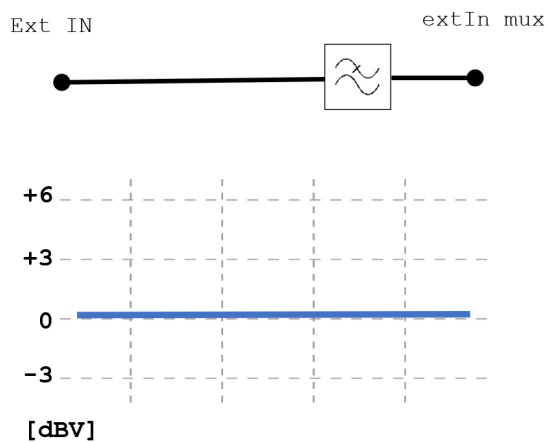


Abbildung 3.3: Pegeldiagramm des Audiopfades von Ext IN bis zum Audio Switch

SB - Ausgangspegel
+6dB bis -40.5
dB (-34.5 -6)

3.3 Schema

3.3.1 Schema Speisung

Nachfolgend wird der Speisungspfad vom USB mini-B Connector bis zu den Regulierten 3.3V beschrieben.

USB Port

Ein standard USB 2.0 Port kann einen maximalen Strom von $I_{max} = 500\text{mA}$ bereitstellen. Dieser Strom ist die obere Grenze für das DSP Board. In keinem Betriebsfall inkl. Akkuladen wird dieser maximale Strom überschritten. Zum Schutz der USB-Host Geräte vor einem Kurzschlussstrom, ist F1 (MF-MSMF110) mit einem Auslösestrom von $I_{trip} = 2.20\text{A}$ **usb-fuse**.

Battery Management (BQ2409x)

Der BQ24093 ist ein single-cell Li-Ion / Li-Po Akkulade-IC, das speziell für USB Applikationen gemacht ist. Die Beschaltung des IC1 ist gemäss Vorgaben aus dem Datenblatt **bq2409x**.

Der Akkumulator ist nicht Teil des Systems. Aus diesem Grund ist ein 2-Pin JST-XH Connector (J6) vorgesehen. Weil der Akkumulator und ein entsprechender temperaturabhängiger Widerstand nicht bekannt ist, wird der Pin für die Temperaturüberwachung mit R19 terminiert. IC1 wird mit dem Pull-Down R51 am ISET2 Pin auf LOW gezogen, was den Ladestrom auf $I_{charge} = 100\text{mA}$ beschränkt. Bei Bedarf kann der Ladestrom vom STM32 über PB14 auf HIGH und damit $I_{charge} = 500\text{mA}$ festgelegt werden.

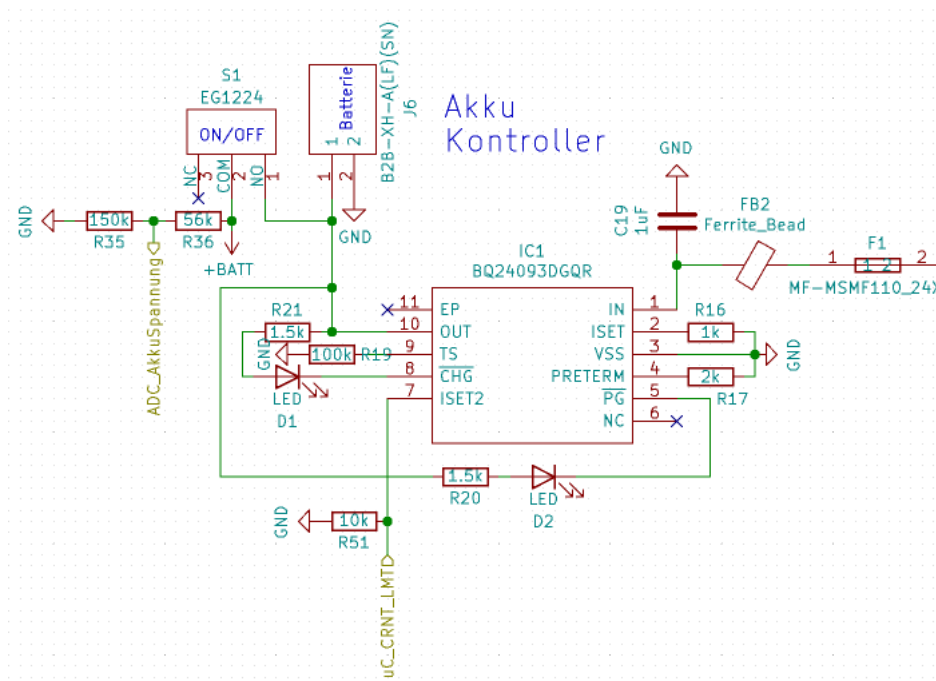


Abbildung 3.4: BQ24093 mit äusserer Beschaltung nach Datenblatt

ADC Akkumulator Spannungsmessung

Die Akkumulatorspannung mit $V_{bat} = 4.2V$ übersteigt den Eingangsspannungsbereich des STM32 Microcontrollers. Damit zur Bestimmung des Ladestandes die Spannung gemessen werden kann, wird diese mit dem Spannungsteiler R35/R36 auf ein Maximum von 3.3V heruntergebrochen. Mit Widerstandswerten aus der E12-Reihe ergibt sich folgender Spannungsteiler.

$$V_{out_{max}} = V_{bat_{max}} * \frac{R35}{R35+R36}$$

$$3.277V = 4.5V * \frac{150k\Omega}{150k\Omega+56k\Omega}$$

Somit muss die gemessene Spannung in der Software um folgenden Faktor korrigiert werden.

$$F_C = \frac{V_{bat_{max}}}{V_{out_{max}}} = \frac{3.27V}{4.5V} = 0.73$$

Energiebedarf der Schaltung

Unten aufgeführt ist eine Abschätzung des Energiebedarfs der Schaltung, die massgebend für die Wahl der Spannungsregler ist. Die Speisung ist in analog und digital aufgeteilt.

Schaltungsteil	I _{max} [mA]
STM32	40
SSD1306	30
SSD1306	30
reserve	50
textbfTotal	textbf150

Schaltungsteil	I _{max} [mA]
MAX4762	0.01
TLV320	26
reserve	30
textbfTotal	textbf56.01

Die verbauten Festspannungsregler TLC7333 (IC2, IC3) mit Low Dropout Voltage können bis zu $I_{out} = 300mA$ liefern.

Verlustleistung der Spannungsregler

Die maximale Verlustleistung an einem der Spannungsregler (IC2, IC3) tritt auf, wenn die Eingangsspannung $V_{in} = 5V$ beträgt und der maximale Strom von $I_{max} = 0.15A$ fließt. Dabei entsteht eine Verlustleistung von: $P_{LDO_{max}} = (5.0V - 3.3V) * 0.15A = 0.255W$

- soll man hier noch mit P_{max} vom LDO vergleichen?

3.3.2 Schema DSP

I2S Schnittstelle

STM32	signal	direction	signal	TLV320
PC2	MISO	←	DOUT	6
PC3	MOSI	→	DIN	4
PB12	WS	→	LRCIN	5
PB12	WS	←	LRCOUT	7
PA2	CKIN	←	CLKOUT	2

Tabelle 3.1: Signale zwischen dem DSP und dem Codec

Bootloader und Bootpins

Die STM32 Familie hat einen integrierten Firmware upgrade Bootloader. Um diesen zu aktivieren müssen die externen BOOT[1:0] Pins im richtigen Muster gesetzt werden. In diesem Projekt wird die Firmware über den USB mini-B Connector auf den STM32 überspielt. Das Application Note AN2606 **AN2606** beschreibt die Pinconfiguration für diesen Anwendungsfall. So muss kein Pull-Up Widerstand an der USB_DP Leitung angeschlossen sein um die OTG-Bedingungen zu erfüllen. Ausserdem muss eine externe Clock mit einer Frequenz zwischen 4MHz und 26MHz verfügbar sein.

Die Bootpins müssen gemäss der Application Note AN2606 **AN2606** mit dem Boot Pattern 1 gesetzt werden um den DFU Bootloader zu starten. Unten in der Tabelle sind die beiden Boot Modi zusammengefasst.

Der BOOT1 Pin ist beim STM32F412 auf PB2.

Boot Mode	BOOT1	BOOT0
Bootloader	0	1
Normal	0	0

Tabelle 3.2: Mode Auswahl über BOOT[1:0] Pins

Rotary Encoder und Buttons

Einige Timer des STM32 unterstützen einen Encoder Modus, bei dem 2 GPIO Inputs zum Zählen der Encoderpulse verwendet werden können.

Alle 4 Pushbuttons sind an interruptfähige (EXTI) GPIO Pins angeschlossen. Die STM32 Familie unterstützt externe GPIO Interrupts an allen Pins. Dabei stehen 16 Interrupt Channel zur Verfügung, von welchen die Channel Nummer jeweils auf die GPIO Port Nummer passen muss. Wie in der nachfolgenden Tabelle aufgeführt, werden gesamt 4 EXTI Channels belegt.

Signal	GPIO	EXTI
Encoder 1 Button	PC12	12
Encoder 2 Button	PB13	13
Button 1	PA0	0
Button 2	PA1	1

Tabelle 3.3: GPIO Mapping auf EXTI Interrupt Channels

3.3.3 Schema Codec

Die Wandlung des analogen Audio-Signale sowie die Rückwandlung der digitalen Signale vom Microcontroller übernimmt ein TLV320AIC23B von Texas Instruments. Dieser Codec hat eine variable Sampling-Frequenz (8kHz-96kHz), einen Kopfhörer- und einen Mikrofon-Vorverstärker. Die Register des Codecs werden über I2C programmiert (wobei auch SPI möglich wäre), dazu wird der Mode-Pin über einen Widerstand auf GND gezogen. Die gesampelten Daten wiederum werden über I2S an den Microcontroller übermittelt. Zusätzlich wird der 12.28 MHz Audio-Clock mit dem der Codec taktet von einem externen ECS-Quartz erzeugt (Abbildung 3.5).

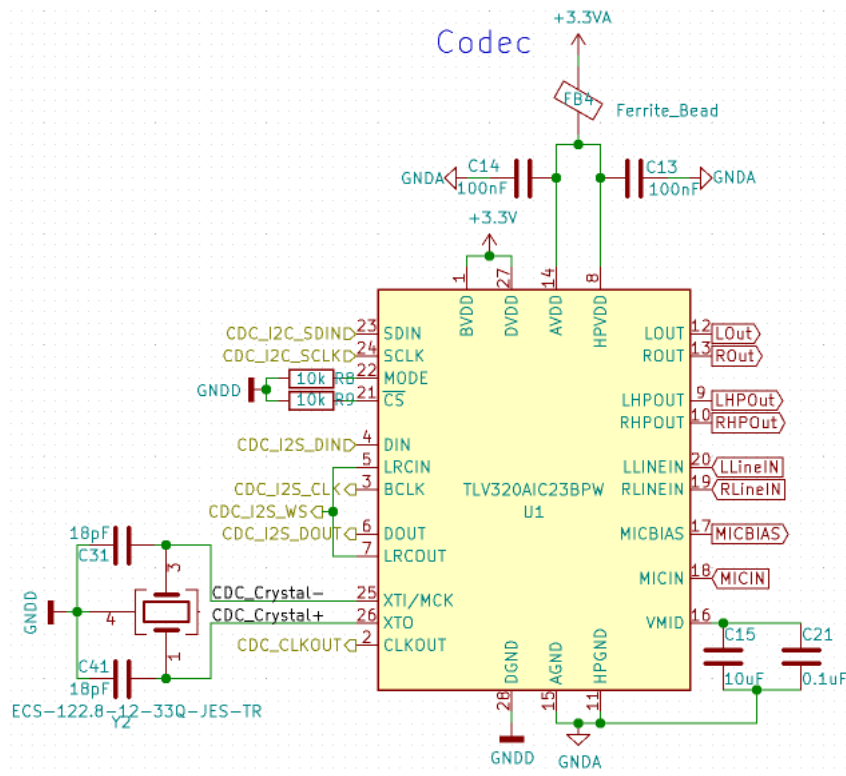


Abbildung 3.5: TLV320AIC23B Codec von Texas Instruments

Die Analog-Speisung des TLVs wird neben den 100nF Stütz-Kondensatoren zusätzlich von einem Ferrit Bead geglättet. Damit wird sichergestellt, dass keine ungewollten hochfrequente Spitzen in der Speisung das Audio-Signal verfälschen.

In den nachfolgenden Unterkapiteln wird genauer auf die In- und Outputs des DSP-Boards, beziehungsweise deren äusseren Beschaltung eingegangen.

Line Input

Das Line-Signal wird am Eingang (Abbildung 3.6) erstmal durch den Spannungsteiler mit $R23$ und $R14$ bzw. $R27$ und $R26$ halbiert. Im Datenblatt des TLV320AIC23B **tlv320** wird empfohlen die Signal-Amplitude von 2V auf 1V runterzubringen. Zudem bilden $R23/R26$ zusammen mit $C23/C26$ einen Tiefpass-Filter mit einer Grenzfrequenz von 604.7 MHz, was schonmal die größten hochfrequenten Störungen filtert. Mit $C24/C20$ wird das Signal zusätzlich galvanisch entkoppelt.

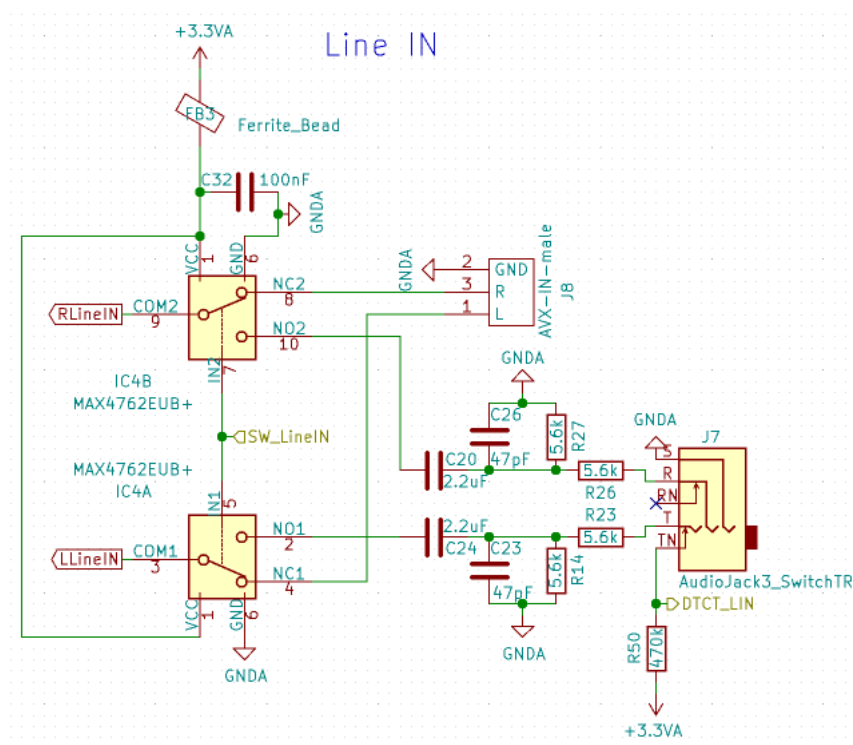


Abbildung 3.6: Line Eingangsstufe mit Audio-Switch

Parallel dazu kommt über einen AVX-Stecker alternativ ein Audio-Signal von einem nächsten DSP-Board rein. Die Auswahl des Line-In Signals erfolgt über einen MAX4762 **max4762** Audioswitch von Maxim.

Line Output

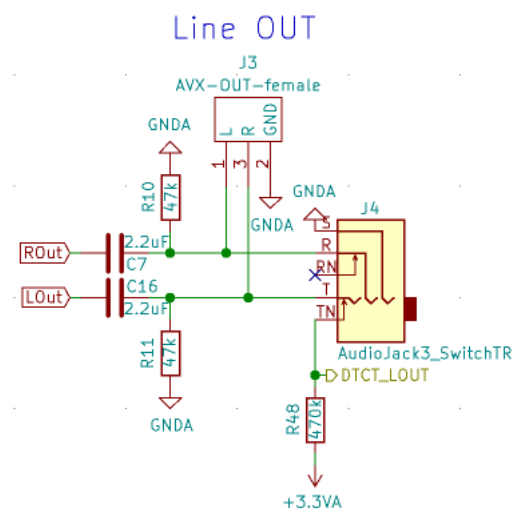


Abbildung 3.7

Headphone Output

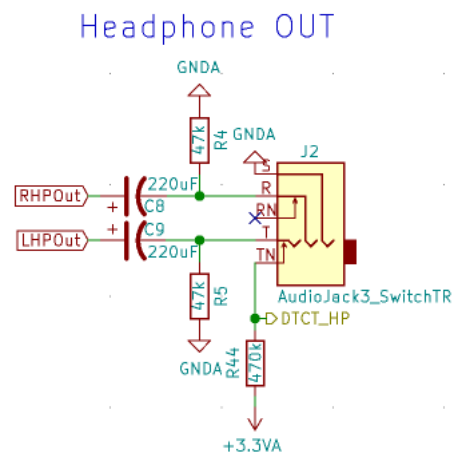


Abbildung 3.8

Mikrofon Input

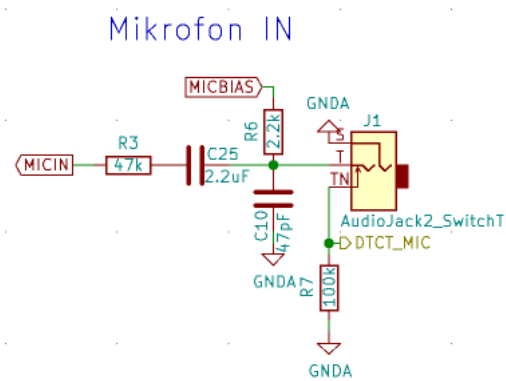


Abbildung 3.9

GND Trennung

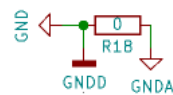


Abbildung 3.10

3.4 PCB

3.5 Kosten

4 C Software

Das Kapitel 4 beschreibt die Entwicklungsumgebung, die Strukturierung des Softwareprojektes sowie die einzelnen Komponenten der Software.

4.1 Entwicklungsumgebung Keil uVision 5

Die Software ist in C mit der Keil uVision 5 IDE geschrieben. Nachfolgend ist beschrieben, welche Schritte unternommen werden müssen, um das Projekt erfolgreich zu kompilieren.

SB - Erklären,
wo, was zu
finden ist mit
Verweis auf
Unterkapitel

4.1.1 Installieren und einrichten von Keil

Dieser Abschnitt beschreibt die Installation von Keil uVision 5 mit den dazugehörigen Packages auf Windows.

4.2 Projektstruktur

Die Dateistruktur des Softwareprojektes in Keil uVision 5 ist unten beschrieben. Es sind nur die Dateien erwähnt, die für die Entwicklung der Software massgeblich sind.

main.c

In der **main.c** Datei befindet sich der Hauptteil des Softwareprojektes. Libraries werden inkludiert, globale sowie lokale Variablen deklariert und die Initialisierung des STM32 und aller Peripherie aufgerufen. Innerhalb der **while(1)** Schleife ist eine State Machine realisierbar, die den Programmablauf des User Interfaces bestimmt. Viele Variablen sind global und **volatile** deklariert und werden laufend von den Interrupt Handlern (IRQ) in **stm32f4xx_it.c** verändert.

stm32f4xx_it.c

In dieser Datei sind die Interrupt Request Handler (IRQ) Funktionen ausprogrammiert. Die definierten Funktionen werden bei verschiedenen Interrupt Events wie externen GPIO Interrupts (EXTI) oder DMA Transmission Completion Interrupt aufgerufen.

dsp_board_bsp.c

Die Datei **dsp_board_bsp.c** stellt ein Board Support Package für das P5 DSP Board dar. In dieser Datei sind Funktionen ausgelagert, die spezifisch für die Hardware gemacht sind. Die Interrupts werden hier abgefangen und über **extern volatile** Variablen an die State Machine im **main.c** weitergegeben. Ausserdem sind gewisse, nicht spezifisch auf die Hardware ausgelegte Helferfunktionen (z.B. Sinusgenerator) in dieser Datei ausgelagert.

ssd1306.c

Die Dateien **ssd1306.c**, **ssd1306_fonts.c** und **ssd1306_tests.c** beinhalten die Funktionen zur Ansteuerung der SSD1306 OLED Displays und bilden die Library.

tlv320aic.c

Die **tlv320aic.c** Datei bildet die Library für den TLV320 Audio Codec und beinhaltet Funktionen zur Lautstärkeregelung und Initialisierung des Codecs.

dsp_processing.c

In der DSP Processing Datei werden die empfangenen Audiodaten an verschiedene DSP-Funktionen wie FIR-Filter verteilt und der Output-Buffer für den DMA Controller befüllt.

fir.c

In dieser Datei ist ein FIR Filter aus der CMSIS/DSP Library implementiert.

4.3 Libraries

Das Kapitel 4.3 beschreibt die im Projekt eingebundenen C Libraries. Dabei werden die STM32CubeMX HAL Libraries nur kurz eingeführt und auf weitere Quellen verwiesen. Der Fokus der folgenden Unterkapitel liegt auf den Peripherielibraries speziell für das DSP Board. Dazu zählen die ssd1306 OLED Displays, die boardspezifischen Funktionen in Form eines Board Support Packages (BSP) als auch die Library für den TLV320 Audio Codec.

4.3.1 SSD1306 C Library

Zur Ansteuerung der OLED Displays wird die Library stm32-ssd1306 von Aleksander Alekseev ?? verwendet.

Spezifikationen

Beschreibung	Wert
Lizenz	MIT
RAM Bedarf	1 kiB pro Display
Textunterstützung	ja
Schriftarten	3 font sizes
Grafikunterstützung	nein

Die Library funktioniert so, dass ein Pixelbuffer pro Display im RAM erstellt wird. Der Buffer wird beim Aufruf der Funktion `ssd1306_UpdateScreen()` über den I2C Bus auf das Display geschrieben. Dadurch entsteht ein RAM Bedarf von:

$$W * H / 8 = 128 * 64 / 8 = 1024 \text{ Bytes}$$

Änderungen an der Library

Die Library unterstützt nur ein Display an einem I2C oder SPI Bus. Da bei diesem Projekt zwei Displays an unterschiedlichen Peripherischnittstellen sitzen, ist die Library für diese Anwendung angepasst. Jeder Funktion muss nun ein Pointer auf einen Display Struct mitgegeben werden. Im folgenden Listing ist dargestellt, wie die Library in C verwendet wird.

```

1  /* USER CODE BEGIN Includes */
2  #include "ssd1306.h"
3  /* USER CODE END Includes */

1  /* USER CODE BEGIN PV */
2  SSD1306_t holed1;    // Display Struct
3  /* USER CODE END PV */

```



```

1  /* USER CODE BEGIN 2 */
2  holed1.hi2cx = &hi2c1;    // set peripheral interface of Struct to I2C
3
4  ssd1306_Init(&holed1);
5  ssd1306_Fill(&holed1, Black);    // all pixels black
6  ssd1306_SetCursor(&holed1, 2, 0);    // x = 2px (from left) / y = 0px (from top)
7  ssd1306_WriteString(&holed1, "FHNW", Font_11x18, White); // medium font
8  ssd1306_UpdateScreen(&holed1);    // write Buffer to OLED Display

```

Die Library hat folgende Einschränkung: Alle Displays müssen entweder über I2C oder SPI Peripheriebusse angeschlossen sein. Ein Mischen von SPI und I2C ist nicht möglich. Ausserdem sind die Funktionen für SPI nicht implementiert.

Copyright Notice

Copyright (c) 2018 Aleksander Alekseev

4.3.2 TLV320 C Library

Zur Konfiguration des Codecs über die I2C Schnittstelle wird eine Library verwendet. Dazu kommt eine auf STM32 angepasste Version der Library von Simon Gerber und Belinda Kneubühler von August 2016 zum Einsatz.

Änderungen an der Library

```

1  /* USER CODE BEGIN 2 */
2
3  // @TODO ???

```

4.3.3 CMSIS / DSP

Der Hersteller der ARM Architektur stellt ebenfalls eine leistungsfähige DSP Library für ARM Cortex CPUs bereit. Diese CMSIS/DSP Library ist bereits als Package über Keil verfügbar. Anschliessend muss im uVision unter **Project > Manage Run-Time Environment** die DSP Library angewählt werden **enable-cmsis-dsp-lib**. Da die Library schon vorkompiliert ist, kann diese entweder als Source oder als Vorkompiliert eingebunden werden.

Die Abbildung 4.1 zeigt die Targetoptionen mit den benötigten Werten.

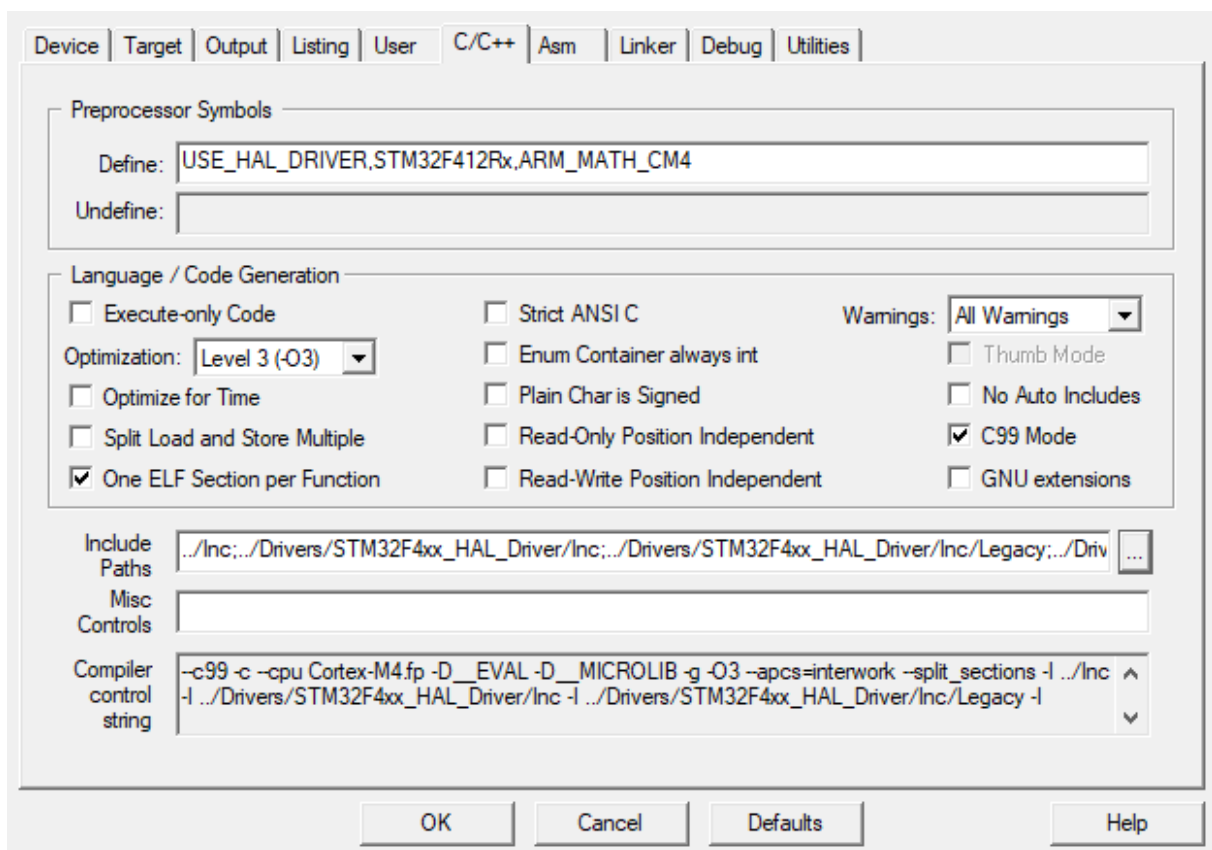


Abbildung 4.1: Options for Target 'P5 DSP Board'

Des weiteren muss in den Projektoptionen die DSP Library zu den Include Paths hinzugefügt werden.

Unter **Project > Options for Target 'DSP Board' > C/C++ > Include Paths** muss der String `../Drivers/CMSIS/DSP/Include` hinzugefügt werden.

Weiterhin muss der CMSIS/DSP Mitgeteilt werden, welche CPU Architektur verwendet wird. Im Falle des STM32F4xx entspricht dies dem Wert `ARM_MATH_CM4`, der als Preprocessor Symbol im selben Konfigurationsfenster hinzugefügt werden muss. Unter **Project > Options for Target 'DSP Board' > C/C++ > Preprocessor Symbols > Define:** muss der oben genannte String angefügt werden.

SB - Muss erklärt werden, wie dieses Package heruntergeladen wird? evtl. bei installation von Keil

4.4 Konfiguration mit STM32CubeMX

In den nachfolgenden Unterkapiteln sind die Konfigurationen in der STM32CubeMX Software abgebildet und beschrieben. Zudem sind Code Beispiele aufgeführt, die zeigen, wie die konfigurierte Peripherie im Projekt verwendet wird.

In der Abbildung 4.2 ist das Pinout des STM32F412 ersichtlich.

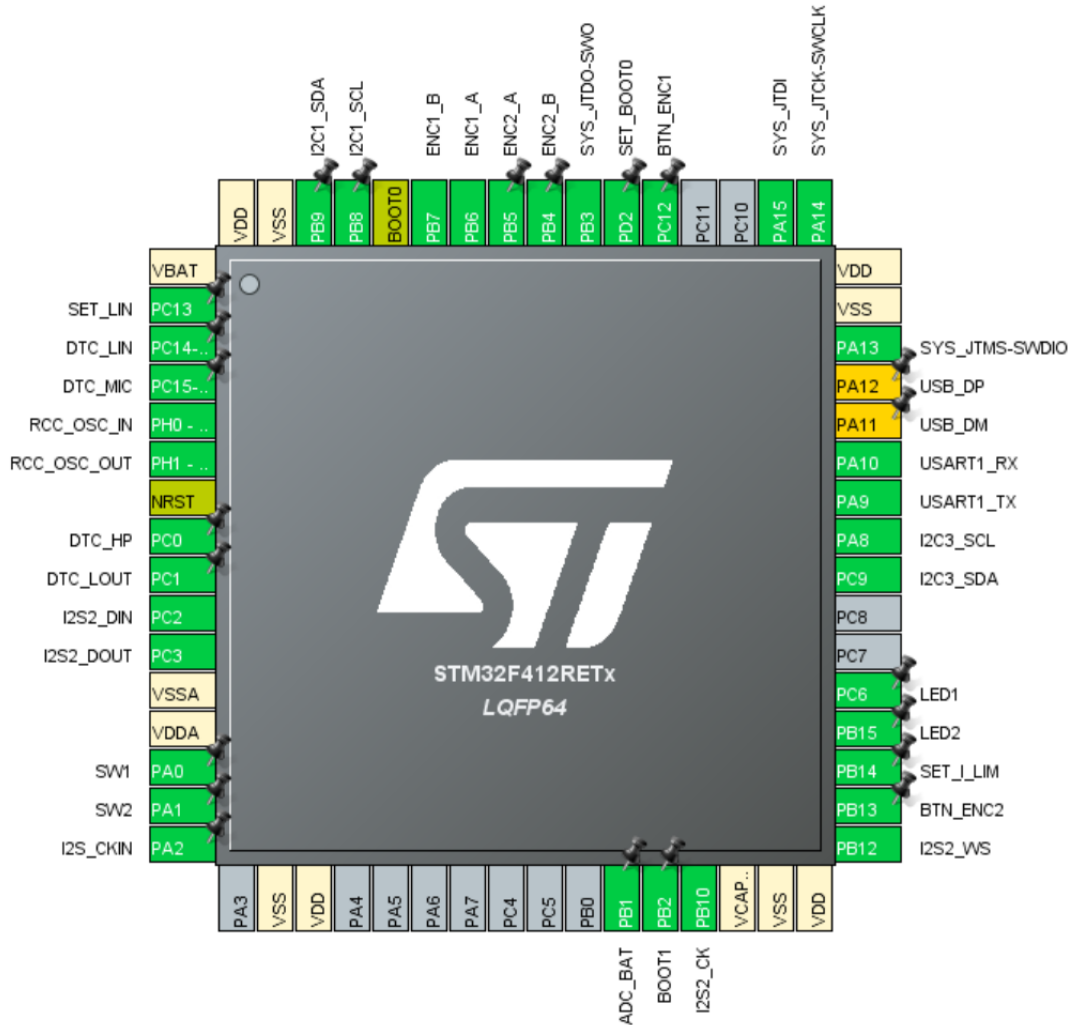


Abbildung 4.2: Pinout des STM32F412 in STM32CubeMX

4.4.1 Encoder Mode mit Hardware Timer

Einige integrierte Timer im STM32 unterstützen einen Encoder Modus, bei dem 2 vorgegebene GPIO Pins den Zählstand des Timers verändern können. In der Konfiguration wird die Zählrichtung mit Counter Mode auf Up gesetzt. Der maximale Zählerwert (Periode) ist der Maximalwert eines uint16_t Datentyps $P_{max} = 2^{16} - 1 = 65'535$. In Abbildung 4.3 ist die Konfiguration für den Timer 3 dargestellt. Der Timer 4 erhält die selbe Konfiguration.

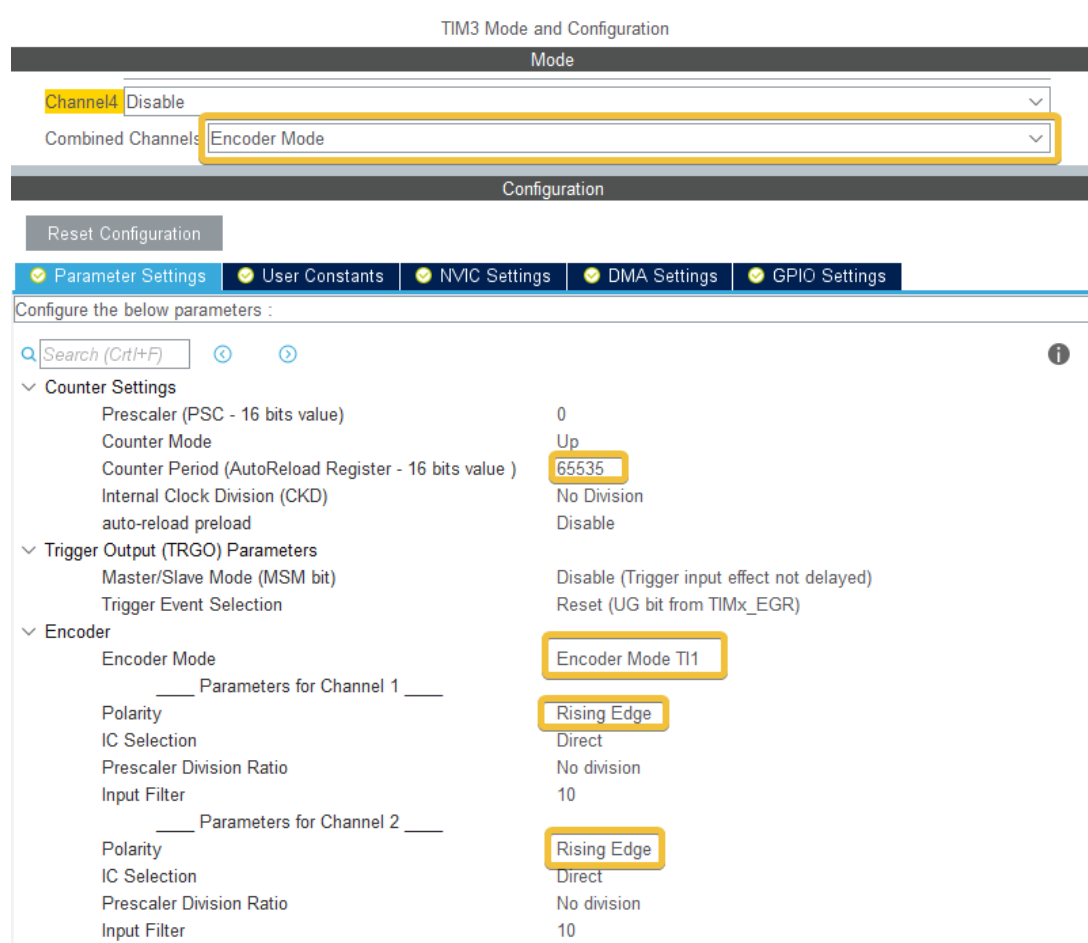


Abbildung 4.3: Konfiguration des Timer 3 im Encoder Modus

Parameter

Die nachfolgende Tabelle listet einige wichtige Parameter und deren Einfluss.

Setting	Werte	Erklärung
Counter Mode	Up Down	Zählrichtung in Abhängigkeit der Drehrichtung
Counter Period		maximaler Zählerwert (z.b. uint16_t)
Encoder Mode	T1 T2	Triggerfokus auf CH1 oder CH2 oder beides. Wenn beide aktiviert sind, zählt der Timer doppelt.

Anwendung im Code

Das untenstehende Listing stellt dar, wie der Timer im `main.c` gestartet werden muss. Eine Initialisierung alleine reicht nicht aus.

```

1 /* USER CODE BEGIN 2 */
2 // start Encoder mode on one channel
3 HAL_TIM_Encoder_Start(&htim2, TIM_CHANNEL_1);
4 /* USER CODE END 2 */

```

Mit dem unten gezeigten Befehl lässt sich der aktuelle Zählstand des Timers resp. Encoders auslesen.

```

1  /* USER CODE BEGIN x */
2  int new_encoder_val = TIM2->CNT;    // read encoder count anywhere in the code
3  /* USER CODE END x */

```

4.4.2 Inter Integrated Sound (I²S)

Der STM32F412 verfügt über mehrere integrierte I²S Interfaces. Für die Kommunikation mit dem TLV320 Codec wird das I²S2 Interface benutzt. Nachfolgend wird die Konfiguration mit der STM32CubeMX beschrieben. Die Abbildung 4.4 zeigt die wichtigsten Parameter für den Datenfluss.

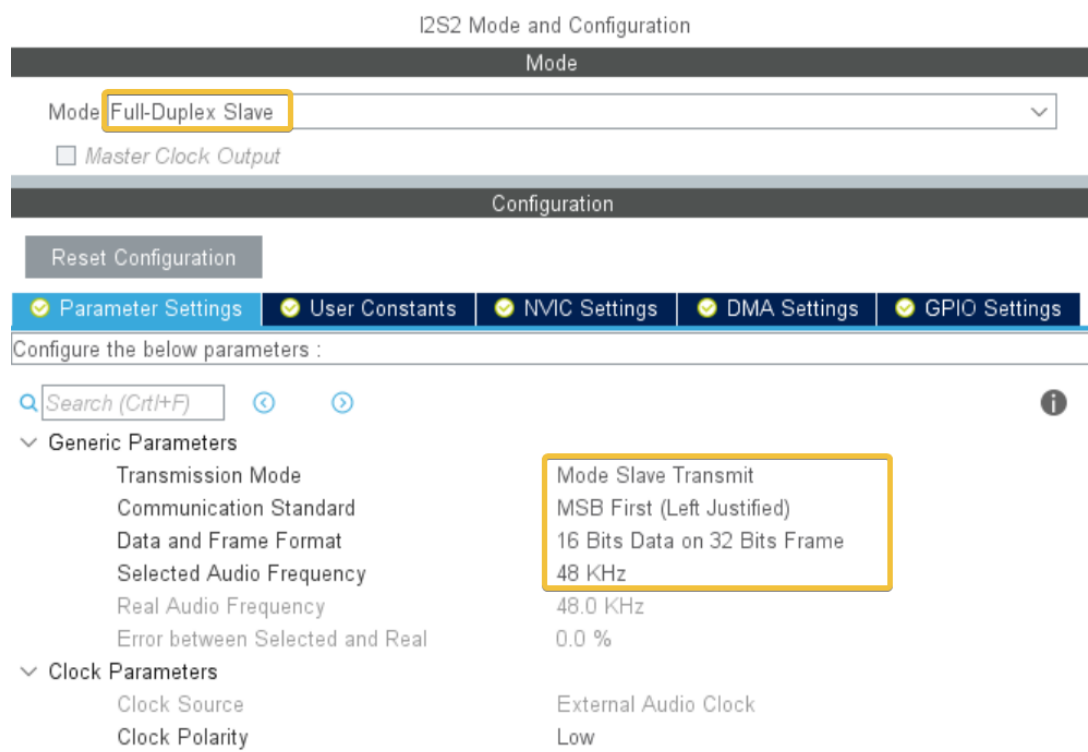


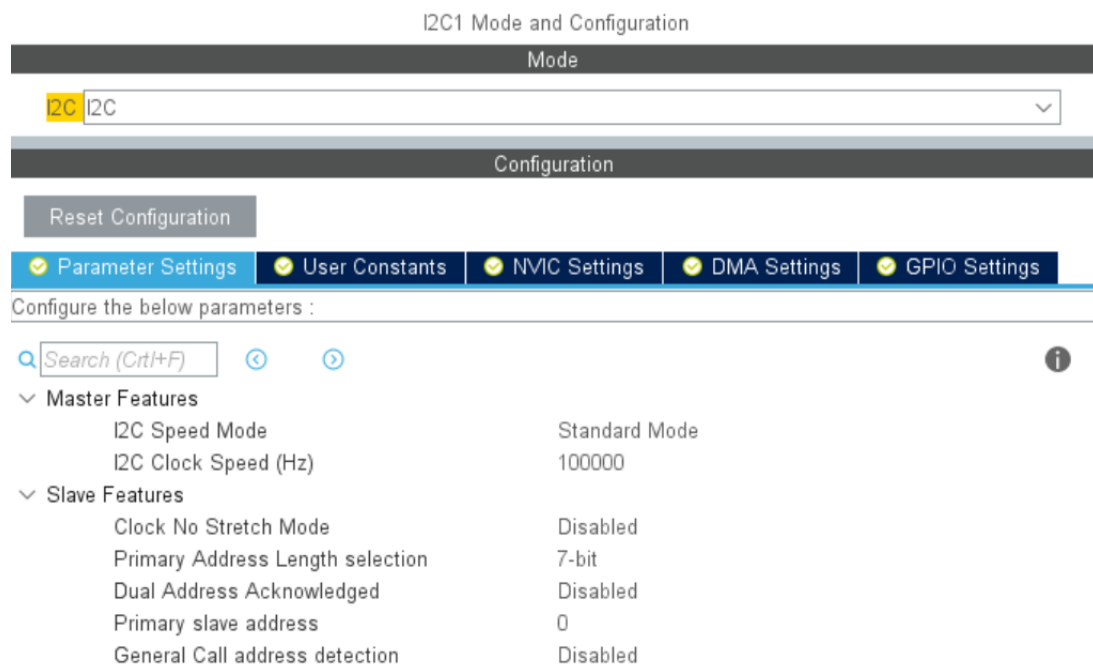
Abbildung 4.4: Parameter Einstellungen der I²S2 Schnittstelle

Da der TLV320 Codec im Master Modus betrieben wird, ist der STM32 im Mode **Slave Transmit**. Auch die Justification und die Wortbreite muss mit dem TLV320 übereinstimmen. Hier wird **MSB First** mit 16 Bits auf einem 32 Bit Frame eingestellt. Die Samplingrate beträgt $f_s = 48\text{kHz}$.

Für die I²S Schnittstelle wird die automatische Datenübertragung mittels DMA verwendet. Die Konfiguration des DMA Controllers ist in Abschnitt 4.4.6 beschrieben.

4.4.3 Inter Integrated Circuit (I²C)

Über insgesamt zwei I²C Busse (I²C1 / I²C3) werden die beiden SSD1306 OLED Displays und der TLV320 angesprochen. Beide Busse werden nach den Standardeinstellungen gemäß Abbildung 4.5 betrieben. Die Clock Frequenz beträgt 100kHz, die Adressbreite 7 Bit. Es werden keine Interrupts oder DMA verwendet.

Abbildung 4.5: Parameter Einstellungen der I²C1 und I²C3 Schnittstelle

4.4.4 Asynchron UART (Debug Interface)

Auf dem DSP Board sind die beiden Pins PA9 (Rx) und PA10 (Tx) auf Testpoints geführt. Über die UART Schnittstelle USART1 kann ein Serieller Adapter angeschlossen werden. Die Abbildung 4.6 zeigt die Standardeinstellungen mit einer Baudrate von 115200Bd. Interrupts oder DMA werden nicht verwendet.

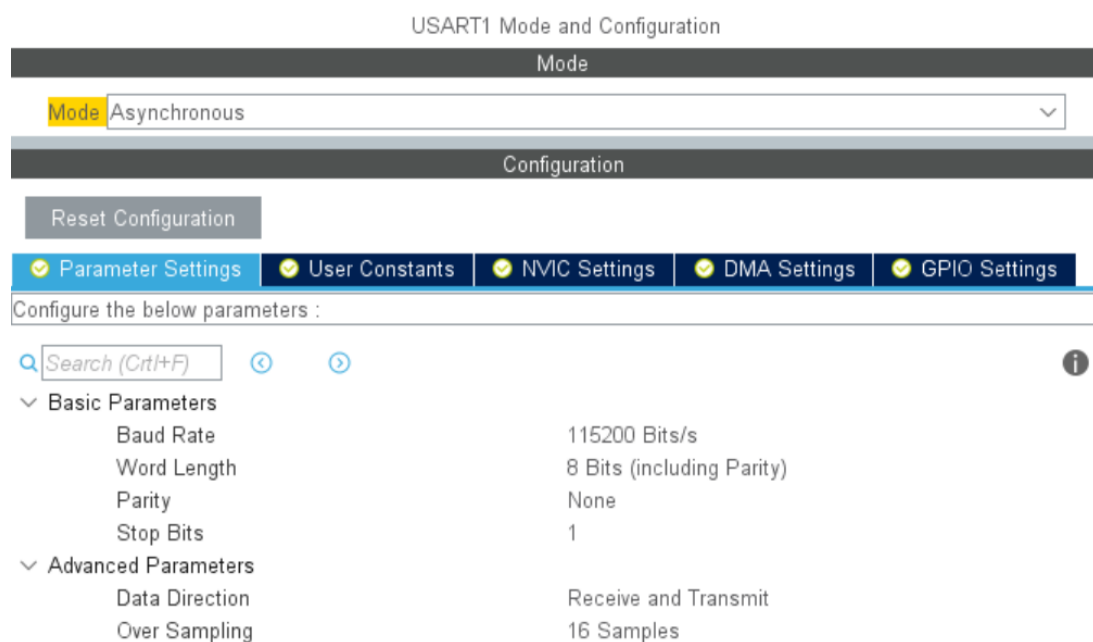


Abbildung 4.6: Parameter Einstellungen der USART1 Schnittstelle

4.4.5 Analog Input GPIO (ADC)

Zur Spannungsmessung am Akkumulator ist der GPIO Pin PB1 als Analog Input konfiguriert. Dieser wird mit dem internen Analog zu Digitalwandler ADC1 auf dem Channel 9 ausgewertet. Die Einstellungen hierfür sind in Abbildung 4.7 ersichtlich und entsprechen den Standardeinstellungen. Die Auflösung beträgt 12 Bit.

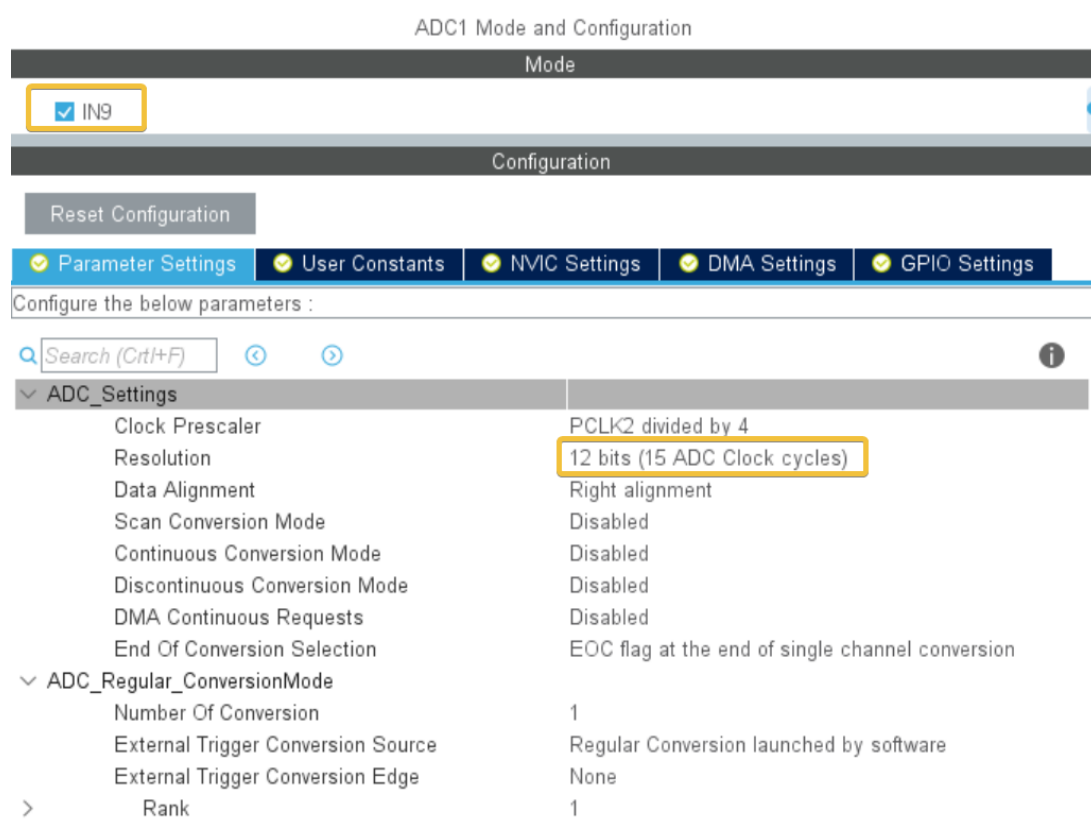


Abbildung 4.7: Parameter Einstellungen ADC Wandlers 1

4.4.6 Direct Memory Access (DMA)

Die I²S Schnittstellen unterstützen den automatischen Datentransfer über den DMA Controller. In der Abbildung 4.8 ist die Konfiguration für den Peripheral to Memory Kanal ersichtlich. Der Memory to Peripheral Kanal wird mit den selben Einstellungen konfiguriert. **Half Word** bedeutet eine Datenbreite von 16 Bit, was der bei der I²S2 Schnittstelle (Abschnitt 4.4.2) eingestellten Datenwortbreite entspricht.

DMA Request	Stream	Direction	Priority
I2S2_EXT_RX	I2S2_EXT_RX	Peripheral To Memory	High
SPI2_TX	DMA1 Stream 4	Memory To Peripheral	High

Add Delete

DMA Request Settings

Mode: Circular
 Increment Address: ☐
 Peripheral: ☐ Memory: ☒

Use Fifo: ☒ Threshold: Full
 Data Width: Half Word Burst Size: Single

Abbildung 4.8: DMA Einstellungen der I²S2 Schnittstelle

4.4.7 Interrupt Funktionen (NVIC)

Neben den Systeminterrupts sind folgende Hardwareinterrupts aktiviert. Für die Abarbeitung des I²S2 DMA Datastreams sind die Interrupts DMA1 Stream3 (Peripheral To Memory) sowie DMA1 Stream4 (Memory To Peripheral) aktiviert. Zudem sind folgende GPIO Interrupts (EXTI) für die Buttons und Encoder Buttons aktiviert.

Port	EXTI	Signal
PA0	EXTI0	User Button SW1
PA1	EXTI1	User Button SW2
PC12	EXTI12	Encoder 1 Button
PB13	EXTI13	Encoder 2 Button

☒ NVIC
 ☒ Code generation

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
Non maskable interrupt	<input checked="" type="checkbox"/>	0	0
Hard fault interrupt	<input checked="" type="checkbox"/>	0	0
Memory management fault	<input checked="" type="checkbox"/>	0	0
Pre-fetch fault, memory access fault	<input checked="" type="checkbox"/>	0	0
Undefined instruction or illegal state	<input checked="" type="checkbox"/>	0	0
System service call via SWI instruction	<input checked="" type="checkbox"/>	0	0
Debug monitor	<input checked="" type="checkbox"/>	0	0
Pendable request for system service	<input checked="" type="checkbox"/>	0	0
Time base: System tick timer	<input checked="" type="checkbox"/>	0	0
PVD interrupt through EXTI line 16	<input type="checkbox"/>	0	0
Flash global interrupt	<input type="checkbox"/>	0	0
RCC global interrupt	<input type="checkbox"/>	0	0
EXTI line0 interrupt	<input checked="" type="checkbox"/>	0	0
EXTI line1 interrupt	<input checked="" type="checkbox"/>	0	0
DMA1 stream3 global interrupt	<input checked="" type="checkbox"/>	0	0
DMA1 stream4 global interrupt	<input checked="" type="checkbox"/>	0	0
ADC1 global interrupt	<input type="checkbox"/>	0	0
TIM3 global interrupt	<input type="checkbox"/>	0	0
TIM4 global interrupt	<input type="checkbox"/>	0	0
I2C1 event interrupt	<input type="checkbox"/>	0	0
I2C1 error interrupt	<input type="checkbox"/>	0	0
SPI2 global interrupt	<input type="checkbox"/>	0	0
USART1 global interrupt	<input type="checkbox"/>	0	0
EXTI line[15:10] interrupts	<input checked="" type="checkbox"/>	0	0
I2C3 event interrupt	<input type="checkbox"/>	0	0
I2C3 error interrupt	<input type="checkbox"/>	0	0
RNG global interrupt	<input checked="" type="checkbox"/>	0	0
FPU global interrupt	<input type="checkbox"/>	0	0

☒ Enabled
 Preemption Priority
Sub Priority

Abbildung 4.9: Alle aktivierten Interrupts

4.4.8 Clock Konfiguration

Der STM32F412 hat einen $f_{HSE} = 8.000\text{MHz}$ Clock von einem Quarz Oszillator. Dazu muss unter **System Core > RCC** die High Speed Clock (HSE) auf **Crystal/Ceramic Resonator** gesetzt werden. Die Abbildung 4.10 zeigt die notwendigen Einstellungen.

Da der TLV320 im Master Mode betrieben wird, kommt auch der Clock für die I²S Peripherie vom TLV320 über den CLK_OUT Pin. Die I²S Clock Frequenz beträgt $f_{clk} = 12.288\text{MHz}$. Im RCC Reiter muss die Audio Clock Input (I2S_CKIN) aktiviert werden.

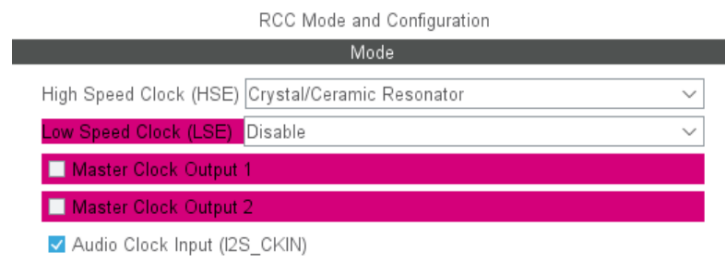


Abbildung 4.10: Auswahl der externen Clock (HSE)

Die maximale Taktfrequenz des STM32F412 liegt bei $f_{CPU} = 100\text{MHz}$. Damit die Geschwindigkeit erreicht wird, wird der interne PLL so konfiguriert, dass die CPU die maximale Taktfrequenz zugewiesen bekommt. Die Abbildung 4.11 zeigt alle notwendigen Konfigurationen inklusive der externen I²S Clock.

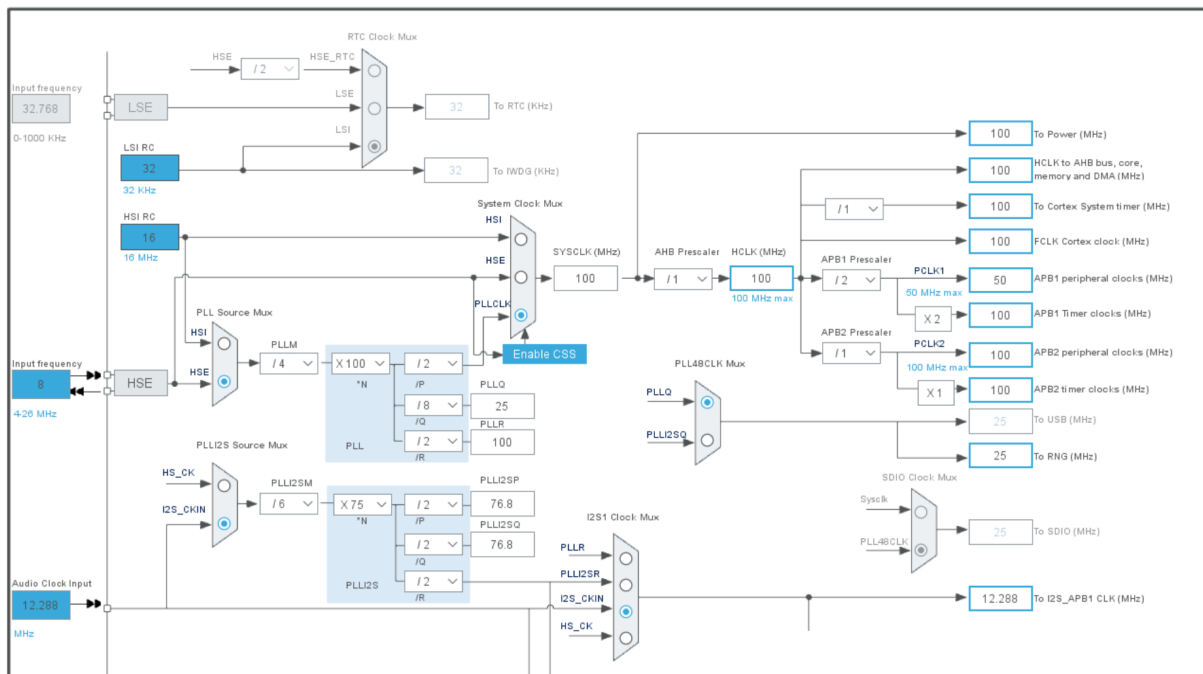


Abbildung 4.11: Gesamtübersicht der Clockkonfiguration mit $f_{CPU} = 100\text{MHz}$

4.5 Digitaler Datenfluss

In diesem Abschnitt wird gezeigt, wie der Datenstrom vom Codec über den DMA-Controller des STM32, durch die CMSIS/DSP Library und zurück über den DMA-Controller zum Codec gelangt. Dabei werden die wichtigen Programmierschnittstellen und Konfigurationen hervorgehoben.

SB

5 Validierung

5.1 Spannungsversorgung

Der Stromverbrauch beträgt 64mA

Kein Audio, beide Displays mit Menutext.

Vin 3.7 .. 4.2V

Messgerät SMU Agilent N6705B Channel 1 Strombegrenzung 600mA PrüfmittelNr. MSZ-M-0064

5.2 Akkumulator

5.2.1 ADC Messung der Batteriespannung

Die Spannungsmessung inkl. Korrekturfaktor für den Spannungsteiler mit Hilfe einer Source Measuring Unit überprüft. Die Dabei entstandenen Werte sind in der Abbildung 5.1 geplottet.

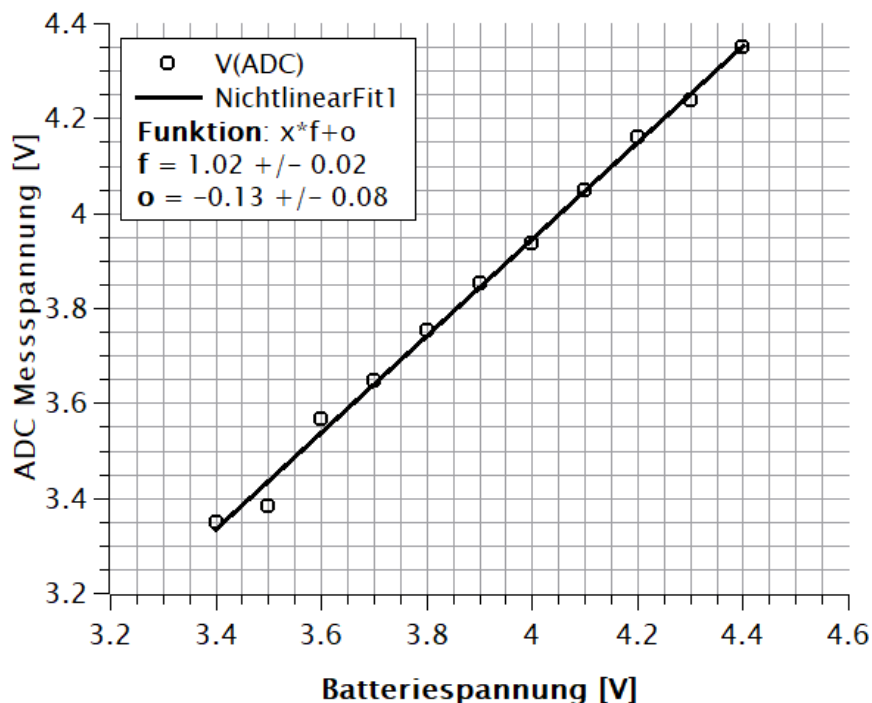


Abbildung 5.1: Akkumulatorspannung im Vergleich mit der gemessenen Spannung am ADC

Auf die gemessenen Werte liefert ein linearer Fit folgende Erkenntnis. Die Messwerte sind durchschnittlich um $V_{offset} = -130\text{mV}$ zu gering.

Verwendetes Messgerät:

Agilent Agilent N6705B (Prüfmittel Nr. MSZ-M-0064) auf Channel 1 mit Strombegrenzung $I_{max} = 600\text{mA}$.

6 Status und Verbesserungen

Pullups Datenleitungen

Analog Enable

FP Audio-Clock - der war doch dann doch kein Problem?

Beschriftungen In-/Outputs oben

Herausführen des PCBs für AVX-Stecker (Platz für Gehäuse)

Audio-Switch auf Board 4 auswechseln (schaltet nur eine Seite)

Battery IC hat kein Kühlpad

Filterkondensator bei ADC Spannungsteiler

7 Todo-Notes

■ SB - Ausgangspegel +6dB bis -40.5 dB (-34.5 -6)	4
■ - soll man hier noch mit Pmax vom LDO vergleichen?	6
■ SB - Erklären, wo, was zu finden ist mit Verweis auf Unterkapitel	11
■ SB - Muss erklärt werden, wie dieses Package heruntergeladen wird? evtl. bei installation von Keil	14
■ SB	23
■ Pullups Datenleitungen	24
■ Analog Enable	24
■ FP Audio-Clock - der war doch dann doch kein Problem?	24
■ Beschriftungen In-/Outputs oben	24
■ Herausführen des PCBs für AVX-Stecker (Platz für Gehäuse)	24
■ Audio-Switch auf Board 4 auswechseln (schaltet nur eine Seite)	24
■ Battery IC hat kein Kühlpad	24
■ Filterkondensator bei ADC Spannungsteiler	24