Using Endianess Conversion in the OMAP5910 Device

Matthias Kassner

ABSTRACT

The OMAP5910™ device features a new dual-core architecture from Texas Instruments™ (TI) that is optimized for multimedia applications in a low-power environment. It couples two processors—a TI enhanced TI925T™ general-purpose processor and an ultralow-power TMS320C55x™ (C55x™) DSP—with a rich set of peripherals and powerful interfaces to achieve optimal performance.

Within the OMAP5910 device, the TI925T operates in little-endian mode, whereas the C55x DSP uses the big-endian data format. Consequently, endianess conversion is required for certain data transfers between these two processors.

This application note discusses the basics of the different endianess data formats, provides an overview of the OMAP5910 architecture involved in endianess conversion, and outlines configuration options.

Although this application note is written for the OMAP5910 device, it applies equally to the OMAP5912™, OMAP1510™, and OMAP1610™ devices.

Contents

Figures

Tables

1 Data Formats

When storing data in memory, each data item is stored at one or more memory addresses. The number of memory addresses required to store an item depends on two factors:

- The size of the data item (8-bit, 16-bit, 32-bit…)
- The addressed data size, also referred to as the minimum addressable unit (MAU)

The addressed data size equals the size of the data item corresponding to a single address. In the common case of a byte-addressable architecture, this size equals one byte (8 bits). The OMAP5910 device is an example.

More generally, the number of memory locations required to store an item is determined by the quotient between the data size and the addressed data size. For example, storing a 16-bit data item requires two address locations for 8-bit addressable architectures, but only one address location for 16-bit addressable architectures.

Storing data items to memory is unambiguous as long as it requires exactly one memory location, meaning that the data size equals the addressed data size. For instance, this is the case when storing a 1-byte data item on a byte-addressable architecture.

If the data size does not equal the access size, two possibilities exist:

- The data size is smaller than the addressed data size; for instance, when storing a bit (C-type Boolean) on a byte-addressable architecture. This usually does not pose a problem because there is a nearly universal consensus to store such items right-justified and to fill the remaining positions with 0s. For instance, when storing a single bit using 1 byte of memory, the bit is allocated at the rightmost (least significant) bit position and the remaining 7 bits are set to 0.
- The data size is larger than the addressed data size; for instance, when storing a 32-bit integer on a byte-addressable architecture. This is a different situation because a data item requires more than one memory location to be stored.

For this case, two approaches have evolved over time. The difference between the two approaches centers on which part of the data word is stored at the first memory location, that is, at the location with the lowest address.

In big-endian data format, the most significant part is stored first and the least significant part last. In little-endian data format, the least significant part is stored first and the most significant part last.

Table 1 shows examples of how different data items are stored in big- and little-endian architectures. These examples assume a byte-addressable architecture.

Table 1. Little-Endian Versus Big-Endian Data Format

Endianess Examples						
Address	32-bit: 0x1234:5678		16-bit: 0x1234		8-bit: 0x12	
	Little Endian	Big Endian	Little Endian	Big Endian	Little Endian	Big Endian
0x50000	0x78	0x12	0x34	0x12	0x12	0x12
0x50001	0x56	0x34	0x12	0x34		
0x50002	0x34	0x56				
0x50003	0x12	0x78				

From these examples you can see that different endianess formats only affect data sizes larger than 1 byte. The conclusion is that different endianess formats affect data accesses only when the access size exceeds the addressed data size.

If your system uses two different endianess formats (as is the case with the OMAP5910 device), data transferred between the big-endian and little-endian subsystems must sometimes be converted into the respective format. This process is called endianess conversion.

1.1 Data Accesses in Mixed-Endian Systems

In this section, the basic procedures used to access data in mixed-endian systems—in systems using two different endianess formats—are discussed. Although the case of a processor accessing memory is discussed, these considerations can be generalized in principle to any type of host (processor, DMA, etc.) accessing any type of target (memory, peripheral, etc.).

In mixed-endian systems, endianess conversion may be required if a processor accesses a memory region containing data stored in a different endianess format; for example, a big-endian processor accessing data stored in little-endian format.

Whether endianess conversion is required depends on the way the data is accessed. When a processor accesses memory, it typically does so by sending the start address of the data item and the data size (1-byte, 2-byte, 4-byte…). The memory system then fetches the required amount of data from the memory and places it onto the memory data bus.

The actual data retrieving is done by the memory system, so the endianess of the requesting processor has no effect on the fetched result. This means that the same value is read by a big- or little-endian processor, as long as the same start address and the same access size are used.

Both big- and little-endian processors read (and write) the same values from (to) memory as long as they provide the same start address and read (write) the same amount of data.

Taking an example from Table 1, a 32-bit read from address 0x50000 results in the same value (0x1234:5678) both for the little-endian and big-endian processors independent of the endianess of the memory system.

When is endianess conversion actually needed? It is assumed that the 32-bit value 0x1234:5678 has been stored in little-endian data format at address 0x50000 as shown in Table 1. It is also assumed that a big-endian processor wants to read this 32-bit data using two 16-bit read operations.

The processor issues a first 16-bit read request to address 0x50000 and reads back the value 0x5678. It then issues a second read request to address 0x50002 and reads back 0x1234. As the processor uses big-endian data format, it interprets the retrieved data as 0x5678:1234. This, however, may not be what had been expected.

Generally, the use of different endianess formats poses problems only if different data sizes are used to access the data. As an example, if a 32-bit value has been stored in little-endian format and is read with two 16-bit accesses by a big-endian processor, then the 16-bit words are swapped. Similarly, when a 16-bit value is read with the two 8-bit accesses, the two 8-bit values are swapped.

In mixed-endian systems it is not always possible to avoid accessing data with such differing access sizes. A process is needed in those cases to reorder the retrieved values into the expected order. This process is called endianess conversion.

1.2 Endianess Conversion in Mixed-Endian Systems

Endianess conversion is required in mixed-endian systems when two processors access shared data using different endianess formats and different access sizes. This means that:

- The endianess format of the processors accessing shared data must be different. This is the case, for instance, when one processor stores data in memory using little-endian format and another processor reads the same data using big-endian data format.
- The access size used by the processors accessing shared data must be different. This is the case, for instance, when one processor stores data using 32-bit accesses while the other processor reads the same data with 16-bit accesses.

When both of these conditions are met, endianess conversion is required. The effect of using different data access sizes on the required endianess conversion can be demonstrated using a data item stored as a 32-bit word as an example:

- If this 32-bit word is accessed with one 32-bit access, then no endianess conversion is required independent of the endianess of the accessing processor.
- If this 32-bit word is accessed with two 16-bit accesses by a processor using a different endianess format, then the two 16-bit half-words within this 32-bit word must be swapped.
- If this 32-bit word is accessed with four 8-bit accesses by a processor using a different endianess format, then the four bytes within this 32-bit word must be swapped.

Similarly, using a data item stored as a 16-bit half-word as an example:

- If this 16-bit half-word is accessed with one 16-bit access, then no endianess conversion is required independent of the endianess of the accessing processor.
- If this16-bit half-word is accessed with two 8-bit accesses by a processor using a different endianess format, then the two bytes within this 16-bit half-word must be swapped.

Table 2 illustrates these conclusions by showing the results of different accesses from a big-endian processor to little-endian byte-addressable memory storing the 32-bit data word 0x1234:5678.

Table 2. Endianess Conversion Versus Access Size

Address	Little Endian 0x1234:5678	One 32-bit Read Access	Two 16-bit Read Accesses	Four 8-bit Read Accesses
0x50000	0x78	0x1234:5678	0x5678	0x78
0x50001	0x56			0x56
0x50002	0x34		0x1234	0x34
0x50003	0x12			0x12

2 OMAP Hardware Architecture

2.1 High-Level Overview

The OMAP5910 architecture consists of several main building blocks aside from the TI925T MPU and the C55x DSP.

A dedicated direct memory access (DMA) engine exists to move data between various sources and destinations. The memory and traffic controller handles all accesses to shared internal and external memories. A rich set of peripherals is provided to communicate with external devices.

Figure 1 gives a high-level view of the OMAP5910 architecture. For the sake of simplicity, it contains only the most important functional units.

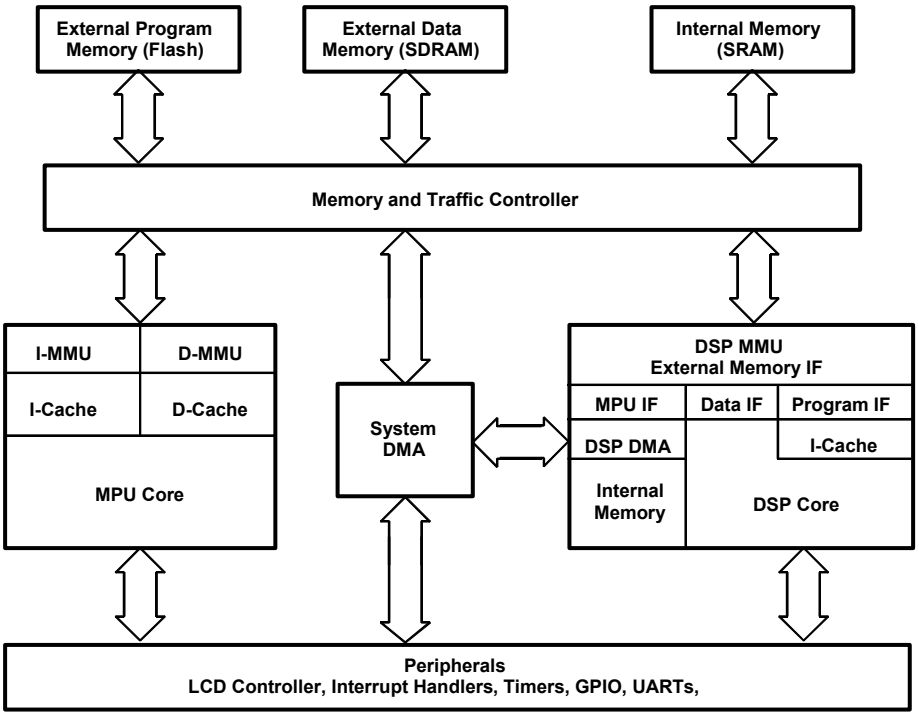

Figure 1. OMAP System Building Blocks

For more details about the OMAP5910 architecture, see the *OMAP5910 Dual-Core Processor Data Manual* (SPRS197C).

2.2 DSP Megacell Architecture

The DSP megacell consists of the C55x DSP core, DSP internal memory, the DSP DMA, multimedia hardware accelerators, the instruction cache, and several interfaces to other OMAP units (see Figure 2).

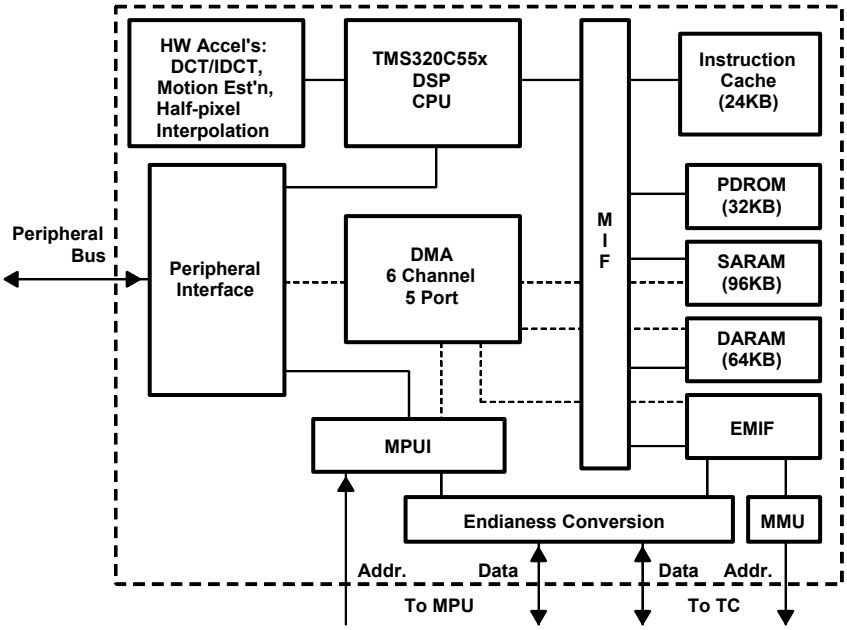

Figure 2. C55x DSP Megacell

2.3 Endianess Within OMAP

Within the OMAP5910 device, the MPU and the memory system operate in little-endian data format, whereas the C55x DSP (including the DSP internal memory) uses big-endian data format.

To establish data sharing between the MPU and the DSP, endianess conversion is required when shared data is accessed by the MPU and the DSP using different access sizes. No endianess conversion is required if both MPU and DSP access data using the same access size.

For efficiency reasons, the endianess conversion is performed at the boundary of the C55x DSP megacell. This means that all functional units outside the C55x megacell use little-endian data format, while all units within the megacell use big-endian data format.

2.4 OMAP Data Paths

To discuss the endianess conversion in the OMAP5910 device, you must first establish which data paths are affected by the different endianess of the two processors.

Three distinct scenarios can be identified where the MPU and the DSP access the same resources, and endianess conversion potentially must be performed.

- MPU or system DMA accesses to the DSP internal memory via the MPU interface. Endianess conversion is performed in the MPU interface.
- DSP or DSP DMA accesses to external memories via the DSP memory management unit (MMU) and the traffic controller. Endianess conversion is performed in the DSP MMU.

- DSP and MPU accesses to shared peripherals. No endianess conversion is performed—both the DSP and the MPU see the same data.

Therefore, endianess conversion is performed in the DSP MMU for accesses from the DSP/DSP DMA to external memories, and in the MPU interface for MPU/system DMA accesses to DSP internal memory. No endianess conversion is implemented for accesses to shared peripherals.

3 Endianess Conversion

3.1 Endianess Conversion in the DSP MMU

The DSP MMU handles all DSP and DSP DMA accesses to external memory. It maps the 24-bit DSP addresses into the 32-bit MPU address space, provides fault and permission checking, and performs endianess conversion. It is configured by the MPU. Figure 3 outlines the role of the DSP MMU within the C55x megacell memory interface.

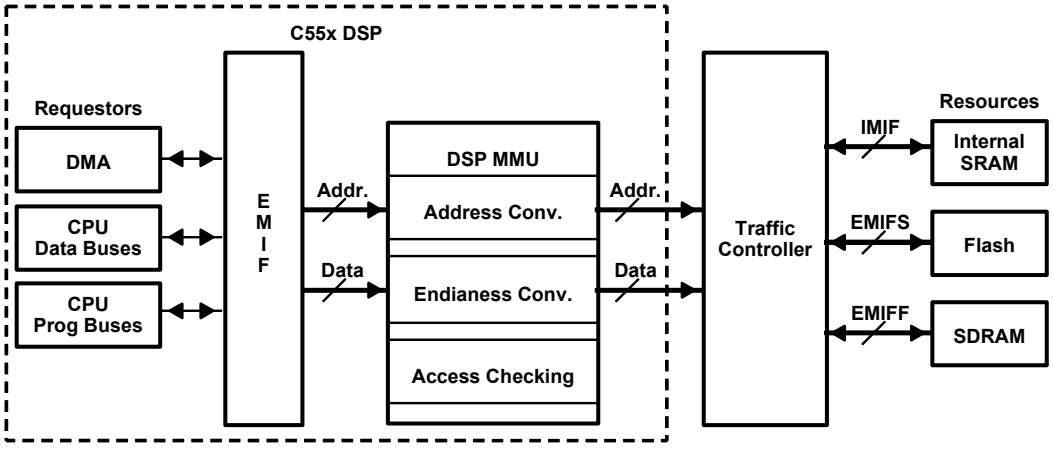

Figure 3. C55x Megacell Memory Interface

3.1.1 Endianess Conversion Architecture

Endianess conversion is performed at the boundary between the C55x DSP and the DSP MMU. The endianess conversion unit splits data accesses into individual bytes and reorders them according to the access type and chosen configuration. Figure 4 outlines the endianess conversion architecture.

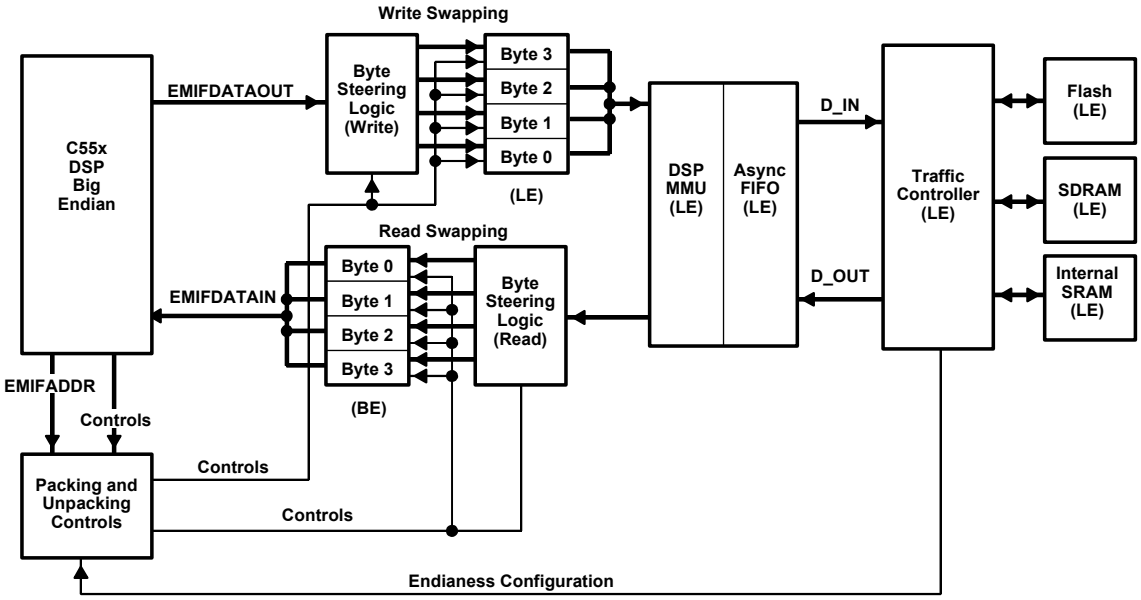

Figure 4. Endianess Conversion Architecture

3.1.2 Configuring DSP MMU Endianess Conversion

The DSP MMU endianess conversion unit is configured by the MPU using the DSP MMU endianess control register located at address 0xFFFE:CC34 (see Table 3).

Table 3. DSP MMU Endianess Control Register

	31	· · · · ·	2	1	0
MMU endianess	Unused			Byte swap/Word swap	Enable conversion

The DSP MMU endianess control register contains two configuration bits that control whether, and how, endianess conversion is performed.

The first bit enables or disables endianess conversion. The second bit selects the type of endianess conversion to be performed—either swapping the 16-bit words only or swapping both the 16-bit words and the bytes within the words. Note that the 16-bit word swapping applies only to 32-bit accesses.

Table 4 outlines the function of the two configuration bits.

Table 4. DSP MMU Endianess Control Register Bits

Name	Reset Value	Description
Enable conversion	0	Enables endianess conversion 0 = Endianess conversion is disabled. 1 = Endianess conversion is enabled.
Byte swap/Word swap	0	Selects between byte swapping and word swapping 0 = Swap bytes and words. 1 = Swap words only.

Typically, both the 16-bit words and the bytes within each word are swapped if data has been written as four times 8-bit by the MPU and is read as one 32-bit word by the DSP. In contrast, only the 16-bit words (but not the bytes within them) are swapped when the MPU has written data in two 16-bit accesses and the DSP reads them using one 32-bit access.

No endianess conversion is required if both the MPU and the DSP access the data as 32-bit. Table 5 lists the effects of the different settings.

Table 5. DSP MMU Endianess Conversion Versus Endianess Settings

DSP Read Access to MPU Data Value 0x1234:5678			
Word/Byte Swap	**Conversion Enable**	**16-Bit Access**	**32-Bit Access**
Don't care	0 = Disabled	0x5678	0x12345678
0 = Byte/word swap	1 = Enabled	0x7856	0x78563412
1 = Word swap only	1 = Enabled	0x5678	0x56781234

3.2 Endianess Conversion in the MPU Interface

The MPU interface enables accesses from the MPU (in the case of the OMAP5910 device, a TI925T) or the system DMA to the DSP resources. Two kinds of resources can be accessed: the DSP internal memory and the DSP public peripherals.

The architecture of the endianess conversion unit within the MPU interface is similar to that used within the DSP MMU shown in Figure 4.

3.2.1 Configuring MPU Interface Endianess Conversion

The endianess conversion within the MPU interface is configured by the MPU using the MPU interface control register at address 0xFFFE:C900 (see Table 6).

Table 6. MPUI Endianess Control Register

	31	· · ·	23	22 21	20	· · ·	18	17 16	15	· · ·	0
MPUI control	Unused			Word swap control	Unused			Byte swap control	Unused		

The endianess configuration in the MPU interface offers a wider range of settings to configure the way endianess conversion is performed, because the MPU interface is used to access both DSP peripherals and DSP internal memory. The endianess conversion unit within this interface offers the options of controlling byte swapping and word swapping depending on the access type (memory or peripheral).

Therefore, the MPU interface control register contains two configuration bit fields that control the way the endianess conversion is handled (see Table 7).

Table 7. DSP MPUI Endianess Control Register Bits

Name	Reset Value	Description
Word swap control	00	Enable/disable word swapping 00: Word swap for all accesses 01: Word swap only for peripheral accesses 10: Word swap only for memory accesses 11: No word swap
Byte swap disable	11	Enable/disable byte swapping 00: No byte swap 01: Byte swap only for peripheral accesses 10: Byte swap for all accesses 11: Byte swap only for memory accesses

Table 7 outlines the way that word swapping and byte swapping can be individually enabled or disabled depending on the access type.

- Access to memory only: Word or byte swapping is performed for memory accesses but not for peripheral accesses.
- Access to peripherals only: Word or byte swapping is performed for peripheral accesses but not for memory accesses.
- Both memory and peripheral accesses: Word or byte swapping is performed for all accesses.

In order to decide which settings apply to memory accesses in your application, see Section 1.2.

When using the same access size on the MPU and the DSP, no endianess conversion is required. In this case, disable both word swapping and byte swapping. Otherwise, turn on word and byte swapping according to the specifics of your application.

For the DSP peripherals, typically neither byte nor word swapping needs to be performed, as they are bit fields of a predefined size rather than numbers. The results of the different endianess settings for MPUI memory accesses are shown in Table 8.

Table 8. MPUI Endianess Conversion Versus Endianess Settings

MPU Read Access to DSP Data Value 0x1234:5678			
Word Swap	**Byte Swap**	**16-Bit Access**	**32-Bit Access**
OFF	OFF	0x1234	0x12345678
OFF	ON	0x3412	0x34127856
ON	OFF	0x1234	0x56781234
ON	ON	0x3412	0x78563412

To summarize, the recommended endianess conversion setting for the MPU interface is:

- Enable word swapping for memory accesses (word swap control = 01).
- Disable byte swapping both for memory and for peripheral accesses (byte swap control = 00).

4 Conclusion

The OMAP5910 deice provides a flexible way to handle endianess conversion between the big-endian C55x DSP subsystem and the little-endian TI925T MPU. This conversion is performed at the boundary of the DSP subsystem.

Two distinct conversion units exist—one associated with the DSP MMU and one with the MPU interface. They are implemented as dedicated hardware controlled by configuration registers.

Several configuration options exist to configure these endianess conversion units. These options are controlled by the MPU using two configuration registers—the DSP MMU endianess control register and the MPU interface control register. Endianess conversion can be configured with these registers according to the specific needs of the application.

5 References

1. *OMAP5910 Dual-Core Processor Data Manual* (SPRS197C)

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.

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

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

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