Fachhochschule Nordwestschweiz
Hochschule für Technik

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

Fachbericht: Projekt 5 - Burkhardt Simon, Studer Mischa
13. November 2019

Betreuung:	Prof. Dr. Markus Hufschmid
Team:	Simon Burkhardt Mischa Studer
Studiengang:	Elektro- und Informationstechnik
Semester:	Herbstsemester 2019

Abstract

Keywords

Inhaltsverzeichnis

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 einegünstige 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

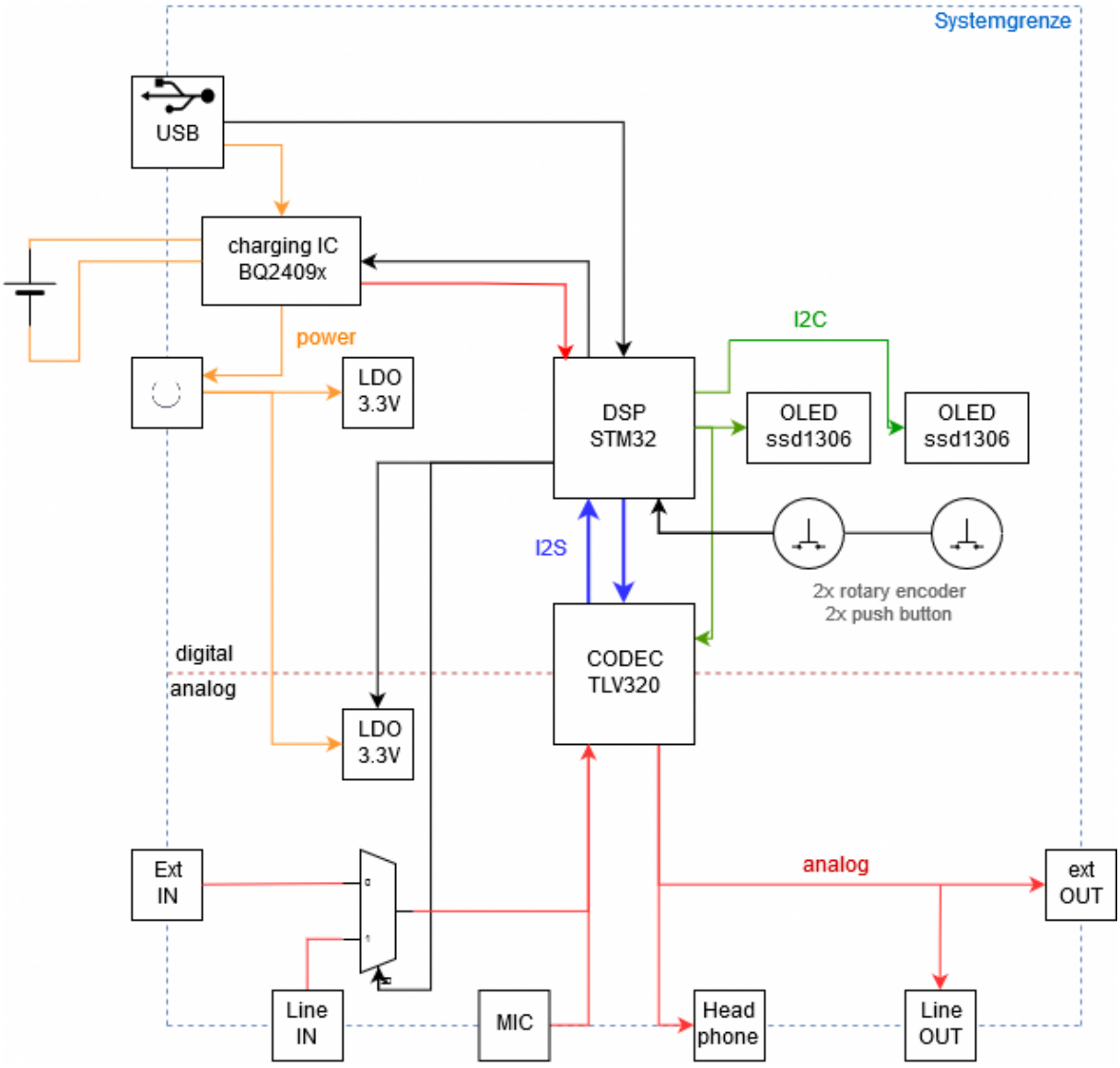

Abbildung 3.1: Blockschaltbild des DSP Boards

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

3.2 Pegeldiagram

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.

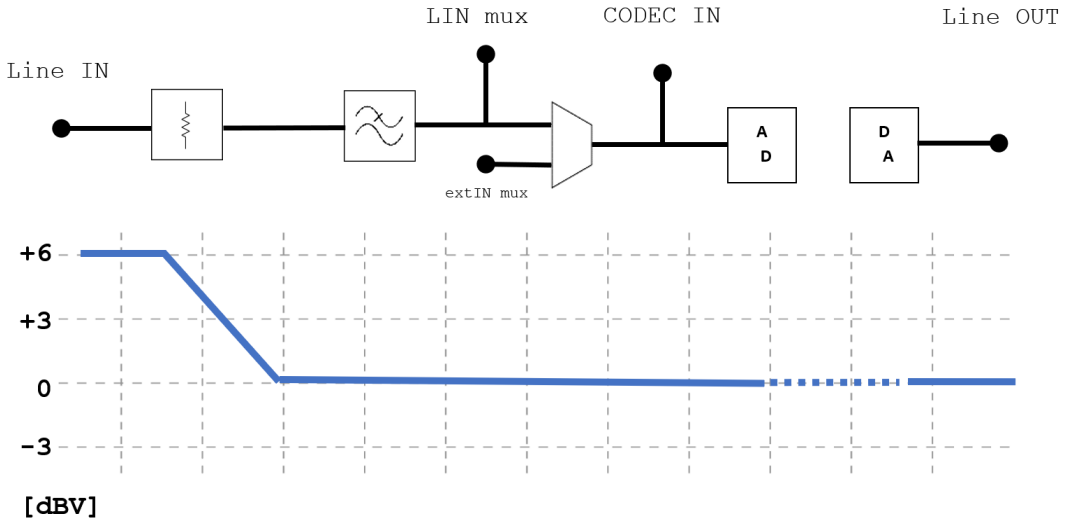

Abbildung 3.2: Pegeldiagram des Audiopfades von Line IN nach Line OUT

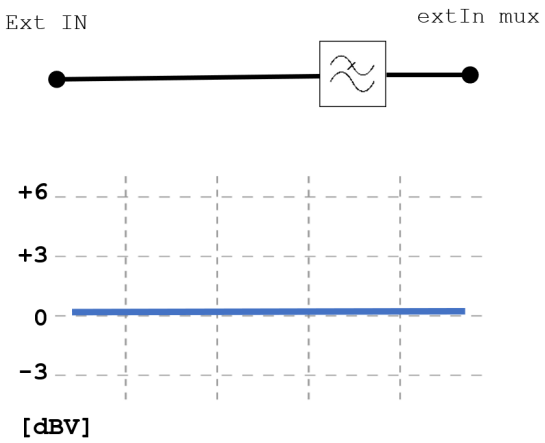

Abbildung 3.3: Pegeldiagram des Audiopfades von Ext IN bis zum Audio Switch

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 Kurschlussstrom, 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.

ADC Akkumulator Spannungsmessung

Die Akkumulatorspannung mit $V_{bat} = 4.2\text{V}$ 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.

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

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

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

$F_C = \frac{V_{bat_{max}}}{V_{out_{max}}} = \frac{3.27\text{V}}{4.5\text{V}} = \underline{0.7\bar{3}}$

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	**Imax** [mA]
STM32	40
SSD1306	30
SSD1306	30
reserve	50
textbfTotal	textbf150

Schaltungsteil	**Imax** [mA]
MAX4762	0.01
TLV320	26
reserve	30
textbfTotal	textbf56.01

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

Verlustleistung der Spannungsregler

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

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

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 Pinkonfiguration 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

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 gesammthaft 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

3.3.3 Schema Codec

3.4 PCB

3.5 Kosten

3.5.1 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:

$W * H/8 = 128 * 64/8 = 1024$ 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.

```
/* USER CODE BEGIN Includes */
#include "ssd1306.h"
/* USER CODE END Includes */
```

```
/* USER CODE BEGIN PV */
SSD1306_t holed1;    // Display Struct
/* USER CODE END PV */
```

```
/* USER CODE BEGIN 2 */
holed1.hi2cx = &hi2c1;   // set peripheral interface of Struct to I2C

ssd1306_Init(&holed1);
ssd1306_Fill(&holed1, Black);        // all pixels black
ssd1306_SetCursor(&holed1, 2, 0);    // x = 2px (from left) / y = 0px (from top)
ssd1306_WriteString(&holed1, "FHNW", Font_11x18, White); // medium font
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

3.5.2 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

```
/* USER CODE BEGIN 2 */
holed1.hi2cx = &hi2c1;   // set peripheral interface of Struct to I2C

ssd1306_Init(&holed1);
ssd1306_Fill(&holed1, Black);        // all pixels black
ssd1306_SetCursor(&holed1, 2, 0);    // x = 2px (from left) / y = 0px (from top)
ssd1306_WriteString(&holed1, "FHNW", Font_11x18, White); // medium font
ssd1306_UpdateScreen(&holed1);       // write Buffer to OLED Display
```

3.5.3 Encoder Mode mit Hardware Timer

Spezifikationen

Setting	Werte	Erklärung
Counter Mode	Up | Down	Zählrichtung in Abhängigkeit der Drehrichung
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

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

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