pcieaer-howto.txt

The PCI Express Advanced Error Reporting Driver Guide HOWTO

T. Long Nguyen <tom.l.nguyen@intel.com>
Yanmin Zhang <yanmin.zhang@intel.com>

07/29/2006

1. Overview

1.1 About this guide

This guide describes the basics of the PCI Express Advanced Error Reporting (AER) driver and provides information on how to use it, as well as how to enable the drivers of endpoint devices to conform with PCI Express AER driver.

1.2 Copyright (C) Intel Corporation 2006.

1.3 What is the PCI Express AER Driver?

PCI Express error signaling can occur on the PCI Express link itself or on behalf of transactions initiated on the link. PCI Express defines two error reporting paradigms: the baseline capability and the Advanced Error Reporting capability. The baseline capability is required of all PCI Express components providing a minimum defined set of error reporting requirements. Advanced Error Reporting capability is implemented with a PCI Express advanced error reporting extended capability structure providing more robust error reporting.

The PCI Express AER driver provides the infrastructure to support PCI Express Advanced Error Reporting capability. The PCI Express AER driver provides three basic functions:

- Gathers the comprehensive error information if errors occurred.
- Reports error to the users.
- Performs error recovery actions.

AER driver only attaches root ports which support PCI-Express AER capability.

2. User Guide

2.1 Include the PCI Express AER Root Driver into the Linux Kernel

The PCI Express AER Root driver is a Root Port service driver attached to the PCI Express Port Bus driver. If a user wants to use it, the driver has to be compiled. Option CONFIG_PCIEAER supports this capability. It depends on CONFIG_PCIEPORTBUS, so pls. set CONFIG_PCIEPORTBUS=y and CONFIG_PCIEAER = y.

2.2 Load PCI Express AER Root Driver

There is a case where a system has AER support in BIOS. Enabling the AER Root driver and having AER support in BIOS may result unpredictable behavior. To avoid this conflict, a successful load of the AER Root driver requires ACPI _OSC support in the BIOS to allow the AER Root driver to request for native control of AER. See the PCI FW 3.0 Specification for

details regarding OSC usage. Currently, lots of firmwares don't provide
_OSC support while they use PCI Express. To support such firmwares,
forceload, a parameter of type bool, could enable AER to continue to
be initiated although firmwares have no _OSC support. To enable the
walkaround, pls. add aerdriver.forceload=y to kernel boot parameter line
when booting kernel. Note that forceload=n by default.

nosourceid, another parameter of type bool, can be used when broken
hardware (mostly chipsets) has root ports that cannot obtain the reporting
source ID. nosourceid=n by default.

2.3 AER error output
When a PCI-E AER error is captured, an error message will be outputed to
console. If it's a correctable error, it is outputed as a warning.
Otherwise, it is printed as an error. So users could choose different
log level to filter out correctable error messages.

Below shows an example:
0000:50:00.0: PCIe Bus Error: severity=Uncorrected (Fatal), type=Transaction
Layer, id=0500(Requester ID)
0000:50:00.0:   device [8086:0329] error status/mask=00100000/00000000
0000:50:00.0:   [20] Unsupported Request   (First)
0000:50:00.0:   TLP Header: 04000001 00200a03 05010000 00050100

In the example, 'Requester ID' means the ID of the device who sends
the error message to root port. Pls. refer to pci express specs for
other fields.

3. Developer Guide

To enable AER aware support requires a software driver to configure
the AER capability structure within its device and to provide callbacks.

To support AER better, developers need understand how AER does work
firstly.

PCI Express errors are classified into two types: correctable errors
and uncorrectable errors. This classification is based on the impacts
of those errors, which may result in degraded performance or function
failure.

Correctable errors pose no impacts on the functionality of the
interface. The PCI Express protocol can recover without any software
intervention or any loss of data. These errors are detected and
corrected by hardware. Unlike correctable errors, uncorrectable
errors impact functionality of the interface. Uncorrectable errors
can cause a particular transaction or a particular PCI Express link
to be unreliable. Depending on those error conditions, uncorrectable
errors are further classified into non-fatal errors and fatal errors.
Non-fatal errors cause the particular transaction to be unreliable,
but the PCI Express link itself is fully functional. Fatal errors, on
the other hand, cause the link to be unreliable.

When AER is enabled, a PCI Express device will automatically send an
error message to the PCIe root port above it when the device captures

an error. The Root Port, upon receiving an error reporting message, internally processes and logs the error message in its PCI Express capability structure. Error information being logged includes storing the error reporting agent's requestor ID into the Error Source Identification Registers and setting the error bits of the Root Error Status Register accordingly. If AER error reporting is enabled in Root Error Command Register, the Root Port generates an interrupt if an error is detected.

Note that the errors as described above are related to the PCI Express hierarchy and links. These errors do not include any device specific errors because device specific errors will still get sent directly to the device driver.

3.1 Configure the AER capability structure

AER aware drivers of PCI Express component need change the device control registers to enable AER. They also could change AER registers, including mask and severity registers. Helper function pci_enable_pcie_error_reporting could be used to enable AER. See section 3.3.

3.2. Provide callbacks

3.2.1 callback reset_link to reset pci express link

This callback is used to reset the pci express physical link when a fatal error happens. The root port aer service driver provides a default reset_link function, but different upstream ports might have different specifications to reset pci express link, so all upstream ports should provide their own reset_link functions.

In struct pcie_port_service_driver, a new pointer, reset_link, is added.

pci_ers_result_t (*reset_link) (struct pci_dev *dev);

Section 3.2.2.2 provides more detailed info on when to call reset_link.

3.2.2 PCI error-recovery callbacks

The PCI Express AER Root driver uses error callbacks to coordinate with downstream device drivers associated with a hierarchy in question when performing error recovery actions.

Data struct pci_driver has a pointer, err_handler, to point to pci_error_handlers who consists of a couple of callback function pointers. AER driver follows the rules defined in pci-error-recovery.txt except pci express specific parts (e.g. reset_link). Pls. refer to pci-error-recovery.txt for detailed definitions of the callbacks.

Below sections specify when to call the error callback functions.

3.2.2.1 Correctable errors

Correctable errors pose no impacts on the functionality of
the interface. The PCI Express protocol can recover without any
software intervention or any loss of data. These errors do not
require any recovery actions. The AER driver clears the device's
correctable error status register accordingly and logs these errors.

3.2.2.2 Non-correctable (non-fatal and fatal) errors

If an error message indicates a non-fatal error, performing link reset
at upstream is not required. The AER driver calls error_detected(dev,
pci_channel_io_normal) to all drivers associated within a hierarchy in
question. for example,
EndPoint<==>DownstreamPort B<==>UpstreamPort A<==>RootPort.
If Upstream port A captures an AER error, the hierarchy consists of
Downstream port B and EndPoint.

A driver may return PCI_ERS_RESULT_CAN_RECOVER,
PCI_ERS_RESULT_DISCONNECT, or PCI_ERS_RESULT_NEED_RESET, depending on
whether it can recover or the AER driver calls mmio_enabled as next.

If an error message indicates a fatal error, kernel will broadcast
error_detected(dev, pci_channel_io_frozen) to all drivers within
a hierarchy in question. Then, performing link reset at upstream is
necessary. As different kinds of devices might use different approaches
to reset link, AER port service driver is required to provide the
function to reset link. Firstly, kernel looks for if the upstream
component has an aer driver. If it has, kernel uses the reset_link
callback of the aer driver. If the upstream component has no aer driver
and the port is downstream port, we will perform a hot reset as the
default by setting the Secondary Bus Reset bit of the Bridge Control
register associated with the downstream port. As for upstream ports,
they should provide their own aer service drivers with reset_link
function. If error_detected returns PCI_ERS_RESULT_CAN_RECOVER and
reset_link returns PCI_ERS_RESULT_RECOVERED, the error handling goes
to mmio_enabled.

3.3 helper functions

3.3.1 int pci_enable_pcie_error_reporting(struct pci_dev *dev);
pci_enable_pcie_error_reporting enables the device to send error
messages to root port when an error is detected. Note that devices
don't enable the error reporting by default, so device drivers need
call this function to enable it.

3.3.2 int pci_disable_pcie_error_reporting(struct pci_dev *dev);
pci_disable_pcie_error_reporting disables the device to send error
messages to root port when an error is detected.

3.3.3 int pci_cleanup_aer_uncorrect_error_status(struct pci_dev *dev);
pci_cleanup_aer_uncorrect_error_status cleanups the uncorrectable
error status register.

3.4 Frequent Asked Questions

Q: What happens if a PCI Express device driver does not provide an

error recovery handler (pci_driver->err_handler is equal to NULL)?

A: The devices attached with the driver won't be recovered. If the error is fatal, kernel will print out warning messages. Please refer to section 3 for more information.

Q: What happens if an upstream port service driver does not provide callback reset_link?

A: Fatal error recovery will fail if the errors are reported by the upstream ports who are attached by the service driver.

Q: How does this infrastructure deal with driver that is not PCI Express aware?

A: This infrastructure calls the error callback functions of the driver when an error happens. But if the driver is not aware of PCI Express, the device might not report its own errors to root port.

Q: What modifications will that driver need to make it compatible with the PCI Express AER Root driver?

A: It could call the helper functions to enable AER in devices and cleanup uncorrectable status register. Pls. refer to section 3.3.

4. Software error injection

Debugging PCIe AER error recovery code is quite difficult because it is hard to trigger real hardware errors. Software based error injection can be used to fake various kinds of PCIe errors.

First you should enable PCIe AER software error injection in kernel configuration, that is, following item should be in your .config.

CONFIG_PCIEAER_INJECT=y or CONFIG_PCIEAER_INJECT=m

After reboot with new kernel or insert the module, a device file named /dev/aer_inject should be created.

Then, you need a user space tool named aer-inject, which can be gotten from:

    http://www.kernel.org/pub/linux/utils/pci/aer-inject/

More information about aer-inject can be found in the document comes with its source code.