intel®

AP-636

APPLICATION NOTE

Preventing BIOS Failures Using Intel's Boot Block Flash Memory

December 1996

Order Number: 292192-001

CONTENTS

REVISION HISTORY

Number	Description
-001	Original Version

1.0 INTRODUCTION

One of the major uses of flash memory technology is to provide basic input/output services (BIOS) for personal computers (PCs). This critical system BIOS is entrusted to flash because of its reliability, ease-of-use and in-system updateability. However, the PC has become more than a simple appliance for word processing and spreadsheets. With internet access, electronic mail and plug and play capabilities, the PC has evolved into a powerful, labor-saving tool. Designing to this new criteria is challenging and sometimes unpredictable. Flash memory can alleviate some of these design woes, but improper flash practices can create additional difficulties. This application note explains system considerations when designing with Intel flash memory. Section 2.1 discusses the debug tools necessary to investigate system problems related to flash. Subsequent sections detail how to identify and correct possible flash failures.

2.0 BIOS FAILURES

The BIOS of a standard computer controls system configuration. This level of control extends from hardware devices to interrupt handling routines. Since this code is so critical to system operation, its safety and protection is absolutely essential. Today's PCs use Intel's Boot Block flash memory for their BIOS. However, if system design is not supportive of flash memory requirements, failures with BIOS are possible. These errors usually occur during system testing or when the PC is turned on. These problems may also occur as part of the overall system test flow. The next few sections identify possible BIOS problems and offer one possible solution for each problem.

2.1 Debug Tools

There are many tools to use for debugging BIOS or flash memory problems. Depending on the depth of analysis needed or wanted, the tools can be as simple as a software update utility or a complicated as a digital storage oscilloscope.

1. Update or Recovery Software–this software is normally available from the motherboard or BIOS manufacturer. Its primary use is to update the flash device with the latest BIOS release. Some types of the software can be used to recover a flash device which has been corrupted or altered in the field. This is useful when the motherboard does not operate correctly. Other types of software can actually capture a BIOS image prior to programming a new revision, which allows the user to save the old revision or corrupted image for later analysis.

2. Post-Aid Card–this is an ISA-compatible add-in card which can give feedback to the user about a motherboard failure. This card can help indicate where in the boot sequence a motherboard is locking up during the boot process. Error codes are captured through Port 80. Some motherboards may not support this type of board.

3. Digital storage or Analog Oscilloscope–this tool is useful in acquiring signals for use in analyzing systems for root cause. Digital storage and analog scopes in the 100 MHz bandwidth range can be used to capture repeatable events in the 10 ns range such as power supply ripple and byte programming setup on the control lines (i.e., WE#, CE#, OE#, RP#).

   Digital storage scopes are recommended for capturing single-shot events such as voltage spikes/dropouts, signal transition noise, and setup/hold events. For single-shot event capture, 100 MHz bandwidth is still adequate; however, sample rate is key and would need to be at least 250 megasamples per second in order to capture a 10 ns event accurately. In choosing a scope, the bandwidth is the maximum frequency that can be captured in any mode and the samples per second limit the maximum frequency that can be captured in single-shot or real-time acquisition (real-time bandwidth = sample rate/2.5).

   If faster rates of signal capture are needed then higher cost, higher bandwidth, higher sample rate scopes are needed. When taking measurements it is recommended to use 10x attenuating probes or higher and make the distance between the probe tip to actual desired measurement point (at the pin under test) as short as possible. This measurement tip will minimize circuit loading and possible antenna effects, allowing outside disturbance during your measurement.

4. Read/Write Programmer –This could range from a low-end PC-based programmer to a high-end automated tester. These tools would allow you to read, erase or program a flash device for the purpose of failure analysis. These particular tools would be valuable in detecting device failure signatures (discussed in Section 2.2).

5. Test Socket–a working model of the motherboard being used with the flash location installed with a prototype socket. Used for failure confirmation and other correlation work.

2.2 System Signatures

There are several system failures that may be indicative of BIOS corruption or other BIOS-related problems.

Signature

1. No video, no beeps from internal speaker, and no POST code activity on Port 80
2. NVRAM checksum or data invalid error
3. System locks-up during boot/power-on process

Possible Cause

1. Corrupted BIOS code in main flash
2. Corrupted User/Configuration Data in parameter block of flash device (e.g., Extended System Configuration Data or ESCD block)
3. Corrupted CMOS configuration information in battery backed-up memory or other component issues not related to flash

Containment Action

In general, flash BIOS corruption is caused by incompatible add-in cards, aborted flash updates caused by power supplies or power fluctuations, or improper BIOS images. These failures can be caused by inconsistent process issues and are not normally device or motherboard related problems. Because of this fact, the failures can be recovered in-system without being a reliability concern.

Depending on the motherboard design two actions can be completed to recover the BIOS and save pulling a component or sending a motherboard back to a supplier.

1. Reset or clear the CMOS memory: this action normally clears the CMOS memory and the ESCD (or Plug and Play data block) of the flash device. This is normally done by a jumper on the motherboard and a software utility supplied by the manufacturer of the motherboard. Figure 2 illustrates typical CMOS / flash hardware setup.

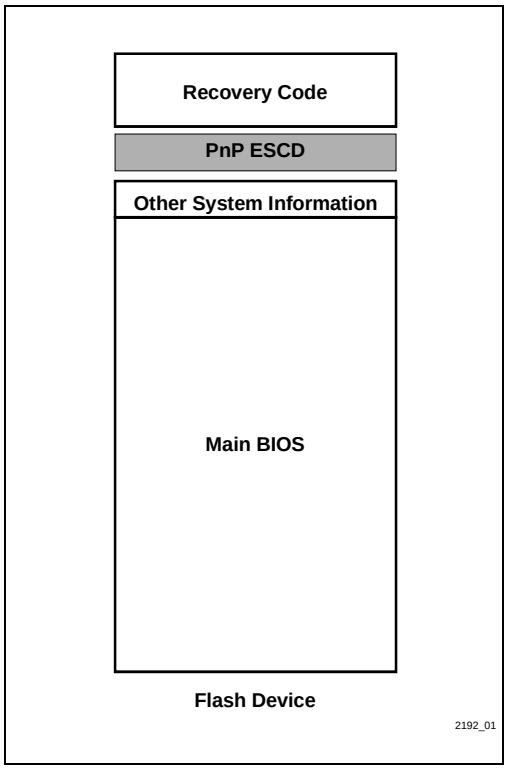


Figure 1. Example of ESCD Storage with Flash

2. Update or recover the main BIOS portion of the flash device through a software update utility: if the board fully boots, this update can occur by a command input at the screen prompt; however if the board is dead or will not boot, some boards can be recovered by moving the flash protection jumper and using a recovery code stored in the boot block of the flash device. Another alternative is to use a special ISA or PCI card intended for BIOS debug. All software utilities and add-in cards should be obtained by contacting your motherboard supplier or BIOS vendor.

Preventive Action

If the corruption or flash failure is not recoverable or fallout becomes expensive to contain, the root cause of the problem needs to be determined. The sections following discuss device fail signatures and how to prevent them in the future.

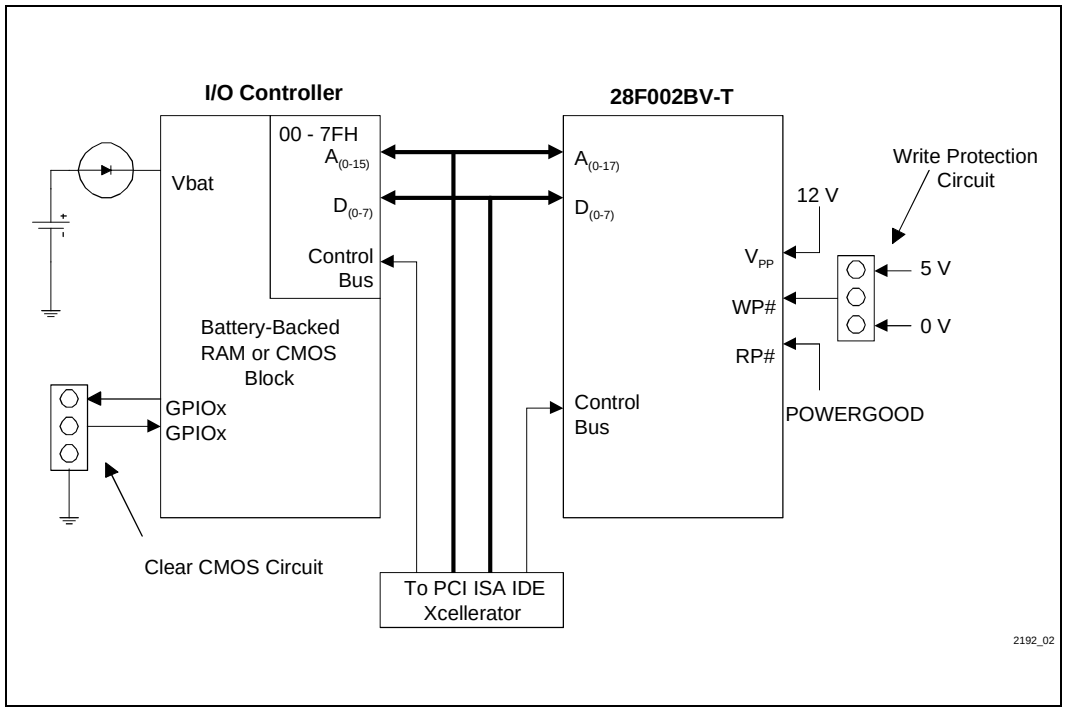


Figure 2. Example CMOS/Flash Hardware Setup

2.3 "Bad BIOS"

"Bad BIOS" refers to a complete BIOS image programmed correctly into the flash device, but the BIOS is actually not the correct or compatible BIOS for the particular motherboard.

Signature

1. System has no video, no beeps from speakers, and/or no POST code activity on Port 80
2. System locks-up during boot/power on process
3. Device reads a byte/word results in partial data and/or "garbage"
4. Device reads a byte/word results in seemingly random data

Possible Causes

1. Improper BIOS image used while pre-programming flash devices on a programmer or during initial system in-circuit testing (e.g., bed-of-nails)
2. Improper BIOS revision used while updating a particular motherboard
3. An incompatible BIOS programmed into the flash device (e.g., after market products)

Preventive Action

A "Bad BIOS" being programmed into a flash device is almost always a process problem. In the factory, process control is key; meaning that controls must be in place for newly released BIOS images, update utilities, and the type of media used to transfer these images.

Floppy disks have historically been very difficult to control as a new release media, and, in a factory environment it is recommended that server download be used when possible. Server download allows the released image to reside in only one place and programming/test stations can all download from a common area. Smaller factories could also consider a hard drive media for new releases.

After-market BIOS images can be bought or acquired from many companies. Some of these BIOS images may be incompatible with a particular motherboard. This type of problem in most cases shows up as a field failure from the end-user. Identification of the problem can sometimes be difficult due to the board not being able to boot. Since the device could have similar fail signatures explained in this document, it may be possible to misdiagnose the root cause of the problem. If the motherboard boots to a point where the monitor receives data, the BIOS revision is usually the first screen displayed item and it can be determined what type and revision is programmed into the flash device.

2.4 Program Errors

Signature

1. Reading a byte/word results in partial data and/or "garbage"

2. Reading a byte/word results in seemingly random data

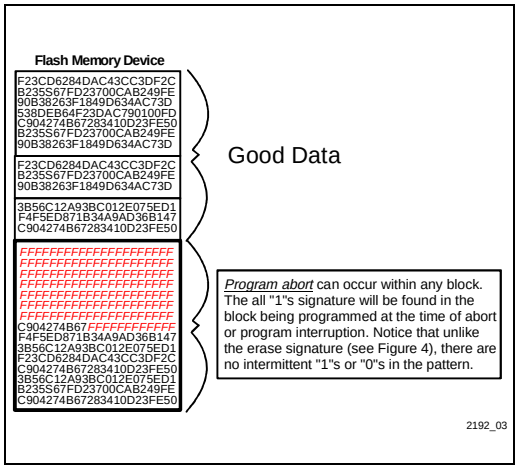


Figure 3. Example Program Abort Signature

Possible Cause

1. V_{CC} or V_{PP} outside valid range after program sequence begins

2. V_{CC} power interruption during a program sequence

3. V_{PP} power interruption during a program sequence

4. Driving RP# to V_{IL} during a program sequence

Preventive Action

As with any memory device, the flash device must first be programmed with data in order for it to be functional within a system. No location within a flash device can be programmed unless it is blank or first erased. Once the location to be programmed is verified blank, programming commences a byte/word at a time. All program errors compromise the data integrity of the location being programmed; this location must be erased and re-programmed with valid data.

Programming is the process by which charge is placed on the floating gate of the flash (changing array bits from "1" to "0"). This fact is very useful in determining when a program failure occurs. Suppose, for example, that prior to the program sequence V_{CC} and V_{PP} are at valid levels. However, during the program sequence, V_{CC} and V_{PP} fall outside the valid range. This "droop" in power will not provide ample current for proper programming. The result is marginal or partially programmed bits. When these bits are read following the program sequence, they will generate seemingly random data because in some instances they may be sensed as a "1" but in other instances they may be sensed as a "0." Voltage, temperature and timing are all factors that affect the outcome of the read.

Power interruption, whether V_{CC} or V_{PP}, during a program sequence immediately aborts the operation. If V_{PP} is interrupted, the V_{PP} error flag (SR.3) will be set. V_{CC} provides power to the entire flash device, including internal circuits, sense amplifiers, etc. Although no error flag is set for V_{CC} interruption, this action will definitely prevent proper programming of the flash.

All Boot Block flash memory devices provide a deep power-down pin for placing the flash device in an ultra low-power mode. This mode is initiated by driving the RP# input to V_{IL}. This action places the outputs in high-Z, resets the internal write state machine (WSM—which controls the automated program and erase sequences), and shuts down all but the most critical internal circuits.

If RP# is driven low during a program sequence, the operation is immediately aborted since the WSM is reset. The memory location being programmed is corrupted and must be erased and re-programmed with valid data upon return from power-down.

2.5 Erase Errors

Signature

1. Reading a byte/word results in partial data, “garbage” and/or “0”s
2. Reading a byte/word results in seemingly random data (especially at low V_{CC} or high temperature)
3. Reading a byte/word results in unchanged data (no erase occurred)

Possible Cause

1. V_{CC} or V_{PP} outside valid range after erase sequence begins
2. V_{CC} power interruption during an erase sequence
3. V_{PP} power interruption during an erase sequence
4. Driving RP# to V_{IL} during an erase sequence
5. Incorrect erase command sequence

Preventive Action

When an erase command is issued to the flash memory, two distinct actions take place: first all bytes within the block to be erased are preprogrammed to “0” (a byte at a time) and then the entire block is erased to “1” simultaneously. The preprogramming prevents over- or under-erase due to too much or not enough charge being placed on the floating gate (this is a “manual” process for bulk erase flash devices). Preconditioning and erase are both handled by the WSM.

If the erase sequence is interrupted during preprogramming, partial data is still contained within the array although some locations have been pre-programmed to “0.” Reading the device at this point will result in erroneous data from all compromised locations. All locations within the block should be assumed corrupted since there is no way of knowing where the preprogramming was interrupted.

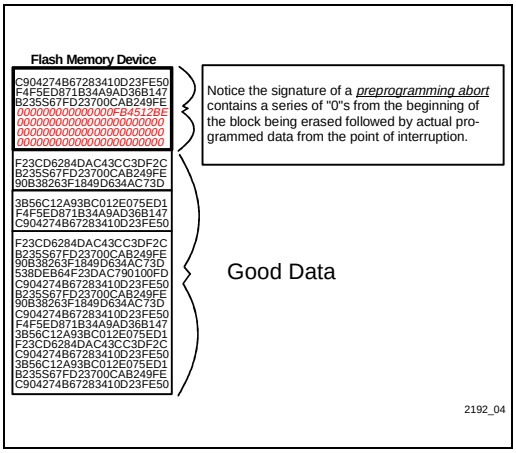


Figure 4. Example Erase Abort Signature (during pre-programming)

If, however, the erase portion of the WSM machine algorithm is interrupted (either through V_{CC} or V_{PP} interruption or RP# being driven to V_{IL}), the array will show either all “0”s or partial “1”s and “0”s when read. This is an undesired state, as proper erase has not occurred. This problem is exacerbated by low V_{CC} or high temperature reads. Attempting to program the flash while in this state may result in program failures.

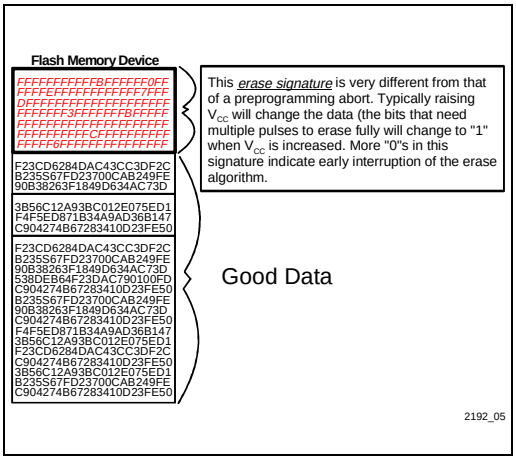


Figure 5. Example Erase Abort Signature (during erase algorithm)

As with partial programming, drooping either V_{CC} or V_{PP} during the erase operation will result in marginally-erased bits. When these bits are later read, they may be sensed as either a "0" or a "1." This depends on several factors, including stability of address transitions, data lines, V_{CC} level, temperature and overall system noise. Proper V_{CC} and V_{PP} levels should be maintained at all times.

Finally, erase consists of a two-command sequence that must be followed. If the Erase Setup command is issued but is not followed by the Erase Confirm command, the erase operation will not take place. No data will be altered within the array and status bits 5 and 4 will be set to indicate an erase command sequence error. This can be corrected by clearing the status register and re-issuing the proper erase command sequence.

2.6 Spurious Writes

Signature

1. Reading a byte/word results in several erroneous bits
2. Random memory locations contain spurious data
3. Multiple reads to a flash location result in different data (especially at low V_{CC} or high temperature)

Possible Cause

1. V_{CC} or V_{PP} noise
2. CE# or WE# glitches
3. Unintentional program commands

Preventive Action

Spurious writes with Boot Block flash devices is not very common because of the many write protection schemes that they possess. However, poor system design or a noisy system can create an environment that is prone to spurious writes.

A spurious write is not necessarily an invalid program of data to the array. Since the flash requires a two-command sequence to initiate most operations, a spurious write could place the flash into an undesirable state. For instance, assume the data bus was previously driven to 10H and currently CE# and WE# are at V_{IL}. A CE# glitch to just above V_{IL} and back to normal V_{IL} levels for a duration of, say, 10 ns could result in the flash latching 10H into its command user interface (CUI). 10H is the alternate Program Setup command. On the next WE# H-L transition, the flash will begin programming data to the address on its bus. If the user were to issue a read to the flash while it was in this state, the flash would respond with status register information rather than array data. In this scenario, the flash array could only have been written spuriously if V_{PP} was at valid program levels. To safeguard against such spurious writes, keep V_{PP} below V_{PPLK} when not programming or erasing.

The best protection against spurious writes is user control of WP# (if applicable) and V_{PP} levels when memory alteration is not desired. Spurious writes can also be prevented through clean address transitions (based on the specifications outlined in the product datasheet) and smooth control input transitions. The greater the noise immunity of a system, the lower the risk to spurious writes.

Power supply up/down ramp sequences can also introduce noise into a system. No flash activity should be initiated until these transitions has settled to steady state or until V_{CC} and V_{PP} inputs are stable. By holding the flash device in RESET (with RP# connected to system POWERGOOD) during power-up/down transitions, invalid bus conditions can be masked to provide another level of protection.

Transient current magnitudes, which are dependent on output capacitive and inductive loading, contribute heavily to system noise problems. All flash devices should be designed with a 0.1 µF capacitor between each V_{CC} and ground connection. Another 0.1 µF capacitor between V_{PP} and ground is also recommended. These capacitors will smooth out transient current peaks and filter other high frequency noise. The capacitors should be placed as close to the flash as possible.

In addition, using wider traces for power and ground will improve current flow/dissipation and reduce potential noise problems. This will make the system environment less prone to spurious writes and unintended data corruption. Table 1 gives a checklist of tests that can help deduce suspected spurious write problems.

Table 1. Spurious Writes Checklist

Test	Test Conditions
Is RP# pin driven as a RESET#?	$V_{CC} > V_{LKO}$
Is RP# low during power ramps?	$V_{CC} > V_{LKO}$
$V_{PP} < V_{PPLK}$ during resets and power ramps (for spurious write protection)?	$V_{CC} > V_{LKO}$, RP# = V_{IH}
Spurious Program or Erase command during normal operation	$V_{CC} > V_{LKO}$, WE# = CE# = V_{IL}, RP# = V_{IH}
If RP# not used, test for spurious writes during power-up/down and resets	WE# = CE# = V_{IL}, RP# = OE# = V_{IH}, $V_{CC} > V_{LKO}$

2.7 Electrical Overstress (EOS)

Signature

1. Unable to read or program the entire flash or portions of the flash
2. The flash draws excessive current than stated by the datasheet
3. System V_{CC} is seemingly pulled above its maximum levels by the flash
4. Operations to specific locations are seemingly ignored

Possible Cause

1. Device insertion or removal while power is active
2. Application of excessive voltages to device pins
3. Low quality power supplies that produce excessive noise when switching
4. Bus contention or output shorts that cause high current flow
5. Incorrect device orientation

Preventive Action

Electrical overstress (EOS) is damage to a component's circuitry caused by thermal overstress (either AC or DC). Excessive electric fields and voltages stress device oxides to their breakdown point, which leads to functional failures and/or excess current draw and heat generation. The amount of damage caused depends on the magnitude and duration of the transient electrical pulse.

Flash components provide maximum device characteristics which must be strictly followed. Stressing the device beyond its maximum limits may permanently damage the device. Providing 15V to the V_{PP} input, for instance, can cause damage to the V_{PP} circuitry and prevent further program or erase sequences to the flash. The extent of the damage depends on how long the voltage was applied as well as the magnitude of the excess voltage. A similar voltage applied to V_{CC} will cause extensive damage to the device since its tolerance to high voltage is much lower than that of V_{PP}. EOS can also lead to melted metalization or open bond wires, both of which will cause device failure.

Most EOS problems occur during testing, device verification or insertion/removal exercises. Employing proper procedures during these stages will improve device susceptibility to EOS.

1. Use quality power supplies with over voltage protection, proper heat dissipation
2. Check test programs for hot switching
3. Adhere to maximum spec ratings
4. Adopt good grounding practices
5. Check system noise levels

Table 2 provides some areas to check if an EOS problem is suspected.

Table 2. Electrical Overstress Checklist

Test	Specification	Test Conditions
V_{CC}	4.5V– 5.5V (7V Absolute Max.)	V_{CC} < 5.5V **or** 5.5V < V_{CC} < 7V for ≤ 20 ns a. Power on b. Power off
V_{PPH}	11.4V– 12.6V (14V Absolute Max.) **or** 4.5V–5.5V	V_{PP} < 12.6V **or** 12.6V < V_{PP} < 14.0V for ≤ 20 ns a. Turn on V_{PP} b. Turn off V_{PP} c. Power on d. Power off
V_{PPL}	0V	–0.5V < V_{PP} **or** -0.5V < V_{CC} < 0V for ≤ 20 ns a. Turn on V_{PP} b. Turn off V_{PP} c. Power on d. Power off
V_{CC} Decoupling	0.1 µF	Capacitor Adjacent to Flash
V_{PP} Decoupling	0.1 µF	Capacitor Adjacent to Flash
Bus Contention	No Bus Contention	Decoder / Driver Design
Hot Switching	No Hot Switching	Power Connection Method
V_{HH}	11.4V– 13.2V (10% V_{PP}) (13.5V Absolute Max.)	V_{HH} < 13.0V **or** 13.0V < V_{HH} < 13.5V for ≤ 20 ns
I/O Pins	–0.5 < V_{IN} < (V_{CC} + 0.5)	–0.5 < V_{IN} < (V_{CC} + 0.5) **or** (V_{CC} + 0.5) < V_{IN} < (V_{CC} + 2.0) for ≤ 20 ns **or** –2.0V < V_{IN} < 0V for ≤ 20 ns

2.8 Electrostatic Discharge (ESD)

Signature

1. Unable to read or program the entire flash or portions of the flash
2. The flash draws excessive current than stated by the datasheet
3. System V_{CC} is seemingly pulled above its maximum levels by the flash
4. Operations to specific locations are seemingly ignored

Possible Cause

1. Devices manually handled in an improperly protected work space
2. Devices transported in non-protected media from place to place
3. Improperly protected/installed automated handling equipment or transport carts
4. Improperly protected/installed equipment on a surface mount/through-hole assembly line (e.g., brushes, rubber mats, etc.)
5. Improper environmental controls (e.g., humidity)

Preventive Action

Electrical Static Discharge (ESD) is a phenomenon which transfers an electrical charge built up on one material to another. Typical ESD voltages for several events is given in Table 2 below. In the example of a flash device or any other semiconductor electrical device, this type of charge, if dissipated through the device, can cause permanent damage.

Table 3. Common ESD Events and Voltages

Event	Volts at Relative Humidity		
	10%	40%	55%
Walking across carpet	35,000	15,000	7,500
Walking across vinyl floor	12,000	5,000	3,000
Motions of a bench worker	6,000	800	400
Removing DIP from plastic tube	2,000	700	400
Removing DIP from vinyl tray	11,500	4,000	2,000
Removing DIP from Styrofoam	14,500	5,000	3,500
Removing bubble wrap from PWB	26,000	20,000	7,000
Packing PWB in foam-lined box	21,000	11,000	5,500

Charges are generated when two materials (insulative materials) are rubbed against each other and separated. For example, when you walk across the floor, the soles of your shoes sliding against the floor will generate a charge. Other common generators of charge are the pulling apart of tape, dry air moving across the surface of a material, wheels on carts or chairs and untreated plastic bags.

Once the charge is generated, a charge can be moved/transferred from one object to another object. Damage to a semiconductor device is caused due to large amounts of energy transferring through the device in a short duration of time. Two ways of minimizing ESD damage is to control the build up of charge and the rate of the charge transfer.

1. Use personal grounding through wrist and heel straps
2. Wear ESD protective clothing or smocks
3. Use dissipative mats for table tops, storage racks, and work areas
4. Install ESD chairs, dissipative flooring (e.g., ESD wax or floor mats)
5. Use air ionizers and other climate control devices
6. Use anti-static or dissipative packaging material and ESD shielding totes or containers for movement within the factory

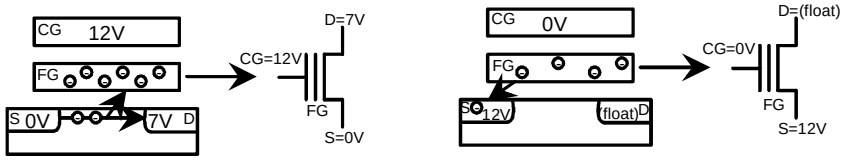
2.9 Overerase (Bulk Devices Only)

Signature

1. Unable to read or program the entire flash or portions of the flash
2. Device is notably slow to program
3. The flash works properly at first then fails to function after some time

Possible Cause

1. Improper adherence to erase algorithm
2. Attempting to program before V_{PP} is at 12V
3. All bytes not preprogrammed prior to erase (this may occur if a part of the flash memory map is masked due to another memory, e.g., an on-board ROM)
4. Failure to preprogram all bits to 00H
5. Excessive number of erase pulses
6. Failure to wait appropriate time before attempting to verify the erase
7. Failure to latch erase verify address with the erase verify command (or changing the address on the subsequent read cycle)



PROGRAM
(add charge to floating gate)

ERASE
(remove charge to floating gate)

2192_06

NOTE:
The migration of charge to or from the floating gate changes the cell threshold, which determines whether a transistor is"on" or "off."

Figure 6. Cell Level Diagram of Program and Erase Operation

Preventive Action

The flash erase process removes charge from the floating gate by lowering cell threshold voltage (Vt). Bulk devices do not have an automated program or erase algorithm. The user must provide all necessary voltage and control inputs and follow all timings to insure proper erasure.

The Quick Erase Algorithm uses an erase verify to signal when ample charge has been removed from the floating gate. If additional erase pulses are given following proper erase verify, more charge is removed from the floating gate. After prolonged exposure to such an algorithm, the cell will always read as erased, even after being programmed. In fact, not only will that cell be "stuck at 1" but all cells on that column will eventually be overerased and read as "1" always. A normal cell only turns on if its gate is selected, but an overerased cell is always on even when not selected; this corrupts reads of that cell and all cells on that column.

Overerase can be prevented by following the syntax of the Quick Erase Algorithm exactly. Deviation from the algorithm as stated is outside the datasheet specification and device operation is not guaranteed. Also, switching from a bulk device to any second generation Intel flash device (all second generation flash devices feature automated program and erase algorithms) will eliminate overerase problems altogether.

3.0 CONCLUSION

Board designs (for PCs and other applications as well) have grown in complexity over the years, and engineers will continue to improve and make changes on a daily basis. Flash memory has enabled these changes to be incorporated into designs without significant impact, if any, to schedules. It is this benefit, in addition to many others, that makes flash memory an ideal solution for many applications.

However, an application cannot yield these benefits if an unsuitable flash design is used. This application note has identified several areas that may require additional attention. More importantly, preventive actions are provided to insure these problematic areas do not occur or significantly impact a design. Careful up-front planning for flash usage can assure continued flash memory benefits out in time.