Error Correction Control and Parity BIOS Implementation Example

White Paper

Revision 1.2

Revision History

Revision	Date	By	Description
0.1	03/14/97	J. Carter	Initial Draft taken from ECC white paper and impromptu knowledge. - JCC
0.2	03/17/97	J. Carter	Added basic schematic drawings and summarized the overview section. Any detail previously in the overview moved to BIOS implementation section. - JCC
1.0	03/19/97	J. Carter	Added subsection titles and changed copyright date to 1997. - JCC
1.1	04/07/97	N Yoke	Changed title from ECC/Parity BIOS Implementation Specification. Changed headers, footers, revision history, and disclaimer to comply with format for other collateral documents. Used "DMI BIOS Support: Interface Requirements Revision 2.1" as guideline. Minor modifications to content of document. - NJY/JCC
1.2	04/18/97	N Yoke	Polished up and replaced text with Flow Chart. Changed Title added disclaimer. - NJY

CONTENTS

1.1 Introduction

This information is intended to allow one to provide the System BIOS interfaces and programming for Intel's ECC/Parity chipset implementation. This document can also be used as a guide, along with the other referenced ECC/Parity documents, to aid ECC/parity instrumentation efforts.

1.2 Definition of Terms

DMI	Desktop Management Interface. See "Desktop Management BIOS Specification" Version 2.0, October 16, 1995 for more details.
ECC	Error Correction Control—Method of handling system memory errors using multiple bits per unit of memory.
NVRAM	Non-Volatile RAM—Entity that describes a type of storage used to retain programmed information after power is removed from the CPU and main memory.
PARITY	Method of handling system memory errors using a single bit per unit of memory.
SCRUBBING	A process of reading a memory location and then writing the value and error bits back to DRAM. If there was a noise-related error, it will be corrected.
SYSTEM EVENT LOG	Log that is used to store critical system status in an area of NV RAM.

1.3 System BIOS Responsibility

This section provides a general overview of the System BIOS responsibilities for handling ECC and Parity implementations. There are four general items defined as:

- **Initialize Hardware**—During POST (Power On Self Test) the BIOS needs to detect the presence of ECC and Parity memory installed in the system. This information also needs to be stored in some form of NV RAM. The BIOS should also initialize ALL DRAM memory used in the system once ECC is enabled in the chipset with the error generation turned off.
- **Report Errors**—The BIOS needs to report errors back to the user. In situations where DMI BIOS Event Logging support is present, the event log must be updated with the error. Refer to the "DMI BIOS Support: Interface Requirements Revision 2.1" document for details on this interface. In situations where DMI BIOS Event Logging support is not present, the BIOS should implement a clear NMI handler as to the effect and location of the error.
- **Setup/Configuration Interface**—The BIOS should provide a setup interface to allow the user to enable/disable ECC or parity checking in the system. In situations where DMI BIOS Event Logging support is present, setup should also provide a switch for enabling/disabling the logging of these type of events, as well as a view log option to view all events logged.
- **Correction of Errors**—Single-bit ECC errors are correctable. This is due to the fact that the specific data bit that has the error can be identified and corrected before the data is passed back to the requesting mechanism.

  This is optimal because it allows the user to keep on working and in the best case being notified that there was an error. This allows the user to take whatever measures are necessary to correct the situation later. This may include replacing a bad SIMM in the case of hardware errors or simply having the memory in question "scrubbed" for noise-related errors.

  "Scrubbing" involves reading a memory location and then writing the value and error bits back to DRAM. If there was a noise-related error it will be corrected. Otherwise, the incorrect checking information will always show up as an error and look like a hardware-related error if that specific memory location is not ever written out.

Hence, the benefits of ECC memory are obvious over Parity. You can probably expect to see certain BIOS or operating system software that actually scrubs memory perhaps in a background routine or SMM for BIOS, invisibly to the user.

1.4 Design Requirements and Intel Chipset Solutions

The board design must supply a predictable mechanism to generate SMI and NMI signals. Intel presently has two production memory controller chips that support ECC and Parity implementations, 82430HX and 82440FX. These chips have different pin layouts for providing the necessary signals to implement an ECC/Parity solution in BIOS.

This section discusses the solutions offered by the North Bridge chips 82430HX and 82440FX in conjunction with its South Bridge counterparts PIIX3 and PIIX4.

1.4.1 North Bridge

The 82430HX and 82440FX chips are similar but are distinguished by some errata and these functional differences:

- The 82430HX allows byte memory access in normal or parity mode and qword accesses only in ECC mode, whereas the 440FX dictates qword access at all times. This means that on the 82430HX, parity modules can support ECC mode but ECC modules cannot support parity mode. Normal or non-checking mode has no effect on the system. Also, the 82440FX can use ECC or parity modules for either mode since qwords are always in order.
- The 82430HX provides a flexible error indicating output signal that can operate in pulse or level mode, whereas the 82440FX does not. This allows the 82430HX to provide an indication of single bit or multiple bit errors through one signal when coupled with a South Bridge chip capable of distinguishing the pulse or level mode of operation.

The errata involved really only hinders the ability to offer a flexible solution and exist in early revisions of the 82430HX chip. It will not be considered in detail here. Later revisions of the chipset will be assumed in which all functionality is intact.

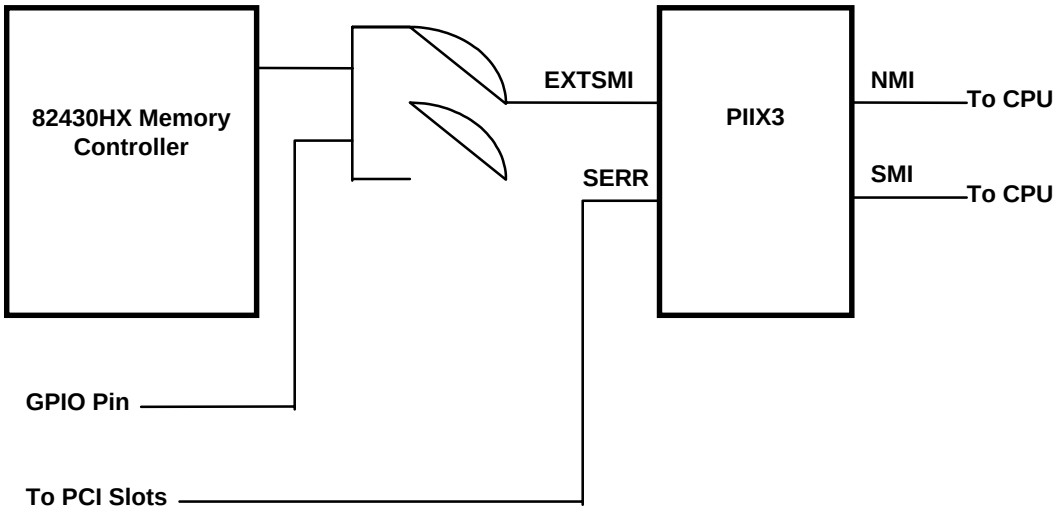
1.4.2 South Bridge

The second part of the equation is the South Bridge chip. Considered here is the PIIX3 and PIIX4. The PIIX3 has a flexible EXTSMI input that accepts pulse mode or level mode indication. When in level mode the input always generates a SMI to the CPU. Whereas, in pulse mode the PIIX3 will generate an NMI if a single pulse is received and a SMI if more than one clock pulse is received. The PIIX4 does not provide this functionality; instead it provides a single EXTSMI input.

1.5 BIOS Implementation Example

This BIOS implementation example for handling ECC uses a memory controller chip with the flexible pulse or level mode output signal and a South Bridge chip with the dual function input EXTSMI signal. The 82430HX and PIIX3 work together in this fashion when the SERR signal of the 82430HX is connected to the EXTSMI of the PIIX3.

Note that the output signal from the memory controller indicating an error is ANDed with some other controllable signal so that error generating can be disabled without hindering other functionality that may be connected to the EXTSMI input of the PIIX. A GPIO is used for this purpose. See the diagram on the next page.



1.5.1 BIOS implementation example

The ECC/Parity output signal (SERR) from the memory controller (82430HX) is connected to the EXTSMI (PIIX3) signal on the South Bridge ANDed with a GPIO. The memory controller signal can be pulsed or level driven and the input on the South Bridge can be programmed to accept either and vary its output via NMI or SMI accordingly. The AND gate is necessary for cases where the BIOS might want to disable error generating of SMI.

Future memory controllers and South Bridge chips may vary the naming of signals but it is expected that the functionality will remain the same to allow this example implementation to continue to operate.

1.6 BIOS Implementation Example Details

1.6.1 BIOS Design Goals

In general, the goal of BIOS and motherboard design is to provide a robust solution. Thus, it is desirable for the BIOS to provide a mechanism to identify parity errors or distinguishable single or multi-bit ECC errors and take appropriate action.

For this example implementation, the hardware configuration was outlined above. With the emergence of manageability applications in the corporate desktop marketplace, this example implementation further facilitates using an SMI handler to log ECC error events, single and multiple alike in the DMI space and report it through some desktop management software.

1.6.2 Detection

As mentioned in the “System BIOS Responsibility” section (1.3), the BIOS must detect ECC/Parity memory installed in the system during POST and initialize this memory appropriately. For the example implementation, detection is accomplished through testing memory by writing out a known pattern with parity or ECC enabled in the chipset. This ensures that the parity or ECC information is also written out to the additional bits used to store the error-checking data. Next, the memory is read back and the error-checking bits are compared with the actual data bits value read.

A failure to compare (as represented by the status register in the chipset) indicates an error. **Any hardware-imposed error will be detected at this point,** since they happen every time the memory in question is accessed. **If an error occurs, ECC or parity must be turned off in the chipset since ECC or parity memory is not present or has inherent hardware problems keeping it from functioning correctly.**

1.6.3 Initialization

Once ECC or parity is detected, this example implementation initializes **ALL DRAM** to the extent that the error-checking bits are initially written with the appropriate data before boot time to ensure that any operating system or application does not try to read a memory location before having this checking data written first.

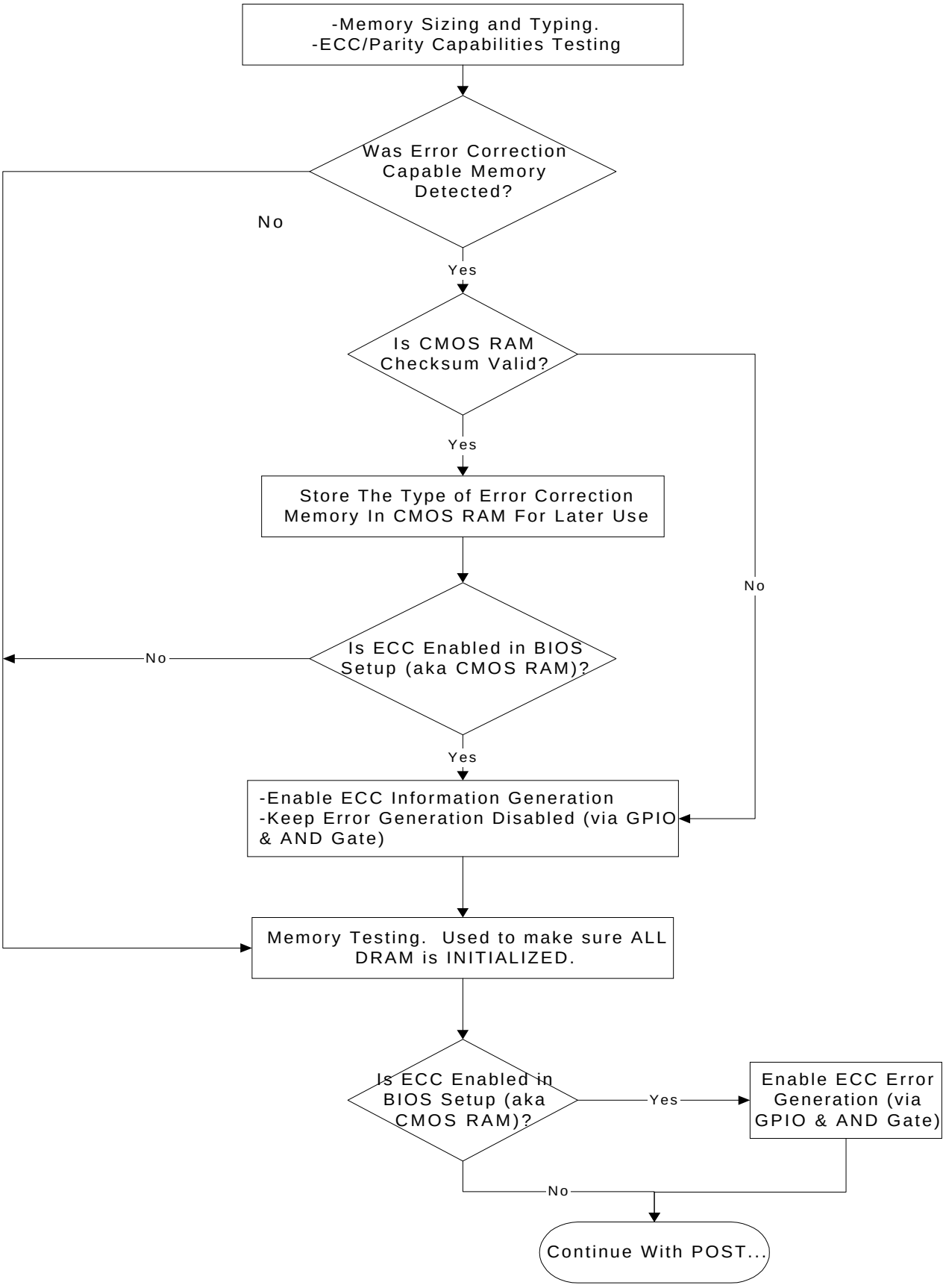
This includes all DRAM space, including but not limited to: Lower 640K, SMRAM, Expansion ROM space, BIOS ROM space, and all extended DRAM from 1M to Top Of Memory (TOM).

1.6.4 Setup

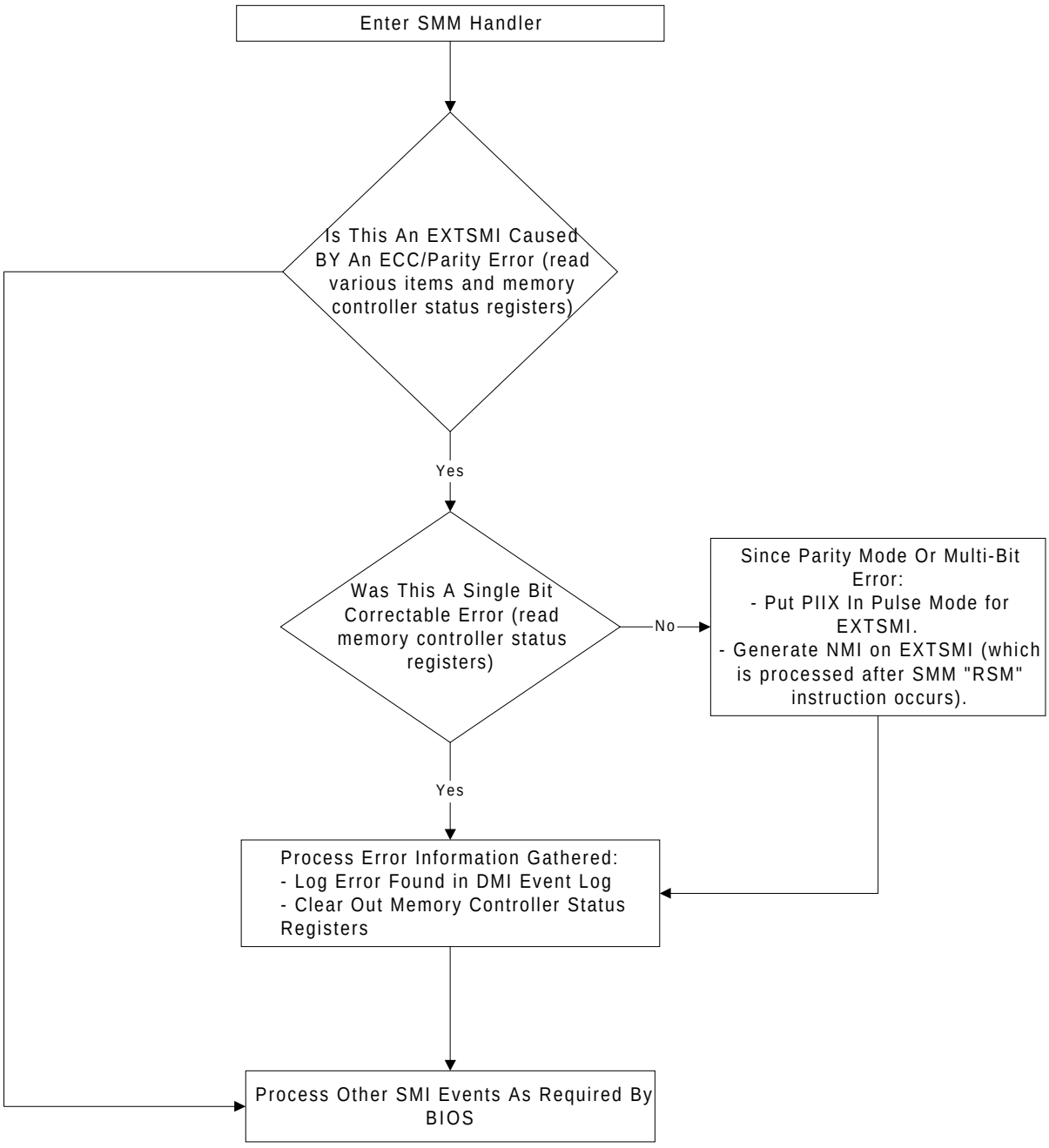
The BIOS example implementation setup reflects the capability of the system. This means that only the setup questions and options available to the system show up through dynamic setup nodes. For example, if the 82430HX memory controller is used and ECC memory is detected in the system, a Parity option is not visible in the option list for "Memory Error Correction." Also, if non-error correcting memory is detected in the system then the setup question for "Memory Error Correction" is not visible in setup.

1.6.5 Additional Important Sequence Details Included In The Sample Implementation

After memory sizing and before memory testing, the sample BIOS implementation tests the DRAM for ECC/Parity capabilities. The exact capabilities are stored in scratch CMOS for setup to use when making dynamic determination of what fields to make visible. If error correction capable memory is detected, the criteria on the next page is used:


-Memory Sizing and Typing.
-ECC/Parity Capabilities Testing
Was Error Correction Capable Memory Detected?
No
Yes
Is CMOS RAM Checksum Valid?
Yes
No
Store The Type of Error Correction Memory In CMOS RAM For Later Use
Is ECC Enabled in BIOS Setup (aka CMOS RAM)?
No
Yes
-Enable ECC Information Generation
-Keep Error Generation Disabled (via GPIO & AND Gate)
Memory Testing. Used to make sure ALL DRAM is INITIALIZED.
Is ECC Enabled in BIOS Setup (aka CMOS RAM)?
Yes
Enable ECC Error Generation (via GPIO & AND Gate)
No
Continue With POST...

The handling of errors at runtime varies, depending on the type of error. For the example implementation, on any error, an SMI is generated with the EXTSMI signal in level mode. The following steps are performed:



1. Therefore, the SMM handler exits normally at this point to allow the NMI to take place.
2. The example BIOS implementation's NMI handler distinguishes between Parity and Multiple bit ECC errors and displays a brief message indicating the type of error and the memory bank containing the

error. Note that in the case where the operating system hooks the NMI handler the BIOS has no control over the message displayed.

1.7 Summary

Most of the emphasis here was placed on the *example* BIOS implementation. The chipset and board design requirements are necessary to accomplish this. When designs vary from this example implementation, it may hinder feature support and ultimately cause customer dissatisfaction.

It is expected that the future memory controllers will provide the ability to write corrected data back out to the DRAM on correctable single bit ECC errors. This eliminates the need for memory scrubbing and soft errors manifesting themselves in multitude thus hindering performance.