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

3 Produktbeschreibung

4 Lösungskonzept

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

5 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 Tacktfrequenz von 200MHz ist wünschenswert, jedoch befinden sich die Cortex-M4 Prozessoren mit 200MHz auf dem Preisniveau von Cortex-M7 Microcontrollern.

6 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.

7 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.

8 Hardware

8.1 Blockschaltbild

8.2 Pegeldiagram

8.2.1 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

8.2.2 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
```

8.2.3 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 */
HAL_TIM_Encoder_Start(&htim2, TIM_CHANNEL_1);   // start Encoder mode on one channel
/* 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 */
```