TI Designs
Voice Band Audio Playback Using a PWM DAC Design Guide

TI Designs

TI Designs provide the foundation that you need including methodology, testing, and design files to quickly evaluate and customize the system. TI Designs help *you* accelerate your time to market.

Design Resources

TIDM-VOICEBANDAUDIO	TI Design Files
MSP-EXP430F5529LP	Product Folder
TPA301	Product Folder

TI E2E™ Community

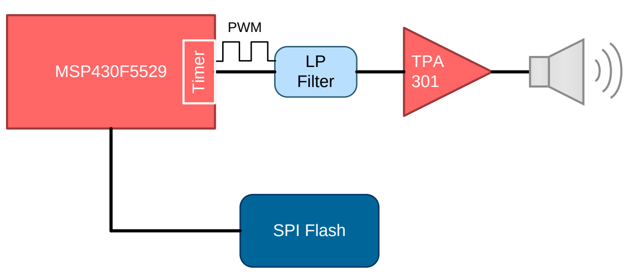
ASK Our E2E Experts
WEBENCH® Calculator Tools

Design Features

- Easily Integrate Code into Existing Systems (Greater Than 2-Kb Flash Required)
- BoosterPack™ Design for Easy LaunchPad™ Use
- 8-Bit PWM DAC playback with low-cost filtering
- Multiple Playback Frequencies Supported
- External SPI Flash Storage for Sound Files
- SPI Flash Loading Software Included
- Headphone Jack or Speaker Output With Volume Control

Featured Applications

- Portable Medical Products
- Assisted Living Products
- Annunciator Panels and Sounders
- Toys and Educational Products



An IMPORTANT NOTICE at the end of this TI reference design addresses authorized use, intellectual property matters and other important disclaimers and information.

1 System Description

This design uses existing modules within the MSP430F5529 to easily add 8-bit voice band audio to larger systems with very few additional external components. Sound is produced by using a timer module to produce a PWM signal, which is passed through an external filtering and amplification circuit. The sound samples are stored in an external SPI flash.

1.1 MSP430F5529 LaunchPad™

The MSP430F5529 LaunchPad provides an easy-form factor to highlight this design. The LaunchPad offers a standard pinout for BoosterPack use that allows for easy prototyping to include this design into existing or future systems. The LaunchPad also provides a great platform to load sound files into the external SPI flash before playback. The MSP430F5529 itself provides a robust communication module to converse with the SPI flash, while providing a flexible clock system that can achieve a range of playback frequencies for this design. This design was originally built on revision 1.4 and later tested on revision 1.6 of the MSP430F5529LP. The version currently available on the TI eStore may differ in revision, and users should check for any possible compatibility issues. The LaunchPad user guide includes a revision history that can help the user reconcile the software for the new hardware revisions.

Figure 1 shows the MAP430F5529LP used for this design; however, code is provided for use on the MSP430FR5969LP and MSP-EXP430G2 (MSP430G2553 LaunchPad). Users can use additional standalone MSP430s after making changes in the provided hardware abstraction layer. The user is responsible for making these changes. See the MSP430™ minimum requirements in Section 6.2.5.

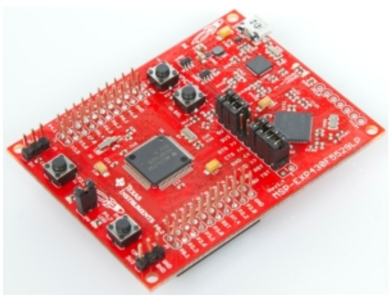


Figure 1. MSP430F5529LP Board Photo

1.2 Voice Band Audio BoosterPack™

The Voice Band Audio BoosterPack (VBA BoosterPack) conforms to the LaunchPad ecosystem pinout and provides alternate pinouts to used pins to increase compatibility with other BoosterPack designs. The VBA BoosterPack contains the design's SPI Flash, simple low-pass filter, and gain amplifier. The TPA301 on the board was chosen for its low power, low cost, and great THD + N performance in the voice-band frequency range. The VBA BoosterPack also contains additional hardware outside the scope of the TI Design including a microphone circuit, an analog switch for switching between speaker or headphone output, and an inline potentiometer for the gain amplifier to adjust volume. The VBA BoosterPack is not currently available on the TI eStore, but all design files for the hardware are included on the TI Design webpage, so the user can make the VBA Boosterpack can be made at his or her own discretion.

Figure 2. Voice Band Audio BoosterPack

2 Block Diagram

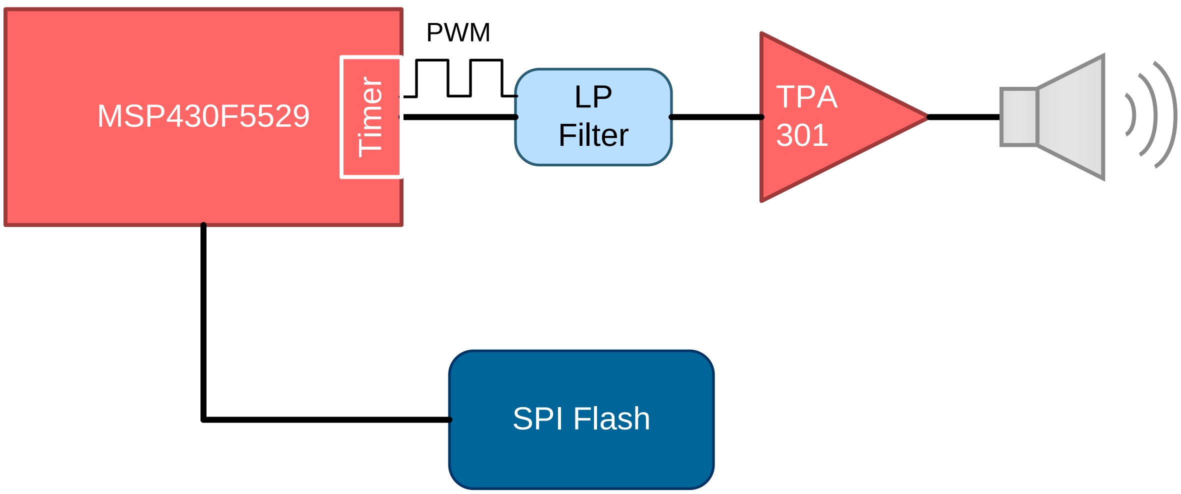


Figure 3. Voice Band Audio Block Diagram

2.1 Highlighted Products

2.1.1 MSP430F5529 LaunchPad

- USB 2.0-enabled MSP430F5529 16-bit MCU
- Up to 25 MHz
- 128-KB Flash and 8-KB RAM
- 12-Bit SAR ADC
- Various USB device class examples and libraries available (CDC, HID, MSC)
- eZ-FET lite: Open source onboard debugger with application UART
- One USB connection for emulator and target via the use of an onboard USB hub
- USB as power source: 5 V and 3.3 V through a high-efficiency DC/DC converter
- 40-pin LaunchPad standard leveraging the BoosterPack ecosystem
- Availability in TI eStore

2.1.2 TPA301—Mono Class A-B Audio Amplifier

- Fully Specified for 3.3-V and 5-V Operation
- Wide Power Supply Compatibility 2.5 V–5.5 V
- Output Power for RL = 8 Ω
  - 350 mW at VDD = 5 V, BTL
  - 250 mW at VDD = 3.3 V, BTL
- Ultralow-Quiescent Current in Shutdown Mode: 0.15 µA
- Thermal and Short-Circuit Protection
- Surface-Mount Packaging
  - SOIC
  - PowerPAD™ MSOP

3 System Design Theory

The main purpose of this design is to produce audio through a PWM DAC created by the MSP430F5529's internal Timer Module and external filters. An audio waveform is produced by smoothing the square wave edges created by the PWM signal. Figure 4 shows the basic premise of this operation. By varying the width of the square-wave, and then feeding this into a low-pass filter, we can create varying analog signal levels. This process is the basic premise of a PWM DAC. By setting the period of the PWM to the playback frequency of a sound file, the raw data can be interpreted as the pulse widths for our PWM output. This guide is a high-level overview of how a PWM DAC works; for more information about PWM DACs, visit the TI application notes for *PWM DAC Using MSP430™ High-Resolution Timer* (SLAA497) and *Using PWM Timer_B as a DAC* (SLAA116).

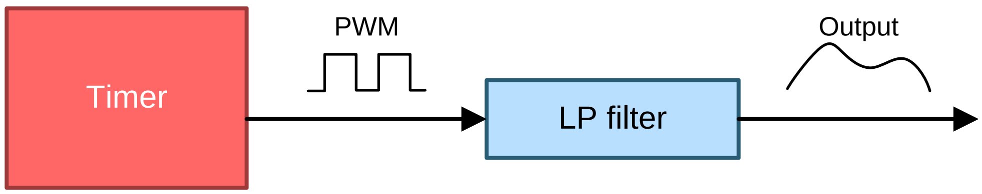


Figure 4. Basic PWM DAC Operation

3.1 Filtering

The filtering used in this design is targeted to be simple and low cost. Audio quality could further improve by implementing additional or more advanced filter stages. The Voice Band has been historically defined by the telephone industry and includes an upper limit of 4 kHz, which allows a minimum sampling frequency of 8 kHz per the Nyquist-Shannon sampling theorem. The low-pass filter on the BoosterPack is set with a cutoff frequency of approximately 3.7 kHz to accommodate the voice band. The value of 3.7 kHz was chosen instead of 4 kHz due to the slow roll-off of a first-order passive low-pass filter. The audio amplifier circuit also acts as a first-order active low-pass filter of 20 kHz. This value is commonly referred to as the upper limit of human hearing. This filtering scheme means that signals between 4 kHz–20 kHz will be attenuated, while frequencies beyond 20 kHz are further reduced by the second LP filter at 20 kHz. The additional attenuation helps remove high-frequency noise introduced by the PWM. Both filters are shown in the Schematics portion of this document. For more information about the active filter configuration associated with the TPA301, please see the *Application Information* section of the TPA301 datasheet. Also available from TI to help develop filter configurations is TI-TINA, a SPICE-Based Analog Simulation Program.

3.2 Timer Frequency Versus Bit Rate

The three components that are pivotal to making an audio signal from a Timer PWM DAC are playback frequency ($F_{playback}$), timer frequency (F_{timer}), and the desired DAC resolution. The relationship between these values can be expressed in the following equation:

$F_{timer} = F_{playback} \times 2^n$ where:

F_{timer} is the required PWM timer frequency

$F_{playback}$ is the PWM signal frequency, which is the DAC update rate

n is the desired resolution of the DAC in bits (1)

The result of (2^n – 1) is the value that will go into the base timer counter register CCR0. The sound sample data will go into the Timer CCRx register for the PWM that we are outputting. For this design the DAC resolution is fixed to 8-bit so a value of 0xFF (2^8 – 1) is loaded into CCR0. Table 1 shows common playback frequencies with the corresponding timer frequencies needed for a given resolution. Entries with an asterisk (*) are combinations that are currently implemented in the provided software. The MSP430F5529 has a maximum timer frequency of 25 MHz, which limits what playback frequency and resolution it can produce. Other devices have different playback frequency limits due to their maximum timer frequency limits. As seen in Table 1, by limiting the resolution to eight bits, we can support the largest range of common playback frequencies for the MSP430F5529.

Table 1. Common Playback, Timer Frequencies, and Resolutions

PLAYBACK FREQUENCY (Hz)	RESOLUTION (BITS)	TIMER FREQUENCY NEEDED (MHz)
8000*	8	2.04
8000	10	8.18
8000	12	32.77
8000	16	524.29
12000	8	3.072
12000	16	786.43
16000*	8	4.08
16000	10	16.38
16000	16	1048.58 (~ 1 GHz)
22050*	8	5.64
44100*	8	11.29
48000	8	12.29

* These entries are currently implemented in the provided software

3.3 Notes on File Lengths

File lengths of the sound sample vary depending on playback frequency, resolution, and duration. This design includes a 2-MB external SPI flash to store sound samples. If sound samples are small enough, they could be placed into the flash on the MSP430. Users can estimate the size of a sound file by knowing the duration, playback frequency, and resolution. Playback frequency and resolution are often expressed with a single term called a bit rate. For example, assuming an 8-bit resolution and 22-kHz playback frequency, users would have a bit rate of 176 kbps (kilobits per second). This example shows that a sample with a length of one second would take up 176 Kb (kilobits) or 22 KB (kilobytes) of space.

4 VBA BoosterPack

The Voice Band Audio BoosterPack is shown in Figure 5. The VBA Boosterpack conforms to the BoosterPack ecosystem standard and additionally offers alternative pins to accommodate conflicts on stacked BoosterPacks. The BoosterPack contains the filter circuitry, audio amplifier, and 2-MB flash. The BoosterPack also features optional audio output to a speaker or headphone jack, a potentiometer for volume (gain) adjustment, and a microphone circuit, which is not used for this TI-Design.

4.1 Pin Configuration and Alternate Pins

The pin configuration for the VBA BoosterPack is outlined in Table 2 and is labeled on the hardware itself. Alternative pins are marked with an asterisk (*) on the board, must be configured in software, and require a change in position for its corresponding zero-Ω resistor to be used.

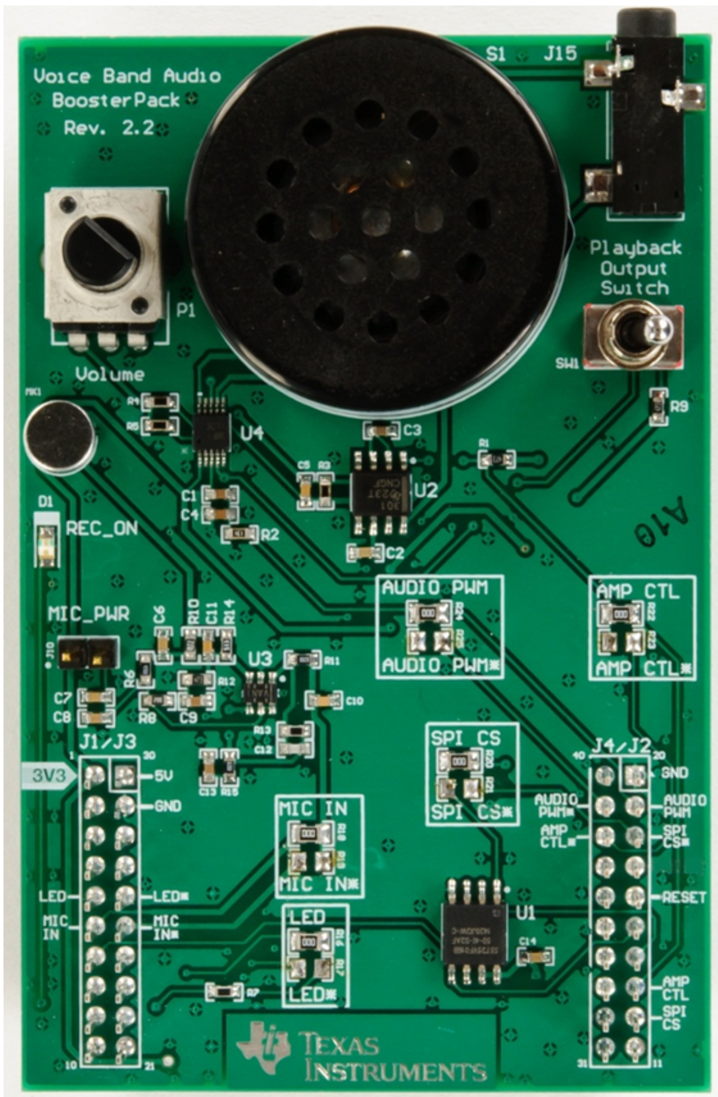


Figure 5. VBA Booster Pack Front View

Table 2. Pin Configurations of the VBA Booster Pack

FUNCTION	BOARD LABEL	LAUNCHPAD HEADER PIN	MSP430F5529 PIN
SPI Clock	NA	J1.7	P3.2
SPI MOSI	NA	J2.15	P3.0
SPI MISO	NA	J2.14	P3.1
SPI CS	SPI CS	J2.12	P2.3
SPI CS Alternative	SPI CS*	J1.7	P2.2
Timer PWM	AUDIO PWM	J2.19	P2.0
Timer PWM Alternative	AUDIO PWM*	J4.39	P2.4
Audio Amp Control	AMP CTL	J2.13	P2.6
Audio Amp Control Alternative	AMP CTL*	J4.36	P1.3
LED	LED	J1.5	P1.6
LED Alternative	LED*	J3.25	P6.2
Microphone Input	MIC IN	J1.6	P6.6
Microphone Input Alternative	MIC IN*	J3.26	P6.3

4.2 Feature List

- LaunchPad and BoosterPack ecosystem header configuration
- 2-MByte SPI Flash
- Low Pass Filter Circuitry
- Audio Amplifier
- Alternate Pin Configurations
- Additional Features
  - Speaker or Headphone Jack output with switch between the two
  - Series Potentiometer for Gain / Volume Adjustment
  - Microphone Circuitry for Future Designs
  - LED

4.3 Additional Features

Following this section are additional features of the VBA BoosterPack that are not a main feature of this TI Design, but are included for user convenience or future designs.

4.3.1 Speaker or Headphone Jack Output

On the VBA BoosterPack, users have the option of choosing an onboard speaker output or headphone jack to be connected to any standard headphones. The mechanical switch located on the top right of the board controls an analog IC switch (TS3A24159). This analog switch changes the signal path of the Audio Signal to output either via speaker or headphone jack output. The appropriate gain resistors for each output are also switched to match the corresponding output.

NOTE: While using headphones, please check output volume before placing close to ears. Volume output could be extremely loud at startup.

4.3.2 Inline Potentiometer for Gain Adjustment

The potentiometer located at the top left of the board controls the gain on the audio amplifier, effectively controlling volume. This 10-kΩ potentiometer is in series with the gain resistor. As resistance is increased, the gain is reduced (volume down), and as resistance decreases, the gain increases (volume up). The potentiometer is set up so that a counter-clockwise turn will increase volume, and a clockwise turn will decrease volume.

4.3.3 Microphone Circuit

The microphone circuit contained on the board is not used in this TI-Design, but was placed on the board for future projects and designs in mind. To use this circuit, the jumper (J10) labeled MIC_PWR must be shorted. Additional software not provided will also need to be created to sample the microphone.

5 Getting Started—Loading Sound Files

This section details how a user can load sound files into the external SPI flash using the provided GUI and passthrough code. A few example sound files are included with the software provided with this design.

5.1 Expected File Format

Sounds files must be in a .wav file, 8-bit, mono, uncompressed (PCM) format for this design. Any other file formats or compressions must be handled by the user and is outside the scope of this design. Users can find free conversion programs online to format sound files in the correct manner. Please be aware that converting between different playback frequencies and resolution may introduce forms of aliasing and noise, which may reduce audio quality of the sound sample. As discussed in Section 3.1, the hardware filters of this design are set at 3.7 kHz and 20 kHz.

5.2 LaunchPad Passthrough Code

The passthrough code is a separate project from the playback project and is a utility program for loading sound files in conjunction with the provided Audio Loading GUI. The project has been developed and tested in CCSv6 only. Additional porting may be needed to work with other IDEs. The project must be built and programmed into the MSP430F5529 device before starting the Audio Loading GUI. No user modification is required for these files. The passthrough code sets up a 9600-Baud UART connection and passes data from the PC into the SPI flash at the locations provided by the user through the GUI. The red LED on the BoosterPack lights up (standard configuration) while a passthrough transaction is in progress, and turns off when it completes.

5.3 Audio Loading GUI

This section gives a step-by-step guide on how to load sound files using the provided GUI. The GUI should be started after connecting the LaunchPad device and loading the passthrough code. The GUI was built with Java v7 update 71.

5.3.1 Connecting to the LaunchPad

After starting the GUI, users must choose the COM Port that contains the backchannel UART on the LaunchPad from the drop-down menu. The number of the COM Port will vary. Please check the device manager on your PC for the correct COM Port (Figure 6). If the correct COM PORT is not displayed, please check physical connections and hit the Refresh button on the GUI. See Figure 7 and Figure 8.

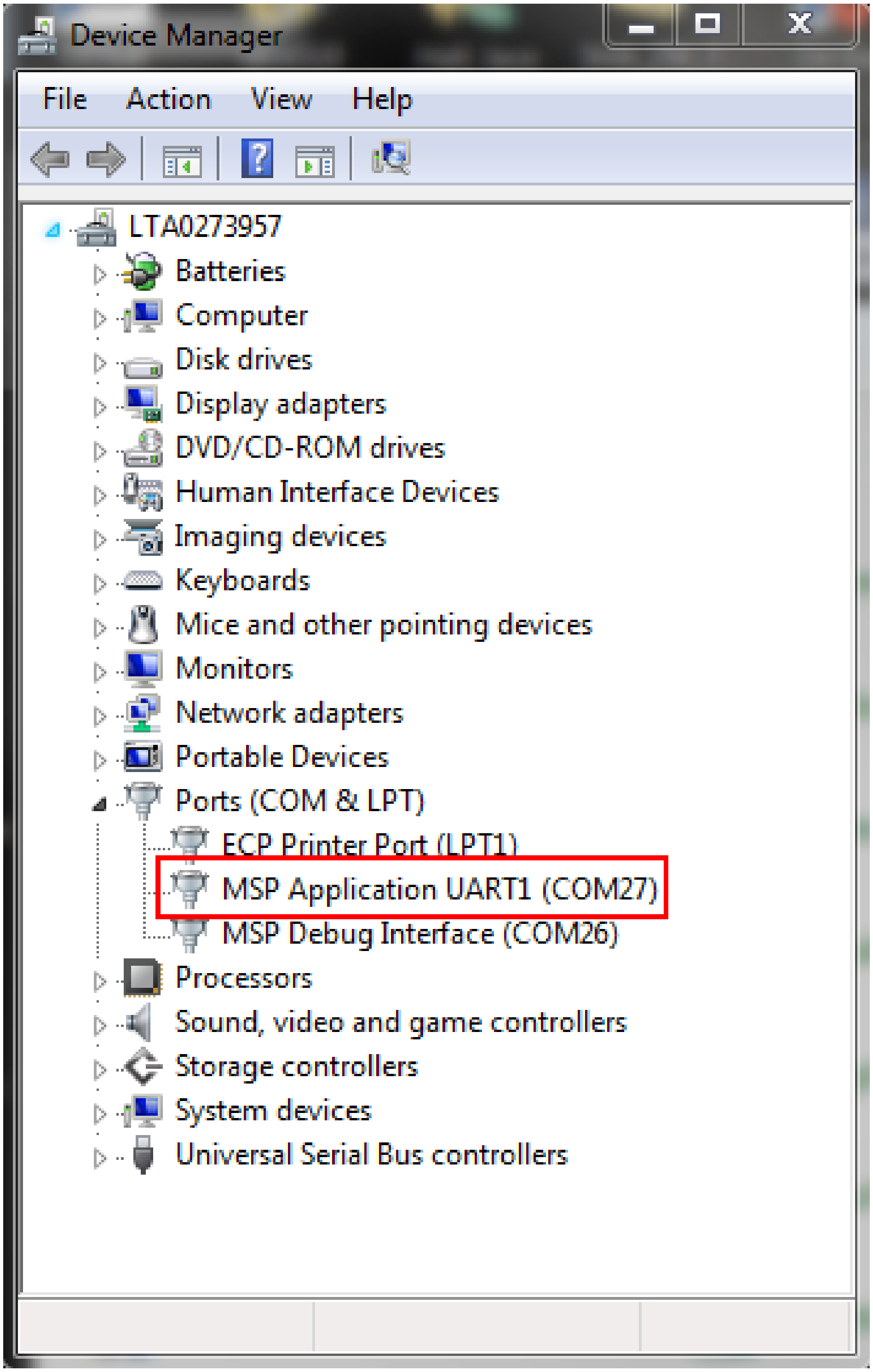


Figure 6. Identifying Application COM Port

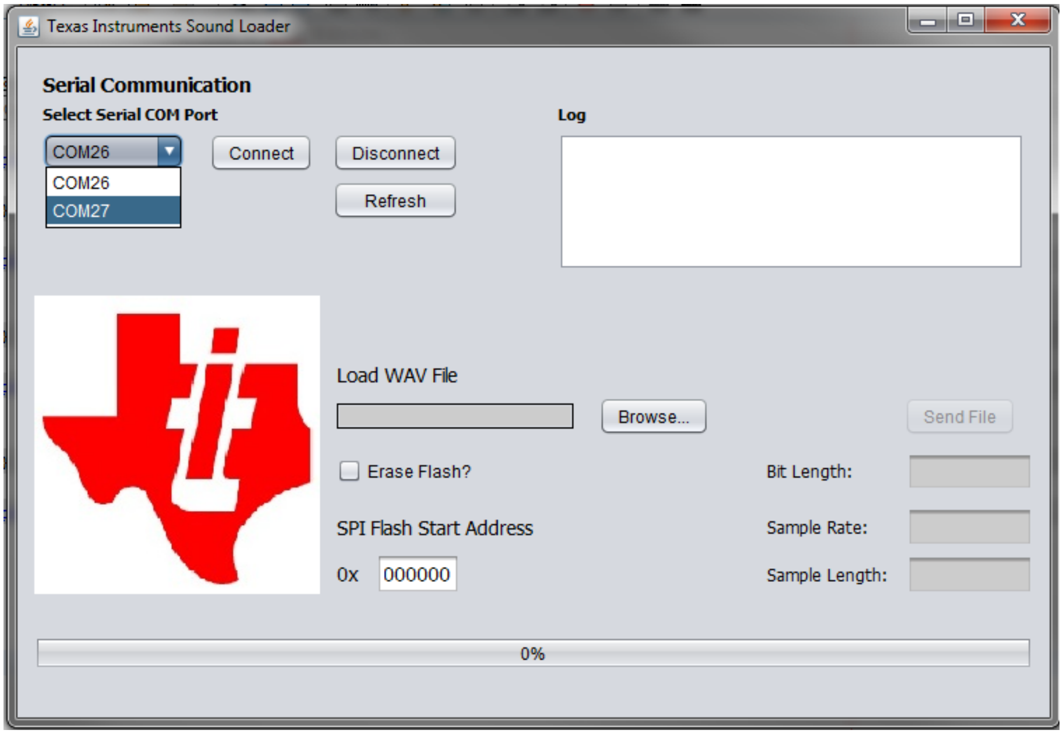


Figure 7. Selecting a COM Port

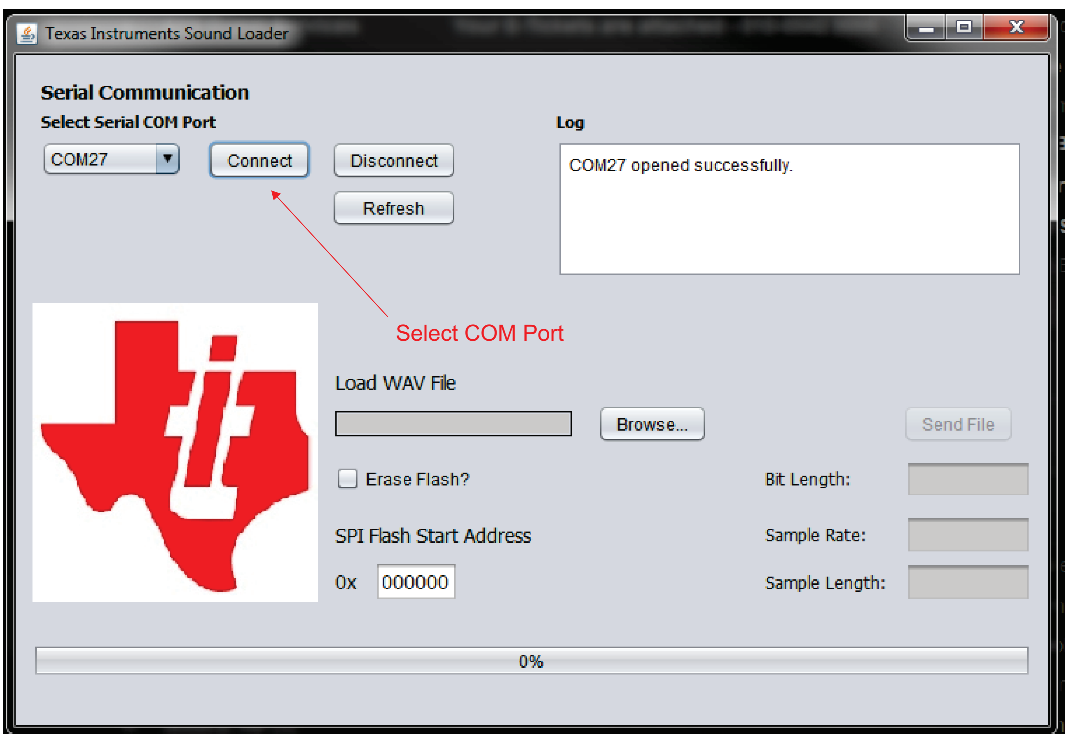


Figure 8. Connecting a COM Port

5.3.2 Selecting a Sound File

Click the Browse button to open another window for finding sound files. Only .wav files can be selected. Once a valid file is chosen, information about the file is shown on the right side of the GUI. See Figure 9 and Figure 10 for selecting a valid file. Also, if a sound file that is not in an 8-bit format is selected, the Bit Length field will show an error and the Send File button will stay inactive, as shown in Figure 11.

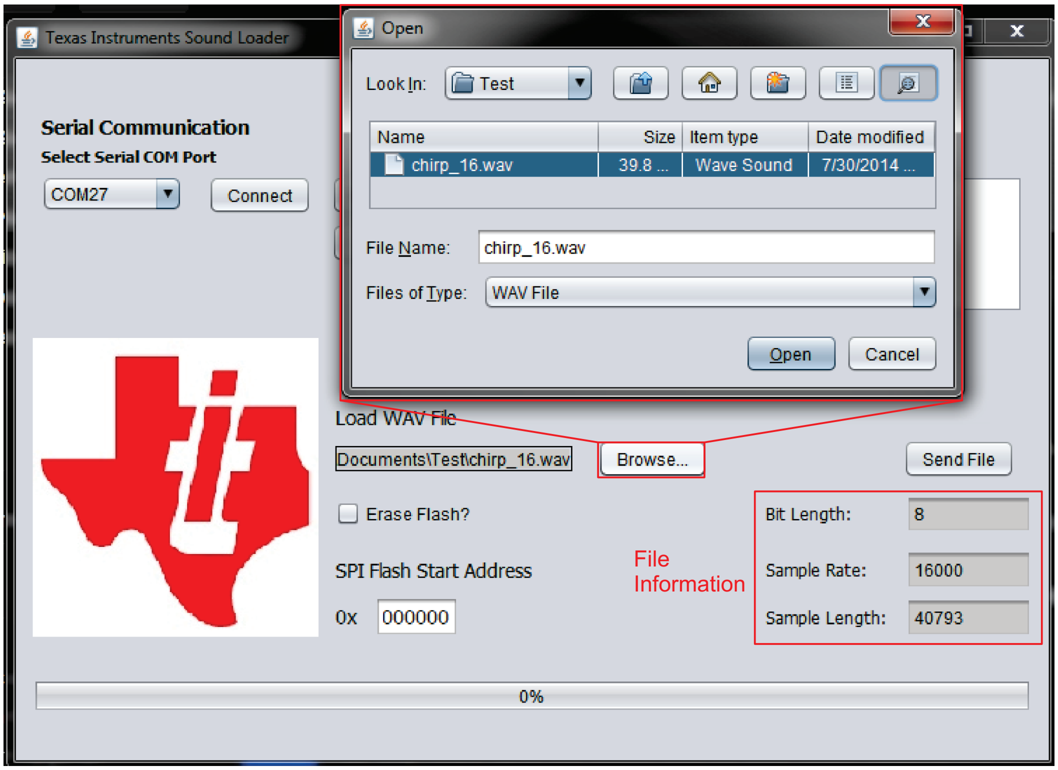


Figure 9. Choosing a File

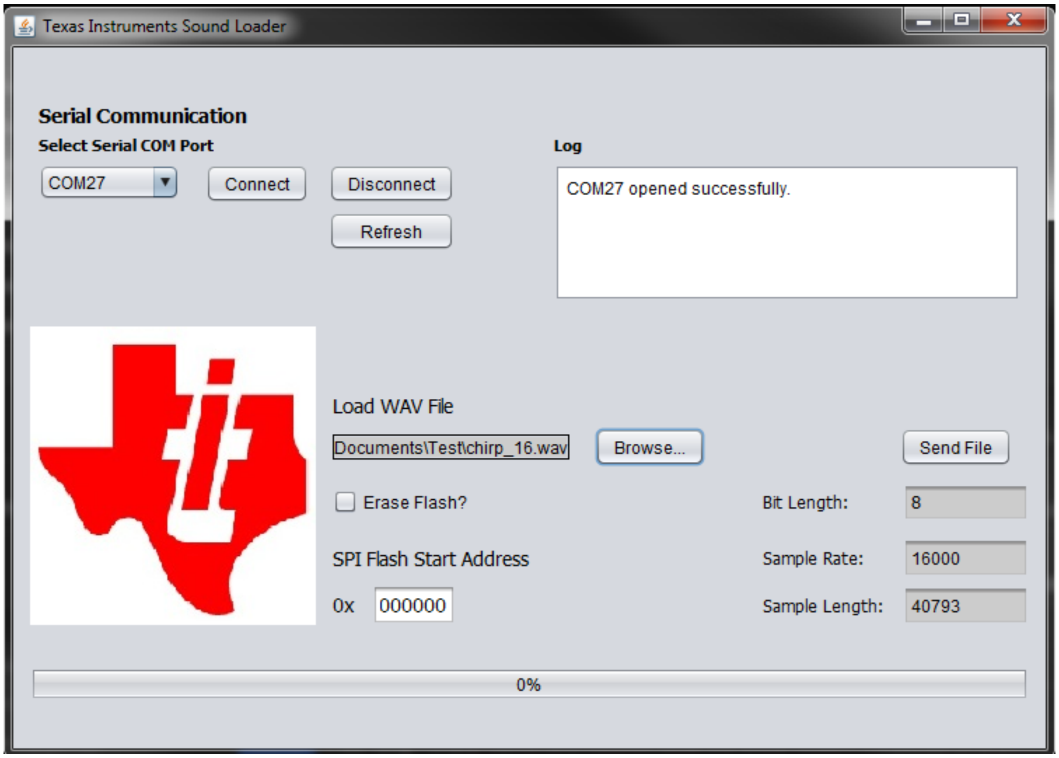


Figure 10. Valid File Selected

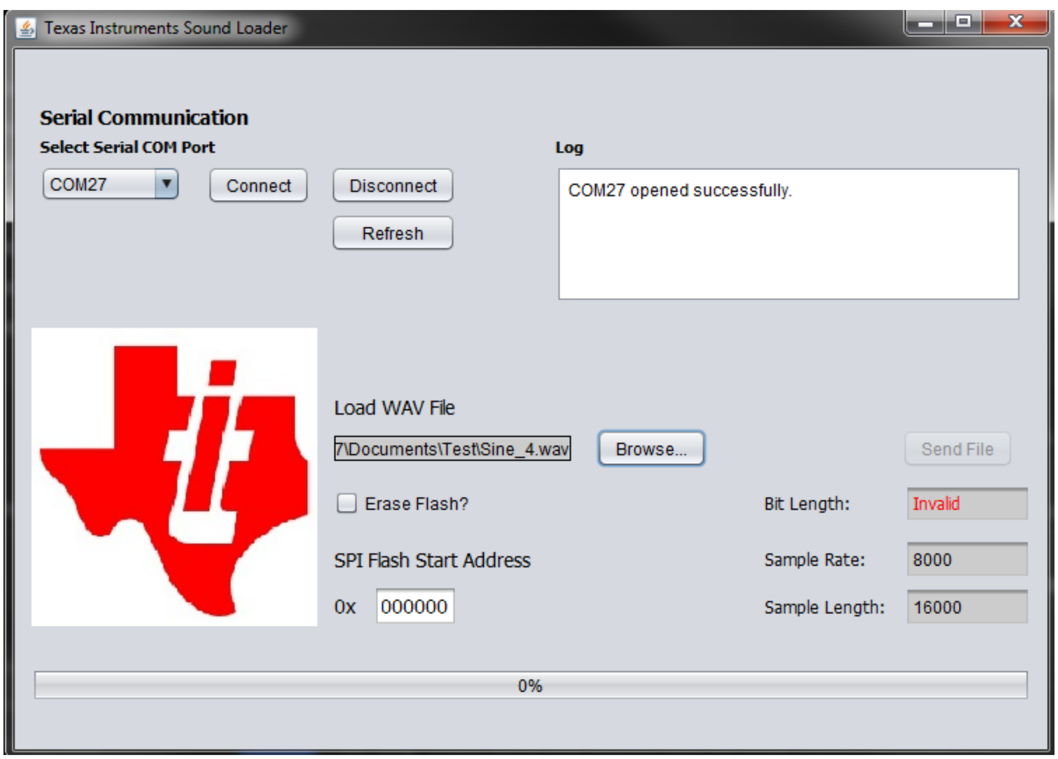


Figure 11. Choosing an Invalid File

5.3.3 Sending a Sound File

Before sending the first sound file, the user must choose to erase the flash through the checkbox. This feature can only be used on the first sound file that the passthrough code on the LaunchPad receives. Also, users must fill in the hex location of the start address where the sound file will be loaded before sending a file or the default location of 0x000000 will be used. See Figure 12 for an example of this process. After users click the Send File button, the progress bar updates with the status of the transfer. To send multiple files, after a transfer is complete, users must choose a new file through the browse button and modify the hex start address.

NOTE: The passthrough code must be running on the MSP430 LaunchPad before users press the Send File button on the GUI

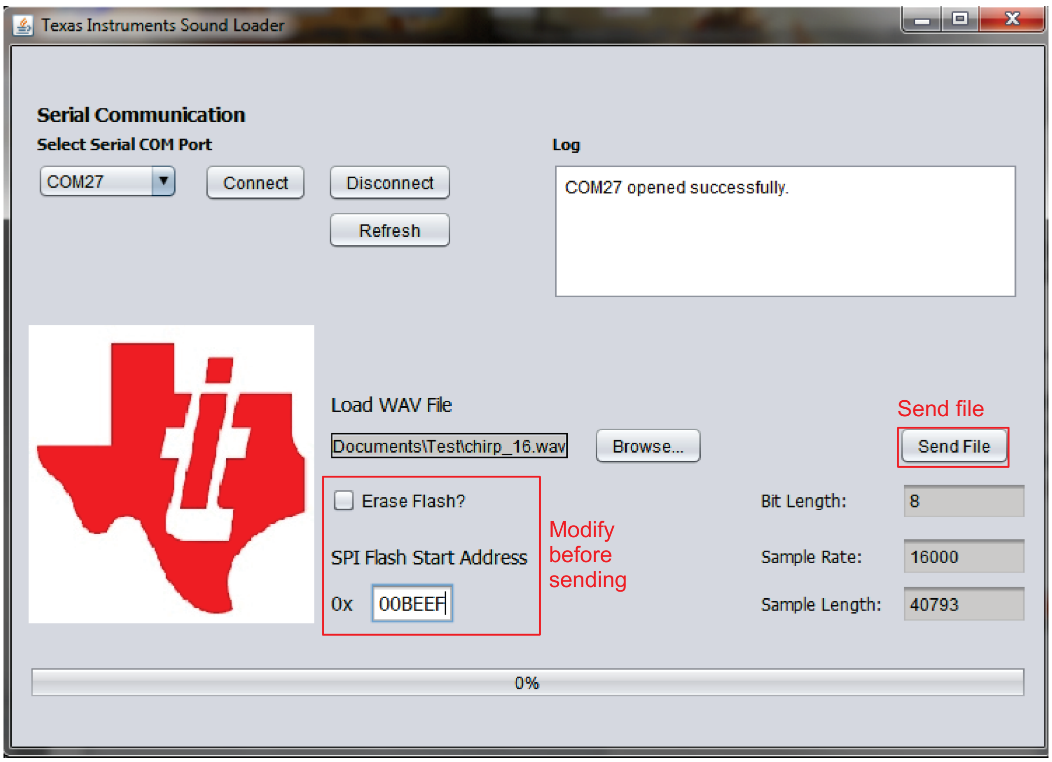


Figure 12. Sending a File

5.4 Memory Organization

During development, a developer may want to organize the locations of sound files in the external SPI flash. This design does not offer any kind of file or organization system and this has been left to the developer to decide. A chart or graphic of such a system during development can help in preventing mistakes in overwriting files, exceeding flash limits, or giving the developer a handy guide to where files are located. Figure 13 shows an example of such a guide alongside corresponding playback code. This code is further discussed in Section 6.2.

```
//***Playback Frequency Select  -- Changes clock settings
// Frequency Options
// FREQUENCY_8kHz
// FREQUENCY_16kHz
// FREQUENCY_22kHz  *Only avilable on MSP430F5529LP
// FREQUENCY_44kHz  *Only avilable on MSP430F5529LP
//***
#define FREQUENCY_SELECT FREQUENCY_44kHz

//*** Sound Sample Lengths defines
//    Recommend length variable to be smaller than actual sample length
//    in order to cut out potential noise at sample end.
//***
const uint32_t soundDataLength1 = 38300; // Length in Hex : 0x959C
const uint32_t soundDataLength2 = 50258; // Length in Hex : 0xC452
const uint32_t soundDataLength3 = 54067; // Length in Hex : 0xD333
const uint32_t soundDataLength4 = 36289; // Length in Hex : 0x8DC1

//*** Start addresses in SPI Flash defines
//***
const uint32_t soundDataStartAddress1 = 0x00000; // Hello
const uint32_t soundDataStartAddress2 = 0x20000; // System is ready
const uint32_t soundDataStartAddress3 = 0x40000; // 5 4 3 2 1
const uint32_t soundDataStartAddress4 = 0x60000; // Goodbye
```

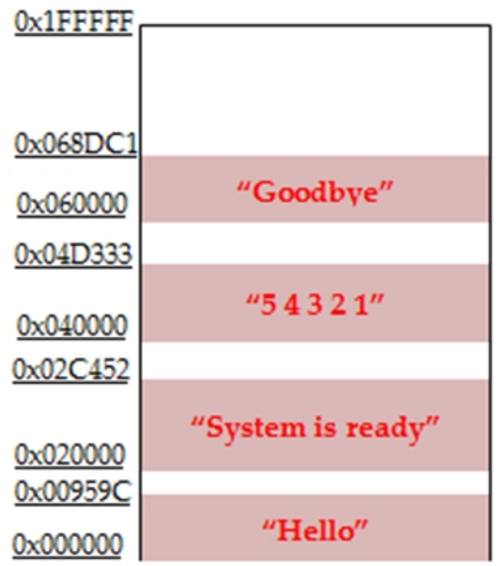


Figure 13. Memory Organization Alongside Setup Code Snippet

6 Playback Software

The playback software consists of three main parts including the main loop, the playback function, and timer interrupt. Each part takes advantage of the MSP430's Low-Power Modes (LPMs) for program flow and to reduce power consumption. SPI transactions occur via a polling method to ensure timely servicing of timer interrupt. Choosing which samples to play occurs in the Button Press Interrupt and currently switches between the three samples with one button, and only the first two with the other. All of the playback software is contained in the code files Playback_main.c, Playback_HAL.h, and Playback_UserDefine.h. The playback software takes up approximately 1.7-k flash space depending on users' optimization settings and trigger options. The code has been developed and tested in CCSv6 only. Additional porting may be needed to work with other IDEs.

NOTE: Playback Frequency is chosen before compilation.

Interrupts beyond the associated playback timer, should be disabled during playback to ensure consistent sound playback.

6.1 Playback Flowchart

The following is the program flow for the playback software.

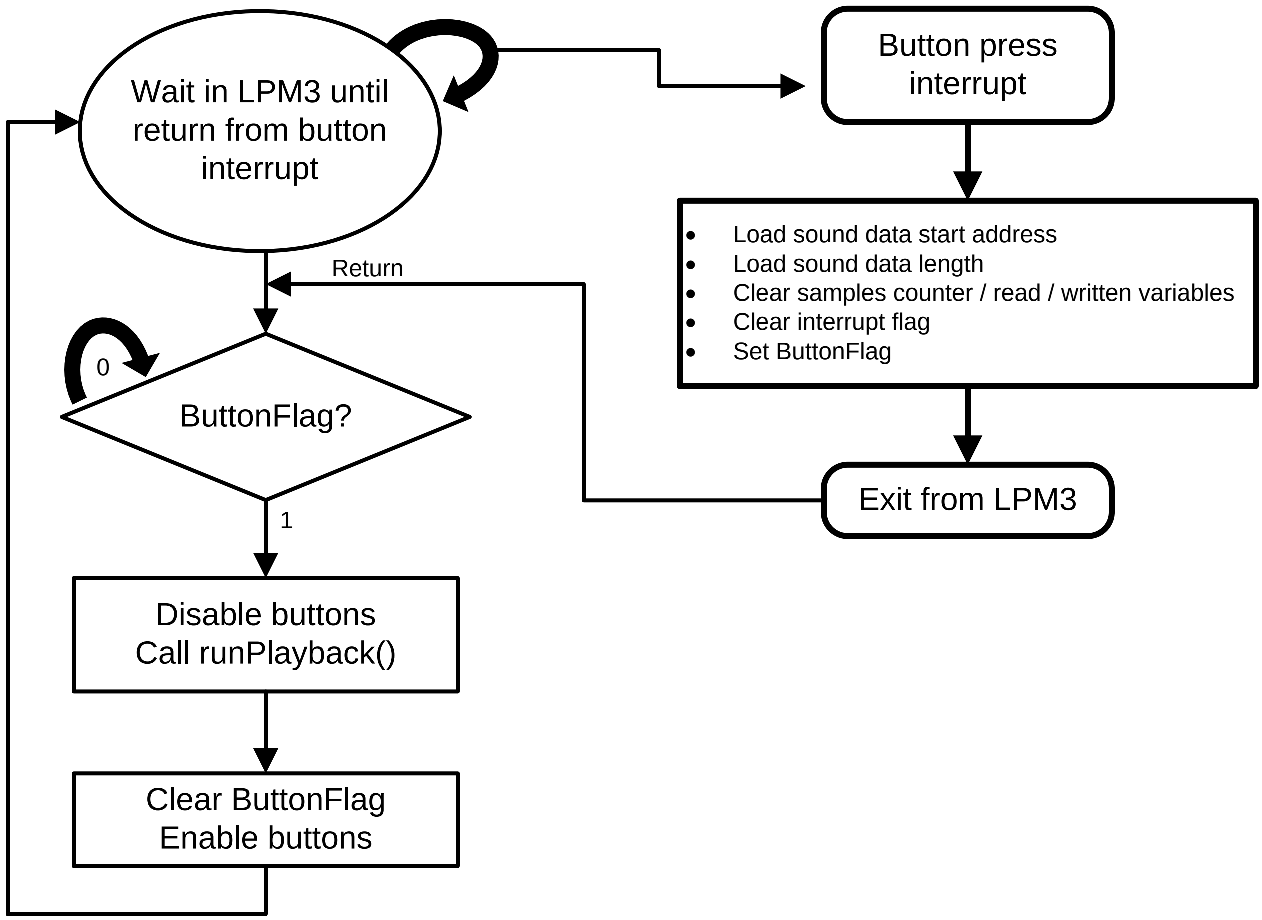


Figure 14. Main Program Flow after Initialization

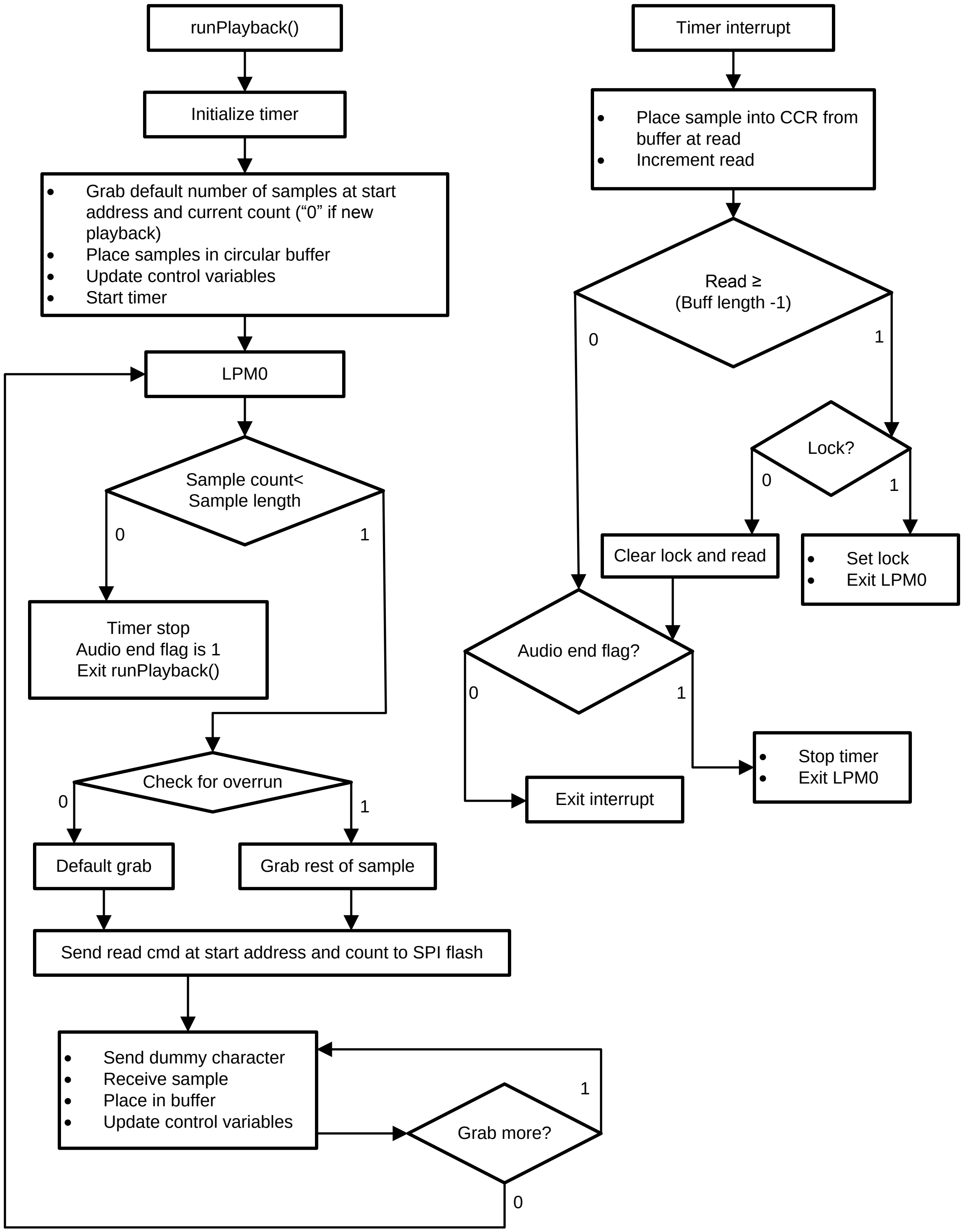


Figure 15. Program Flow of runPlayback()

Figure 16 and Figure 17 show the SPI and Debug lines on a logic analyzer during 16-kHz playback.

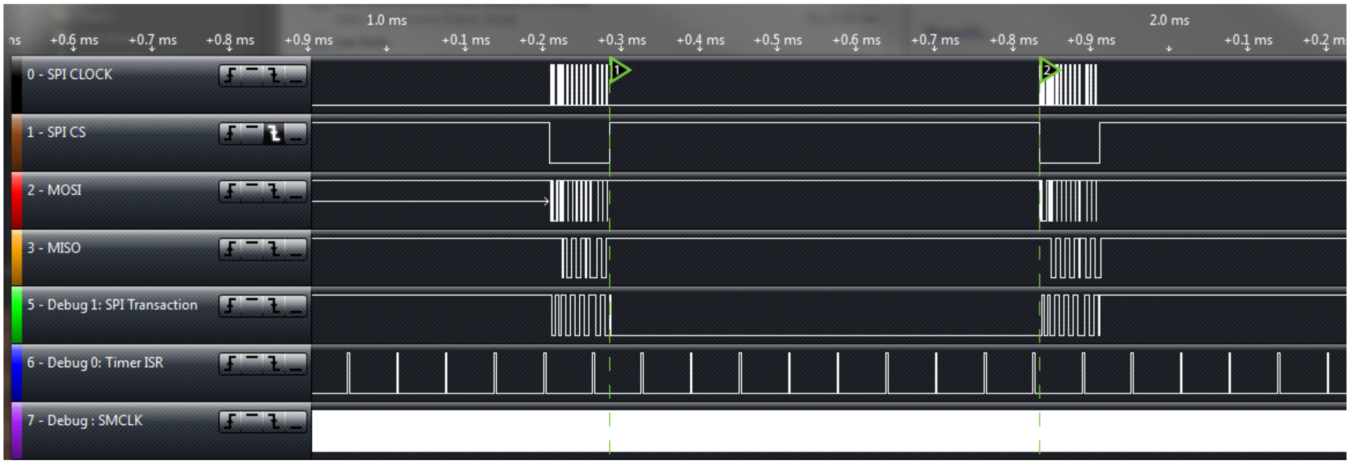


Figure 16. Logic Analyzer View during Playback at 16 kHz

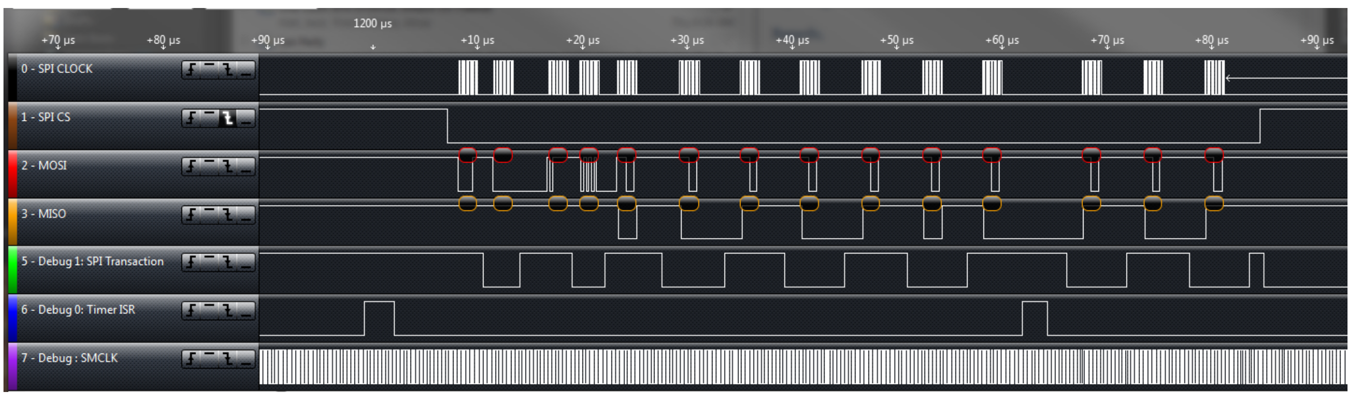


Figure 17. Logic Analyzer View of Single Playback Transaction at 16 kHz

6.2 Customizing Playback

The runPlayback() function makes certain assumptions about the system to play back sound. The user must assure these assumptions are met for playback to be possible in this design, new designs, or while integrating into existing designs. The code supplied with this design takes care of these assumptions and is discussed in the next section. The assumptions that must be met before runPlayback() is called are listed as follows:

- Global variables declaring sample length and SPI Flash location are given values
- Interrupts besides the timer interrupt to be used are disabled for the duration of playback
- Clocks have been initialized to generate correct playback frequency (Timer is initialized within the runPlayback() function)
- Communication peripheral is initialized for SPI operation

NOTE: SPI frequency in the provided code is set to the Timer frequency to ensure samples are ready before needed.

To successfully play sound files with the Playback software, users must modify a few variables and #defines. For user convenience, all of these modifiable values are stored in the Playback_UserDefine.h file. These values and options for playback are discussed and defining where samples are stored in the SPI flash.

6.2.1 FREQUENCY_SELECT #define

This #define selects the playback frequency by changing clock and timer settings at compile time. Although more frequencies are possible, only the following frequencies are supported in this design. These options are also listed in Playback_UserDefine.h file.

Table 3. Options for FREQUENCY_SELECT #define

PLAYBACK FREQUENCY (HZ)	#DEFINE VALUE
44100	FREQUENCY_44kHz
22050	FREQUENCY_22kHz
16000	FREQUENCY_16kHz
8000	FREQUENCY_8kHz

6.2.2 Sample Length Variables

The sample-length variables are global 32-bit variables used during playback and describe how long (how many 8-bit samples) an audio sample is in decimal form. The user must fill in the value of this variable before playback can occur. Currently, this variable is stored as a constant and is of the form soundDataLengthX, where X the number of the sample of interest.

6.2.3 Sample Start Address Variables

The sample start address variables are global 32-bit variables used during playback and describe where an audio sample is stored in the SPI flash by its hex address. The user must fill in the value of this variable before playback can occur. This variable is stored as a constant and is of the form soundDataStartAddressX, where X is the sample of interest. Add additional sample start addresses to a project as needed.

6.2.4 Alternative Pins

Alternative pins are available for use on the VBA BoosterPack for Chip Select, Audio Amp Shutdown, and the Audio PWM. The following defines chose the alternate pin setup: CHIPSELECT_ALTERNATE, AUDIOAMPSHUTDOWN_ALTERNATE, and AUDIOPWM_ALTERNATE. A 0 for each of these defines means the default pins are used, while a 1 uses the alternates. See the hardware section of this guide in Section 4.1 for pin descriptions and what HW changes need to be made. The DEBUG #define enables two debug pins that toggle at each SPI transaction and every Timer ISR entrance and exit in the code. The DEBUG #define also outputs SMCLK so the timer frequency can be monitored. See the Playback_HAL.h file for which pins are being used for debug.

6.2.5 Information and Requirements for Adding Playback Functionality to Existing Applications

- Any MSP430 with two available timer channels for PWM creation
- SPI communication
- Available playback frequencies can vary and are dependent on the flexibility of the clock system of the MSP430.
  - Please see Section 3.2 for more information about frequency selection.
- Outside of the runPlayback() function,
  - A playback trigger is needed to reset playback variables and call runPlayback().
    - An example of this can be seen within the Button ISRs in the Playback_main.c file.
  - A Timer ISR is associated with timer channels for creating the audio PWM. Within the Timer ISR, the CCR is filled with the latest sample. Grab-limit bounds and file-length bounds are also checked.
    - An example of this can be found within Timer ISR in the Playback_main.c file.
- For more information about what the runPlayback() expects before being called, please see Section 6.2.
- Modifications to Playback_HAL.c will be needed to use a MSP430 outside of the supported MSP430s.

7 Test Data

7.1 Power Profiles at Different Playback Speeds

The following data was taken using MSP430 Energy Trace Technology to obtain power profile information. The data was taken over a period of ten seconds using the same playback sound at the different playback speeds. The sound file contained human speech counting “3…2…1…”. The voltage during measurement was approximately 3 V, and Table 4 lists an additional measurement of sleep power / current with the VBA BoosterPack attached.

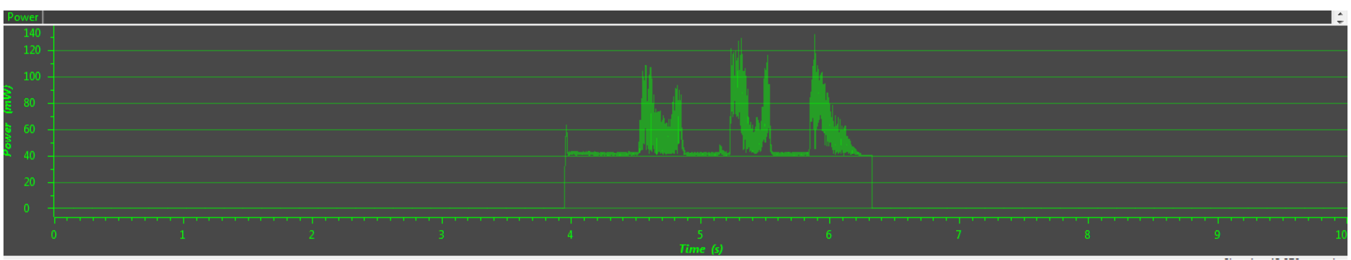


Figure 18. Power Profile at 44kHz Playback

Table 4. Power and Current Values at Different Playback Frequencies

PLAYBACK FREQUENCY	MEASUREMENT	MEAN	MIN	MAX
8 kHz	Power (mW)	33.64	0.143	139.170
	Current (mA)	11.31	0.048	46.811
16 kHz	Power (mW)	19.41	0.163	132.378
	Current (mA)	6.52	0.055	44.497
22 kHz	Power (mW)	12.21	0.149	132.155
	Current (mA)	4.10	0.050	44.362
44 kHz	Power (mW)	11.70	0.143	14.269
	Current (mA)	3.93	0.048	47.118
Sleep with BoosterPack attached, no playback	Power (mW)	0.4	0.394	0.536
	Current (mA)	0.13	0.133	0.180

7.2 Analysis of 1-kHz Sine Wave Output

The following graphs display the output and FFT of a 1-kHz sine wave, which was outputted using this design at the playback frequencies indicated. For each playback frequency, a 1-kHz sine wave was created using third-party software, placed into the SPI flash of the design, and played out through the headphone jack. The yellow wave in the plot is a 1-kHz Sine wave from a lab function generator and serves as reference. The blue wave is the output signal taken from the headphone jack of this design. The red plot is the FFT of said output signal.

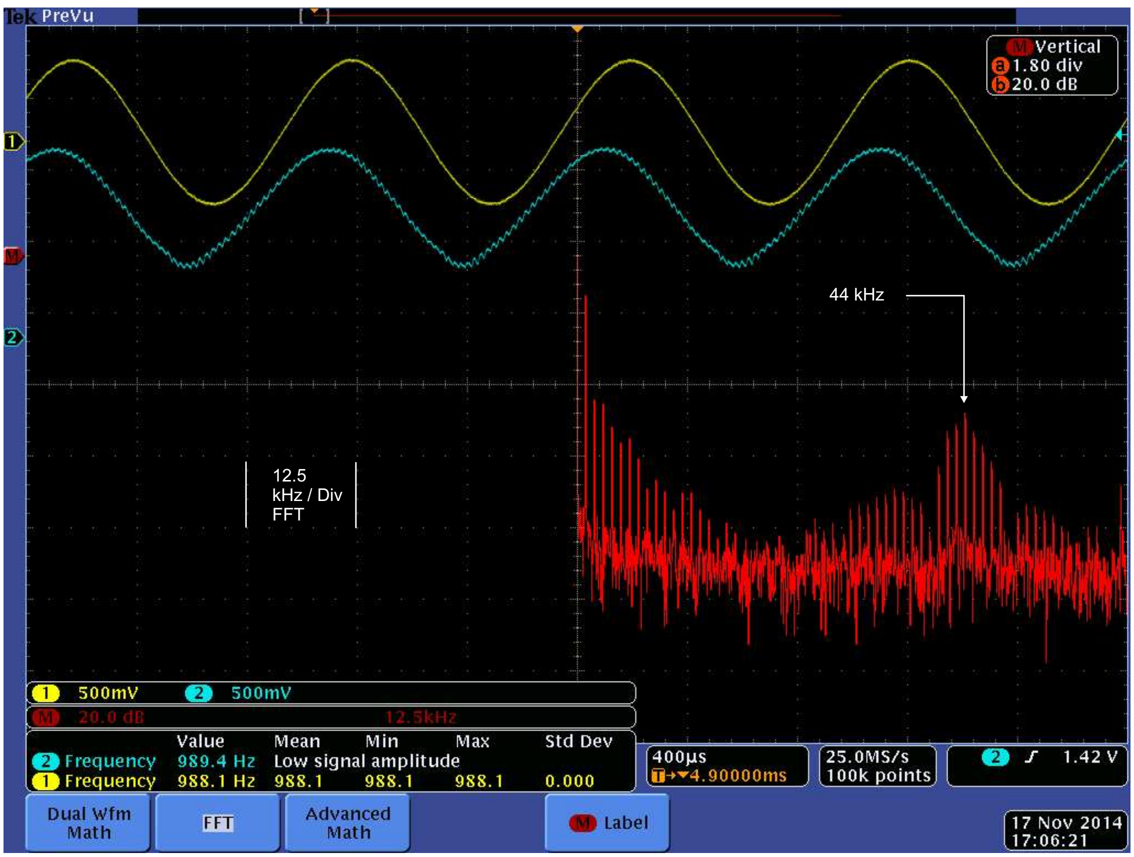


Figure 19. 1-kHz Sine Wave at 44-kHz Playback Frequency

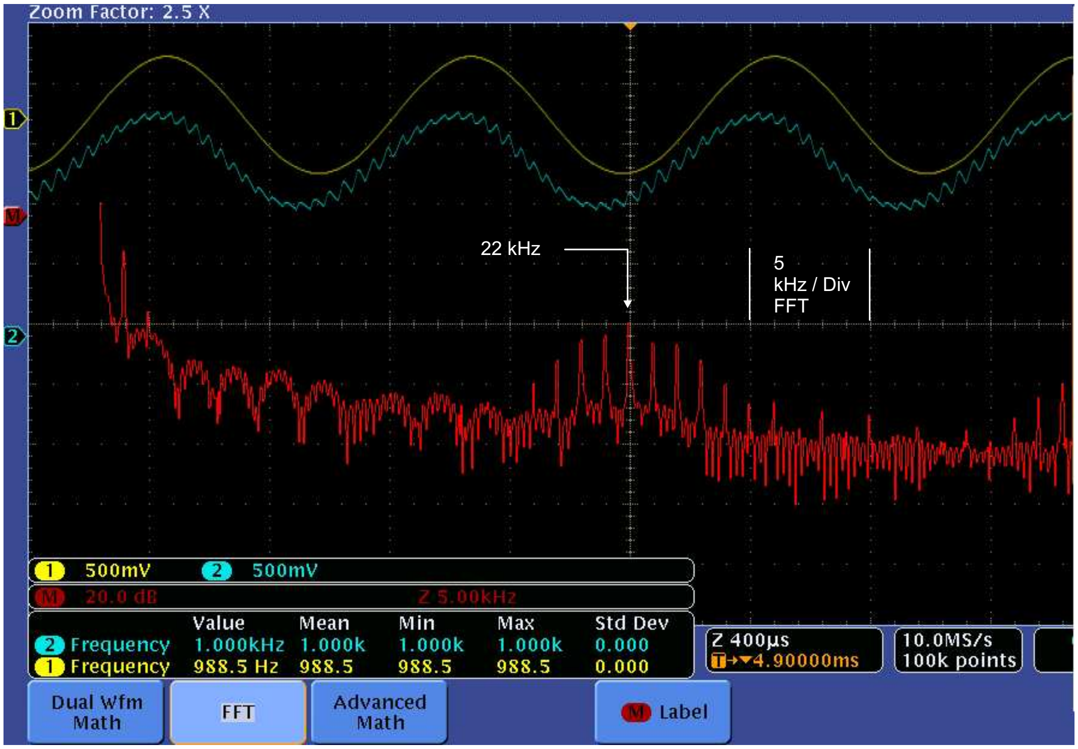


Figure 20. 1-kHz Sine Wave at 22-kHz Playback Frequency

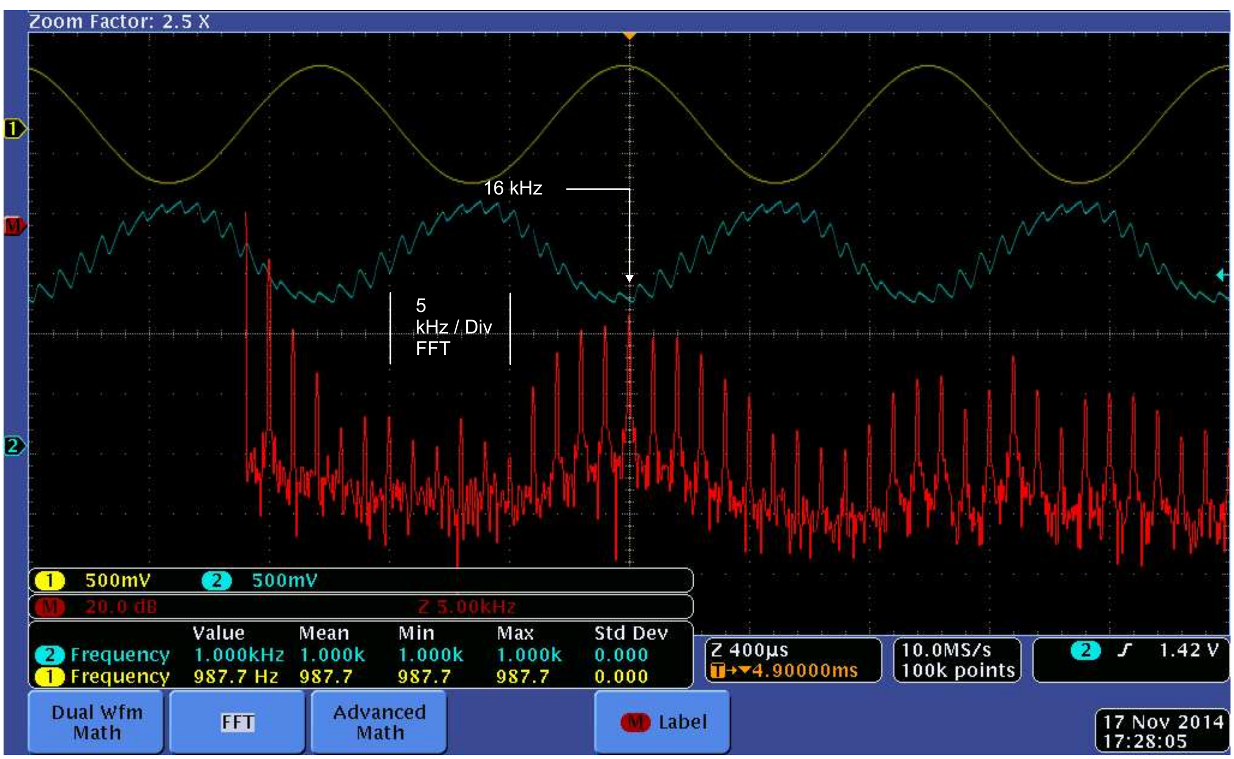


Figure 21. 1-kHz Sine Wave at 16-kHz Playback Frequency

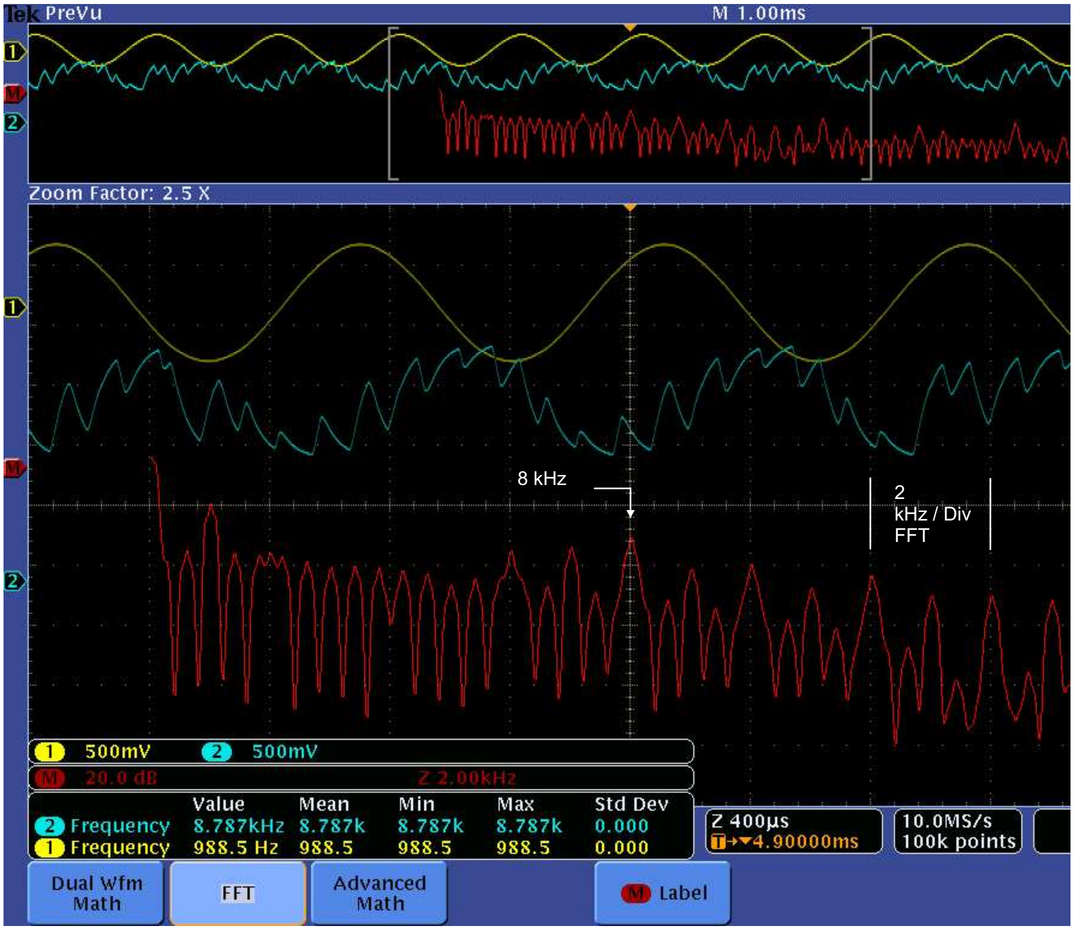


Figure 22. 1-kHz Sine Wave at 8-kHz Playback Frequency

7.3 THD + N Analysis of TPA301 OP-Amp plus Filters

The following chart describes the Total Harmonic Distortion + Noise (THD + N) versus Power. An 8-ohm load and 1-kHz input signal were applied to the audio signal chain of this design. At approximately 10% THD, inputs signals will start to sound distorted.

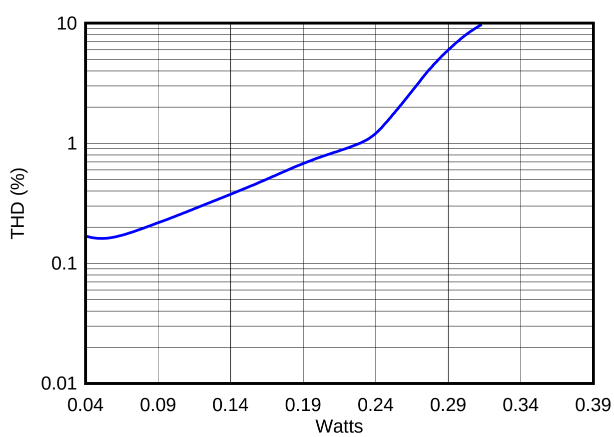


Figure 23. THD + N versus Power

8 Design Files

8.1 Schematics

To download the Schematics for each board, see the design files at http://www.ti.com/tool/TIDM-VOICEBANDAUDIO.

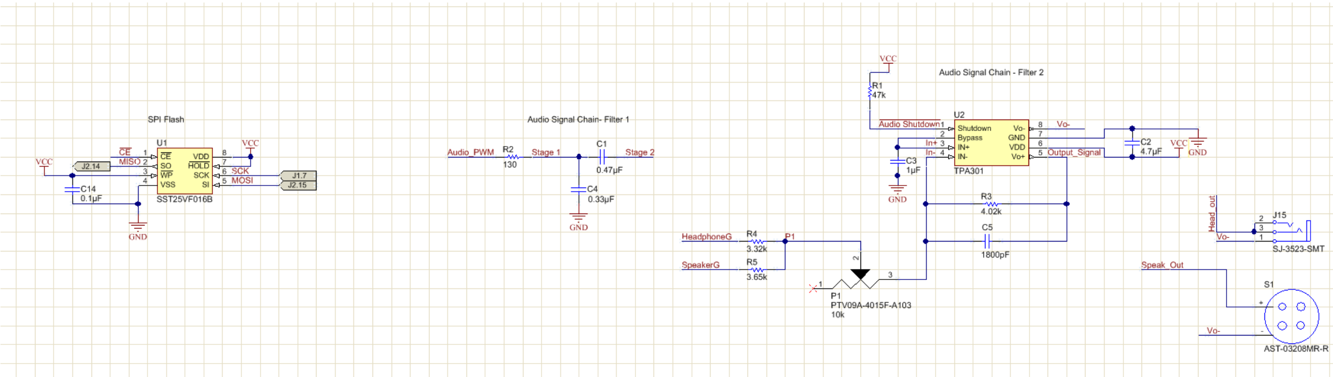


Figure 24. Signal Chain and SPI Flash Schematic

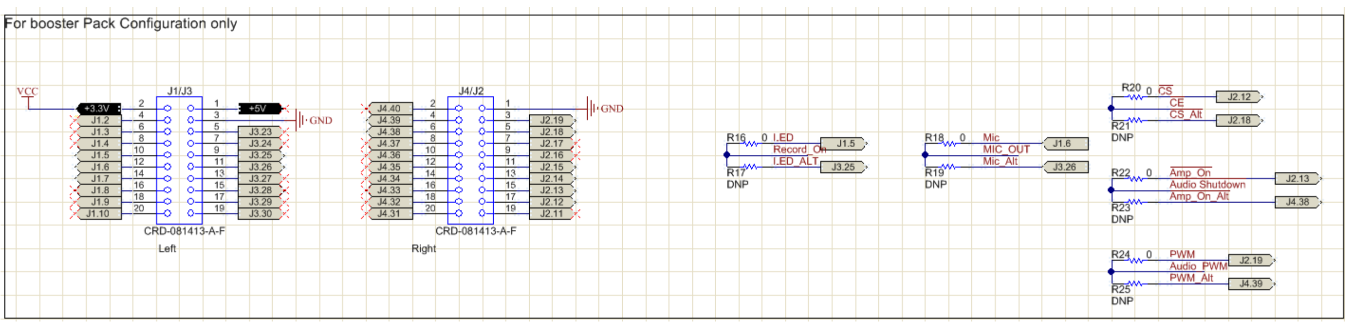


Figure 25. BoosterPack Header Schematic

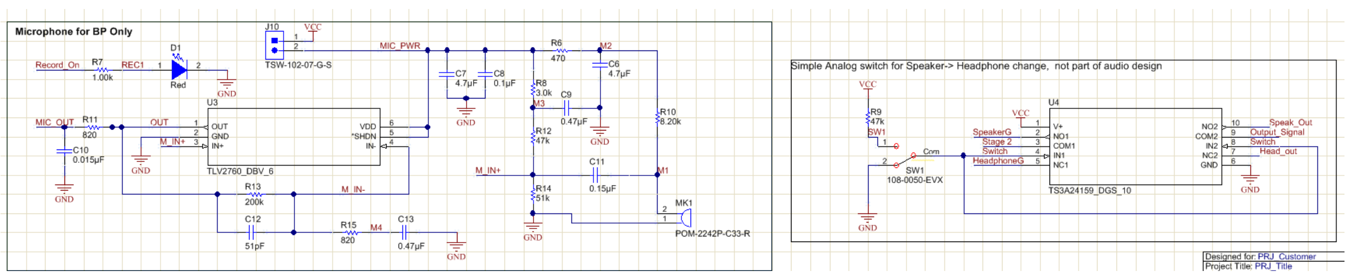


Figure 26. Microphone and Switch Schematic

8.2 Bill of Materials

To download the bill of materials (BOM), see the design files at TIDM-VOICEBANDAUDIO.

8.3 Layer Plots

To download the layer plots, see the design files at TIDM-VOICEBANDAUDIO.

8.4 Altium Project

To download the Altium project files, see the design files at TIDM-VOICEBANDAUDIO.

8.5 Layout Guidelines

To download the layout guidelines, see the design files at TIDM-VOICEBANDAUDIO.

8.6 Gerber Files

To download the Gerber files, see the design files at TIDM-VOICEBANDAUDIO.

8.7 Assembly Drawings

To download the assembly drawings, see the design files at TIDM-VOICEBANDAUDIO.

8.8 Software Files

To download the software files, see the design files at TIDM-VOICEBANDAUDIO.

9 References

1. *MSP-EXP430F5529LP – MSP430F5529 USB LaunchPad Evaluation Kit* (slau533)
2. *TPA301 – 350-mW Mono Class-AB Audio Amplifier* (TPA301 Datasheet)
3. Texas Instruments Application Report, *PWM DAC Using MSP430 High-Resolution Timer*, (slaa497), 2011
4. Texas Instruments Application Report, *Using PWM Timer_B as a DAC*, (slaa116), 2000

10 About the Author

JACE HALL is an Application Engineer on the MSP430 Customer Applications Team at Texas Instruments. Jace joined TI in 2012 into the Digital ARP program where he worked with various business groups and expanded his knowledge across several TI products. Jace now concentrates on direct customer support of the MSP430 ultra-low-power microcontroller family. Jace earned his Bachelor of Science in Computer Engineering from the Georgia Institute of Technology in Atlanta, GA.

IMPORTANT NOTICE FOR TI REFERENCE DESIGNS

Texas Instruments Incorporated ("TI") reference designs are solely intended to assist designers (“Buyers”) who are developing systems that incorporate TI semiconductor products (also referred to herein as “components”). Buyer understands and agrees that Buyer remains responsible for using its independent analysis, evaluation and judgment in designing Buyer’s systems and products.

TI reference designs have been created using standard laboratory conditions and engineering practices. **TI has not conducted any testing other than that specifically described in the published documentation for a particular reference design.** TI may make corrections, enhancements, improvements and other changes to its reference designs.

Buyers are authorized to use TI reference designs with the TI component(s) identified in each particular reference design and to modify the reference design in the development of their end products. HOWEVER, NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY THIRD PARTY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT, IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

TI REFERENCE DESIGNS ARE PROVIDED "AS IS". TI MAKES NO WARRANTIES OR REPRESENTATIONS WITH REGARD TO THE REFERENCE DESIGNS OR USE OF THE REFERENCE DESIGNS, EXPRESS, IMPLIED OR STATUTORY, INCLUDING ACCURACY OR COMPLETENESS. TI DISCLAIMS ANY WARRANTY OF TITLE AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, QUIET ENJOYMENT, QUIET POSSESSION, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS WITH REGARD TO TI REFERENCE DESIGNS OR USE THEREOF. TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY BUYERS AGAINST ANY THIRD PARTY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON A COMBINATION OF COMPONENTS PROVIDED IN A TI REFERENCE DESIGN. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, SPECIAL, INCIDENTAL, CONSEQUENTIAL OR INDIRECT DAMAGES, HOWEVER CAUSED, ON ANY THEORY OF LIABILITY AND WHETHER OR NOT TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, ARISING IN ANY WAY OUT OF TI REFERENCE DESIGNS OR BUYER’S USE OF TI REFERENCE DESIGNS.

TI reserves the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques for TI components are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

Reproduction of significant portions of TI information in TI data books, data sheets or reference designs is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards that anticipate dangerous failures, monitor failures and their consequences, lessen the likelihood of dangerous failures and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in Buyer’s safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed an agreement specifically governing such use.

Only those TI components that TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components that have ***not*** been so designated is solely at Buyer's risk, and Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from Original (February 2015) to A Revision	Page
• Updated Figure 13, *Memory Organization Alongside Setup Code Snippet*	14
• Updated *Playback Software* section, "Playback.c and PlaybackHAL.h" became "Playback_main.c, Playback_HAL.h, and Playback_UserDefine.h"	14
• Updated Figure 14, *Main Program Flow After Initalization* and Figure 15, *Program Flow of runPlayback()*	15
• Changed all instances of Play() to runPlayback()	17
• Changed "the beginning of the Playback_HAL.h file surrounded by comments signifying the User Defined Area." to "the Playback_UserDefine.h file"	17
• Removed section 6.2.1 Processor #define, Changed Frequency Select to FREQUENCY_SELECT in section 6.2.2 title and table title, Removed (FS) in title of section and table	18
• Changed "the Playback_HAL.h file before the User Define area." to "listed in Playback_UserDefine.h file."	18
• Updated Table 4. Options for FREQUENCY_SELECT #define	18
• In section 6.2.3, Changed "data_len_X" to "soundDataLengthX"	18
• In section 6.2.4, changed data_start_X to soundDataStartAddressX	18
• Changed "Timer PWM" to "Audio PW" in section 6.2.5 Alternative Pins.	18
• Added "The following defines chose the alternate pin setup: CHIPSELECT_ALTERNATE, AUDIOAMPSHUTDOWN_ALTERNATE, and AUDIOPWM_ALTERNATE." to section 6.2.5 Alternative Pins.	18
• Changed "debug #define" to "DEBUG #define" in section 6.25 Alternative Pins.	18