AN1112: Helix MP3 Decoder on Series 1 EFM32 MCU Devices

This application note describes the fixed point implementation of the Helix MP3 decoder on EFM32PG12 and EFM32GG11 devices.

The MP3 data can be read from external Micro SD card or internal flash. The internal 12-bit VDAC or external 16-bit I^2S codec is used for MP3 decoded audio output.

KEY FEATURES

- Supports MPEG-1 layer 3, MPEG-2 layer 3 and MPEG-2.5 layer 3.
- Supports bitrate up to 320 Kbps.
- Supports mono and stereo modes.
- This application note includes:
  - This PDF document
  - Source files
  - Multiple IDE projects

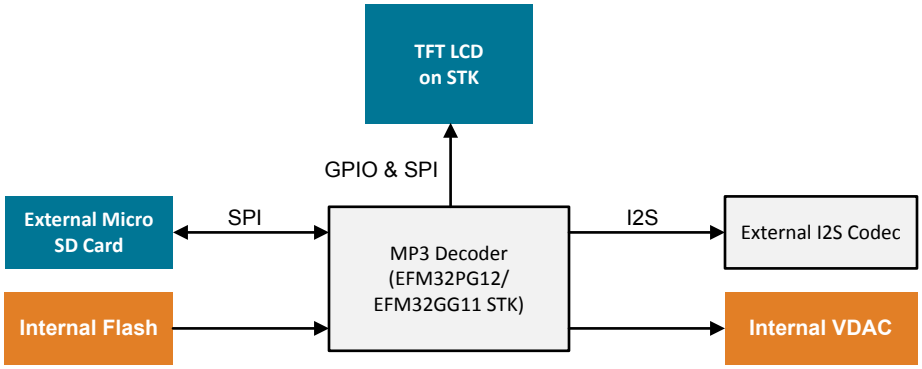


Table of Contents

1. Device Compatibility

This application note supports multiple device families. Note that some functionality is device-specific.

Devices supported include the following EFM32 Series 1 families:

- EFM32 Pearl Gecko (EFM32PG12/EFM32PG13)
- EFM32 Giant Gecko (EFM32GG11)

This project can also be ported to other EFM32 devices with at least 40 KB of RAM:

- EFM32 Giant Gecko (EFM32GG)

2. Introduction

The Helix MP3 decoder provides the Layer 3 support for MPEG-1, MPEG-2, and MPEG-2.5. It supports variable bit rates, constant bit rates, and stereo and mono audio formats. For more information on Helix MP3 decoder, see http://datatype.helixcommunity.org/Mp3dec.

Helix MP3 decoder source code is open source and is governed by the license described in files `RPSL.txt`, `RCSL.txt`, and `LICENSE.txt` in the `fixpt` folder that accompany the source code. Users are encouraged to read these license files and are requested to ensure compliance.

On April 23, 2017, Technicolor's MP3 licensing program for certain MP3-related patents and software of Technicolor and Fraunhofer IIS was discontinued. For more details, see www.mp3licensing.com.

3. Software Overview

This section describes the software drivers required for the MP3 decoder.

3.1 Helix MP3 Decoder

The Helix MP3 decoder provides MPEG-compliant decoding of MP3 content. The fixed-point decoder is optimized especially for ARM processors but can run on any 32-bit fixed-point processor which can perform a long multiply operation (two 32-bit inputs generating a 64-bit result) and long multiply-accumulate (long multiply with 64-bit accumulator).

3.1.1 Porting

The Helix MP3 decoder source code in this document is already modified to run on the MCU Series 1 devices.

Table 3.1. File Changes on Porting Helix MP3 Decoder

File	Changes
`assembly.h`	Add defines for EFM32 devices Inline functions for EFM32 devices
`mp3dec.h`	Add defines for EFM32 devices
`buffers.c`	Use static buffer instead of dynamic memory allocation

3.1.2 Output Buffer

The frame size is the number of samples contained in a frame. It is constant and always 1152 16-bit samples for Layer 3. Therefore, the maximum size required by the output buffer for stereo audio data is 2304 (1152 x 2) 16-bit samples.

3.1.3 Program Flow

The `MP3FindSyncWord(...)` function is used to search the MP3 Sync Word (`0xFFF` or `0xFFE`) to locate a potential MP3 frame. The `MP3GetNextFrameInfo(...)` function can then be used to extract the information about the yet to be decoded MP3 frame. The information is returned in a `MP3FrameInfo` type data structure. An error is returned in case of an invalid MP3 frame

The `MP3Decode(...)` function is used to decode the encoded MP3 frame. This function invokes the core MP3 decoding algorithm. One frame is decoded to get the output format for the samples, mainly this is sample rate and number of audio channels. These settings are only set once for each mp3 file.

The MP3 data for `MP3Decode(...)` function is loaded into the `readBuf[]` if Micro SD card option is selected; otherwise the MP3 data is directly read from the internal flash.

When the `getNextFrame` flag from LDMA interrupt service routine indicating the ping or pong audio buffer is empty the MP3 Decoder will decode one frame, first it searches for next valid MP3 Sync Word, and then decode and place the raw data into the ping or pong audio buffer. The audio buffer size is 2304 x 16-bit (`outBuf[]`) which is exactly two output frames from the MP3 Decoder, almost always be one frame being decoded and one being played.

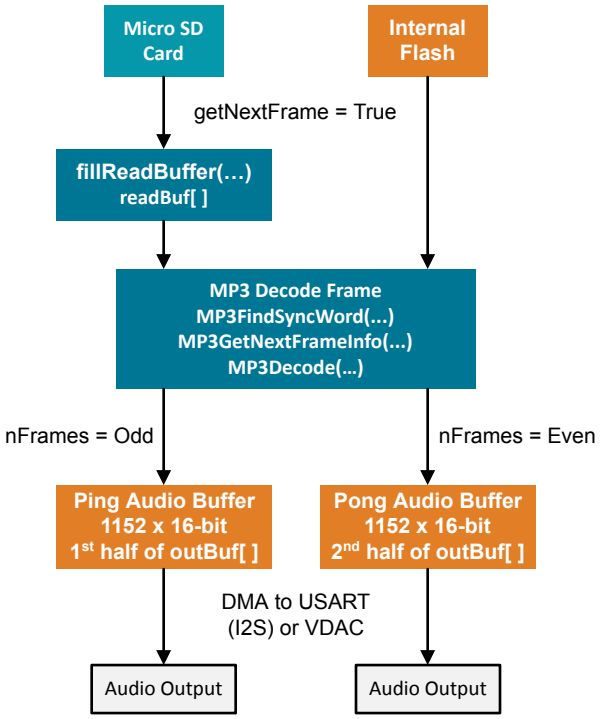


Figure 3.1. Block Diagram of Program Flow

3.2 Compile Options

The decoder uses the header file `mp3config.h` to set up the software and hardware environment. The hardware environment is configured by the items as shown in the following table.

Table 3.2. Table Parameters in `mp3config.h` File for Hardware Configuration

Item	Usage	Default Value
MP3_MEDIA_SELECT	Define the media for MP3 data storage 0 – Internal flash 1 – External Micro SD card	0
AUDIO_OUTPUT_SELECT	Define the source for audio output 0 – Internal 12-bit VDAC 1 – External 16-bit I^2S codec	0
I2S_CODEC_SELECT	Define which I^2S codec should be used (valid if AUDIO_OUTPUT_SELECT = 1) 0 – NXP UDA1334ATS 1 – Tempo TSCS25x	0

3.3 Micro SD card

This document modifies the `microsd.c` and `microsd.h` files in the following Windows directory to implement the Micro SD card support for MCU Series 1.
`C:\SiliconLabs\SimplicityStudio\v4\developer\sdks\gecko_sdk_suite\v1.1\hardware\kit\common\drivers`

The FAT support for Micro SD card is provided by FatFs, version R0.11 or above (support `f_findfirst()` and `f_findnext()` functions) is required to search all MP3 files in root directory of Micro SD card and play continuously.

See AN0030 FAT on SD card for more information. Application notes can be found on the Silicon Labs website (http://www.silabs.com/32bit-appnotes) or in Simplicity Studio (http://www.silabs.com/simplicity).

3.4 Outputting Data Using the VDAC

The internal 38 MHz HFRCO is selected for HFCLK if audio output is from VDAC. The VDAC clock is pre-scaled to about 1 MHz, and output enable is controlled by the PRS signal from the TIMER.

3.4.1 TIMER and PRS

The TIMER triggers the VDAC through PRS every sampling period, and one frame of data for VDAC conversion is collected by the LDMA from the software MP3 decoder. No MCU intervention is required during the VDAC data conversion phase, and the device stays in EM1 sleep mode.

3.4.2 LDMA

Ping-pong mode DMA is used to ensure the background VDAC conversion during the MP3 decoding process. The ping-pong buffer layout for the VDAC LDMA transfer depends on the MP3 sampling rate and is described in the following table.

Table 3.3. VDAC LDMA Ping-Pong Buffer Size

Sampling Rate (Hz)	Ping-Pong Buffer Size (Mono Mode)	Ping-Pong Buffer Size (Stereo Mode)
8000	576 x 16-bit	1152 x 16-bit (576 x 32-bit)
11025	576 x 16-bit	1152 x 16-bit (576 x 32-bit)
12000	576 x 16-bit	1152 x 16-bit (576 x 32-bit)
16000	576 x 16-bit	1152 x 16-bit (576 x 32-bit)
22050	576 x 16-bit	1152 x 16-bit (576 x 32-bit)
24000	576 x 16-bit	1152 x 16-bit (576 x 32-bit)
32000	1152 x 16-bit	2304 x 16-bit (1152 x 32-bit)
44100	1152 x 16-bit	2304 x 16-bit (1152 x 32-bit)
48000	1152 x 16-bit	2304 x 16-bit (1152 x 32-bit)

3.4.2.1 Mono Mode

In mono mode, the left channel decoded PCM data (LLL…) in the buffer is used for VDAC0 CH0, and VDAC0 CH1 is disabled. The DMA request source is VDAC0CH0 from the left channel, and 16-bit PCM data is written to VDAC0->CH0DATA through DMA.

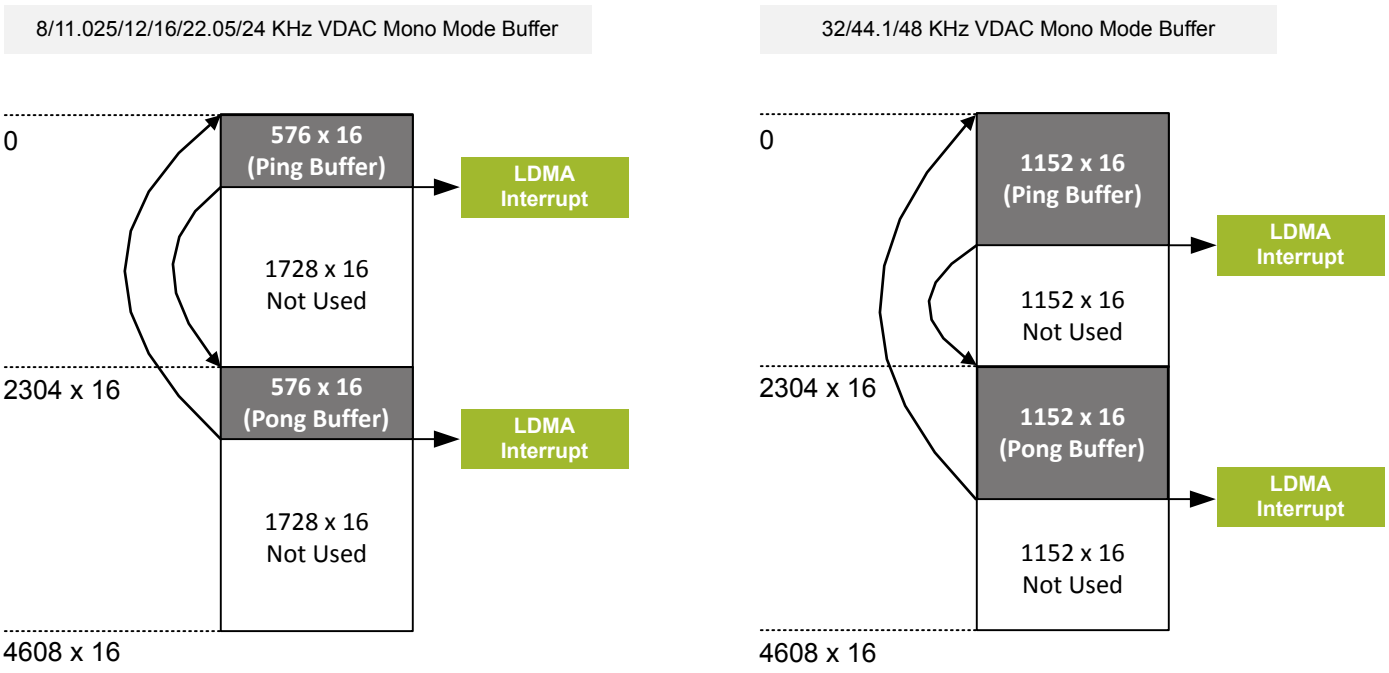


Figure 3.2. Ping-Pong Buffer Layout for VDAC Mono

3.4.2.2 Stereo Mode

In stereo mode, the interleaved left and right channel decoded PCM data (LRLRLR …) in the buffer is used for VDAC0 CH0 and VDAC0 CH1. The DMA request source is VDAC0CH0 from the left channel, and 32-bit PCM data for left and right channels is written to the VDAC0->COMBDATA register through DMA.

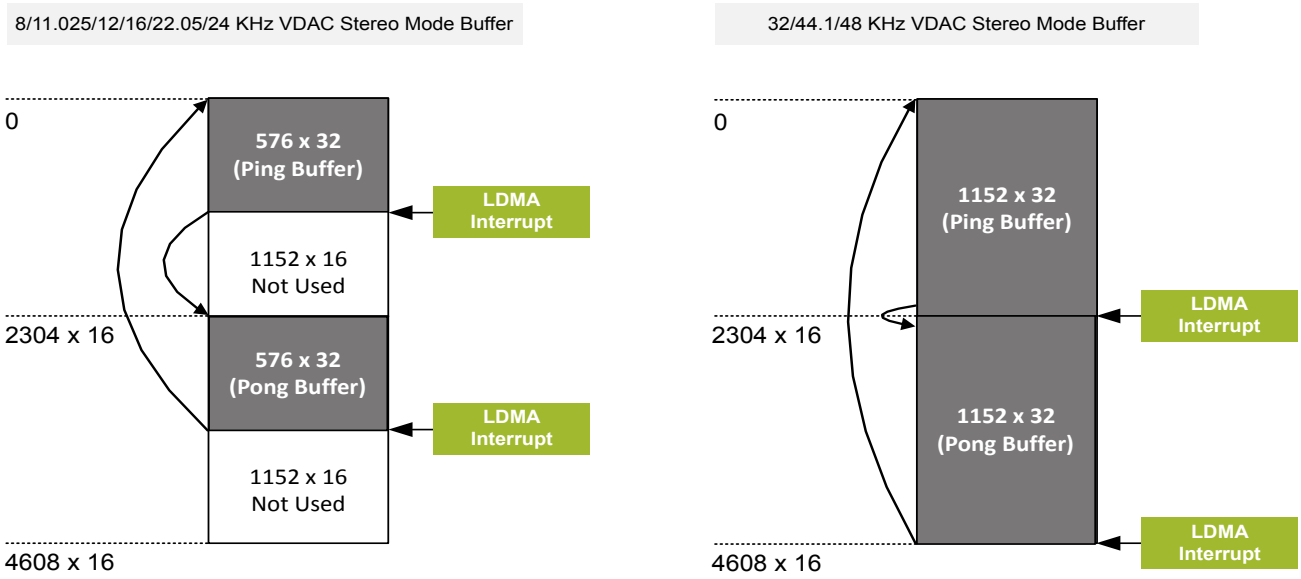


Figure 3.3. Ping-Pong Buffer Layout for VDAC Stereo

3.5 Outputting Data Using I²S

The USART is configured as I²S master, stereo 16-bit word, 16-bit data, transmit only, and regular I²S mode.

3.5.1 DPLL

The DPLL is used to generate a dedicated HFRCO frequency for different sampling rates. The LFXO is used as the DPLL reference clock. The HFRCO frequency should be ≥ 32 MHz to handle the software MP3 decoding process.

The DPLL can only generate 33882112 Hz from the 32768 Hz LFXO, which is a bit higher than the required 33868800 Hz for 11025, 22050, or 44100 sampling rates. The DPLL and USART settings for different sampling rates are described in the following table.

Table 3.4. DPLL and USART Settings for Different Sampling Rates

Sampling Rate (Hz)	I²S Bit Clock (Hz)	DPLL Factor N	DPLL Factor M	DPLL HFRCO (Hz)	USARTn_CLKDIV
8000	256000	2249	1	36864000	0x4700 (/144)
16000	512000	2249	1	36864000	0x2300 (/72)
32000	1024000	2249	1	36864000	0x1100 (/36)
11025	352800	2067	1	33882112	0x2F00 (/96)
22050	705600	2067	1	33882112	0x1700 (/48)
44100	1411200	2067	1	33882112	0x0B00 (/24)
12000	384000	2249	1	36864000	0x2F00 (/96)
24000	768000	2249	1	36864000	0x1700 (/48)
48000	1536000	2249	1	36864000	0x0B00 (/24)

Note:

$$I^2S\,Bit\,Clock = 2(Channels) \times 16(Data\,Width) \times Sampling\,Rate$$

$$DPLL\ \ HFRCO = 32768\left(LFXO\,Frequency\right) \times \frac{(DPLL\ \ Factor\,N + 1)}{(DPLL\ \ Factor\,M + 1)}$$

$$USARTn\_CLKDIV = 256 \times \left(\frac{DPLL\ \ HFRCO}{2 \times I^2S\,Bit\,Clock} - 1\right)$$

The 5-bit fractional part of USARTn_CLKDIV should be 0 for 50/50 duty cycle square wave on USART clock output.

3.5.2 LDMA

Ping-pong mode DMA is used to ensure background I^2S data transmission during the MP3 decoding process. No MCU intervention is required during the I^2S data transmission phase, and the device stays in EM1 sleep mode. The ping-pong buffer layout for I^2S LDMA transfer depends on the MP3 sampling rate and is described in the following table.

Table 3.5. I^2S LDMA Ping-Pong Buffer Size

Sampling Rate (Hz)	Ping-Pong Buffer Size (Mono Mode)	Ping-Pong Buffer Size (Stereo Mode)
8000	576 x 16-bit	1152 x 16-bit
11025	576 x 16-bit	1152 x 16-bit
12000	576 x 16-bit	1152 x 16-bit
16000	576 x 16-bit	1152 x 16-bit
22050	576 x 16-bit	1152 x 16-bit
24000	576 x 16-bit	1152 x 16-bit
32000	1152 x 16-bit	2304 x 16-bit (2 x 1152 x 16-bit)
44100	1152 x 16-bit	2304 x 16-bit (2 x 1152 x 16-bit)
48000	1152 x 16-bit	2304 x 16-bit (2 x 1152 x 16-bit)

3.5.2.1 Mono Mode

In mono mode, the left channel decoded PCM data (LLL…) in the buffer is used for both I²S left and right channels. Two DMA channels are required, and the DMA requests should be separated for left and right data by setting DMASPLIT to 1 in the USARTn_I2SCTRL register. The DMA request source is USARTnTXBL from the left channel and USARTnTXBLRIGHT from the right channel.

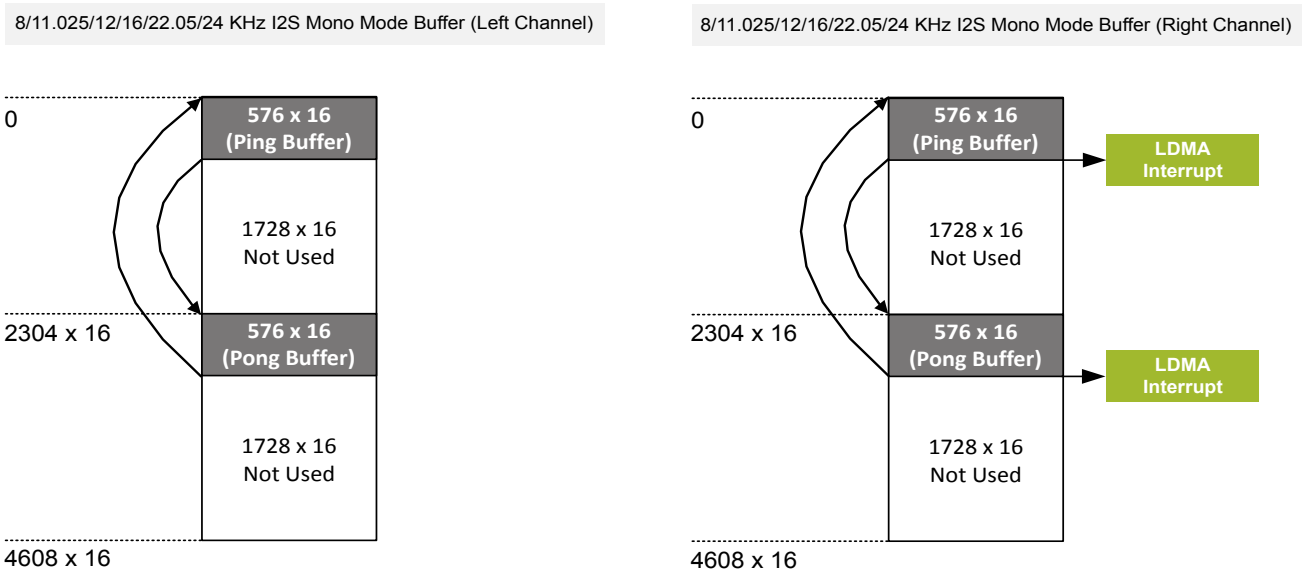


Figure 3.4. Ping-Pong Buffer Layout for I²S Mono (MPEG-2 and MPEG-2.5)

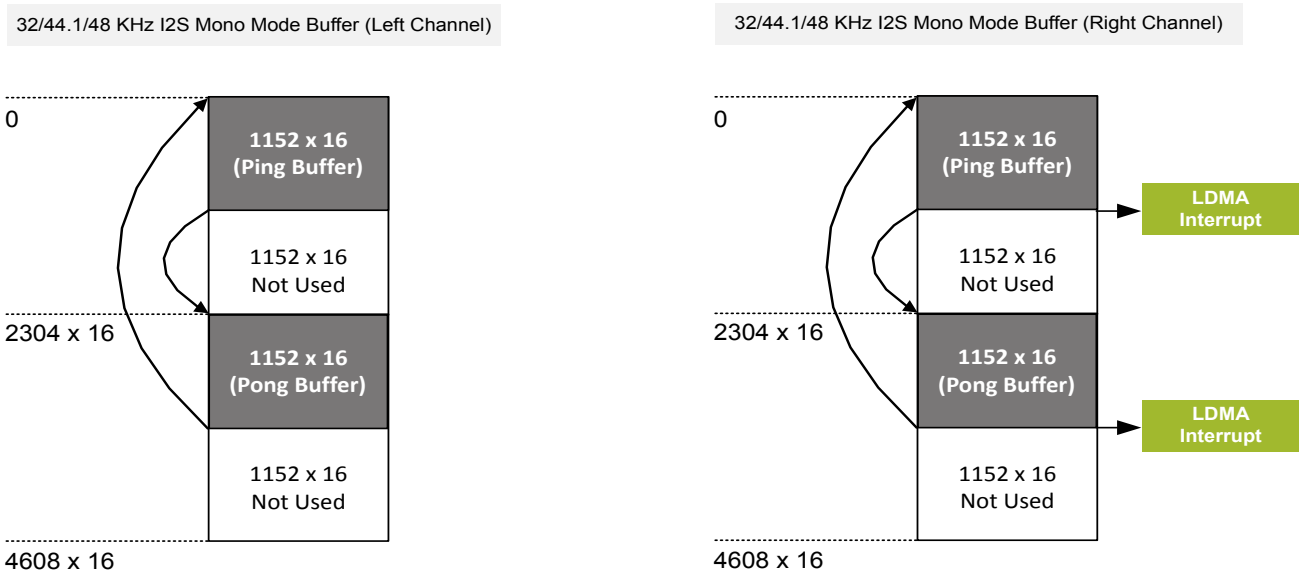


Figure 3.5. Ping-Pong Buffer Layout for I²S Mono (MPEG-1)

3.5.2.2 Stereo Mode

In stereo mode, the left and right channel decoded PCM data (LRLRLR...) is interleaved in the buffer. One DMA channel is enough, and the DMA requests should be combined for left and right data by clearing DMASPLIT to 0 in the USARTn_I2SCTRL register. The DMA request source is USARTnTXBL from the left or right channel. The maximum LDMA descriptor transfer count is 2048, so 2304 data transfers must be separated into two descriptors, each with 1152 transfer counts.

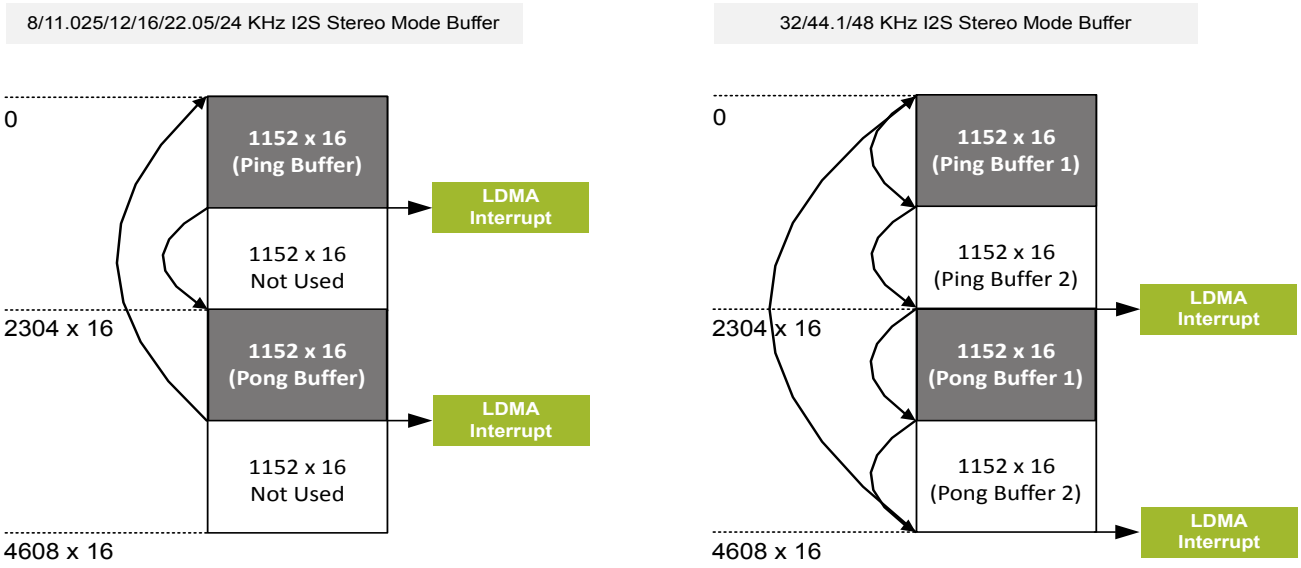


Figure 3.6. Ping-Pong Buffer Layout for I²S Stereo

3.6 Compiler Setting and Memory Footprint

The MP3 decoder should be compiled with the high-speed optimization option. The stack size is set to 0x0800 (2 KB) for safety. With these settings, the flash usage is about 70 KB and RAM usage is about 38 KB.

4. Hardware Overview

The MP3 decoder reads data from the Micro SD card or internal flash and decodes it to the memory buffer for output using an internal VDAC or external I2S codec.

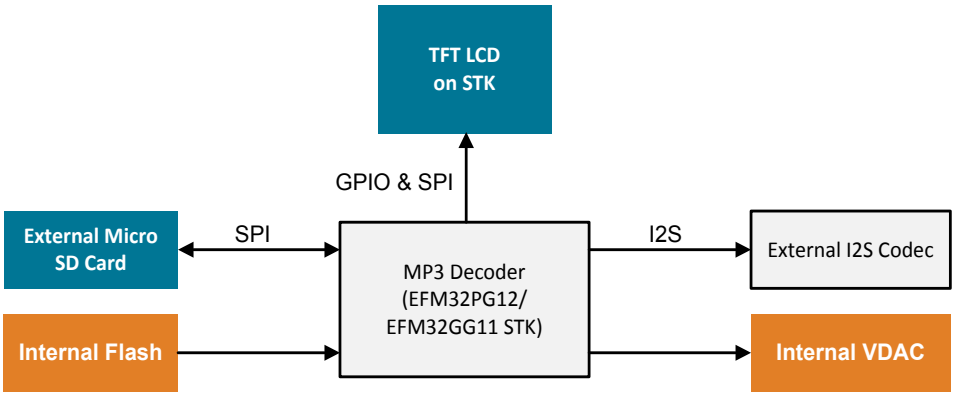


Figure 4.1. Block Diagram of MP3 Decoder

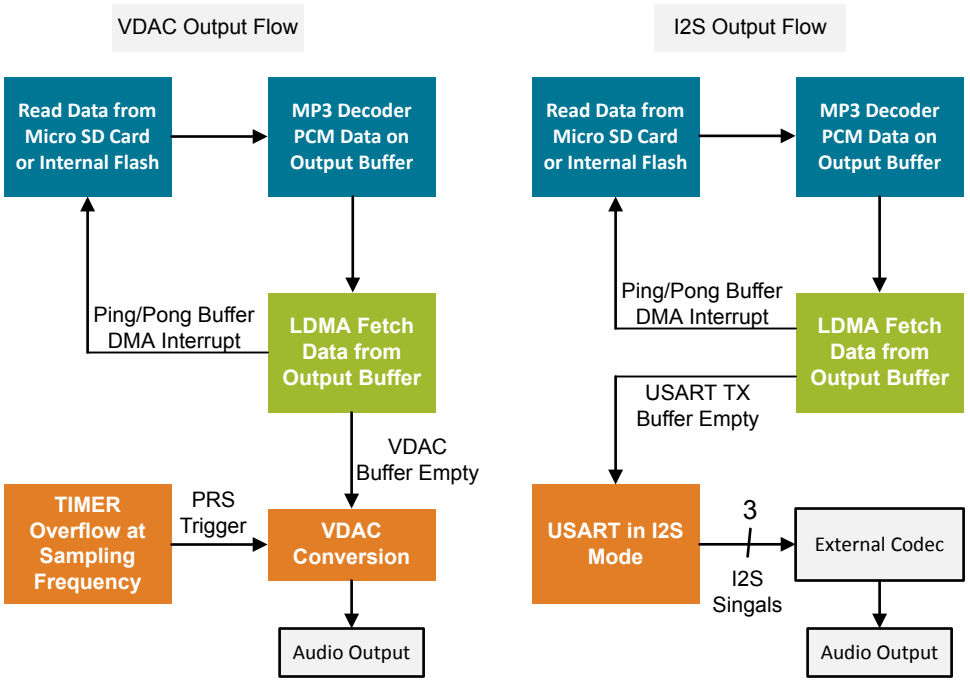


Figure 4.2. Block Diagram of Output Flow

4.1 Hardware Platform

The SLSTK3402A_EFM32PG12 and SLSTK3701A_EFM32GG11 STK are used as the hardware platform of the MP3 decoder.

Table 4.1. Table Resources of SLSTK3402A_EFM32PG12 STK used by Decoder

Signal	GPIO	Peripheral	Pin on STK
Micro SD card DI	PA6	US2_TX#1	EXP header pin 4
Micro SD card DO	PA7	US2_RX#1	EXP header pin 6
Micro SD card SCLK	PA8	US2_CLK#1	EXP header pin 8
Micro SD card CS	PA9	—	EXP header pin 10
I2S Codec DI	PB6	US3_TX#10	EXP header pin 7
I2S Codec BCLK	PB8	US3_CLK#10	EXP header pin 11
I2S Code WS	PD8	US3_CS#29	EXP header pin 13
I2C_SDA (For Tempo Codec)	PC10	I2C0_SDA#15	EXP header pin 16
I2C_SCL (For Tempo Codec)	PC11	I2C0_SCL#15	EXP header pin 15
I2S Codec RESET (For Tempo Codec)	PD10	—	EXP header pin 12
VDAC CH0	PA3	VDAC0_OUT0	J101 pin 11
VDAC CH1	PA2	VDAC0_OUT1ALT#1	J101 pin 9
EFM_DISP_ENABLE	PD15	—	—
EFM_DISP_MOSI	PC6	US1_TX#11	—
EFM_DISP_SCLK	PC8	US1_CLK#11	—
EFM_DISP_CS	PD14	—	—
EFM_DISP_COM	PD13	PRS_CH4#4	—

Table 4.2. Table Resources of SLSTK3701A_EFM32GG11 STK used by Decoder

Signal	GPIO	Peripheral	Pin on STK
Micro SD card DI	PE10	US0_TX#0	EXP header pin 4
Micro SD card DO	PE11	US0_RX#0	EXP header pin 6
Micro SD card SCLK	PE12	US0_CLK#0	EXP header pin 8
Micro SD card CS	PE13	—	EXP header pin 10
I2S Codec SDIN	PI0	US4_TX#2	J101 pin 5
I2S Codec BCLK	PC4	US4_CLK#0	EXP header pin 7
I2S Code LRCLK	PC5	US4_CS#0	EXP header pin 9
I2C_SDA (For Tempo Codec)	PC0	I2C0_SDA#4	EXP header pin 16
I2C_SCL (For Tempo Codec)	PC1	I2C0_SCL#4	EXP header pin 15
I2S Codec RESET (For Tempo Codec)	PE8	—	EXP header pin 12
VDAC CH0	PB11	VDAC0_OUT0	EXP header pin 11
VDAC CH1	PC12	VDAC0_OUT1ALT#0	J102 pin 21

Signal	GPIO	Peripheral	Pin on STK
EFM_DISP_ENABLE	PA9	—	—
EFM_DISP_MOSI	PA14	US1_TX#6	—
EFM_DISP_SCLK	PC15	US1_CLK#3	—
EFM_DISP_CS	PC14	—	—
EFM_DISP_COM	PA11	PRS_CH11#0	—

4.2 Memory for MP3 Data Storage

A compile option (Table 3.2 Table Parameters in `mp3config.h` File for Hardware Configuration on page 7) is used to select the Micro SD card or internal flash for the MP3 data storage.

4.2.1 Micro SD card

The Micro SD card is accessed through the SPI interface and the file system is handled by FatFs. Note that the SDIO interface in EFM32GG11 STK can also be used for the Micro SD card. This topic is, however, not discussed in this document.

4.2.2 Internal Flash

The MP3 file can be converted to C header file for internal Flash by emwin Bin2C.exe in the following Windows directory: `C:\SiliconLabs\SimplicityStudio\v4\developer\sdks\gecko_sdk_suite\v1.1\util\third_party\emwin\exe`.

4.3 Audio Output

A compile option (Table 3.2 Table Parameters in `mp3config.h` File for Hardware Configuration on page 7) is used to select an internal VDAC or external I^2S codec for the MP3 audio output.

Note: Do not attach or use headphones with the audio output. Use small loudspeakers with built in amplification, ensuring volume is at an acceptable level. Exposure to loud noises from any source for extended periods of time may temporarily or permanently affect your hearing.

4.3.1 VDAC

VDAC CH0 is used for mono mode. VDAC CH0 and VDAC CH1 are used for stereo mode. VDAC is periodically triggered by the TIMER PRS based on the sampling rate of the MP3 file.

4.3.2 External I^2S Codec

The two I^2S codecs supported in this application note are Tempo TSCS25x and NXP UDA1334ATS. A compile option (Table 3.2 Table Parameters in `mp3config.h` File for Hardware Configuration on page 7) is used to select which codec to use for audio output.

The reference circuits can be found in schematic of CP2615 Evaluation Kit (CP2615-EK) for Tempo TSCS25x and schematic of EFM32 Giant Gecko Development Kit (EFM32GG-DK3750) for NXP UDA1334ATS. The Tempo TSCS25x is configured by the MCU through the I^2C interface.

The schematics for the CP2615-EK board and EFM32 Giant Gecko Development Kit are available through Simplicity Studio (http://www.silabs.com/simplicity) when the kit documentation package has been installed. These documents are also available on the Silicon Labs website (https://www.silabs.com/support/resources.ct-schematic-and-layout-files).

Note: The Tempo TSCS25x codec does not support the 8 kHz sampling rate. The I^2S interface can be supported by USART1, USART3, or USART4.

4.4 LDMA

The LDMA is set up in ping-pong mode to allow the MCU to decode MP3 data for one buffer while data in the other buffer transmits to VDAC or USART (I^2S).

4.5 Display

The on board TFT display shows the related information of the MP3 file and RTCC (clocked by LFRCO) is for polarity inversion of the EFM_DISP_COM pin.

```
MP3 Player VDAC demo

Playing S_48K.MP3

mp3FrameInfo
outputSamps: 2304
bitrate:     128000
bits p.s:    16
nChans:      2
sampRate:    48000
layer:       3
version:     0
```

Note: Version 0, 1, and 2 stand for MPEG-1, MPEG-2 and MPEG-2.5.

5. Testing

5.1 MPEG-n Audio Layer 3 Sample Rates and Bit Rates

The available sampling rates and bit rates of MPEG-n Audio Layer 3 are listed in the following table.

Table 5.1. Available Sampling Rates and Bit Rates of MPEG-n Audio Layer 3

	MPEG-1	MPEG-2	MPEG-2.5
Sampling Rates (Hz)	32000	16000	8000
	44100	22050	11025
	48000	24000	12000
Bit Rates (Kbit/s)	32	8	8
	40	16	16
	48	24	24
	56	32	32
	64	40	40
	80	48	48
	96	56	56
	112	64	64
	128	80	
	160	96	
	192	112	
	224	128	
	256	144	
	320	160	

5.2 Test Files

The files for testing are saved in the `mp3file` and `mp3headerfile` folders of this application note. The properties of these files are listed in the following table. The `*.mp3` files in the `mp3file` folder can be copied to the root directory of the Micro SD card for external memory playback, whereas the `*.h` files in `mp3headerfile` folder can be included in `mp3config.h` for internal flash memory playback.

Table 5.2. MP3 Files for Testing

File Name	Sampling Rate (Hz)	Bit Rate (Kbit/s)	Mono/Stereo	Data Size
`m_8k.mp3/h`	8000	64	Mono	386059
`m_11k025.mp3/h`	11025	64	Mono	385915
`m_12k.mp3/h`	12000	64	Mono	385675
`m_16k.mp3/h`	16000	128	Mono	770827
`m_22k05.mp3/h`	22050	128	Mono	770437
`m_24k.mp3/h`	24000	128	Mono	770059
`m_32k.mp3/h`	32000	128	Mono	770251
`m_44k1.mp3/h`	44100	128	Mono	770019
`m_48k.mp3/h`	48000	128	Mono	769675
`s_8k.mp3/h`	8000	64	Stereo	386059
`s_11k025.mp3/h`	11025	64	Stereo	385915
`s_12k.mp3/h`	12000	64	Stereo	385675
`s_16k.mp3/h`	16000	128	Stereo	770827
`s_22k05.mp3/h`	22050	128	Stereo	770437
`s_24k.mp3/h`	24000	128	Stereo	770059
`s_32k.mp3/h`	32000	128	Stereo	770251
`s_44k1.mp3/h`	44100	128	Stereo	770019
`s_48k.mp3/h`	48000	128	Stereo	769675

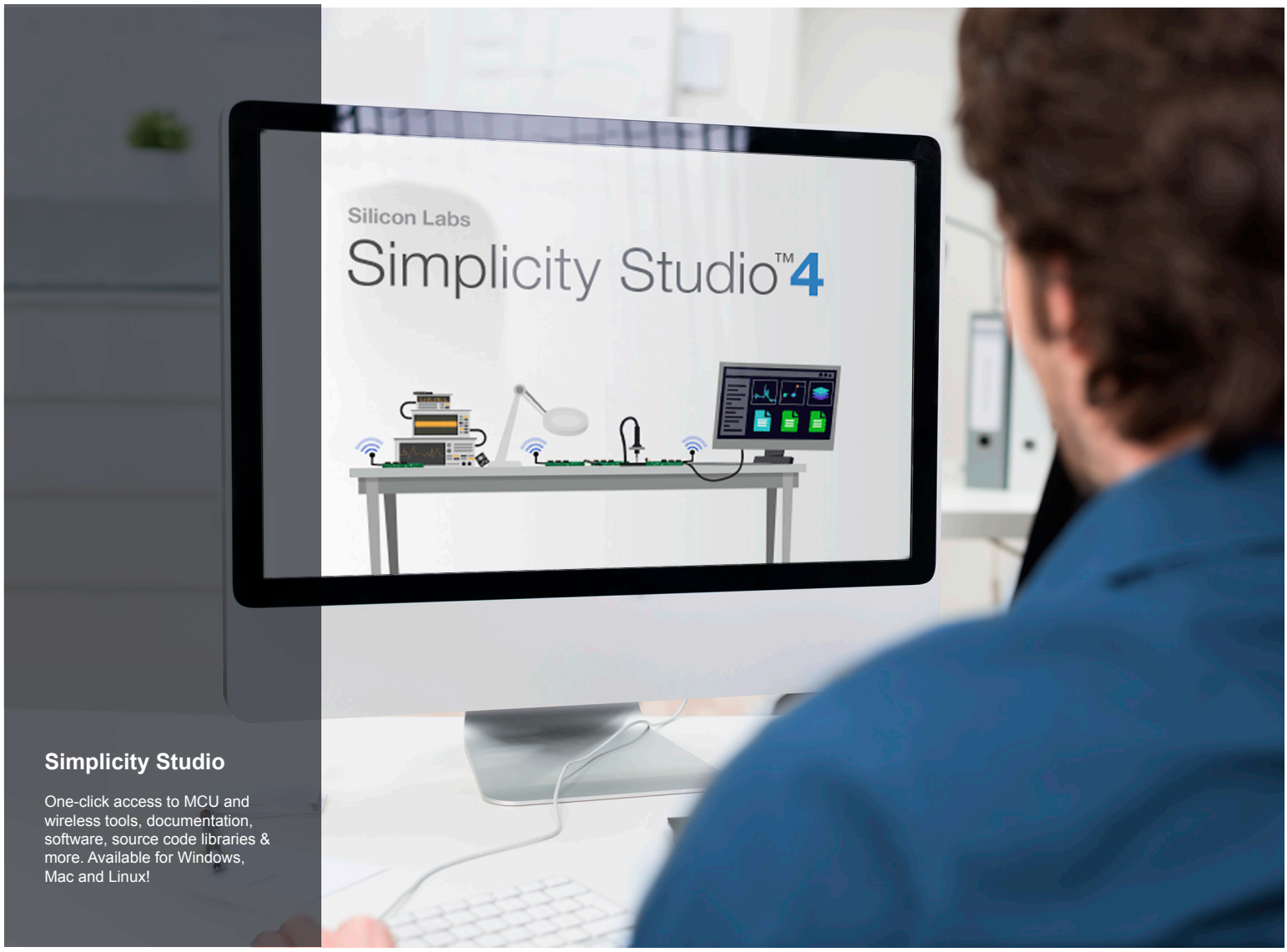
Note: Data size is bitrate-dependent, converts stereo to mono, and does not directly affect the data size of a MP3 file.

6. Revision History

6.1 Revision 0.1

2017-8-8

Initial revision.



IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701