How to Use Low-Energy Accelerator on MSP MCUs

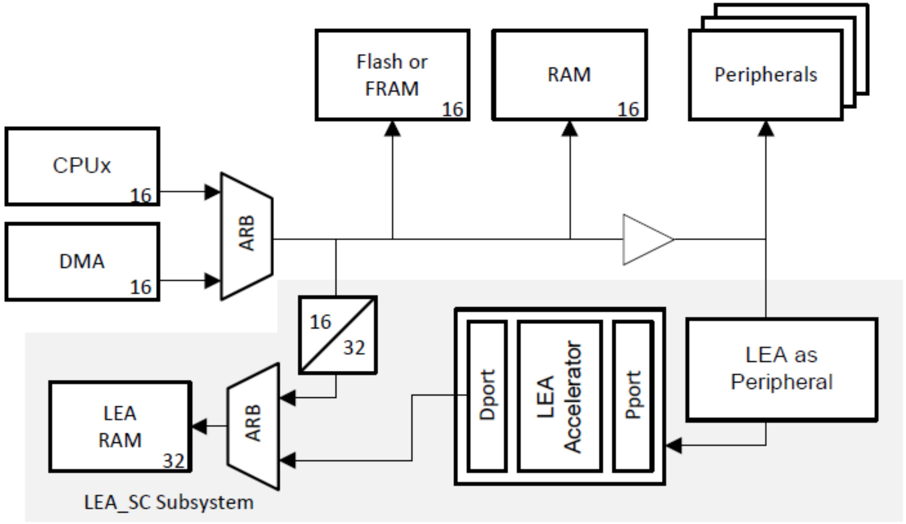
Cash Hao

Sept 2016

Agenda

1. The Overview of Low-Energy Accelerator (LEA)
2. Getting Started Firmware on CCS and IAR
3. Finite Impulse Response (FIR) Example

The Overview of Low-Energy Accelerator (LEA)



LEA System Block Diagram

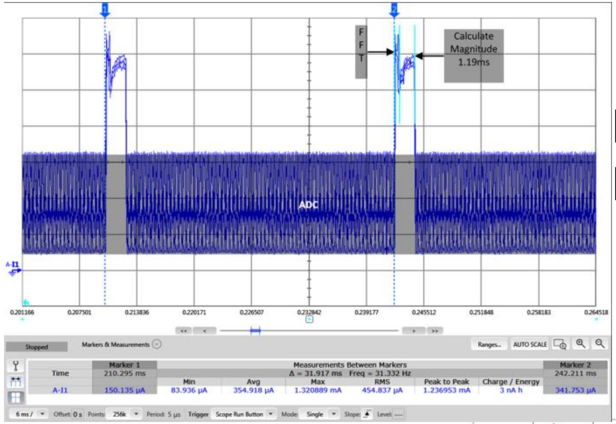
LEA is a hardware math engine that is hosted on the MSP430FR599x MCU family.

1. 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.

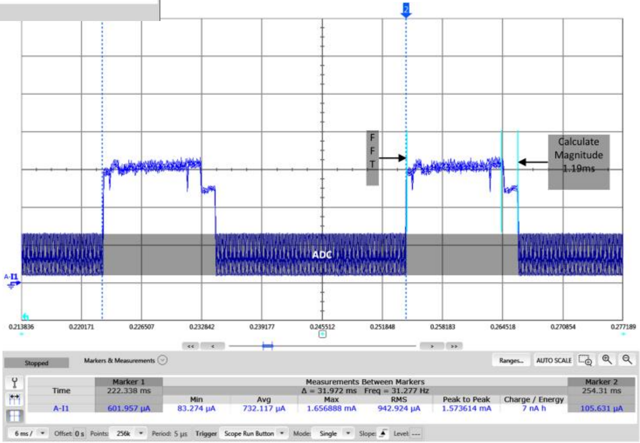
2. The LEA can perform vector-based digital signal processing computations such as FIR or IIR, FFT, matrix multiplications and etc..

The Overview of Low-Energy Accelerator (LEA)

Compared to using the CPU for these operations, the LEA offers up to 19 times faster performance and up to 20 times less energy consumption.



FFT with or without LEA compare graphs.

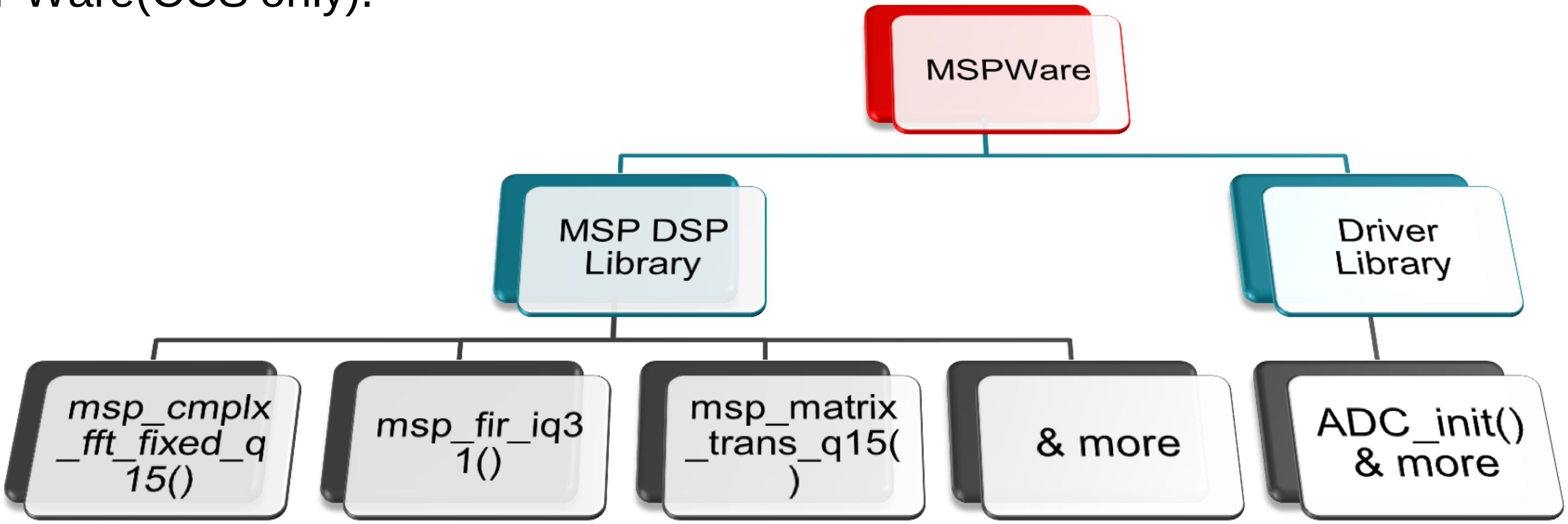


This is a real-time 256-points complex FFT using LEA when compared to the CPU.

	CPU Cycles	FFT Energy (µJ)	Ave Current (mA)
MSP430 Assembly	77086	39.95 µJ	1.377 mA
LEA in LPM0	4730	1.5 µJ	0.913457 mA
Improvement	**16.3x**	**26.6x**	**1.5x**

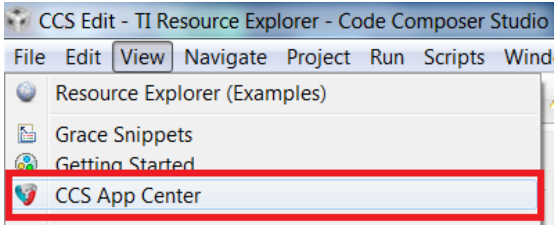
Project Setup

The Digital Signal Processing (DSP) Library for MSP Microcontrollers enables operation of the LEA. The DSP Library for MSP Microcontrollers offers easy-to-use APIs that utilize the benefits of the LEA. The DSP Library is included in the MSPWare(CCS only).



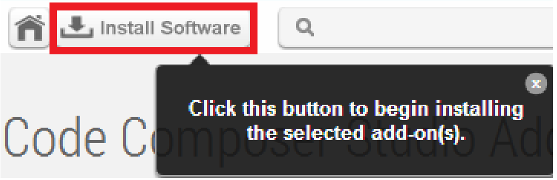
Code Composer Studio™ IDE Project Setup

1. Get a CCS 6.0 or higher version http://www.ti.com.cn/tool/cn/ccstudio
2. Open up CCS
3. Click "View->CCS App Center"



4. Click the checkbox next to MSP430Ware

5. Then click "Install Software" in the top left corner of the app center.

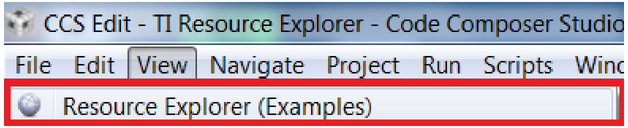


6. Let CCS restart and discover the software. You can now use TI Resource Explorer within CCS

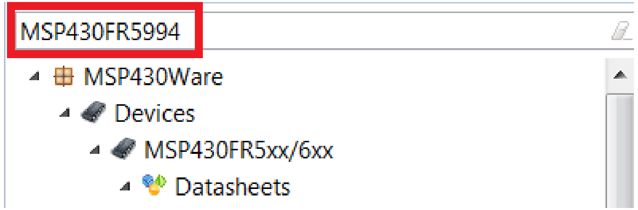
Code Composer Studio™ IDE Project Setup

After setting up MSPWare on your CCS, we can easily get a FFT or FIR examples projects and start using by several steps.

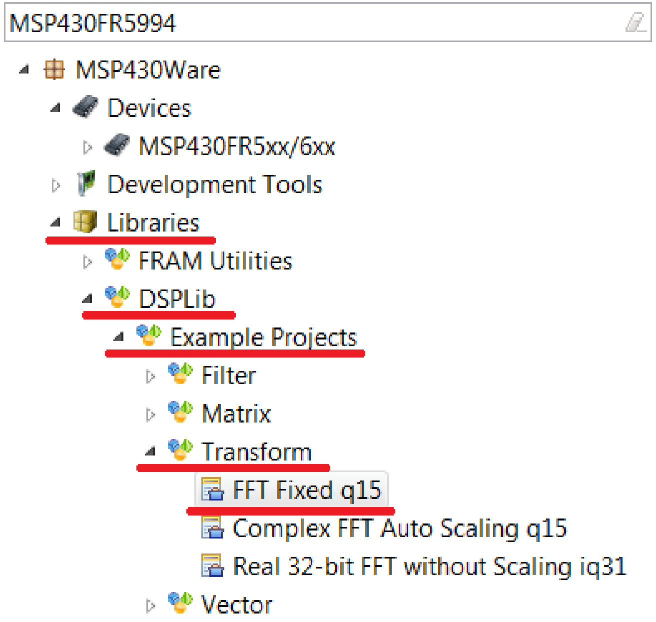
1. Click “View->Resource Explorer(Examples)”



2. Search “MSP430FR5994”

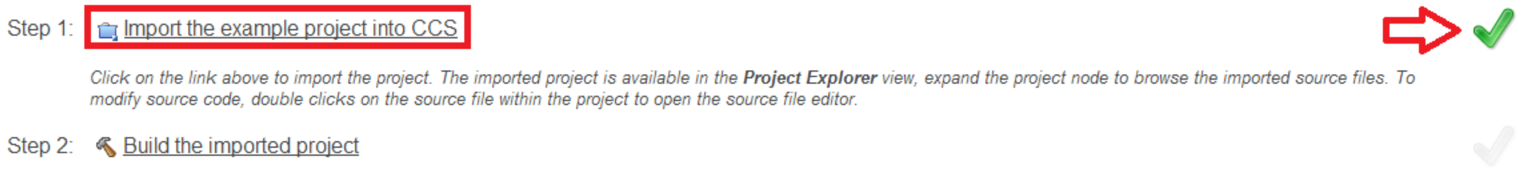


3. Click ”Libraries->DSPLib->Examples Projects->Transform->FFT Fixed q15”

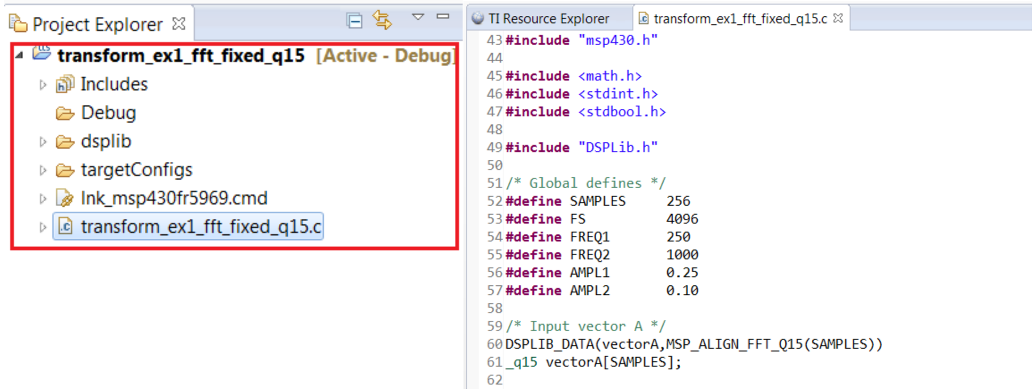


Code Composer Studio™ IDE Project Setup

4. Click “Import the example project into CCS”, make the check mark turn green.



5. Now we get a FFT example project which is ready to use.



IAR Embedded Workbench Project Setup

1. Get a IAR for 430.
2. Get DSP Library Software from http://www.ti.com/tool/msp-dsplib , click “Get software”

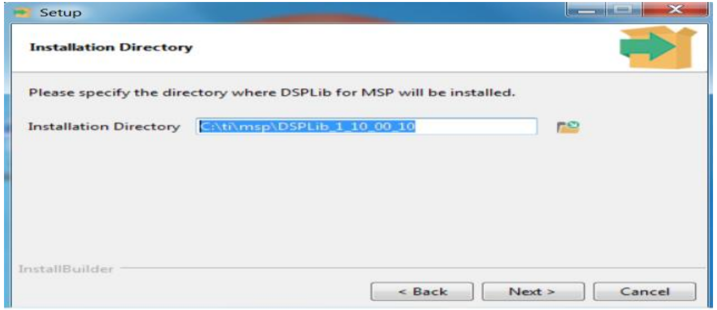
Part Number	Buy from Texas Instruments or Third Party	Status
MSP-DSPLIB: Digital Signal Processing (DSP) Library for MSP Microcontrollers	Get Software	ACTIVE

3. Choose one of these folders to download according to your system.

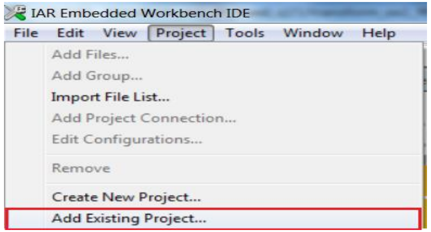
DSPLib Product downloads

Title	Description	Size
DSPLib		
DSPLib_1_10_00_10_windows_installer.exe	DSPLib Installer for Windows	6372K
DSPLib_1_10_00_10_linux_installer.run	DSPLib Installer for Linux	4548K
DSPLib_1_10_00_10.zip	DSPLib Zip	1088K
Release Notes	Release Notes	8K

4. Take windows system for example, download and install the DSPLib.

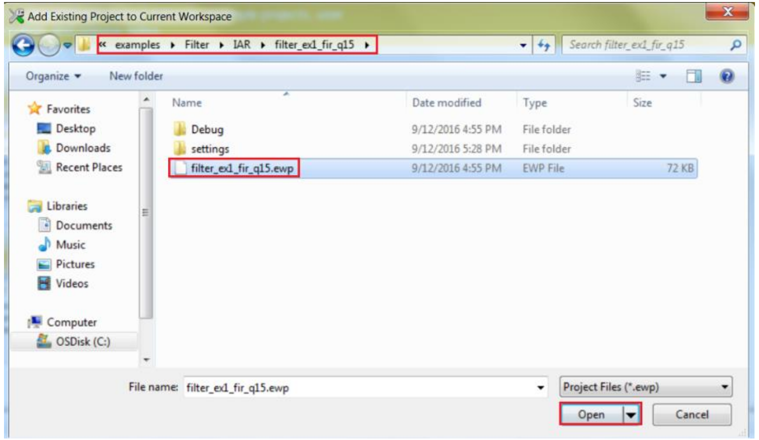


5. Open up IAR, click “Project->Add Existing Project”

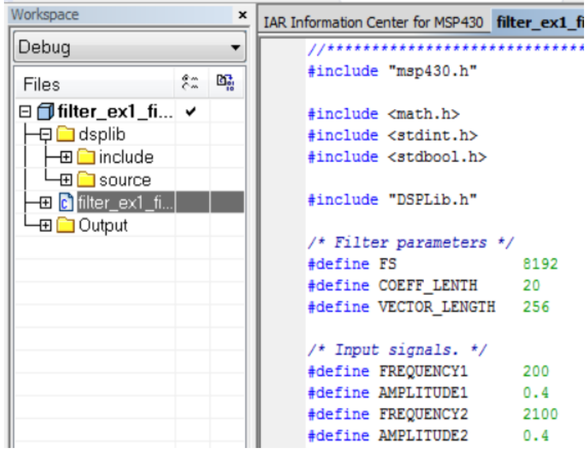


IAR Embedded Workbench Project Setup

6. Browse the folder where you setup DSPLib, and then click "DSPLib_1_10_00_10->examples->Filter->IAR->filter_ex1_fir_q15->filter_ex1_fir_q15.exp", then click "open".



7. Now we get a FIR example project which is ready to use. The other functions are all in the examples folder. We can choose any example projects we want.



Finite Impulse Response (FIR) Example

We now need to design a real 16-bit high-pass FIR Filter. The filter parameters are as follows.

Sample freq.	20000
Taps	100
Passband freq.	2000Hz-10000Hz
Passband gain	1
Passband ripple	1dB
Stopband freq.	0Hz-1700Hz
Stopband gain	0
Stopband ripple	-80dB

We can use the LEA to do this job, according to the following steps.

1. Use the real 16-bit FIR filter API "msp_fir_q15 (const msp_fir_q15_params *params, const _q15 *src, _q15 *dst)"

```
msp_status msp_fir_q15(const msp_fir_q15_params *params,
                       const _q15 *src,
                       _q15 *dst)
```

There are three arguments in this function. They are msp_fir_q15_params *params, _q15 *src and _q15 *dst.

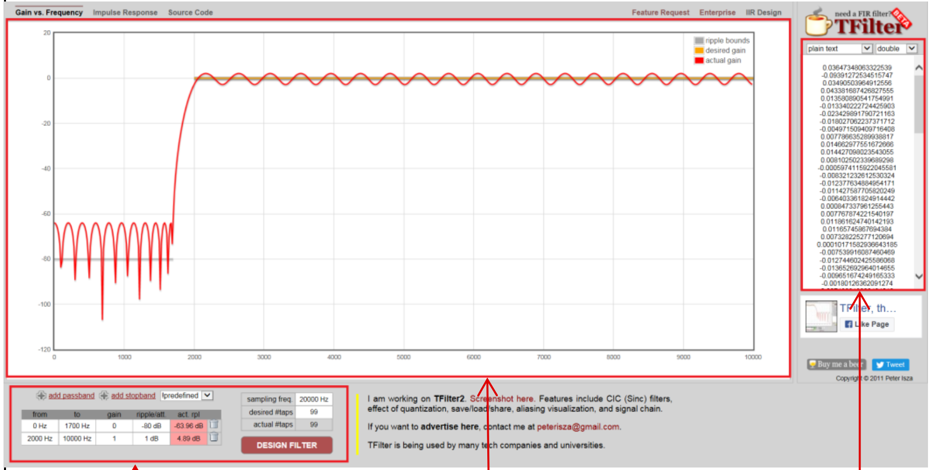
Finite Impulse Response (FIR) Example

2. Write parameter "msp_fir_q15_params *params"

```
typedef struct msp_fir_q15_params
{
    uint16_t length;
    uint16_t tapLength;
    const _q15  *coeffs;
} msp_fir_q15_params;
```

length: Length of output, must be a multiple of two.
tapLength: Number of taps (coefficients) of the filter.
*coeffs: Pointer to coefficients comprising the FIR filter.

We can design our filter coefficients on http://t-filter.engineerjs.com/



Filter parameters

Generated bode diagram

Generated coefficients

Finite Impulse Response (FIR) Example

3. Write parameters "const _q15 *src" and "_q15 *dst" .

_q15 *src is a pointer to the filter input data. Notice that the input data length equals to the result length plus taps length combination.

```
_q15 input[VECTOR_LENGTH+COEFF_LENTH];
```

_q15 *dst is a pointer to the filter output data.

```
_q15 result[VECTOR_LENGTH];
```

4. After writing these parameters, we can use this real 16-bit FIR filter API "msp_fir_q15 (const msp_fir_q15_params *params, const _q15 *src, _q15 *dst)" now.

When MSP430FR599x complete this function, we can find the result after filtering in the address "*dst" points to.

Thanks