

Audio Decode by HELIX MP3

Introduction

This application note introduces how to port the open source Helix MP3 decoding algorithm to AT32F4xx MCUs. The source code in this application note demonstrates an MP3 player application using Helix MP3 decoding algorithm. This MP3 player uses SDIO interface to read MP3 content in SD Card, and then decode the content and play through a high-quality stereo CODEC WM8988.

The main contents of this document are as follows:

1. Introduction to Helix MP3 decoding algorithm
2. System hardware and software flow
3. Hardware resources
4. Operation procedures

Note: The corresponding code in this application note is developed on the basis of BSP_V2.x.x provided by Artery. For other versions of BSP, please pay attention to the differences in usage.

Applicable products:

Part number	AT32F403 series AT32F403A series AT32F407 series AT32F413 series AT32F415 series
-------------	--

Contents

1	Overview	5
1.1	HELIX MP3 decoding algorithm.....	5
1.2	MP3 player based on AT32 MCUs.....	5
2	MP3 software decoding.....	6
3	How to implement MP3 software decoding	8
3.1	Hardware.....	8
3.2	How to use MP3 after software decoding	9
4	Revision history.....	11

List of tables

Table 1. Document revision history.....	11
---	----

List of figures

Figure 1. MP3 system for software decoding	5
Figure 2. MP3 software decoding flow	6
Figure 3. State machine of MP3 software decoding	7
Figure 4. AT-START evaluation board and AT32 Audio EV Board	8
Figure 5. Wiring of AT-START and SD Card Module	8
Figure 6. Print information during MP3 decoding	10
Figure 7. Print information after MP3 decoding	10

1 Overview

MPEG1 Layer-3, otherwise known as MP3, is a format for audio. MPEG-1 is a compression standard for audio and video, which is classified into Layer1, Layer2 and Layer3 according to the compression quality and encoding complexity, corresponding to MP1, MP2 and MP3 audio files, respectively. The MP3 file takes one frame as an encoding unit, and the encoding data of each frame is independent.

1.1 HELIX MP3 decoding algorithm

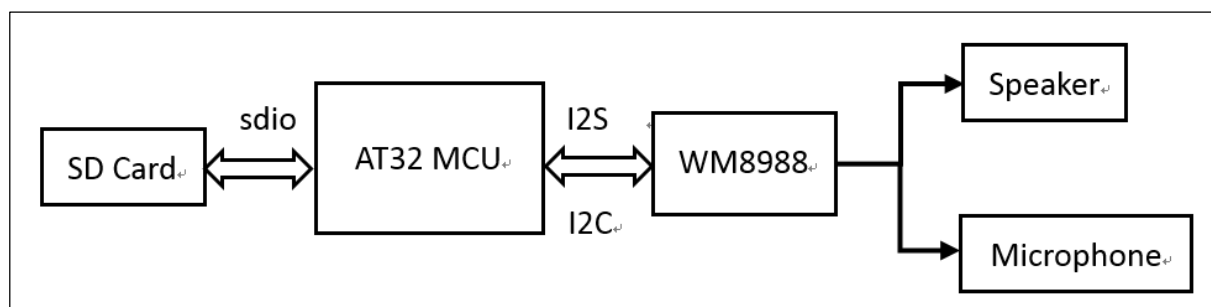
The Helix decoding algorithm can be implemented by fixed-point and floating-point operations. It is recommended to port Helix decoding algorithm to AT32 MCUs through fixed-point operation. This algorithm can run on any 32-bit fixed-point processor, and be fully encoded in C language, with some code segments optionally replaceable with optimized assembly instruction.

Helix supports MPEG-1, MPEG-2 and MPEG-2.5 Layer3, as well as variable bit rate, constant bit rate, stereo/mono audio formats. For more information about implementation and characteristics, please visit [Helix Community](#).

1.2 MP3 player based on AT32 MCUs

This application note introduces how to use Helix decoding algorithm to implement MP3 player on AT32F403 MCU, and provides source code based on AT32F4xx_StdPeriph_Lib and Helix algorithm. This MP3 player uses SDIO interface to read MP3 content in SD Card, and then decode the content and play through a high-quality stereo CODEC WM8988. Artery provides an Audio EV Board that incorporates Speaker, Microphone and WM8988 circuit, and uses I²C interface for WM8988 control and I²S interface for audio data transfer. Users can use the Audio EV Board and AT-START Board to set up an MP3 player.

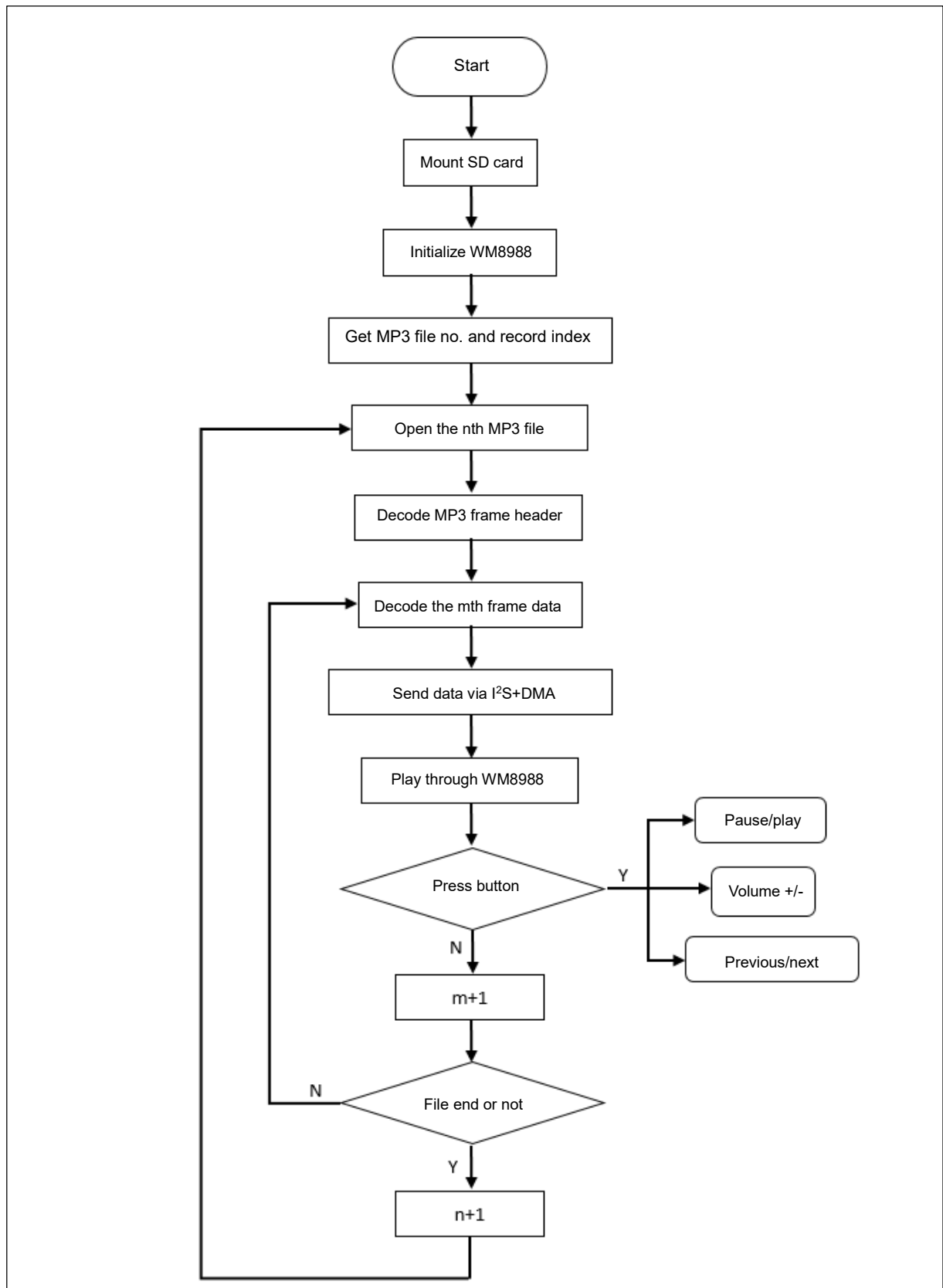
Figure 1. MP3 system for software decoding



2 MP3 software decoding

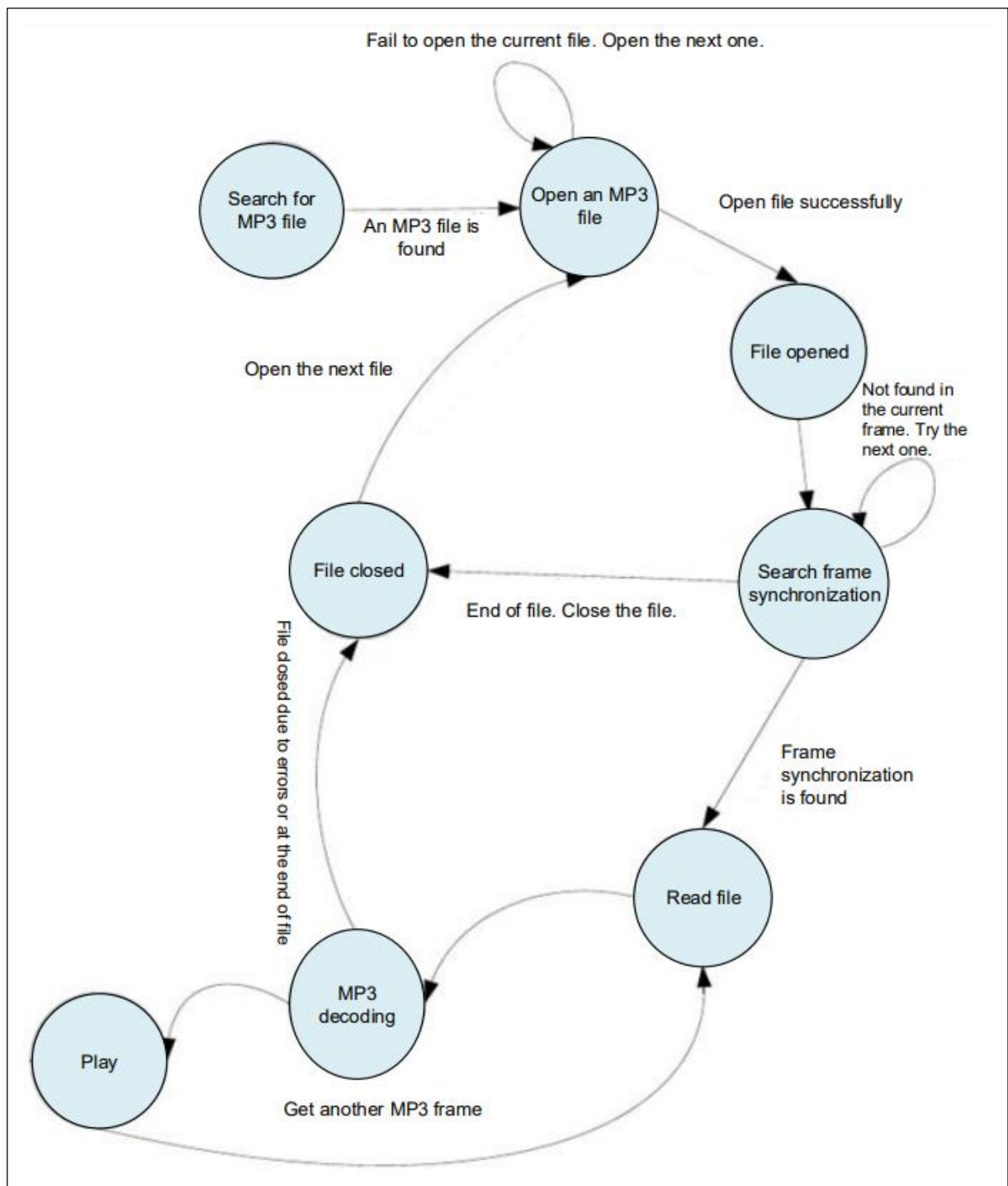
Figure 2 shows the MP3 software decoding flow.

Figure 2. MP3 software decoding flow



As shown in Figure 2, the kernel logic of MP3 decoding is implemented in the form of state machine, as shown in Figure 3 below. In idle state, the state machine waits for SD card connection. After the device is connected, the code starts to initialize SD card file system, and the application searches for MP3 files under "MUSIC" folder in SD card. If an MP3 file is found, the file name is to be stored in the "mp3indextbl []" array. Then, open the first file, and the code starts to search the first start of frame. If the start of frame is found, read the remaining of the frame from the file and transfer to the decoder function. The decoded audio frames are transferred to WM8988 audio DAC device for playback. The file read and decoding operations continue until the end of file. After this file ends and is closed, open the next file.

Figure 3. State machine of MP3 software decoding



3 How to implement MP3 software decoding

3.1 Hardware

- 1) AT-START-F403A V1.0 evaluation board
- 2) AT32 Audio EV Board V2.0 (WM8988)
- 3) External SD Card
- 4) 3.5mm headphone

Figure 4. AT-START evaluation board and AT32 Audio EV Board

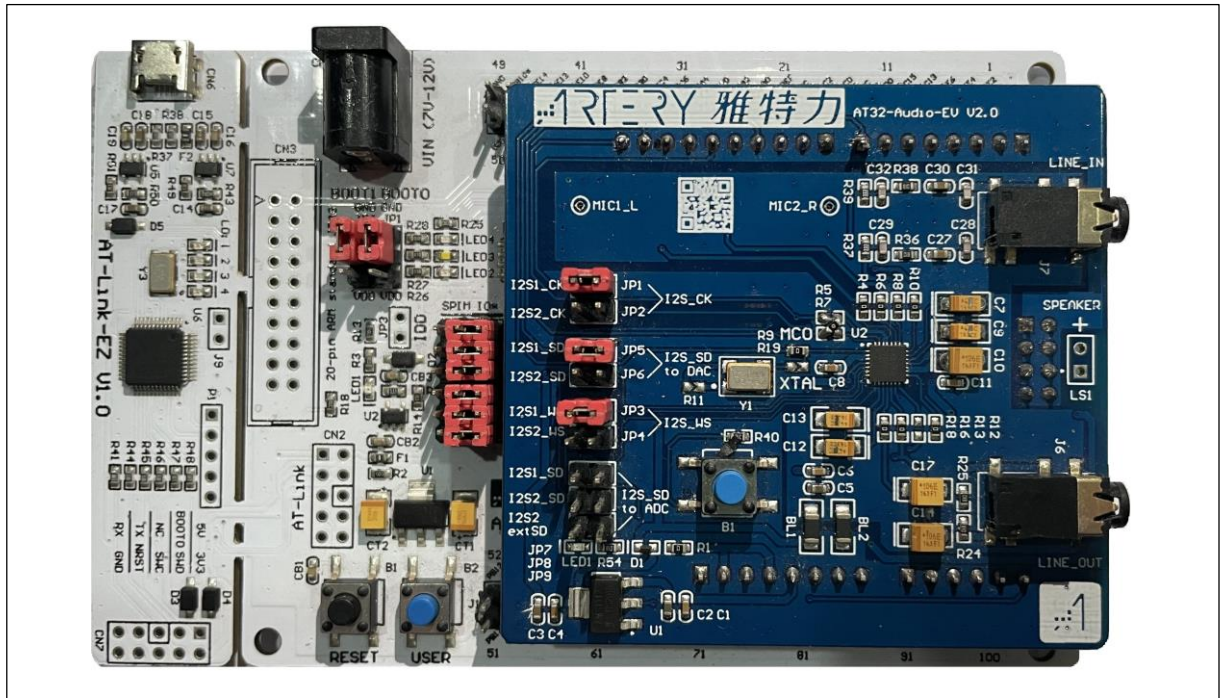
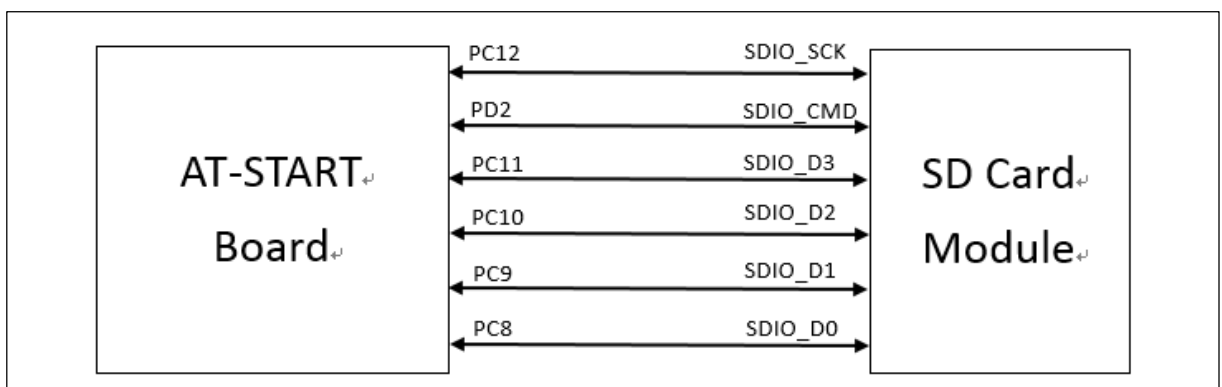


Figure 5. Wiring of AT-START and SD Card Module



Note: All projects are based on Keil 5. For usage in other compiling environment, please refer to the corresponding environment (IAR6/7, Keil 4/5) in AT32xxx_Firmware_Library_V2.x.x\project\at_start_xxx\templates and complete proper modifications.

Note: This demo is based on the hardware resources of AT32F403A. For usage in other AT32 MCU series, please modify configurations properly.

Note: The AT-START evaluation board does not incorporate an SD card circuit; therefore, an external SD card is required.

Note: The USER_KEY of AT-START evaluation board uses PA0, by default, which is changed to PC13 in this demo according to the AT_START_F403A_V1.0_SCH.pdf.

3.2 How to use MP3 after software decoding

1. Create a "MUSIC" folder in the root directory of the SD card to store audio files in mp3 format.
2. Connect the SD card to the corresponding pins on AT-START Board according to Figure 5.
3. Open the MP3 decode project source program; compile and then download to the evaluation board.
4. Connect the 3.5 mm headphone to the LINE_OUT port of Audio EV Board.
5. Press RESET button to play music.
6. Users can press B1/USER_KEY button to implement functions such as pause/play, volume+/-, or previous/next.

Only USER_KEY and B1 are available when connecting the Audio EV Board to the AT-START Board; therefore, these two buttons are reused to implement functions such as pause/play, volume+/-, or previous/next, following the below logic:

USER_KEY

- 1) Switch functions 1-5 of B1

B1

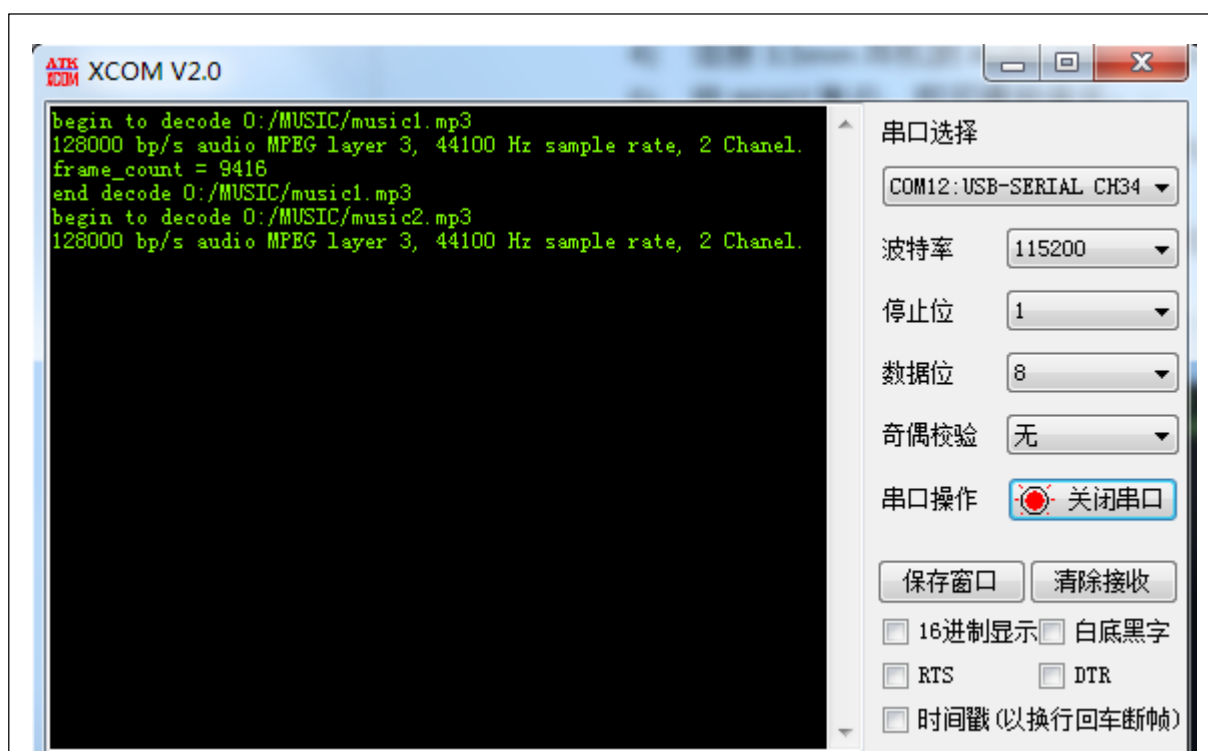
- 1) Function 1: pause/play
- 2) Function 2: volume +
- 3) Function 3: volume -
- 4) Function 4: previous
- 5) Function 5: next

In addition, the USART1_TX(PA9) can be connected to view the print information, including the music name, MP3 frame header and error, as shown in Figure 6 and Figure 7.

Figure 6. Print information during MP3 decoding



Figure 7. Print information after MP3 decoding



4 Revision history

Table 1. Document revision history

Date	Version	Revision note
2021.12.17	2.0.0	Initial release.

IMPORTANT NOTICE – PLEASE READ CAREFULLY

Purchasers are solely responsible for the selection and use of ARTERY's products and services; ARTERY assumes no liability for purchasers' selection or use of the products and the relevant services.

No license, express or implied, to any intellectual property right is granted by ARTERY herein regardless of the existence of any previous representation in any forms. If any part of this document involves third party's products or services, it does NOT imply that ARTERY authorizes the use of the third party's products or services, or permits any of the intellectual property, or guarantees any uses of the third party's products or services or intellectual property in any way.

Except as provided in ARTERY's terms and conditions of sale for such products, ARTERY disclaims any express or implied warranty, relating to use and/or sale of the products, including but not restricted to liability or warranties relating to merchantability, fitness for a particular purpose (based on the corresponding legal situation in any unjudicial districts), or infringement of any patent, copyright, or other intellectual property right.

ARTERY's products are not designed for the following purposes, and thus not intended for the following uses: (A) Applications that have specific requirements on safety, for example: life-support applications, active implant devices, or systems that have specific requirements on product function safety; (B) Aviation applications; (C) Aerospace applications or environment; (D) Weapons, and/or (E) Other applications that may cause injuries, deaths or property damages. Since ARTERY products are not intended for the above -mentioned purposes, if purchasers apply ARTERY products to these purposes, purchasers are solely responsible for any consequences or risks caused, even if any written notice is sent to ARTERY by purchasers; in addition, purchasers are solely responsible for the compliance with all statutory and regulatory requirements regarding these uses.

Any inconsistency of the sold ARTERY products with the statement and/or technical features specification described in this document will immediately cause the invalidity of any warranty granted by ARTERY products or services stated in this document by ARTERY, and ARTERY disclaims any responsibility in any form