GENERAL

stm 32f4 discovery keil example code codec

Published: September 3, 2026 | Index ID: #-

Introduction

When you first pick up the STM32F4 Discovery board, you're immediately drawn to its powerful Cortex M4 core, rich peripheral set, and the abundance of tutorials that showcase its capabilities. Among the many features that make this board a favorite among embedded developers, the integrated **audio codec** stands out. Whether you're building a simple audio playback system, a voice activated device, or a complex digital signal processing (DSP) application, mastering the codec is essential.

In this article, we'll walk through a **Keil example code** that demonstrates how to configure and use the codec on the STM32F4 Discovery board. We'll cover everything from hardware setup and Keil project configuration to detailed code explanations, debugging techniques, and performance tips. By the end, you'll have a solid foundation for creating your own audio projects on STM32F4.

1. Hardware Overview

1.1 STM32F4 Discovery Board

The STM32F4 Discovery board is built around the STM32F407VG microcontroller, featuring a 168 MHz Cortex M4 core, 1 MB flash, 192 KB SRAM, and a host of peripherals including USB, Ethernet, CAN, and, crucially for this article, an audio codec.

1.2 Integrated Audio Codec

Inside the board lies the **ST25L16C** audio codec, a dual channel 16 bit I2S audio interface capable of handling both playback and recording. It's connected to the microcontroller via I2C for control and I2S for data transfer. The codec supports a range of audio formats and sampling rates, making it versatile for many audio applications.

2. Getting Started

2.1 Prerequisites

- STM32F4 Discovery board
- USB cable (micro USB to USB A)
- Keil µVision IDE (Version 5 or later)
- STMicroelectronics STM32CubeMX (optional but recommended)
- ST-Link debugger/programmer (included with the board)
- Basic knowledge of C programming and embedded systems

2.2 Connecting the Board

Plug the USB cable into the board's micro USB port and connect the other end to your PC. The board should appear as a USB device, and Keil will automatically detect the ST-Link debugger.

3. Keil Project Setup

3.1 Creating a New Project

1. Open Keil µVision.

2. Navigate to Project !’ New µVision Project….
3. Select the STM32F4 series and choose STM32F407VG as the target device.
4. Give your project a name (e.g., CodecDemo) and choose a folder.

3.2 Adding Startup Files

Keil will prompt you to add a startup file. Accept the default **startup_stm32f407xx.s**. This file contains the reset vector and interrupt vector table.

3.3 Configuring the Toolchain

1. Open the Options for Target “CodecDemo” dialog.
2. Under the Target tab, set the Device to STM32F407VG.
3. Enable the Use MicroLIB option if you want a lightweight C library; otherwise, stick with the standard library.
4. In the Code Generation tab, set the Flash Run Mode to Run in Flash for optimal performance.

3.4 Adding Middleware and Drivers

Keil’s **Pack Installer** can automatically add the required CMSIS, HAL, and I2C/I2S drivers. Open **Pack Installer !’ Manage Packs** and install the **STM32CubeF4** pack. Then, use the **Import**feature to add the HAL drivers for I2C and I2S.

4. Codec Fundamentals

4.1 I2C Control Interface

The codec’s control interface is accessed via I2C. Typical commands include setting the audio format, enabling/disabling channels, and configuring power modes.

4.2 I2S Data Path

Audio data is transferred over the I2S bus. The STM32F4’s **I2S2**peripheral is used to send and receive 16 bit audio frames to and from the codec.

4.3 Common Parameters

- Sampling Rate: 8 /kHz, 16 /kHz, 32 /kHz, 44.1 /kHz, 48 /kHz, etc.
- Bit Depth: 16 /bit (default) or 24 /bit.
- Channels: Mono or Stereo.
- Audio Format: PCM, MSB Justified, LSB Justified.

5. Writing the Example Code

5.1 Project Structure

A typical Keil project for the codec example might contain the following files:

- main.c – application entry point.
- codec.c / codec.h – codec driver functions.
- i2s.c / i2s.h – I2S initialization and data handling.
- i2c.c / i2c.h – I2C driver for codec configuration.
- system_stm32f4xx.c – system clock configuration.
- stm32f4xx_hal_conf.h – HAL configuration.

5.2 System Clock Configuration

In **system_stm32f4xx.c**, configure the system clock to run at 168 MHz using the external 8 MHz crystal. This

ensures the I2S peripheral has enough bandwidth for high sample rate audio.

5.3 Initializing I2C

In **i2c.c**, set up I2C1 (or I2C2) with the following parameters:

- Mode: Fast (400 /kHz).
- Addressing: 7 bit.
- Own Address: 0x00 (master).

Example snippet:

```
I2C_HandleTypeDef hi2c1;
hi2c1.Instance = I2C1;
hi2c1.Init.ClockSpeed = 400000;
hi2c1.Init.DutyCycle = I2C_DUTYCYCLE_2;
hi2c1.Init.OwnAddress1 = 0x00;
hi2c1.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
hi2c1.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
hi2c1.Init.OwnAddress2 = 0x00;
hi2c1.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
hi2c1.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
HAL_I2C_Init(&amp;hi2c1);
```

5.4 Initializing I2S

Configure I2S2 in master transmit mode for playback or master receive mode for recording. Sample configuration for 48 /kHz stereo playback:

```
I2S_HandleTypeDef hi2s2;
hi2s2.Instance = SPI2;
hi2s2.Init.Mode = I2S_MODE_MASTER_TX;
hi2s2.Init.Standard = I2S_STANDARD_PHILIPS;
hi2s2.Init.DataFormat = I2S_DATAFORMAT_16B;
hi2s2.Init.MCLKOutput = I2S_MCLKOUTPUT_ENABLE;
hi2s2.Init.AudioFreq = I2S_AUDIOFREQ_48K;
hi2s2.Init.CPOL = I2S_CPOL_LOW;
hi2s2.Init.ClockSource = I2S_CLOCK_PLL;
hi2s2.Init.FullDuplexMode = I2S_FULLDUPLEXMODE_DISABLE;
HAL_I2S_Init(&amp;hi2s2);
```

5.5 Codec Driver Functions

In **codec.c**, implement functions to write to and read from the codec registers over I2C. A simple write function might look like this:

```
HAL_StatusTypeDef Codec_Write(uint8_t reg, uint8_t data) {
  uint8_t buf[2] = {reg, data};
  return HAL_I2C_Master_Transmit(&amp;hi2c1, CODEC_I2C_ADDR, buf, 2, HAL_MAX_DELAY);
}
```

Use similar logic for reading.

5.6 Configuring the Codec

After initializing I2C and I2S, configure the codec registers to set the audio format, enable the DAC/ADC, and power up the device. Typical sequence:

1. Reset the codec.
2. Set the Power Management registers to enable the DAC and ADC.
3. Configure the Digital Audio Interface registers for 16 bit PCM, 48 /kHz.
4. Enable the Volume control and set the desired level.

Example: To enable the DAC and set volume:

```
Codec_Write(0x01, 0x00); // Reset
Codec_Write(0x02, 0x80); // Power up DAC
Codec_Write(0x04, 0x10); // Set volume to 50%
```

5.7 Main Application Loop

In **main.c**, after initialization, create a simple audio buffer and stream data over I2S. For playback, you might read from a file stored in flash or generate a sine wave on the fly.

Example sine wave generation:

```
#define SAMPLE_RATE 48000
#define FREQUENCY 1000
#define AMPLITUDE 32767

int16_t buffer[256];
for (int i = 0; i
```

This loop continuously streams a 1 /kHz sine wave at full amplitude.

6. Debugging and Verification

6.1 Using the ST-Link Debugger

Set breakpoints at key initialization points to verify that I2C and I2S peripherals are correctly configured. Watch the **hi2c1** and **hi2s2** handles for any error flags.

6.2 Monitoring I2C Traffic

Use the **I2C Monitor** feature in Keil or a logic analyzer to confirm that the codec receives the correct register writes.

6.3 Audio Output Verification

Connect a 3.5 mm audio jack to the board's audio out port. Use a simple audio player or the sine wave generator to confirm that sound is produced. If no sound, check the power management registers and ensure the volume is not muted.

7.

Baca Juga (Artikel Terkait)

- **automaker mini software version v8 erptoboss**
  URL: http://apexgalvanizing.com/automaker-mini-software-version-v8-erptoboss.pdf
- **automata and computability**
  URL: http://apexgalvanizing.com/automata-and-computability.pdf
- **automata and languages theory and applications**
  URL: http://apexgalvanizing.com/automata-and-languages-theory-and-applications.pdf
- **automata and mechanical toys**
  URL: http://apexgalvanizing.com/automata-and-mechanical-toys.pdf

Tags: #32f4 #discovery #keil #example #code #codec