

# ***Efficient MSP430 Code Synthesis for an FIR Filter***

Kripasagar Venkat

MSP430

## **ABSTRACT**

Digital filtering can be easily accomplished on the MSP430 using efficient multiplication.[1] The tool accompanying this document automatically converts FIR filter coefficients to MSP430 assembly code that can be used in any application. Horner's method and CSD format is used to accomplish the efficient multiply operations. The performance of the filter on the MSP430 is shown by evaluating the gain across all frequencies. Performance in terms of CPU cycles, code size, and frequency response of low-pass, high-pass, band-pass, band-stop, and notch filters on the MSP430 is shown in [Appendix A](#).

## **Contents**

1	Introduction .....	2
2	FIR Filter Code Synthesizer.....	2
3	References .....	4
Appendix A	FIR Filter Examples .....	5
Appendix B	File List .....	10

## **List of Figures**

A-1	Low-Pass FIR Filter Response .....	5
A-2	High-Pass FIR Filter Response.....	6
A-3	Band-Pass FIR Filter Response .....	7
A-4	Band-Stop FIR Filter Response .....	8
A-5	Notch FIR Filter Response .....	9

## 1 Introduction

An FIR filter, known for its inherent stability and linear phase property, sometimes is an ideal choice for digital filtering.[2] The filter coefficients are always floating point numbers that need to be scaled to the nearest integer for their operation in fixed-point machines, such as the MSP430 microcontrollers.[3] In addition, such a filtering in the absence of a hardware multiplier becomes expensive in terms of CPU cycles. The solution to both these concerns is Horner's method. Horner's method has the ability to perform an integer-float multiply, thus eliminating the need for scaling of the filter coefficients. The efficiency in terms of CPU cycles is achieved by using only shift and add instructions to perform the multiplication.[1] The tool downloadable with this document generates efficient MSP430 code for any FIR filter, given its coefficients. Additionally, a C wrapper file and data file are generated that perform a frequency sweep of the data to verify the filter's performance.

## 2 FIR Filter Code Synthesizer

The FIR filter code synthesizer `FIR_filter_codegen.exe` is a tool that accompanies this document. Input to this tool are the FIR filter coefficients, filter length, bit resolutions (integer and fractional part) for the coefficients, and the sampling frequency.

### 2.1 Input Parameters

When the tool is executed, an interactive command window appears asking for the input parameters previously discussed. The performance of the filter code generated entirely depends on these parameters, and entry in the incorrect format leads to wrong code generation and filter performance.

#### 2.1.1 Filter Coefficients

The filter coefficients of the FIR filter in floating point format must be copied and pasted in the file `FIR_filter_coeff.dat`, and this file must reside in the same directory as the tool.

#### 2.1.2 Filter Length

The filter length corresponds to number of filter coefficients. This number should match the number of coefficients stored in the file `FIR_filter_coeff.dat`. Any mismatches are not reported by the tool and lead to incorrect filter performance.

#### 2.1.3 Bit Resolution for the Filter Coefficients

The filter's performance greatly depends on the resolution chosen to represent the coefficients. Separate bit resolutions are necessary for the integer part and the fractional part. The fractional bit resolution is always chosen to have better resolution as it has a direct impact on performance. This increase in bit resolution results in a proportional increase in code size and CPU cycles. These resolutions (fraction or integer part) are held constant for each coefficient.

#### 2.1.4 Sampling Frequency

The sampling frequency entered should match the sampling frequency that was used to generate the FIR filter coefficients. This parameter is used by the tool to generate sample data across valid frequencies and evaluate the frequency response using the output time samples. Mismatches in this parameter would lead to misinterpretation of the filter's performance.

## 2.2 Output

Once the input parameters are entered, the tool generates a set of files that must be included as an MSP430 project using the IAR Embedded Workbench™.

### 2.2.1 Frequency Sweep Data

The file `FIR_sine_data.dat` is a data file that has sine data for frequencies that range from 10 Hz to  $[(\text{sampling frequency}/2) - 10]$  Hz equally spaced over 44 frequencies. For each frequency, 400 data samples in integer format ranging from  $-2047$  to  $+2047$  are generated. This format is chosen to remain consistent with a 12-bit ADC that is present on some of the MSP430 devices. This data facilitates the verification of the generated FIR filter's frequency response.

### 2.2.2 FIR Filter MSP430 Assembly Code

The file `FIR_filter.s43` contains the MSP430 assembly code that performs the FIR filtering. Function calls are made to this function on a sample-by-sample basis for each of the 400 samples at every frequency. This function returns one output sample which is then used to evaluate the gain at each frequency.

### 2.2.3 Wrapper C File

The file `FIR_filter_wrapper.c` initializes all the variables necessary to simulate the filter's performance on the MSP430. It makes function calls to the assembly function `FIR_filter.s43`. The output samples are accumulated to perform an approximate frequency response by evaluating the gain at the end of 400 samples for each frequency. This normalized gain versus frequency plot is shown in Appendix A for the examples considered. These 44 accumulated gain values are also printed in the Terminal I/O window selected from the View menu of IAR Embedded Workbench. These values can be graphed by entering them in the Excel file `FIR_gain_plot.xls`.

## 2.3 Summary

In this section is a summary of instructions that need to be followed to use the FIR filter synthesizer tool. [Appendix B](#) lists and describes each file included in the accompanying zip file.

To use the FIR filter synthesizer tool:

1. Decompress the zip file that accompanies this document.
2. A sample coefficient file, `FIR_filter_coeff.dat`, is provided in the parent directory. To verify the performance of each filter example, overwrite the coefficients in this sample file with the coefficients of the filter example included in the corresponding directories. To generate the code for any FIR filter, paste the new set of coefficients in the sample file, maintaining the same format.
3. Execute `FIR_filter_codegen.exe` and enter the required parameters. Exact instructions have been provided in an accompanying file, `Instructions.pdf`.
4. The output of the tool is a C-wrapper file, MSP430 assembly code, and a sine data file generated in the same directory.
5. Create a new C project using IAR, add the C and the assembly files, and build. Open the Terminal I/O window from the View menu of IAR and run the code to see the gain at each frequency.

---

**Note:** The C wrapper file uses file operations and `printf()` statements that require a very large code size. Hence, it is recommended to run the project in simulator mode on one of the MSP430 devices that have a larger memory model to test the functionality. The C wrapper file only demonstrates the verification of the FIR filter on the MSP430 using simulated data. In a real application, the MSP430 assembly code file is the only file necessary for FIR filter operation.

---



---

**Note:** The tool generates assembly code that is compatible in all of the MSP430 family of devices. However, if CPUx architecture is chosen, the last instruction, `ret`, in the assembly file `FIR_filter.s43` should be replaced by the instruction `reta`.

---

### **3   References**

1. Venkat, Kripasagar, *Efficient Multiplication and Division Using MSP430*, Texas Instruments, [SLAA329](#)
2. Mitra, S. K., *Digital Signal Processing: A Computer-Based Approach*, Second Edition, McGraw-Hill, 2001.
3. Texas Instruments MSP430 family user's guides

## Appendix A FIR Filter Examples

This appendix shows the performance of basic FIR filters that have been generated using the tool and executing the source files generated. The code size, CPU cycles and approximate plot of the frequency response for each example is shown. In each example, the filter coefficients were all less than one, hence the bits for Integer part are set to zero. However, the tool generates valid MSP430 code if some or all of the coefficients are greater than one.

### A.1 Low-Pass Filter

Filter specifications:

Filter length = 21

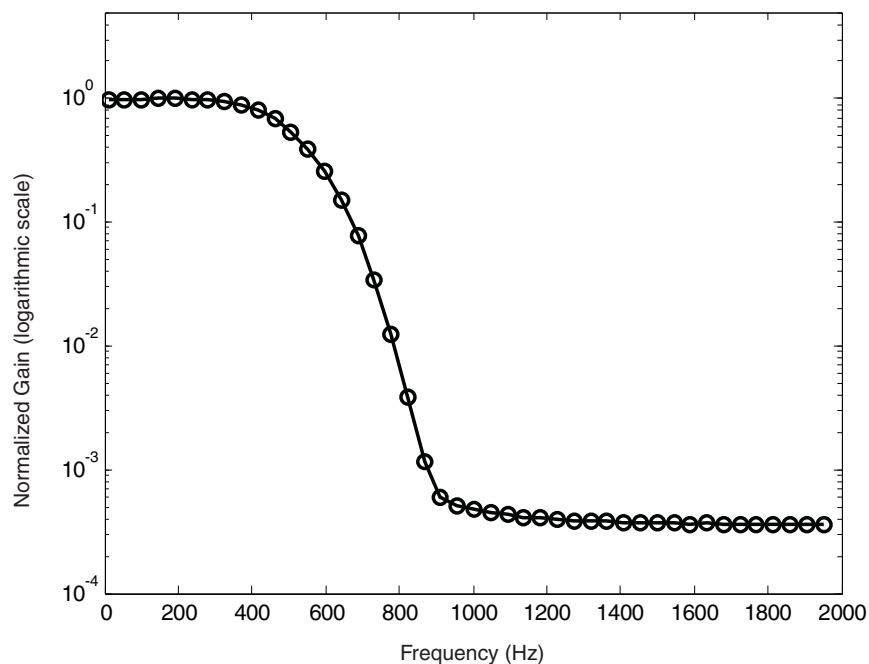
Bits for integer part = 0

Bits for fraction part = 15

Sampling frequency = 4000 Hz

Cut-off frequency = 600 Hz

Figure A-1 shows the approximate frequency response of the filter after the execution of the code generated from the tool on the MSP430. The plot of the normalized gain on the logarithmic scale versus the frequency conforms to its design specifications.



**Figure A-1. Low-Pass FIR Filter Response**

Filter performance:

CPU cycles = 662

Code size in bytes = 1076

## A.2 High-Pass Filter

Filter specifications:

Filter length = 31

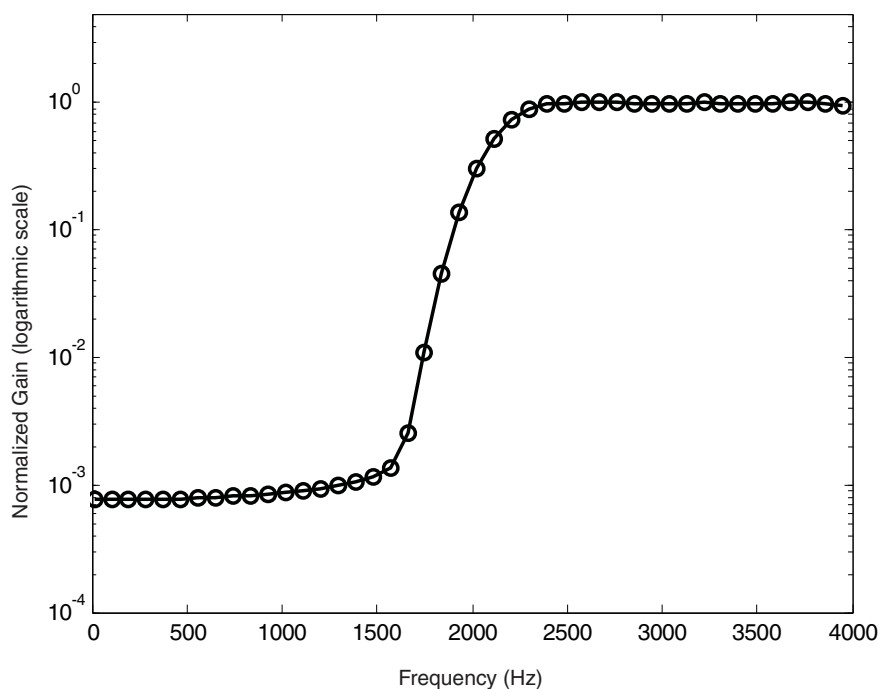
Bits for integer part = 0

Bits for fraction part = 15

Sampling frequency = 8000 Hz

Cut-off frequency = 2000 Hz

Figure A-2 shows the approximate frequency response of the filter after the execution of the code generated from the tool on the MSP430. The plot of the normalized gain on the logarithmic scale versus the frequency conforms to its design specifications.



**Figure A-2. High-Pass FIR Filter Response**

Filter performance:

CPU cycles = 636

Code size in bytes = 976

### A.3 Band-Pass Filter

Filter specifications:

Filter length = 41

Bits for integer part = 0

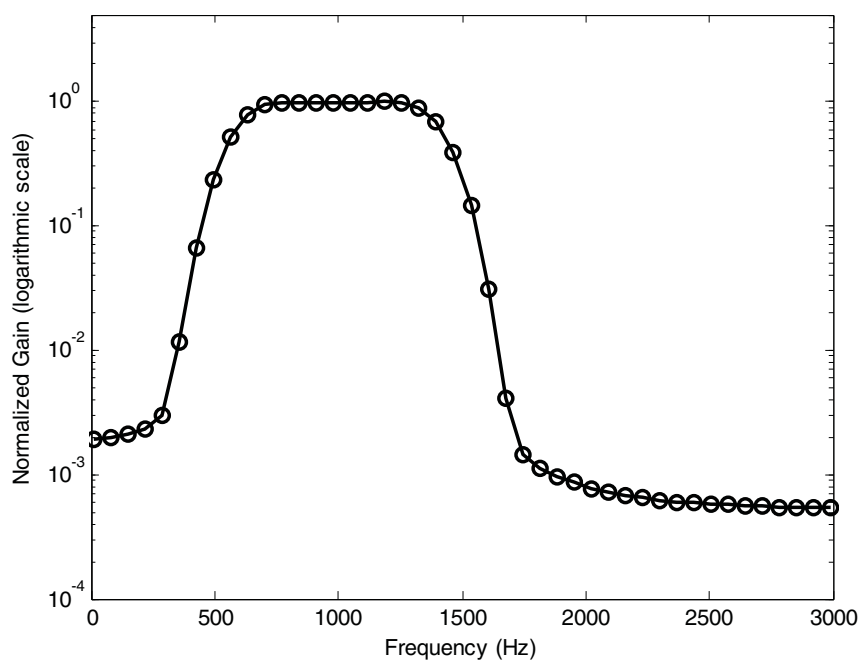
Bits for fraction part = 15

Sampling frequency = 6000 Hz

Lower Cut-off frequency = 500 Hz

Upper Cut-off frequency = 1500 Hz

Figure A-3 shows the approximate frequency response of the filter after the execution of the code generated from the tool on the MSP430. The plot of the normalized gain on the logarithmic scale versus the frequency conforms to its design specifications.



**Figure A-3. Band-Pass FIR Filter Response**

Filter performance:

CPU cycles = 1187

Code size in bytes = 1910

## A.4 Band-Stop Filter

Filter specifications:

Filter length = 31

Bits for integer part = 0

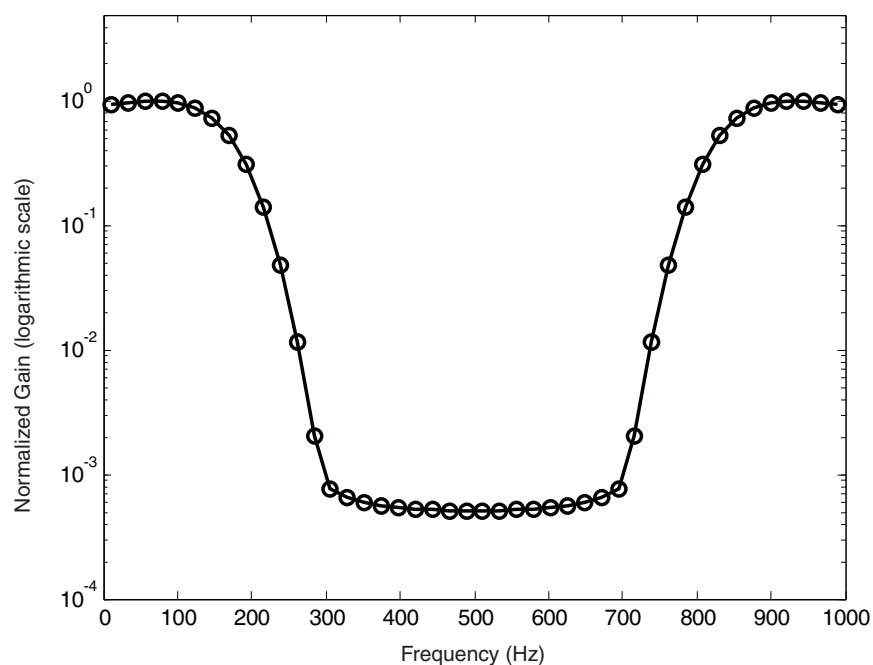
Bits for fraction part = 15

Sampling frequency = 2000 Hz

Lower Cut-off frequency = 200 Hz

Upper Cut-off frequency = 800 Hz

Figure A-4 shows the approximate frequency response of the filter after the execution of the code generated from the tool on the MSP430. The plot of the normalized gain on the logarithmic scale versus the frequency conforms to its design specifications.



**Figure A-4. Band-Stop FIR Filter Response**

Filter performance:

CPU cycles = 531

Code size in bytes = 1187



## A.5 Notch Filter

Filter specifications:

Filter length = 60

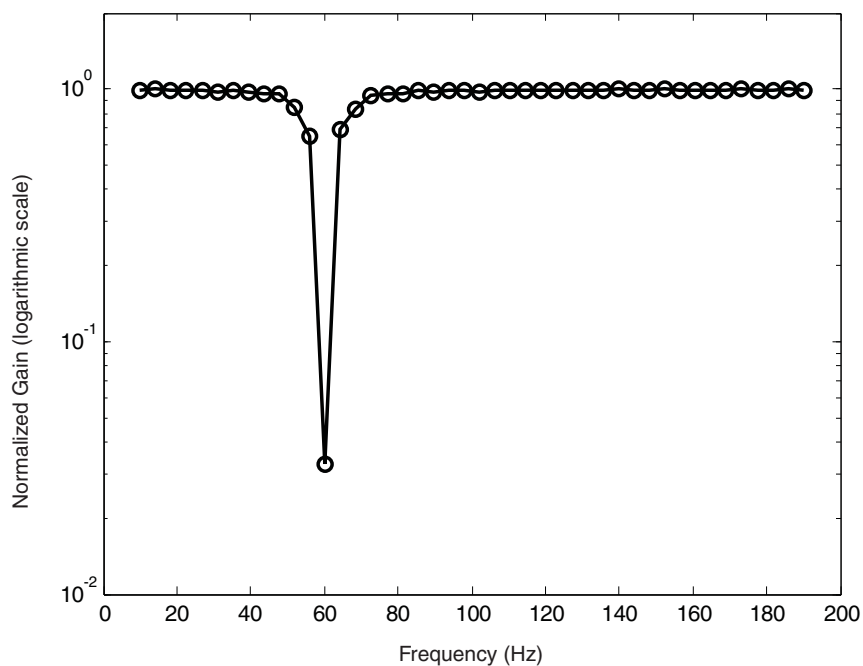
Bits for integer part = 0

Bits for fraction part = 15

Sampling frequency = 400 Hz

Notch frequency = 60 Hz

Figure A-5 shows the approximate frequency response of the filter after the execution of the code generated from the tool on the MSP430. The plot of the normalized gain on the logarithmic scale versus the frequency conforms to its design specifications.



**Figure A-5. Notch FIR Filter Response**

Filter performance:

CPU cycles = 1891

Code size in bytes = 3060

## Appendix B File List

This appendix includes the list of files contained within the zip file that accompanies this document, along with a description of their functionality. Separate directories have been created to include files that belong to each example.

### B.1 File List

FIR\_filter\_codegen.exe

This is the FIR filter code synthesizer utility that generates code to run on the MSP430.

FIR\_filter\_coeff.dat

This file must be present in the same directory as the code synthesizer. It has the FIR filter coefficients in floating point format. A sample file has been included with the zip file, and the user can overwrite this file with new coefficients.

FIR\_filter\_wrapper.c

This is the C wrapper file generated when the tool FIR\_filter\_codegen.exe is executed.

FIR\_filter.s43

This is the MSP430 assembly code file generated by the tool FIR\_filter\_codegen.exe.

FIR\_sine\_data.dat

This file contains the frequency sweep sine data generated by the tool FIR\_filter\_codegen.exe. The file has a total of 17600 samples in integer format ranging from -2047 to +2047. 400 samples are stored continuously for each of the equally spaced 44 distinct frequencies ranging from 10 Hz to  $[(\text{sampling frequency}/2) - 10]$  Hz. It must reside in the same directory of the C wrapper and MSP430 assembly file to evaluate the frequency response.

FIR\_gain\_plot.xls

This file is used to graphically depict the approximate frequency response of the FIR filter when executed on the MSP430. The user can overwrite the parameters and the gain values for each frequency obtained at the Terminal I/O to see the filter's response.

LPF\_21\_coeff.dat

This file contains the filter coefficients used to generate the example in [Section A.1](#).

LPF\_wrapper.c

This file is the C wrapper file filter for the example in [Section A.1](#).

LPF\_filter.s43

This file is the MSP430 assembly code that performs the filtering as specified in the example in [Section A.1](#).

LPF\_sine\_data.dat

This file is the simulated frequency sweep data for the verification of the filter's frequency response for the example in [Section A.1](#).

**HPF\_31\_coeff.dat**

This file contains the filter coefficients used to generate the example in [Section A.2](#).

**HPF\_wrapper.c**

This file is the C wrapper file filter for the example in [Section A.2](#).

**HPF\_filter.s43**

This file is the MSP430 assembly code that performs the filtering as specified in the example in [Section A.2](#).

**HPF\_sine\_data.dat**

This file is the simulated frequency sweep data for the verification of the filter's frequency response for the example in [Section A.2](#).

**BPF\_41\_coeff.dat**

This file contains the filter coefficients used to generate the example in [Section A.3](#).

**BPF\_wrapper.c**

This file is the C wrapper file filter for the example in [Section A.3](#).

**BPF\_filter.s43**

This file is the MSP430 assembly code that performs the filtering as specified in the example in [Section A.3](#).

**BPF\_sine\_data.dat**

This file is the simulated frequency sweep data for the verification of the filter's frequency response for the example in [Section A.3](#).

**BSF\_31\_coeff.dat**

This file contains the filter coefficients used to generate the example in [Section A.4](#).

**BSF\_wrapper.c**

This file is the C wrapper file filter for the example in [Section A.4](#).

**BSF\_filter.s43**

This file is the MSP430 assembly code that performs the filtering as specified in the example in [Section A.4](#).

**BSF\_sine\_data.dat**

This file is the simulated frequency sweep data for the verification of the filter's frequency response for the example in [Section A.4](#).

**Notch\_60\_coeff.dat**

This file contains the filter coefficients used to generate the example in [Section A.5](#).

**Notch\_wrapper.c**

This file is the C wrapper file filter for the example in [Section A.5](#).

**Notch\_filter.s43**

This file is the MSP430 assembly code that performs the filtering as specified in the example in [Section A.5](#).

**Notch\_sine\_data.dat**

This file is the simulated frequency sweep data for the verification of the filter's frequency response for the example in [Section A.5](#).

## IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

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

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI 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 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. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

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

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

### Products

Amplifiers	<a href="http://amplifier.ti.com">amplifier.ti.com</a>
Data Converters	<a href="http://dataconverter.ti.com">dataconverter.ti.com</a>
DSP	<a href="http://dsp.ti.com">dsp.ti.com</a>
Interface	<a href="http://interface.ti.com">interface.ti.com</a>
Logic	<a href="http://logic.ti.com">logic.ti.com</a>
Power Mgmt	<a href="http://power.ti.com">power.ti.com</a>
Microcontrollers	<a href="http://microcontroller.ti.com">microcontroller.ti.com</a>
Low Power Wireless	<a href="http://www.ti.com/lpw">www.ti.com/lpw</a>

### Applications

Audio	<a href="http://www.ti.com/audio">www.ti.com/audio</a>
Automotive	<a href="http://www.ti.com/automotive">www.ti.com/automotive</a>
Broadband	<a href="http://www.ti.com/broadband">www.ti.com/broadband</a>
Digital Control	<a href="http://www.ti.com/digitalcontrol">www.ti.com/digitalcontrol</a>
Military	<a href="http://www.ti.com/military">www.ti.com/military</a>
Optical Networking	<a href="http://www.ti.com/opticalnetwork">www.ti.com/opticalnetwork</a>
Security	<a href="http://www.ti.com/security">www.ti.com/security</a>
Telephony	<a href="http://www.ti.com/telephony">www.ti.com/telephony</a>
Video & Imaging	<a href="http://www.ti.com/video">www.ti.com/video</a>
Wireless	<a href="http://www.ti.com/wireless">www.ti.com/wireless</a>

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265  
Copyright © 2007, Texas Instruments Incorporated