Low-Energy Accelerator (LEA) Frequently Asked Questions (FAQ)

ABSTRACT

This document answers common questions about the low-energy accelerator (LEA) module built for signal processing on the MSP430™ FRAM MCUs. It describes how to get started with devices that support the LEA module and use it to perform efficient signal processing, matrix multiplication, and other operations.

Contents

1 What is the Low-Energy Accelerator (LEA) for Signal Processing?

The LEA module is a 32-bit hardware accelerator that is new to the MSP430 FRAM MCU family. This accelerator can perform signal processing, matrix multiplications, and other operations such as FIR, IIR, and FFT, that normally take large amounts of time and energy to calculate during application runtime. The LEA module is a low-power coprocessor that performs operations without any CPU intervention and triggers an interrupt when the operation is completed. The MSP430FR599x MCU family has a total of 8KB of SRAM, of which 4KB is shared with the LEA module for data input, output, and parameters. The coprocessor operates based on the commands that are provided during configuration. The commands are pointers to memory input or output buffers and the type of operation. The LEA commands are used in the MSP DSP Library for the most efficient and easy-to-use operation. See Section 5, *Where do I start with the LEA module?*, for more information. For more information on LEA hardware and the MSP430FR599x MCU family, see the *MSP430FR58xx, MSP430FR59xx, MSP430FR68xx, and MSP430FR69xx Family User's Guide*.

2 Does the LEA module support floating or fixed point operations?

The LEA module supports both 16-bit and 32-bit fixed point operations in both the real and complex domains.

3 What devices support the LEA module?

The LEA module is currently available on the MSP430FR599x MCU family with plans to expand to other devices in the future. For more information on the MSP430FR599x MCU family, go to the MSP430FR5994 product page.

4 What performance advantage does the LEA module give me? Where is the benchmark data?

The LEA module has advantages over vector-based software signal processing algorithms that are implemented without an accelerator. The LEA module is both more energy efficient and requires fewer clock cycles to perform a variety of different signal processing algorithms. To view the benchmarking results, see *Benchmarking the Signal Processing Capabilities of the Low-Energy Accelerator on MSP430™ MCUs*. *Setting a new standard for MCU performance while minimizing energy consumption* explores at a higher level how the performance advantages can translate into overall system performance.

5 Where do I start with the LEA module?

Operations of the LEA module can be accessed using the Digital Signal Processing (DSP) Library for MSP MCUs for ease of use and efficiency. The following few questions describe how to get started with the LEA module using MSP DSP Library.

6 What is MSP DSP Library?

The Texas Instruments (TI) Digital Signal Processing (DSP) library is a set of highly optimized functions to perform many common signal processing operations on fixed-point numbers for MSP430 microcontrollers. This function set is typically used for applications in which processing-intensive transforms are done in real-time for minimal energy and with very high accuracy. This library's optimal utilization of the MSP families' intrinsic hardware for fixed-point math allows for significant performance gains.

The DSP Library for MSP MCUs (DSPLib), which is available for all MSP MCUs, automatically selects and uses the LEA module for appropriate functions. The library also provides preprocessor macros to help use the LEA module by automatically placing DSPLib data structures with the correct alignment. For more information on DSP Library functionality such as the preprocessor macros and other functions, see the MSP DSP Library API Guide.

7 How do I install DSPLib and DSPLib GUI?

You can install the DSPLib and DSPLib GUI package in a variety of ways. Go to the MSP DSP Library page or access the latest version of MSP430Ware™ software.

On the MSP DSP Library homepage, click Get Software and choose the DSPLib installer appropriate for the operating system.

Another way to access MSP DSPLib is through the MSP430Ware software package. Download MSP430Ware for the desktop by downloading it from the MSP430Ware home page and installing it to the Code Composer Studio™ IDE directory, or by downloading it from the Code Composer Studio IDE App Center. If running CCS version 6.2 or greater, access MSP430Ware through the new Resource Explorer embedded in CCS.

After installation is complete, click View at the top of the window, click Resource Explorer, and look for MSP430Ware. Next, click Libraries → DSPLib and explore the provided options. In Examples Projects in DSPLib, there are a variety of examples that use the LEA module when compiling to an MSP430FR599x MCU family device target. Launch the DSPLib GUI by clicking DSPLib GUI, then start designing and generating filter structures with coefficients and export them as C-code for the application.

8 Where can I find examples that showcase the LEA module?

Examples that showcase the LEA module are shown in the MSP DSP Library package. MSP DSP Library is included in MSP430Ware v3.60 or greater. You can find examples for MSP DSP Library under MSP430Ware>Libraries>DSPLib>Example Projects. You can also access the examples online through the online version of TI Resource Explorer. Because MSP DSP Library uses the LEA module whenever the peripheral is available on the target device, the user does not have to manually tell DSP Library to use the LEA module. DSP Library does the work for you to enable fast and efficient signal processing.

9 What if I'm used to using Python or Matlab for my filter generation?

The MSP DSP Library provides a variety of Python and Matlab examples that can help you generate various filter coefficients. Go to the home director of your MSP DSP Library install and look for the folder "scripts". This folder includes both Matlab and Python scripts built to help you generate your own filters. Consult the MSP DSP Library API Guide for more information on what packages are necessary for Matlab and Python to support the scripts.

10 How do I use the LEA module in my program?

By default, MSP DSP Library checks the header file of the target device and enables the LEA module for APIs that support the LEA module. If the LEA module is available, then the APIs set up and enable the LEA module and then disable it at the end of any function call. To determine whether or not a function uses the LEA module, see the MSP DSP Library API Guide or see Section 16. Both references indicate whether or not each API uses the LEA module.

Before using the DSPLib APIs, first complete these steps:

1. Specify the input and output vectors and align the vectors to reside within the shared 4KB of LEA SRAM memory.
2. Set up the parameters for the selected function or functions.
3. Call the desired function to execute.

For the first step, specify the input and output memory locations by allocating the array in which the location needs to reside within the shared 4KB of LEA SRAM memory. This can be done by using the DSPLIB_DATA macro provided in DSPLib. For example, to allocate memory for a 256-point Complex FFT, the data input array consists of 256-word real and 256-word complex values which totals to 512 words (1024 bytes). Therefore the following code snippet would be used.

```
#define SAMPLES             256
DSPLIB_DATA(input,MSP_ALIGN_CMPLX_FFT_Q15(SAMPLES))
_q15 input[SAMPLES*2];
```

MSP_ALIGN_CMPLX_FFT_Q15(SAMPLES) is a macro that helps calculate the alignment automatically for a 16-bit complex FFT based on number of samples. For more in-depth examples on how to align data properly for other functions in the MSP DSP Library, see the examples in MSP DSP Library and see the API Guide.

After the application has allocated the memory, the parameters for the selected function must be set. These parameters can vary between functions and are described in the MSP DSP Library API Guide.

After the appropriate parameters are set up, call the selected function, and the library takes care of the rest and returns a result.

11 How do I generate my own filter coefficients using DSPLib GUI?

To generate filter coefficients, open DSPLib GUI, design the filter in the "Design" panel on the left-hand side of the screen, then go to File → Export Filter from the drop-down menu at the top. A .c and a .h file are created in the specified location with the name given in the design panel. A simple way to design a filter in DSPLib GUI is choose a predesigned filter in the Example Filters drop-down menu (see Figure 1).

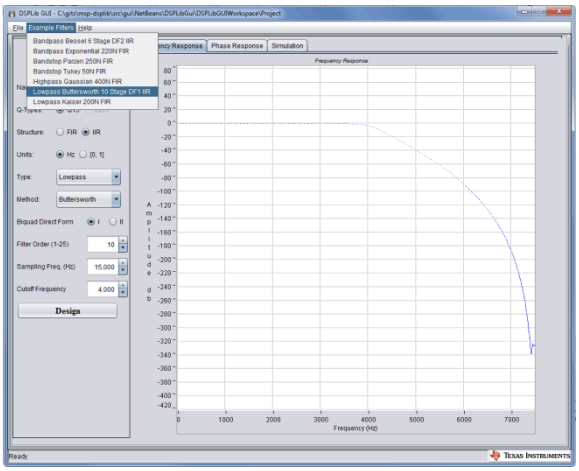

Figure 1. Choosing Predefined Low-Pass Buttersworth IIR Filter in DSPLib GUI

12 How do I apply the coefficients that I generated from DSPLib GUI to my code?

To use the filter coefficients in your code, export the filter from DSPLib GUI using the File → Export Filter drop-down menu at the top, then open an example project from DSPLib. Right-click the project, click "add files", add the generated filter_name.c and filter_name.h files, and then add the line `#include "filter_name.h"` to your source code (see Figure 2). You can then reference the generated coefficients under filter_name.

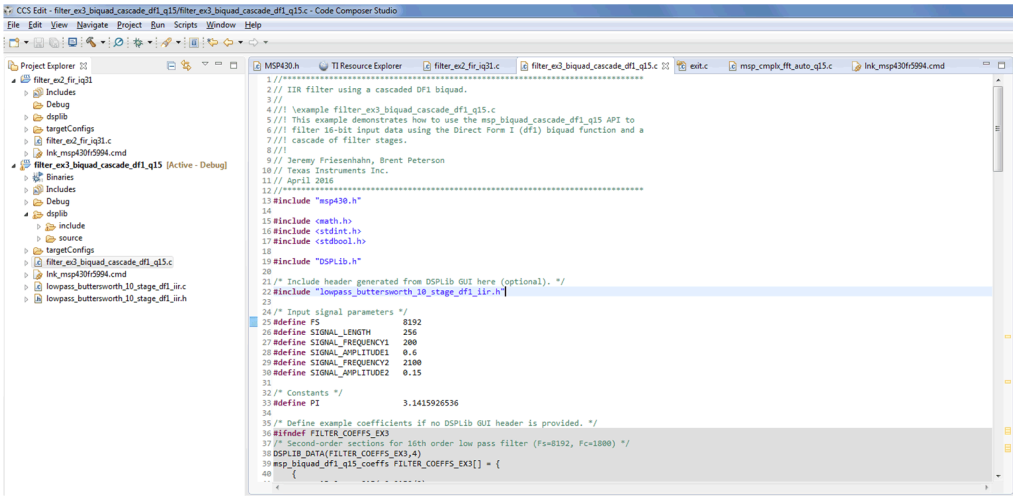

Figure 2. Including Generated Filter Into CCS DSPLib Example Project

13 What other collateral does TI have that leverages the LEA module such as TI Designs reference designs?

TI is always building out a portfolio of example applications, called TI Designs, to showcase a variety of TI technology. In the case of the MSP430FR599x MCU family, TI currently has the following TI Designs, with more to come.

The Filtering and Signal Processing Reference Design using MSP430 FRAM Microcontroller TI 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 module provides a boost of 13.8x over a traditional C implementation for a 256-point complex FFT. The LEA module also provides real-time FIR filtering performance at a high audio sampling rate of 20 kHz.

The EEPROM Emulation and Sensing With MSP430 FRAM Microcontrollers TI Design describes an implementation of emulating EEPROM using ferroelectric random access memory (FRAM) technology on MSP430 ultra-low-power microcontrollers (MCUs) combined with the additional sensing capabilities that can be enabled when using an MCU. The reference design supports both I^2C and SPI interface to a host processor with multiple slave addressing.

As other TI Designs that support the LEA module are released, they will be available on the TI Designs website.

14 Where do I go if I have more questions on the LEA module?

If you have any further questions on the LEA module, search for an answer on the TI E2E™ Community forums. MSP MCUs have a dedicated E2E forum to handle MSP-related questions. Along with a very supportive and knowledgeable community of MSP fans, TI has a team of MSP engineers dedicated to supporting our devices, ensuring that you can get a quick and meaningful response. Go to https://e2e.ti.com/support/microcontrollers/msp430/.

15 How much energy or how many cycles will my function take to compute using the LEA module?

The LEA module overall is more efficient and faster at computing various signal processing algorithms than software-enabled algorithms on both 16-bit and some 32-bit platforms. *Benchmarking the Signal Processing Capabilities of the Low-Energy Accelerator on MSP430™ MCUs* lists the APIs along with a formula to calculate how long the LEA module takes to calculate a function. Keep in mind that with each calculation, there is a small amount of overhead to enable and disable the LEA module.

16 Which APIs use LEA functions among the DSPLib APIs? What functions does the LEA module support?

Function	Description	LEA Support
msp_add_q15	Addition of two real source vectors	Yes
msp_add_iq31	Addition of two real source vectors	Yes
msp_sub_q15	Subtraction of two real source vectors	Yes
msp_sub_iq31	Subtraction of two real source vectors	Yes
msp_mpy_q15	Multiplication of two real source vectors	Yes
msp_mpy_iq31	Multiplication of two real source vectors	Yes
msp_mac_q15	Multiply and accumulate of real source vectors	Yes
msp_mac_iq31	Multiply and accumulate of real source vectors	Yes
msp_neg_q15	Negation of a source vector	Yes
msp_neg_iq31	Negation of a source vector	Yes
msp_abs_q15	Absolute value of a real source vector	No
msp_abs_iq31	Absolute value of a real source vector	No
msp_offset_q15	Constant offset of a real source vector	Yes
msp_offset_iq31	Constant offset of a real source vector	Yes

Function	Description	LEA Support
msp_scale_q15	Scale a real source vector	No
msp_scale_iq31	Scale a real source vector	No
msp_shift_q15	Bitwise shift of a real source vector	Yes
msp_shift_iq31	Bitwise shift of a real source vector	Yes
msp_max_q15	Signed maximum of a source vector	Yes
msp_max_iq31	Signed maximum of a source vector	Yes
msp_max_uq15	Unsigned maximum of a source vector	Yes
msp_max_uq31	Unsigned maximum of a source vector	Yes
msp_min_q15	Signed minimum of a source vector	Yes
msp_min_iq31	Signed minimum of a source vector	Yes
msp_min_uq15	Unsigned minimum of a source vector	Yes
msp_min_uq31	Unsigned minimum of a source vector	Yes
msp_cmplx_add_q15	Addition of two complex source vectors	Yes
msp_cmplx_add_iq31	Addition of two complex source vectors	Yes
msp_cmplx_sub_q15	Subtraction of two complex source vectors	Yes
msp_cmplx_sub_iq31	Subtraction of two complex source vectors	Yes
msp_cmplx_mpy_q15	Multiplication of two complex source vectors	Yes
msp_cmplx_mpy_iq31	Multiplication of complex source vectors	No
msp_cmplx_mpy_real_q15	Multiplication of complex source vector by real source vector	No
msp_cmplx_mpy_real_iq31	Multiplication of complex source vector by real source vector	Yes
msp_cmplx_mac_q15	Multiply and accumulate of complex source vectors	Yes
msp_cmplx_mac_iq31	Multiply and accumulate of complex source vectors	No
msp_cmplx_conj_q15	Conjugation of a source vector	No
msp_cmplx_conj_iq31	Conjugation of a source vector	Yes
msp_cmplx_scale_q15	Scale a complex source vector	No
msp_cmplx_scale_iq31	Scale a complex source vector	No
msp_cmplx_shift_q15	Bitwise shift of a complex source vector	Yes
msp_cmplx_shift_iq31	Bitwise shift of a complex source vector	Yes
msp_matrix_add_q15	Addition of two real source matrices	Yes
msp_matrix_add_iq31	Addition of two real source matrices	Yes
msp_matrix_sub_q15	Subtraction of two real source matrices	Yes
msp_matrix_sub_iq31	Subtraction of two real source matrices	Yes
msp_matrix_mpy_q15	Multiplication of two real source matrices	Yes
msp_matrix_mpy_iq31	Multiplication of two real source matrices	No
msp_matrix_trans_q15	Transposition of a source matrix	No
msp_matrix_trans_iq31	Transposition of a source matrix	No
msp_matrix_neg_q15	Negation of a source matrix	Yes
msp_matrix_neg_iq31	Negation of a source matrix	Yes
msp_matrix_abs_q15	Absolute value of a real source matrix	No
msp_matrix_abs_iq31	Absolute value of a real source matrix	No
msp_matrix_offset_q15	Constant offset of a real source matrix	Yes
msp_matrix_offset_iq31	Constant offset of a real source matrix	Yes
msp_matrix_scale_q15	Scale a real source matrix	No
msp_matrix_scale_iq31	Scale a real source matrix	No
msp_matrix_shift_q15	Bitwise shift of a real source matrix	Yes
msp_matrix_shift_iq31	Bitwise shift of a real source matrix	Yes
msp_fir_q15	Discrete-time convolution of a source vector with real coefficients to apply an FIR filter	Yes
msp_fir_iq31	Discrete-time convolution of a source vector with real coefficients to apply an FIR filter	Yes

Function	Description	LEA Support
msp_cmplx_fir_q15	Discrete-time convolution of a complex source vector with complex coefficients to apply an FIR filter	Yes
msp_cmplx_fir_iq31	Discrete-time convolution of a complex source vector with complex coefficients to apply an FIR filter	Yes
msp_biquad_df1_q15	Second-order direct form 1 biquad filter	Yes
msp_biquad_df2_q15	Second-order direct form 2 biquad filter	Yes
msp_biquad_df2_ext_q15	Second-order direct form 2 biquad filter extended with DC bias and minimum and maximum tracking	Yes
msp_biquad_cascade_df1_q15	Cascaded direct form 1 biquad filter	Yes
msp_biquad_cascade_df2_q15	Cascaded direct form 2 biquad filter	Yes
msp_biquad_cascade_df2_ext_q15	Cascaded direct form 2 biquad filter extended with DC bias and minimum and maximum tracking	Yes
msp_fft_auto_q15	Real forward FFT function with auto-scaling	Yes
msp_fft_fixed_q15	Real forward FFT function with fixed scaling by two at each stage	Yes
msp_fft_iq31	Real forward FFT function without scaling	Yes
msp_ifft_auto_q15	Real result inverse FFT function with auto-scaling	Yes
msp_ifft_fixed_q15	Real result inverse FFT function with fixed scaling by two at each stage	Yes
msp_ifft_iq31	Real result inverse FFT function without scaling	Yes
msp_cmplx_fft_auto_q15	Complex forward FFT function with auto-scaling	Yes
msp_cmplx_fft_fixed_q15	Complex forward FFT function with fixed scaling by two at each stage	Yes
msp_cmplx_fft_iq31	Complex forward FFT function without scaling	Yes
msp_cmplx_ifft_auto_q15	Complex inverse FFT function with auto-scaling	Yes
msp_cmplx_ifft_fixed_q15	Complex inverse FFT function with fixed scaling by two at each stage	Yes
msp_cmplx_ifft_iq31	Complex inverse FFT function without scaling	Yes
msp_copy_q15	Real Q15 vector copy	Yes
msp_copy_iq31	Real IQ31 vector copy	Yes
msp_fill_q15	Real Q15 vector fill with constant	Yes
msp_fill_iq31	Real IQ31 vector fill with constant	Yes
msp_cmplx_fill_q15	Complex Q15 vector fill with constant	Yes
msp_cmplx_fill_iq31	Complex IQ31 vector fill with constant	Yes
msp_deinterleave_q15	Extract a single channel from multiple-channel source	Yes
msp_deinterleave_iq31	Extract a single channel from multiple-channel source	Yes
msp_cmplx_bitrev_q15	Complex bit-reversal function	Yes
msp_cmplx_bitrev_iq31	Complex bit-reversal function	Yes
msp_cmplx_q15	Converts a q15 real vector to have a complex component	Yes
msp_cmplx_iq31	Converts a iq31 real vector to have a complex component	Yes
msp_interleave_q15	Insert a single channel into a multiple-channel destination	Yes
msp_interleave_iq31	Insert a single channel into a multiple-channel destination	Yes
msp_iq31_to_q15	Convert IQ31 vector to Q15 format	Yes
msp_q15_to_iq31	Convert Q15 vector to IQ31 format	Yes
msp_sinusoid_q15	Generate a sinusoid with specified amplitude and frequency	Yes

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

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

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

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

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

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

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

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

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

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

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

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

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