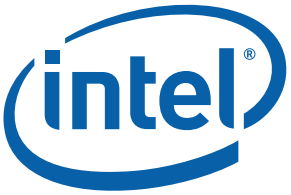

Firmware Resiliency with Intel® Ethernet Products

Application Note

Ethernet Products Group (EPG)

August 2020

Revision 2.0
616500-001

Revision History

Revision	Date	Comments
2.0	August 20, 2020	Initial public release.

Contents

NOTE: ***This page intentionally left blank.***

1.0 Introduction

Security of Intel products and our customers' trust is a top priority. This document provides product guidance for customers interested in enhanced platform firmware resiliency such as described in *NIST Special Publication 800-193 Platform Firmware Resiliency Guidelines*, referred to later in this document as "NIST SP800-193".

This document reviews the resiliency capabilities of several Intel® Ethernet LAN products, as well as best known design practices guidance to system designers using Intel® Ethernet products to improve the overall platform firmware resiliency against accidental or malicious modification of device firmware. As Intel® Ethernet products are only one ingredient in a system solution, this guidance is intended to complement an overall comprehensive system-level firmware resilience architecture.

As the NIST SP800-193 publication was released in March, 2018. Intel® Ethernet products released prior to 2018 require additional external circuitry, board or system design support to implement a NIST SP800-193 compliant design. This document outlines several strategies which can be used to provide security solutions for these prior devices.

As no one firmware resilience solution addresses all customer needs and requirements, this document provides design guidance relevant at the Intel® Ethernet component level, as well as highlighting existing relevant system manageability features, and lastly design consideration applicable at the board-layout level. System designers and integrators are encouraged to review and decide which recommendation best fits their specific combination of product usage and security considerations, design complexity and ease-of-use.

2.0 Choice of Security Approaches

NIST SP800-193 outlines three key principles for firmware resilience:

- **Protection** methods of the device integrity through ensuring the authenticity and integrity of firmware updates, and protecting critical data from corruption.
- **Detection** methods to identify when firmware or critical data has been corrupted.
- **Recovery** methods to restore firmware and critical data integrity in the event the firmware or critical data has been corrupted.

Overall, there are two general design strategies for preventing accidental or malicious modification. The first design strategy is Detect-and-Recover, where it is assumed that either the system or the end user detects the networking interface has become inoperable, and the system or end user attempts to recover or repair the interface. The second is Protect, using design strategy actively thwarts attempts to modify the device configuration or firmware, thus preventing the interface from entering a non-operational state. While some implementations historically used only one method, NIST SP800-193 recommends the use of both strategies to implement a more robust resiliency solution.

Detect-and-Recover approaches are generally easier to implement, with the compromise of the lag between the observation of the non-functioning or abnormal operation of the network interface, to the response where the fault is isolated and recovery initiated. During this time, the availability of the network interface may be reduced or exceed end customer business impact and risk objectives.

Protect approaches which thwart firmware modification address potential response lag time concerns, with the associated risk of exposure. However, these approaches come with the possible trade-off of making the device more difficult to configure in the field and service with firmware updates. As not all product design weaknesses may be known when an attack is initiated, the active thwart strategy is improved when coupled with an ongoing event detection and alerting capability at the system level.

As with most security solutions, the system designer typically must make a trade-off between ease-of-use, robustness of the security solution, and implementation cost. Typically choosing two categories forces a decision on the third. For example, a solution may be very robust and inexpensive to implement, yet increases the difficulty for the end-user to activate when needed.

Table 1 and Table 2 list categories of security approaches used on Intel Products.

Table 1. Approaches Used on Intel Products

Approach	Type	Description
Device-Level Resilience	Protection	Reflects devices which incorporate cryptographically signed firmware and device configuration images. A subsequent table further clarifies the resilience features as measured by NIST SP800-193 requirements.
I/O Interface Snooping	Detection	Applies to use of device IO interfaces such as SMBus, PCIe, SPI, etc. Does not thwart attempts at NVM modification, instead the Firmware and Configuration content is verified by a trusted agent elsewhere in the system.
I/O Interface Recovery	Detection Recovery	Same as Snooping, with the addition that the external agent can be used to overwrite the NVM Firmware and Configuration, providing a Recovery mechanism.
Simple Symbiont	Recovery	Applies to use of simple hardware, like a jumper, under control of a trusted agent. NIST SP800-193 includes several provisions where local physical presence obviates several of the more complex security requirements. The assumption in NIST SP800-193 indicates physical presence is a sufficient security countermeasure to ensure the integrity of device upgrades or recovery is under controlled circumstances. These devices implement hardware-default values used when the SPI or EEPROM is not detected, allowing for a very basic recovery method if NVM content is corrupted. Furthermore, configuration NVM write protection may also be manually controlled.
Symbiont	Protection Detection Recovery	Covers a general industry solution described in NIST SP800-193 as the symbiont model, where an external security device implements the various security controls on behalf of another device.

Table 2. Intel Products and Types of Approaches Supported

Ethernet Device	Resilience	SMBus Recovery	Jumper Recovery	Symbiont Support
Intel® Ethernet 800 Series Controllers	Yes		Yes	Yes
Intel® Ethernet 700 Series Controllers (X710/XXV710/XL710)	Yes		Yes	Yes
Intel® Ethernet Controller X710-TM4/AT2 (X710-TM4/AT2)	Yes		Yes	Yes
Intel® Ethernet Controller X550	Yes		Yes	Yes
Intel® Ethernet Connection X722	Yes		N/A	Yes
Ethernet Controller in the Intel® Xeon® Processor D-1500 Product Family LAN Controller	Yes		Yes	Yes
Ethernet Controller in the Intel Atom® Processor C3000 Series	Yes		N/A	Yes
Intel® Ethernet Controller I210/I211	Yes		Yes	Yes
Intel® Ethernet Controller I350		Yes	Yes	Yes
Intel® Ethernet Network Connection I347		Yes	Yes	Yes
Intel® Ethernet Controller I225		Yes	Yes	Yes
Intel® Ethernet Controller X540		Yes	Yes	Yes
Intel® 82599 10 Gigabit Ethernet Controller		Yes	Yes	Yes
Intel® 82574 Gigabit Ethernet Controller			Yes	Yes

3.0 Device-Level Resilience

As detailed in Table 2, Intel® Ethernet products incorporate device resilience as implemented by signed firmware update authentication methods, which resist unauthorized modification of device firmware and device critical initialization settings, and PCI Expansion ROM images.

The Intel® Ethernet Controller E810 (E810) is the first Intel® Ethernet LAN product to implement the immutable root of trust requirements called for in NIST SP800-193. Several prior Intel LAN products with resilience implemented many of the NIST SP800-193 requirements, including cryptographically signed and authenticated NVM updates, methods to thwart corruption of the device firmware or configuration, anti-rollback controls, as well as a recovery mode feature.

The primary gap with prior generation resilient products revolve around firmware verification on device boot - as these earlier generation products only verified firmware update with mutable authentication methods and keys. If NIST SP800-193 compliance is not a requirement, these earlier generation products may be sufficient to meet overall system security requirements.

NIST SP800-193 requirements can be met with Intel® Ethernet resilient products prior to the E810, as well as even older, yet still very popular, Intel LAN products when used in conjunction with system and board-level design solutions. NIST SP800-193 generally describes this model of applying security externally to a device as **symbiont mode**. Some symbiont design strategies, considerations and key product features for these older products are described later in this document.

Table 3 summarizes the mandatory functional requirements of NIST SP800-193, as well as compliance of various resilient Intel® Ethernet devices.

Table 3. NIST SP800-193 Compliance of Intel® Ethernet Devices

Ethernet Device	Intel® Ethernet Controller I210	Intel® Ethernet Controller X550 (including D-1500 and C3000)	Intel® Ethernet 700 Series (including X722)	Intel® Ethernet 800 Series
Hardware-based RoT	No	No	No	Yes
Signed and Authenticated Firmware Updates	Yes	Yes	Yes	Yes
RoT-based Recovery Mode	No	No	No	Yes
FIPS 186-4 Digital Signature Standard approved signing algorithm	Yes	Yes	Yes	Yes
Signatures with 112-bit security strength (SP 800-57)	Yes	Yes	Yes	Yes
Firmware Updates signed by Authorized Entities[1] SP800-89	Yes	Yes	Yes	Yes
Update only via secure method except with physical presence (no bypass requirement)	Yes	Yes	Yes	Yes
Built-in detection of firmware or critical setting corruption	No	Yes	Yes	Yes
Device firmware and critical setting protection from malicious modification	Yes	Yes	Yes	Yes
Factory Defaults protected from malicious modification	No	Yes	Yes	Yes
Firmware integrity verification with cryptographic algorithm on device boot	No	No	No	Yes
Device critical settings verification on device boot	No	Yes	Yes	Yes
Device initialization watchdog timer	No	Yes	Yes	Yes

Table 3. NIST SP800-193 Compliance of Intel® Ethernet Devices [continued]

Ethernet Device	Intel® Ethernet Controller I210	Intel® Ethernet Controller X550 (including D-1500 and C3000)	Intel® Ethernet 700 Series (including X722)	Intel® Ethernet 800 Series
Autonomous activation of device recovery on detection of corruption	No	Yes	Yes	Yes
Device recovery uses authenticated update methods to implement image restore	No	Yes	Yes	Yes
Recovery protects against unauthorized recovery to an earlier, less secure firmware image	Yes	Yes	Yes	Yes
Known good device critical settings protected from malicious modification	No	Yes	Yes	Yes
Device critical settings are automatically stored.	No	Yes	Yes	Yes
Recovery must be capable of restoring factory defaults	No	Yes	Yes	Yes

1. Currently Intel is the only "Authorized Entity". Intel does not currently support a co-signing model.

NVM updates to resilient Intel LAN products are cryptographically verified currently with a SHA2 256b hash and RSA 2048 public signature. The E810 update process requires a firmware reset, which on the 810, repeats a ROM-based authentication out of device boot.

The 810 corruption detection and recovery modes of operation are controlled from a chained root of trust via enhanced device parameter verification, a second stage firmware boot-up watchdog timer, and lastly recovery mode activation and a partially automatic device recovery process.

With the Intel LAN release version **24.1** or later, resilient Intel LAN devices implement a built-in recovery mode option. Recovery mode is automatically activated upon a device startup watchdog timer expiration. The watchdog timer is activated after power-on, and if the firmware fails to start as measured by excessive firmware resets, several device settings are proactively restored to factory default settings, and minimal functionality enabled to allow additional host software or manageability devices to complete restoring the device to the desired configuration. Customers are encouraged to review the related *Recovery Mode in Intel® Ethernet Devices/Adapters Application Note* (Document Number: 606286), for additional details about the device and system behavior.

Note: The Intel® Ethernet Controller I210 (I210) does not implement an automatic recovery mechanism.

All Intel® Ethernet products are provided with signed UEFI Option ROM pre-boot UNDI drivers, and system integrators are encouraged to implement firmware signature verification in their system firmware initialization.

Note: The I210 does not explicitly prevent modification of the PCI Expansion ROM image, although UEFI UNDI drivers loaded from the Expansion ROM BAR can still be independently authenticated by the system firmware. Other resilient Intel LAN products perform an additional integrity check on update preventing non-Intel authentic option ROM images from being applied to the device.

Lastly, Intel provides the NVMCHECK software application which can be used to measure the running firmware and configuration measurement hash value, and verify the cryptographic integrity of the running firmware installed in the device. Similar functionality is incorporated in the PROSet online diagnostics utilities installed with Microsoft* Windows* Intel® Ethernet device drivers.

4.0 I/O Interface Support with Intel Products

For Intel® Ethernet devices without device-level resiliency (or not fully NIST SP800-193 compliant), alternatives exist for system designers to provide a level of recovery and potentially detection.

"I/O Interfaces" refers to standard adapter interfaces, such as PCIe and SMBus. For example, on the PCIe Adapter Edge connection.

This method uses these interfaces to provide detection and/or recovery for on-board NVM. For example, some on-board EEPROMs (FRUs) or other devices may be written from the SMBus. Likewise the controller device Flash may be written over PCIe or SMBus in conjunction with device firmware.

5.0 Symbiont Support with Intel Products

5.1 Symbiont Overview

For Intel® Ethernet devices without device-level resiliency, or not fully NIST SP800-193 compliant, alternatives exist for system designers to include Intel LAN in an overall platform firmware resiliency strategy. Recalling the NIST SP800-193 principles of protection, detection and recovery, this section discusses suggested design options aligned along these guiding principles.

For Intel LAN devices which do not incorporate an internal hardware-based root of trust (RoT), nor an immutable root of trust for recovery, or do not cryptographically verify firmware prior to device boot, a dedicated **external** security device may be attached to implement such functionality. We will refer to external security devices implementing the RoT for both integrity, protection and recovery the **external Root of Trust** (eRoT).

If NIST SP800-193 is not a requirement, existing resilient Intel LAN device solutions may be sufficient, or potentially lower-cost design options may achieve targeted system security requirements.

5.2 Symbiont Models

As mentioned in Table 1, two models for Symbiont Support are discussed here, a simple model and a full model.

In the Simple model, the eRoT is used to control recovery. In this case, it masters the Flash or EEPROM control signals such that a new, golden image can be written.

In the Full model, the eRoT provides protection, detection, and recovery. It acts as an interposer or look-aside agent.

5.3 Simple-Symbiont Security

There are two manifestations of the simple-symbiont option.

- Jumper-based with software update.
- Jumper-based with hardware update.

While jumper-based security is perhaps the least expensive option to implement for resiliency, it is also the most intrusive for end users to implement protection and recovery options. By its very nature, jumper-based solutions require a physical presence. Jumper-based solutions do not provide ongoing integrity checking, although through a combination of a jumper on the write protection (WP#) pins to the configuration NVM, as well as jumper on the chipset select (CS#) signal to the devices, a basic security solution to lock and unlock the device configuration is possible.

Intel® Ethernet devices perform an initial very basic two bit EEPROM signature check on contents of the attached EEPROM or SPI FLASH device used for storing device configuration and firmware. This check is performed on LAN_PWR_GOOD assertion. The primary purpose of this signature check is detect either a disconnected or blank EEPROM or SPI device.

5.3.1 Jumper-based with Software Update

This applies to devices which support a blank-flash-programming mode, like the Intel® Ethernet 800 Series. In this simplest case a jumper is used on the chip select signal for the Flash, which allows the eRoT to disconnect chip select at device boot-time. This puts the device into blank-flash-programming mode and allows software to write a new, golden image into the flash. The device starts with built-in hardware default values, and can be restored using Intel-provided NVM update tools. After update, the device must be reset again with a LAN_PWR_GOOD cycle to load the new configuration settings.

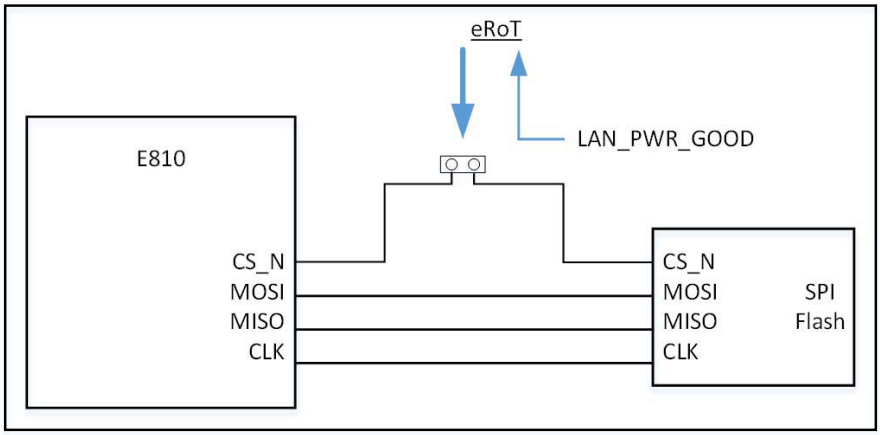

Figure 1. Example Recovery Jumper Implementation with Software Update

5.3.2 Jumper-based with Hardware Update

In this case all the control signals are brought to a header under eRoT control. The eRoT directly writes the Flash device during the window prior to LAN_PWR_GOOD assertion, when the controller device has not booted yet.

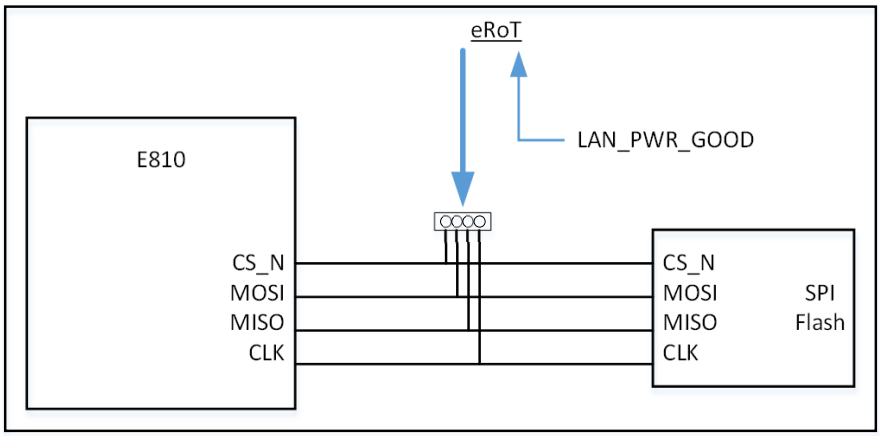

Figure 2. Example Recovery Jumper Implementation with Hardware Update

5.4 Full Symbiont Security

5.4.1 Protection, Detection, and Recovery

All of the products discussed here use SPI flash devices or four-wire EEPROM devices. Using an eRoT device, a common design approach inserts the security device between the Intel LAN device and the configuration SPI or EEPROM, this is called the interposer model. An alternative method implements a look-aside model.

This section describes the pros and cons of both methods in terms of protection, detection, and recovery.

In all of these cases, the eRoT is assumed to perform a cryptographically based measurement of the device NVM. The eRoT is assumed to perform a re-measurement after any firmware update or device configuration change, which is assumed to be under controlled conditions.

5.4.2 Look-aside Model

In the look-aside model, the eRoT acts as a supervisor to the device boot process. In this model, the device is assumed to be good until proven corrupt. This solution works best for devices in the SMBus-based recovery category.

Note: For normal boot operation, the eRoT does not interfere with the boot sequence or timing.

Note: For Intel LAN solutions that support SMBus-based recovery, this method can also be adapted to apply to add-in Intel® I350 adapter solutions where the platform also connects a trusted host interface to the SMBus signals to the PCIe slot.

5.4.3 Look-aside Example 1

Figure 3 illustrates one possible implementation of this model, where the primary configuration Flash is can be monitored and updated by the eRoT.

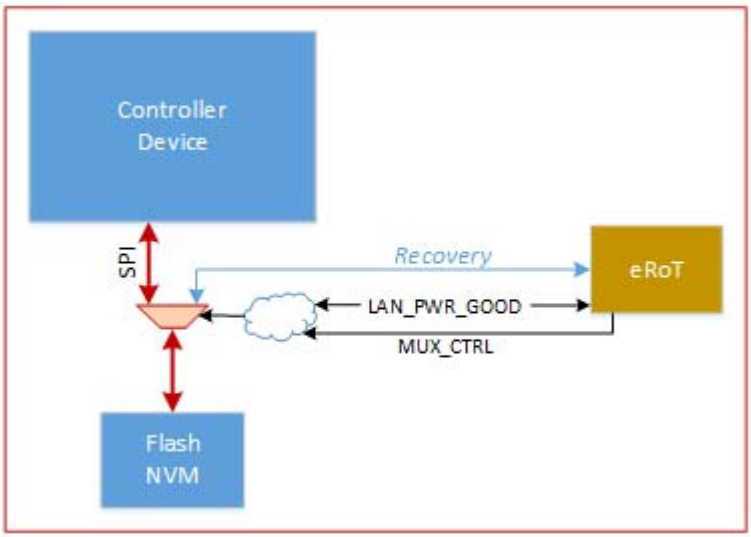

Figure 3. Look-aside Model Example with eRoT Monitor/Recovery

In this example, the eRoT has the ability to read the Flash image prior to Controller boot by holding LAN_PWR_GOOD de-asserted and interrogating the Flash image. The eRoT can also overwrite the Flash image with a golden image.

5.4.4 Look-aside Example 2

Figure 4 illustrates another possible implementation of this model, where the primary configuration Flash is muxed with a golden Flash under control of the eRoT.

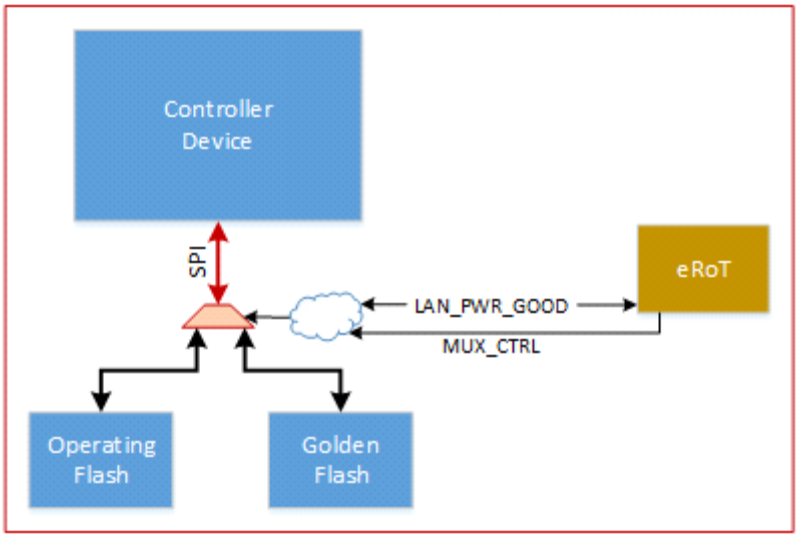

Figure 4. Look-aside Model Example with eRoT Muxing Between Operating and Golden Flashes

In this example, a redundant reference, called a **golden image** Flash is installed alongside the primary Intel Controller configuration Flash, and the Flash connection to the Intel Controller is selectable via a mux driven by the eRoT.

During manufacturing, both Operating and Golden Flashes are programmed to contain identical content, with the Golden Flash protected from modification with an option to tie the Flash write protect (WP#) to ground.

5.4.5 Look-aside Model Detection

To implement detection, while holding LAN_PWR_GOOD de-asserted, the eRoT measures the EEPROM or Flash image using a FIPS-146 compliant cryptographic hash algorithm. This measurement can furthermore be optionally stored in the platform TPM PCR7 register to be included as part of the system secure boot. For resilient Intel LAN devices, the NVMCHECK tool, or associated Intel NVMCHECK library, can be used to implement this integrity check algorithm.

Note:The eRoT must either control LAN_PWR_GOOD or guarantee access timing such that the eRoT completes its operations prior to device boot.

If a measurement unexpectedly fails, the eRoT initiates the recovery flow described below.

5.4.6 Look-aside Model Protection

To ensure ongoing integrity and protection, the configuration is locked down via assertion of the WP# signal, with the assumption configuration settings never need to change after initial system provisioning, except for firmware updates.

5.4.7 Look-aside Model Recovery

To implement recovery, the eRoT controls the device MUX, cycles LAN_PWR_GOOD, which initializes the device from of a known good golden image protected via physical means.

At this point, the system designer has a choice whether to perform a factory restore operation, or a repair operation. The difference between the two is whether to read the golden image, instruct the eRoT to toggle the recovery signal, then write into the primary device, or bypass reading the golden image and use Intel provided firmware update tools to write a known good firmware image. Intel provided firmware update tools by design preserve mutable settings, such as MAC address, PCI VPD data, PBA identification, pre-boot configuration settings, and other possible OEM-specific settings.

The look-aside model can be implemented with fewer pins routed to a dedicated eRoT, where the recovery partitions can be located closer physically to the protected device.

Note: The eRoT must be able to control the reset sequence including LAN_PWR_GOOD in order to reboot the device after an update. Alternatively the system can be rebooted or power-cycled after an update.

5.4.8 Look-aside Model Example Operation

During platform startup, the eRoT holds the Intel Controller in LAN_PWR_GOOD reset, the Flash mux switched to enable the eRoT to directly access the Operating Flash, and the eRoT performs a measurement of the configuration Flash. If the measurement results in an unexpected result, the eRoT may automatically initiate a recovery mode.

During normal operation, the eRoT controls mux select such that the Operating Flash is connected to the Intel Controller. Ongoing protection of the configuration Flash may be implemented by assertion of the WP# signal to the configuration Flash.

If a device recovery mode is activated, the Intel Controller is held in LAN_PWR_GOOD reset, and the Flash mux is switched to enable the eRoT to directly access the Operating Flash. The eRoT is assumed to keep a protected, known-good restoration image for the specific device, which includes unique device settings such as the MAC address. How this restoration image is provisioned or updated in the recovery-mode partition is outside of the scope of this document.

After the configuration Flash restore is complete, the eRoT sets the mux for normal operation, which re-connects the Intel Controller to the Operating Flash. The configuration is protected by the eRoT by asserting WP# to the configuration Flash. The eRoT releases the Intel Controller from reset. The Intel Controller begins normal device startup with the new settings.

5.4.9 Interposer Model

If the eRoT acts as a complete interposer between the device and the NVM, the eRoT can directly measure the configuration NVM and verify integrity, implement ongoing patrolling of runtime accesses to the NVM preventing corruption from accidental or malicious actions, and implement recovery operations independent of the Intel LAN controller.

It is important to note that for this model the eRoT must meet all protocol, electrical, and timing constraints of a non-interposed design. Effectively, the controller cannot know the difference between connecting to the eRoT or the NVM it expects. For this reason, Intel cautions use of this model.

To implement detection, sometime prior to device startup to no later than OS boot, the EEPROM or SPI image is measured using a FIPS-146 compliant cryptographic hash algorithm. This measurement can furthermore be optionally stored in the platform TPM PCR7 register to be included as part of the system secure boot.

To implement ongoing protection, either the eRoT may implement patrolling of the EEPROM or SPI transactions, or alternatively control the write protection (WP#) signal to the configuration EEPROM or SPI. For resilient Intel LAN products, the eRoT can rely on built-in defenses in the Intel LAN device designed to thwart attempts to bypass protection. For non-resilient Intel LAN products, the configuration must be locked down via assertion of the WP# signal, or use an Intel I350 specific method of protection described later. Configuration lock-down is generally acceptable, as configuration settings generally do not change after initial system provisioning, except for firmware updates.

Patrolling of transactions requires the eRoT to incorporate device-specific EEPROM layout details to identify what specific regions of the protected device must be protected, which could be modified. Device firmware updates may change layout details from device NVM version to version. Because of these complexities, Intel does not recommend patrolling on Intel Ethernet controllers.

A more pragmatic method is simply to only allow updates during certain trusted windows of time during the system boot process prior to operating system boot. Any configuration changes are applied using trusted methods with integrity, evaluate changes, roll back if necessary, and otherwise lock down the configuration via assertion of the WP# signal. In practice, configuration settings typically do not change after initial system provisioning, except for firmware updates.

Note: The Intel® Ethernet QSFP+ Configuration Utility (QCU tool) modifies the Intel® Ethernet Network Adapter X710 NVM to reconfigure the device into either four individual 10 Gb/s ports or a single 40 Gb/s port. This tool is often run from the host operating system.

Note: The Intel® Ethernet Controller I350 includes a programmatic method to lock down the EEPROM configuration image. Details are presented Section 5.7.

To implement recovery, device-specific firmware updates must be verified and staged in the recovery partition used by the eRoT. Device-specific parameters must be merged into the firmware update to preserve critical mutable settings, such as MAC Address, PCI VPD data, PBA identification, pre-boot configuration settings, and other possible OEM-specific settings. To apply the update, the eRoT would initiate a recovery mode on the device to apply the update from the recovery partition rather than depend on sequencing operations with host-based update software.

The interposer solution presents a practical scaling problem, as designing a single eRoT with enough dedicated pins to service all of the critical platform devices is complex, as is board routing issues for accessing devices distributed physically across the motherboard. As a result, only a critical few platform devices are protected with such a scheme, and does not extend to add-in adapters.

5.4.10 Interposer Example

Figure 5 illustrates one possible concept of this model, where the primary configuration Flash is fully interposed by the eRoT between the Flash and the Intel Controller.

Note: This concept as a design implementation has not been validated by Intel. It is for conceptual purposes only.

Note: This design concept eliminates the direct attached Flash and instead routes all SPI Flash transactions through the eRoT instead. This diagram is intended to make the comparison between the interposer and look-aside models easier to visualize.

Note: Another implementation concept could replace the Flash or EEPROM device completely and the eRoT could emulate that device behavior.

Note: The eRoT must be fully compliant with the interface protocol (i.e. SPI or I^2C), electricals, and timing for both sides of the bus.

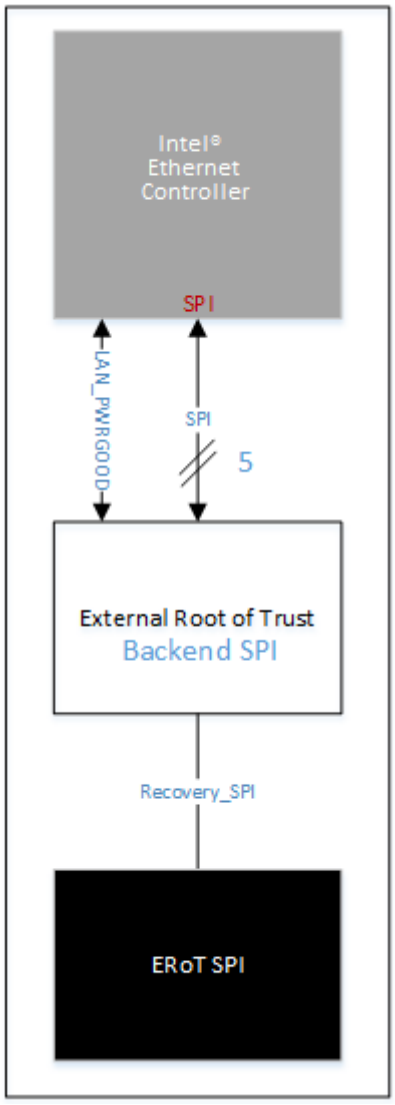

Figure 5. Interposer Model Example with eRoT Directly Accessing and Restoring Device Flash

Figure 6 an example where the eRoT is translating from a local SPI Flash directly connected to the eRoT to the controller which uses a 2-wire EEPROM interface.

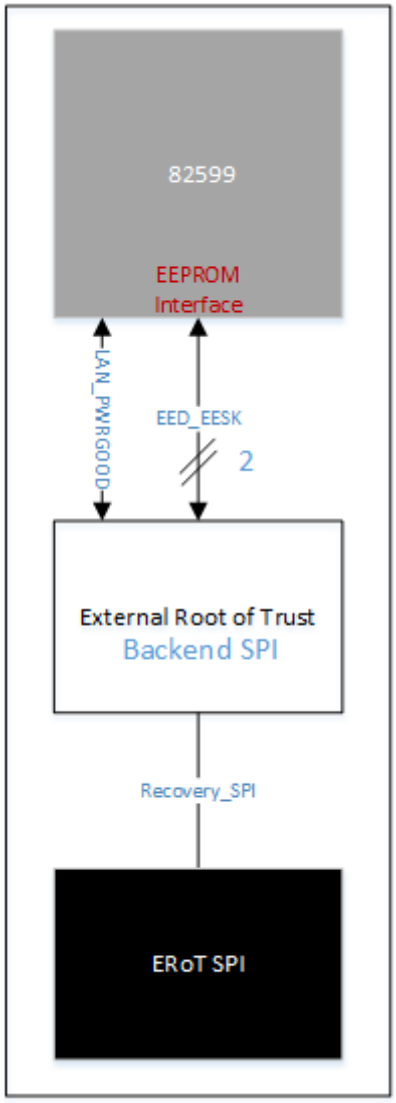

Figure 6. Interposer Model Example with eRoT Translating 2-Wire to SPI

This model provides Protection, Detection, and Recovery. Because the eRoT fully interposes the NVM-to-Controller path, all interactions are monitored and controlled.

5.5 Reset Sequence and Timing of Control

For all designs discussed, system designers must be aware that generally Intel LAN device initialization functions occur on power good (LAN_PWR_GOOD) assertion to the device, and a subset of functions are reset on PCIe reset assertion. It is not generally sufficient to assert reset (PERST#) to the LAN device to prevent the device from initialization. the eRoT must observe and/or control the LAN_PWR_GOOD signal assertion to the Intel LAN device to prevent device startup.

Figure 7 shows an example for the case of eRoT update of the operating Flash. The eRoT window for interrogating or updating the Flash lies before LAN_PWR_GOOD is asserted. The eRoT may directly control LAN_PWR_GOOD to extend this window.

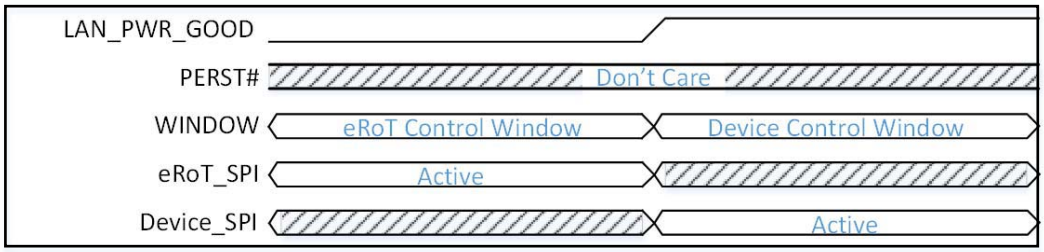

Figure 7. LAN_PWR_GOOD Conceptual Timing Diagram

5.6 JTAG and Security

JTAG (or Boundary Scan) is a common tool used for debug and manufacture of boards, and is specified in IEEE 1149.1. Beyond this specification, some devices allow this port to be used to access and program internal registers and function.

The JTAG hardware interface is utilized and required for initial board debug on a lab bench, and then also for manufacturing. This is intended for Intel use. Typically, there is a header on the board, which is populated for initial debug and connected to an external JTAG access tool. After initial debug, the header is depopulated, however, the pads for the header remain. In manufacture an ICT tester (or equivalent) is used to conduct boundary scan by touching-down on the pads. Because the pads are exposed on the PCB, there is a security risk which falls under the category of a "physical access risk". An attacker would need to have physical access to the board.

5.6.1 Intel® Ethernet Controller E810 - Secure JTAG

Unlike earlier devices, the Intel® Ethernet Controller E810 protects its JTAG interface via an unlock mechanism, so physical access to the JTAG header and/or pads is not sufficient to allow an attacker to gain JTAG access to the device. Only Intel can unlock a device for JTAG debug.

5.7 Intel® Ethernet Controller I350 Specific EEPROM Protection Method

For Intel® Ethernet devices without resiliency, such as the Intel® Ethernet Controller I350, recovery can be initiated via an SMBus message to invalidate the EEPROM or SPI FLASH attached to the LAN device.

ROM-based firmware included in the Intel LAN devices can receive an SMBus command which invalidates a simple signature used by the device to detect a valid device configuration. Afterwards, the BIOS or operating system initiates a reset to force an EEPROM or SPI FLASH auto-load process that fails, and the Intel LAN device initializes with hardware defaults and enables host access to the device.

Firmware is programmed to receive such a command only from PCIe reset until one of the functions changes its status from D0u (PCI Configuration BARs uninitialized) to D0a (PCI Configuration BARs initialized). Once one of the functions moves to D0a, it can be safely assumed that the device is accessible to the host and there is no further need for this function.

The command is sent on a fixed SMBus address of 0xC8. The format of the SMBus Block Write command is as follows:

Function	Command	Data Byte
Release EEPROM	0xC7	0xB6

Notes:

- In case more than one device is in a state to accept this solution, all devices connected to the same SMBus accept the command. The devices in D0u state invalidate their EEPROM as a consequence.
- After receiving a release EEPROM command, firmware is still in this hardware-default state until the EEPROM is re-initialized, and the firmware reset.
- A pseudocode example to issue the SMBus command from a baseboard management controller (BMC) follows:

```
Smbus_Block_Write(Address=0xC8, Command=0xC7, NumberBytes=1, Data[0]=0xB6)
```

The Intel® Ethernet Controller I350, implements an EEPROM protection method that can be used to lock down the configuration EEPROM from modification by host software. The modification makes the entire EEPROM read-only to software running on the host CPU. This EEPROM protection can only be reset by SMBus-based recovery command as described later in this document.

To implement this read-only protection, the configuration EEPROM must be accessed and re-programed:

1. Program start and end address of protection - EEPROM 2C=0, 2Dh=size in words of EEPROM.
2. Set the EPROM size in the EEPROM word 12h[13:10].
3. Enable rollover protect - EEPROM 12h[8].
4. Set EE Burst Limit - EPROM 12h[9].
5. Set EEPROM read/write protect region - EEPROM word 12h[3:0] = 4'b0000.
6. Set EEPROM Protection bit - EEPROM Word 12h[4].
7. Reboot or issue PERST# to the adapter.

NOTE: ***This page intentionally left blank.***

LEGAL