Using physical DMA provided by OHCI-1394 FireWire controllers for debugging

Introduction

Basically all FireWire controllers which are in use today are compliant to the OHCI-1394 specification which defines the controller to be a PCI bus master which uses DMA to offload data transfers from the CPU and has a "Physical Response Unit" which executes specific requests by employing PCI-Bus master DMA after applying filters defined by the OHCI-1394 driver.

Once properly configured, remote machines can send these requests to ask the OHCI-1394 controller to perform read and write requests on physical system memory and, for read requests, send the result of the physical memory read back to the requester.

With that, it is possible to debug issues by reading interesting memory locations such as buffers like the printk buffer or the process table.

Retrieving a full system memory dump is also possible over the FireWire, using data transfer rates in the order of 10MB/s or more.

With most FireWire controllers, memory access is limited to the low 4 GB of physical address space. This can be a problem on IA64 machines where memory is located mostly above that limit, but it is rarely a problem on more common hardware such as x86, x86-64 and PowerPC.

At least LSI FW643e and FW643e2 controllers are known to support access to physical addresses above 4 GB, but this feature is currently not enabled by Linux.

Together with a early initialization of the OHCI-1394 controller for debugging, this facility proved most useful for examining long debugs logs in the printk buffer on to debug early boot problems in areas like ACPI where the system fails to boot and other means for debugging (serial port) are either not available (notebooks) or too slow for extensive debug information (like ACPI).

Drivers

The firewire-ohci driver in drivers/firewire uses filtered physical DMA by default, which is more secure but not suitable for remote debugging. Pass the remote_dma=1 parameter to the driver to get unfiltered physical DMA.

Because the firewire-ohci driver depends on the PCI enumeration to be completed, an initialization routine which runs pretty early has been implemented for x86. This routine runs long before console_init() can be called, i.e. before the printk buffer appears on the console.

To activate it, enable CONFIG_PROVIDE_OHCI1394_DMA_INIT (Kernel hacking menu: Remote debugging over FireWire early on boot) and pass the parameter "ohci1394_dma=early" to the recompiled kernel on boot.

Tools

firescope - Originally developed by Benjamin Herrenschmidt, Andi Kleen ported it from PowerPC to x86 and x86_64 and added functionality, firescope can now be used to view the printk buffer of a remote machine, even with live update.

Bernhard Kaindl enhanced firescope to support accessing 64-bit machines from 32-bit firescope and vice versa: - http://v3.sk/~lkundrak/firescope/

and he implemented fast system dump (alpha version - read README.txt): - http://halobates.de/firewire/firedump-0.1.tar.bz2

There is also a gdb proxy for firewire which allows to use gdb to access data which can be referenced from symbols found by gdb in vmlinux: - http://halobates.de/firewire/fireproxy-0.33.tar.bz2

The latest version of this gdb proxy (fireproxy-0.34) can communicate (not yet stable) with kgdb over an memory-based communication module (kgdbom).

Getting Started

The OHCI-1394 specification regulates that the OHCI-1394 controller must disable all physical DMA on each bus reset.

This means that if you want to debug an issue in a system state where interrupts are disabled and where no polling of the OHCI-1394 controller for bus resets takes place, you have to establish any FireWire cable connections and fully initialize all FireWire hardware __before__ the system enters such state.

Step-by-step instructions for using firescope with early OHCI initialization:

1. Verify that your hardware is supported:

   Load the firewire-ohci module and check your kernel logs. You should see a line similar to:

   ```
   firewire_ohci 0000:15:00.1: added OHCI v1.0 device as card 2, 4 IR + 4 IT
   ... contexts, quirks 0x11
   ```

when loading the driver. If you have no supported controller, many PCI, CardBus and even some Express cards which are fully compliant to OHCI-1394 specification are available. If it requires no driver for Windows operating systems, it most likely is. Only specialized shops have cards which are not compliant, they are based on TI PCILynx chips and require drivers for Windows operating systems.

The mentioned kernel log message contains the string "physUB" if the controller implements a writable Physical Upper Bound register. This is required for physical DMA above 4 GB (but not utilized by Linux yet).

2. Establish a working FireWire cable connection:

   Any FireWire cable, as long at it provides electrically and mechanically stable connection and has matching connectors (there are small 4-pin and large 6-pin FireWire ports) will do.

   If an driver is running on both machines you should see a line like:

   ```
   firewire_core 0000:15:00.1: created device fw1: GUID 00061b0020105917, S400
   ```

   on both machines in the kernel log when the cable is plugged in and connects the two machines.

3. Test physical DMA using firescope:

   On the debug host, make sure that /dev/fw* is accessible, then start firescope:

   ```
   $ firescope
   Port 0 (/dev/fw1) opened, 2 nodes detected

   FireScope
   ---------
   Target : <unspecified>
   Gen    : 1
   [Ctrl-T] choose target
   [Ctrl-H] this menu
   [Ctrl-Q] quit
   ```

   ```
   ------> Press Ctrl-T now, the output should be similar to:
   ```

   ```
   2 nodes available, local node is: 0
    0: ffc0, uuid: 00000000 00000000 [LOCAL]
    1: ffc1, uuid: 00279000 ba4bb801
   ```

   Besides the [LOCAL] node, it must show another node without error message.

4. Prepare for debugging with early OHCI-1394 initialization:

   4.1) Kernel compilation and installation on debug target

   Compile the kernel to be debugged with CONFIG_PROVIDE_OHCI1394_DMA_INIT (Kernel hacking: Provide code for enabling DMA over FireWire early on boot) enabled and install it on the machine to be debugged (debug target).

   4.2) Transfer the System.map of the debugged kernel to the debug host

   Copy the System.map of the kernel be debugged to the debug host (the host which is connected to the debugged machine over the FireWire cable).

5. Retrieving the printk buffer contents:

   With the FireWire cable connected, the OHCI-1394 driver on the debugging host loaded, reboot the debugged machine, booting the kernel which has CONFIG_PROVIDE_OHCI1394_DMA_INIT enabled, with the option ohci1394_dma=early.

   Then, on the debugging host, run firescope, for example by using -A:

   ```
   firescope -A System.map-of-debug-target-kernel
   ```

   Note: -A automatically attaches to the first non-local node. It only works reliably if only connected two machines are connected using FireWire.

   After having attached to the debug target, press Ctrl-D to view the complete printk buffer or Ctrl-U to enter auto update mode and get an updated live view of recent kernel messages logged on the debug target.

   Call "firescope -h" to get more information on firescope's options.

Notes

Documentation and specifications: http://halobates.de/firewire/

FireWire is a trademark of Apple Inc. - for more information please refer to: https://en.wikipedia.org/wiki/FireWire