TI Designs: TIDEP-0088

Audio Pre-Processing System Reference Design for Voice-Based Applications Using 66AK2G02

Description

The TIDEP-0088 uses multiple microphones, beamforming, and other algorithms to clean up and extract clear speech from an environment containing noise sources. The rapid 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 and speech recognition. This design guide walks through running a demonstration on the 66AK2G02 device using a circular microphone board (CMB) and also discusses the various concepts used to clean up audio.

Resources

TIDEP-0088	Design Folder
TIDA-01454 (CMB)	Design Folder
PCM1864	Product Folder
EVMK2G	Tools Folder
PROCESSOR-SDK-RTOS-K2G	Tools Folder
TELECOMLIB	Tools Folder

TI E2E™ Community — ASK Our E2E Experts

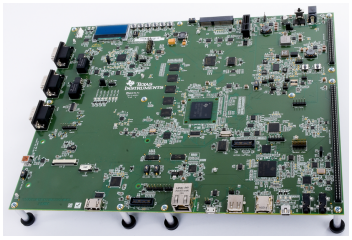
Features

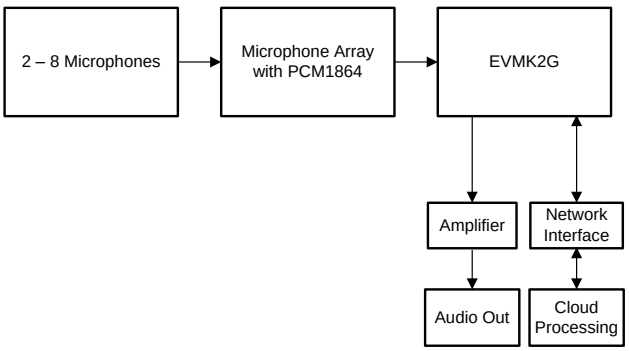
- Uses Single Digital Signal Processor (DSP) and Circular Microphone Board (CMB) to Extract Speech From a Noisy Environment
- Eliminates Background Noise From Audio Source
- CMB Provides 360° Coverage of Audio Source
- Enables Better Voice Recognition by Extracting Cleaner Speech
- Offers Complete System Reference Design Using TI-Provided Software, Evaluation Module, and CMB

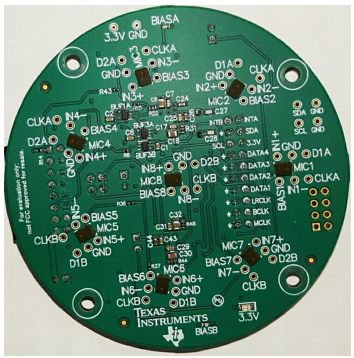
Applications

- 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)

Applications







1 System Description

The TIDEP-0088 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, reverberation, and acoustic echo.

The TI Design uses a beamforming algorithm to form virtual-directional microphones that point at the direction of the speaker or the desired audio source. The microphones then amplify the speech signal from the desired direction, which 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, such as dynamic-range compression.

Section 5 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 microphones' array and the processor must support streaming of multiple data inputs. The data rate depends on the application requirements. The TIDEP-0088 streams eight microphones mounted on a CMB (seven microphones on the circle and one in the center that is used for reference), samples in 16-bits at 16000 samples per second, and uses the analog-to-digital converter (ADC) PCM1864 for inter-IC sound (I2S) interface to the EVMK2G board.

The EVMK2G supports multiple audio output venues. The audio data can be sent to the network using an Ethernet port. An application that uses local processing, such voice-recognition remote-control appliances, can process the data locally. The TIDEP-0088 loops the output clear audio back into the left channel of the stereo audio output interface using a EVMK2G onboard audio codec. The reference microphone (located in the center of the circle) plays in the right channel. This structure enables the user to compare the quality of a standard microphone output to the quality of the system output.

This design includes full source code that can be modified to support various applications.

For a cloud-based, voice-activated digital assistant, the output signal can be sent to the network using the network interface that is supported by the device. Return audio signal from the network will be sent to the device codec to be played by a speaker.

Local (limited dictionary) voice recognition for voice-activated digital assistant applications control the system and use the DSP for voice recognition. The DSP in the EVMK2G is a high-performance, floating-point DSP clocked in at 600 MHZ and supports up to 8GB of external memory in addition to over 2MB of internal memory. The DSP has enough power and memory to support voice recognition of a limited dictionary.

Conference calls and other speech-processing applications require additional features (mixing of signals, better acoustic echo cancellation, and so on). As stated above, the DSP in the EVMK2G has enough power and memory to process more speech algorithms. Note that TI audio libraries include optimized audio algorithms that can be used by speech applications. See *Video Demonstrating Voice Preprocessing on the EVMK2G*[10] for a training video.

1.1 Key System Specifications

Table 1 shows the key system specifications.

Table 1. Key System Specifications

COMPONENT	DESCRIPTION	DETAILS
CMB	Eight microphone array with seven microphones mounted in an equal arc along the circle and one mounted in the center of the circle	Section 2.2.1
PCM1864	PCM1864 audio ADC provides I2S interfaces to the EVMK2G.	Section 2.2.2
66AK2Gx (K2G) evaluation module (EVMK2G)	Evaluation board based on the 66AK2G02 (C66 DSP + ARM® Cortex®-A15 processor)	Section 2.2.3
Processor software development kit (SDK) real time operating system (RTOS) (PROCESSOR-SDK-RTOS-K2G)	Standard TI software release for multiple devices, which includes tools, utilities, drivers, operating system support, optimized library, and more	Section 2.2.4
Executable K2G_bf_rt	DSP executable code that processes multiple microphones streaming audio and generates a virtual-directional microphone audio stream	Section 2.2.4.1
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 2.2.4.2
Executable offlineSignalProcessing (for debug and illustration 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 2.2.4.3
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 2.2.4.4
TI Audio Libraries (or TELECOMLIB)	TI-optimized audio processing AEC-AER and VOLIB libraries	Section 2.2.5
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

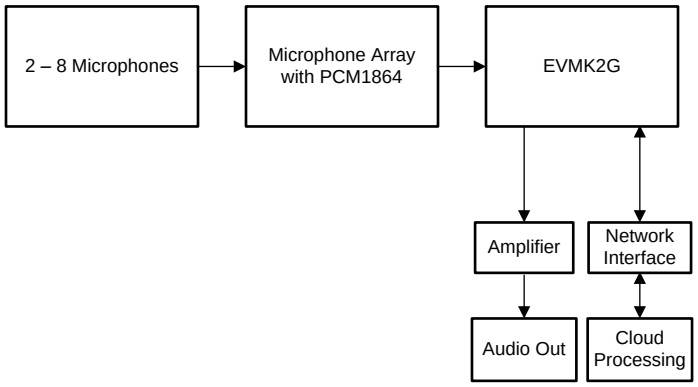


Figure 1. TIDEP-0088 Block Diagram

2.2 Highlighted Products

2.2.1 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 K2G audio expansion (AE) connection on the EVM. Schematics of the CMB are available in Section 6.1.

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

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 EVMK2G

The 66AK2Gx (K2G) evaluation module (EVMK2G) is an evaluation module that is based on the 66AK2G02 processor. For a full description of the EVMK2G, see ti.com's EVMK2G tools folder.

For more information on the 66AK2G02, see ti,com's 66AK2G02 product folder.

Figure 2 shows a block diagram of the 66AK2G0x.

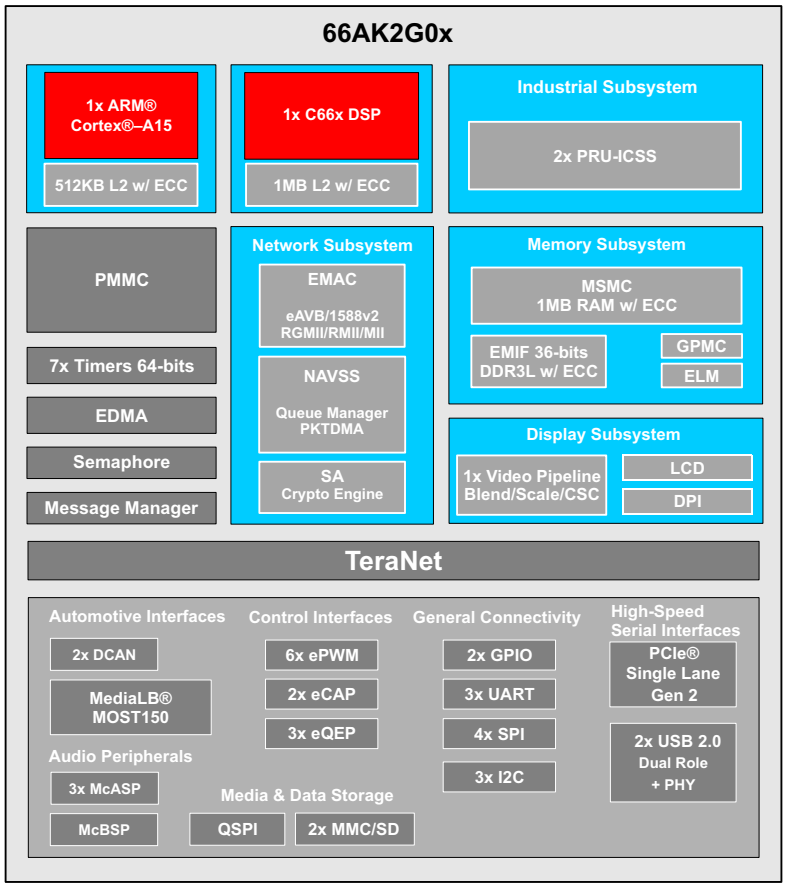


Figure 2. 66AK2G0x Block Diagram

2.2.4 PROCESSOR-SDK-RTOS-K2G

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 perspectives: LINUX and RTOS. The LINUX perspective (PROCESSOR-SKD-LINUX-K2G) contains LINUX tools, drivers, and utilities for ARM processors. The RTOS (PROCESSOR-SDK-RTOS-K2G) perspective contains RTOS, drivers, utilities, tools, high-level drivers, and optimized library as well as examples and demonstration projects. The TIDEP-0088 requires the Processor SDK RTOS perspective only.

Free download of PROCESSOR-SDK-RTOS-K2G is available at ti.com's Processor SDK K2G tools folder.

NOTE: Similar download pages are available for other devices (search *Processor SDK* and *device name*).

2.2.4.1 K2G_bf_rt Project

The K2G_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 EVMK2G 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 EVMK2G 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 2.2.4.2.

NOTE: K2G_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 (K2G_bf Project).

2.2.4.2 audioAnalogLoopbackTest Project

The cmb project 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 EVMK2G onboard audio codec. Note that the user can 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 EVMK2G onboard audio codec. All streams from the other six 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 which microphone is 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 EVMK2G.

2.2.4.3 K2G_bf Project

The K2G_bf project 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 and ASNR, and uses MSS to choose the best directional virtual microphone and store the microphone 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 4.1.2 describes how to generate filter coefficients.

2.2.4.4 Source Code and Makefiles

In addition to the three executables that are mentioned in Section 2.2.4.2, Section 2.2.4.1, and Section 2.2.4.3, 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 K2G_bf_rt and the K2G_bfprojects are located in the Processor SDK under the demos folder.

2.2.5 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 level that PDK is installed.

3 Getting Started Hardware and Software

3.1 Hardware

3.1.1 66AK2G02 GP EVMK2G Hardware Setup

Detailed steps on how to setup the EVMK2G are given in ti.com's 66AK2Gx (K2G) Evaluation Module tools folder.

Figure 3 shows the EVMK2G's layout and key components.

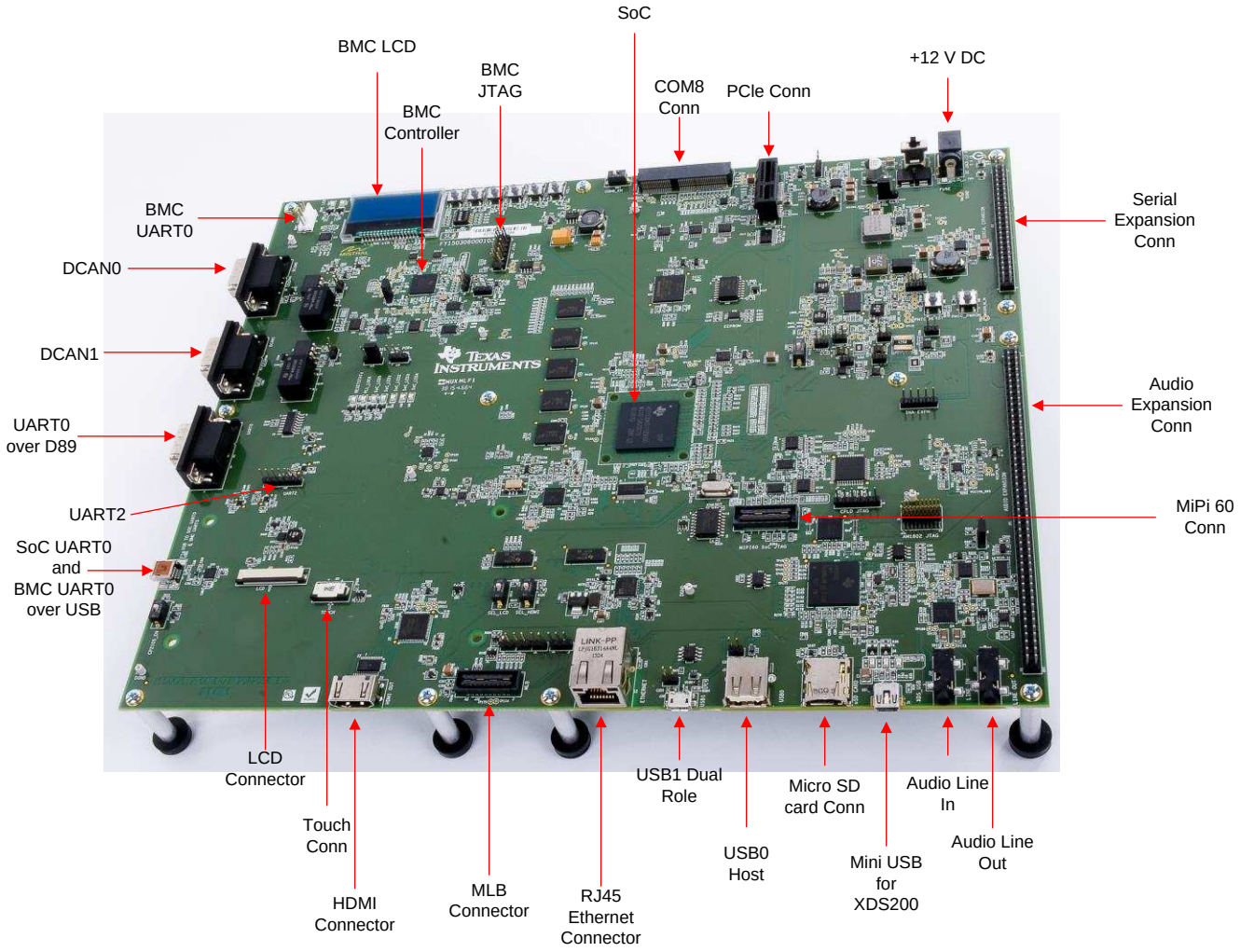


Figure 3. 66AK2G02 GP EVMK2G Layout

3.1.2 Connecting the CMB to EVMK2G

14 wires connect the audio expansion connector on the EVMK2G (see lower left of Figure 6) to the appropriate pins on the CMB.

Figure 4 shows a layout of the CMB.

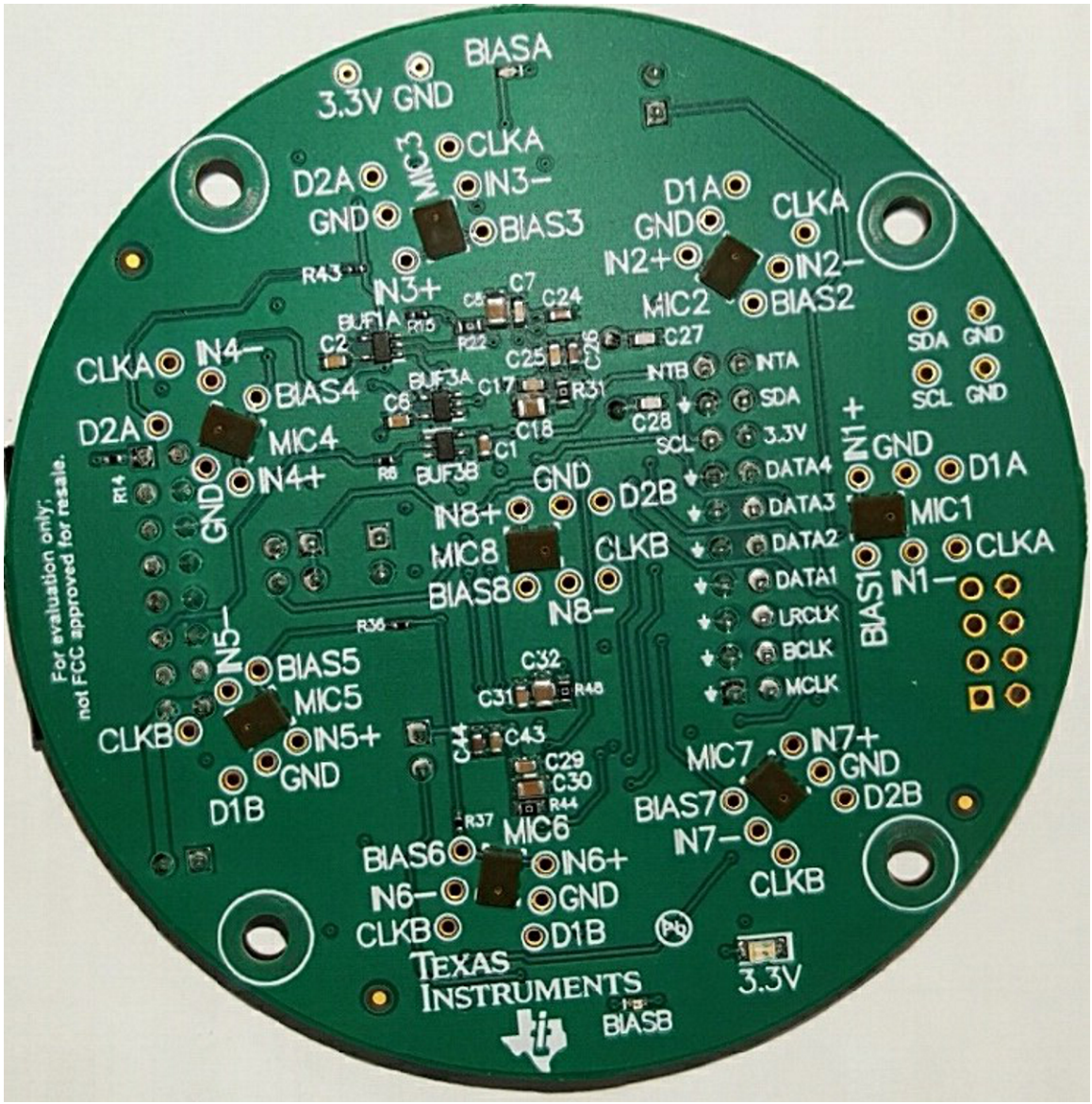


Figure 4. CMB Layout

Table 2 summarizes the connections between the EVMK2G and the CMB.

Table 2. Connections Between EVMK2G and CMB(1)

K2G_AE(2)	CMB PIN
K2G_AE_Pin48	CMB_DSP_MCLK
K2G_AE_Pin33	CMB_DSP_BCLK
K2G_AE_Pin41	CMB_DSP_LRCLK
K2G_AE_Pin29	CMB_DSP_DATA1
K2G_AE_Pin36	CMB_DSP_DATA2
K2G_AE_Pin47	CMB_DSP_DATA3
K2G_AE_Pin40	CMB_DSP_DATA4
K2G_AE_Pin98	CMB_DSP_SDA
K2G_AE_Pin97	CMB_DSP_SCL

(1) DATA and clock connections
(2) K2G_AE is the audio expansion connection on the K2G board, which is located on the lower right of Figure 3.

3.1.3 Power Connections to the CMB Card

Table 3 shows the power connections to the CMB card.

Table 3. Power Connections to CMB Card

K2G_AE	CMB PIN
K2G_AE_Pin10	CMB_DSP_3.3V
K2G_AE_Pin8	CMB_DSP_GND
K2G_AE_Pin38	CMB_DSP_GND
K2G_AE_Pin50	CMB_DSP_GND
K2G_AE_Pin100	CMB_DSP_GND

3.1.4 CMB Jumpers Configuration

Table 4 shows the configuration of the CMB jumpers.

Table 4. CMB Jumpers Configuration

PIN		PARAMETER
J3		ON
J8	Pins 1 and 2	ON
	Pins 3 and 4	OFF
J10		ON
J11		ON

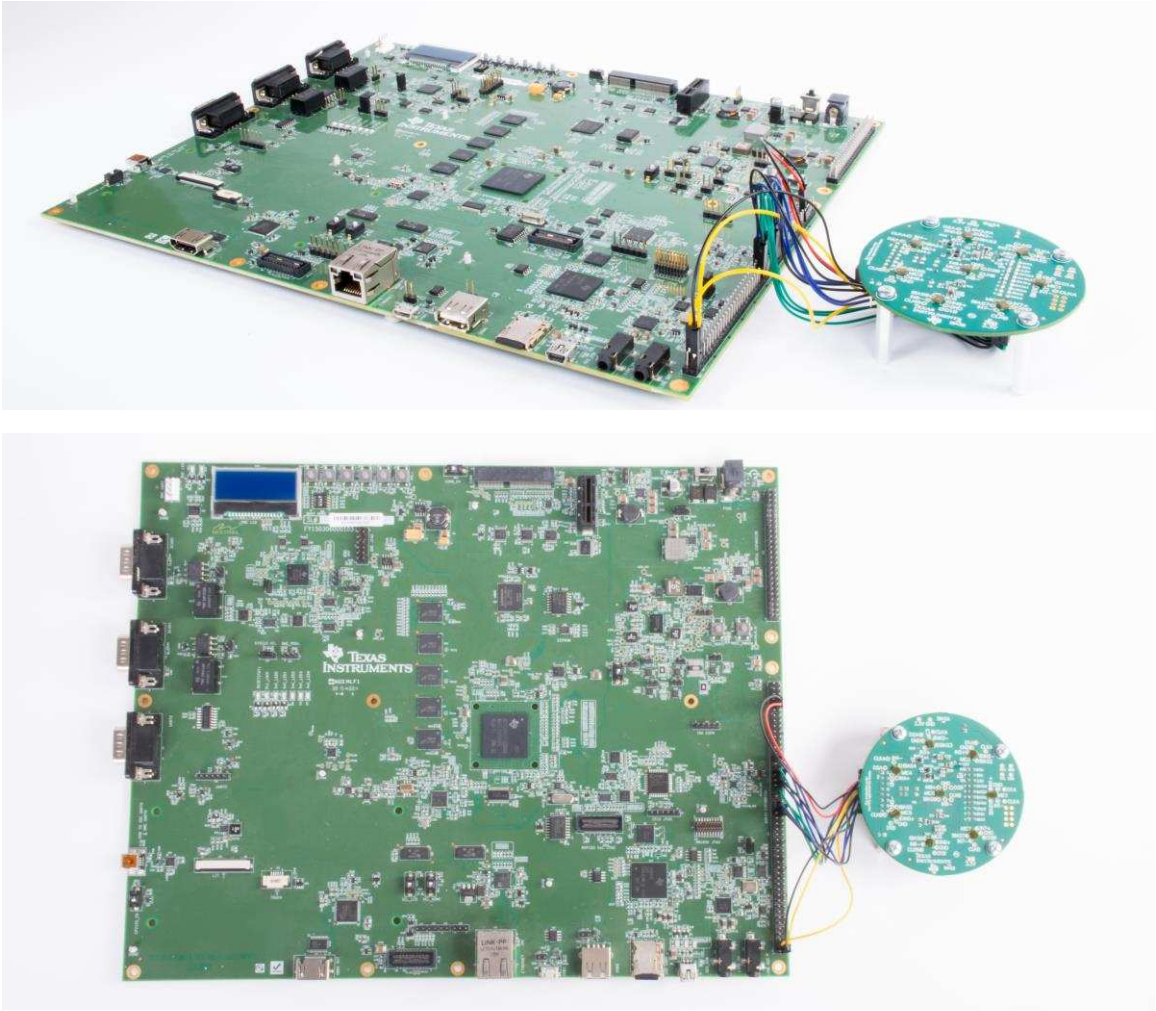


Figure 5. EVMK2G With CMB Connected

3.1.5 Connect Earphone

A stereo headset should be connected to the output audio. The audio connector *line out* is located at the bottom left of the EVMK2G next to the audio expansion connector (see Figure 6).

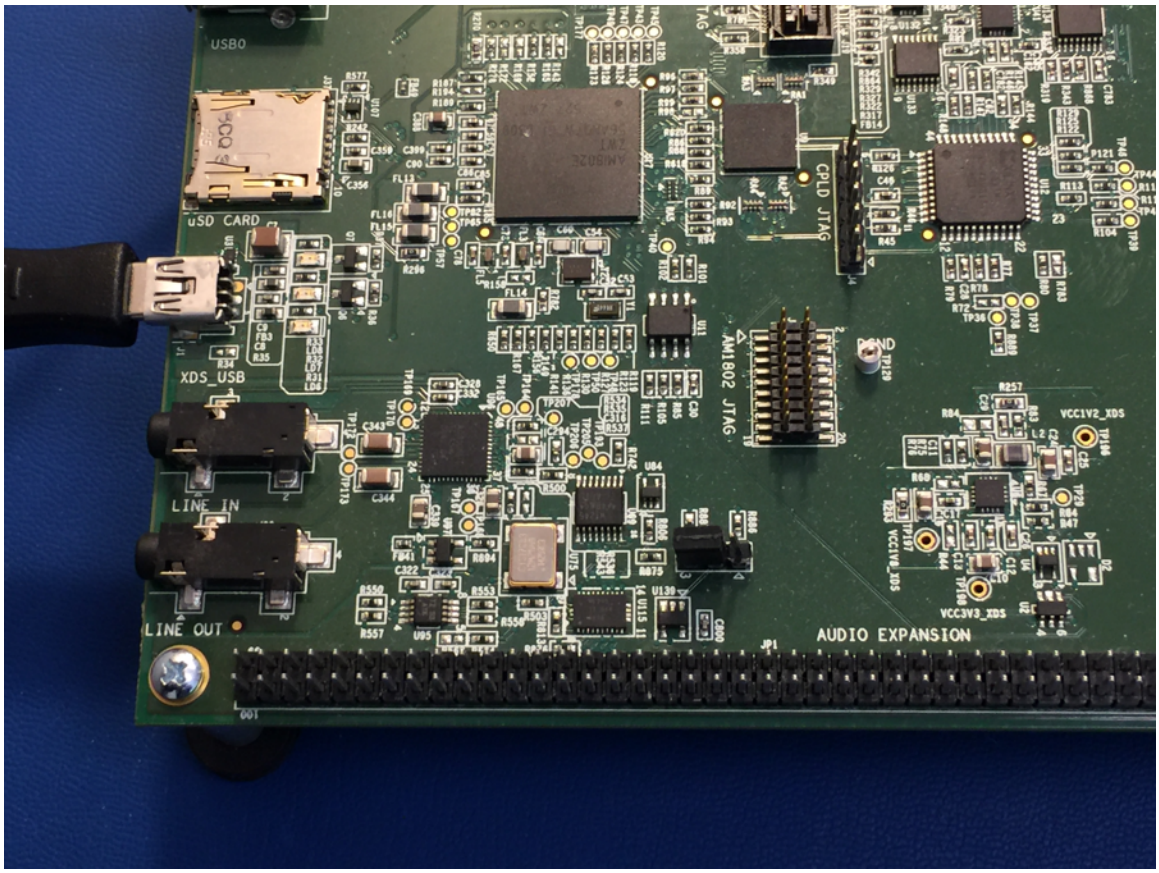


Figure 6. Audio Extension and Audio Line Out on EVMK2G

3.2 Software

3.2.1 Install Processor SDK and the Audio Libraries

The latest release of Processor SDK RTOS for K2G can be downloaded from ti.com's PROCESSOR-SDK-K2G 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-K2G installed and is familiar with the user guides. Processor SDK RTOS K2G is installed in the directory of the user's system named PROCESSOR_SDK_INSTALL_DIR.

This TI Design requires two audio libraries, as discussed in Section 2.2.5: AEC-AER library and VOLIB library. It is recommended that the two libraries be installed in a subdirectory of PROCESSOR_SDK_INSTALL_DIR. These libraries can be downloaded from ti.com's TELECOMLIB tools folder. Note that the C66 core uses the same AER library as C64+.

4 Testing

Testing of the beamforming system involves two steps. The first step is testing the complete audio path from the microphones to the EVMK2G onboard codec. The second step is testing the beamforming processing quality. Objective testing is done by connecting audio signal to a PC and using an audio tool like Audacity® to compare two audio streams and to gauge their quality. Chapter 6 of *Processor SDK RTOS Audio Pre-Processing*[1] details 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 EVMK2G onboard codec and comparing the quality.

The following subsections detail how to build, run, and test the three projects.

4.1 Testing Setup

4.1.1 Build and Run Executable

Three prebuilt executables are part of the TIDEP-0088 TI Design. The Processor SDK RTOS K2G 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 pages.

The first step in building a project in Windows is to configure environment variables. The file *pdksetupenv.bat* located in directory *PROCESSOR_SDK_INSTALL_DIR\pdk_k2g_1_0_4\packages* is a batch file that sets the environment variables. This file is necessary to run in *any new CMD window* before stating a build. 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* is the location where Processor SDK was installed (*PROCESSOR_SDK_INSTALL_DIR*).

Note that SDK_INSTALL_PATH directory has many subdirectories, which includes the PDK directory *pdk_k2g_1_0_4* (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 detailed information how to set the environment variables.

Building the K2G_bf_rt project and the K2G_bfproject requires configuring more environment variables. The file setupenv.bat is in the upper directory where processor_sdk_rtos_k2g_3_xx_xx_xx was installed. The user must configure SDK_INSTALL_PATH (line 54) 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 59) to the upper directory where CCS was installed.

4.1.1.1 Build and Run K2G_bf_rt Project

The K2G_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 EVMK2G. 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 Realtime Audio Pre-Processing*[1] wiki page details instructions to build and run the K2G_bf_rt project.

1. After building the executable, the user should load this executable to the K2G 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 K2G 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 Realtime 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 EVMK2G onboard codec and comparing the quality.

4.1.1.2 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 EVMK2G. The project is described in Section 2.2.4.2.

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 K2G DSP core using CCS as described in the wiki and run the code in a room with audio sources.
2. The K2G 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 4.2.
3. An easy way to subjectively verify that all microphones are working properly is to connect the EVMK2G 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.

4.1.1.3 Build and Run K2G_bf Project

The purpose of the K2G_bf.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 *PROCESSOR_SDK_INSTALL_DIR\processor_sdk_rtos_k2g_3_02_00_05\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 *PROCESSOR_SDK_INSTALL_DIR\processor_sdk_rtos_k2g_3_02_00_05\demos\audio-preprocessing\common\filters*.

Detailed instructions how to build and run the K2G_bf project is available in *Processor SDK RTOS Audio Pre-Processing*[1].

4.1.1.3.1 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.

4.1.1.3.1.1 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

4.1.1.3.1.2 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 *PROCESSOR_SDK_INSTALL_DIR\processor_sdk_rtos_k2g_3_02_00_05\demos\audio-preprocessing\file_demo_bios\k2g* 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.

4.1.1.3.2 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 K2G_bf 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 4.1.1.3.3.

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 7) 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 7 shows the fixed-point algorithm option.

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

Figure 7. Setting sysConfig and bfConfig

4.1.1.3.3 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 *PROCESSOR_SDK_INSTALL_DIR\processor_sdk_rtos_k2g_3_03_00_00\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 4.1.2 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.

4.1.2 Calculating Filter Coefficients

As was stated in Section 4.1.1.3.3, 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 5.

Table 5. 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	Needed for graphical illustration. Leave as default.
Polar frequency	Needed 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-directionalmicrophone. 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 8 shows the configuration of the filter generation tool. The filter coefficients are stored in filterCoeff.log.

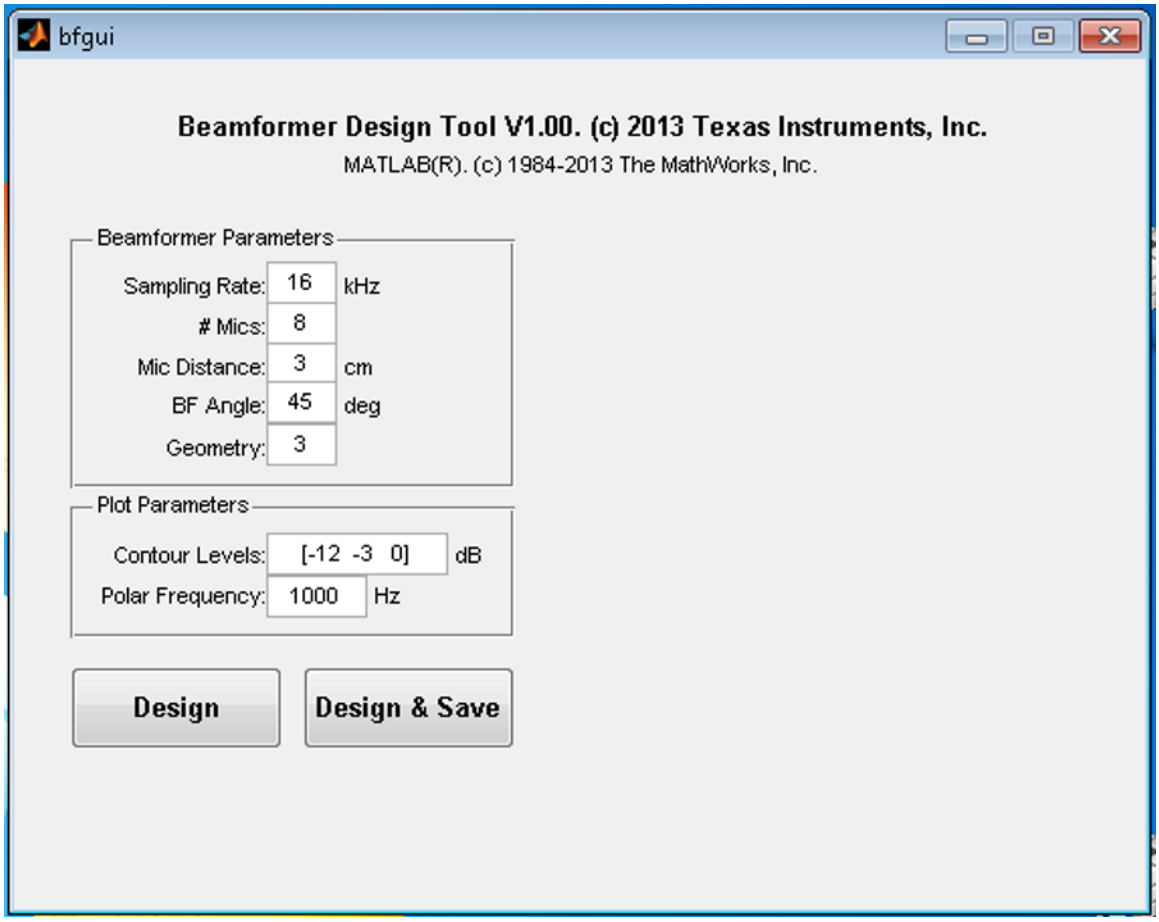


Figure 8. Configuration of Filter Generation Tool

4.2 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 EVMK2G. When the streaming data accumulates 10 ms of data, the K2G 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_k2g_1_0_4\packages\ti\addon\cmb\test\evmK2G\analog\loopback\src* (SDK_DIRECTORY is the directory where Processing SDK was installed) controls the internal K2G 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.

4.2.1 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.

4.2.2 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.

4.3 Benchmarks

Table 6 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 6. MIPS Count Summary

SYSTEM CHARACTERISTICS	MIPS
AM572 GP EVM 8 virtual microphones	39 MIPS
AM572 GP EVM 12 virtual microphones	58 MIPS

5 More About 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-0088 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 9 shows the distance and the phase difference between two microphones as a function of the direction of the signal arrival.

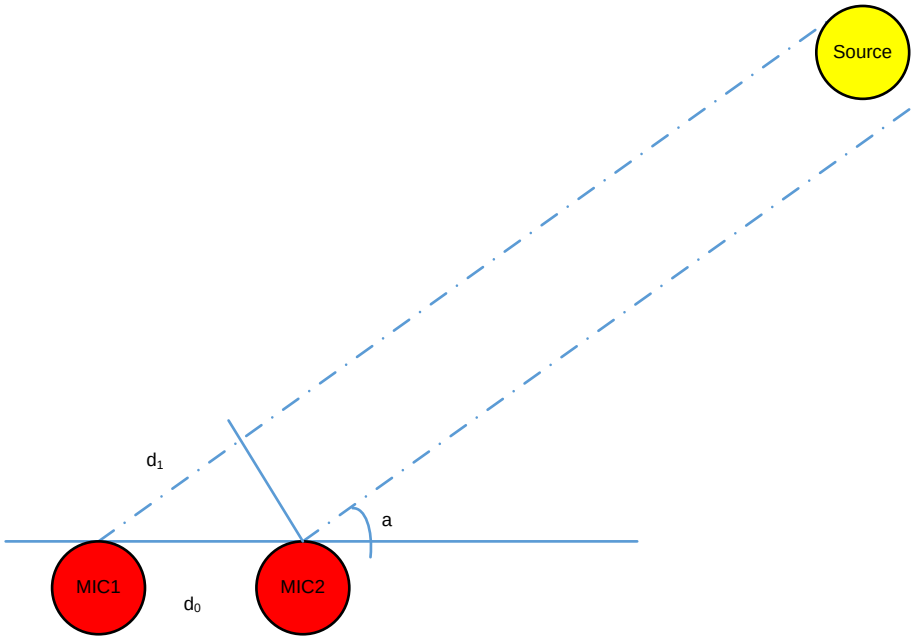


Figure 9. Differences in Distance, Time, and Phase Between Two Microphones

From Figure 9, d1 is calculated in Equation 1.

$$d_1 = d_0 \times \cos(\alpha) \quad (1)$$

The signal time difference, Δ_t , between mic1 and mic 2 is d1 divided by the speed of sound, as shown in Equation 2.

$$\Delta_t = \frac{d_1}{sos} \quad (2)$$

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

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

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 10 shows the reduction of noise as a function of frequency when the beamforming and the ASNR are applied.

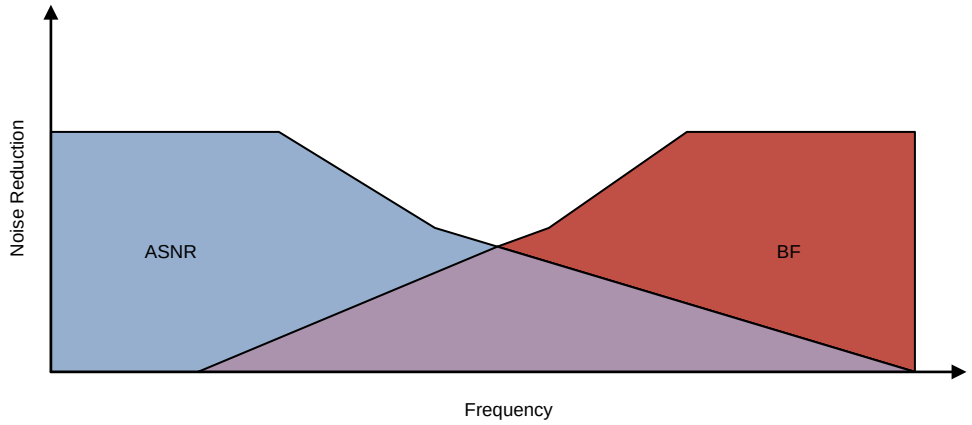


Figure 10. ASNR and BF Noise Reduction as a Function of Noise Frequency

5.1 Multi-Angle Beamforming

Figure 11 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 angle of arrival (AOA) and behaves like a directional microphone; therefore, it is called virtual microphone. An MSS algorithm chooses the best fit virtual microphone.

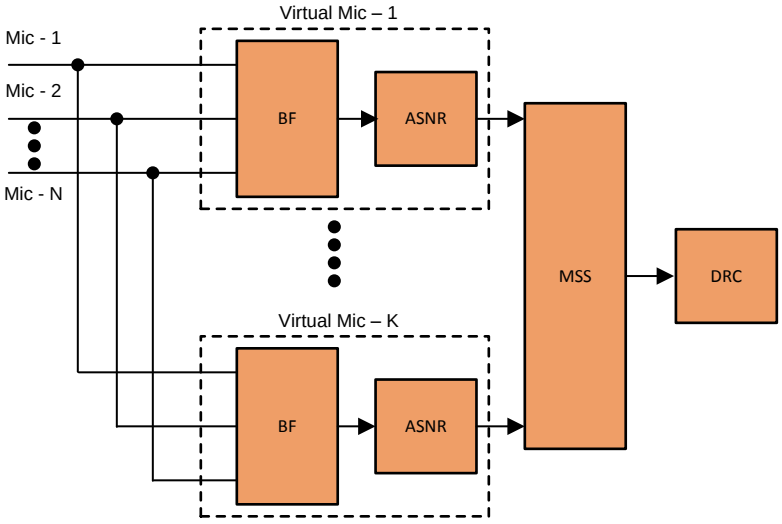


Figure 11. Multi-Angle Beamforming System

6 Design Files

6.1 Schematics

To download the Schematics for each board, see the design files at TIDEP-0088.

6.2 Bill of Materials

To download the Bill of Materials (BOM) for each board, see the design files at TIDEP-0088.

6.3 PCB Layout Recommendations

6.3.1 Layout Prints

To download the Layout Prints for each board, see the design files at TIDEP-0088.

6.4 Altium Project

To download the Altium project files for each board, see the design files at TIDEP-0088.

6.5 Gerber Files

To download the Gerber files for each board, see the design files at TIDEP-0088.

6.6 Assembly Drawings

To download the Assembly Drawings for each board, see the design files at TIDEP-0088.

7 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 EVMK2G*, Training Video

7.1 Trademarks

8 About the Authors

RAN KATZUR is a senior application engineer at TI where he supports the Sitara™ and the DSP families of system-on-a-chip (SOC) devices. Ran brings his extensive experiences and knowledge in parallel processing and optimization to this role. Ran earned B.Sc., M.Sc. and Ph.D. in applied mathematics from Tel- Aviv University

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.

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ('TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY 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 products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources 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 RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products http://www.ti.com/sc/docs/stdterms.htm), evaluation modules, and samples (http://www.ti.com/sc/docs/sampterms.htm).

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