TI Designs: TIDEP-0099

Audio Pre-Processing System Reference Design for Voice-Based Applications Using C6747

Description

The TIDEP-0099 uses multiple microphones (eight), beamforming, and other algorithms to clean up and extract clear speech from an environment containing noise sources. The increase in voice-activated applications has created a demand for systems that can extract clear voice from noisy environments. These systems are especially important in applications that have voice triggering or speech recognition. This design guide walks through running a demonstration on the C6747 device using a circular microphone board (CMB), an OMAP-L137/TMS320C6747 Floating Point Starter Kit (SK) and also discusses the various concepts used to filter audio.

Resources

TIDEP-0099	Design Folder
TIDA-01454 (CMB)	Design Folder
PCM1864	Product Folder
TMDSOSKL137	Tools Folder
PROCESSOR-SDK-RTOS-OMAPL13X	Tools Folder
TELECOMLIB	Tools Folder

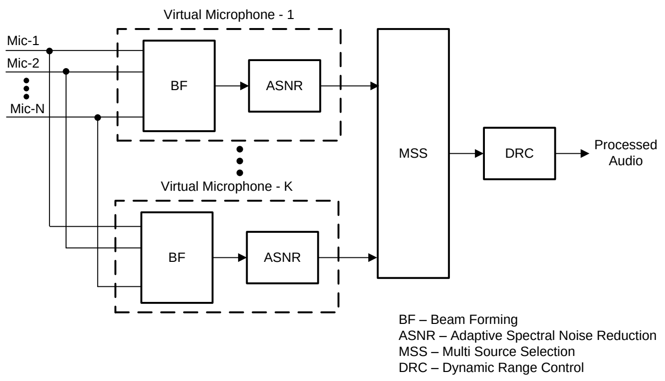
ASK Our E2E Experts

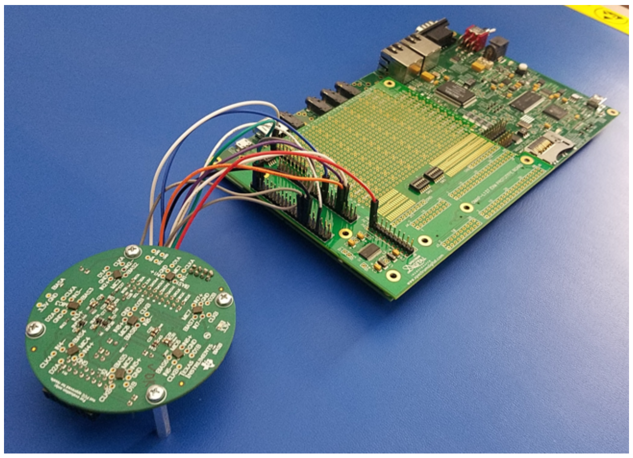
Features

- Uses Single Digital Signal Processor (DSP) and Microphone Array to Extract Clear Speech from a Noisy Environment
- Eliminates Background Noise and Clutter From Audio Source
- Enables Better Voice Recognition by Presenting Clean Speech Audio to the Recognition Engine
- Offers a Complete System Reference Design Using TI-Provided Software, Evaluation Module, and Microphone Array

Application

- Interface-to-Cloud-Based Voice Recognition for Voice-Activated Digital Assistant Applications
- Interface-to-Cloud-Based Voice Recognition for Smart Home Applications
- Local (Limited Dictionary) Voice Recognition for Voice-Based Appliances Control
- Voice and Speech Applications (Such as Video Conferencing)





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

The TIDEP-0099 uses TI hardware and sophisticated, field-proven software algorithms to obtain clear speech and audio from noisy environments. The ability to extract clear speech or audio from a noisy environment is important to many applications that use voice-activation, such as digital assistants, telephone and video conferencing, and other high-quality speech systems. Typical sources of sound clutter are undesired background noise sources and so forth. This TI Design uses a beamforming algorithm to form a virtual-directional microphone that points at the direction of the speaker or the desired audio source. This amplifies the speech signal from the desired direction, and attenuates signals from all other directions. In addition to beamforming, TI offers a set of audio algorithms that further improves the quality of sound. Section 2.3 summarizes the theory of the beamforming, which uses multiple microphones, an associated adaptive spectral noise reduction (ASNR) filter, and the multiple source selection (MSS) algorithm to obtain the virtual-directional microphone signal.

The interface between the microphone array and the processor must support streaming of multiple data inputs. The data rate depends on the application requirements. The TIDEP-0099 streams eight microphones mounted on circular microphone board (CMB), samples in 16-bits at 16000 samples per second, and uses the analog-to-digital converter (ADC) PCM1864 for an inter-IC sound (I2S) interface to the evaluation module (EVM) board. The PCM1864 is a highly flexible audio front end supporting input levels from small-mV microphone inputs to 2.1 VRMS line inputs without external resistor dividers. This device is highly configurable and able to support both analog and digital microphones.

The EVM supports multiple audio output avenues. The audio data can be processed locally or sent out through one of the OMAP-L137/TMS320C6747 Floating Point SK external ports.

An application that uses local processing, such as voice-recognition remote-control appliances, can process the data locally. The TIDEP-0099 loops the clean audio back into the left channel of the stereo audio output interface from the onboard AIC3204 audio codec. The reference microphone (one of the microphones in the circular microphone array) plays out of the right channel. This setup enables the user to compare the quality of processed and unprocessed audio.

This TI Design includes full source code that can be modified to support various applications. For an optional cloud-based, voice-activated digital assistant design, the output signal can be sent to a network interface device using external interface, such as UART, SPI, or USB. The return audio signal from the network can be sent to the device codec to be played by a speaker. Local (limited dictionary) voice recognition for voice-activated digital assistant applications could use the DSP to do voice recognition. The DSP in the C6747 is a high-performance, floating-point VLIW DSP core clocked at 456 MHZ. The DSP has enough power and memory to support voice recognition of a limited dictionary. Conference call and other speech-processing applications require additional features (mixing of signals, acoustic echo cancellation, and so on). As stated above, the DSP in the EVM has enough power and memory to process limited speech algorithms. Note that TI audio libraries include optimized audio algorithms that can be used by speech applications.

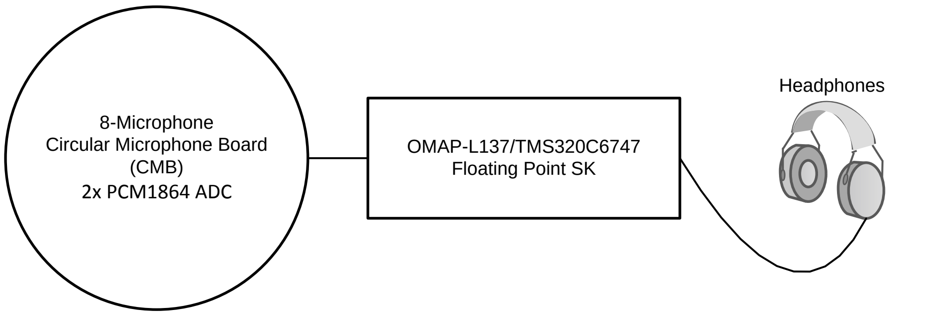


Figure 1. Hardware Block Diagram of TIDEP-0099

1.1 Key System Specifications

Table 1. Key System Specifications

COMPONENT	DESCRIPTION	DETAILS
Circular Microphone Board (CMB)	Eight microphone array with seven microphones mounted in an equal arc along the circle and one mounted at the center.	Section 2.2.1
PCM1864	PCM1864 audio ADC provides I2S interfaces to the SK.	Section 2.2.2
OMAP-L137/TMS320C6747 Floating Point SK	Evaluation board based on the OMAP-L137 (C6747 DSP + ARM9® processor)	Section 2.2.4
OMAP-L137 prototyping module	Prototype card with expansion connectors populated for the audio expansion connector	—
Processor software development kit (SDK) real time operating system (RTOS) (processor_sdk_rtos_omapl137)	Standard TI software release for multiple devices, which includes tools, utilities, drivers, operating system support, optimized library, and more	Section 2.2.5
Executable OMAPL137_bf_rt	DSP executable code that processes multiple microphones streaming audio and generates a virtual-directional microphone audio stream	Section 2.2.5.1
Executable audioAnalogLoopbackTest (for debug purposes)	Executable code that connects one microphone to the left channel of the output stereo codec and a second microphone to the right channel of the output stereo codec using the CMB library	Section 2.2.5.2
Executable offlineSignalProcessing (for debug and simulation purposes)	DSP-executable code that reads offline simulated microphones data streams from files, processes the simulated microphones, and generates a virtual-directional microphone audio stream stored in an output file, which is used for debug and demonstration purposes	Section 2.2.5.3
Application source code and Makefiles	Source code for the data path unit test and for the applications that enables the user to modify or rebuild the code	Section 2.2.5.4
TI Audio Libraries (or TELECOMLIB)	TI-optimized audio processing AEC-AER and VOLIB libraries	Section 2.2.6
Code Composer Studio™ (CCS) version 6 or newer	TI-integrated development environment (IDE) that is used to run the executables and can be used to build the executables (It is assumed that the user is familiar with CCS.)	—

2 System Overview

2.1 Block Diagram

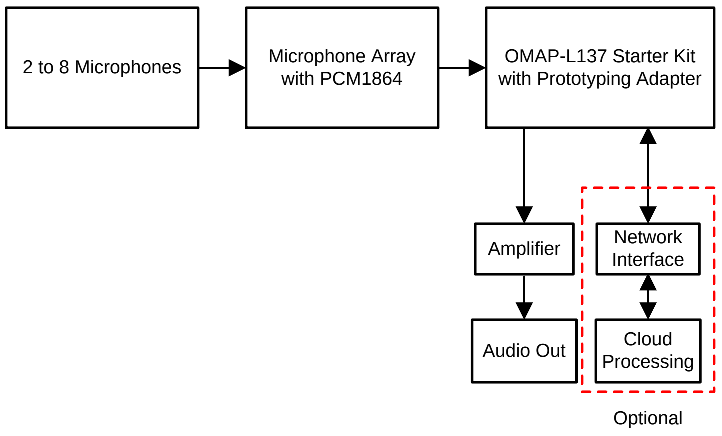


Figure 2. Block Diagram

2.2 Highlighted Products

2.2.1 TMS320C6747

The TMS320C6745/6747 device is a low-power digital signal processor based on a TMS320C674x DSP core. The device consumes significantly lower power than other members of the TMS320C6000 platform of DSPs.

The TMS320C6745/6747 DSP core uses a two-level cache-based architecture. The Level 1 program cache (L1P) is a 32-KB direct mapped cache and the Level 1 data cache (L1D) is a 32-KB 2-way set-associative cache. The Level 2 program cache (L2P) consists of a 256-KB memory space that is shared between program and data space. L2 memory can be configured as mapped memory, cache, or combinations of the two.

Figure 3 highlights the set of peripherals available on the C6747.

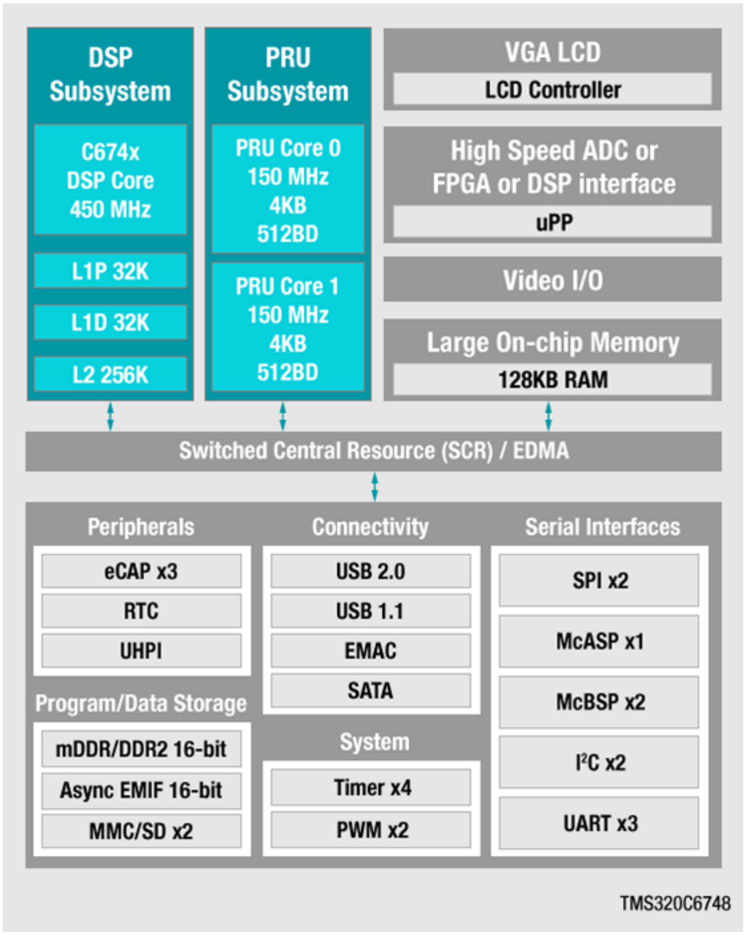


Figure 3. Overview of C6747 SoC

- Software support
  - Processor SDK for C6747
- DSP
  - 456-MHz C6747 VLIW DSP
- C6747 instruction set features
  - Superset of the C67x+ and C64x+ ISAs
  - Up to 3648 MIPS and 2736 MFLOPS C674x
  - Byte-addressable (8-, 16-, 32-, and 64-Bit Data)

- C674x two-level cache memory architecture
  - 32KB of L1P program RAM/Cache
  - 32KB of L1D data RAM/Cache
  - 256KB of L2 unified mapped RAM/Cache
  - Flexible RAM/Cache partition (L1 and L2)
- Enhanced direct memory access controller 3 (EDMA3):
  - Two transfer controllers
  - 32 independent DMA channels
  - Eight quick DMA channels
  - Programmable transfer burst size
- 128KB of RAM shared memory
- 3.3-V LVCMOS IOs (except for USB interfaces)
- Two external memory interfaces
- Three configurable 16550-type UART modules
- LCD controller
- Two serial peripheral interfaces (SPIs) each with one chip select
- Multimedia card (MMC) and secure digital (SD) card interface with secure data IO (SDIO)
- Two master and slave inter-integrated circuit (I2C Bus)
- One host-port interface (HPI) with 16-bit-wide muxed address and data bus for high bandwidth
- Programmable real-time unit subsystem (PRUSS)
- USB 1.1 OHCI (host) with integrated PHY (USB1)
- USB 2.0 OTG port with integrated PHY (USB0)
- Three multichannel audio serial ports (McASPs):
  - Six clock zones and 28 serial data pins
  - Supports TDM, I2S, and similar formats
  - DIT-capable (McASP2)
  - FIFO buffers for transmit and receive
- 10/100 Mbps Ethernet MAC (EMAC):
  - IEEE 802.3 compliant (3.3-V IO Only)
  - RMII media-independent interface
  - Management Data IO (MDIO) module
- Real-time clock with 32-kHz oscillator and separate power rail
- One 64-bit general-purpose timer (configurable as two 32-bit timers)
- One 64-bit general-purpose watchdog timer (configurable as two 32-bit general-purpose timers)
- 256-ball Pb-free plastic ball grid array (PBGA) [ZKB Suffix], 1.0-mm Ball Pitch
- Commercial, industrial, extended, or automotive temperature

2.2.2 PCM1864

The PCM1864 is a 103-dB, two stereo channel (four channels total), SW-controlled audio ADC with universal front end.

See ti.com's PCM1864 product folder for a full description of this device.

2.2.3 CMB

Seven microphones are mounted at equal arc distances from each other on a circle. The eighth microphone is mounted at the center of the circle. The PCM1864 samples the eight microphones and streams the digital values using McASP interfaces to the OMAP-L137/TMS320C6747 Floating Point SK audio expansion (AE) connection on the EVM. Schematics of the CMB are available in Section 4.1.

This board is available from ti.com; see *Processor SDK RTOS Audio Pre-Processing* [1] for more information.

2.2.4 OMAP-L137/TMS320C6747 Floating Point SK

The OMAP-L137/TMS320C6747 Floating Point SK is an evaluation module based on the OMAP-L137 processor. For a full description of the SK, see ti.com's OMAP-L137/TMS320C6747 Floating Point SK tools folder.

Figure 4 shows a block diagram of the OMAP-L137/TMS320C6747 Floating Point SK.

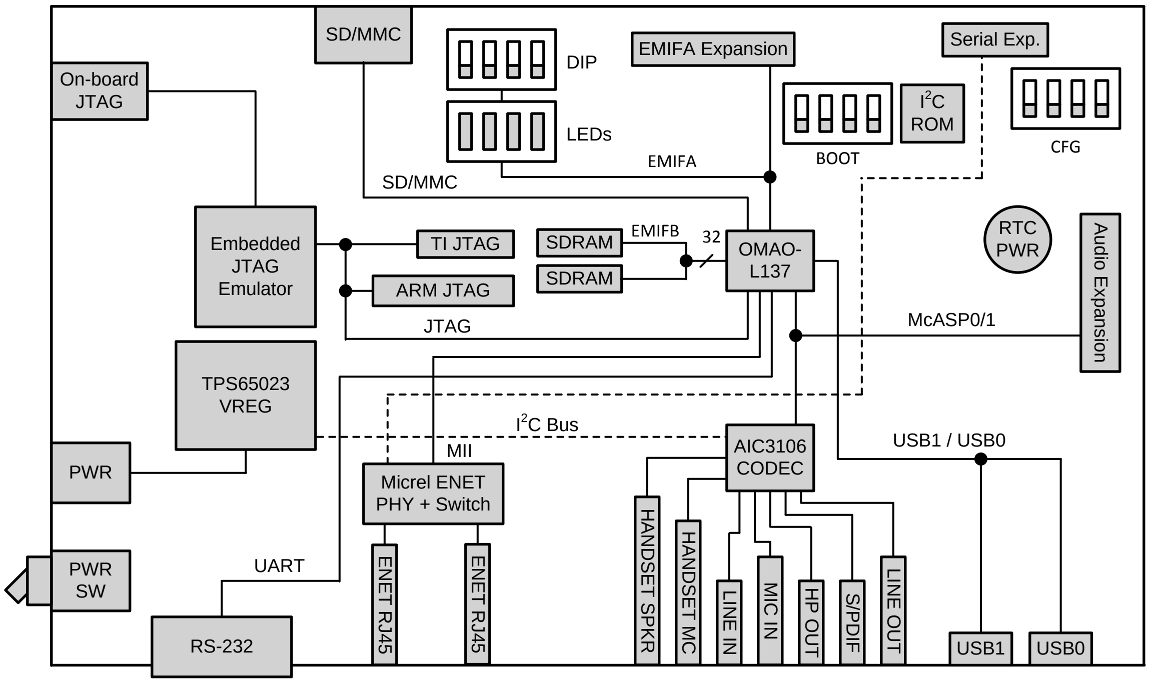


Figure 4. OMAP-L137/TMS320C6747 Floating Point SK Block Diagram

2.2.5 PROCESSOR-SDK-RTOS-OMAP-L137

Processor SDK is a unified software platform for TI embedded processors, which provides easy setup and fast out-of-the-box access to benchmarks and demonstrations. All releases of Processor SDK are consistent across TI's broad portfolio, which allows developers to seamlessly reuse and migrate software across devices. The Processor SDK consists of two types: LINUX and RTOS. The LINUX release (PROCESSOR-SDK-LINUX-OMAPL137) contains LINUX tools, drivers, and utilities for ARM processors. The RTOS (PROCESSOR-SDK-RTOS-OMAPL137) release contains RTOS, drivers, utilities, tools, high-level drivers, and optimized library as well as examples and demonstration projects. The TIDEP-0099 requires the Processor SDK RTOS.

Free download of PROCESSOR-SDK-RTOS-OMAPL137 is available at ti.com's Processor SDK OMAPL137 tools folder.

2.2.5.1 OMAPL137_bf_rt Project

The OMAPL137_bf_rt project is part of the Processor SDK RTOS package. The project contains code that processes seven streams of audio from the CMB, and applies beamforming, ASNR, and MSS to obtain a single, virtual-directional microphone directed at speech, and cleans up noise from the environment. The processed audio is linked to the left channel of the stereo audio output interface using the OMAP-L137/TMS320C6747 Floating Point SK onboard audio codec. The microphone that is mounted at the center of the CMB is used as a reference microphone. The audio from the reference microphone is linked to the right channel of the stereo audio output interface using the OMAP-L137/TMS320C6747 Floating Point SK onboard audio codec. This feature enables the user to compare the original audio with the processed audio. This project uses the same CMB library that is mentioned in Section 2.2.5.2.

NOTE: OMAPL137_bf_rt is the main project in this TI Design. The other two projects are for hardware debugging of the CMB board (audioAnalogLoopbackTest Project) and for illustrating the quality improvements of using beamforming and the other algorithms (DA830_bf Project).

2.2.5.2 audioAnalogLoopbackTest Project

The CMB project (C:\ti\pdk_omapl137_xx_xx_xx\packages\ti\addon\cmb) is part of the Processor SDK RTOS package and may be used to debug the CMB board.

The project contains code to receive eight microphone streams from the CMB. The first of the eight microphones is linked to the left channel of the stereo audio output interface using the OMAP-L137/TMS320C6747 Floating Point SK onboard audio codec. Note that the user may choose any one of the eight microphones to link to the left channel of the stereo audio output interface and rebuild the project. The microphone that is at the center of the CMB is linked to the right channel of the stereo audio output interface, which uses the SK onboard audio codec. All streams from the other seven microphones of the CMB are ignored. The user can choose any one of the eight microphones to connect to the onboard codec. If the user manipulates the desired microphone channeled to the onboard audio codec, the project must be rebuilt.

The project uses the CMB library (ti.addon.cmb.ae66 for little endian and ti.addon.cmb.ae66e for big endian). The CMB library contains sets of utilities and drivers to control audio interfaces on the CMB and to facilitate audio streaming IO to and from the OMAP-L137/TMS320C6747 Floating Point SK.

2.2.5.3 DA830_bf Project

The DA830_bf project (C:\ti\processor_sdk_rtos_omapl137_xx_xx_xx_xx\demos\audio-preprocessing\file_demo_bios\omapl137) is part of the Processor SDK RTOS package and may be used to illustrate the quality improvements of using beamforming and the other algorithms. The project contains code that reads prerecorded audio files from the array of microphones, performs multidirectional beamforming, ASNR, and uses MSS to choose the best directional virtual microphone and store the microphone output in a file. This project can be used to illustrate the quality of beamforming and for debugging and demonstration. This code requires multiple simulated or recorded microphone audio data that can be purchased from third party (for example, from Harman Kardon™). The filter coefficients for the beamforming should be generated by the user. The AER package contains a tool that generates filter coefficients based on the geometry of the microphones. Section 3.2.1.5 describes how to generate filter coefficients.

2.2.5.4 Source Code and Makefiles

In addition to the three executables that are mentioned in Section 2.2.5.1, Section 2.2.5.2, and Section 2.2.5.3, the Processor SDK RTOS contains full source code and a set of makefiles. The source code can be used as a starting point for user applications. In addition, TI provides a set of makefiles to build the projects. The location of the audioAnalogLoopbackTest project is in PDK\ti\addon\cmb\test folder where PDK is the directory where Process SDK RTOS Platform or Processor Development Kit (PDK) is installed. The OMAPL137_bf_rt and the DA830_bf_bios projects are located in the Processor SDK under the demos folder.

2.2.6 TI Audio Libraries

TI Audio Libraries (TELECOMLIB) consists of two optimized libraries that are used in this reference design: the Acoustic Echo Cancellation-Removal (AEC-AER) library and the Voice Library (VOLIB). In addition, the Processor SDK includes a set of optimized libraries that can be used, such as DSPLIB, that contains many signal processing optimized algorithms. AEC-AER and VOLIB can be downloaded from ti.com's TELECOM tools folder.

NOTE: The user must install AEC-AER and VOLIB libraries as subdirectories of the Processor SDK. That is, the same root location that PDK is installed.

2.3 System Design Theory

2.3.1 Beamforming

The ability to extract clear speech or audio from a noisy environment is important to many applications that use voice-activated techniques, such as telephone and video conferencing and other high-quality speech systems. Typical sources of sound-clutter are undesired background noise sources, reverberation, and acoustic echo. The TIDEP-0099 uses a beamforming algorithm to form a virtual-directional microphone that points to the direction of the speaker or the desired audio source. The beamforming algorithm amplifies the speech signal from the desired direction and attenuates all signals from all other directions. In addition to beamforming, TI offers a set of audio algorithms that may further improve the quality of sound-like dynamic range compression. An overview of the audio beamforming mathematics and algorithm can be found in *Acoustic Source Localization and Beamforming: Theory and Practice* [8] and *Beamforming* [9] on Wikipedia. A group of microphones are mounted in predefined locations, which are either along a straight line or on a circle. A point sound source reaches different microphones with different phase delay. The phase delay depends on the frequency, the speed of sound, the distance between each microphone, and the sound source. The distances between the source and the microphones are function of the direction.

Figure 5 shows the distance and the phase difference between two microphones as a function of the direction of the signal arrival.

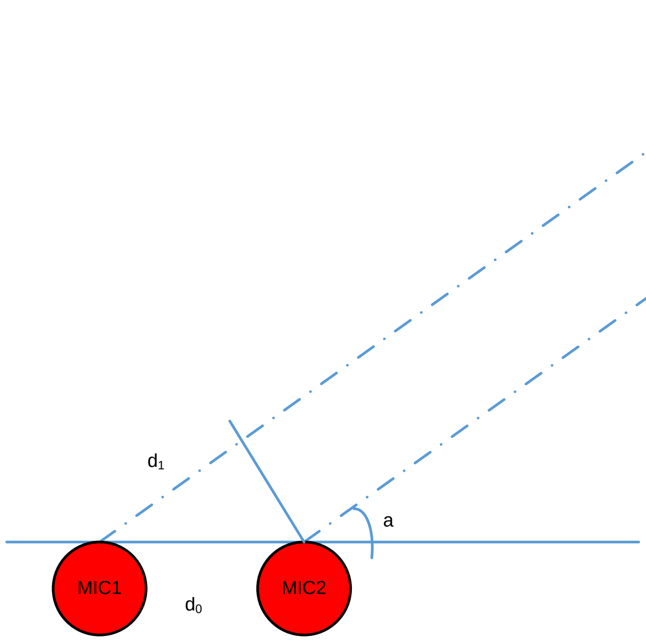


Figure 5. Differences in Distance, Time, and Phase Between Two Microphones

From Figure 5, d_1 is calculated in Equation 1.

$$d_1 = d_0 \times \cos(\alpha) \tag{1}$$

The signal time difference, Δ_t , between mic1 and mic2 is d_1 divided by the speed of sound, as shown in Equation 2.

$$\Delta_t = \frac{d_1}{sos} \tag{2}$$

The phase difference between mic1 and mic2 is shown in Equation 3.

$$\Delta_0 = 2 \times \pi \times \Delta_t \times f = 2 \times \pi \times f \times \frac{d_1}{sos} = 2 \times \pi \times f \times d_0 \times \frac{\cos(\alpha)}{sos} \tag{3}$$

Where:

- Δ_θ is the phase difference
- f is the signal frequency
- d_0 is the distance between two microphones
- α is the angle of arrival and sos is the speed of sound

In a multi-microphone beamforming system, the algorithm applies a set of delay filters to the microphones' signals to shift the signal phase and get the same phase for all the signals (from all microphones) that arrive from one direction. The contribution of all filtered microphones signals are summed together. Thus, the process amplifies signals that arrive from that direction. Because the phase shift (see Equation 3) depends on the angle of arrival (AOA), the phases of filtered signals that arrive from other directions are not the same. Therefore, the sum of all the signals from another direction is decreased, and the energy of the noise (undesired signal that comes from another direction) is reduced.

From Equation 3, it is clear that the quality of the reduction of noise depends on the noise frequency. While the beamforming filters are designed to reduce noise from typical mid-range and higher frequencies, low-frequency noise will not be reduced. An adaptive ASNR filter is applied to reduce the effect of low-frequency noise. Figure 6 shows the reduction of noise as a function of frequency when the beamforming and the ASNR are applied.

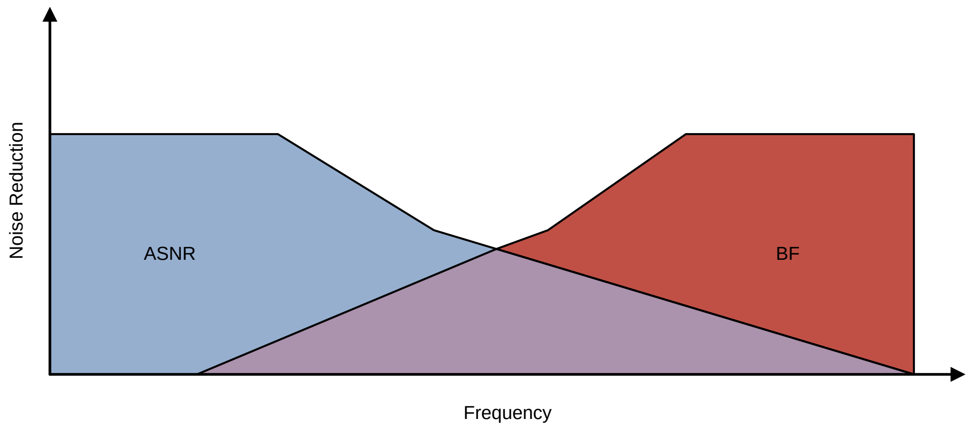


Figure 6. ASNR and BF Noise Reduction as a Function of Noise Frequency

2.3.2 Multi-Angle Beamforming

Figure 7 shows typical multi-angle beamforming where multiple beamforming delay filters and ASNR filters are applied to the microphone sets' data. Each BF and ASNR output corresponds to a different AOA and behaves like a directional microphone; therefore, it is called virtual microphone. An MSS algorithm chooses the best fit virtual microphone.

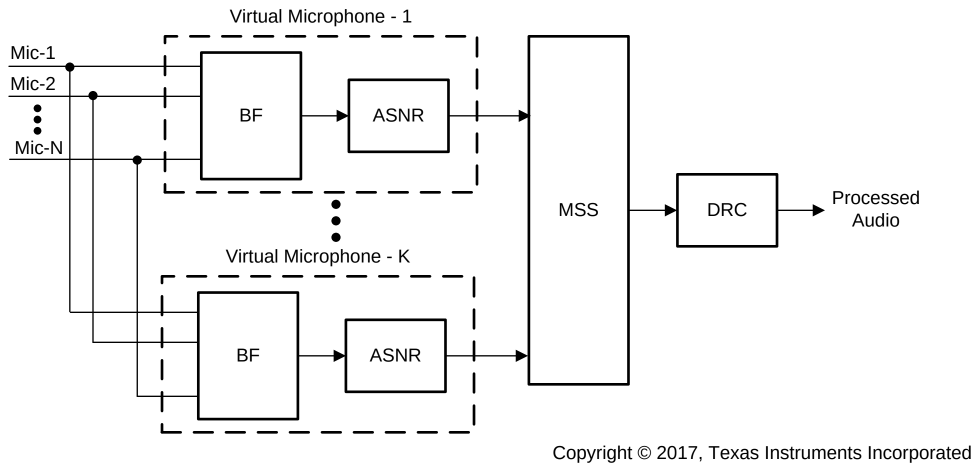


Figure 7. Multi-Angle Beamforming System

3 Hardware, Software, Testing Requirements, and Test Results

3.1 Required Hardware and Software

TIDEP-0099 features a real-time BF demonstration, which runs on the OMAP-L137/TMS320C6747 Floating Point SK. This section will demonstrate how to setup the hardware and software in order to test this demonstration.

3.1.1 Hardware

Here is a list of hardware components needed to run the demonstration:

- OMAP-L137/TMS320C6747 Floating Point SK with power supply
- OMAP-L137 Prototyping Board
- Circular Microphone Board (CMB)
- Female-to-female jumper wires to connect the CMB to the headphones

3.1.1.1 OMAP-L137/TMS320C6747 Floating Point SK Hardware Setup

Detailed steps on how to setup the OMAP-L137/TMS320C6747 Floating Point SK are given in ti.com's tools folder.

Figure 4 shows the OMAP-L137/TMS320C6747 Floating Point SK layout and key components.

3.1.1.2 OMAP-L137 Prototyping Board

The OMAP-L137 prototyping board is an adapter board that gives access to various peripherals pins through headers.

- Over 7 in^2 of prototyping area
- Seven mounted expansion connectors
- Through hole breakout for expansion connectors
- Six power test points
- One 4-position user DIP switch and two 6-position user DIP switch for bus configuration
- Compatible with OMAP-L137 EVM and DA830 EVM
- Four power bars: GND, 5 V, 3.3 V, 1.8 V
- Size: 4.95 in × 3.35 in (125 mm × 110 mm)

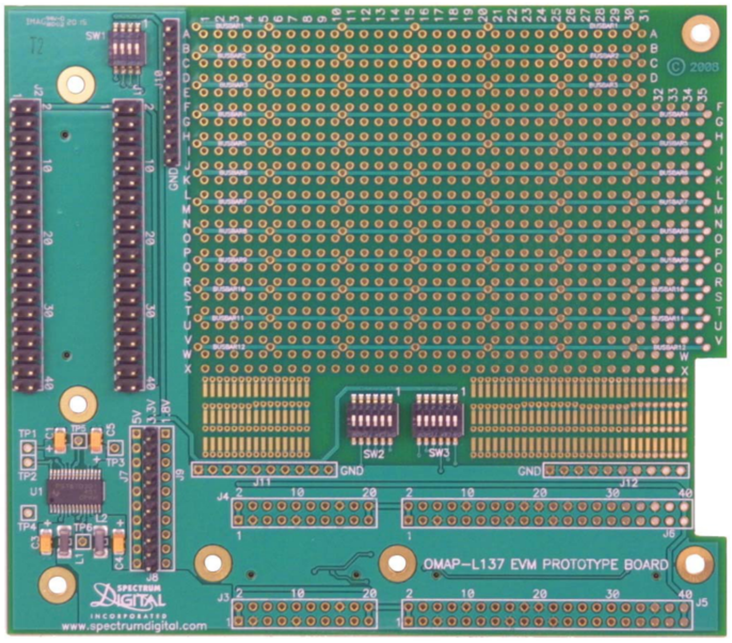


Figure 8. OMAP-L137 Prototyping Board

3.1.1.3 Connecting the CMB to OMAP-L137 Prototyping Board

Ensure the OMAP-L137 Prototyping Board is seated on top of the OMAP-L137/TMS320C6747 Floating Point SK. See Figure 10 for correct orientation. 12 wires connect the audio expansion connector on the OMAP-L137 Prototyping Board to the appropriate pins on the CMB.

Figure 9 shows a layout of the CMB.

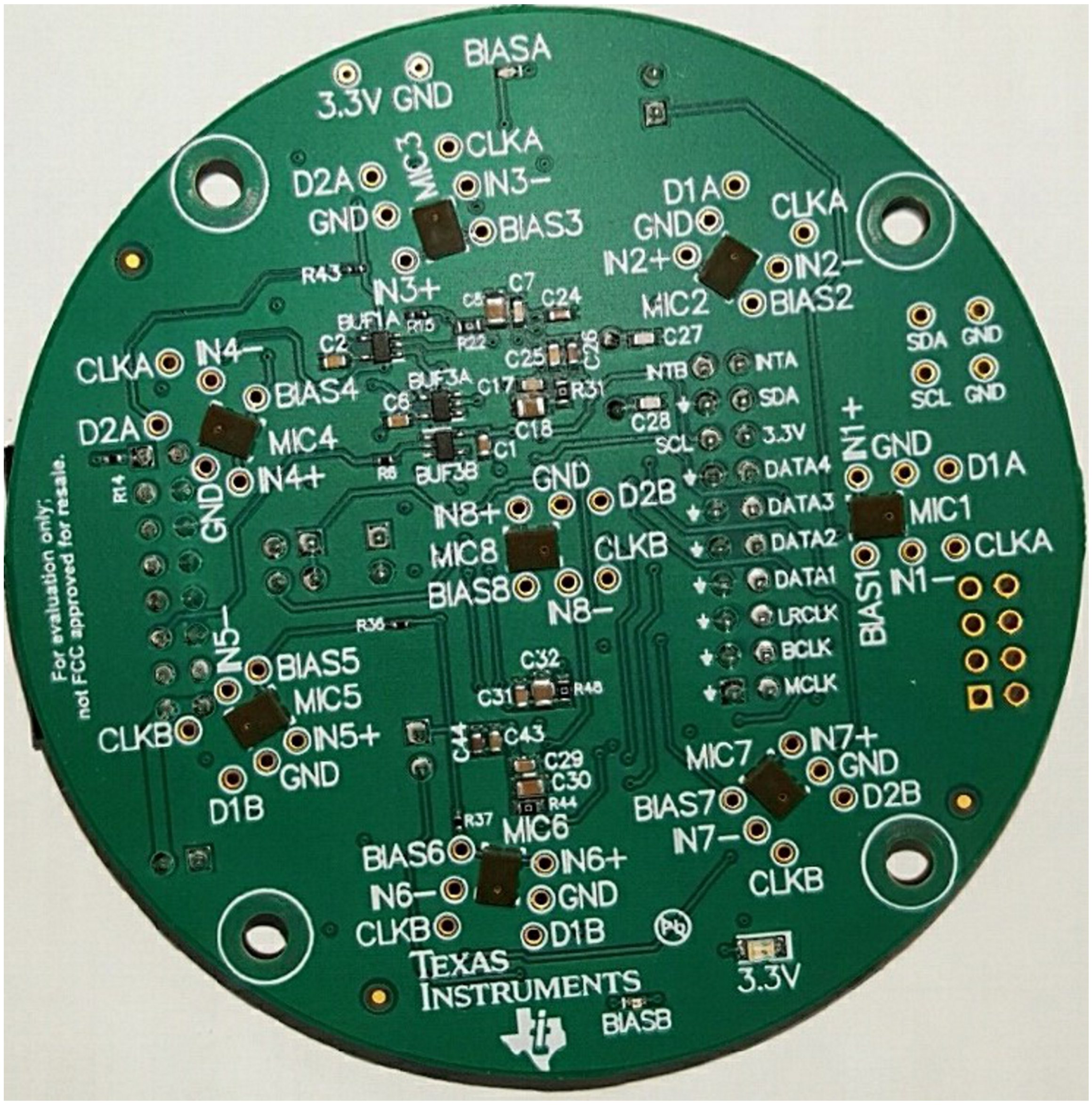


Figure 9. CMB Layout

Table 2 lists the connections between the CMB and the OMAP-L137/TMS320C6747 Floating Point SK.

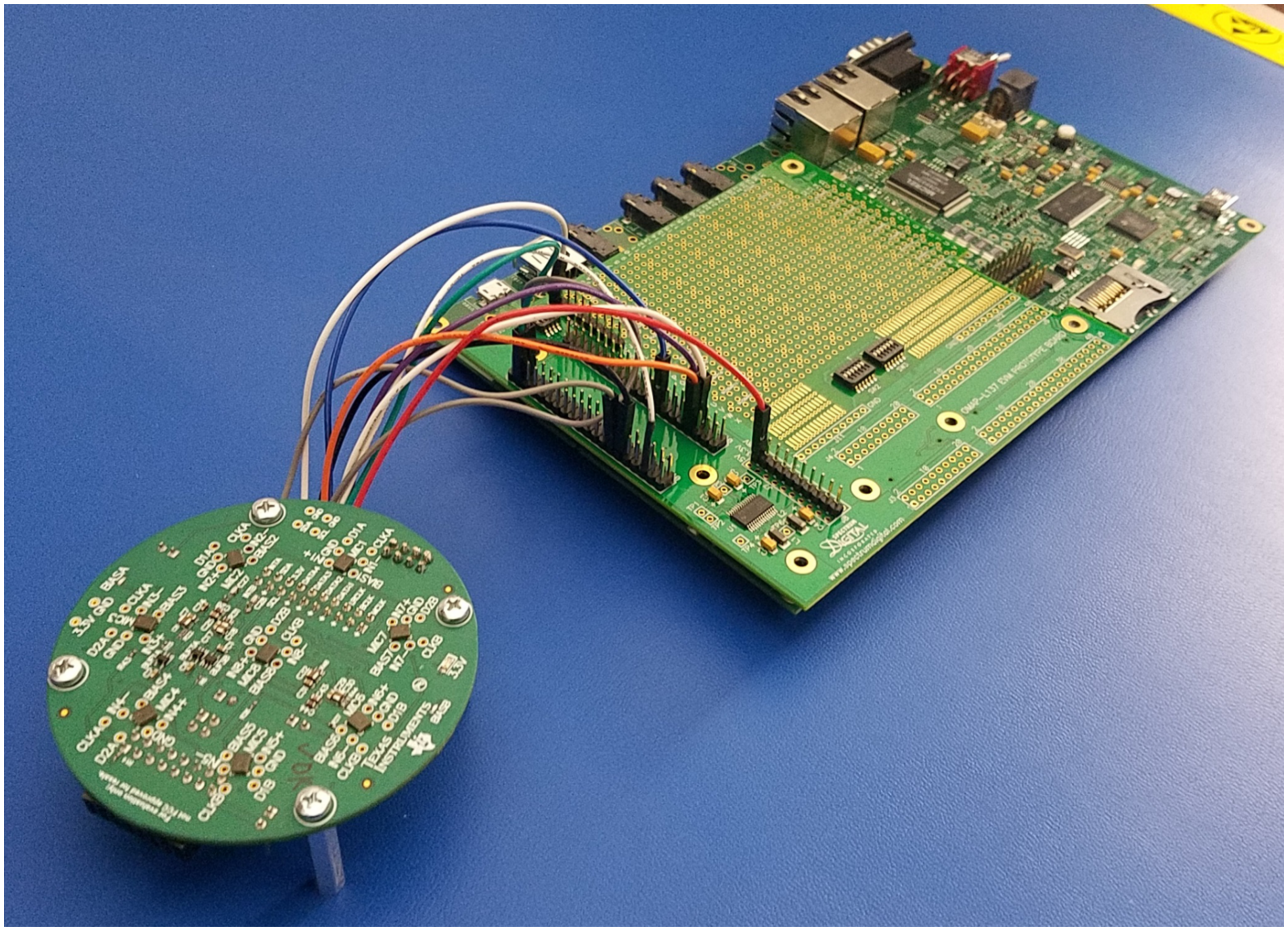
Table 2. Connections Between CMB and OMAP-L137 Prototyping Board

CMB	OMAP-L137 PROTYPING BOARD
CMB_MCLK	OMAPL137_PB_J1_Pin25
CMB_BCLK	OMAPL137_PB_J2_Pin29
CMB_LRCLK	OMAPL137_PB_J2_Pin27
CMB_DATA1	OMAPL137_PB_J1_Pin32
CMB_DATA2	OMAPL137_PB_J1_Pin33
CMB_DATA3	OMAPL137_PB_J1_Pin34
CMB_DATA4	OMAPL137_PB_J2_Pin35
CMB_SDA	OMAPL137_PB_J2_Pin3
CMB_SCL	OMAPL137_PB_J2_Pin4
CMB_3.3V	OMAPL137_PB_J8_3.3V
CMB_GND	OMAPL137_PB_J10_GND

Table 3 summarizes various jumper settings on the CMB.

Table 3. CMB Jumper Settings

CMB HEADER		STATE
J3		OFF
J8	Pins 1 and 2	ON
	Pins 3 and 4	OFF
J10		ON
J11		ON



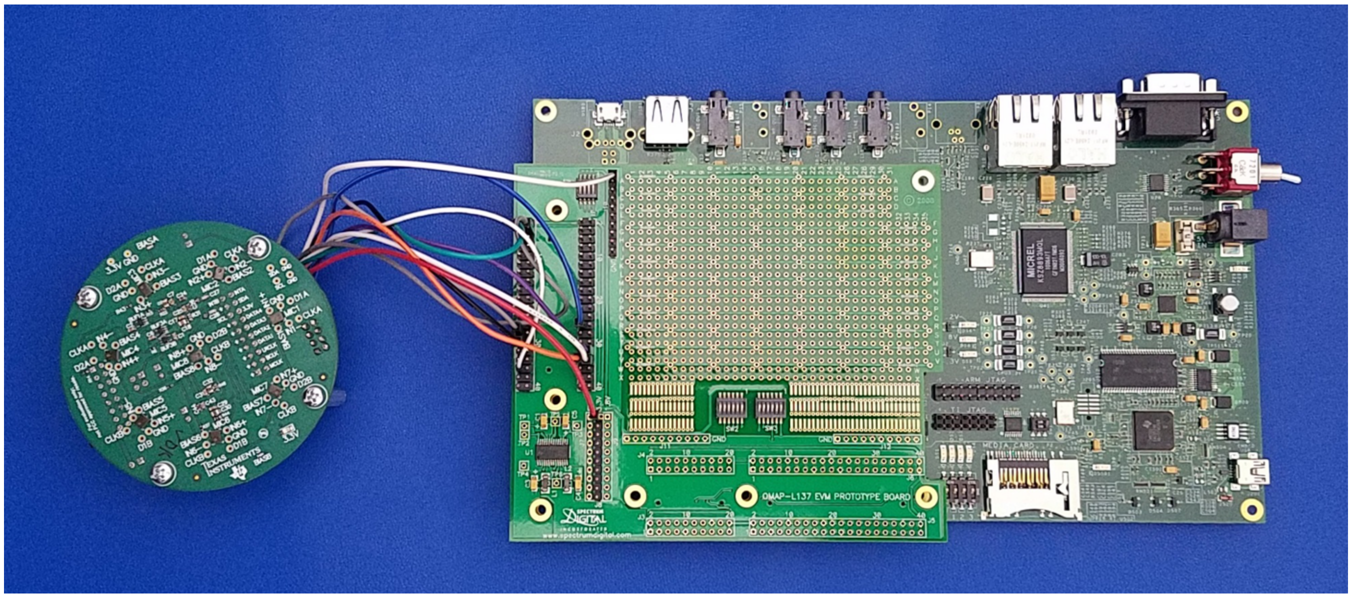


Figure 10. OMAP-L137/TMS320C6747 Floating Point SK With CMB Connected

3.1.1.4 Connect Earphones

A stereo headset should be connected to the output audio. The audio connector line out is located at the bottom left of the OMAP-L137/TMS320C6747 Floating Point SK next to the audio expansion connector (see Figure 11).

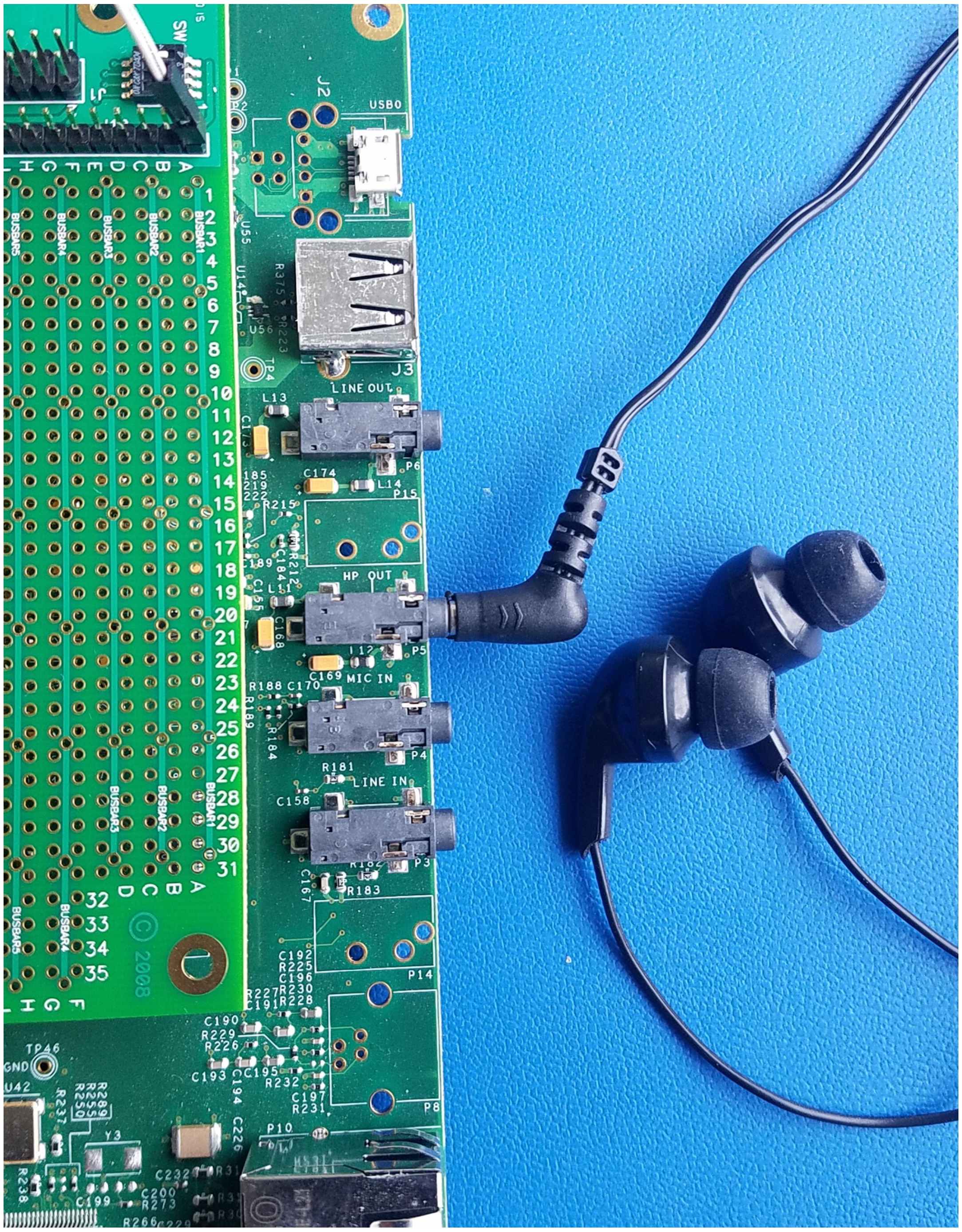


Figure 11. Audio Extension and Audio Line Out on OMAP-L137/TMS320C6747 Floating Point SK

3.1.2 Software

The latest release of Processor SDK RTOS for OMAPL13x can be downloaded from ti.com's PROCESSOR-SDK-RTOS-OMAPL137 tools folder.

See the Processor SDK RTOS Getting Started Guide[3] for information to start working with Processor SDK. For an advanced guide, see the *Processor SDK RTOS Software Developer Guide* [4]. The *Rebuilding The PDK* [5] wiki page has instructions to create all the PDK example projects.

This TI Design assumes that the user has PROCESSOR-SDK-RTOS-OMAPL13X installed and is familiar with the user guides.

This TI Design requires two audio libraries, as discussed in Section 2.2.6: AEC-AER and VOLIB library. The default C:\ti\ location can be used to install these dependent libraries. These libraries can be downloaded from ti.com's TELECOMLIB tools folder. Note that the C67x core uses VOLIB and AER library for C64Px.

3.2 *Testing and Results*

Testing of the beamforming system involves two steps. The first step is testing the complete audio path from the microphones to the OMAP-L137/TMS320C6747 Floating Point SK onboard codec. The second step is testing the beamforming processing quality. Objective testing is done by connecting the output audio signal to a PC and using an audio tool like Audacity® to compare two audio streams and to gauge their quality. Subjective testing is done by listening to the left and right channels streaming of the OMAP-L137/TMS320C6747 Floating Point SK onboard codec (headphone out) and comparing the quality. The following subsections detail how to build, run, and test the three projects.

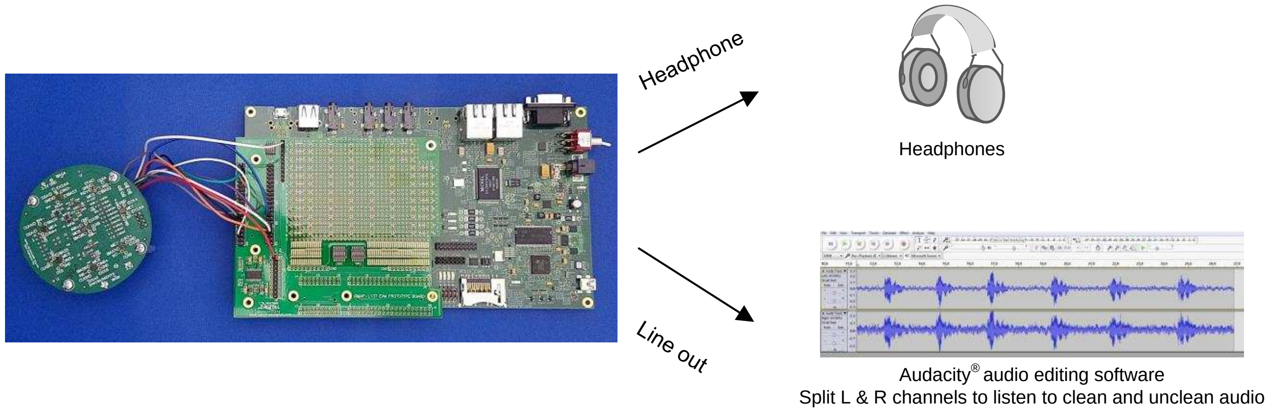
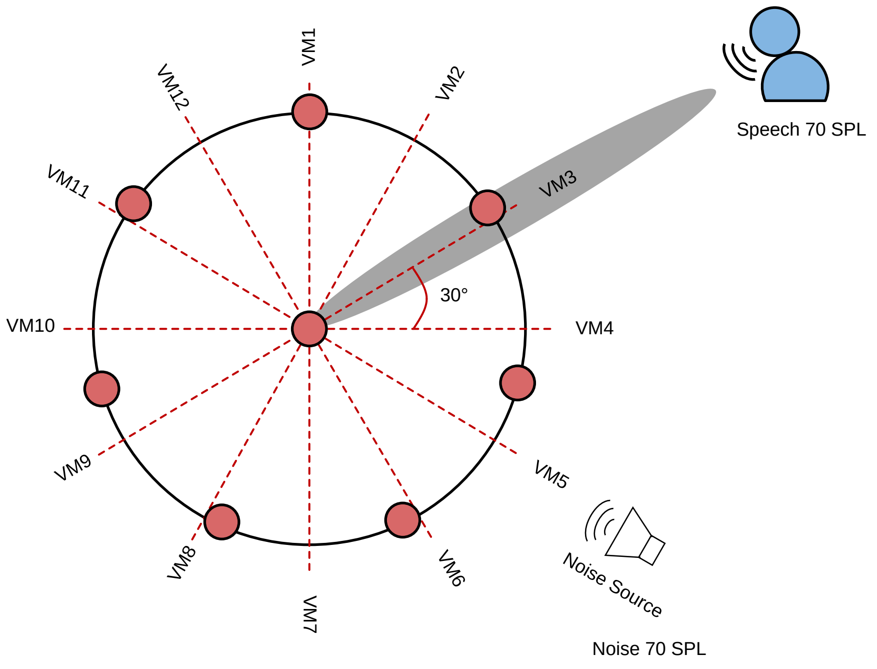

Headphone
Headphones
Line out
Audacity® audio editing software
Split L & R channels to listen to clean and unclean audio

Figure 12. Overview on Evaluating Audio Output of Voice Pre-processing Demonstration on OMAP-L137/TMS320C6747 Floating Point SK

Figure 13 describes the test environment.



Test Environment

Office Room

White Noise at –30° at 70 SPL

Speech 30° at SPL

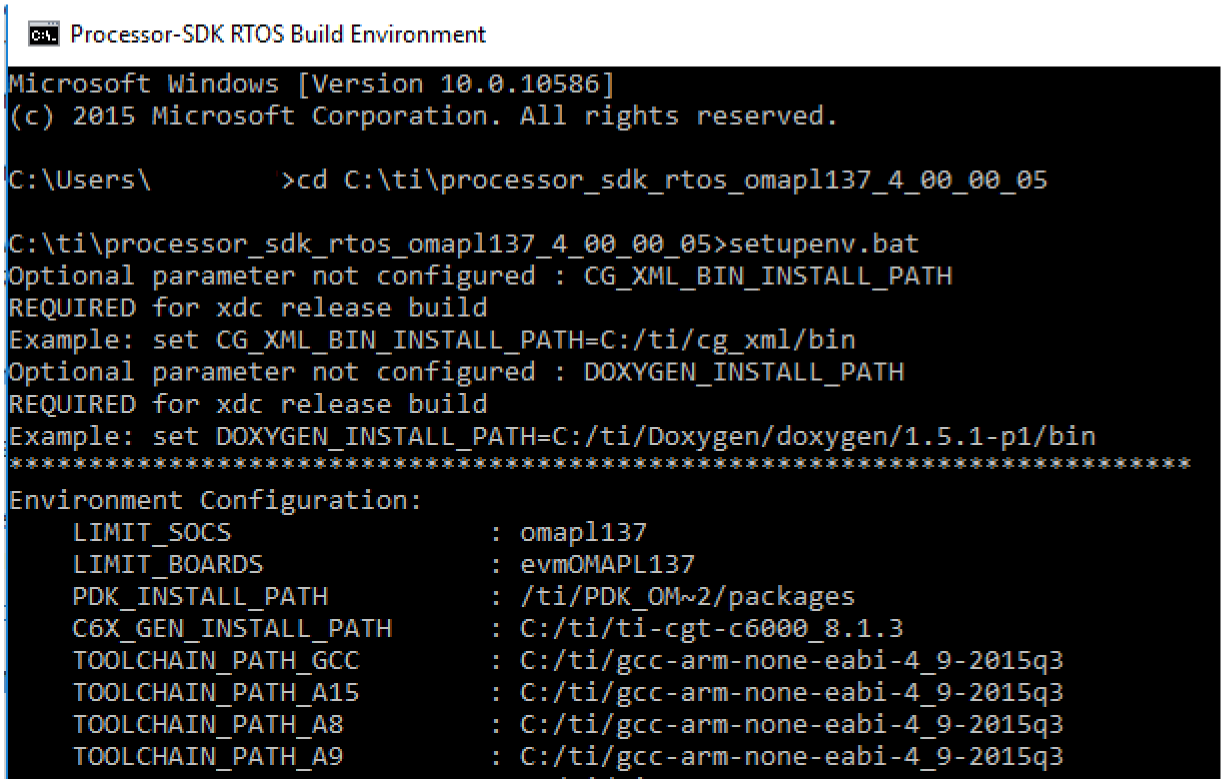
Figure 13. Test Environment

3.2.1 Test Setup

3.2.1.1 Build and Run Executable

Three prebuilt executables are part of the TIDEP-0099 TI Design. The Processor SDK RTOS OMAPL13X contains the executables as well as all source code and makefiles to build the executable. This chapter contains instructions for building and running the three executables. The following instructions are for a Windows® computer. Instructions for LINUX® computer are given in the *Processor SDK RTOS Audio Pre-Processing* [1] wiki page.

The first step in building a project in Windows is to configure environment variables. The file *setupenv.bat* located in directory *C:\ti\processor_sdk_rtos_omapl137_x_xx_xx_xx* is a batch file that sets the environment variables. This file is necessary to run in any new CMD window before starting a build (Figure 14). The user must configure the SDK_INSTALL_PATH and the TOOLS_INSTALL_PATH values based on the user directory structure unless Processor SDK and CCS were installed in the default location, which is c:\ti. SDK_INSTALL_PATH the location where Processor SDK was installed (PROCESSOR_SDK_INSTALL_DIR).


Processor-SDK RTOS Build Environment

```
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\        >cd C:\ti\processor_sdk_rtos_omapl137_4_00_00_05

C:\ti\processor_sdk_rtos_omapl137_4_00_00_05>setupenv.bat
Optional parameter not configured : CG_XML_BIN_INSTALL_PATH
REQUIRED for xdc release build
Example: set CG_XML_BIN_INSTALL_PATH=C:/ti/cg_xml/bin
Optional parameter not configured : DOXYGEN_INSTALL_PATH
REQUIRED for xdc release build
Example: set DOXYGEN_INSTALL_PATH=C:/ti/Doxygen/doxygen/1.5.1-p1/bin
**************************************************************************
Environment Configuration:
    LIMIT_SOCS                  : omapl137
    LIMIT_BOARDS                : evmOMAPL137
    PDK_INSTALL_PATH            : /ti/PDK_OM~2/packages
    C6X_GEN_INSTALL_PATH        : C:/ti/ti-cgt-c6000_8.1.3
    TOOLCHAIN_PATH_GCC          : C:/ti/gcc-arm-none-eabi-4_9-2015q3
    TOOLCHAIN_PATH_A15          : C:/ti/gcc-arm-none-eabi-4_9-2015q3
    TOOLCHAIN_PATH_A8           : C:/ti/gcc-arm-none-eabi-4_9-2015q3
    TOOLCHAIN_PATH_A9           : C:/ti/gcc-arm-none-eabi-4_9-2015q3
```

Figure 14. Setting up Build Environment by Running setenv.bat

Note that SDK_INSTALL_PATH directory has many subdirectories, which includes the PDK directory pdk_omapl137_1_0_1 (or a newer version). The TOOLS_INSTALL_PATH is the location where CCS was installed; this location has several subdirectories including ccv6 (or newer). See the *Processor SDK RTOS Install In Custom Path* [6] wiki page. The CCS and SDK RTOS in Custom Path chapter in *Processor SDK RTOS Install in Custom Path* [6] provides details on how to set the environment variables.

Building the DA830_bf_bios project and the OMAPL137_bf_rt project with a custom path installation, requires configuring more environment variables.

The file setupenv.bat is in the top-level directory where *processor_sdk_rtos_omapl137_x_xx_xx_xx* was installed.

The user must configure SDK_INSTALL_PATH (line 52) to the upper directory where Processor SDK was installed. This directory contains the two audio libraries: AEC-AER and VOLIB. The user must configure TOOLS_INSTALL_PATH (line 57) to the upper directory where CCS was installed.

3.2.1.1.1 Build and Run OMAPL137_bf_rt Project

The OMAPL137_bf_rt project applies the beamforming algorithm and ASNR algorithms on seven microphones to generate eight virtual-directional microphones, which correspond to signal arrival angles of 0°, 45°, 90°,135°, 180°, 225°, 270°, and 315°. The MSS algorithm chooses the virtual-directional signal with the most energy and channels its signal to the left channel headset through the audio codec on the OMAP-L137/TMS320C6747 Floating Point SK. The eighth microphone (the reference microphone in the center of the CMB) is channeled directly to the right headset channel as a reference.

The *Processor SDK RTOS Audio Pre-Processing* [1] wiki page details instructions to build and run the OMAPL137_bf_rt project.

1. After building the executable, the user should load this executable to the C674X DSP core as described in the wiki page and run the code.
2. The ideal method to evaluate the quality of audio processing is to have human speech tested in a noisy environment. Generate noise (white noise or music) from multiple locations in the room. Then, ask a person to speak or play an audio clip through PC speaker. At the same time, listen to the audio output via the stereo headset that is connected to the OMAP-L137/TMS320C6747 Floating Point SK onboard audio codec. In the stereo headset, the left side will output the processed audio (beamforming, ASNR, and MSS processing of the seven periphery microphones of the CMB), and the right side will play the reference unprocessed audio as is recorded by the microphone at the center of the CMB (microphone eight).
3. Objective testing is done by connecting audio signal to the line input connection of a PC and use an audio tool like Audacity to record and then compare two audio streams and to gauge their quality.

Chapter 6 of *Processor SDK RTOS Audio Pre-Processing* [1] shows how to read the input and output audio files and how to compare them. Subjective testing is done by listening to the left and right channels streaming of the OMAP-L137/TMS320C6747 Floating Point SK onboard codec and comparing the quality.

3.2.1.1.2 Build, Run, and Test audioAnalogLoopbackTest Project

The audioAnalogLoopbackTest.out is a stream through project where eight microphones from the CMB are streamed through the I2S connection from the CMB to the OMAP-L137/TMS320C6747 Floating Point SK. The project is described in Section 2.2.5.2.

The *Processor SDK RTOS CMB AddOn* [2] wiki page provides detailed instructions to build and run the audioAnalogLoopbackTest.

1. After building the executable, the user should load this executable to the C6747 DSP core using CCS as described in the wiki and run the code in a room with audio sources.
2. The OMAP-L137/TMS320C6747 Floating Point SK onboard audio codec left channel will stream one of the microphones (default microphone one), and the right side will stream a second microphone [default microphone eight (the one located in the center of the CMB)]. During objective testing, the output of the audio codec is connected to an audio utility on a PC that can show the audio signal and the characteristic of each side and verify that the two microphones and the path to the onboard codec is working correctly. Next two different microphones are connected, the project is rebuilt, and the experiment is repeated. Instructions how to change the microphones that are streaming audio are given in Section 3.2.1.6.
3. An easy way to subjectively verify that all microphones are working properly is to connect the OMAP-L137/TMS320C6747 Floating Point SK onboard codec to a set of earphones and touch the microphones one after the other. The touching sound is heard only when the two connected microphones are touched. Next, change which microphones are connected to the audio codec and repeat the experiment.

3.2.1.1.3 Build and Run file_demo_bios (DA830_bf_bios) Project

The purpose of the DA830_bf_bios.out is to easily demonstrate the performances of the beamforming, ASNR, and MSS algorithm. The program plays seven prerecorded streams of raw audio data recorded from the seven microphones, processes the data, and generates output audio into a file. Each of the prerecorded files as well as the output audio file can be played on a speaker using a third-party audio tool, such as Audacity, to demonstrate the quality of the processing.

The prebuilt program assumes each of the seven microphones samples at 16000 times per second, each sample is embedded in 16-bit data, and each recording is 40 seconds or less; therefore, the size of each of the prerecorded files is limited to 1.28 MB (2 bytes per sample, 16000 samples per second, 40 seconds). The user must pre-load the seven files into the DSP memory prior to the execution.

Prerecorded raw data files can be obtained from a third party. Instructions on how to load these files manually or using a script file are given in Section 4.1.1.3.1. A set of synthetic audio files are part of the Processor SDK release in directory *C:\ti\processor_sdk_rtos_omapl137_xx_xx_xx_xx\demos\audio-preprocessing\common\t8*.

The beamforming filters depend on the geometry of the microphone. Using prerecorded audio files requires a different set of filter coefficients. A set of filter coefficients that were built for the synthetic audio files are part of the Processor SDK release in directory *C:\ti\processor_sdk_rtos_omapl137_xx_xx_xx_xx\demos\audio-preprocessing\common\filters*.

Detailed instructions how to build and run the DA830_bf_bios project is available in *Processor SDK RTOS Audio Pre-Processing* [1].

3.2.1.2 Loading Audio Files to Core Memory

The *Processor SDK RTOS Audio Pre-Processing* [1] wiki page describes how to load the prerecorded audio files that are part of the TI release. If the user wants to use different prerecorded audio files, the user must load the new audio files to the core memory by either using manual load, modifying the GEL file, or building a new GEL file.

3.2.1.2.1 Manual Load

The offline signal processing project dedicates eight 1.28-MB buffers for loading prerecorded raw audio data files. filBuf0 though filBuf6 are seven buffers in the DSP memory map that must be loaded with the prerecorded raw data. The global addresses of these files are given in the project map file.

To manually load the prerecorded raw-audio-data files:

1. Open CCS, and connect to the DSP target.
2. Select the DSP core, and load the out file executable.
3. Before starting execution, open a memory browse window (from the debug perspective view tab).
4. Right-click in the memory window, and choose *Load Memory*.
5. A dialog window will open. Fill out the required information (file name, attributes, and memory address).
6. Wait for the loading to complete, and load the next prerecorded raw audio data file.
7. Repeat until all eight files are loaded.

3.2.1.2.2 Using a General Extension Language (GEL) File

Information about GEL scripts is available from TI's GEL wiki page. The function GEL_Memory_Load() loads raw data file into the memory. Pages 12 through 28 of *Code Composer Studio User's Guide* defines how to use the *GEL_Memory_Load* function.

The GEL file files_io_7.gel from directory *C:\ti\processor_sdk_rtos_omapl137_xx_xx_xx_xx\demos\audio-preprocessing\file_demo_bios\omapl137* contains instructions to load TI Design example audio files (which reside in subdirectory common\t8) into the core memory. To load a different set of audio files the user can modify the files_io_7.gel file.

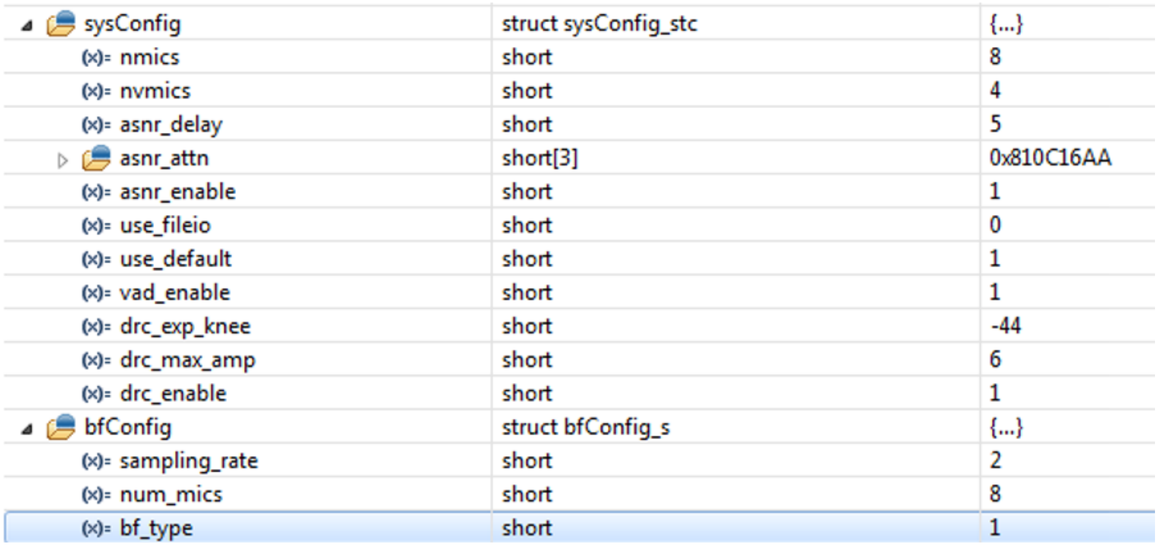
NOTE: The audio files are binary files in raw format. The user must verify that the directory and names of the audio files in the GEL are correct. See *Processor SDK RTOS Audio Pre Processing* [1] for information to use the GEL file.

3.2.1.3 Changing the Beamforming Structures for Different Geometry

NOTE: The following is only relevant if the user wants to change any of the parameter in the OMAPL137_bf_rt project, such as the number of microphone and the geometry of the microphone array, the number of virtual microphones, or sampling rate. Any of these changes requires recalculating of the filter coefficients and rebuilding of the project with the new coefficients as described in Section 3.2.1.1.

Before downloading a non-default set of filter coefficients, the user should update the beamforming configuration structures. In the expression window of the debug perspective, write sysConfig (see Figure 15) and configure the following parameters (all other parameters must remain as default):

- nmics is the number of microphones or raw-data audio files. This value is set depending on the number of files used.
- nvmics is the number of virtual microphones. The default value is eight, which represents the eight directions that the MSS looks at; that is, there are eight systems of beamforming (BF) and ASNR. Each system represents a different direction, and the MSS chooses the best one. The user can choose a different number of virtual microphones.
- The second structure is bfConfig. The sampling rate value of one translates into 8000 samples per second. The sampling rate value of two translates into 16000 samples per second.
- The value of num_mics is the number of virtual microphones that are connected to the MSS algorithms, which is the same as in sysConfig.
- The value of bf_type describes the type of BF calculations. This value can be either fixed-point short (bf_type = 1) or float (bf_type = 2). Figure 15 shows the fixed-point algorithm option.



sysConfig	struct sysConfig_stc	{...}
nmics	short	8
nvmics	short	4
asnr_delay	short	5
asnr_attn	short[3]	0x810C16AA
asnr_enable	short	1
use_fileio	short	0
use_default	short	1
vad_enable	short	1
drc_exp_knee	short	-44
drc_max_amp	short	6
drc_enable	short	1
bfConfig	struct bfConfig_s	{...}
sampling_rate	short	2
num_mics	short	8
bf_type	short	1

Figure 15. Setting sysConfig and bfConfig

3.2.1.4 Changing the Filter Coefficients

The beamforming filter coefficients depend on the geometry of the microphone array. The synthetic, raw audio data files that are part of this project in directory *C:\ti\processor_sdk_rtos_omapl137_xx_xx_xx_xx\demos\audio-preprocessing\common\t8* were generated under the assumption that the microphones are along a circular array of seven microphones. The filter coefficients in the this project were calculated accordingly. Should the user wish to use a different microphone array or use prerecorded raw data recorded from microphones in a different geometry, new filter coefficients are necessary. Section 3.2.1.5 describes how to calculate a new set of filter coefficients for the geometry of the microphone array. The new filters' coefficients buffers are updated and the project should be rebuilt. See *Processor SDK RTOS Audio Pre-Processing* [1] for instructions to add new filter coefficients.

3.2.1.5 Calculating Filter Coefficients

As was stated in Section 3.2.1.4, the beamforming filter coefficients depend on the geometry of the microphone array and the angle of the direction of the source with respect to the microphone array. bfgui.exe is a tool to generate beamforming filter coefficients and is part of the AER library in directory *AER\AER_64\tools\bf_tool* (AER is the directory where the user installed the AER library). A user's guide for the beamforming design tool bfgui.pdf is in the same directory as well. The user is strongly encouraged to read bfgui.pdf because it gives insight into the general theory of beamforming.

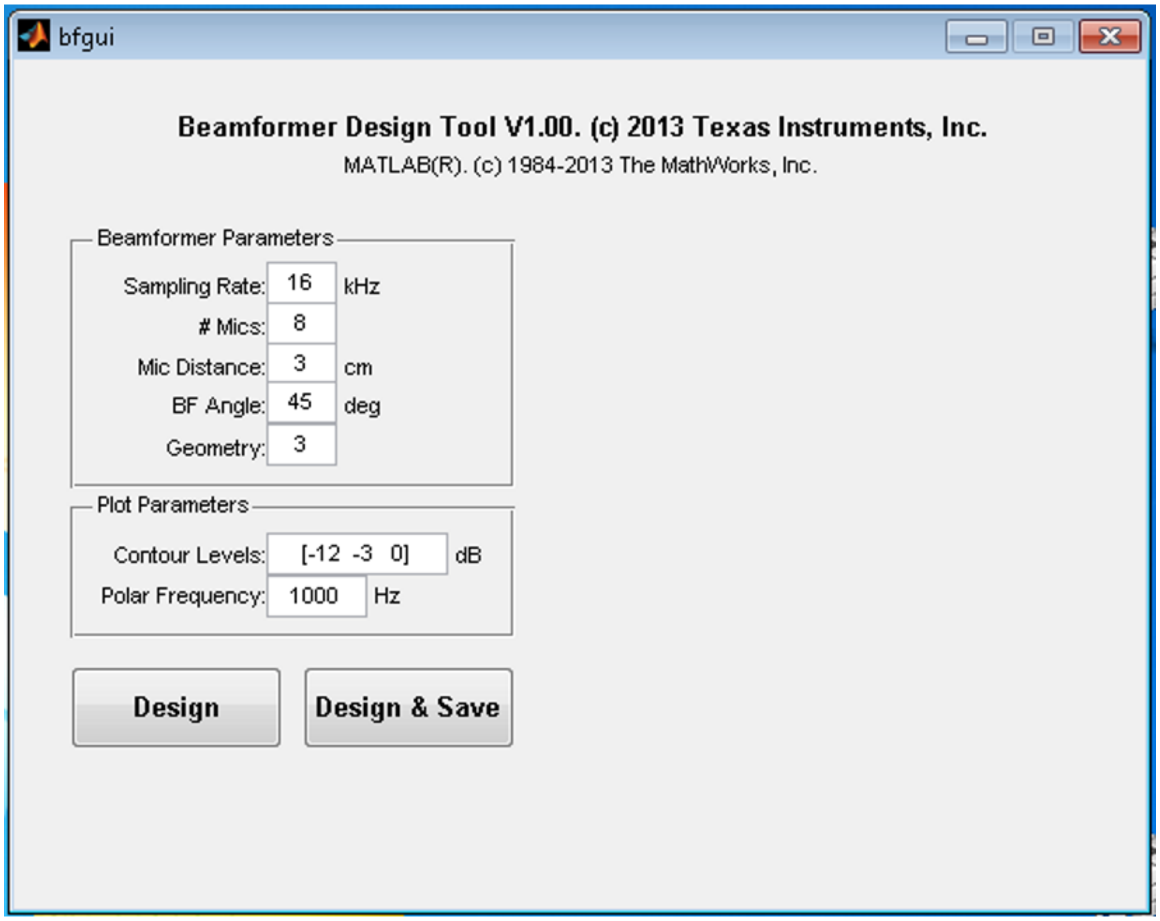
Upon starting bfgui.exe, the user should configure the following values, as shown in Table 4.

Table 4. Beamforming Design Tool Values

VALUE	COMMENTS
Sampling rate (KHz)	This TI Design uses 16 (16000). The default value of the tool is 8000.
Number of microphones	For using CMB (the real-time project), seven microphones are used. The offline project uses eight microphones.
Microphone distance	Distance in centimeters between two adjacent microphones. Equal distance between any two microphones is assumed. For linear array, it is the linear distance, and for circular array, it is the linear distance of the chord between two microphones.
BF angle	The utility generates a set of filters for a single angle of arrival; that is, for a single directional virtual microphone. For a system with multiple directional virtual microphones like the one that is used in this TI Design, the utility must be called multiple times, each time for a different angle. The real-time project defines eight directional virtual microphones for the following angles: 0°, 45°, 90°, 135°, 180°, 225°,270°, and 315°.
Geometry	Microphone array geometry: 0 for 1D linear, 1 for 2D linear, 2 for 2D rectangular, and 3 for circular. Using CMB requires geometry be set to 3.
Contour levels	Required for graphical illustration. Leave as default.
Polar frequency	Required for graphical illustration. Leave as default.

The number of virtual-directional microphones was set to eight so that every 45° (360° divided by 8) is a virtual-directional microphone. The trade-off is the more virtual-directional microphones, the better the focus on the desired audio source and better elimination of undesired sounds and clutter noises. On the other hand, the processing load of the beamforming and the ASNR depends linearly on the number of virtual-directional microphones.

Following the instructions in the user's guide (bfgui.pdf), configure the filter coefficients tool for eight microphones with circular geometry with 3-cm equal distance between any two microphones. This filter is for 45° of arrival. Figure 16 shows the configuration of the filter generation tool. The filter coefficients are stored in filterCoeff.log.



bfgui

Beamformer Design Tool V1.00. (c) 2013 Texas Instruments, Inc.

MATLAB(R). (c) 1984-2013 The MathWorks, Inc.

Beamformer Parameters

Sampling Rate:	16	kHz
# Mics:	8	
Mic Distance:	3	cm
BF Angle:	45	deg
Geometry:	3	

Plot Parameters

Contour Levels:	[-12 -3 0]	dB
Polar Frequency:	1000	Hz

Figure 16. Configuration of Filter Generation Tool

3.2.1.6 *Modifying the Microphones for audioAnalogLoopbackTest*

The microphones in the CMB are sampled 16000 times a second and 24-bit padded in 32-bit (4 bytes) each sample. The eight microphones are interleaved and sent through a McASP port to the C6747.

When the streaming data accumulates 10 ms of data, the C6747 DSP starts processing the data. In 10 ms each microphone samples 160 times, so the total size of the processing buffer is 1280 (160 × 8), 32-bit values or 5120 bytes.

The file mcasp_cfg.c in directory *SDK_DIRECTORY\pdk_omapl137_1_0_1\packages\ti\addon\cmb\test\evmOMAPL137\analog\loopback\src* (SDK_DIRECTORY is the directory where Processing SDK was installed) controls the internal C6747 stream connection. The BUFLEN value represents the number of samples from two microphones in 10-ms frame, that is 320 samples. BUFSIZ represents the number of bytes in the two microphones in 10 ms, which is 320 × 4. The pointer tempRxPtr points to the current streaming data. The eight microphones are interleaved in the following way:

- BUFLEN 32-bit words of microphone one and microphone two interleaved; that is, first value microphone one, first value microphone two, second value microphone one, and so on.
- BUFLEN 32-bit words of microphone five and microphone six interleaved; that is, first value microphone five, first value microphone six, second value microphone five, and so on.

- BUFLEN 32-bit words of microphone four and microphone three interleaved; that is, first value microphone four, first value microphone three, second value microphone four, and so on.
- BUFLEN 32-bit words of microphone eight and microphone seven interleaved; that is, first value microphone eight, first value microphone seven, second value microphone eight, and so on.

The pointer tempTxPtr points to the stream that is sent to the onboard codec. The left and right channels of the codec are interleaved as follows: first left channel, first right channel, second left channel, second right channel, and so on.

The code that copies data from tempRxPtr to tempTxPtr starts at line 546 of mcasp_cfg.c (in the current release; this may change for future releases) where there is a for loop that loops BUFLEN two times. The memcpy instruction copies one 32-bit (4 bytes) value from the tempRxPtr to tempTxPtr. The following illustrates how to choose the output microphones.

3.2.1.6.1 Connecting the Left-Channel Linked Microphone

The first memcpy in the for loop copies a microphone stream that is linked to the left channel of the onboard codec. The source code specifies multiple microphones streams. The user should comment out all memcpy except the one microphone that will be linked to the left channel of the codec. The default microphone is microphone number one.

Following the change of a microphone, the project must be rebuilt.

3.2.1.6.2 Connecting the Right-Channel Linked Microphone

The second memcpy in the for loop copies a microphone stream that is linked to the right channel of the onboard codec. The source code specifies multiple microphone streams. The user should comment out all memcpy except the one microphone that will be linked to the right channel of the codec. The default microphone is microphone number eight.

Following the change of a microphone, the project must be rebuilt.

3.2.1.7 Benchmarks

Table 5 shows a summary of the necessary MIPS-count measurements for offline calculations of beamforming where the sampling rate is 16 KHz, 16-bits for a sample. Two configurations were benchmarked: 8 virtual microphones (45° apart) and 12 virtual microphones (30° apart).

Table 5. MIPS Count Summary

SYSTEM CHARACTERISTICS	MIPS
OMAP-L137 SK 8 virtual microphones	40 MIPS
OMAP-L137 SK 12 virtual microphones	60 MIPS

3.2.2 Test Results

Figure 17 and Figure 18 show the test results of running the demonstration and capturing clean and unclean audio from line in to a PC with Audacity. The left channel is clean and the right unclean.

If a system was designed with a voice recognition engine, the clean audio would be fed into that engine. Further spectral analysis can be conducted on the results to evaluate the types of frequencies filtered during the tuning of a final design.

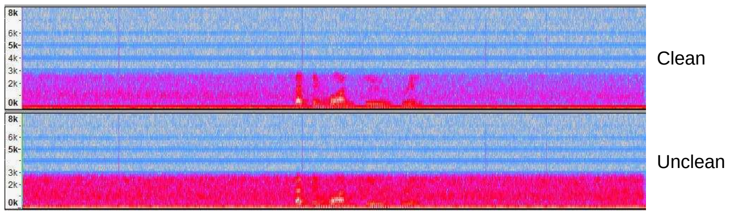


Figure 17. Test Results: Spectogram

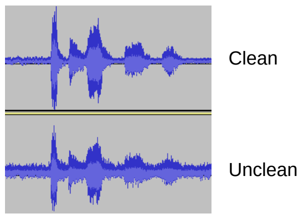


Figure 18. Test Results: Waveform

4 Design Files

4.1 Schematics

To download the schematics, see the design files at TIDEP-0099.

4.2 Bill of Materials

To download the bill of materials (BOM), see the design files at TIDEP-0099.

4.3 PCB Layout Recommendations

4.3.1 Layout Prints

To download the layer plots, see the design files at TIDEP-0099.

4.4 Altium Project

To download the Altium project files, see the design files at TIDEP-0099.

4.5 Gerber Files

To download the Gerber files, see the design files at TIDEP-0099.

4.6 Assembly Drawings

To download the assembly drawings, see the design files at TIDEP-0099.

5 Software Files

To install the software files, download and install the Processor SDK RTOS for OMAP-L137.

6 Related Documentation

1. Texas Instruments, *Processor SDK RTOS Audio Pre-Processing* , Wiki Page
2. Texas Instruments, *Processor SDK RTOS CMB AddOn* , Wiki Page
3. Texas Instruments, *Processor SDK RTOS Getting Started Guide* , Wiki Page
4. Texas Instruments, *Processor SDK RTOS Software Developer Guide* , Wiki Page
5. Texas Instruments, *Rebuilding the PDK* , Wiki Page
6. Texas Instruments, *Processor SDK RTOS Install In Custom Path* , Wiki Page
7. Texas Instruments, *Processor SDK RTOS Setup CCS* , Wiki Page
8. Chen, Joe C., Kung Yao, and Ralph E. Hudson. *Acoustic Source Localization and Beamforming: Theory and Practice* . EURASIP Journal on Advances in Signal Processing 2003, no. 4 (2003): 359-70.
9. Wikipedia, *Beamforming* , Article
10. Texas Instruments, *Demonstrating Voice Preprocessing on the OMAP-L137* , Training Video

6.1 Trademarks

7 About the Author

LALINDRA JAYATILLEKE is a Digital Applications Engineer at TI specializing in embedded processing applications including audio processing systems. Lali earned a B.Sc.in Electrical Engineering from the University of the District of Columbia, Washington DC.

MING WEI is a senior software engineer at TI where he develops and supports the Processor SDK RTOS for Sitara and the DSP families of SOC devices. Ming brings his extensive experiences and knowledge in real-time systems, signal processing, and code optimization to this role. Ming earned B.Sc., M.Sc. and Ph.D. in computer sciences from Xi'an Jiao-tong University and University of North Texas.

BOBBY TUFINO is an audio systems engineer at TI, where he supports the Sitara and DSP families of SOC devices. He has spent over 15 years supporting customers in the home audio and automotive audio markets. Bobby earned a B.Sc. in audio engineering and a B.Mus. in music engineering technology from the University of Miami.

Revision A History

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

Changes from # Revision (June 2017) to A Revision	**Page**
• Changed link for PROCESSOR-SDK-RTOS-OMAPL13X	1
• Changed processor_sdk_rtos_omapl13x to processor_sdk_rtos_omapl137.	3
• Changed instances of PROCESSOR-SDK-RTOS-OMAP-L13X to PROCESSOR-SDK-RTOS-OMAP-L137 in Section 2.2.5.	6
• Changed C:\ti\pdk_omapl13x_xx_xx_xx\packages\ti\addon\cmb to C:\ti\pdk_omapl137_xx_xx_xx\packages\ti\addon\cmb in Section 2.2.5.2.	7
• Changed C:\ti\processor_sdk_rtos_omapl13x_xx_xx_xx_xx\demos\audio-preprocessing\file_demo_bios\omapl137 to C:\ti\processor_sdk_rtos_omapl137_xx_xx_xx_xx\demos\audio-preprocessing\file_demo_bios\omapl137 in Section 2.2.5.3.	7
• Changed PROCESSOR-SDK-RTOS-OMAPL13X to PROCESSOR-SDK-RTOS-OMAPL137 in Section 3.1.2.	17
• Changed *C:\ti\processor_sdk_rtos_omapl13x_x_xx_xx_xx* to *C:\ti\processor_sdk_rtos_omapl137_x_xx_xx_xx* in Section 3.2.1.1.	19
• Changed Figure 14.	19
• Changed pdk_omapl13x_1_0_1 to pdk_omapl137_1_0_1 in Section 3.2.1.1.	19
• Changed *processor_sdk_rtos_omapl13x_x_xx_xx_xx* to *processor_sdk_rtos_omapl137_x_xx_xx_xx* in Section 3.2.1.1.	19
• Changed *C:\ti\processor_sdk_rtos_omapl13x_xx_xx_xx_xx\demos\audio-preprocessing\common\t8* to *C:\ti\processor_sdk_rtos_omapl137_xx_xx_xx_xx\demos\audio-preprocessing\common\t8* in Section 3.2.1.1.3.	21
• Changed *C:\ti\processor_sdk_rtos_omapl13x_xx_xx_xx_xx\demos\audio-preprocessing\common\filters* to *C:\ti\processor_sdk_rtos_omapl137_xx_xx_xx_xx\demos\audio-preprocessing\common\filters* in Section 3.2.1.1.3.	21
• Changed *C:\ti\processor_sdk_rtos_omapl13x_xx_xx_xx_xx\demos\audio-preprocessing\file_demo_bios\omapl137* to *C:\ti\processor_sdk_rtos_omapl137_xx_xx_xx_xx\demos\audio-preprocessing\file_demo_bios\omapl137* in Section 3.2.1.2.2.	21
• Changed *C:\ti\processor_sdk_rtos_omapl13x_xx_xx_xx_xx\demos\audio-preprocessing\common\t8* to *C:\ti\processor_sdk_rtos_omapl137_xx_xx_xx_xx\demos\audio-preprocessing\common\t8* in Section 3.2.1.4.	23
• Changed *SDK_DIRECTORY\pdk_omapl13x_1_0_1\packages\ti\addon\cmb\test\evmOMAPL137\analog\loopback\src* to *SDK_DIRECTORY\pdk_omapl137_1_0_1\packages\ti\addon\cmb\test\evmOMAPL137\analog\loopback\src* in Section 3.2.1.6.	24
• Changed Processor SDK RTOS for OMAP-L13x to Processor SDK RTOS for OMAP-L137.	27