FFDShow – *Intel* *QuickSync Decoder*

This FFDShow build is aimed at harnessing the power of HW accelerated decoding by abstracting the HW acceleration and mimicking the behavior of a SW decoder with the hassle of DXVA.

Target audiences are battery powered platforms such as Laptops, Ultrabooks and tablets as well as HTPCs. Video power users can harness the QuickSync decode technology to accelerate their custom playback or transcoding pipelines by using the power of all the GPUs as well as the CPU.

The **Intel *QuickSync*** technology enables HW accelerated decoding of H264, MPEG2 and VC1; encoding of H264, MPEG2 (no VC1 ☹) as well as high quality video processing.

The FFDShow – QuickSync project currently uses the decoding engine.

# Main Features

* Wicked fast and low power HW accelerated decoding using Intel’s QuickSync technologies.
* Abstracts HW acceleration trickery – used like a standard SW decoder.
* Time stamp correction.
* Inverse telecine.
* Multi GPU or hybrid GPU support (Intel HW decoder + discrete renderer)
* Very low overhead on top of pure DXVA solutions.
* Open source and free for all (BSD license for the QuickSync decoder and GPL for FFDShow).

# Minimum HW Requirements:

* **2nd Generation Core i3-i5-i7** (codename SandyBridge) or newer.
* Intel Processor graphics - Intel HD Graphics 2000/3000 (GT1/2). Intel GPU must be enabled in BIOS and a driver must be installed. Older platforms are not supported. Current and future Intel platforms without an active Intel GPU are not supported (e.g. Xeon CPUs). Atom processors are also not supported. Multi GPU setup is supported. See section below.

# Minimum SW & OS Requirements:

* Windows 7 or newer. 32 or 64 bit. Should work on Vista 32/64 as well, but not officially supported or tested. Windows XP and older operating systems are not supported and will not be supported.
* Recent Intel HD Graphics driver is recommended (tested on 2538, 2509 & 2372).

# Updates, feature requests, bug reports and just saying hello are done in these forums:

[Doom9 thread](http://forum.doom9.org/showthread.php?t=162442)

[AVS Forum thread](http://www.avsforum.com/avs-vb/showthread.php?t=1358343)

# Thanks to the people who helped.

Special thanks to the doom9/avsforum users CruNcher and nevcairiel. These guys donated a lot of their time and knowledge in the various forums.

Blight (ZoomPlayer) for various tips and getting this project jump started publically.

The Intel Media SDK support team.

# Version history:

## 0.23 Beta

* Added multithreaded decoding.
* Optimized multithreaded code.
* Minor bug fixes.

## 0.22 Beta

* Reworked multithreaded code.
* Fixed dynamic aspect ratio change – wasn’t handled in previous releases.

## 0.21 Beta

* Added Multithreaded support: frames are copied on a secondary thread.

## 0.20 Alpha

* Fixed support for WMC full screen exclusive mode:
  + Works on multi GPU setup.
  + Thumbnail extraction is done in SW (via host DirectShow filter – e.g. FFDShow).

## 0.19 Alpha

* Added limited support for WMC full screen exclusive mode:
  + Renderer must be connected to the decoder directly - no intermediate filters.
  + Screen is connected to the Intel GPU (decoder shares device with renderer).
  + Might only work on a single monitor.
* Decoder has exposed its configuration options GetConfig/SetConfig - must be called before initialized.
* Padding the image to mod16 width is now off by default. Works with vobsub.
* Decoder can be tested for compatibly with media types via the TestMediaType method.
* Fixed leaving IVTC too soon on some cases.

## 0.18 Alpha

* Fixed FFDShow’s H264 sequence header parsing crash. A lot of users reported crashes with the last build. This was a long standing FFDShow issue that affected specific clips.
* Added black borders to images with non 16 modulo width. Retaining non standard width can cause downstream filters to crash (dvobsub/vsfilter).

## 0.17 Alpha

* Support variable frame rate video.
* More stable time stamps (audio sync issues).
* Fixed FFDShow’s frame rate measurement to better view frame rate changes.
* Better Media SDK initialization.

## 0.16 Alpha

* Support multi GPU setups. Now the decoder can run on separate HW then the renderer. Even without connecting the Intel GPU to a screen. See Multi GPU below for details.
* This version will be the first version on SourceForge.
* Updated to ffdshow build 3996.
* Some fixes to the timestamp code. Now supporting streams with no frame rate.
* Fixed several aspect ratio issues.
* Very initial support for DVD playback. Menus are not displayed right yet. WIP. Recommend not to use except for testing purposes.
* Changed mechanism for handling flush & seek event. Code is faster and more robust. A critical stage for playing DVDs.
* Added a new callback for FFDShow’s internal decoders – EndFlush. This is needed for DVD playback. Other decoders do not need to implement it.
* Various bug fixes.
* Added lots of trace prints for easy debugging (debug builds only).
* More internal documentation.
* Enhanced FFDShow’s code with a faster memcpy function (SSE2 based). This replaces calling memcpy. The original source code would use ffmpeg to do it, but it crashes on NV12 images.

## 0.15 Alpha

* Rewrote time stamp handling code. Decoder now calculates frame rate if missing, corrects for splitters reporting double frame rate for interlaced content. Handles PTS and DTS time stamps. Broken streams that alternate frequently between telecined and interlaced frames are not handles perfectly (yet!).
* Handled unsupported H264 formats by reverting to libavcodec silently within ffdshow. HW acceleration is limited to H264 simple, main and high profiles. Previous version would crash on unsupported formats.
* Added support for WMV3 (part of the VC1 HW decoder).
* Various bug fixes and better decoder error handling. As reported by various users for the 0.14 release.
* Cleaned up minor memory leaks.

## 0.14 Alpha

* Created ffdshow installer. Installer will default to enabling the Intel QuickSync decoder on new installations.
* More speed optimizations. CPU is at its lowest frequency during playback with very low utilization. 2-3% on desktop and 5-6% on mobile. Mobile lowest frequency is half of desktop (800/1600).
* Fixed handling of interlaced content which is encoded progressive frames.
* More robust and faster codec initialization

## 0.13 Alpha

* Optimized memory copy even further. Memory copy has 2-6% overhead (out of process CPU usage).
* Fixed bug in memory copy when frame width wasn't mod128.
* VC1 playback is more stable. Still corruption on some clips.
* Fixed some small memory leaks.
* Compatibility with 2509 driver.
* Bug fixes & cleanup.
* Tested with driver versions 2509 and 2372.

## 0.12 Alpha

* 64bit version is working.
* Optimized CPU usage (faster copying from GPU to CPU)
* More stable with LAV splitter. Previous version crashed on several MPEG2 transport with AVC1 (H264) video.
* Added time stamp stabilizing (transport stream issues).
* Added inverse telecine when the stream has the right flags.

## 0.11 Alpha

* Fixed skipping issues. Seeks are now instant.  
  Fixed handling of sequence headers for all supported formats. It also fixes image corruption in some clips.
* Created 64bit version. Released a non working version. Do not use.

## 0.1 Alpha

* Initial release

# Next steps or “what’s still not working ☺”:

* High CPU usage on several clips. Identified as partial HW acceleration due to frame size (width or height) or improper sequence header decoding. WIP to resolve most of the issues.
* EVR Custom Presenter (MPC-HC renderer) does not display all frames in interlaced content. Frame rate will be half than it should be. EVR is fine.
* 64 bit version under Lucid Virtu (v1.05 & v1.06) will produce choppy video with frames displayed out of order. Root caused to be a Virtu bug. Happens in other decoders as well.
* Investigate image corruption (rare) and crashes (very rare and only vc1) on various formats. Samples sent to Media SDK team for debug.
* Add HW accelerated video post processing.
* Test on IvyBridge.
* Test non-AVC1 H264 streams (not tested enough).
* Test with a switchable graphics enabled platform.

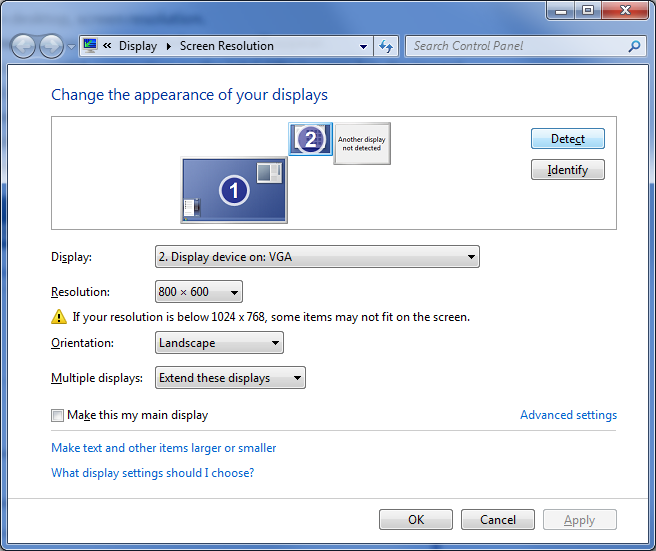
# Installation:

Execute the installer (32 and/or 64 bit) and select the options suited for you. On a clean install, the installer will ask your preferences regarding which codecs to enable. Intel QuickSync is enabled by default to the currently supported formats.

# Multi GPU Support

In multiple GPU systems, it’s possible to utilize the HW decoder of the Intel GPU one of the following ways:

* Install Lucid’s Virtu driver (see next section for more details). This will force the player to use only the Intel GPU (Decoder, video processing).
* Connect a screen to the Intel GPU. The HW decoder will use the Intel GPU and the renderer will use the GPU connected to the screen the video is viewed on.
* Cable free setup. Like with the cable setup, the renderer will use the GPU connected to the screen. Follow these steps:
  + Right click on desktop, screen resolution.
  + Click the **Detect** button. Unconnected screens will appear.
  + Extend desktop to a VGA connection on the Intel GPU (screen 2 in the image).
  + Drag the 2nd screen to a corner of the primary screen so the mouse boundaries of the primary screen will remain (almost) the same.
  + Click OK/Apply.
  + A reboot is recommended.



All the above methods produce a similar effect; they cause DirectX (and thus DXVA) to enumerate the disconnected Intel GPU. Without DirectX enumerating the GPU, initialization will fail and FFDshow will revert to use libavcodec. Unfortunately, it seems there’s no programmatic way to achieve this.

# Lucid Virtu

Lucid Virtu lets you play games or run CAD applications on a discrete and usually much stronger GPU and run selected applications on the processor graphics. The default GPU where all applications run by default is the GPU connected to the main display. Note that currently the 64bit version isn't 100 functional and frames might play out of order as well as produce heap corruption (non fatal) when opening or closing the DXVA/Direct3D environment.

* For peak performance in games, connect the display to the discrete GPU. You’ll need to add the video player to the list of application that uses the secondary GPU – the Intel processor graphics.
* For power savings, connect the display to the processor graphics. Video players are automatically enabled with **QuickSync**.

# License

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of INTEL CORPORATION nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.