DMA Multiplexing for Firewire Isochronous Receive

Overview

Unibrain, being the leader in Firewire software for the Windows platform, is pleased to announce that in its latest release of the ubCore Firewire driver suite, version 5.50, a crucial capability has been added to its arsenal of solutions.

Unibrain's ubCore drivers now implement the feature known as *DMA Multiplexing* for isochronous receive operations, or *MultiDMA* for short. This is the ability to receive multiple isochronous channels using the same DMA context at the OHCI level.

The OHCI specification requires that an isochronous DMA context must be allocated in order to perform operations on an isochronous channel. Each isochronous receive DMA context can only be programmed to receive a single isochronous channel number.
Although the OHCI specification allows implementations to support up to 32 isochronous transmit DMA contexts and 32 isochronous receive DMA contexts, most 1394 chips typically support 4 isochronous receive contexts and 8 for isochronous transmit. There are several implementations that support 8 isochronous receive contexts, but the majority supports only 4.

This limitation is crucial on solutions that need to operate more than 4 isochronous channels on the same bus, often leading vendors to install multiple Firewire adapters on a PC just so that they can have extra DMA contexts available.

However, the OHCI specification allows for exactly one isochronous receive DMA context to be configured as "shared", that is to be set up in a way that it may receive isochronous packets from multiple channel numbers. This way the software that controls the OHCI chip can overcome the limitation imposed by the number of available isochronous receive DMA contexts.

Unibrain Solution

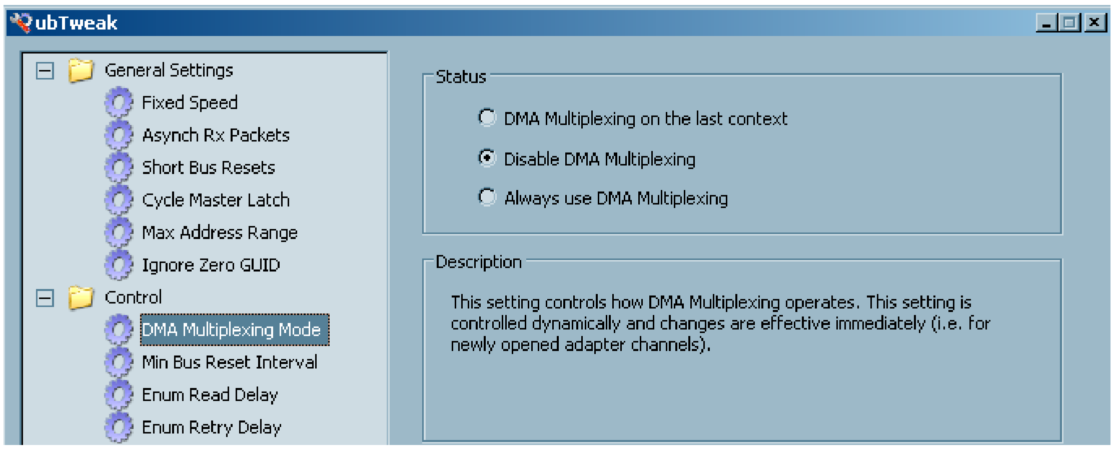
Programming a DMA context in this "shared" mode is significantly more complex compared to the "dedicated" DMA contexts, requiring delicate and timely handling, which is the main reason why software vendors had not implemented this feature before.

Unibrain has finally made this feature available, in an absolutely transparent way for applications. There is no special code required for an isochronous receive operation to run on the "shared" MultiDMA context. The binary of the application will run in exactly the same manner, regardless of whether it is running on a "dedicated" DMA context or on the "shared" MultiDMA context.

Still the application developer can have control of where the isochronous receive operations execute and thus configure the solution as desired.

In order to make ubCore 100% backwards compatible and not break existing code and running systems, it is possible to operate the Firewire adapter in the "Only Dedicated DMA context" mode, in fact this is the default mode of isochronous receive operation.

The Isochronous Receive DMA mode is now a configuration setting of ubCore. Actually it is not only a registry setting that determines how the system operates at startup, but can also be changed dynamically for easier testing.
The setting is actually named **DMA Multiplexing Mode**, as shown below in the ubTweak utility:



Status:
- DMA Multiplexing on the last context
- [✓ Disable DMA Multiplexing]
- Always use DMA Multiplexing

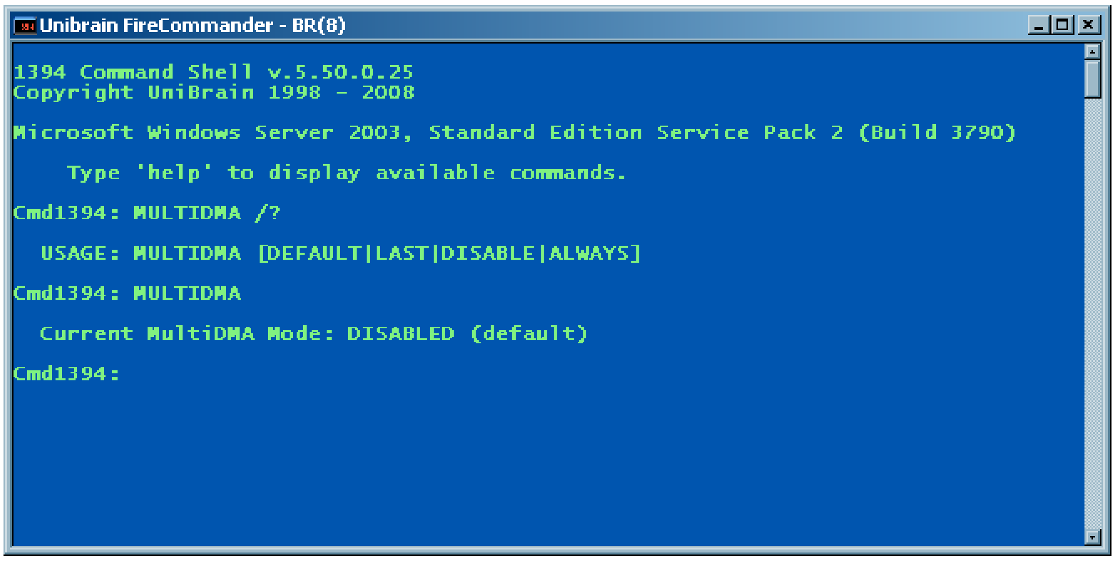
Description: This setting controls how DMA Multiplexing operates. This setting is controlled dynamically and changes are effective immediately (i.e. for newly opened adapter channels).

- **Always use DMA Multiplexing**: When this option is selected, then all isochronous receive operations get executed on the "shared" DMA context, leaving the rest of the isochronous receive DMA contexts inactive.
  This option has been added primarily for testing reasons. It allows application designers to easily stress test the DMA Multiplexing implementation of both the software and the underlying 1394 adapter.
- **Disable DMA Multiplexing**: This is the default option after the installation of ubCore 5.50, for reasons of backwards compatibility.
  When this option is active then all isochronous DMA contexts are operated in "dedicated" mode.
- **DMA Multiplexing on the last context**: When this option is selected then the operation of MultiDMA is enabled. The option name actually describes the internal logic of isochronous receive DMA programming.
  When there are more than one available isochronous receive DMA contexts then a newly opened *isochronous adapter channel* is operated in "dedicated" mode. When there is only one available, the last free one, then the DMA context is operated in "shared" mode.
  This of course means that if there are 4 isochronous receive DMA contexts available and the application sets up isochronous receive on 4 channels, the fourth will be running on "shared" mode, even though it will be the only one sharing the DMA context.
  This is implemented this way because it is technically impossible to shift a DMA context from "dedicated" to "shared" without disrupting isochronous receive on the context, which would result in at least one failed isochronous operation for an application, simply because another application tried to set up its own isochronous receive operations.
  In practice however, solution designers usually know the number of isochronous operations that will be running at one time, and if they require a maximum of 4 then they can simply disable MultiDMA.

Changing the DMA Multiplexing Mode setting from ubTweak saves the new value in the registry and optionally immediately applies it.

You can also use the **MULTIDMA** command in FireCommander to see and dynamically change the current setting, without saving it in the registry.

Type **MULTIDMA /?** to see the supported options as shown below:



```
1394 Command Shell v.5.50.0.25
Copyright UniBrain 1998 - 2008

Microsoft Windows Server 2003, Standard Edition Service Pack 2 (Build 3790)

    Type 'help' to display available commands.

Cmd1394: MULTIDMA /?

  USAGE: MULTIDMA [DEFAULT|LAST|DISABLE|ALWAYS]

Cmd1394: MULTIDMA

  Current MultiDMA Mode: DISABLED (default)

Cmd1394:
```

Entering **MULTIDMA** without parameters displays the current setting.

The DMA Multiplexing Mode value **can only be dynamically changed** when there are no isochronous receive adapter channels open. Doing a dynamic change of the operating mode is supported mainly for testing reasons, but could theoretically be utilized in some specialized scenarios as well.

Performance Considerations

There are mainly two performance issues to consider with regards to DMA Multiplexing:

- Can the 1394 chip handle heavy loads properly?
- Is there extra CPU overhead when a DMA context is shared?

DMA Multiplexing Operation under Heavy Loads

Our experimentation with many different Firewire adapters has shown that all of them can handle the maximum isochronous load (100% utilization of available isochronous bandwidth) with **no errors** (FIFO overruns, bad CRCs, etc).

Isochronous data reception works perfectly even if the MULTIDMA mode is set to ALWAYS, and the full isochronous load is handled by the “shared” DMA context.

Needless to say, it is not possible for Unibrain to test the hardware-level performance of all Firewire adapters available in the market. Solution designers should, as part of their hardware evaluation and selection process, test the performance of DMA Multiplexing on the actual hardware they consider using.

CPU Overhead

When the isochronous receive DMA context is operating in “shared” mode it is potentially receiving multiple isochronous packets per cycle, from different channel numbers. It is impossible for the DMA context or the code that controls it to predict the sequence that the packets will appear in so that the DMA context could be preprogrammed to copy the packets into their target buffers.

This means that all isochronous packets received on the “shared” context are received into an intermediate memory buffer.

The driver code that handles the hardware interrupt for isochronous receive, “parses” i.e. walks through this intermediate buffer and based on the packet header copies the isochronous data at their target buffer. So all data being received through the “shared” context are being memory copied by the CPU (DMA is used for getting the data from the Firewire adapter into the intermediate buffer).

This extra CPU overhead is a consideration that must be evaluated by solution designers when considering utilizing DMA Multiplexing versus additional Firewire adapters.

Unibrain measurements indicate that on modern PCs the overhead ranges between 1% and 2% which is almost negligible, but definitely non zero. On older PCs it may get up to 5% or 6%.

For this reason it is suggested that solution designers implement their solutions in such a way that the isochronous channels with the biggest payloads operate on “dedicated” DMA contexts, while smaller payloads run on the “shared” context. This way the amount of data being copied is minimized.

Asynchronous Streams

From the point of view of the OHCI chip, incoming asynchronous stream packets are not different from isochronous stream packets. This means that in order to receive asynchronous stream packets an isochronous receive DMA context has to be used. With a limited number of isochronous receive DMA contexts on the OHCI chip, this issue can be a major headache when there is a mix of asynchronous stream and isochronous stream traffic.

Putting all the asynchronous streams on the “shared” DMA context lets designers overcome this problem thus gaining significant flexibility in utilizing all the features of Firewire.

Conclusions

DMA Multiplexing is a very important capability that was missing from all Firewire software. With this new option at their disposal, solution designers and implementers that use the Unibrain drivers can take more fully advantage of the capabilities of Firewire hardware.