TMS320VC5510

Digital Signal Processor

Silicon Errata

SPRZ008K
September 2001 − Revised May 2004

REVISION HISTORY

This revision history highlights the technical changes made to the SPRZ008J revision to make it an SPRZ008K revision.

Scope: This document has been reviewed for technical accuracy; the technical content is up-to-date as of the specified release date.

PAGE(S) NO.	ADDITIONS/CHANGES/DELETIONS
5	Section 1.1: − replaced “Quality and Reliability Conditions” section with “Device and Development Tool Support Nomenclature” section

Contents

1 Introduction

This document describes the silicon updates to the functional specifications for the TMS320VC5510, silicon releases 2.1 and 2.2. Please consult your local sales representative if you need information concerning previous silicon revisions. Issues related to CPU operation are documented in the *TMS320C55x DSP CPU Programmer's Reference Supplement* (literature number SPRU652).

A quick reference table (Table 1) is included so that advisory descriptions can be quickly located from the short description of each advisory. The advisory number in the first column of the table is referenced in the title of each advisory description that follows in Section 2.

Table 1. Quick Reference Table

Advisory Number		Silicon revision:	2.1	2.2
		TI boot code revision:	2.2	2.2
	Instruction Cache			
IC_3	Ramset Data May Become Corrupted During Competing Linefills		X	X
IC_4	Two-Way Set-Associative Line Valid Bit Set Incorrectly		X	fixed
IC_5	2-Way Misses to the Same Line Can Become Corrupted		X	fixed
	Enhanced Host Port Interface (EHPI)			
EHPI_1	HPIA Address Read Does Not Increment		X	X
EHPI_12	HPID Read Following a HPID Write While HRDY Low Corrupts the Read		X	X
EHPI_13	HPIC/HPIA Access Following an Autoincremented HPID Write Causes Next HPID Address to Increment to the Incorrect Address		X	X
	Emulation			
EMU_1	Emulation Prone to Failure Under Certain Situations		X	X
	Power Management			
PM_1	Repeated Interrupts During CPU Idle		X	X

1.1 Device and Development Tool Support Nomenclature

To designate the stages in the product development cycle, TI assigns prefixes to the part numbers of all TMS320™ DSP devices and support tools. Each TMS320™ DSP commercial family member has one of three prefixes: TMX, TMP, or TMS (e.g., TMS320VC5510). Texas Instruments recommends two of three possible prefix designators for its support tools: TMDX and TMDS. These prefixes represent evolutionary stages of product development from engineering prototypes (TMX/TMDX) through fully qualified production devices/tools (TMS/TMDS).

Device development evolutionary flow:

TMX Experimental device that is not necessarily representative of the final device's electrical specifications

TMP Final silicon die that conforms to the device's electrical specifications but has not completed quality and reliability verification

TMS Fully qualified production device

Support tool development evolutionary flow:

TMDX Development-support product that has not yet completed Texas Instruments internal qualification testing.

TMDS Fully qualified development-support product

TMX and TMP devices and TMDX development-support tools are shipped against the following disclaimer:

"Developmental product is intended for internal evaluation purposes."

TMS devices and TMDS development-support tools have been characterized fully, and the quality and reliability of the device have been demonstrated fully. TI's standard warranty applies.

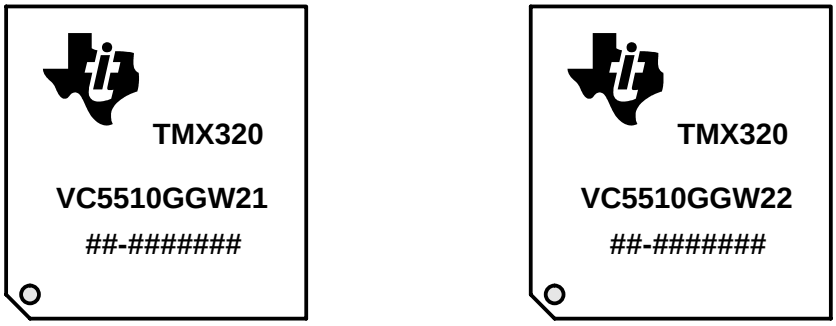
Predictions show that prototype devices (TMX or TMP) have a greater failure rate than the standard production devices. Texas Instruments recommends that these devices not be used in any production system because their expected end-use failure rate still is undefined. Only qualified production devices are to be used.

TMS320 is a trademark of Texas Instruments.

1.2 Revision Identification

The device revision can be determined by the symbols marked on the top of the GGW package are shown in Figure 1 and Figure 2.

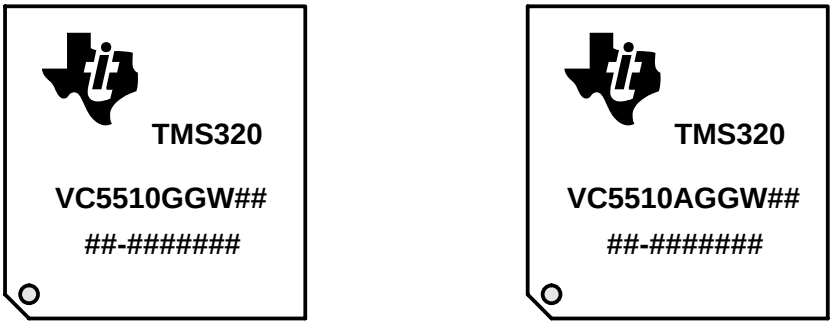
Some prototype devices may have markings different from those shown in Figure 1 and Figure 2 with the device name in the following format: XDVC5510GGWnn where nn is the revision number.



Revision 2.1 Revision 2.2

NOTE: Qualified devices are marked with the letters TMS at the beginning of the device name, while nonqualified devices are marked with the letters TMX at the beginning of the device name.

Figure 1. Example Markings for TMX320VC5510, GGW Package, Revisions 2.1 and 2.2



Revision 2.1 Revision 2.2

NOTE: Qualified devices are marked with the letters TMS at the beginning of the device name, while nonqualified devices are marked with the letters TMX at the beginning of the device name.

Figure 2. Example Markings for TMS320VC5510, GGW Package, Revisions 2.1 and 2.2

2 Silicon Revisions 2.1 and 2.2 Known Design Advisories to Functional Specifications

2.1 Instruction Cache Advisory Descriptions

Advisory IC_3	*Ramset Data May Become Corrupted During Competing Linefills*

Revision(s) Affected: 2.1 and 2.2

Details: The two ramset banks inside the instruction cache are preloaded when the corresponding ramset tag registers' content are updated through I/O access. The ramset preload uses the same line fill mechanism as a cache miss line fill; therefore, there is a conflict of resource when the ramset refill happens while cache miss line fill is ongoing. Although there is logic to arbitrate this conflict, the content of ramset may become corrupted under the following conditions:

1. The instruction cache is enabled with the ramset(s) enabled.
2. The instruction(s) updating the ramset tag register(s) are embedded with other program code in external memory.
3. The instructions which update the ramset tag registers are located in external memory, but are not present in the instruction cache (i.e., generates a cache miss).

The following example describes the assembly code that is likely to encounter this faulty corner case.

Example

```
Address         Instruction
...
Ext Memory      DSP code
Ext Memory      *port(#0x1400) = #0xce2f     ; Configure cache
Ext Memory      bit(ST3,#14) = #1            ; Turn on cache
Ext Memory      *port(#0x1406) = #0x0800     ; Update ramset tag for bank1
Ext Memory      *port(#0x1408) = #0x0801     ; Update ramset tag for bank2
Ext Memory      DSP code
...
```

While the ramset tag is being updated by the I/O bus, the CPU IBQ is prefetching instructions not present in the instruction cache. There are some latencies in completing I/O access to ramset tag registers, the exact time of when ramset tag update happens is not controllable.

Assembler Notification: N/A

Ramset Data May Become Corrupted During Competing Linefills (Continued)

Workaround: Perform the following steps:

1. Put the code which updates the ramset tag(s) in internal memory (non-cacheable memory space).
2. Branch from external memory program code to the ramset tag update code.
3. Continue polling the ramset tag valid bit to make sure the ramset preload has finished.
4. Branch back to external memory for normal program execution after the preload has finished.

Example

```
Address          Instruction
...
Ext Memory       DSP code
Ext Memory       *port(#0x1400) = #0xce2f     ; Configure cache
Ext Memory       bit(ST3,#14) = #1            ; Turn on cache
Ext Memory       goto Preload_ramsets

Preload_ramsets:
Int Memory       *port(#0x1406) = #0x0800     ; Update ramset tag for bank1

Scan0:
Int Memory       TC1 = bit(*port(0x1405),#15) ; Read the Tag Valid bit of ramset bank0
Int Memory       nop_16
Int Memory       if (!TC1) goto Scan0         ; Continue polling if the Tag Valid bit
                                              ; is not 1

Int Memory       *port(#0x1408) = #0x0801     ; Update ramset tag for bank1

Scan1:
Int Memory       TC1 = bit(*port(0x1407),#15) ; Read the Tag Valid bit of ramset bank1
Int Memory       nop_16
Int Memory       if (!TC1) goto Scan1         ; Continue polling if the Tag Valid bit
                                              ; is not 1

Int Memory       goto Back_from_ramset_preload

Back_from_ramset_preload:
Ext Memory       DSP code
Ext Memory       DSP code
...
```

Advisory IC_4 — *Two-Way Set-Associative Line Valid Bit Set Incorrectly*

Revision(s) Affected: 2.1, fixed on 2.2

Details: Under a certain combination of configuration and events, there is a possibility for a line valid bit to be set incorrectly to valid while the instruction packet in the subject line is not actually correct. Therefore, if there is ever a hit to this line, the instruction cache returns an invalid packet and the DSP could try to decode an invalid instruction packet. The following configuration and events are necessary for this issue to occur:

1. The Instruction Cache is configured in 2-way, set-associative mode. Note that ramset(s) can be enabled and are not associated with this issue.
2. CPU and EMIF clocks are the same frequency.
3. DSP program is in either external SDRAM or internal RAM.
4. The Instruction Cache has been globally flushed during the invocation of a new program. (The DSP's program is changed and the instruction cache is flushed.)
5. There is, particular to this bug, a combination of out-of-order DSP accesses to a cache line (speculative branch) and specific timing-related events that are beyond the scope of this document.

Assembler Notification: N/A

Workaround: Due to the timing nature of the problem, the problem does not occur if the DSP program is located in the EMIF and the CPU clock rate is at least twice the EMIF clock rate.

Advisory IC_5 — *2-Way Misses to the Same Line Can Become Corrupted*

Revision(s) Affected: 2.1, fixed on 2.2

Details: A failure may occur when a speculative branch occurs inside the DSP which coincides with a miss to the same line inside the I-cache.

Within the C55x DSP core, the Instruction Buffer Queue (IBQ) always tries to stay several instruction fetches ahead of the decoder. Any time a conditional branch occurs within the DSP, the false condition would be to continue executing the next instruction, which is already in the IBQ. Therefore, in anticipation that the conditional test might be true, the IBQ sometimes fetches the instruction from the "true" condition as well. This allows the necessary code to exist within the IBQ for quick access, regardless of whether the condition proves to be true or false. This operation of fetching the instructions for both true and false conditions is referred to as "speculative fetching".

2-Way Misses to the Same Line Can Become Corrupted (Continued)

Since the instructions are fed into the IBQ 4 bytes per fetch, but are extracted from the IBQ at a variable rate of 1 to 6 bytes per instruction, the IBQ occasionally stops fetching when it is sufficiently far ahead of the instruction decoder.

- For relative branches, a speculative fetch happens for offsets of 0x20 to 0x27.
- For absolute branches, speculative fetches always occur.
- For conditional goto insturctions, the IBQ (instruction buffer queue) fetches the target address prior to the condition being evaluated.

This bug exists if the speculative fetches and normal execution fetches miss within the same I-cache line and coincide with a specific sequence.

Each I-cache line consists of 16 bytes, which are addressed in a sequence of 4 bytes, addressable on 4-byte boundaries.

Consider the following program sequence:

```
if (cond false) goto A3
...
...
A1          ...
A2          ...
            ...
A3          ...
```

Typically, when A1 is returned from the I-cache, A2 is requested next as it is linear code. Also, when the conditional goto is decoded, the DSP requests A3 for the speculative fetch.

The IBQ stores requests prior to decode, so many instructions can be fetched from the I-cache prior to decode. While the IBQ is filling up and addresses are sequentially fetched, the address A1 is eventually requested.

If the IBQ is sufficiently far ahead of the decoder at this point, the DSP stops fetching prior to requesting A2. If the conditional branch is decoded at this point, the speculative fetch occurs next, retrieving A3. The resulting order of fetches would then be A1-A3-A2.

If the test of the conditional branch results in a false condition, *and* the resulting order of A1-A3-A2 matches one of the conditions listed below, *and* the instructions all exist within the same I-cache line, *and* the instruction request from the cache results in a cache miss, *then* the program *may* become corrupted.

2-Way Misses to the Same Line Can Become Corrupted (Continued)

A1	A3	A2	
0	0	8	
0	0	C	
0	8	0	
0	8	4	
0	C	0	
0	C	4	(This is the software example used below.)
0	C	8	
4	0	C	
4	4	C	
4	C	0	
4	C	4	
4	C	8	

Addresses coinciding with this sequence do not necessarily cause a problem since the corner condition depends on the latency of the target memory and the time when the DSP issues the program request. In order to ensure that this bug is not present, these sequences need to be avoided.

The hardware corner case is very rare (17 out of 100 million cycles of targeted random simulation) since it is based on protocol interactions between the DSP and target memory, IBQ state, and the I-cache line needs to be a miss. Of all of the theoretical values for A1, A2, and A3 within the same I-cache line, only those above have the potential to exhibit the error. (Note that in isolation and fixing this bug, all possible combinations have been analyzed and simulated.) However, since this corner case is difficult to isolate in software, speculative fetches need to be isolated entirely to avoid this bug.

If this bug occurs, an invalid 32-bit instruction packet is returned by the I-cache that causes the program counter to become corrupted.

2-Way Misses to the Same Line Can Become Corrupted (Continued)

Absolute Conditional Branch is the failing test case shown in the following example.

Example

```
                MAIN_2
                    .align 16
0x400000                    nop
0x400001                    nop
0x400002                    nop
0x400003                    goto next0
0x400005                    nop
0x400006                    nop
0x400007                    nop
0x400008        next0       nop
0x400009                    nop
0x40000a                    nop
0x40000b                    nop
0x40000c                    nop
0x40000d                    nop
0x40000e                    repeat (#79)
0x400010                    nop
0x400011                    nop
0x400012                    if (ac1 != #0) goto 0x40002c      ; goto A3
0x400017                    nop
0x400018                    nop_16 || nop_16
0x40001c                    nop_16 || nop_16
0x400020                    nop_16 || nop_16                  ; A1
0x400024                    AC0 = #3                          ; A2
0x400026                    nop_16
0x400028                    nop_16 || nop_16
0x40002c                    AR0 = AR0 + #1                    ; A3
0x40002e                    nop
0x40002f                    nop
0x400030                    nop_16 || nop_16
0x400034                    nop_16 || nop_16
0x400038                    nop_16 || nop_16
0x40003c                    nop_16 || nop_16
```

Note that because of the corner case nature of this bug and the low likelihood of encountering a problem, the instruction cache can be used successfully. However, if a problem is suspected or found, then the following actions can be taken:

1. Disable I-cache
2. Freeze I-cache 32 bytes prior to a conditional branch
3. Freeze I-cache 32 bytes prior to a relative branch ≥ 0x20

Assembler Notification: N/A

Workaround: N/A

2.2 Enhanced Host Port Interface (EHPI) Advisory Descriptions

Advisory EHPI_1	*HPIA Address Read Does Not Increment*

Revision(s) Affected: 2.1 and 2.2

Details: Host reads of the HPIA register returns the base address instead of the incremented address from any autoincrement cycles. Host has no visibility of incremented address.

Assembler Notification: N/A

Workaround: Following a single or multiple autoincremented access, a host must update the HPIA register to initiate the next EHPI access.

Advisory EHPI_12	*HPID Read Following a HPID Write While HRDY Low Corrupts the Read*

Revision(s) Affected: 2.1 and 2.2

Details: Whether in muxed or non-muxed mode, if a data read overlaps with a HRDY low from a previous data write, the EHPI address used for the read access is the same as the one preceding the data write.

The only case not affected by this bus is the non-autoincremented write−>read from the same address. In the case of autoincremented write followed by autoincremented read, normally, the address is incremented after the write, but if the read access is initiated while the HRDY is low, it is not incremented due to the problem. If one more autoincremented read is performed after this, the EHPI address used is twice the incremented version.

Assembler Notification: N/A

Workaround:

- Muxed Mode: Following a sequence of EHPI write(s), perform a HPIA update before initiating an EHPI read access.
- Non-Muxed Mode: Following a sequence of EHPI write(s), add a dummy HPID read before initiating the actual EHPI read access.

Advisory EHPI_13 *HPIC/HPIA Access Following an Autoincremented HPID Write Causes Next HPID Address to Increment to the Incorrect Address*

Revision(s) Affected: 2.1 and 2.2

Details: If any HPIA/HPIC access is pipelined with a previous autoincremented write, the succeeding autoincremented HPID access (whether write or read) uses the same address as the previous write. In other words, the address is not incremented. Only Muxed Mode is affected.

Assembler Notification: N/A

Workaround: Muxed Mode: Following a sequence of EHPI write(s), perform an HPIA update before initializing an EHPI read/write access.

2.3 Emulation Advisory Descriptions

Advisory EMU_1	*Emulation Prone to Failure Under Certain Situations*

Revision(s) Affected: 2.1 and 2.2

Details:

Under certain conditions, the emulation hardware may corrupt the emulation control state machine or may cause it to lose synchronization with the emulator software. When emulation commands fail as a result of the problem, Code Composer Studio™ Integrated Development Environment (IDE) may be unable to start or it may report errors when interacting with the TMS320C55x™ DSP (for example, when halting the CPU, reaching a breakpoint, etc.).

This phenomenon is observed when an erroneous clock edge is generated from the TCK signal inside the C55x™ DSP. This can be caused by several factors, acting independently or cumulatively:

- TCK transition times (as measured between 2.4 V and 0.8 V) in excess of 3 ns.
- Operating the C55x DSP in a socket, which can aggravate noise or glitches on the TCK input.
- Poor signal integrity on the TCK line from reflections or other layout issues.

A TCK edge that can cause this problem might look similar to the one shown in Figure 3. A TCK edge that does not cause the problem looks similar to the one shown in Figure 4. The key difference between the two figures is that Figure 4 has a clean and sharp transition whereas Figure 3 has a “knee” in the transition zone. Problematic TCK signals may not have a knee that is as pronounced as the one in Figure 3. Due to the TCK signal amplification inside the chip, any perturbation of the signal can create erroneous clock edges.

As a result of the faster edge transition, there is increased ringing in Figure 4. As long as the ringing does not cross logic input thresholds (0.8 V for falling edges, and 2.4 V for rising edges), this ringing is acceptable.

When examining a TCK signal for this issue, either in board simulation or on an actual board, it is very important to probe the TCK line as close to the DSP input pin as possible. In simulation, it should not be difficult to probe right at the DSP input. For most physical boards, this means using the via for the TCK pad on the back side of the board. Similarly, ground for the probe should come from one of the nearby ground pad vias to minimize EMI noise picked up by the probe.

Emulation Prone to Failure Under Certain Situations (Continued)

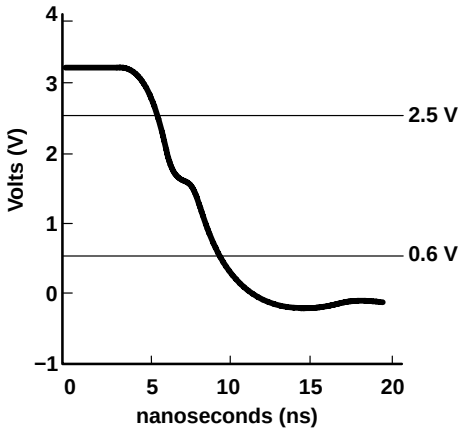


Figure 3. Bad TCK Transition

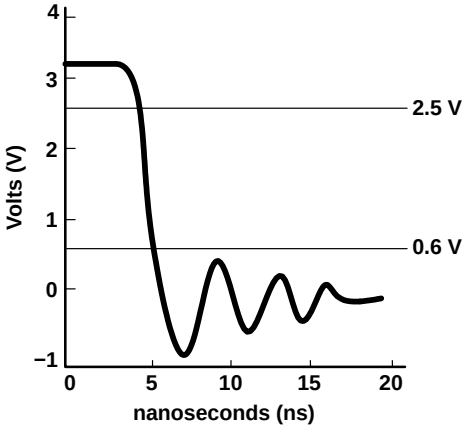


Figure 4. Good TCK Transition

Workaround: As the problem may be caused by one or more of the above factors, one or more of the steps outlined below may be necessary to fix it:

- Avoid using a socket
- Ensure the board design achieves rise times and fall times of less than 3 ns with clean monotonic edges for the TCK signal.
- For designs where TCK is supplied by the emulation pod, use a C55x Emulation Adapter Board, part number DSP8102U. To order a C55x Emulation Adapter Board, please contact the TI Product Information Center (PIC).

2.4 Power Management Advisory Descriptions

Advisory PM_1	*Repeated Interrupts During CPU Idle*

Revision(s) Affected: 2.1 and 2.2

Details: An external interrupt that stays low for an extended period should generate only one interrupt. The interrupt signal should normally be required to go high then low again before additional interrupts would be generated. However, on the 5510, if the external interrupt stays low while the CPU domain enters the idle state, the associated interrupt flag is set again. This causes the CPU to exit the idle state and if the associated interrupt enable bit is set, the interrupt service routine will also be executed.

When the CPU is not in idle, the interrupt responds as expected (only a single interrupt is generated).

Assembler Notification: N/A

Workaround: Limit the low pulse durations of external interrupts so that they are not still asserted when the CPU goes into idle.

3 Documentation Support

For device-specific data sheets and related documentation, visit the TI web site at: **http://www.ti.com**

For further information regarding the TMS320VC5510, please refer to:

- *TMS320VC5510 Fixed-Point Digital Signal Processor* Data Manual (literature number SPRS076)
- *TMS320C55x™ DSP Functional Overview* (literature number SPRU312)

For additional information, see the latest versions of:

- *TMS320C55x DSP CPU Reference Guide* (literature number SPRU371)
- *TMS320C55x DSP Mnemonic Instruction Set Reference Guide* (literature number SPRU374)
- *TMS320C55x DSP Algebraic Instruction Set Reference Guide* (literature number SPRU375)
- *TMS320C55x DSP Peripherals Overview Reference Guide* (literature number SPRU317)
- *TMS320C55x DSP CPU Programmer's Reference Supplement* (literature number SPRU652)
- *TMS320VC5501/5502/5509/5510 DSP Multichannel Buffered Serial Port (McBSP) Reference Guide* (literature number SPRU592)
- *TMS320C55x Assembly Language Tools User's Guide* (literature number SPRU280)
- *TMS320C55x Hardware Extensions for Image/Video Applications Programmer's Reference* (literature number SPRU098)
- *TMS320C55x Image/Video Processing Library Programmer's Reference* (literature number SPRU037)
- *TMS320VC5510 DSP Instruction Cache Reference Guide* (literature number SPRU576)
- *TMS320VC5509/5510 DSP Direct Memory Access (DMA) Controller Reference Guide* (literature number SPRU587)
- *TMS320VC5510 DSP Host Port Interface (HPI) Reference Guide* (literature number SPRU588)
- *TMS320VC5509/5510 DSP Timers Reference Guide* (literature number SPRU595)
- *Using the TMS320VC5510 Bootloader* Application Report (literature number SPRA763)
- *TMS320VC5510 DSP External Memory Interface (EMIF) Reference Guide* (literature number SPRU590)

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