AN2400
Application note

Audio player evaluation board based on ST7Lite

Introduction

This application note demonstrates how to add audio playback to any application using a general-purpose ST7 microcontroller. To demonstrate this feature, an evaluation board, source code in C, schematics, and layout are available.

The Audio player evaluation board reconstructs audio signals through the PWM of a ST7FLITES2 microcontroller. Serial Flash is used to store the audio data in a binary file format. To store this audio data in the Flash, the sound files need to be in *.wav* format so that can be converted into *.bin* file format with the help of the PC GUI available with this package. Once the data is stored, the microcontroller reads it through an SPI interface and generates the sound using the PWM feature. A sixth order low pass filter removes any unwanted high frequency components from the signal before passing it through a speaker.

The key features of the Audio player evaluation board are as follows:

- 1% calibrated on-chip RC oscillator. The used microcontroller, ST7FLITES2, contains an internal RC oscillator with an accuracy of 1% for a given device. It can be calibrated to obtain a frequency required for the application. There is therefore no need for any external oscillator.
- LVD (brown-out) on chip. The purpose of the Low Voltage Detector (LVD) is to ensure that the ST7 always functions in its safe area. No external reset circuit is required.
- Small footprint solution.
- Low power operation. Reduces battery/power supply cost (through extended battery life).
- Supports lower density serial Flash if fewer messages needed.
- Order of filter can be reduced to cut cost even more.

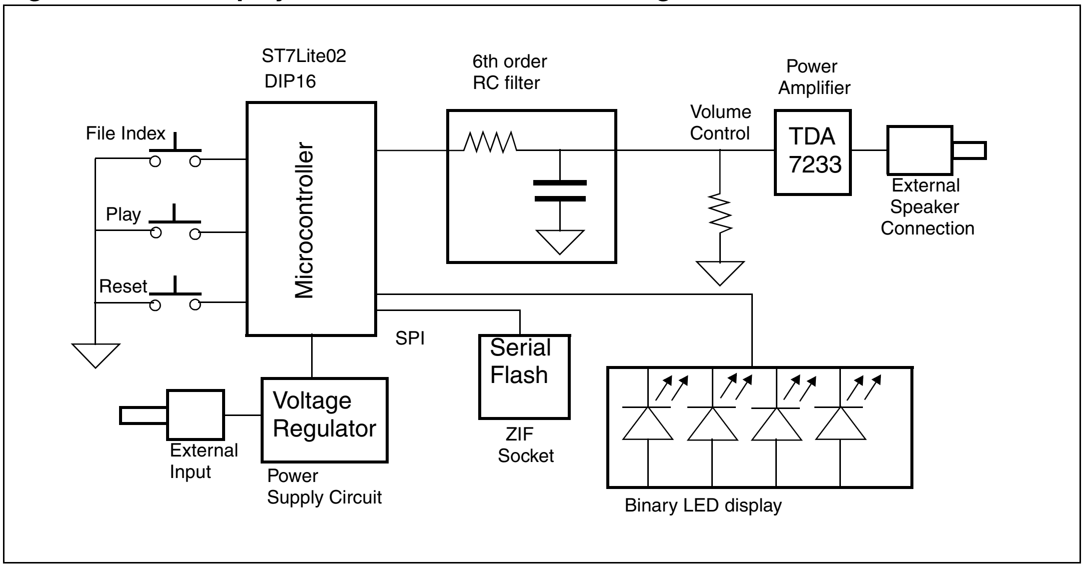
Contents

1 Audio player evaluation board overview

This evaluation board includes the following main components:

- 8-bit microcontroller: ST7FLITES2. However, the ST7Flite02 can also be used.
- An external 8 MBit Serial Flash, M25P80 family
- External components to build a low cost passive filter.
- An audio amplifier (TDA7233)
- 3.3V low-cost voltage regulator

Figure 1. Audio player evaluation board block diagram



The main features of the evaluation board are as follows:

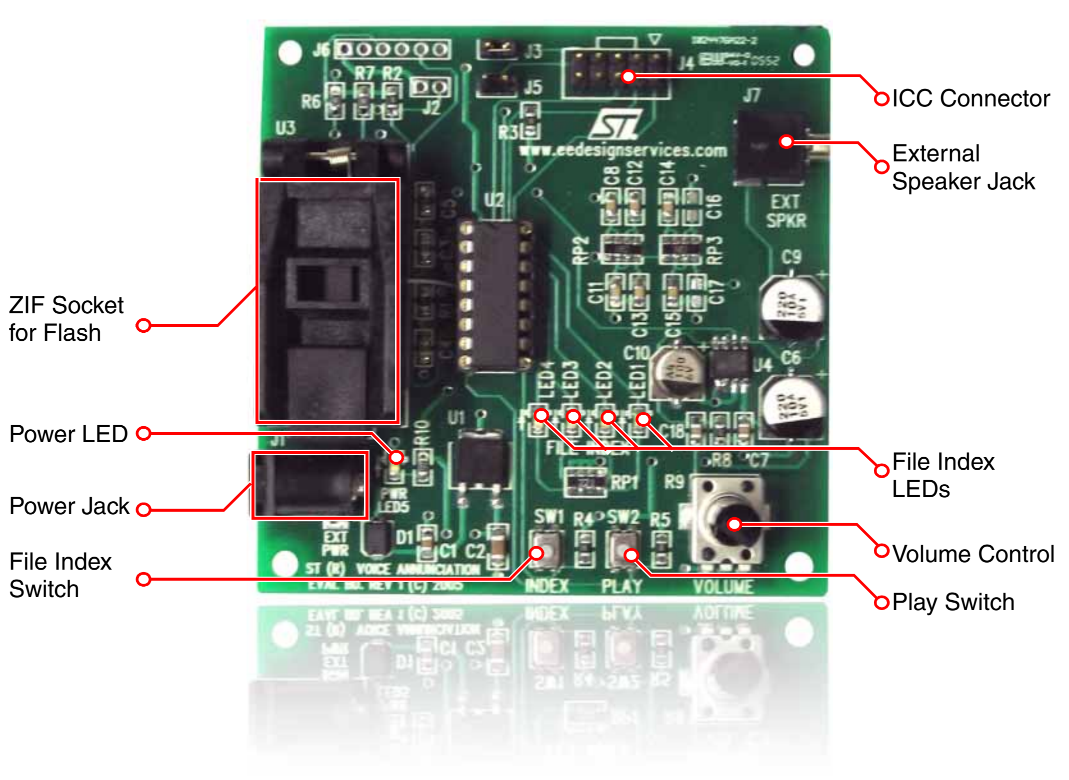
- Plays pre-recorded audio clips stored on an ST 8-Mbit serial Flash
- Can play up to 15 different audio files
- Easy to use **Index** and **Play** push-buttons
- Power input 6V DC
- Phone-quality sound: 8 kHz/8 bit

1.1 Instructions for use

Assuming the external Flash is already programmed with sounds, you need to:

1. Power-on the board
2. Connect the board to the speaker
3. Select the desired sound by pressing the **Index** button. The **File Index** indicates the sound to be played with the help of 4 LEDs.
4. Push the **Play Sound** button.
5. Adjust the **volume** potentiometer to your convenience.

Figure 2. Labeled diagram of Audio player evaluation board



2 Audio player evaluation board concept

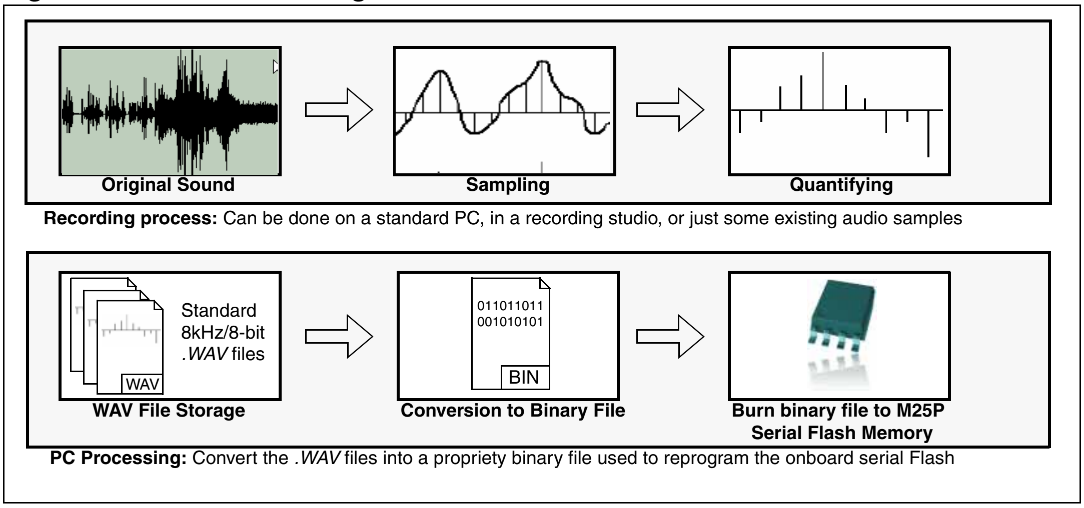
2.1 Audio recording

A PC is used to prepare audio to be programmed onto the external serial Flash. The file must contain audio data as well as some basic file management structure to allow the ST7 to pick the correct audio during playback.

The serial Flash is used to store the audio files in binary format. For this purpose, it is necessary to have the audio files in *.wav* format which allows them to be converted to a *.bin* file to be programmed into the serial Flash.

For conversion to binary, the *.wav* files should have an 8 kHz sampling frequency and 8-bit quantization format. These files are input into the provided WAV Converter utility to generate a single *.bin* file.

Figure 3. Sound recording flow

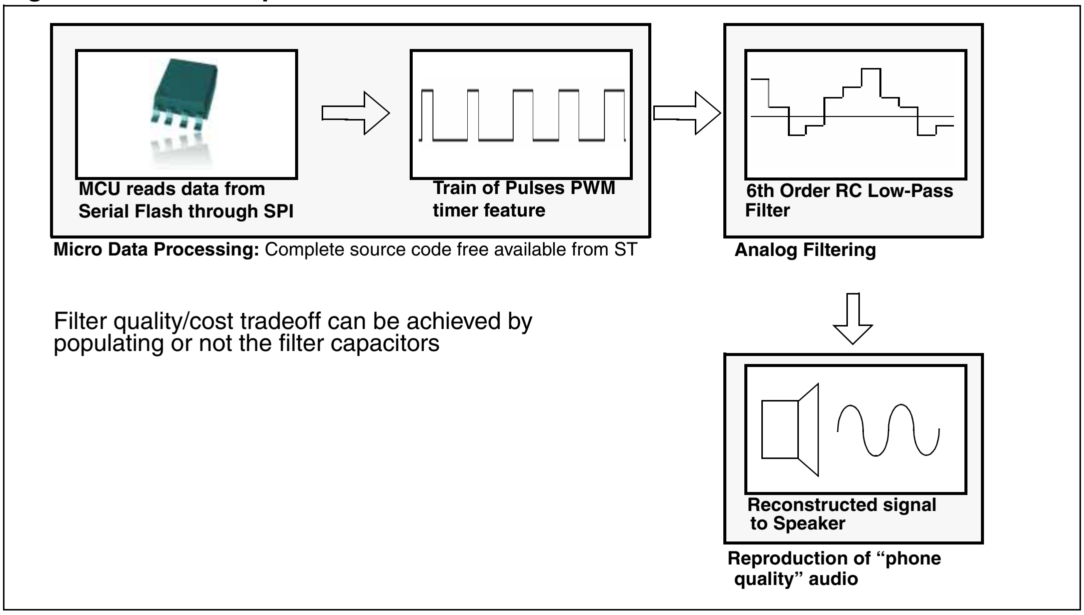


Once this process is completed, the board is ready to play customized sounds provided by the user.

A sound is selected using the **Index** push-button, and then heard by pressing the **Play** button.

3 Audio reproduction flow

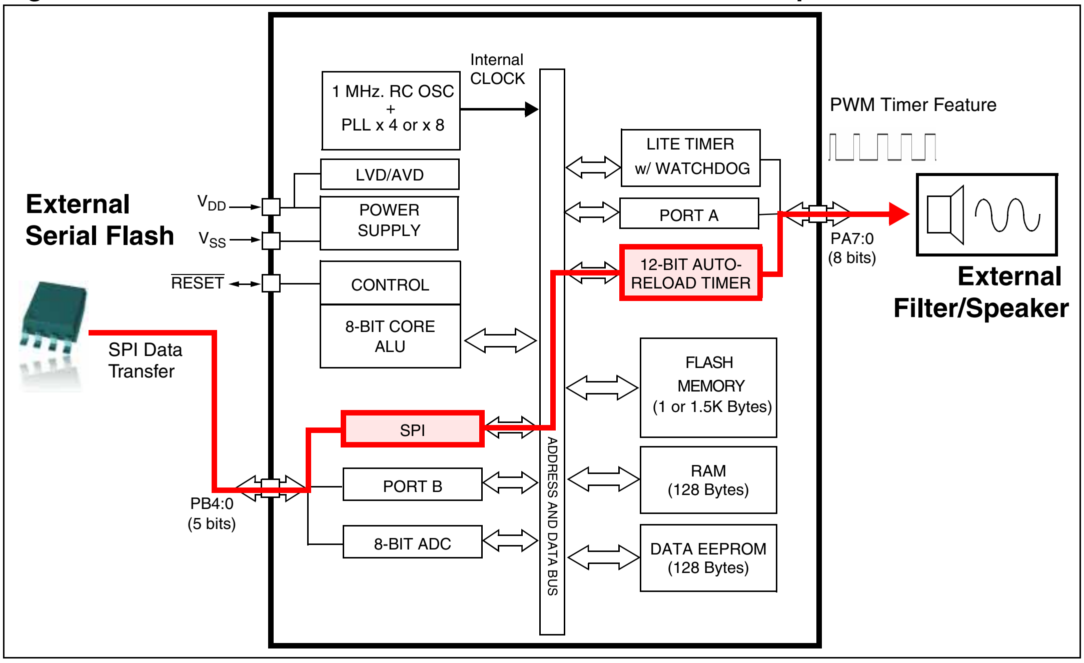
Figure 4. Audio reproduction flow



Data, which is stored in the serial Flash, is read by the microcontroller through the SPI interface. This data is then fed through the timer registers to generate a PWM output whose duty cycle varies according to the data value.

The sequential path of the data flow from the serial Flash to the speaker is shown in *Figure 5*.

Figure 5. Flow of data from external serial flash to external speaker



The microcontroller initiates the communication when it selects the Flash and starts reading the data on a full duplex, synchronous basis. Once the microcontroller reads the data, it generates a train of pulses using the PWM feature of the 12-bit auto reload timer. This is explained with the help of an example:

Audio samples of 8 kHz, 8-bit format are to be reproduced. This data is coded in an 8-bit format (with values from 0 to 255).

To reproduce the audio with sampling rate of 8 kHz, the microcontroller outputs each value in every 1/8000 sec (every 125 µs). This means that the period of the PWM is set at 125µs.

If the coded value of the audio signal is “1”, the microcontroller needs to generate the PWM signal with a HIGH output for 1 count and LOW for the rest of the 255 counts. This is shown in the figure below:

Figure 6. Waveform depicting the microcontroller output

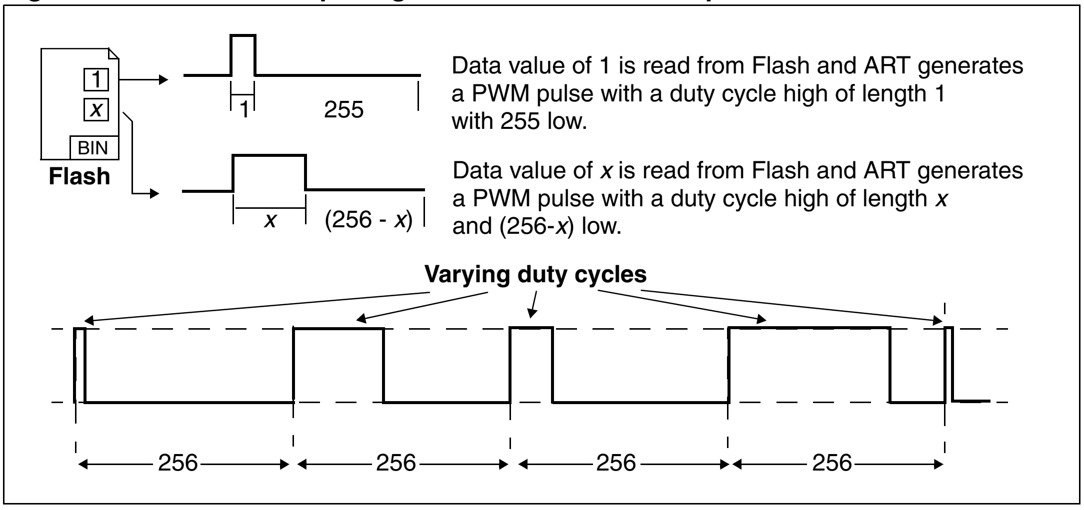


Figure 7 below shows how the duty cycle register generates this output.

This process uses a duty cycle register, a 12-bit auto reload register and an up-counter.

When an up-counter overflow occurs (OVF event), the ATR value is loaded into the up-counter, the preloaded duty cycle is transferred to the duty cycle register. The PWM0 signal is then set to a high level. Finally, when the upcounter matches the DCRx value, the PWM0 signal is set to a low level.

To use this sequence for audio generation, the following steps are taken:

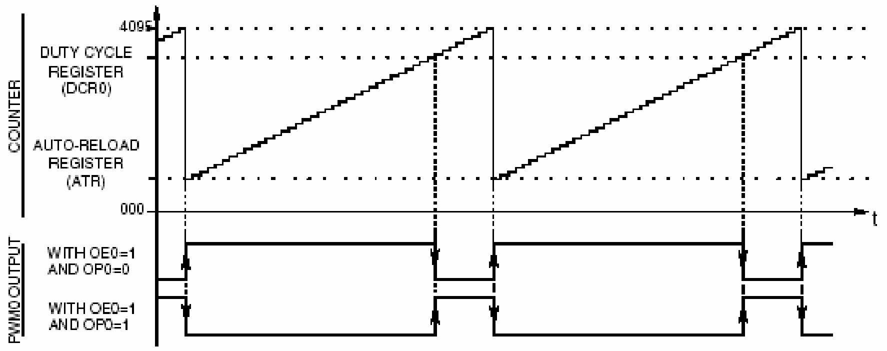
1. When the up-counter reaches 0xFFF, an overflow event occurs. At this point, the PWM output becomes high.
2. The ATR register value is loaded into the up-counter which in this example is 0xF00. 0xF00 is selected to maintain a range of 0-FF i.e. 0xF00-0xFFF for an 8-bit resolution *.wav* file.
3. The DCR lower 8-bit register (DCR0L) is loaded with a value that the microcontroller has read from the serial Flash. The DCR0H register value remains 0x0F. In this

example therefore, the microcontroller reads “1”. The complete contents of the duty cycle register is 0xF01.

4. The PWM output remains high until the point at which the up-counter contents do not match the DCR register contents which corresponds to the value from the serial Flash.
5. As the up-counter value matches the DCR register value, the PWM output goes low and remains low until the next overflow, i.e. until 0xFFF.

Note: To obtain a PWM signal, the DCR register value should be greater than the up counter starting value.

Figure 7. PWM function



ATR = Sampling Frequency (Divided /n from 1..x if output oversampled)

DCR=Duty Cycle= Sample value (coded on sample size Bit)

The resultant output signal of the microcontroller has a duty cycle corresponding to the 8-bit of quantization and 8kHz sampling frequency. So following that, the timer frequency will be 8 kHz * 256 = 2 MHz.

This means that 2MHz is the minimum PWM frequency needed for use of the PWM for audio generation. With the ST7Lite device, the timer can receive Fcpu=8MHz. To utilize the 8MHz timer frequency, the firmware puts 4 times the sample value on the PWM.

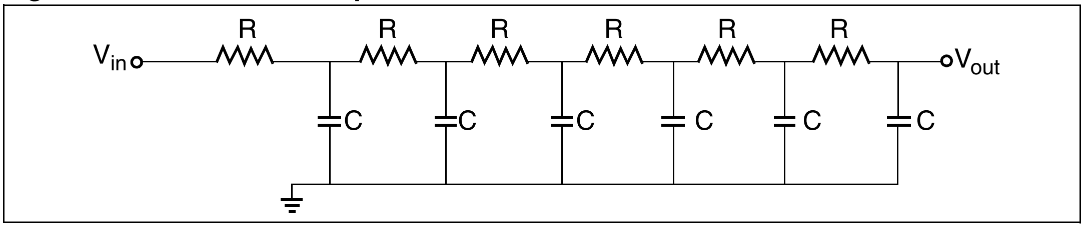
The 8 kHz sampling frequency has a very high pitch and is audible to the human ear, so the signal needs to be reconstructed, passing it through a 6th-order low pass filter. This removes any unwanted high frequency component from the signal. The signal is then passed to the amplifier and through to the speaker. The filter cut-off frequency is set to 8 kHz. If it is necessary to change the sampling frequency of the audio to be played, the cut-off frequency of the filter needs to be adjusted according to signal frequency range. The human ear can detect sounds between 20Hz to 20kHz. However, a range of 4 kHz to 8kHz is suitable for human voice, hence the 8 kHz is the minimum required sampling frequency.

The microcontroller runs on the internal RC oscillator. It is important to carefully calibrate the internal RC frequency, as any discrepancy in f_{CPU} value has a huge impact on the audio reproduction.

4 Filter design

Here, a 6th-order low pass filter is used. A low pass filter passes all the signals below the cutoff frequency and attenuates the signal above cutoff frequency. The reason behind using a sixth order filter is to have a steeper slope and therefore a better frequency response. It is created by cascading six first-order low pass filters as shown in figure below:

Figure 8. Sixth order low pass RC-filter



Calculating resistance of low pass filter:

The value of C is taken as 0.01 µf and the cutoff frequency f_C is 8kHz.

Cutoff frequency: $f_C = 1 / (2\pi RC)$

so: $R = 1 / (2\pi C f_C)$

$= 1 / (2\pi (.01\ \mu F)(8\ kHz))$

$= 1990\ \Omega$

$R = 1.9\ k\Omega$

1.9 kΩ is not a standard value, so 1.8 kΩ is used.

5 WAV file conversion process

(Utility provided to convert *.wav* -> *.bin*)

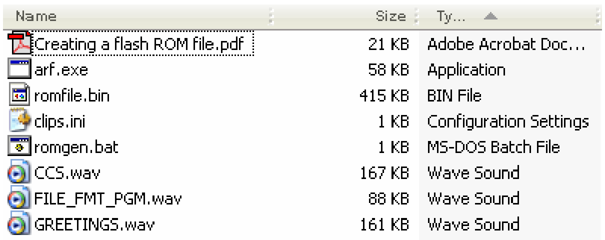
The key point of this demonstration board is to be able to customize it to play a set of sounds defined by the user. The user input to perform this operation will be the user *.wav* files. The *.wav* format is part of the RIFF bitstream format. This is one of the most current of audio formats for uncompressed PCM (Pulse Code Modulation) audio data as well as for computer audio storage. As it is required to be able to play multiple sounds in this demo, all audio data needs to be together in a file that can be programmed in the serial Flash. This is the purpose of the ARF utility.

The utility generates a *.bin* file such that:

- It can be used to flash the content of the M25P serial Flash chip
- It is in optimized format for the ST7FLiteS2 MCU to read and playback
- It can contain up to 15 separate sounds that can be accessed individually
- 1 sec of audio uses 64 KB of Flash.
  (With its 8 MB Flash, this board can playback up to 2 min of audio)

The folder where you have stored the utility should look similar to the following:

Figure 9. Files used to create a *.bin* file from a *.wav* file



1. Copy all *.wav* files into the same folder. (only 8 kHz/8-bit Files).
2. Copy *romgen.bat* and *arf.exe* into the same folder.
3. Edit *clips.ini* with names of sound clips, ensure that there are NO SPACES in the filenames.
4. Run *romgen.bat*.
5. Program the Serial Flash with the *romfile.bin* file.

Note: 1 A semicolon in front of a line in the INI file creates a comment line which is not processed by the ARF ROM generator utility.

2 The ARF utility program supports from 1 to 15 audio files, with a total file size supported up to the storage capacity of the 8-Mbit ST M25P80 Serial Flash ROM chip.

This process generates the *romfile.bin* file in the same folder with the following structure (as shown in *Table 1*). The microcontroller correctly interprets this structure while accessing the audio samples.

Table 1. Storage of data in the flash

Offsets	Description
00000000	Gives the number of audio files stored in the serial flash. 1st Byte contains the value 1-15. Next 3 bytes are unused.
00000004-00000007	Gives the first record size (LSB first)
00000008-0000000B	Gives the first record start address (LSB first)
0000000C-0000000F	Gives the second record size (LSB first)
00000010-00000013	Gives the second record start address (LSB first)

6 Conclusion

This solution provides an easy way to generate audio using an ST7 microcontroller. This can be customized to fit wide application requirements in term of audio. The solution provided can also be easily adapted to target a different microcontroller from the same family, or while using a different timer a wider range of platform options (ST7, STR7…).

Appendix A Files description

Table 2. Files and their description

Files	Description
Main.c	Contains the parameters, constants (#define), macros, global variables, function prototypes and their definition.
interrupt_vector.c	Contains the basic interrupt vector tables for ST7 devices and the preprocessors.
interrupt_routines.c	Contains the interrupt service routines and preprocessors.
interface.h	Contains the external declaration for the global variables.
interrupt_routines.h	Contains the external declaration for the interrupt service routines.
define.h	Contains the preprocessors and constants.
io7flite0.h	Contains IO definitions for the various registers of ST7FLITES2
lib_bits.h	Contains the constants (#define) and the public macros and declarations.
bitdef.h	Contains the constants (#define) for the various registers of ST7FLITES2.

A.1 Function description

A.1.1 RC_calibration()

Input Parameter	None
Output Parameter	None
Global Variables	None
Dependencies	None
Description	Used for the RC oscillator frequency adjustment. An 8-bit RC control register is used for the adjustment.

A.1.2 IO_Init()

Input Parameter	None
Output Parameter	None
Global Variables	None
Dependencies	None
Description	Used to configure the data direction register and option register in output mode and in push pull mode respectively.

A.1.3 SPI_Init()

Input Parameter	None
Output Parameter	None
Global Variables	None
Dependencies	None
Description	1) Configures the SPICSR and SPICR register. 2) Enables the microcontroller in master mode.

A.1.4 PWM_Init()

Input Parameter	None
Output Parameter	None
Global Variables	None
Dependencies	None
Description	Enables the overflow, input capture and compare interrupts. PWM channel 0 is also enabled.

A.1.5 LiteTimer_Init()

Input Parameter	None
Output Parameter	None
Global Variables	None
Dependencies	None
Description	Enables the timebase interrupt as soon as the counter overflow occurs. Thereby, time base flag is set by hardware.

A.1.6 GlobalVar_Init()

Input Parameter	None
Output Parameter	None
Global Variables	1) max_file_index 2) file_index 3) IButton_flag 4) PButton_flag 5) tcnt

Dependencies	None
Description	1) Determines the maximum file index. 2) Determines the current file index to be played. 3) Gives the distinction between an index button and a play button. 4) Sets a timer counter for blinking of all the LEDs when the soundboard is switched on.

A.1.7 LED_Display()

Input Parameter	None
Output Parameter	None
Global Variables	None
Dependencies	None
Description	Switches ON the LEDs in a binary format

A.1.8 Read_1_Byte()

Input Parameter	None
Output Parameter	None
Global Variables	fbuff [index]
Dependencies	None
Description	Reads the first byte of the flash, thereby establishing a communication path between microcontroller and flash device through SPI interfacing.

A.1.9 Check_Indexbutton()

Input Parameter	None
Output Parameter	None
Global Variables	IButton_flag
Dependencies	1) Key_Debounce() 2) SetNew_Index() 3) LED_Display()
Description	Checks whether the index button has been pressed or not. If pressed then switching over to next index file is done and thereby LED display also changes.

A.1.10 Check_Playbutton()

Input Parameter	None
Output Parameter	None
Global Variables	

Dependencies	Key_Debounce() Get_Current_Pointers ()
Description	Checks which file is to be played.

A.1.11 Key_Debounce()

Input Parameter	None
Output Parameter	None
Global Variables	tcnt
Dependencies	None
Description	Time delay for debouncing of switch

A.1.12 SetNew_Index()

Input Parameter	None
Output Parameter	None
Global Variables	1) IButton_flag 2) file_index 3) max_file_index
Dependencies	None
Description	Ensures that the index files do not exceed the required limit.

A.1.13 Get_Current_Pointers()

Input Parameter	None
Output Parameter	None
Global Variables	None
Dependencies	None
Description	It works in the following way : 1) selects the flash 2) reads the data from it 3) sends the data to microcontroller from the flash 4) reads and finds the start address of the data

A.1.14 Playbyte()

Input Parameter	None
Output Parameter	None
Global Variables	1) pwmrdy 2) pwm_val

Dependencies	Key_Debounce()
Description	Plays the index files by reading the data from the flash through SPI interface.

A.2 Interrupt routine description:

A.2.1 LTIC_ISR (Lite timer compare interrupt service routine)

Input Parameter	None
Output Parameter	None
Global Variables	Tcnt
Dependencies	none
Description	Decrements the counter from xx to 00. After decrementing, it clears the interrupt by reading the LTCSR register.

A.2.2 TO_ISR (Timer overflow interrupt service routine)

Input Parameter	None
Output Parameter	None
Global Variables	1) pwmrdy 2) pwm_val
Dependencies	none
Description	Whenever this interrupt occurs, the variable pwmrdy is incremented, and the PWM value is loaded in the duty cycle register.

A.2.3 TC_ISR

Input Parameter	None
Output Parameter	None
Global Variables	None
Dependencies	None
Description	Clears the interrupt generated by the PWM0 signal.

Appendix B Schematics

Figure 10. Schematic 1of 2

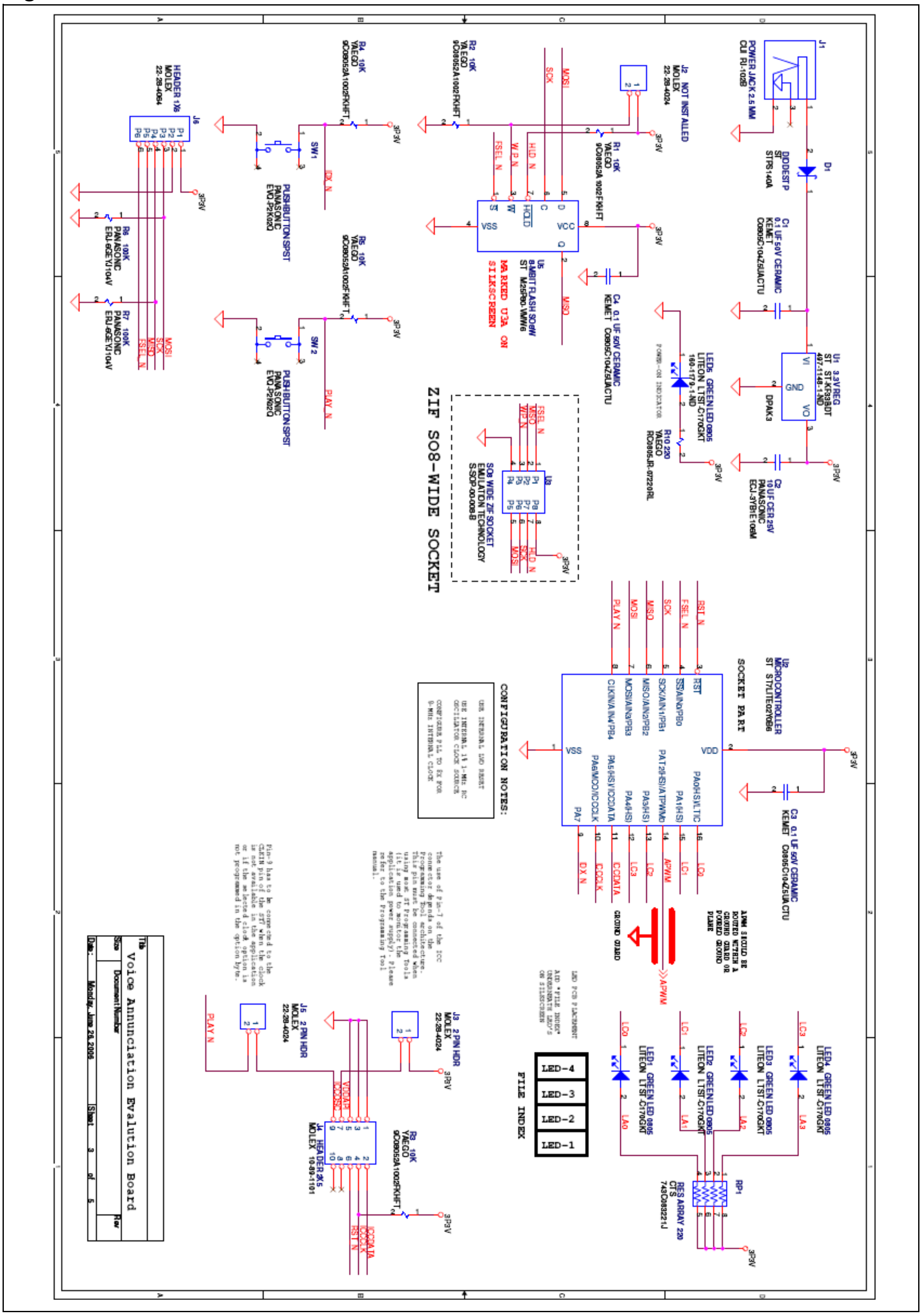
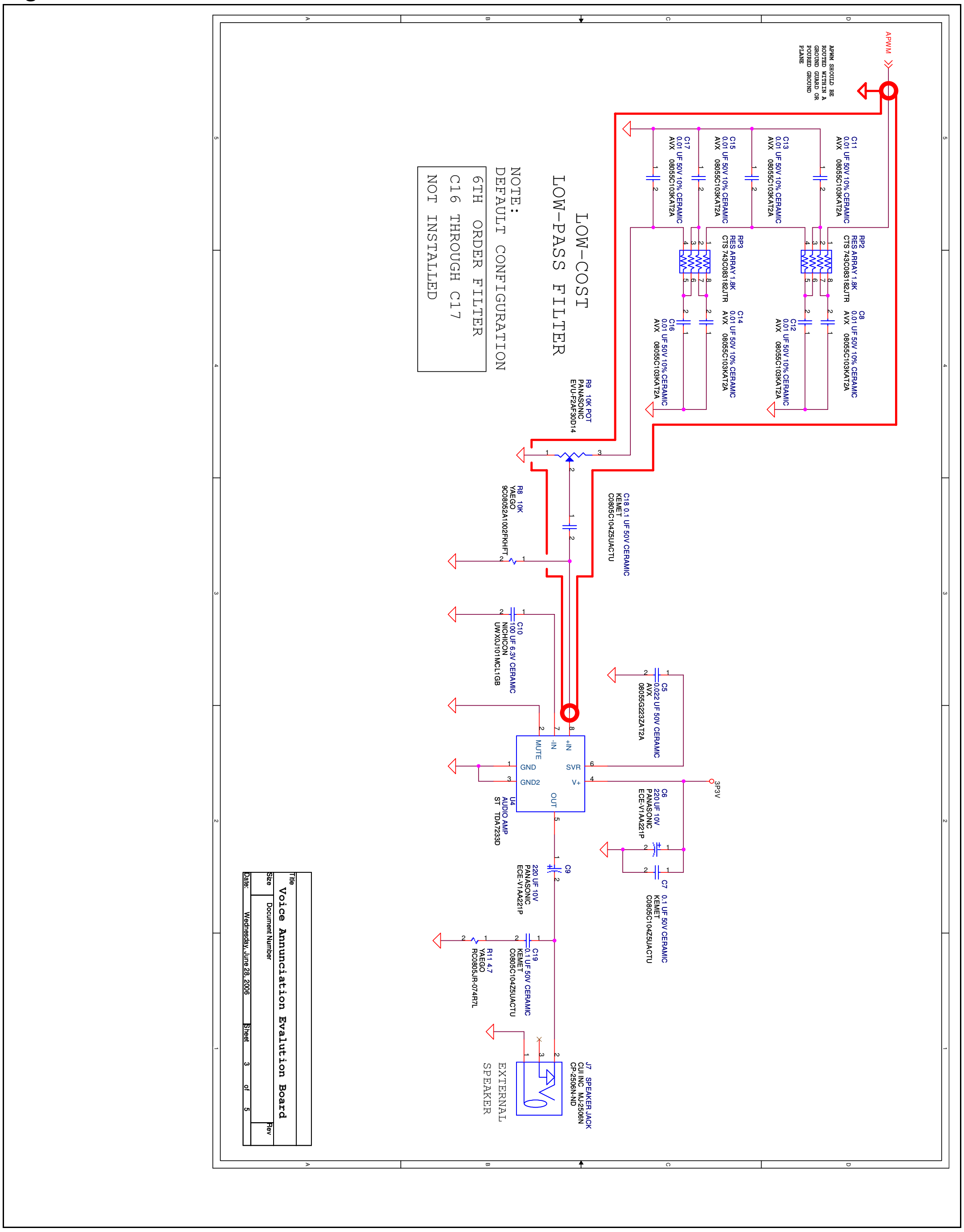


Figure 11. Schematic 2 of 2



Appendix C Bill of materials

Table 3. Bill of materials for Audio player evaluation board

Index	Quantity	Reference	Value / Generic Part Number	Package	Manufacturer	Manufacturer ordering code/ Orderable Part Number
1	6	C1, C3, C4, C7, C18, C19	0.1 UF 50V CERAMIC	SMD0805	KEMET	C0805C104Z5 UACTU
2	1	C2	10 UF CER 25V	1206	PANASONIC	ECJ-3YB1E106M
3	1	C5	0.022 UF 50V CERAMIC	SMD0805	AVX	08055G223ZA T2A
4	2	C9,C6	220 UF 10V	8 * 6.2 mm	PANASONIC	ECE-V1AA221P
5	8	C8, C11, C12, C13, C14, C15, C16, C17	0.01 UF 50V 10% CERAMIC	SMD0805	AVX	08055C103KA T2A
6	1	C10	100 UF 6.3V CERAMIC	6.3 *5.4 mm	NICHICON	UWX0J101MC L1GB
7	1	D1	DIODESTP	SMA	ST	STPS140A
8	1	J1	POWER JACK 2.5MM	POWER JACK-14.17 * 8.96 mm	CUI	PJ-102B
9	1	J2	NOT INSTALLED	NOT INSTALLED	MOLEX	22-28-4024
10	2	J5, J3	2 PIN HDR	2 PIN HDR	MOLEX	22-28-4024
11	1	J4	HEADER 2X5	HEADER 2X5	MOLEX	10-89-1101
12	1	J6	HEADER 1X6	HEADER 1X6	MOLEX	22-28-4064
13	1	J7	SPEAKER JACK	SPEAKER JACK	CUI INC	MJ-2506N
14	5	LED1, LED2, LED3, LED4, LED5	GREEN LED	SMD0805	LITEON	LTST-C170GKT
15	1	RP1	RES ARRAY 220	0805	CTS	743C083221J
16	2	RP2, RP3	RES ARRAY 1.8K	0805	CTS	743C083182J TR
17	6	R1, R2, R3, R4, R5, R8	10K	SMD0805	YAEGO	9C08052A100 2FKHFT
18	2	R6, R7	100K	SMD	PANASONIC	ERJ-6GEYJ104V

Index	Quantity	Reference	Value / Generic Part Number	Package	Manufacturer	Manufacturer ordering code/ Orderable Part Number
19	1	R9	10K POT	11.48 * 9.71 mm	PANASONIC	EVU-F2AF30D14
20	1	R10	220E	SMD0805	YAEGO	RC0805JR-07220RL
21	1	R11	4.7E	SMD0805	YAEGO	RC0805JR-074R7L
22	2	SW1, SW2	PUSH BUTTON SPST	4.7X3.5X2.1 mm	PANASONIC	EVQ-P2K02Q
23	1	U1	3.3 VREG	DPAK	ST	KF33BDT
24	1	U2	MCU	DIP16	ST	ST7FLITES2Y0B6
25	1	U3	WIDE ZIF SOCKET	SO8	EMULATION TECHNOLOGY	S-SOP-00-008-B
26	1	U4	AUDIO AMP	SO8	ST	TDA7233D
27	1	U5	8-MBIT FLASH	SO8W	ST	M25P80-VMW6G

7 Revision history

Table 4. Document revision history

Date	Revision	Changes
10-Nov-2006	1	Initial release.