Design Guide: TIDM-FILTERING-SIGNALPROCESSING

Filtering and Signal Processing Reference Design Using MSP430 FRAM Microcontroller

Description

This reference design showcases the performance of the Low-Energy Accelerator (LEA) on MSP430™ FRAM microcontroller (MCUs) in performing advanced filtering and signal processing while maintaining ultra-low power on a 16-bit MCU. The LEA provides a boost of 13.8x over a traditional C implementation for a 256-point complex FFT. LEA also provides real-time FIR filtering performance at a high audio sampling rate of 20 kHz.

Design Resources

TIDM-FILTERING—SIGNALPROCESSING	Design Folder
MSP430FR5994	Product Folder
MSP-EXP430FR5994	Tools Folder
BOOSTXL-AUDIO	Tools Folder
430BOOST-SHARP96	Tools Folder
BOOSTXL-SHARP128	Tools Folder

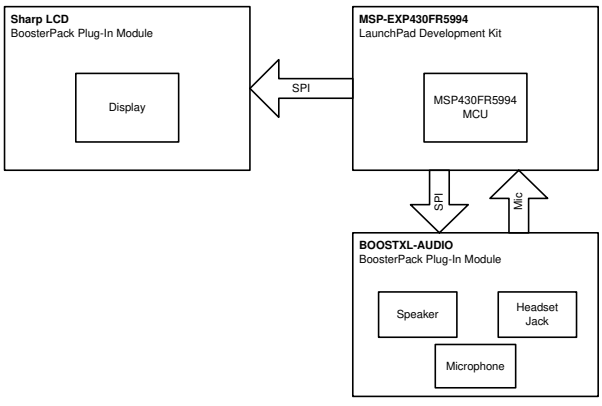
Search Our E2E™ support forums

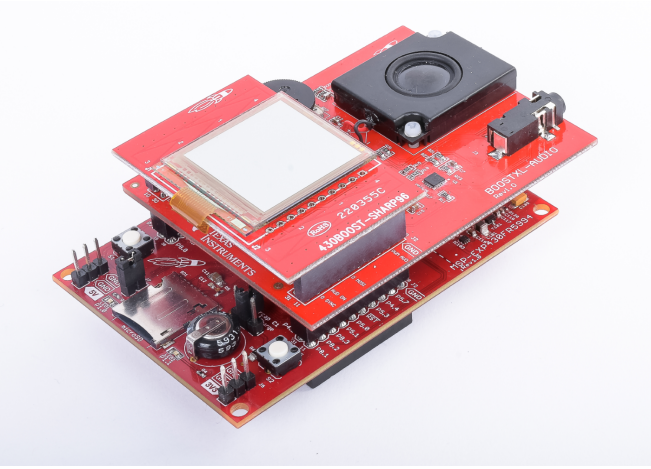
Features

- Real-time FFT comparing non-accelerated to accelerated performance
- Real-time FIR filtering with several pre-set filter coefficients

Applications

- Flow metering
- Industrial sensing
- Portable health monitoring





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

1 Design Overview

This reference design showcases the performance of the Low Energy Accelerator (LEA) on MSP FRAM Microcontroller (MCUs) in performing advanced filtering and signal processing while maintaining ultra-low-power on a 16-bit microcontroller (MCU). The LEA module provides a boost of 13.8x over a traditional C-based 16-bit MCU implementation for a 256-point complex FFT. The LEA module also provides the ability to perform real-time FIR filtering performance at a high audio sampling rates of up to 20 kHz.

2 System Description

The Fast Fourier Transform (FFT) algorithm converts a time-domain signal to frequency-domain and vice-versa where FFT is manually used in many applications in signal processing. Additionally, the Finite Impulse Response (FIR) filter is a filter that uses an impulse response of a finite duration that is mainly used in applications for digital filtering. One of the examples in this reference design showcases the clock cycles required to execute a real-time 256-point FFT using LEA when compared to the CPU. By toggling button S1, the application then showcases the capability of performing real-time 120-tap FIR filter with various filter presets by actively recording, filtering, and playback. FFT and FIR are used in applications that require advanced signal processing in the frequency domain. This design uses the MSP DSP Library which leverages LEA if available.

To get started with this reference design, the user is required to use the MSP-EXP430FR5994 LaunchPad™, BOOSTXL-AUDIO BoosterPack™ and either the 430BOOST-SHARP96 BoosterPack or the BOOSTXL-SHARP128 BoosterPack. See Section 5.2 on getting started with the Audio BoosterPack in conjunction with the Sharp® LCD Boosterpack.

2.1 MSP430FR5994 MCU

The MSP430FR5994 MCU device is a 16-bit ultra-low-power MCU with 256 KB of FRAM, 8-KB of shared SRAM, a breadth of analog peripherals, and the LEA vector math accelerator. This reference design requires the use of LEA. To find out more on MSP430FR5994, see the product page.

2.2 LEA

LEA is a vector math engine that performs signal processing, matrix multiplications, and many more operations. The LEA is a low-power co-processor that is capable of performing operations without any CPU interventions and triggers an interrupt when the operation is completed. The accelerator operates based on the commands that are provided during configuration in which the commands are pointers to memory input or output buffers and the type of operation. The MSP430FR5994 MCU has a total of 8-KB of SRAM, where 4-KB are shared with LEA for all data input, output, and parameters.

LEA operations may be accessed using the intuitive and easy-to-use MSP DSP Library. Before using the MSP DSP Library APIs, the user would first need to specify the input and output memory locations by allocating an array in which the location needs to reside within the shared 4-kB LEA SRAM. For example, to perform a 256-point complex FFT with LEA, the data input array consists of 256-word real and 256-word complex values that total to 512-words (1024-bytes).

A more detailed explanation on the usage of the DSP Library APIs is available through the DSP Library API Programmers Guide.

2.3 FRAM

Ferroelectric Random Access Memory (FRAM) is a non-volatile memory that may be used for application, constants, variables, stack, and heap. FRAM allows the user to scale their application size as required based on its needs. The user may also place variables in FRAM as needed by using the compiler directive *persistent*, in which case these variables are sustained through a reset and remain uninitialized by the compiler at startup. To learn more about the compiler directive*persistent*, see *MSP430™ FRAM Technology – How To and Best Practices* (SLAA628). FRAM is ideal for data logging as it may accommodate a very high number of write and ease cycles of up to 10 ^ 15. In this reference design, the DAC buffers are allocated within the FRAM area, while the ADC buffers are allocated within the SRAM memory.

FRAM is also used to store the state of the application, as shown in this reference design http://www.ti.com/tool/TIDM-FRAM-CTPL. Upon a state change from a button press, the application state is automatically stored in FRAM. When the application powers-up from a cold boot or resets, the application starts up from exactly where it left off.

2.4 12-bit ADC

The MSP430FR5994 microcontroller has an on-chip 12-bit SAR ADC that is used to continuously sample the microphone output at either 8-kHz or 20-kHz, depending on the application. The microphone has a front-end amplifier that may be powered through a GPIO from the LaunchPad (P4.1). The output of the amplified microphone is connected to ADC Channel 15 (P3.3). To ensure a deterministic sampling rate, the ADC is automatically triggered by Timer_A0. Once the ADC sample is completed, the data is then automatically moved using DMA Channel 0. The ADC may be configured to output as unsigned binary or 2's complement format.

2.5 14-bit DAC (DAC8311)

To output high-quality audio, an external 14-bit DAC (TI DAC8311) is used to generate the audio output. The output is then fed through an audio amplifier before driving the onboard speaker or headset. All of this playback hardware is managed by the Audio BoosterPack. The DAC interface is a 3-pin SPI (SCLK, MOSI, and SYNC) in which the SCLK is driven at 4 Mbps. In this reference design, the DAC is only used for the FIR filter operation where the post-filtered ADC input is outputted to the DAC. While the DAC is used, the LCD display remains static. See note below:

NOTE: The SPI interface is shared between the Sharp Graphics LCD BoosterPack and the DAC8311 device. Only one device may be driven at once by using the appropriate Chip Select (LCD) or SYNC (DAC) pin. Also, the clock polarity of the LCD and DAC are different, thus, the application must re-configure the SPI clock polarity before writing to either interface.

3 Block Diagram

Figure 1 shows the block diagram theory.

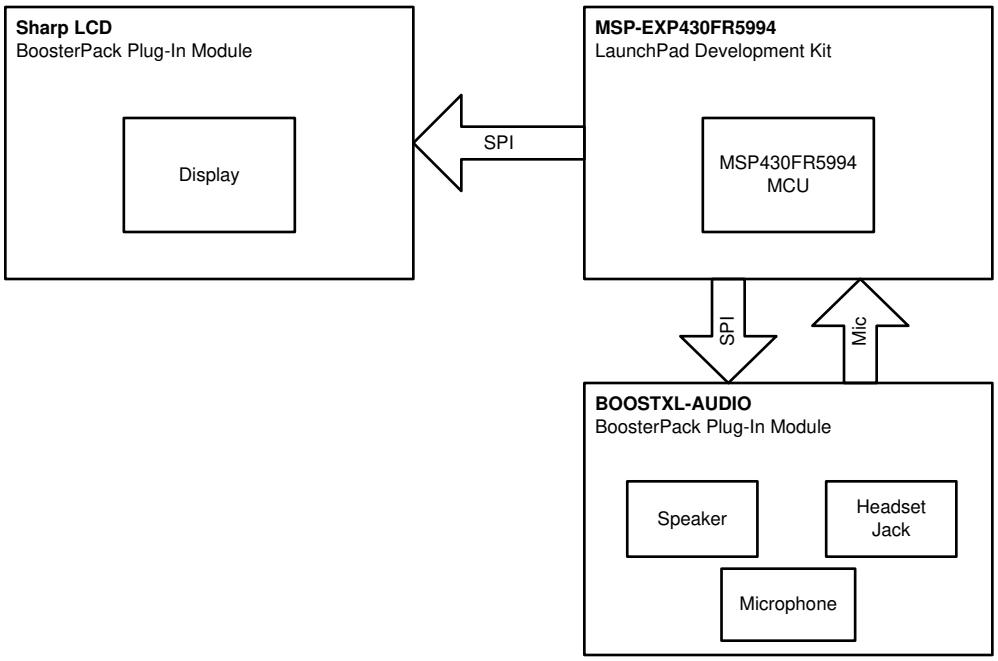


Figure 1. Block Diagram Theory

4 System Design Theory

This design uses the LEA to perform real-time FFT or FIR filtering with audio sampled using the onboard 12-bit ADC from the Audio BoosterPack's microphone. The Audio BoosterPack is also capable of automatically detecting a headset that is plugged-in and switches from the onboard speaker to the headset. If the headset has a built-in microphone, typically a 4-pin 3.5-mm jack, the Audio BoosterPack automatically switches and uses the headset microphone.

To ensure a real-time performance with no sample loss during processing, this application implements dual-buffered memory to store the ADC samples. When a buffer a full, the application starts processing the buffer while the application continues to sample the ADC in the background into the secondary buffer. The ADC sample size varies depending if it is performing an FFT or FIR. For example, the FFT requires 256 ADC samples before calling the FFT operation with a sample rate of 8 kHz. The FIR is designed to have 240 ADC samples at a 20-kHz sample rate before the FIR filter is called.

As mentioned in Section 3, the application states are stored in FRAM as a persistent variable, meaning that the application starts exactly where it last left off. To switch between FFT with LEA, FFT without LEA, and FIR Filtering, toggle the S1 button on the LaunchPad. The S2 button is only used when the application is in FIR Filtering mode. Figure 2 shows the sequence on how to interact with the application. On default load, the application starts with 256-point FFT with LEA.

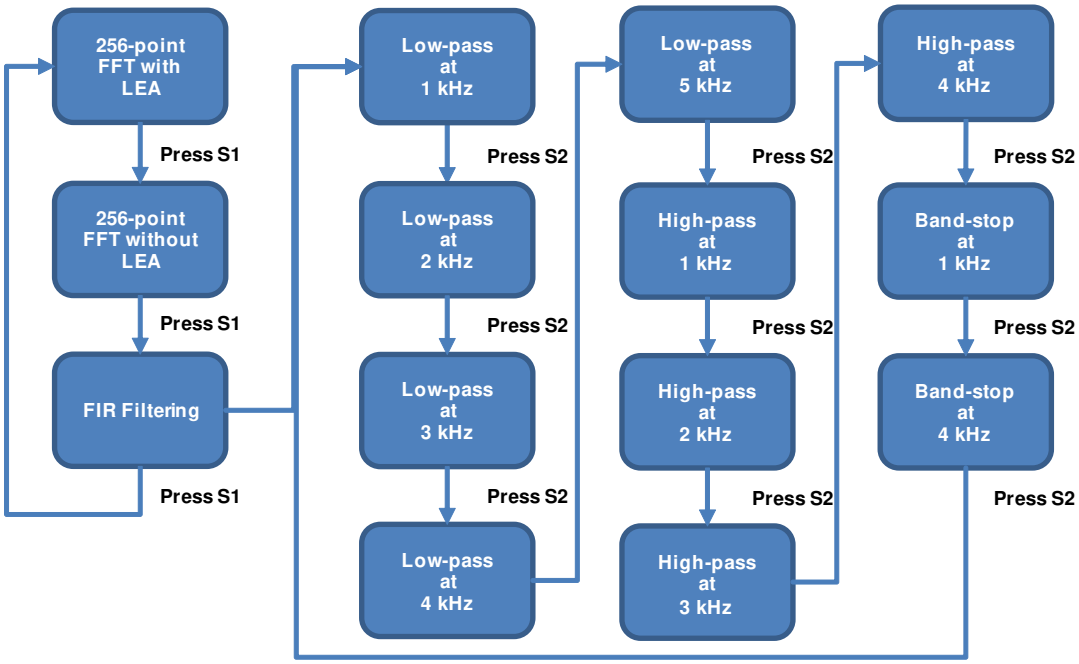


Figure 2. Application Flowchart

4.1 Real-Time 256-Point Complex FFT

The low-energy accelerator on the MSP430FR5994 MCU is capable of supporting up to a 512-point complex FFTs or 1024-point real FFTs. In this design, the performance of a 256-point complex FFT using LEA by displaying the number of clock cycles used to perform the FFT is demonstrated. This design collects from the onboard microphone and samples up to 256 ADC samples before starting the FFT routine. To ensure real-time FFT performance, this application implements a dual-buffered array where one buffer is always being filled, while the other is processed. The audio is sampled at a rate of 8-kHz using onboard intelligent peripherals such as the timer that automatically triggers the ADC to sample, and the Direct Memory Access (DMA) moves the data to memory.

Because this is a complex FFT implementation, the input audio samples must be interleaved between real and imaginary values where the imaginary values are forced set to zero (0). This step is completed post-sampling the ADC. The MSP DSP Library API is then used to perform the complex FFT where the FFT output also contains real and imaginary values. The magnitude is then calculated, and contains 128-point positive and negative frequencies. Because the 430BOOST-SHARP96 display is only 96 pixels wide, only the positive frequency portion of the FFT is displayed and only up to 96 pixels, the other 32 pixels (3-kHz to 4-kHz FFT magnitude) is not displayed. The BOOSTXL-SHARP128 LCD can display all 128 points. See Figure 3 on the timing diagram.

This design showcases and compares the FFT performance using LEA vs non-LEA by displaying the number of CPU cycles required to execute the FFT routine. This application demonstrates that the MSP DSP Library complex FFT routine with LEA requires 5,578 clock cycles (IAR). This clock cycle count is higher than running the MSP DSP Library FFT routine stand-alone because the CPU is being interleaved with the DMA operation to move data from ADC memory to SRAM for the dual-buffered operation. On the non-LEA FFT implementation in a custom assembly code is optimized to perform FFT takes approximately 77,000 cycles.

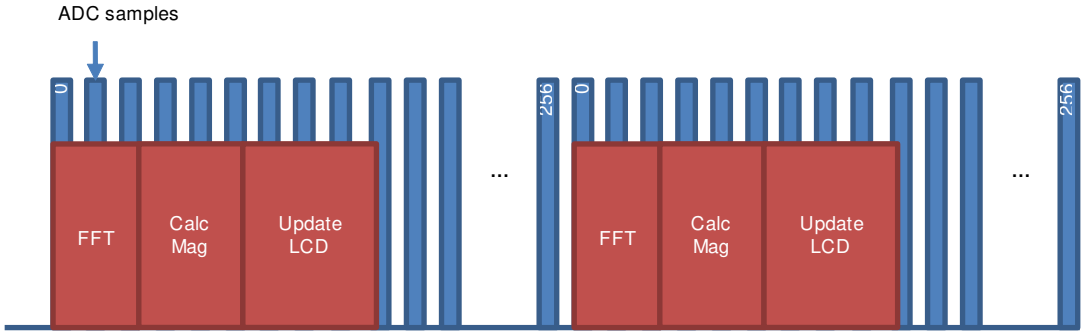


Figure 3. Generic Timing Diagram for FFT Operation

4.2 Real-Time FIR Filtering

One of the LEA's signal processing capabilities is FIR filtering. This reference design showcases the capability of performing a real-time FIR filtering which includes recording, filtering the audio with LEA, and playback. The application uses the MSP DSP Library FIR function that is flexible in allowing the user to configure various data input sizes and the FIR coefficient size.

To simplify the design, the FIR input audio data and coefficient size in this design is fixed. The audio input and output size is currently fixed to 240 samples with a 120 FIR coefficient size. There is also approximately 12-ms of delay with a 20-kHz sample rate between recording, filtering, and playback. The FIR coefficients are generated using the MATLAB or Python scripts included in the */src/scripts* folder of the software zip package. This design demonstrates the ability to perform low-pass (filter high-frequencies), high-pas (filter low-frequencies), and band-stop (filter a specific frequency) filtering. In some cases, the FIR coefficients have to be odd sized of 119, where the last byte is padded with 0.

To ensure continuous FIR filtering, FIR requires past history audio data. The past history data must be equal to the size of the FIR coefficients. Before performing the FIR function, the last 120 ADC input samples are copied to the beginning of the ADC input. Then, 240 new ADC samples are copied in front of the past history ADC input samples. Therefore, the total input buffer size that is inserted into the FIR is 360 samples. The output of the FIR data must be equal to the input audio size. Figure 4 shows the FIR design implementation.

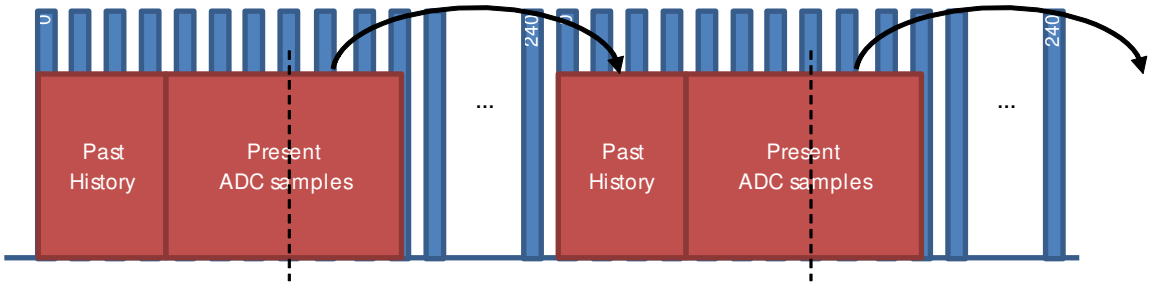


Figure 4. FIR Design Implementation

NOTE: TI recommends to use a headphone for this example as the onboard loudspeaker and microphone may create audio feedback, see Figure 5 .

Figure 5 shows the stacked LaunchPad and BoosterPack with headphones reference design.

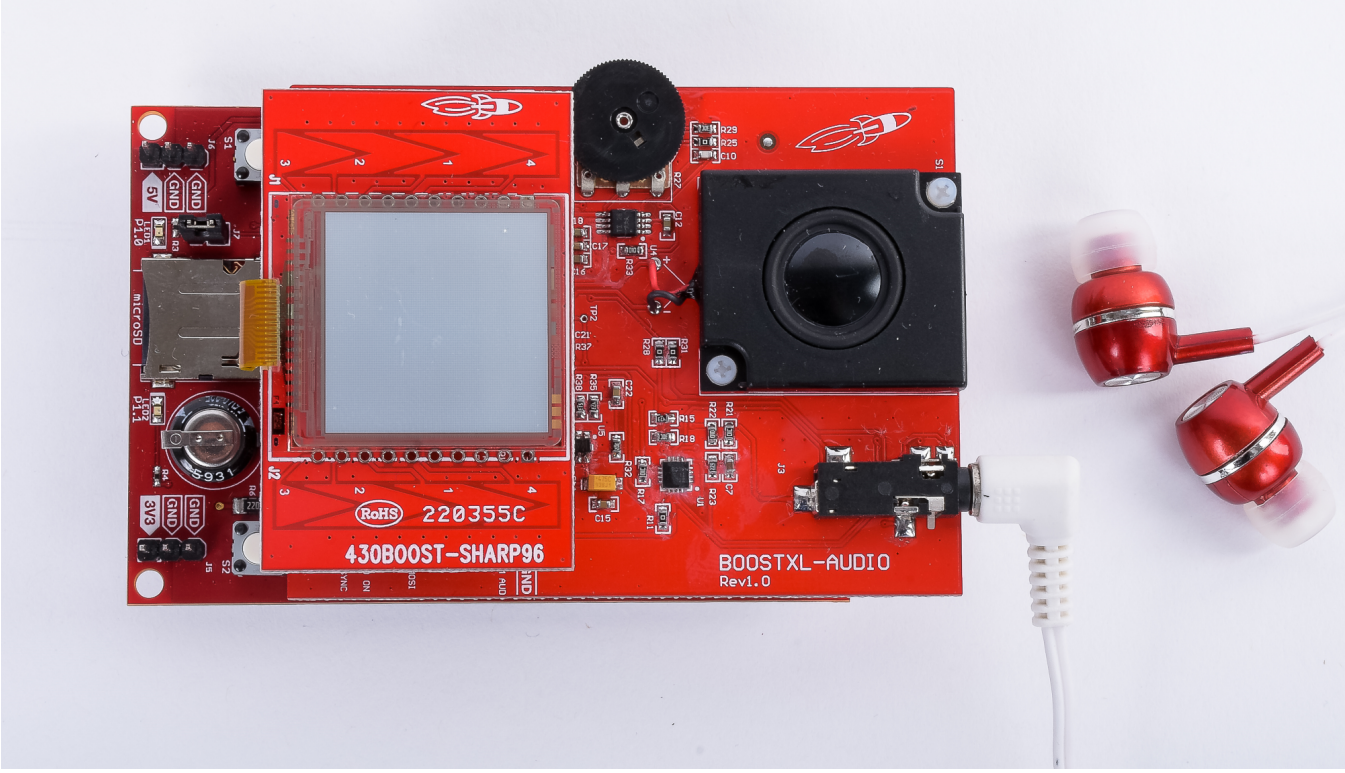


Figure 5. Stacked LaunchPad and BoosterPack With Headphones Reference Design

Because the graphics LCD display shares the same SPI pins with the DAC for audio playback, the graphics LCD display remains static throughout the FIR mode. The display only refreshes when the user changes to a different FIR filter. For example, one changes from low-pass 5-kHz cut-off to a high-pass 1-kHz cut-off when the S2 button is pressed.

5 Getting Started Hardware

To power the LaunchPad, the user may use either the Micro-USB or use an external power source like this Li-ion battery BoosterPack.

Figure 6 shows the stacked LaunchPad and BoosterPacks reference design.

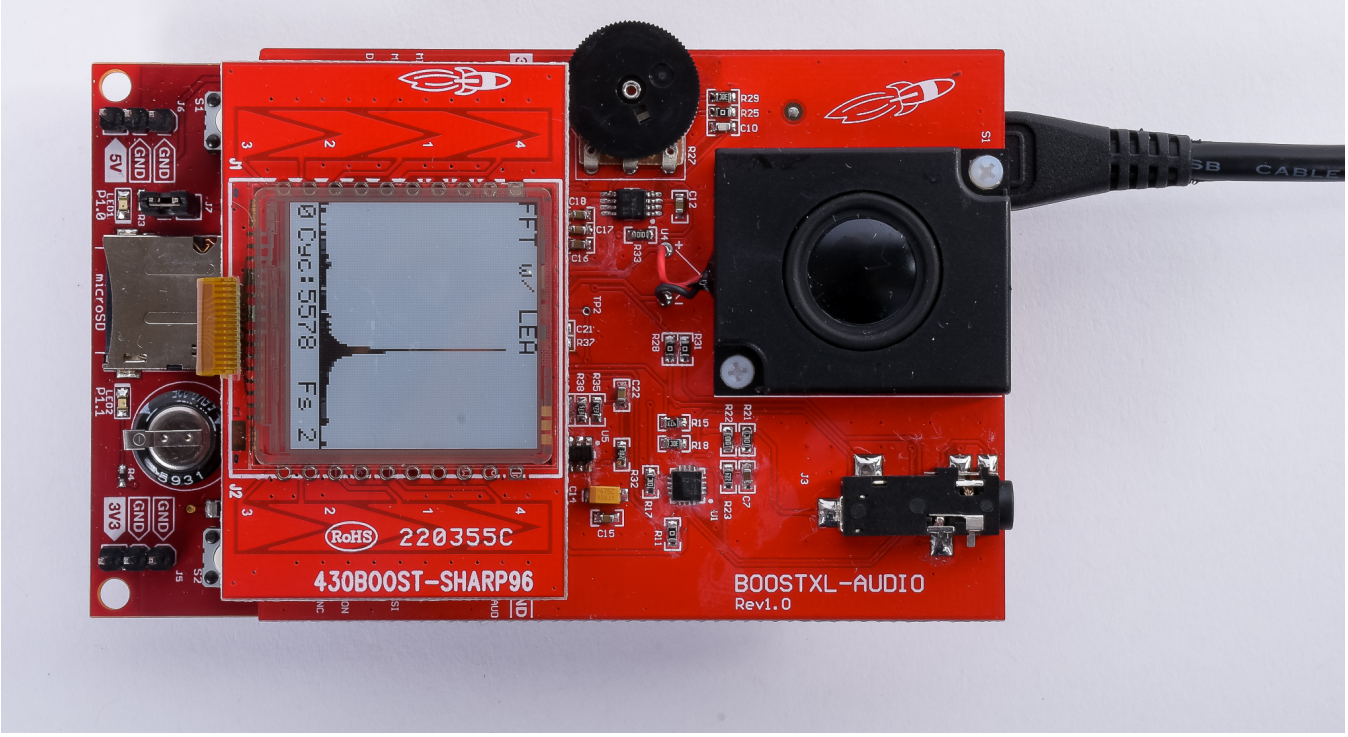


Figure 6. Stacked LaunchPad and BoosterPacks Reference Design

5.1 MSP-EXP430FR5994 LaunchPad

The MSP-EXP430FR5994 LaunchPad is based on MSP430FR5994 MCU device which uses a 40-pin BoosterPack standard with an onboard SD card slot.

To learn more about this LaunchPad, see the product page.

5.2 BOOSTXL-AUDIO BoosterPack

To use the graphics LCD BoosterPack in conjunction with the Audio BoosterPack, the microphone power and output on the Audio BoosterPack must be modified to use the alternate pins. The reason for this is because these two-pins conflict with the graphics LCD BoosterPack. Using a soldering iron, remove the 0-Ω resistors from R1 and R4 and replacing the 0-Ω resistor to R3 and R5, as shown in Figure 7.

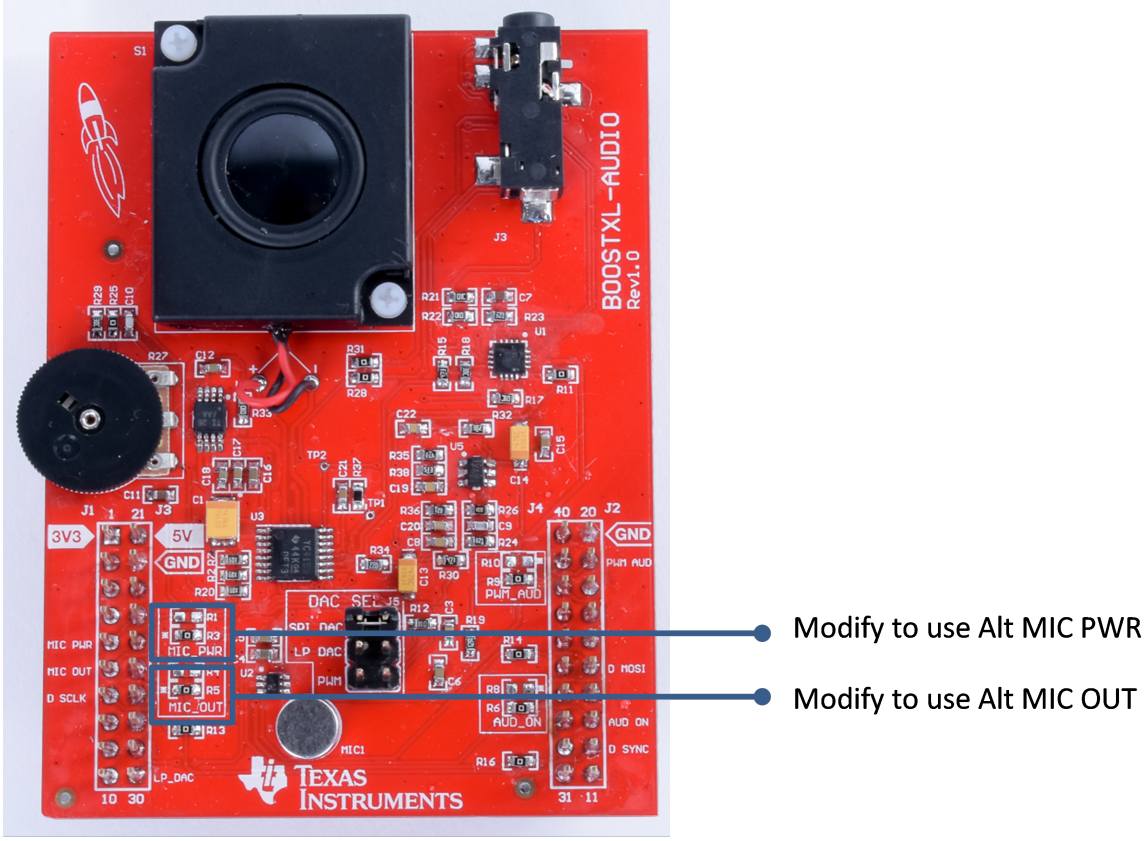


Figure 7. BOOSTXL-AUDIO BoosterPack Required Modification

5.3 Sharp Memory LCD BoosterPack

Figure 8 shows the 430BOOST-SHARP96 Graphics LCD BoosterPack. A graphics display that the 430BOOST-SHARP96 has is used to display the FFT output and also indicate the FIR current mode.

Figure 8. 430BOOST-SHARP96 Graphics LCD BoosterPack

5.4 Sharp 128x128 Memory LCD and microSD card TI BoosterPack

Figure 8 shows the BOOSTXL-SHARP128 Graphics LCD BoosterPack. The display in this BoosterPack is used to display the FFT output and also indicate the FIR current mode.

Figure 9. BOOSXL-SHARP128 Graphics LCD BoosterPack

6 Getting Started Firmware

To get started, download the ZIP package from the reference design. This design supports both the Code Composer Studio™ v9.01 or Embedded Workbench IAR v7.12.x development environments and newer.

6.1 Code Composer Studio™ (CCS)

To get started with CCS:

1. Launch Code Composer Studio.
2. Navigate to **Project > Import CCS Projects**.
3. Click **Browse**.
4. Navigate to the **/src/CCS** folder.
5. Click **Finish**.
6. Click **Project > Build Project**.
7. Click the **debug** button to download and debug the project.

NOTE: Check the revision of the MSP430FR5994 on the MSP-EXP430FR5994 Launchpad. Then, in the code, go to */src/DSPLib/include/DSPLib_lea.h* and check the line "#define MSP_LEA_REVISION" and ensure that the specified revision "MSP_LEA_REVISION_A" is used for hardware revision A, or "MSP_LEA_REVISION_B" for any later hardware revision.

6.2 Embedded Workbench IAR

To get started with CCS:

1. Navigate to the **/src/IAR** folder.
2. Double-click **tidm-filtering-signalprocessing-lea.eww** to open the IAR workspace project.
3. Click **Project > Make** to build the project.
4. Make sure that the Micro-USB is plugged in first to download and debug the application.
5. Click **Project > Download and Debug**.

NOTE: Check the revision of the MSP430FR5994 on the MSP-EXP430FR5994 Launchpad. Then, in the code, go to */src/DSPLib/include/DSPLib_lea.h* and check the line "#define MSP_LEA_REVISION" and ensure that the specified revision "MSP_LEA_REVISION_A" is used for hardware revision A, or "MSP_LEA_REVISION_B" for any later hardware revision.

7 Application Execution Performance

The signal processing performance includes the clock cycles required to execute each FFT routine, and the average energy consumed by the application.

7.1 FFT

In the FFT example using LEA, the number of cycles required to perform a 256-point complex FFT is approximately 5,578 cycles, and takes approximately 700 µs at 8 MHz (IAR). This number of clock cycles is slightly higher than the actual number of cycles required performing the FFT, at 4,730 cycles. To explain the difference, the background DMA operation moves the ADC data to the dual-buffer design. The DMA is currently configured to block the clock cycles when it moves the data thus increasing the number of cycles for the FFT.

For the entire application where it performs the FFT followed by updating the internal display buffers, calculating the FFT magnitude, and refreshing the graphics LCD display, it takes approximately 15.1 ms, as shown in Figure 10.

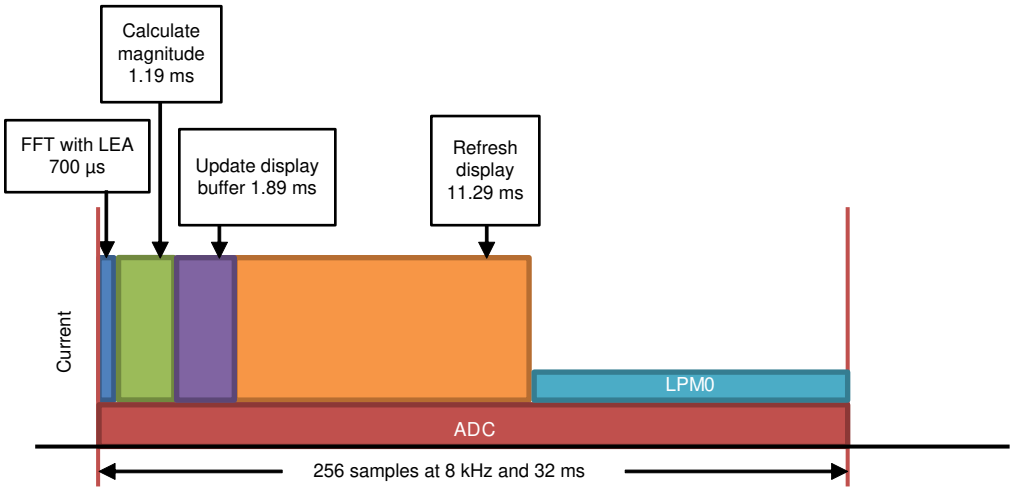


Figure 10. FFT With LEA Per Update Execution Timeline

This reference design also showcases FFT without using LEA where optimized assembly code was written to perform the FFT. In this case, the FFT itself took approximately 77,098 cycles that takes 9.64 ms at 8 MHz as shown in Figure 11. By using LEA to perform the FFT, the user observes an improved performance that is about 13.8 times faster. The total application duty cycle time is approximately 24.22 ms. Given the time taken to process the FFT alone is 13.8 times higher when LEA is not used. Observe that the remaining overhead time to perform any application-level tasks is significantly reduced.

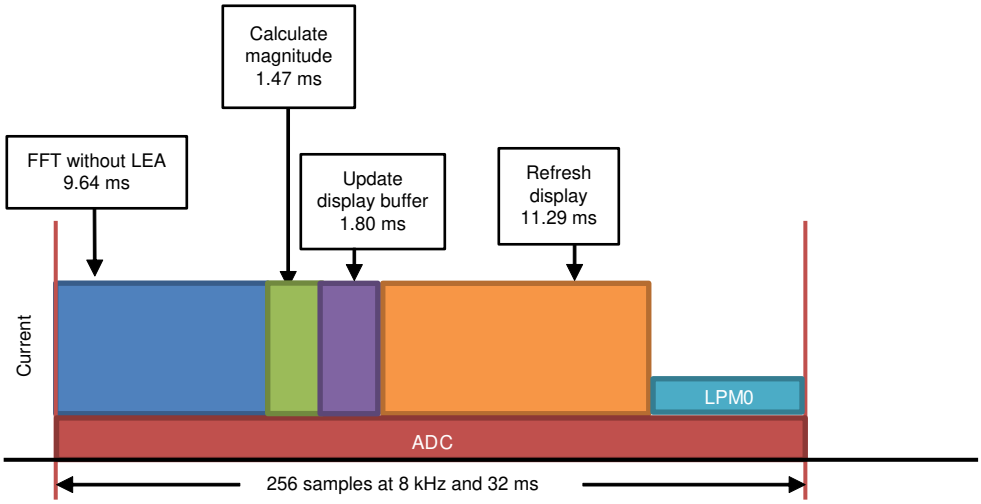


Figure 11. FFT Without LEA Per Update Execution Timeline

7.2 FIR

To ensure continuous real-time FIR performance, DMA is only used to move ADC data to memory which triggers every 50 us (20 kHz), and another timer is used for audio playback at 20 kHz. It is critical that both ADC and audio playback are at the same sample rate. To ensure critical timing, no other DMA channels may be used, thus, most of the data copy is completed using the CPU active time. Total time is 6.685 ms and is well within the sample window range of 12 ms. Figure 12 shows the FIR per update execution timeline.

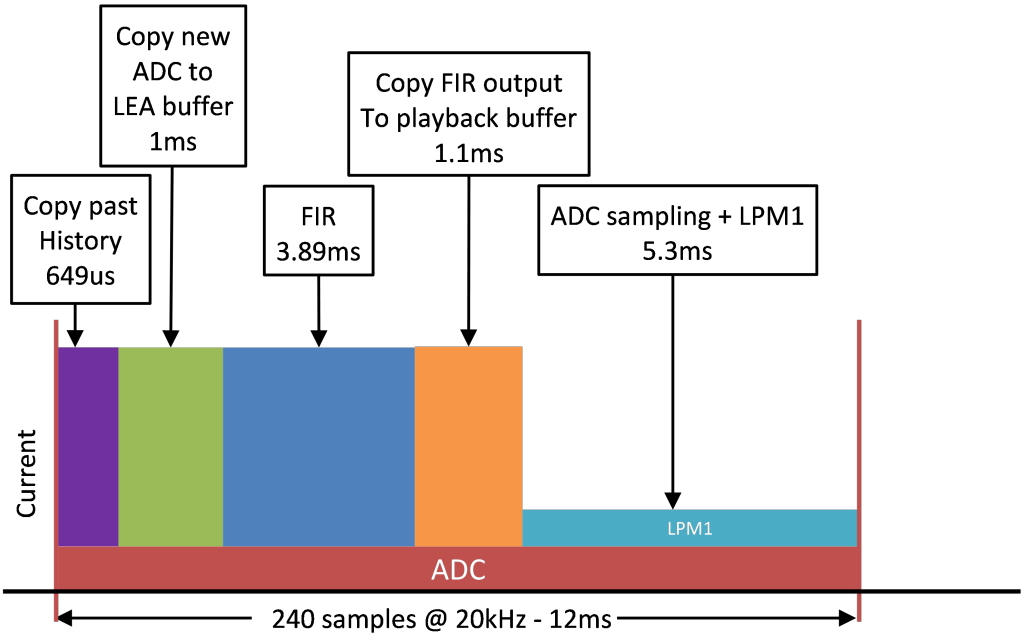


Figure 12. FIR Per Update Execution Timeline

8 Power Measurements

To quantify the energy for the application, the following FFT and FIR showcases the current consumption with the Audio BoosterPack and without the Audio BoosterPack attachment (no LCD update code either). The analysis to do without the Audio BoosterPack is to showcase the power consumption of the MSP MCU along with the CPU, LEA, and ADC running.

8.1 FFT

Figure 13 shows the FFT with LEA, Audio BoosterPack, and graphics LCD average current consumption.

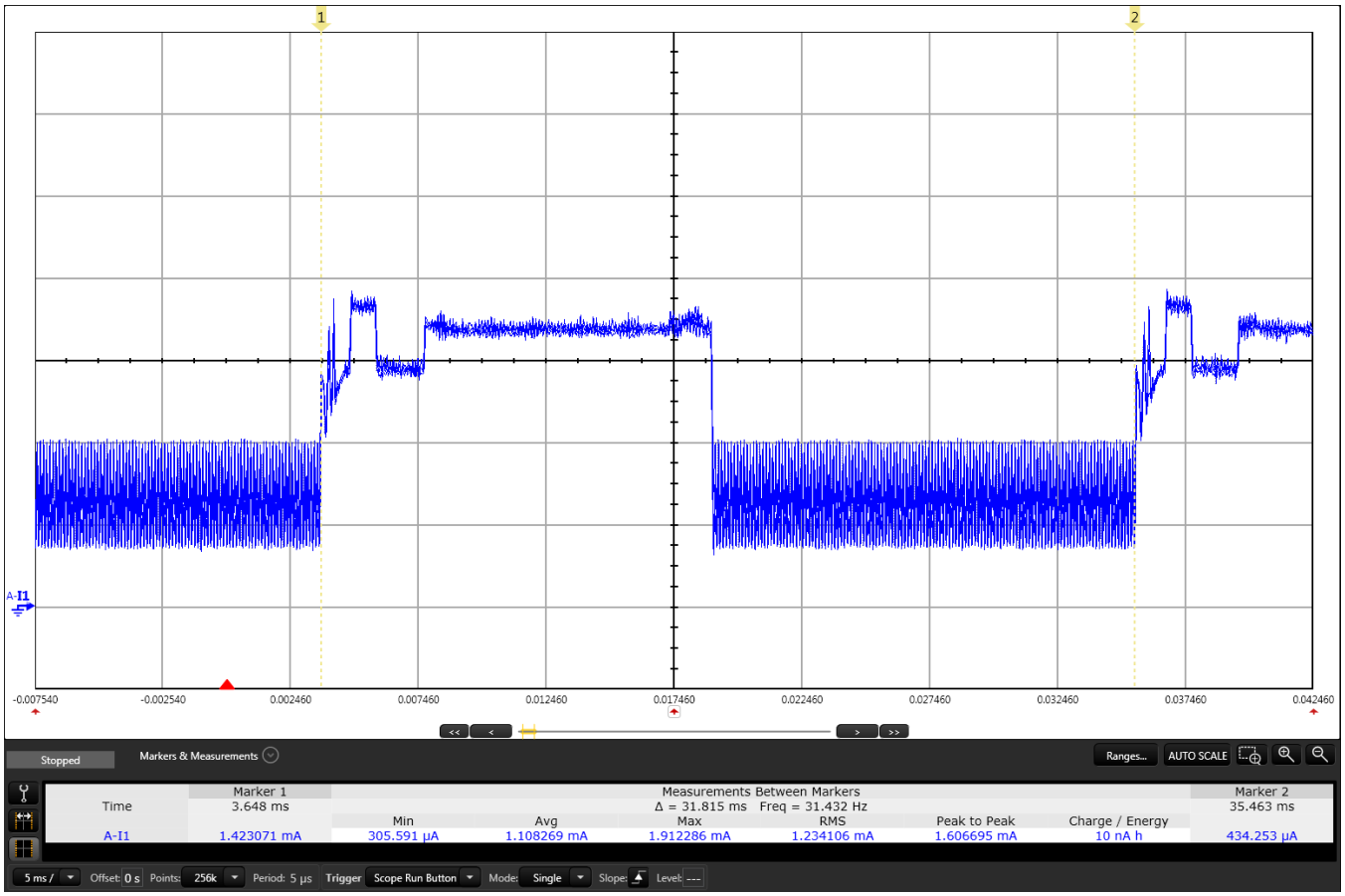


Figure 13. FFT With LEA, Audio BoosterPack and Graphics LCD Average Current Consumption

To analyze the power performance of MSP430FR5994 MCU using LEA, the Audio BoosterPack and graphics LCD is then removed. In addition, the code that refreshes the LCD screen is then **removed** by uncommenting "#define PWR_BENCHMARK" in the file *application.h* and typically takes 12.77 ms to refresh the display. The motivation to remove the LCD refresh is to compare the performance of the CPU and the analog peripheral running at the same time, which represents a typical application. See Figure 14 for the current profile.

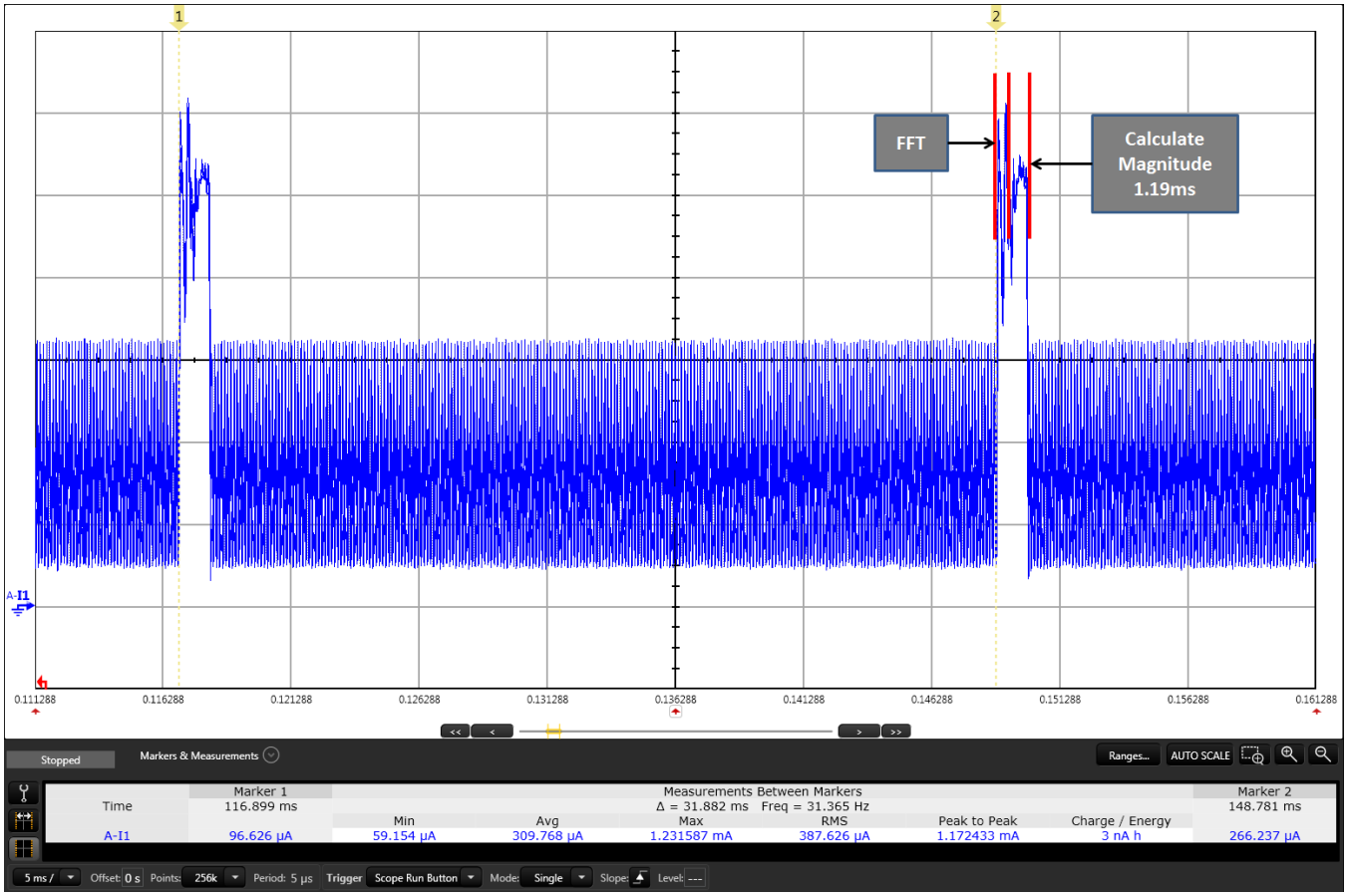


Figure 14. FFT With LEA Without BoosterPacks Average Current Consumption

As observed in Figure 14, the average current to perform an FFT with LEA and calculating the magnitude while the ADC is continuously sampling is approximately 309 µA. Observe in Figure 15 that the average current to perform an FFT without LEA is approximately 669 µA.

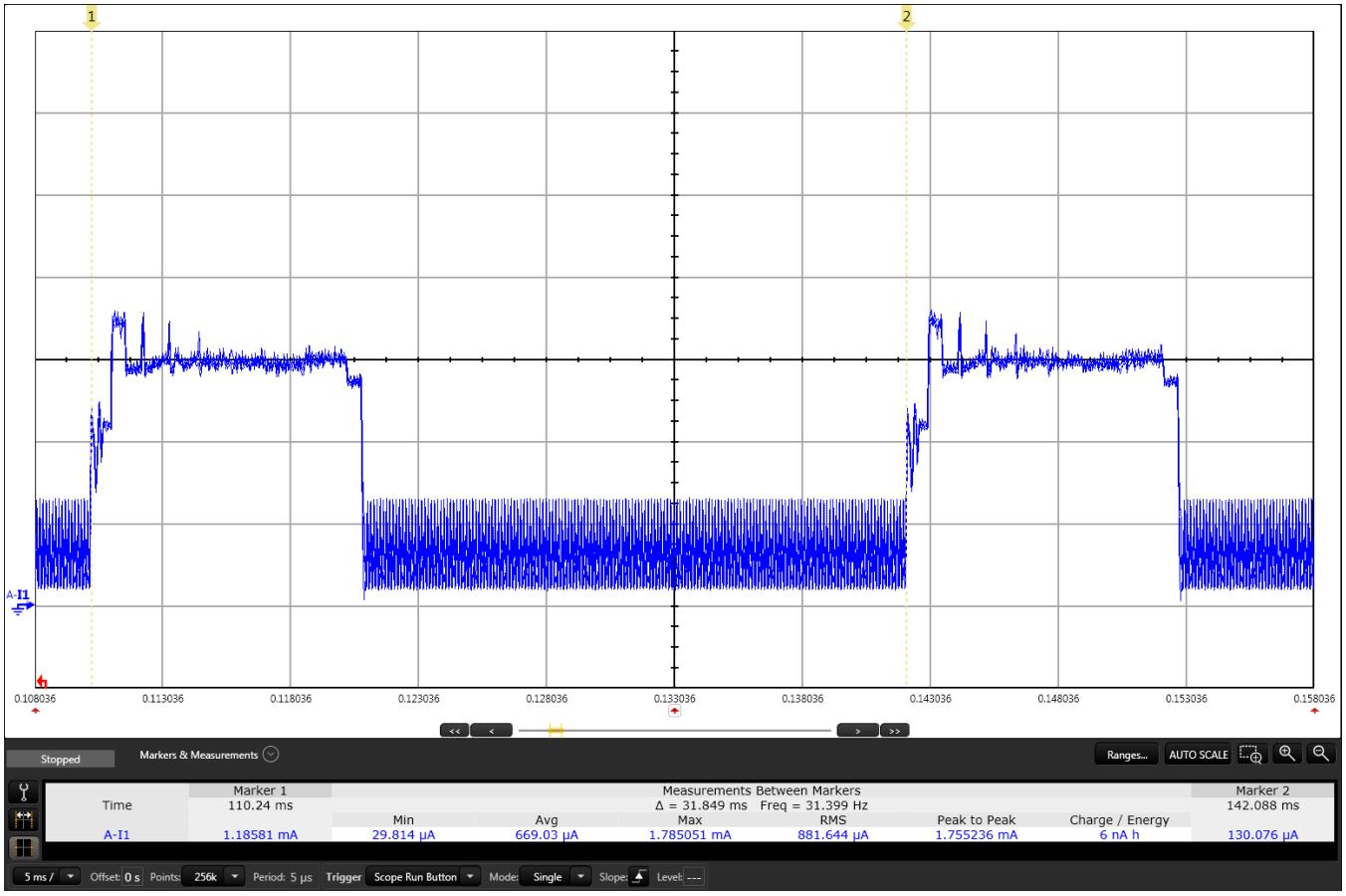


Figure 15. FFT Without LEA Without BoosterPacks Current Consumption

In summary, the power savings by using LEA to perform FFT vs assembly code is approximately **53.8%** and still leaves processing power available for other tasks.

8.2 FIR

Figure 16 shows the real-time FIR average current consumption where the ADC is actively sampling, DAC is actively playing audio while performing the FIR filtering. This figure shows the average current with the Audio BoosterPack + graphics LCD BoosterPack attached.

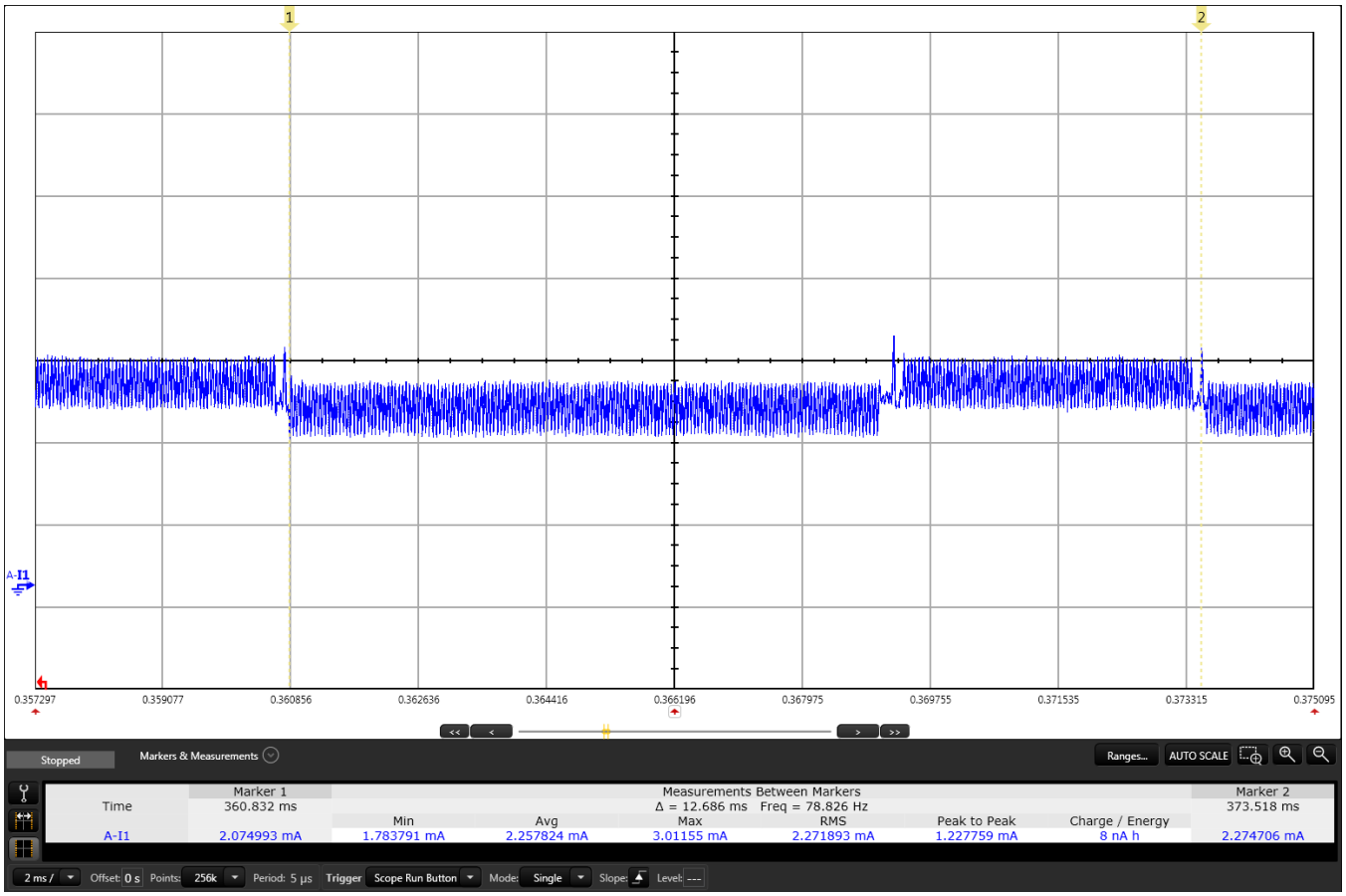


Figure 16. FIR Average Current With Audio BoosterPack and Graphics LCD

Figure 17 shows the current consumption without the BoosterPacks attached, and also shows the breakdown of each timing section that correlates with Figure 12. The average current to perform FIR including ADC and SPI actively writing to the DAC is approximately 1.15 mA.

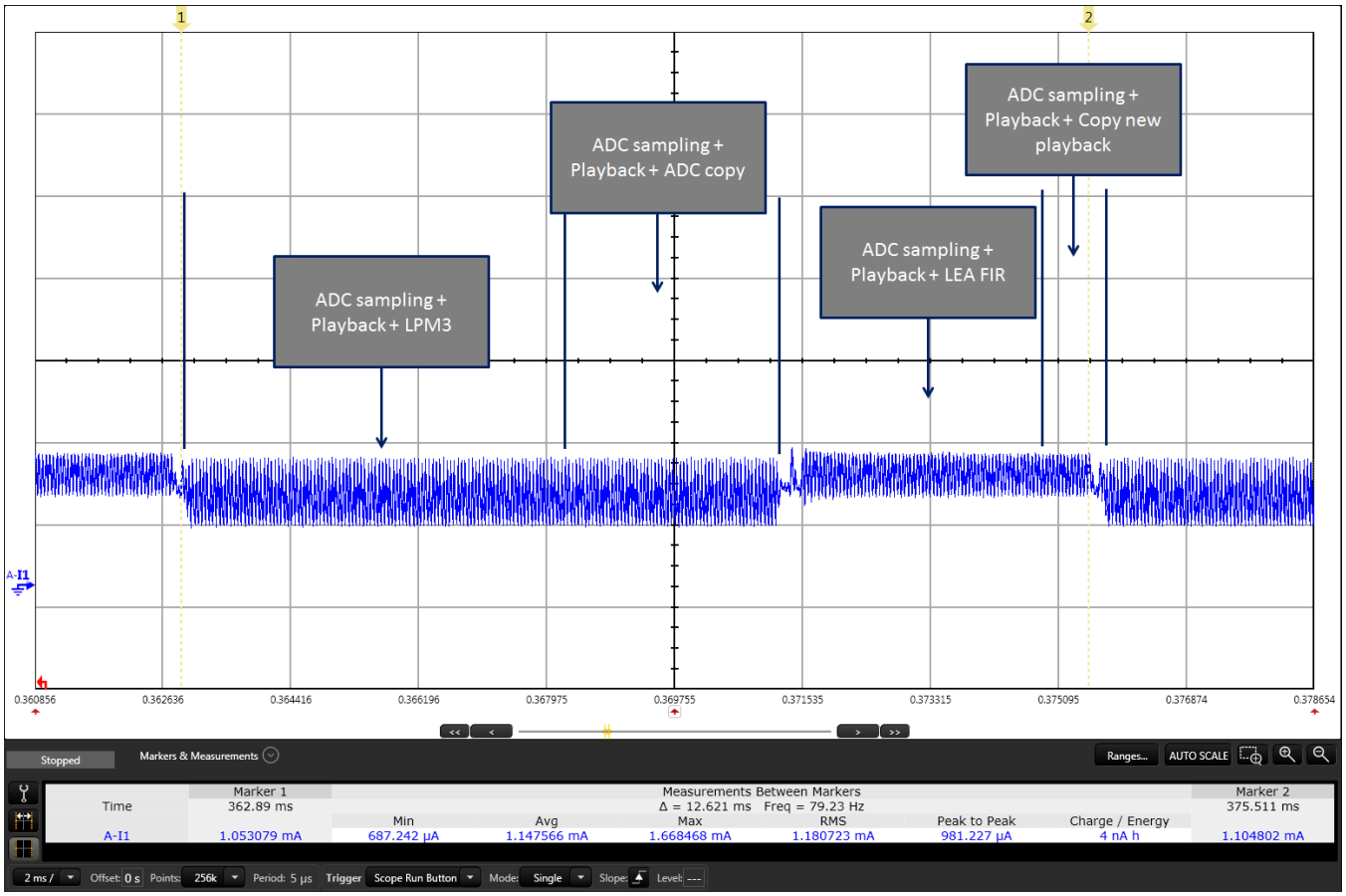


Figure 17. FIR Average Current Without Audio BoosterPack and Graphics LCD

9 EnergyTrace™++ Technology

EnergyTrace++ Technology allows the user to quickly and easily analyze the application state for debugging their application. See the EnergyTrace product page for more information.

In this reference design, Figure 18 shows the profile for FFT with LEA where the application remains in LPM1 during sleep mode, as the SMCLK must be active to actively wake the ADC to sample. This trace may not provide a full picture of the state as EnergyTrace++ technology is only capable of reading data at max 4-kHz. Thus, for fast transitions such as the ADC sampling at every 8-kHz, may not be properly captured.

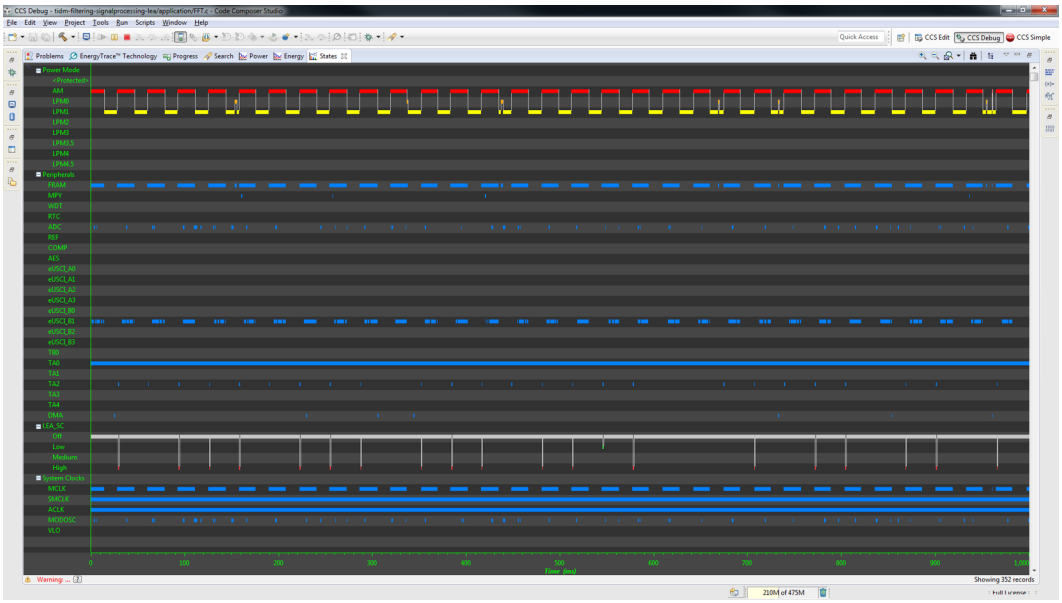


Figure 18. EnergyTrace++ Profile for FFT With LEA

Figure 19 shows the true energy and current captured over 5 seconds using EnergyTrace mode. This capture includes the Audio BoosterPack and graphics LCD being refreshed. The average current consumption here is close to the average current that was observed using the power profile meter, as seen in Figure 13.

EnergyTrace™ Profile

Name	Live
◢ System	
Time	5 sec
Energy	19.372 mJ
◢ Power	
Mean	4.4659 mW
Min	0.0000 mW
Max	5.7433 mW
◢ Voltage	
Mean	3.2889 V
◢ Current	
Mean	1.3579 mA
Min	0.0000 mA
Max	1.7462 mA
Battery Life	2xAA: 72.6 day (est.)

Figure 19. EnergyTrace Energy and Current Over 5 Seconds

10 Software Files

To download the software files for this reference design, see the link at Filtering and Signal Processing Reference Design using MSP430 FRAM Microcontroller.

11 Related Documentation

1. *MSP430FR599x, MSP430FR596 Mixed-Signal Microcontroller* User's Guide
2. *MSP430FR58xx, MSP430FR59xx, MSP430FR68xx, and MSP430FR69xx Family User's Guide* data sheet
3. MSP-EXP430FR5994 tool page

11.1 Trademarks

12 About the Authors

William Goh is a product systems engineer for the SimpleLink devices. Prior to that, he was an MSP430 applications engineer for 12 years.

Luis Reynoso is the manager of the MSP430 Industrial Sensing Applications team at Texas Instruments. He has taken multiple customer-facing, and systems-applications roles in the embedded industry, and during this time he has published several application notes and papers for microcontrollers. He joined the MSP430 Applications team in 2010.

Revision History

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

Changes from B Revision (March 2018) to C Revision **Page**

- Changed title from "Filtering and Signal Processing With Low Energy Accelerator (LEA) on MSP FRAM Microcontroller" to "Filtering and Signal Processing Reference Design Using MSP430 FRAM Microcontroller" 1
- Changed block diagram 1
- Changed BOOSXL-SHARP128 Graphics LCD BoosterPack image 10
- Added the author William Goh 20

Changes from A Revision (January 2017) to B Revision **Page**

- Changed Code Composer Studio™ v6.1 to Code Composer Studio™ v6.2 11

Changes from Original (March 2016) to A Revision **Page**

- Changed information regarding the generation of FIR coefficients from "The FIR coefficients were generated from http://t-filter.engineerjs.com" to "The FIR coefficients are generated using the MATLAB or Python scripts included in the */src/scripts* folder of the software zip package." 6
- Added note regarding the revision of the MSP430FR5994 on the MSP-EXP430FR5994 Launchpad (Code Composer Studio) 11
- Added note regarding the revision of the MSP430FR5994 on the MSP-EXP430FR5994 Launchpad (Embedded Workbench IAR) 11
- Changed "LCS average current consumption" to "LCD average current consumption" 14
- Changed *Figure 13* to updated scope shot 14
- Added detail regarding removal of code for refreshing LCD screen "by uncommenting "#define PWR_BENCHMARK" in the file *application.h*" 15
- Changed *Figure 14* to updated scope shot 15
- Changed the value to "309 µA" for the average current to perform an FFT with LEA and calculating the magnitude while the ADC is continuously sampling 16
- Changed the value to "669 µA" for the average current to perform an FFT without LEA 16
- Changed *Figure 15* to updated scope shot 16
- Changed the value of power savings by using LEA to perform FFT vs assembly code from an approximate "51.5%" to "53.8%" 16
- Changed *Figure 16* to updated scope shot 17
- Changed the value of the average current to perform FIR including ADC and SPI actively writing to the DAC from an approximate "1.16 mA" to "1.15 mA". 17
- Changed *Figure 17* to updated scope shot 18
- Changed *Figure 19* to updated image of the EnergyTrace profile 19
- Changed to updated link for *Software Files* and updated the *.zip* file in the tool folder on TI.com 19

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

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