Application Report
SPRAC97–June 2017

Sitara™ Linux ALSA DSP Microphone Array Voice Recognition

Michael Erdahl — *Catalog Processor Linux Software Applications Team*

ABSTRACT

Voice recognition applications requiring a superior user experience typically use a number of digital signal processing algorithms to filter, clean, and enhance microphone sampled audio. Frequently these systems will employ multiple microphones where more sophisticated algorithms can be used. These systems can leverage a Texas Instruments Digital Signal Processor to offload the complex real-time computation from the main applications processor. This application report details the hardware and software interfacing necessary to integrate a Texas Instruments Sitara ARM processor running the TI Processor Linux® SDK with the Texas Instruments C5000™ DSP evaluation module running a fixed-function four-microphone beamforming demo application as an Advanced Linux Sound Architecture (ALSA) device.

Contents

List of Figures

List of Tables

Trademarks

1 Introduction

This application report describes how to integrate a complete voice recognition system with silicon and software provided by Texas Instruments. The 4-microphone PCM1864 Linear Microphone Board captures voice and is processed by a TI C5517 low-power DSP. TI provides royalty-free firmware featuring a full complement of advanced audio processing algorithms as a starting point. The C5517 DSP outputs processed voice data via standard I2S digital audio interface, abstracting away the details on the DSP side, allowing easy integration with a Texas Instruments applications processor as an Advanced Linux Sound Architecture (ALSA) device. The Sitara AM335X processor with its rich set of peripherals and mainline Linux support enables a superior user experience, including high-definition audio, video, 3D graphics, realtime I/O applications with the PRU-ICSS, and wired or wireless network connectivity for cloud voice services.

1.1 Features

- Uses single digital signal processor (DSP) and microphone array to extract clear speech from a noisy environment
- DSP and microphone array are abstracted away as an ALSA device using the Linux simple-audio-card framework
- DSP algorithms eliminates background noise and clutter from audio source
- Enables better voice recognition by presenting clean speech audio to the recognition engine.
- Combines a complete system reference design using TI-provided software, evaluation modules, microphone array and an applications processor providing cloud connectivity

1.2 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)

1.3 Key system Specifications

Table 1 shows the key system specifications.

Table 1. Key System Specifications

Component	Description	Details
PCM1864LMBEVM	Two to four microphones of the LMB can be used with the C5517	Section 2.2
PCM1864	PCM1864 audio analog-to-digital converter (ADC) provides interfaces to the EVM. Each PCM1864 supports up to four audio microphones.	Section 2.2
TMDSEVM5517 DSP EVM	Evaluation board based on the C55x DSP	Section 2.2.3
BeagleBone Black	Community produced AM335X-based Sitara ARM processor running Linux	Section 2.2.4
Executable BF_rt_bios	DSP executable code that processes multiple microphones streaming audio and generates a virtual-directional microphone audio stream	Section 2.3.2
Application ource code and Code Composer Studio™ (CCS) projects	Source code for the data path unit test and for the applications that enables the user to modify or rebuild the code	[5]
TI audio libraries (or TELECOMLIB)	TI-optimized audio processing AEC-AER and VOLIB libraries	[5]
CCS version 6.1.3. CCS v6.2 and v7 are not supported at this time.	TI-integrated development environment (IDE) that is used to run the executables and can be used to build the executables. The project was built and tested with Code Composer Studio™ (CCS) version 6.1.3 and code generation tools CGT for 5500 version 4.4.1 or higher (It is assumed that the user is familiar with CCS.)	-
TI Processor Linux SDK	Linux kernel, filesystem and toolchain	Section 4.2

2 System Overview

2.1 Block Diagram

Figure 1 is a simplified system block diagram highlighting separate processing elements for microphone audio and cloud connectivity. DSP processed multi-channel microphone data is presented to the applications processor as regular audio data, abstracting away the details of beamforming, noise reduction and multi-source selection. This architecture can lower processing and memory requirements on the applications processor. Additionally, this two-chip solution provides a path to add a microphone array to an existing product, without changing a single line of application code.

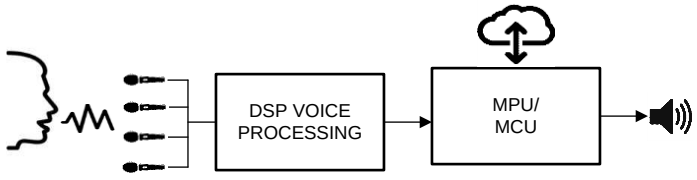

Figure 1. Generalized Voice Triggering System Block Diagram

2.2 Hardware Architecture

Figure 2 is a block diagram of the system described in this document. From left to right:

- Linear Microphone Board (LMB), TI p/n PCM1864LMBEVM, with four analog microphones arranged in a linear pattern, connected to the PCM1864 four-channel ADC with universal front end. The PCM1864 outputs four channels of digital audio on two Inter-IC Sound (I2S) channels.
- The C5517 DSP receives the digital audio from the LMB, performs several DSP algorithms in realtime and outputs to a single I2S channel.
- The AM335X applications processor receives the DSP processed audio for further action, such as sending voice commands a cloud assistant.

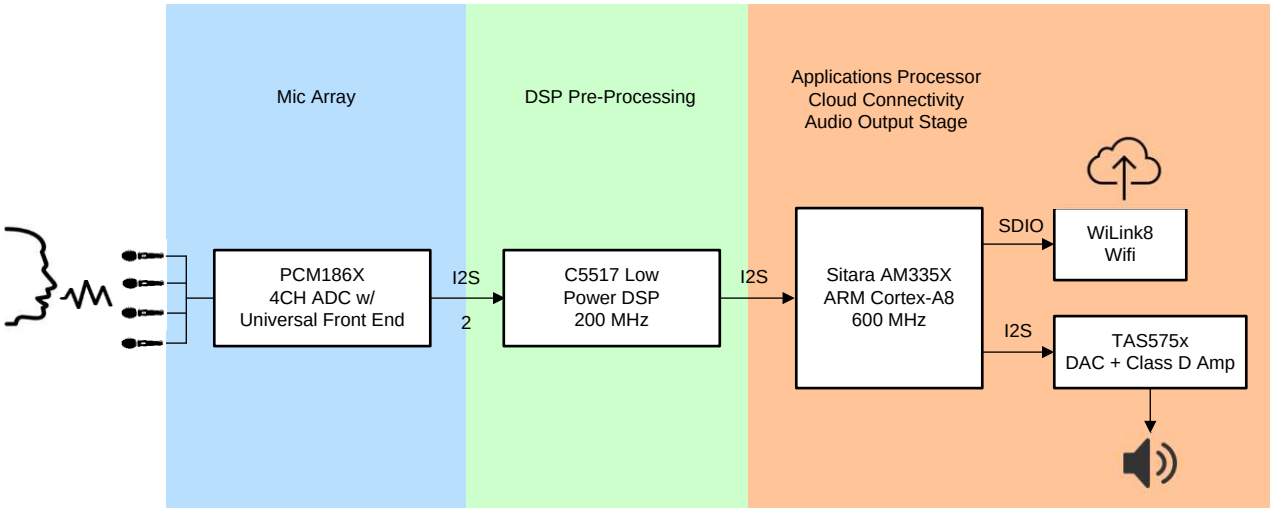

Figure 2. System Block Diagram

2.2.1 I2S digital Audio Format Explained

I2S is a digital audio transport standard used throughout the system to move digital audio. Keeping the audio signal digital saves cost by eliminating unnecessary digital-to-analog conversion while preserving sampled audio quality.

I2S is a master/ slave architecture. Slaves derive their clocks from the master. I2S typically comprises a minimum of three signals:

- BCK – Drives the bit conversion of the ADC. The bit clock frequency can be calculated by the following equation:
  - For example, 2 channels 32-bit audio sampled at 16 KHz:
    1. BCLK = nChannels * nBitsPerChannel * sample rate
    2. BCLK = 2 * 32 * 16000 = 10240000 or 1024 MHz
- LRCK – Indicates left vs right channel. Frequency of this signal is equal to the sample rate.
- DOUT – I2S audio channel data stream

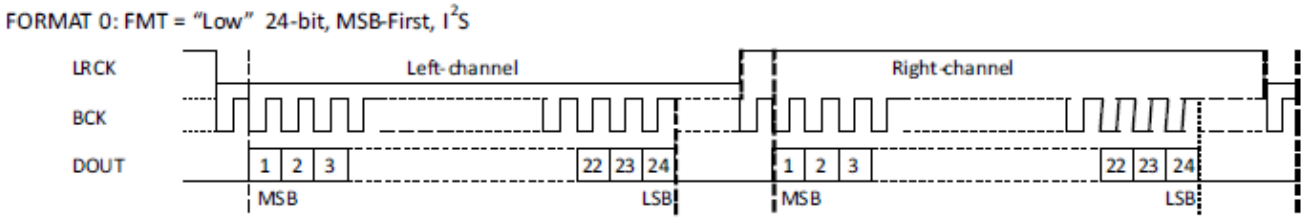

Figure 3. I2S Audio Format

2.2.2 PCM1864 Linear Microphone Board

The PCM1864 LMB is a low-cost, easy-to-use reference design for applications that require clear spoken audio, such as voice triggering and speech recognition. The microphone array captures voice signal and converts it to digital stream that can be used by DSP system to extract clear audio from noisy environments

The LMB is fitted with an array of four analog MEMS microphones, connected to the PCM1864 4-channel ADC. The PCM1864 is the I2S master in the system. The PCM1864 outputs the four channels of digitized audio on two I2S data lines, each carrying two channels (see Figure 4).

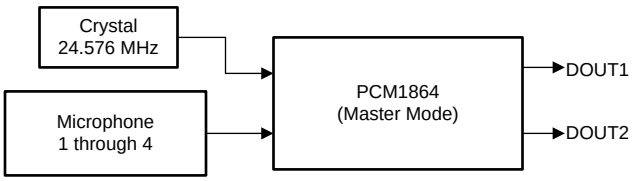

Figure 4. LMB Hardware Block Diagram

2.2.3 C5517 DSP

The C5517 DSP-based processor supports the real-time processing necessary along with the scalability desirable for voice triggering and related products. Figure 5 highlights the set of peripherals available on the C5517 DSP. For more information about Audio Pre-processing, voice triggering and post processing on C55x, see the TI Design TIDEP-0077 [3].

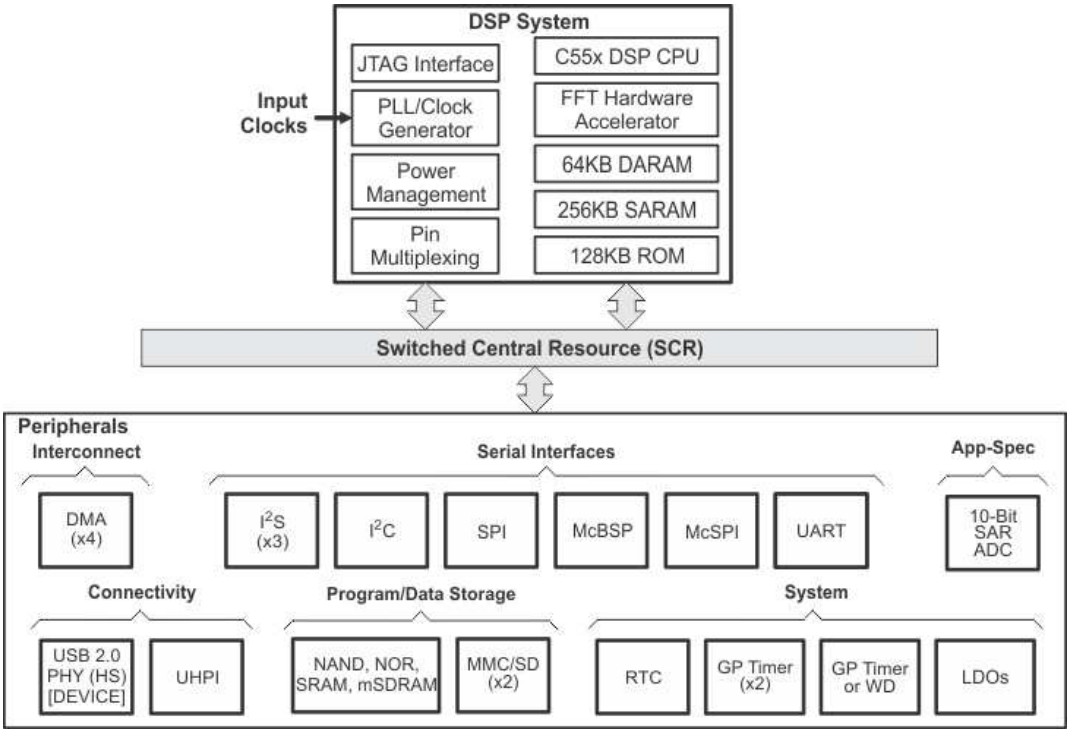

Figure 5. C5517 DSP Block Diagram

2.2.4 Sitara AM335X

The AM335X processor (see Figure 6) receives audio from the C5517 DSP as an I2S slave on one McASP, and sources digital audio as an I2S master on another McASP to an external audio output device. The AM335X is running the TI Processor Linux SDK, with the network stack providing cloud connectivity to voice services, and ALSA libraries for audio mixing, voice detection and audio playback.

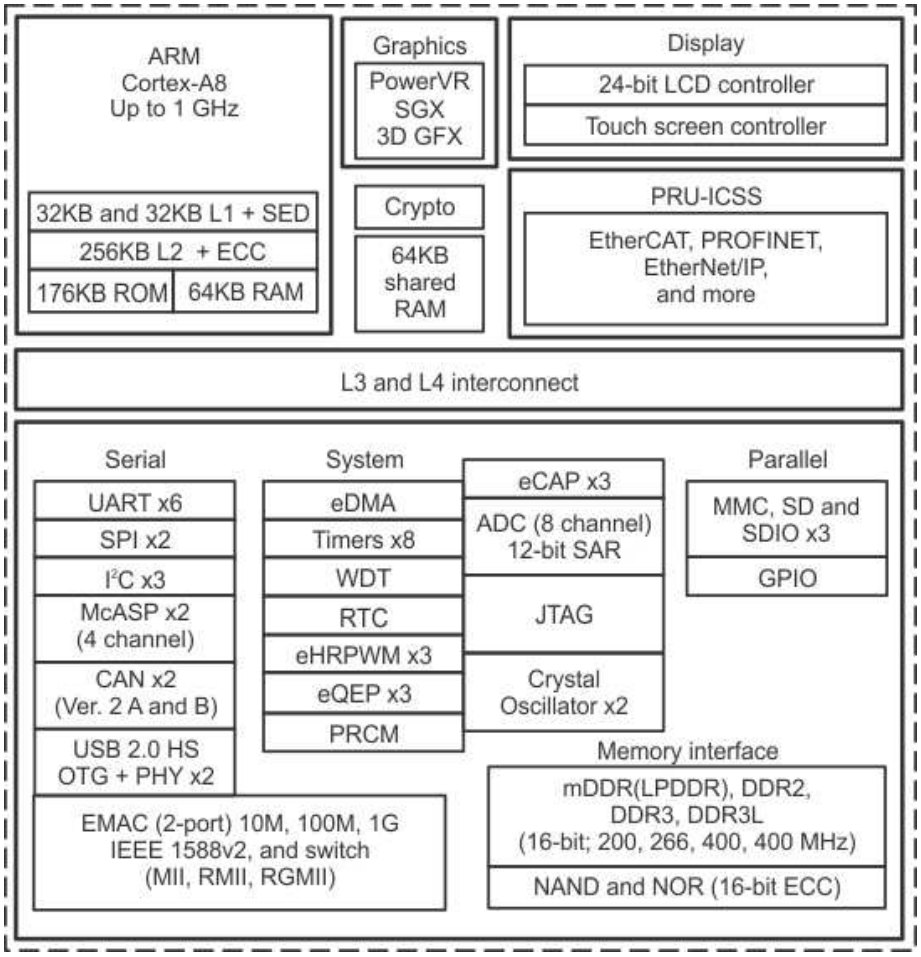

Figure 6. AM335X Sitara Applications Processor Block Diagram

2.3 Software Architecture

2.3.1 DSP Audio Pre-Processing (BF_rt_bios project)

The system software architecture on the DSP and ARM platforms are briefly described in the following subsections. The BF_rt_bios project is part of the C55xx_csl release. The project gets two or four audio streams from the microphone array and applies beamforming (BF), Adaptive Spectral Noise Reduction (ASNR), Multiple Source Selection (MSS), and Dynamic Range Control (DRC) to obtain a single, virtual directional microphone directed at speech and to remove clutter from a noisy environment.

The Wiki page C55x CSL Audio Pre-Processing [5] provides the most updated details on the project including how to build, run, and test the results.

Figure 7 shows the multi-angle beamforming that is running in the demo. Multiple BF 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 a virtual microphone. An MSS algorithm chooses the virtual microphone with the highest audio energy.

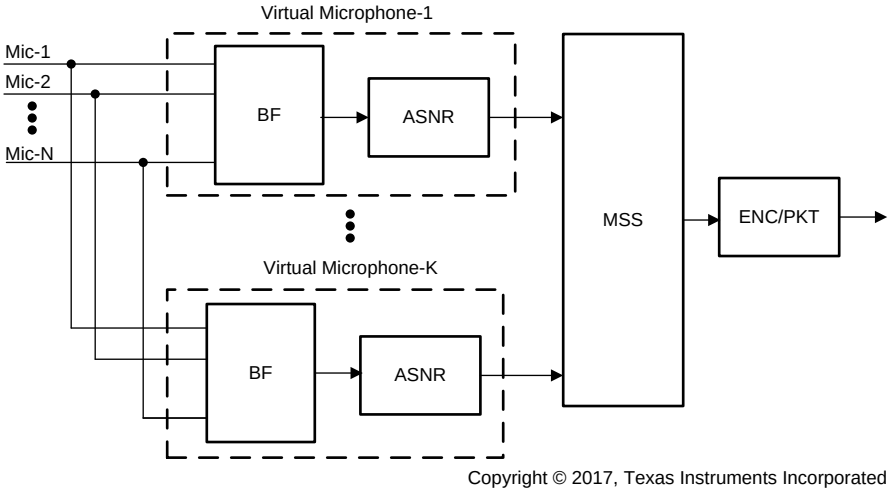

Figure 7. BF_rt_bios Project Block Diagram

2.3.2 ARM/Advanced Linux Sound Architecture

ALSA provides the interface to the available audio hardware. For more information about how ALSA relates to the TI Processor Linux SDK, see the following links: [6] [7].

Figure 8 highlights the audio portions of the Linux kernel driver architecture responsible for transporting digital audio from the hardware, through kernel space and on to user space.

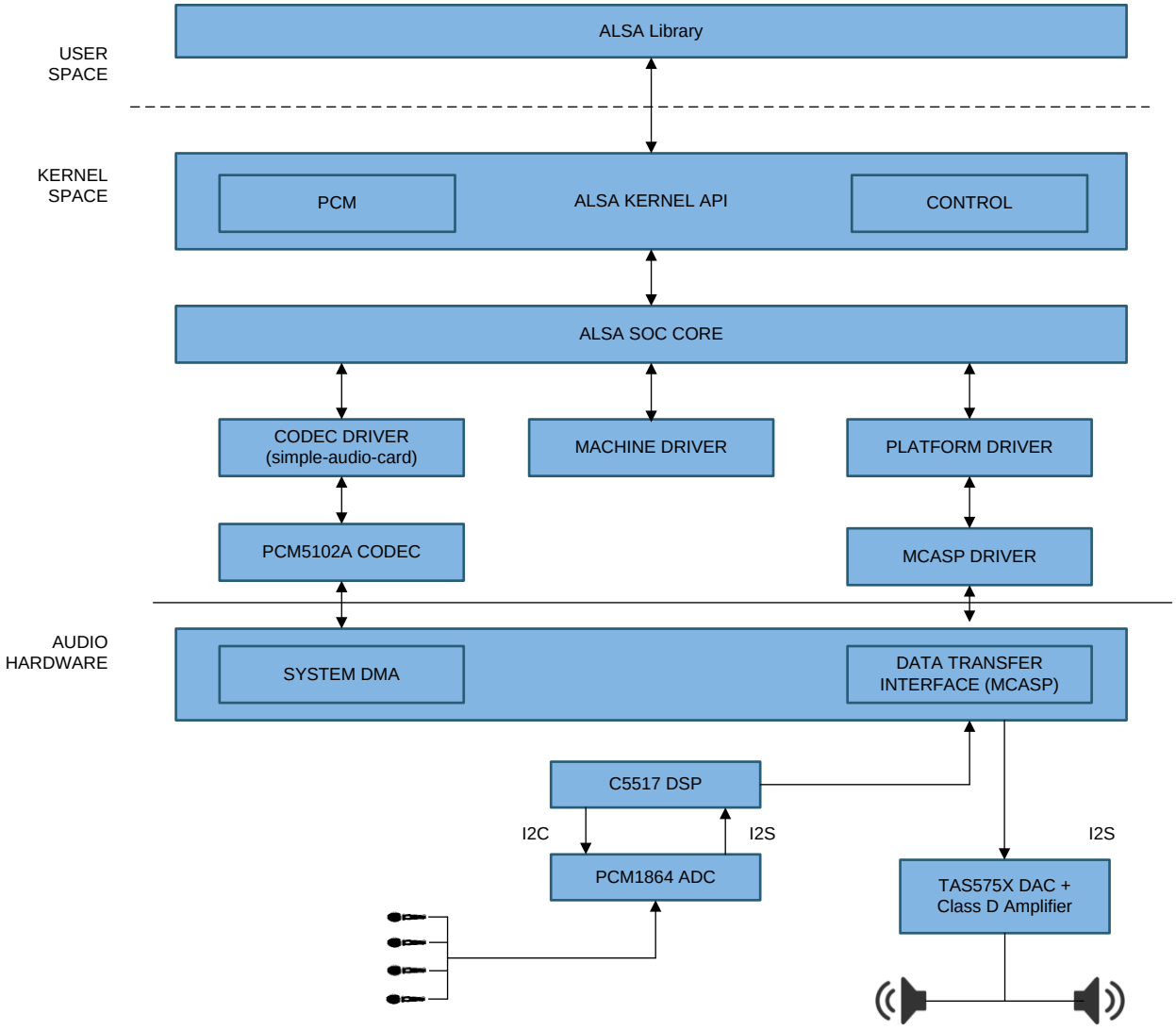

Figure 8. ALSA, Linux Kernel and Hardware Integration

3 Hardware integration

This section details the hardware interface between the LMB, C5517 EVM and BeagleBone Black

Evaluation boards used in this document:

- TMDSEVM5517EVM
- PCM1864LVMEVM
- BeagleBone Black

3.1 Linear Microphone Board and C5517 EVM Connections

Table 2 lists the connections required to interface the LMB and C5517 EVM.

Table 2. LMB to C5517 EVM Connections

Power		
Signal Name	**LMB**	**TMDSC5517 EVM**
3.3V	LMB_3.3V	J10_Pin9
GND	LMB_GND	J10_Pin5
MIC 1 and MIC 2		
I2C SCL	LMB_SCL	J14_Pin16
I2C SDA	LMB_SDA	J14_Pin20
I2S Bit Clock	LMB_BCLK	J27_Pin3 (no jumper)
I2S Frame Clock	LMB_LRCLK	J27_Pin4 (no jumper)
I2S Data 1	LMB_DATA1	J30_Pin2 (no jumper)
		J29_Pin1_Pin3 (jumper on)
		J29_Pin2_Pin4 (jumper on)
MIC 3 and MIC4		
I2S3 Bit Clock	I2S_BCLK	J31_Pin3
I2S3 Frame Clock	I2S_LRCLK	J31_Pin2
I2S3 Data 3	LMB_DATA3	J31_Pin1
		UART_EN (no jumper)

Figure 9 illustrates the header locations on the C5517 EVM.

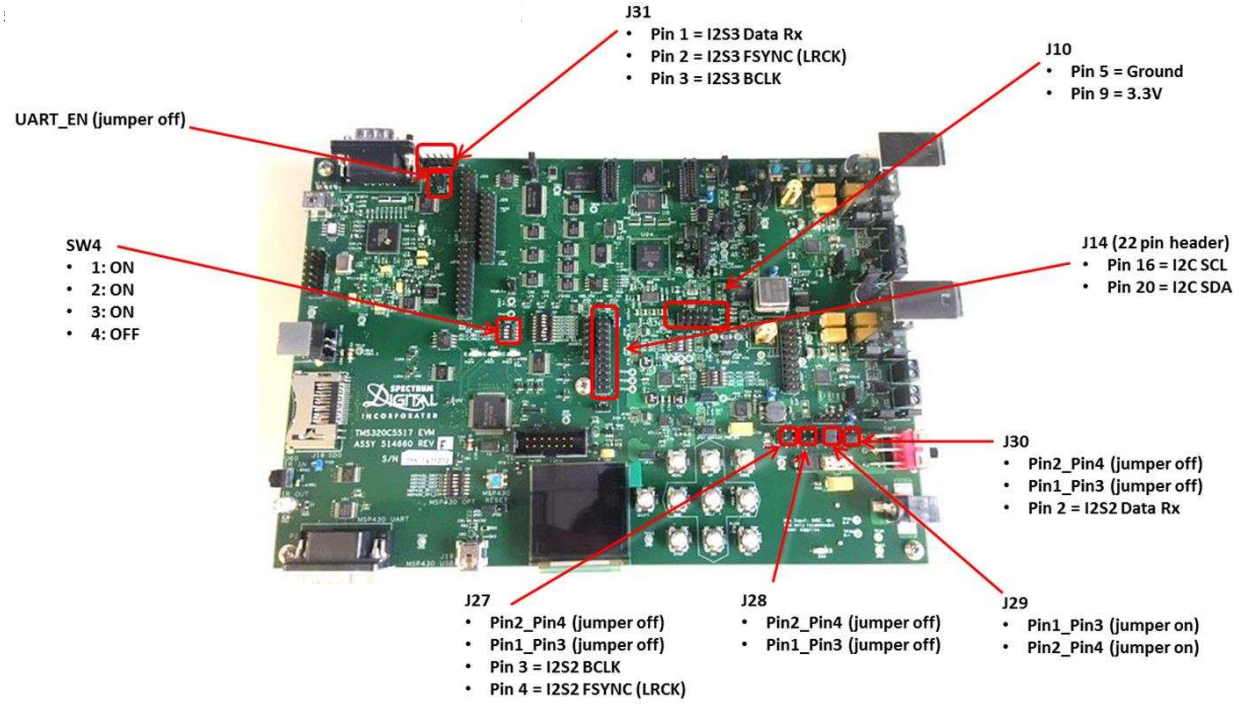

Figure 9. C5517EVM Header Locations

Table 3 lists the connections required to interface the C5517 EVM and BeagleBone Black. Connections to J29 should tap the existing jumper connection.

Table 3. C5517 EVM to BeagleBone Black Connections

Digital Audio Input (McASP1)		
Signal Name	TMDSC5517 EVM	BeagleBone
MCASP1_ACLKR (Bit Clock)	J29_Pin3	P9_42
MCASP1_FSR (Frame Sync)	J29_Pin4	P9_27
MCASP1_AXR0 (I2S Data)	J30_Pin1	P9_41
I2C (future)		
I2C SCL	TBD	P9_19
I2C SDA	TBD	P9_20

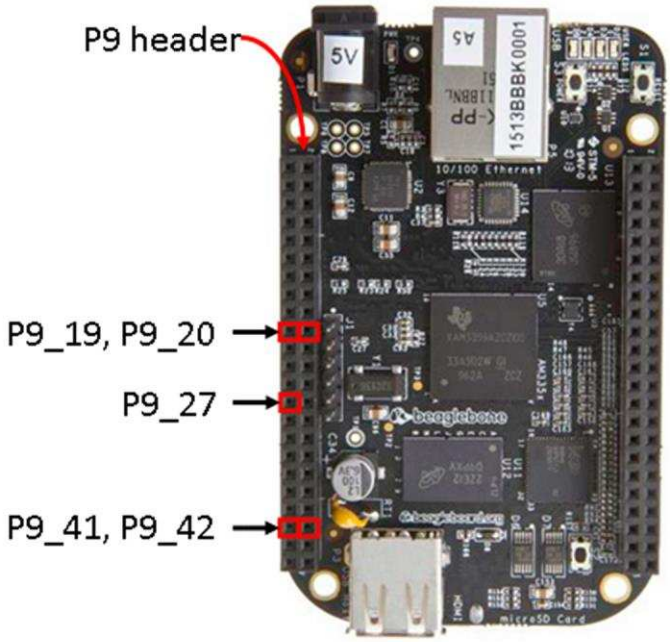

Figure 10. BeagleBone Black Pin Locations

4 Software Integration

This section describes the steps necessary to acquire, build, and deploy the various software components of the system described in this application report.

4.1 C55x CSL Audio Pre-Processing Firmware

The C55x CSL Audio Pre-Processing firmware is a pre-built binary. For bringing up the DSP firmware on the C5517 EVM, see the C55x CSL Audio Pre-Processing wiki [3] [5].

4.2 TI Processor Linux SDK integration

The following sections will cover integrating a digital audio source with the Linux kernel and ALSA.

Always develop using the latest version of the TI Processor Linux SDK when possible: http://www.ti.com/tool/PROCESSOR-SDK-AM335x. This system was developed and tested using TI Processor Linux SDK version 3.3.0.4.

4.2.1 Linux Kernel Modifications

Minor modifications to the Linux kernel must be made to support ALSA receiving the digital audio data output by the C5517 DSP.

For help with procedures in this section, see the TI Processor Linux SDK wiki: http://processors.wiki.ti.com/index.php/Linux_Kernel_Users_Guide. Another available resource is the E2E forum.

4.2.1.1 PCM5102A Dummy CODEC

The PCM5102A is a simple 2-channel DAC with I2S digital audio input. The PCM5102A driver is modified to support audio input, called a capture element. This device is called a “dummy CODEC” because it is only being used to facilitate audio data transfer from hardware (McASP) to the ALSA layer (see Figure 8).

4.2.1.1.1 Adding Capture Element

Because the PCM5102A is an output-only device, extend the driver’s functionality by adding a capture element.

Older kernels may not have the PCM5102A driver. Source can be found here: http://git.ti.com/ti-linux-kernel/ti-linux-kernel/blobs/ti-linux-4.9.y/sound/soc/codecs/pcm5102a.c (added capture element shown with existing playback element).

PATH	KERNEL/sound/soc/codecs/pcm5102a.c

```
#include <linux/init.h>
#include <linux/module.h>
#include <linux/platform_device.h>

#include <sound/soc.h>

static struct snd_soc_dai_driver pcm5102a_dai = {
    .name = "pcm5102a-hifi",
    .playback = {
        .channels_min = 2,
        .channels_max = 2,
        .rates = SNDRV_PCM_RATE_8000_192000,
        .formats = SNDRV_PCM_FMTBIT_S16_LE |
               SNDRV_PCM_FMTBIT_S24_LE |
               SNDRV_PCM_FMTBIT_S32_LE
    },
    .capture = {
        .stream_name = "Capture",
        .channels_min = 1,
        .channels_max = 2,
        .rates = SNDRV_PCM_RATE_8000_192000,
        .formats = SNDRV_PCM_FMTBIT_S16_LE |
               SNDRV_PCM_FMTBIT_S24_LE |
               SNDRV_PCM_FMTBIT_S32_LE
    },
};
```

4.2.1.1.2 Makefile Modification (older kernels)

For older kernels missing the PCM5102A driver, the Makefile below must be modified:

PATH	KERNEL/sound/soc/codecs/Makefile

```
snd-soc-pcm5102a-objs := pcm5102a.o
obj-$(CONFIG_SND_SOC_PCM5102A) += snd-soc-pcm5102a.o
```

4.2.2 Kernel Configuration

By default, the PCM5102A driver is not marked for inclusion in the kernel provided by the TI Processor Linux SDK. Menuconfig is typically used to select drivers for inclusion in the kernel.

The SDK toolchain PATH referred to in the following sections can be found in the TI Processor Linux SDK:

PATH	TI-PROCESSOR-SDK-LINUX/linux-devkit/sysroots/x86_64-arago-linux/usr/bin

4.2.2.1 Adding PCM5102A to Kconfig

Menuconfig options are built up with Kconfig files, located in the kernel source alongside related driver source files. Verify the PCM5102A driver option is available; without it, the driver will not show up in menuconfig.

Add the following entry to the Kconfig file, if missing:

PATH	KERNEL/sound/soc/codecs/Kconfig

```
config SND_SOC_PCM5102A
    tristate "Texas Instruments PCM5102a Dummy Codec Driver"
```

4.2.2.2 Selecting PCM5102A With Menuconfig

Enable the PCM5102A driver with the menuconfig make target. The following steps assume your Linux .config has not already been created by the proper defconfig; tisdk_am335x-evm_defconfig is used as an example below for the BeagleBone Black. Use the proper defconfig for the target platform.

PATH	KERNEL/

```
$ export PATH=${SDK_TOOLCHAIN_PATH}:$PATH
$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- tisdk_am335x-evm_defconfig
$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- menuconfig
```

In the menuconfig application, navigate to the “Texas Instruments PCM5102a Dummy Codec Driver” option, press ‘Y’ to mark the driver to be built into the Linux kernel.

```
> Device Drivers
   > Sound card support
      > Advanced Linux Sound Architecture
         > ALSA for SoC audio support
            > CODEC drivers
```

If the Texas Instruments PCM5102a Dummy Codec Driver does not appear as shown in Figure 11, verify the Kconfg file mentioned previously.

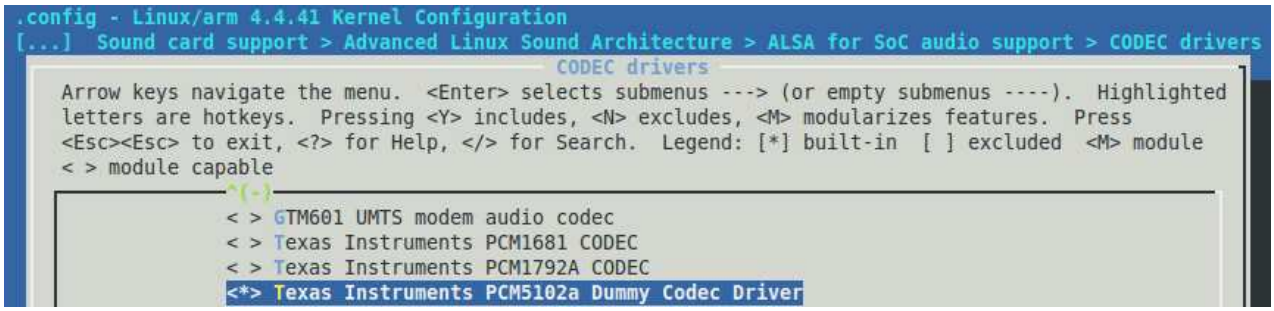

Figure 11. Menuconfig - PCM5102 Dummy Codec Driver

4.2.2.3 Building the Linux Kernel

Build the Linux kernel to include the PCM5102A driver into the kernel. The following make command assumes the PCM5102A device driver was marked to be built into the kernel. If the PCM5102A device driver was marked as a loadable module. For instructions on building and installing the Linux kernel modules, see the TI Processor Linux SDK documentation [9].

The following is an excerpt from the kernel build process, showing the PCM5102A device driver getting built – if the driver is not built, verify that the Makefile has been properly modified as detailed previously.

PATH	KERNEL/

```
$ export PATH=${SDK_TOOLCHAIN_PATH}:$PATH
$ ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- make -j8 zImage
  …
  CC      sound/soc/codecs/pcm5102a.o
  LD      sound/soc/codecs/snd-soc-pcm5102a.o
```

4.2.3 Device Tree Modification

Pinmuxing and driver connections are made with a new device tree include file described the following sections.

4.2.3.1 Simple Audio Card

Plumbing between the PCM5102A device driver and the AM335X processor is done in the device tree with the simple-audio-card device tree bindings. For more information about available simple-audio-card device tree properties, see the following kernel documentation:

PATH	Documentation/devicetree/bindings/sound/simple-card.txt

4.2.3.2 Creating CODEC Device Tree Fragment

The following are bindings necessary for the PCM5102A dummy CODEC. Pinmux must be verified for potential overlap in an end system.

PATH	KERNEL/arch/arm/boot/dts/am335x-boneblack-pcm5102a.dtsi

```
*
 * Copyright (C) 2016 Texas Instruments Incorporated - http://www.ti.com/
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2 as
 * published by the Free Software Foundation.
```

```
 */

&am33xx_pinmux {
    mcasp1_pins: mcasp1_pins {
        pinctrl-single,pins = <
            /* sink must enable receivers */
            0x1a0 0x23    /* P9_42 mcasp1_aclkx - bit clock */
            0x1a4 0x23    /* P9_27 mcasp1_fsx - frame sync */
            0x1a8 0x23    /* P9_41 mcasp1_axr0 - i2s input */
        >;
    };
};

&mcasp1 {
    #sound-dai-cells = <0>;
    pinctrl-names = "default";
    pinctrl-0 = <&mcasp1_pins>;
    status = "okay";
    op-mode = <0>;    /* MCASP_IIS_MODE */
    tdm-slots = <2>;
    num-serializer = <4>;
    serial-dir = < /* 1 TX 2 RX 0 unused */
            2 0 0 0
        >;
    rx-num-evt = <1>;
    tx-num-evt = <1>;
};

/ {
    pcm5102a: pcm5102a {
        #sound-dai-cells = <0>;
        compatible = "ti,pcm5102a";
        status = "okay";
    };

    sound1: sound@1 {
        compatible = "simple-audio-card";
        simple-audio-card,name = "PCM5102a";
        simple-audio-card,format = "i2s";
        simple-audio-card,bitclock-master = <&sound1_master>;
        simple-audio-card,frame-master = <&sound1_master>;
        simple-audio-card,bitclock-inversion;

        simple-audio-card,cpu {
            sound-dai = <&mcasp1>;
        };

        sound1_master: simple-audio-card,codec {
            #sound-dai-cells = <0>;
            sound-dai = <&pcm5102a>;
            clocks = <&mcasp1_fck>;
            clock-names = "mclk";
        };
    };
};
```

Add the following to include statement at the bottom of the am335x-beaglebone.dts file to complete the device tree integration:

PATH	KERNEL/arch/arm/boot/dts/am335x-boneblack.dts

```
#include "am335x-boneblack-pcm5102a.dtsi"
```

4.2.3.3 Building Device Tree

Execute the following commands to build the new device tree with the PCM5102A include file:

PATH	KERNEL/

```
$ export PATH=${SDK_TOOLCHAIN_PATH}:$PATH
$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- am335x-boneblack.dtb

  DTC     arch/arm/boot/dts/am335x-boneblack.dtb
```

4.2.4 ALSA Configuration

Once the previous steps are completed and the BeagleBone is booted, ALSA needs to be configured to map the default audio playback and capture devices. Here, “hw:1,0” maps to an existing USB audio playback device connected to the BeagleBone Black. “hw:2,0” maps to the PCM5102A capture device, which is attached to the DSP-processed microphone array.

```
pcm.dmixed {
  type dmix
  ipc_key 1024
  ipc_key_add_uid 0
  ipc_perm 0666
  slave.pcm "hw:1,0"
}
pcm.dsnooped {
  type dsnoop
  ipc_key 1025
  slave.pcm "hw:2,0"
}

pcm.duplex {
  type asym
  playback.pcm "dmixed"
  capture.pcm "dsnooped"
}

pcm.!default {
  type plug
  slave.pcm "duplex"
}

ctl.!default {
  type hw
  card 1
}
```

5 Results and Testing

5.1 Verifying Linux Driver

After the system boots, check the kernel logs for information about the PCM5102:

```
/$ dmesg | grep pcm
[  29.040432] asoc-simple-card sound@1: pcm5102a-hifi <-> 4803c000.mcasp mapping ok
```

5.2 Verifying ALSA Detection

Verify ALSA as registered the PCM5102 as a capture device:

```
$ arecord -l
**** List of CAPTURE Hardware Devices ****
card 1: PCM5102a [PCM5102a], device 0: davinci-mcasp.0-pcm5102a-hifi pcm5102a-hifi-0 []
  Subdevices: 0/1
  Subdevice #0: subdevice #0
```

NOTE: By default, the Beamformer demo running on the C5517 DSP EVM always outputs audio in 32-bit format, sampled at 16 KHz.

5.3 Audio Analysis

Connect the headphone out from the USB audio card of the BeagleBone Black to the line-in on a PC. Using a program such as Audacity, the audio can be recorded, split into left and right channels, and evaluated independently.

Testing was conducted with a white noise generator positioned 0° to the LMB, with a measured output of -70dB at the LMB. A person speaking at an angle of 30° to the LMB measured an average SPL of -70dB (see Table 4 and Figure 12).

Table 4. LMB Beamformer Demo Test Conditions

Test Environment
Office room
White noise 0° at 70 SPL
Speech 30° at 70 SPL

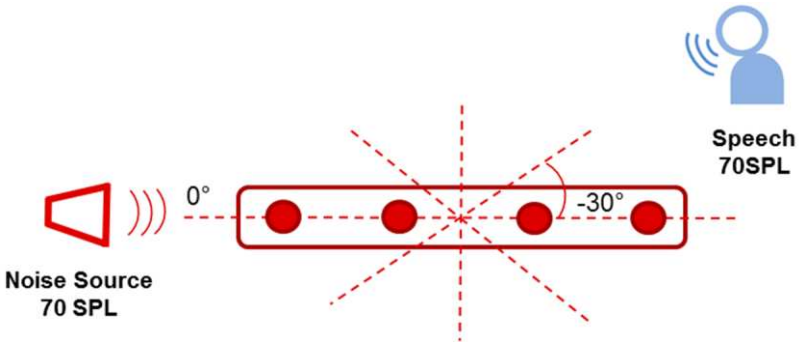

Figure 12. Test Environment Diagram

Figure 13 and Figure 14 show a spectrogram and waveform generated by Audacity, comparing DSP-processed vs untouched audio sampled from the LMB.

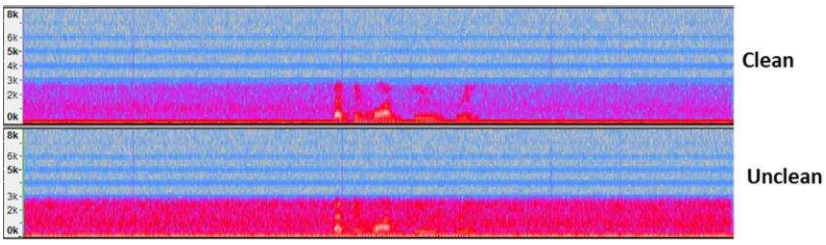

Figure 13. Spectrogram Clean vs Uncleaned Audio From LMB

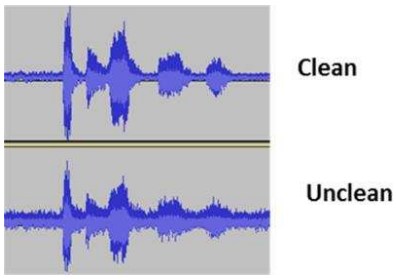

Figure 14. Waveform Clean vs Uncleaned Audio From LMB

6 Future Work

This application report describes how to integrate a microphone array pre-processed by a C5517 DSP and an AM335X applications processor running Linux. The next steps include:

- Provide hardware and software interface to configure common parameters on the DSP and microphone array board, (sampling rate, ADC gain, muting)
- Provide seamless integration with ALSA including mixer controls
- Integrate an I2S input DAC + digital audio amplifier (TAS575X) for complete TI solution from input to output.

7 References

This applications report is based in part on several TI Designs, and uses hardware that is available from TI.com.

1. *PCM1864 based Circular Microphone Board Reference Design*
2. *C5517 General Purpose EVM User Guide*, Wiki Article
3. *Audio Pre-Processing System Reference Design for Voice-Based Applications Using C5517*
4. *Audio Pre-Processing Reference Design for Voice-Based Applications Using 66AK2G02* (TIDUCR7)
5. *C55x CSL Audio Pre-Processing*, Wiki Article
6. *Linux Core Audio User's Guide*, Wiki Article
7. *Sitara SDK Linux Audio ALSA User Space*, Wiki Article
8. *Free Electrons, ASoC: Supporting Audio on an Embedded Board*, Alexandre Belloni, PDF training document
9. *Linux Kernel Users Guide*, Wiki Article

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