

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	1
2	Analyse und Konzept	2
3	Produktbeschreibung	2
4	Ziele	2
4.1	Harte Ziele	2
4.2	Weiche Ziele	2
4.3	Nicht Ziele	2
5	Lösungskonzept	2
6	Anforderungen an Microcontroller	3
7	Konzept USB Akkuladeregler IC	3
8	Kaskadierung mehrerer Boards	3
9	Hardware	4
9.1	Blockschaltbild	4
10	Pegeldiagramm	4
10.0.1	SSD1306 C Library	5
10.0.2	TLV320 C Library	6

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

3 Produktbeschreibung

4 Ziele

4.1 Harte Ziele

Nr	Ziel	Erreichungsgrad
1.1	Microcontroller mit Cortex-M4F Architektur	erfüllt/nicht erfüllt
1.2	Audio Passthrough von Line-In nach Line-Out	erfüllt/nicht erfüllt
1.3	Audio Schnittstelle (analog) - Line-IN - Line OUT	erfüllt/nicht erfüllt
1.4	2 Stk. Drehencoder für HMI	erfüllt/nicht erfüllt
1.5	2 Stk. Taster für HMI	erfüllt/nicht erfüllt
1.6	1 Display zur Anzeige des Funktionsmodus	erfüllt/nicht erfüllt
1.7	Microcontroller ohne Debugger über USB programmierbar	erfüllt/nicht erfüllt

4.2 Weiche Ziele

Nr	Ziel	Erreichungsgrad
2.1	Anzahl Layer der Leiterplatte	$n_{Layer} \leq 2$
2.2	Bauteile sind Handbestückbar - SMD passiv ≥ 0603 - SMD cases: keine QFN / BGA	erfüllt/nicht erfüllt
2.3	Stromverbrauch erlaubt Betrieb über USB 2.0 Speisung	$I_{USB} \leq 0.5A$
2.4	Audio Schnittstelle (analog) - Headphone OUT - Microphone IN	erfüllt/nicht erfüllt
2.5	Audio Verbindung (digital) - IN / OUT Board-to-Board Kommunikation Kaskadierung mehrerer Boards	erfüllt/nicht erfüllt
2.6	Akkubetrieb möglich	erfüllt/nicht erfüllt
2.7	zusätzliche (farbige) LEDs als Anzeige des Betriebsmodus	erfüllt/nicht erfüllt
2.8	Materialkosten inkl. PCB pro Stück bei 10 Stk.	$k \leq 50CHF$

4.3 Nicht Ziele

Nr	Ziel	Erreichungsgrad
3.1	DSP Board für den Unterricht	erfüllt/nicht erfüllt
3.2	Aufwändiges Gehäuse	erfüllt/nicht erfüllt

5 Lösungskonzept

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

6 Anforderungen an Microcontroller

Anforderungen an den Prozessor sind: ARM-Cortex M4 mit DSP und FPU sowie Schnittstelle(n) zur Kommunikation mit dem Audio Codec. Der Codec wird aufgrund der genaueren Samplingrate als Masterbetrieben. 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 Taktfrequenz von 200MHz ist wünschenswert, jedoch befinden sich die Cortex-M4 Prozessoren mit 200MHz auf dem Preisniveau von Cortex-M7 Microcontrollern.

7 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.

8 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 Verbinder soll kleiner als D-Sub sein. Auf eine digitale Schnittstelle wird wegen Clock-synchronisation und Kosten (der Steckverbinder) verzichtet.

9 Hardware

9.1 Blockschaltbild

10 Pegeldiagramm

10.0.1 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

10.0.2 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  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
```