

Daniel Allred,
Applications Engineer
Gaurav Agarwal,
Applications Engineer
Texas Instruments

Software and hardware design challenges due to the dynamic raw NAND market

Abstract

This white paper will highlight the difficulties in developing and maintaining firmware (bootloaders), middleware (OS/application drivers) and hardware IPs for interfacing with raw NAND devices. Determining raw NAND geometry, NAND command set and ECC requirements across NAND manufacturers and NAND product lines, combined with short life cycle of raw NAND, brings significant challenges to both software and hardware designers. Various approaches to resolve the issue will be discussed. Managed NAND as an alternative solution will also be examined.

Introduction

This white paper will highlight the difficulties in developing and maintaining firmware (bootloaders), middleware (operating system (OS)/application drivers) and hardware IPs for interfacing with raw NAND devices, with an emphasis on embedded processor and SoCs. It begins with some history and comparison with other memory devices and then moves on to a more detailed presentation of issues and challenges particular to NAND memories. We provide some recommendations for handling the issues at the various levels and discuss the future of NAND memories in embedded systems.

Background on non-volatile memory technologies

In this section we will detail some of the current non-volatile memory technologies in use today, culminating with the dominant type – NAND flash.

EEPROM

Electrically Erasable Programmable Read-only Memory, or EEPROM, is one of the oldest forms of technology still in use for user-modifiable non-volatile memories. In modern usage, EEPROM has come to mean any non-volatile memory where individual bytes can be read, erased or written independently of all other bytes in the memory device. This capability requires more chip area as each memory cell requires its own read, write and erase transistor. As a result, the size of EEPROM devices is small (64 Kbytes or less).

EEPROM devices are typically wrapped in a low-pin-count serial interface, such as I²C or SPI. Parallel interfaces are now uncommon due to the required larger pin count and resulting larger footprint and layout costs. Like almost all available non-volatile memory types, EEPROMs use floating-gate technology in a complementary metal-oxide-semiconductor (CMOS) process. The floating gate is insulated by oxide on all sides so that charge that is deposited on the gate does not simply flow away. Via different electrical programming techniques, electrons can be placed on or removed from the floating gate, thus storing a single bit of memory.

Flash

Flash memory, which was invented at Toshiba in 1980 (though commercialization and production didn't come until several years later), is a modified form of EEPROM memory in which some operations happen on blocks of memory instead of on individual bytes. This change allowed higher densities to be achieved, as much of the circuitry surrounding each memory cell was removed and then placed around entire blocks of memory cells.

There are two types of flash memory arrays in the marketplace – NOR flash and NAND flash. Though the naming is derived from the internal organization and connection of the memory cells, the two types have also come to signify a particular external interface as well. Both types of memory use floating gates as the storage mechanism, though the operations used to erase and write the cells may be different.

NOR flash

NOR flash was the first version of flash memory and, until about 2005, it was the most popular flash memory type (measured by market revenue)ⁱ. In a NOR flash, the bytes of memory can be read or written individually, but erasures happen over a large block of bytes. Because of the capability to read and write individual bytes, NOR flash devices are not suitable for using with any kind of block error correction. Therefore, there is an implicit requirement that NOR memory be robust to errors. The write/erase cycle endurance of NOR flash is typically 100,000 to 1 million times.

The capability to read individual bytes also means that it is appropriate to act as a random access memory (RAM), and NOR flash devices will typically employ an asynchronous parallel memory interface with separate address and data buses. This allows NOR flash devices to be used for storing code that can be directly executed by a processor (such as in the BIOS of a PC or the bootloader of an embedded system). NOR flash can also be found wrapped in serial interfaces, such as SPI, in which case they act very much the same as SPI EEPROM implementations for reading and writing.

NAND flash

The organization and interface of the NOR flash devices places limitations on how they can scale with process shrinks. With a goal of replacing spinning hard disk drives, the inventor of NOR flash later created NAND flash. The goal at the time was to sacrifice some of the speed offered by NOR flash in order to gain compactness and a lower cost per byteⁱⁱ. This goal has largely been met in recent years, with NAND sizes increasing to multiple gigabytes per die, while NOR sizes have stagnated at around 128 MB. But this has come at a cost, as will be discussed later.

Raw NAND memory is organized into blocks, where each block is further divided into pages. For example, MT29F2G08AACWP has a block size of 128 KBytes (excluding spare area) and 64 pages per block.ⁱⁱⁱ

In NAND memories, read and write operations happen on a per page basis, but erase operations happen per block. The fact that read and write operations are done block-wise means that it is suitable to employ block error correction algorithms on the data. As result, NAND manufacturers have built in spare bytes of

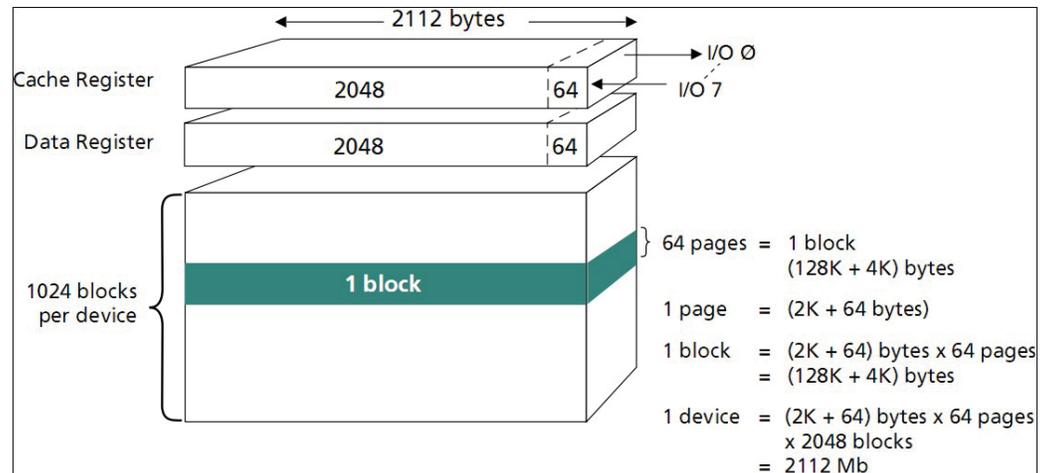


Figure 1. MT29F2G08AACWP NAND memory organization (courtesy Micron Inc.)

memory for each page to be used for holding this and other metadata. NOR flash, on the other hand, does not have such spare bytes.

Also in contrast to NOR flash, the NAND flash interface is not directly addressable and code cannot be executed from it. The NAND flash has a single bus for sending command and address information as well as for sending and receiving memory contents. Therefore, reading a NAND device requires a software device driver of some kind. For example, booting an embedded processor from a NAND flash would require a specialized boot ROM, internal to the processor, to correctly interface to and copy data from a NAND flash at startup.

NAND flash is the underlying memory type for USB memory sticks, memory cards (e.g., SD cards and compact flash cards) and solid state hard drives. In all cases, the raw NAND flash devices are coupled with a controller that translates between the defined interface (e.g., USB, SD and SATA) and the NAND's own interface. In addition, these controllers are responsible for handling a number of important tasks for maintaining the reliability of the NAND memory array.

Raw NAND issues and requirements

Throughout the remainder of this paper, we will focus solely on NAND technologies and the challenges presented by trying to incorporate raw NAND devices into an embedded system. In this section we will look at some of the issues that arise when interacting with raw NAND devices. In the context of embedded systems, we are discussing situations where one or more NAND devices are connected directly to an embedded processor or system-on-chip (SoC) with its own external memory interface or NAND controller.

Errors and error correction

As stated previously, despite being based on the same underlying floating-gate technology, NAND flash has scaled in size quickly since overtaking NOR flash. But this has come at a cost of introducing errors into the memory array.

To increase density, NAND producers have resorted to two main techniques. One is the standard process node and lithography shrinks, making each memory cell and the associated circuitry smaller. The other has been to start storing more than one bit per cell. Early NAND devices could store one of two states in a memory cell, depending on the amount of charge stored on the floating gate. Now, raw NAND comes in three flavors: single-level cell (SLC), multi-level cell (MLC) and, more recently, tri-level cell (TLC). These NANDs differ in the number of charge levels possibly used in each cell, which corresponds to the number of bits stored in each cell. SLC is the original two levels per cell and therefore stores 1 bit of information per cell. MLC uses four levels and stores 2 bits, and TLC uses eight levels and stores 3 bits.

While reducing silicon feature sizes and storing more bits per cell reduces the cost of the NAND flash and allows for higher density, it increases the bit error rate (BER). To overcome the increasing noisiness of this storage medium requires larger and larger error correcting codes (ECCs). An error correcting code (ECC) is redundant data added to the original data. In the event of errors, the combined data allows the recovery of the original data. The number of errors that can be recovered depends on the algorithm used.

For example, the latest SLC NANDs in the market requires 4 or 8 bits ECC per 512 bytes, while MLC NAND requires more than 16 bits ECC per 512 bytes. But four years ago, SLC NANDs only required 1 bit of ECC and the first MLC NANDs only required 4 bits of ECC.

What are the various algorithms and the differences used to implement ECC?

Ideally any ECC algorithm can be used as long as the encoder and decoder matched. The popular algorithms used for NAND ECC are:

- a. Hamming Code: For 1-bit correction^v
- b. Reed Solomon: For up to 4 bits of correction. This is less common^v
- c. BCH: For 4 or more bits of correction^{vi}

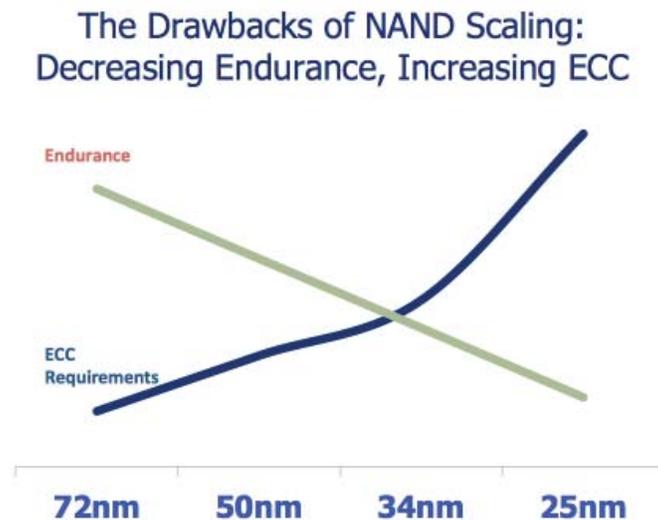


Figure 2: Device issues versus process node shrinks (courtesy Micron)

Where is ECC stored in the NANDs?

Extra memory (called the “spare memory area” or “spare bytes region”) is provided at the end of each page in NAND. This area is similar to the main page and is susceptible to the same errors. For the present explanation, assume that the page size is 2048 bytes and the ECC requirements are 4 bits per 512 bytes. Let’s assume that the ECC algorithm generates 16 bytes of redundant data per 512 bytes. For 2048 bytes page, 64 bytes of redundant data will be generated. For example, in current TI embedded processors, the ECC data is generated for every 512 bytes. In this case, the spare bytes area will be filled with the ECC redundant data. As ECC requirements have gone up, the size of the spare regions provided by the NAND manufacturers have increased as well.

What will happen if insufficient ECC is used with a NAND device requiring more?

The manufacturers of the NAND devices specify the data retention and the write/erase endurance cycles under the assumption of the specified ECC requirements being met. When insufficient ECC is used, the device’s usable lifetime is likely to be severely reduced. In this scenario, if more errors are detected than can be corrected, data will be unrecoverable.

Geometry detection

Before raw NAND operations can begin, the first step for the firmware/middleware is to determine the NAND geometry and parameters. The following list is the minimum set of NAND parameters needed by a bootloader or other software layer to determine NAND geometry:

- 8-bit or 16-bit data width
- Page size
- Number of pages per block (block size)
- Number of address cycles (usually five in the current NANDs)

Methods used by NAND manufacturers

Raw NAND provides various ways to determine its geometry at run time. This section discusses various methods used by NAND manufacturers to provide this information.

- **4th byte ID:** All raw NANDs have a READ ID (0x90 at address 0x00) operation which returns 5 bytes of identifier code. In these 5 bytes, the first and second byte (if the starting byte number is 1 aka “one based”) are the manufacturer and device IDs, respectively. The fourth byte (one based) has information on the NAND parameters (discussed above), which can be used by the RBL. Figure 17 from the MT29F2G08AACWP datasheet (page no. 24) shows the READ ID operation. For this particular NAND, the four ID bytes are (from the datasheet, page no. 25):

Byte number returned from READ ID command	Value
1	0x2C
2	0xDA
3	Don't care
4	0x15

This 4th byte information can be used to determine raw NAND geometry. However, the interpretation of the 4th byte ID changes from raw NAND manufacturer to manufacturer and between generations of raw NANDs. Two interpretations are listed below:

1. A format used by Toshiba, Fujitsu, Renesas, Numonyx, STMicroelectronics, National and Hynix, with certain bits used to represent the page size, data bus size, and spare bytes size and number of pages per block.
2. **Samsung format:** A format particular to the latest Samsung NANDs (e.g., K9LBG08UXD), holding similar information to the first representation, but represented by different bit combinations representing different possible values.

Since the 4th ID byte format is not standardized in any way, its use for parameter detection is not reliable.

- **ONFI:** To overcome some of the issues discussed in the previous section, many NAND manufacturers (e.g., Hynix, Micron, STMicroelectronics, Spansion and Intel) have joined hands to simplify NAND flash integration and offer Open NAND Flash Interface (ONFI)-compliant NANDs. ONFI also offers a standard approach to read NAND parameters.

Other concerns

Another concern that must be addressed is that of the physical connection between the embedded processor and the raw NAND device. For instance, NAND devices can be acquired that operate at either 3.3V or 1.8V. One should be sure to purchase NANDs with compatible voltage levels. It should be pointed out that 1.8V NAND devices are often specified with worse performance than 3.3V equivalent parts.

Another aspect that needs to be considered is whether asynchronous or synchronous NANDs will be used. The synchronous interface was something introduced with the ONFI 2.0 specification. Historically, NAND interfaces were asynchronous. However, to reach higher performance levels for data movement, clock synchronized data movement with DDR signaling was provided as an interface option. This type of interfacing may be common in SSD drives but is not common in the typical embedded processor or SoC.

Challenges to address

Hardware design challenges (memory interfaces)

Advanced ECC operations, like Reed Solomon or BCH, are computationally very expensive. Many solution offers hardware (HW) support for ECC. However, these fixed solutions lag behind the growing ECC requirement.

For example, MLC NANDs may now require more than 16 bits ECC per 512 bytes. If the hardware (HW) support was designed a few years ago, it might not support 16 bit. In that case, the HW ECC support would become useless, and either the NAND could not be used or the ECC computation would need to be done in the software (SW) (which results in taking CPU cycles from other important tasks and assigning it to ECC calculation).

Firmware and ROM bootloaders

The dynamic raw NAND market (raw NANDs have relatively short life cycles of about two years) and the initial lack of standardization has resulted in heterogeneous interfaces from different NAND manufacturers. Not only do these approaches vary between manufacturers, but, at times, they also differ between different generations of NANDs from within the same manufacturer! This offers significant challenges for firmware and middleware that might not be updated very often (if at all).

The main concern when designing ROM bootloaders for raw NAND booting is whether or not future NAND devices will work. The ONFI standard helps alleviate this somewhat, since it provides a way to guarantee device identification commands that should not have to change in the future. The use of the 4th ID byte is not a reliable mechanism. Another major concern related to the hardware design issue is what level of ECC is sufficient. Since the NAND parts that will be connected to the system cannot be known a priori, the safest solution is to leverage the maximum ECC possible with your memory interface or controller. Using more ECC than required for booting simply improves robustness, with the possible downsides being increased boot time (though this may be implementation dependent) and more complex factory programming procedures.

Since NAND manufacturers do not guarantee that all blocks of the memory are good (nor will all blocks remain good over the device's lifetime), another issue that must be addressed is how bad blocks can be handled if encountered during booting. To clarify, a bad block is one that contains any unrecoverable errors (that is, any errors not corrected by ECC). Some strategies include placing multiple copies of the boot image and letting the boot loader locate and load the first good one or having the boot loader respect a bad block table stored somewhere else in the NAND. Another useful strategy might be to have the system run-time software periodically check and correct any issues with the boot block (perhaps every 100 boot cycles). Regardless of what approaches are taken, a ROM bootloader needs to be able to deal with recovery from bad blocks and firmware upgrades gone bad (flash corrupted), so the above issues need to be considered carefully in designed procedures for booting and flashing.

Middleware/OS software issues

The middleware, or run-time software, suffers from similar issues to those faced by the ROM boot loader. One well-known middleware example is the memory technology device (MTD) layer of the Linux® OS kernel. Although it might be easier to adapt the middleware to handle newer devices, newer detection schemes and newer command sets offered by more recent device, it does present an overhead every time a change has

to be propagated through different support structure from middleware teams to customers. For example, the MTD layer of Linux had issues when device sizes reached 4GB and above since the size had originally been defined as a 32-bit value. In another case, there was no support for NAND device with page sizes larger than 2KB. Modern NAND devices have 4KB or 8KB page sizes. Fixing these issues is not necessarily trivial.

In addition, the run-time middleware, which will likely support some kind of read/write filesystem on top of the NAND flash, **must** deal with activities such as wear leveling and bad block management. Wear leveling is software mechanism to spread the write/erase cycles around the chip so that all blocks wear evenly. Failure to do so will result in oft-used blocks failing very early. This is more important now than ever before, as the cells of the MLC devices have much reduced endurance rating (3,000–5,000 compared to 10,000 to 100,000 for SLC NANDs). The middleware must also track which blocks are bad, either from the factory or having failed over time, and make sure they are not used for any further reads and writes. The more stringent requirements of recent and future NAND devices may require even more complicated schemes to be enacted for the above two activities.

Supporting different custom ECC hardware is another challenge as one ports the middleware from one generation of processor to the next, and changes and improvements to the ECC capabilities must be handled in new drivers. Additionally, there is no good solution if the ECC HW cannot meet the ECC requirement of the NAND, as software ECC has proven to be too slow and cumbersome for most embedded processors.

***Texas Instrument's (TI's)
view: Solutions and
recommendations***

In this author's opinion, the main issue with using raw NAND devices in the embedded processor space is the skyrocketing ECC requirements. In five years, ECC requirements have gone from 1 bit/4 bit to pushing past 24 bits. Memory interface hardware designed six years ago is very likely incompatible with any new chips available on the market today. Though the issue of device identification and parameterization has been a problem, it is not considered as critical as the fundamental incompatible resulting from insufficient ECC hardware in the memory controller/interface of embedded processors. Given the lifetime of products based on such devices (10–15 years), the lack of NAND supply that fits the original design requirements could force major rework in both hardware and software part of the way through the product life cycle.

Fortunately, the memory manufacturers have realized the issues with rapidly increasing ECC requirements and have taken steps to address it. The solution in two words is Managed NAND.

Managed NANDs are devices that perform some or all of the three NAND management tasks (e.g., ECC, wear leveling and bad block management) on the memory device instead of in a host controller. Perhaps the most compelling of these is the eMMC product. eMMC is a Joint Electron Devices Engineering Council (JEDEC) standard that combines electrical and physical chip specifications with the interface commands and protocols of the MMC 4.3 into a single entity^{viii}. The eMMC architecture is shown below. In such a design, the details of ECC, wear leveling and bad-block management are encapsulated into the controller core and are no longer a concern for the system developer. In this case, the chip-to-chip interface becomes a standard MMC interface.

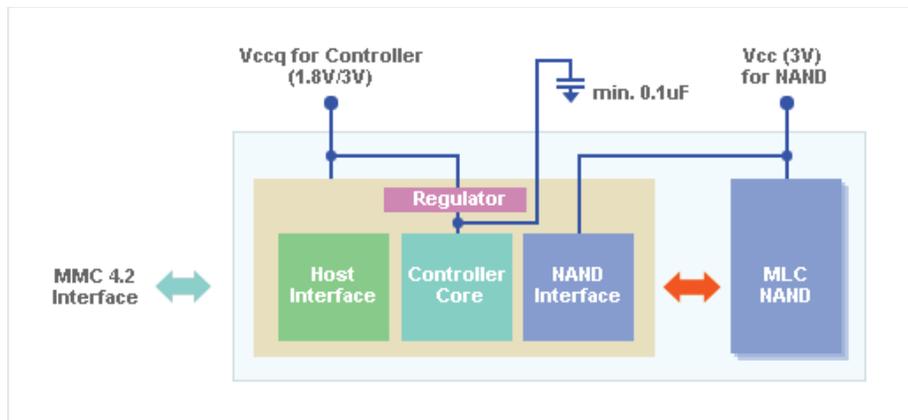


Figure 3. eMMC block diagram (courtesy Samsung)

There are also what could be termed partially-managed NAND devices that maintain the NAND interface but move the most worrying aspect of dealing with raw NANDs – the ECC – into the memory device. The solution, which Micron has branded as ClearNAND™, may be attractive to those looking to replace the NANDs in current designs with minimal changes to the system SW or HWⁱⁱⁱ. Toshiba, another major NAND manufacturer, has recently released a nearly identical solution, dubbed SmartNAND™. It seems certain that other NAND vendors will soon follow the same path.^x

There are additional costs to the managed NAND solutions, so a design may pay for this convenience. But having seen the problems of a rapidly maturing NAND market as both a silicon producer and a silicon consumer, it is clear that the some form of managed NAND is the only sensible choice for future design and development. Embedded processor vendors, such as Texas Instruments, will continue to support the existing ECC hardware in their memory interfaces, but there is little reason for them to spend design time and resources trying to keep pace with the skyrocketing ECC requirements. Jim Cooke of Micron recently wrote,

Coupling Error Management With NAND

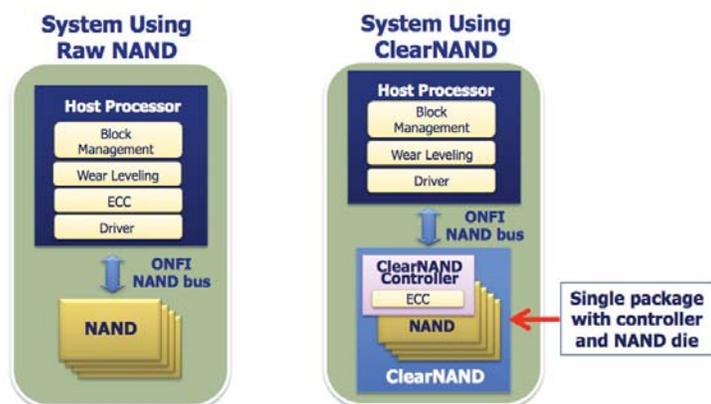


Figure 4. Standard raw NAND versus ClearNAND (courtesy Micron)

“In the near future, interfacing to raw NAND Flash will be an option for only a minority of the market, the very large customers who are almost as familiar with NAND flash as the manufacturers are.” This author agrees completely with his assessment. At this point in the evolution of NAND, it is best to leave the implementation of advanced ECC solutions to the memory vendors, as they push their devices to the physical limits of CMOS technology.

References

- ⁱ <http://www.eetimes.com/electronics-news/4123962/NAND-flash-to-overtake-NOR-in-2005-says-analyst>
- ⁱⁱ <http://www.forbes.com/global/2002/0624/030.html>
- ⁱⁱⁱ http://download.micron.com/pdf/datasheets/flash/nand/2_4_8gb_nand_m49a.pdf
- ^{iv} <http://download.micron.com/pdf/technotes/nand/tn2908.pdf>
- ^v http://en.wikipedia.org/wiki/Reed%E2%80%93Solomon_error_correction
- ^{vi} http://en.wikipedia.org/wiki/BCH_code
- ^{vii} <http://www.jedec.org/standards-documents/focus/flash/e-mmc>
- ^{viii} http://www.micron.com/products/nand_flash/cleannand.html
- ^{ix} http://www.toshiba.com/taec/news/press_releases/2011/memy_11_608.jsp
- ^x <http://www.eetimes.com/design/memory-design/4213098/NAND-201--the-continued-evolution-of-NAND-Flash>

Important Notice: The products and services of Texas Instruments Incorporated and its subsidiaries described herein are sold subject to TI's standard terms and conditions of sale. Customers are advised to obtain the most current and complete information about TI products and services before placing orders. TI assumes no liability for applications assistance, customer's applications or product designs, software performance, or infringement of patents. The publication of information regarding any other company's products or services does not constitute TI's approval, warranty or endorsement thereof.

All trademarks are the property of their respective owners.

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 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. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

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

Audio	www.ti.com/audio
Amplifiers	amplifier.ti.com
Data Converters	dataconverter.ti.com
DLP® Products	www.dlp.com
DSP	dsp.ti.com
Clocks and Timers	www.ti.com/clocks
Interface	interface.ti.com
Logic	logic.ti.com
Power Mgmt	power.ti.com
Microcontrollers	microcontroller.ti.com
RFID	www.ti-rfid.com
RF/IF and ZigBee® Solutions	www.ti.com/lprf

Applications

Communications and Telecom	www.ti.com/communications
Computers and Peripherals	www.ti.com/computers
Consumer Electronics	www.ti.com/consumer-apps
Energy and Lighting	www.ti.com/energy
Industrial	www.ti.com/industrial
Medical	www.ti.com/medical
Security	www.ti.com/security
Space, Avionics and Defense	www.ti.com/space-avionics-defense
Transportation and Automotive	www.ti.com/automotive
Video and Imaging	www.ti.com/video
Wireless	www.ti.com/wireless-apps

TI E2E Community Home Page

e2e.ti.com

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2011, Texas Instruments Incorporated