AJA VIDEO

Programmer's Guide to the NTV-2 Software Development Kit

The NTV-2 SDK provides tools, documentation, and guidelines for writing applications for NTV-2 series PCI cards. NTV-2 is the term used to describe the 2nd generation NTV cards. These include the KSD, KHD, and HD-NTV. The HD-NTV card is similar in design and function to the KSD/KHD, and uses a subset of the NTV-2 SDK functionality. If you have installed the SDK for one of the NTV-2 cards, there is no need to re-install it for the others. The SDK's are identical. (You may need 3 separate SDK CD's, simply for installing the drivers, Control Panels, and utility programs - but you don't need to re-install the SDK itself each time.)

There is also a separate SDK available for Linux developers.

There is no Macintosh-based SDK. The Kona-SD and Kona-HD cards are almost identical to the KSD and KHD cards, but are bundled with Black Magic Designs QuickTime software for the Macintosh.

This document describes version 2.2 of the Windows SDK. Version 2.0 was the fist version which supports the KSD board. Version 2.1 supports simultaneous use of multiple DMA engines, and will run safely on multi-processor PC's. Version 2.2 includes some Control Panel improvements, audio output to the PC's sound-card (in Ntv2Demo), and some bug fixes.

SDK Contents

NTV-2 User Software (driver and ControlPanel) is installed from the Installer on the SDK CD. NTV2UserGuide.pdf is in the root directory on the CD. This document (NTV2SDKGuide.pdf), NTV2.pdf, and HDNTV.pdf are unzipped to your hard drive during SDK install. The "API and Samples" directory, Watcher, and DebugView are also unzipped to your hard drive during SDK install.

- **NTV-2 User Software**

  Drivers for Windows NT and Windows 2000/XP, plus the KSD and KHD (or HDNTV) Control Panel applications, and Photo-Shop plug-ins.

  There are two drivers (free build and checked build) provided for each NT and Windows Plug-and-Play systems (2000/XP). You may want to distribute the "free" (release) build to end-users. You will want to run the "checked" (debug) build on your development system, to enable debug messages from the driver (see DebugView). The "free" builds are provided in the board-specific "NT4 Driver" and "Win2K Driver"

folders on the CD, while the “checked” builds are in the “NTV2SDK\NT4 checked Driver” and “NTV2SDK\Win2k checked Driver” folders on your hard drive (where ‘NTV2SDK’ is the folder you unzipped the SDK files to during SDK installation).

The Control Panel is a Windows Tray Application which provides a basic interface for the NTV2 card. It is recommended that the Control Panel be run automatically at system start-up, and left running always. The Control Panel provides a consistent method of initializing the NTV2 card and allowing access to basic functionality. For KSD and KHD boards, the Control Panel automatically downloads a *.bit* file to the Xilinx FPGA. The *sd_top.bit* (for KSD) or *hd_top.bit* file (for KHD) residing in the same directory as the installed ControlPanel .exe file is automatically downloaded when the ControlPanel is started. (The Control Panel installation procedure puts the *.bit* file into the proper place.)

It is possible to build your application so that it does not depend on running the Control Panel, but you must download the Xilinx file yourself, and be sure to perform all necessary initializations for the card.

- **NTV-2 Installation Guide (NTV2InstallationGuide.pdf)**

  Documents general installation of the NTV-2 cards and software.

- **This document (NTV2SDKGuide.pdf)**
- **NTV2-Series PCI Video I/O Cards hardware interface document(NTV2.pdf)**

  Register definitions and detailed descriptions of the hardware-level interface to the KSD and KHD cards.

- **HDNTV PCI Video I/O Cards hardware interface document(HDNTV.pdf)**

  Register definitions and detailed descriptions of the hardware-level interface to the HDNTV card which has a slightly different feature set.

- **SDK API and Sample Projects (C++ Workspace)**

  The NTV-2 Application Programming Interface (API) is a collection of software Classes to help in developing NTV-2 applications. (See the NTV2 API section, below.) The Sample Projects included in the NTV2SDK workspace provide examples using the Classes API. (See the SDK Sample Projects section, below.)

- **NTV-2 Register Watcher**

  KSDWatcher.exe (KHDWatcher.exe for KHD, and HDWatcher.exe for HD-NTV). Program that monitors and/or sets registers.

  The Watcher provides direct access to the NTV2 Register set. The registers are listed in order, as defined in NTV2.pdf. You can change the Register contents by typing into the edit-box to the left of each Register name. The static-text box to the right of each Register name provides a real-time display of the Register values. Note, however, that when the watcher is not the Active Window, values for some of the Registers may become ‘stale’. Clicking on the Watcher window will update all Register values.

  If there are two or more identical boards installed in the system, the Watcher will default to monitoring Board 0. You can specify the board number with a command-line argument. (For example: “KSDWatcher 0”, “KSDWatcher 1”, etc.)

It is possible, with the Watcher, to set invalid Register values which would cause the card to stop working correctly, or cause the system to crash entirely. In particular, the Global Control Register must specify a Video Format which is valid for the card, and you must not set the 'GO' bits in the DMA Control Register without first setting up the Scatter/Gather list properly.

The 'Restart Driver' function in the Watcher does not work under Windows 2000 or XP. It will work under Windows NT, if there are no CNTV2Card objects open.

Note that, when the Watcher is running, a lot of Debug messages are generated. It is sometimes useful to shut down the Watcher program when looking at the Debug message output.

- **DebugView**

Two tools you can use to display the debug messages from the driver are DebugView (from SysInternals) and DebugMon (from OSR). DebugView can be downloaded from the SysInternals website (www.sysinternals.com). DebugMon can be downloaded from the OSR website (www.osr.com). Shortcuts to both downloads are provided on the SDK CD, in the DbgPrint folder.

SDK Software Environment

The SDK provides a set of C++ classes for interfacing to the KSD and KHD cards, along with supporting tools and libraries. The SDK is provided with ".dsw" and ".dsp" files for use with Microsoft Visual C++. **All sample projects are accessed under the "NTV2SDK" workspace,** and the active project is selected under the **Project** menu, **Set Active Project.**. The libraries may not be linked properly if you open the .dsp file by itself. **Always work under the NTV2SDK workspace when using the sample projects.**

The Microsoft Platform SDK is required to build WinGraphLib projects such as Ntv2Demo. Make sure you have the Platform SDK installed.

NTV2 API

The purpose of the SDK is to provide a simple programming interface to the NTV-2 cards. The 'Classes' library provides the C++ class "CNTV2Card", which has member functions to perform all common functions required of the card. In addition, 'paintlib' and 'WinGraphLib' provide useful functions related to the card's intended usage. These libraries are used in some of the Sample Projects.

- **Classes**

Useful C++ classes to control the NTV2 boards.

- **CNTV2Card**

Here are several of the basic categories of functions provided in the CNTV2Card class:

Open/Close

Open must be called to initialize the class. Close may be called explicitly, or will be called automatically when the class goes out of scope.

Read/Write Frame Buffers

DMA functions for accessing Frame Buffers

Get/Set Parameters

Provide a high-level abstraction of parameters controlled by the Registers.

Read/Write Registers

Low-level direct control of the Registers

Mapping to User Space

Set up an interfaces through the Windows OS to the Frame Buffers (through PIO) and Registers.

Interrupt Control

Enable or Disable particular interrupts.

Subscribing to Interesting Events

You may subscribe to be notified when any of these events occur: each type of interrupt; or, any Register change. Subscribed events notifications are received via the WaitFor... set of functions.

Waiting For Events

This is the method used to synchronize with Subscribed Events. The calling thread will be blocked until the Event happens. Note that the CPU is not being used during this wait, and other threads have full access to the CPU.

The combination of WaitForVerticalInterrupt() and checking the Field bit in the Status Register is encapsulated in WaitForFieldID().

Auto-Circulate

Using these methods, the output frame can be automatically incremented (resetting to frame zero after the last frame) each Vertical Interval.

- **CNTV2CaptureToDisk** - captures a stream of frames (to disk) using DMA (*not supported in HD-NTV*)
- **CNTV2PlaybackFromDisk** - inverse of CNTV2CaptureToDisk (*not supported in HD-NTV*)
- **CNTV2CaptureToWindow** - captures a stream of frames (to a window) using DMA (*not supported in HD-NTV*)
- **CNTV2AudioLoopback -** Loops 6 Channels of audio input to the audio output. (*not supported in HD-NTV*)
- **CNTV2XilinxProgram** - code for programming large Virtex Xilinx (*not supported in HD-NTV*)
- **CNTV2WavePlay** - Plays a ripped(from CD,16 bit stereo) wave file. (*not supported in HD-NTV*)
- **CNTV2WavePlayThread** - allows wave file(from CNTV2WavePlay) to be played in a separate thread. (*not supported in HD-NTV*)
- **CNTV2Capture** - Captures to a file or the Clipboard...stills only
- **CNTV2ScanConvert** - Transfers clipboard bitmap and graphics file data to the framestore..stills only.

- **CNTV2Draw** - Supports drawing lines and ellipses to the board.
- **CNTV2Pan** - not supported by KSD or KHD(at this time). On HD-NTV, turns frame buffer into 1 big frame buffer to support rolls and crawls.
- **CNTV2Status** - reports input status as well as version status.
- **CNTV2TestPattern** - for rendering testpatterns.
- **CNTV2VidProc** - to setup video processing on the board
- **CPerformanceCounter** - utility to do some performance analysis
- **CRegistry** - interfaces to registry
- **ActiveObject** - wrapper for windows threads(contains Thread class)
- **Event** - wrapper for windows event object
- **Lock** - wraps Mutex...which is actually a Critical Section wrapper
- **Mutex** - actually uses CriticalSection for better speed
- **Thread** - wrapper for CreateThread.
- **AtomicCounter** - wrapper for InterlockedIncrement

- **paintlib**

  Opensource graphics file encoding/decoding code. This is necessary for the Ntv2io and WinGraph Programs (Ntv2Demo and Pan) projects to build.

- **Includes**

  Include files for HD-NTV, KSD, and KHD programs.

- **WinGraphLib**

  A general-purpose class library with graphics support. An alternative to MFC.

- **ControlPanel Remote Control**

  A method of commanding the Control Panel to set Registers for you. This method is preferable to using the 'Classes' API directly to set the equivalent parameters, as it prevents conflicts between your Application and the Control Panel.

SDK Installation

Before building the SDK projects, the paintlib library must be built. Paintlib comes pre-built on the SDK CD. You may need to re-build paintlib for your OS, if you encounter any paintlib-related problems.

To build paintlib, open the workspace in the folder Ntv2Projects\paintlib\paint-lib\win\paintlib, set the active project to be 'piclook', and build both Release and Debug configurations. (Paintlib is useful for its public-domain file-type and format-conversion

routines. It is included as a seperate workspace on the SDK CD, and may also be acquired directly from http://www.paintlib.de/paintlib/)

After building paintlib, you should be able to build any of the sample projects.

SDK Sample Projects

The SDK includes sample projects demonstrating basic usage of the classes and libraries provided for interfacing to the card. **All sample projects are accessed under the "NTV2SDK" workspace. (In the "API and Samples\NTV2Projects\Ntv2Sdk" folder, open NTV2SDK.dsw.)** 'Simple' and 'Synchronize' projects demonstrate the basic functions of NTV-2 cards, and apply equally as well to the HD-NTV card. 'Ntv2Demo' demonstrates advanced features of the NTV-2 cards, and can not be used with the HD-NTV card.

- **DrawDemo**

  Very Simple Console application that demonstrates writing to the framebuffer.

- **Simple**

  The 'Simple' project is a Console application which demonstrates reading from the card and writing to the card using PIO mode.

- **Ntv2io**

  MFC project that demonstrates simple programming of the board.

  This Project is a demonstration project of how to control the HD-NTV card (or KSD or KHD) via your own software. The project has been kept very simple. There are project configurations to build to the KSD, KHD, or HD-NTV cards. For the KSD, the program builds KSDIO.exe.

  You need to build the paintlib project first, however. After this you can build the HDNTVIO.exe program. When the program starts up it determines the video standard (1080/720 or 525/625) used by the NTV2 Card and captures an image from the card and displays it in the SDI View window. Subsequent "New" commands from the menu will initiate a new capture from the board. An "Open" will open a graphics file(.bmp,.jpg.,.png and some others) and display the contents of the graphics file into the SDI window and also download that image to the NTV-2 board.

  The following are points to keep in mind when trying to decipher the code.

  1.This project uses the paintlib library to decode/encode various file formats. You must build this library before building the NTV2IO Program.

  2.The main class to control the HD-NTV card is CNTV2Card found in the files ntv2card.(h,cpp). To understand this file you need NTV2.pdf. This is a very simple class which encapsulates the DeviceIOControls.

  Other classes derived from CNTV2Card are:

- CNTV2ScanConvert(ntv2scanconvert.h) - decodes graphics files and downloads them to the HD-NTV card.

- CNTV2Capture(ntv2capture.h) - captures stills from HD-NTV and writes them to a graphics file or clipboard.

- CNTV2Status(ntv2status.h) - reports various status info from the card and driver.

- CNTV2TestPattern(ntv2testpattern.h) - writes test patterns to HD-NTV board.

- CNTV2VidProc(ntv2vidproc.h) - Performs mixes, wipes, splits and keys.

Other utility files that are useful:

- ntv2utils.cpp - Packing and Unpacking routines. When 10 YCbCr is the frame buffer format, the data is packed.

- transcode.cpp - transcode between YCbCr and RGB

- verticalfilter.cpp - used to filter to an interlaced format.

- resample - cubic resampling to stretch or compress for aspect ration conversion.

Includes files that are almost always included:

- AJATypes.h - For portability. Windows Applications need to define MSWindows=1 in their projects.

- NTV2Enums.h

- VideoDefines.h

- **RemoteControl**

The 'RemoteControl' project demonstrates remote control of the Control Panel application via the use of Windows Messages. By sending messages to the Control Panel to affect Register changes (instead of using equivalent CNTV2Card functions), you eliminate conflicts between the Control Panel and your application.

The features which can be controlled by remote messages to the Control Panel are:

Board Settings Dialog

Video Processing Dialog

Write Test Patterns

Export graphics images or testpattern data

Capture graphics stills

- **Synchronize**

The 'Synchronize' project demonstrates the use of interrupts.

This application is a simple MFC dialog-based application. The application simply counts vertical interrupts generated from the HD-NTV. It subscribes to Interrupt Events from the driver, and Waits on the Event. When the vertical interrupt is enabled, an Event is fired every Vertical Interrupt from the driver. The application Waits on this event, and updates the displayed count.

Using the Ntv2Card class function SubscribeOutputVerticalEvent will create an Event which the driver will then signal.

By using the CNTV2Card class function WaitForVerticalInterrupt, you can synchronize your code to Vertical Interrupt. This function encapsulates the Win32 API function WaitForSingleObject.

But note, Windows is not a real time operating system so depending on how well behaved your system is, the Event might not be signalled in a timely manner.

Also note that the Event is signalled every interrupt so that in an interlaced video format you will get an interrupt every field whereas you will be writing frames to the board.

To determine what field is currently being displayed after receiving an Event you need to read the status register which is documented in ntv2.pdf (or hdntv.pdf) as Register 21. If Bit 23 is '0' Field 0 is being output and if bit 23 is '1' then Field 1 is being output. Reading the status register is encapsulated in the ReadStatusRegister method in the CNTV2Card class.

WaitForFieldID() is used to call VerticalInterrupt() until the Status Register yields the requested FieldID.

Everything interesting in the source code for this project is in CSynchronizeDlg::OnStartcounting() in SynchronizeDlg.cpp.

All it does is put out TimeCode for 3 seconds. Then the performance counter spits out how long it took. Most often, it should spit out 3.00 seconds. It can be less depending on what field the counting started on. If it takes more than 3.00 seconds then the system is not well-behaved and vertical interrupts are being missed.

In addition, however, it also writes a frame of video to the board from the file "test.ntv" (IF the file exists on your hard drive). This will slow down the performance considerably, unless you have a really fast RAID system.

The "test.ntv" file can be created by unzipping "test.ntv.zip", or by running "makeframes.exe". This file only works with the HD-NTV and KHD however. And you will need a 100 mbytes/second RAID system to play it back smoothly.

- **Ntv2Demo**

The ‘Ntv2Demo’ project demonstrates the use of DMA and audio. It utilizes the WinGraphLib class library. For this you need the Platform SDK installed.

The Ntv2Demo project supports configurations for KSD and KHD only. There is no HDNTV configuration, because the project uses features not found on the HDNTV board.

Five modes are supported: ‘Capture To Window’ , ‘Capture To Disk’, ‘PlayBack From Disk’, ‘Playback Audio From WaveFile’, and ‘Loop Back Audio’.

All modes require considerable system resources. This program (and any program which attempts to process video in real-time) should be run as a ‘High Priority’ process, to try to keep other processes from hogging the CPU, which would prevent Ntv2Demo from keeping up with real-time.

To set the priority if a process, you can run Task Manager, right-click on the Process in the 'Processes' pane, and choose "Set Priority". To start a program as a High-Priority process, you can get a Command-Window and use the command START -HIGH (for example, START -HIGH KSDDEMO). (You can define a shortcut on the desktop that will automate this.)

'Capture To Window' mode should be usable on any PC, but needs a CPU/Graphics card combination capable of re-drawing the video in the window in real-time, or else Vertical Intervals will be missed, resulting in dropped frames to the output window. If a Vertical is missed, a message (specifying the number of Verticals missed) is printed to the NTV2Demo Log Window.

'Capture To Window' mode uses the NTV2 card's on-board DMA Engine to transfer data from the card to Host memory. The particular DMA Engine used is selectable in the 'Video DMA' menu.

'Capture To Disk' and 'PlayBack From Disk' modes will only be usable dependant on the speed and configuration of your PC's hard drive. A disk or RAID with 40MB/second(KSD) or 160MB/second(KHD) throughput is required. In addition, the write block size must be matched to the hard drive's sector size (see CNTV2CaptureToDisk::GetWriteSize()).

'Capture To Disk' and 'PlayBack From Disk' modes use the Disk Controller's DMA engine to transfer data between the disk and the NTV2 card. They do not use the NTV2 card's DMA engines.

The 'Loop Back Audio' function reads audio from the Audio Input Buffer,and simply writes that data into the Audio Output Buffer. If the process does not get enough CPU time to keep up with its task, the Audio Output Buffer may become empty. (It is the application's job to not let it become empty.) If the Audio Output Buffer becomes empty, the hardware will lose synchronization with the data, and will become unreliable. (Channel 1-2 data may be output on Channels 3-4, etc.) When this happens, the audio hardware will set the Audio State bits in the Audio Control register to 'Fifo Went Empty' (bits 11 and 12 set). Ntv2Demo monitors these bits, and Resets the Audio hardware if this failure is indicated. An error message is also written to the Log window.

'Loop Back Audio' mode uses the NTV2 card's on-board DMA Engines to transfer data from the card to Host memory, and from Host Memory to the card.. The particular DMA Engines used are selectable in the 'Audio DMA' menu.

- **pan**

The 'pan' project demonstrates the use of Panning as might be used by a Character Generator for "rolls" and "crawls". It maps 8 frames into one 'large frame', with the output frame being a selectable region within the 'large frame'.

The pan project supports configurations for HD-NTV only. There are no KSD or KHD configurations, because the project uses features found only on the HD-NTV board.

Panning transforms the HD-NTV framestore from 8 individual frames to 1 large 4096 pixel x 4096 line frame. The Pan Position determines where the hardware starts reading out from the framestore. So if the Pan Position is 0,0 then the upper left 1902x1080 pix-

els are read out in 1080 mode and the upper left 1280x720 pixels are read out in 720 mode. If a right or bottom boundary is met, then the output video wraps back to the left or top. Pan Mode only works correctly if the framebuffer is in RGBA mode.

This demo program can read in a .jpg file and converts it to a 4096x4096 image that fills the framebuffer for Pan Mode. Pressing the left mouse button on the image on the monitor moves the Pan position to that point in the image. Dragging the mouse over the image pans through the image.

The CNTV2Pan class in the classes directory is the demonstration class for this. It inherits from CNTV2Draw to allow easy drawing into the pan buffer.

For rolls and crawls to be smooth, you will need to synchronize the setting of the Pan Postion to the output vertical interrupt. The synchronize project in the SDK shows how this is done.

Creating a new project with the SDK

To create a new project using visual studio you simply use the wizard to start whatever kind of project you like.

Include in each file that will use the HD-NTV board:

(or add to stdafx.h)

```
#include "AJATypes.h"
#include "NTV2Enums.h"
#include "VideoDefines.h"
```

Then you add the following to the project settings

C/C++ Tab Preprocessor settings-> Preprocessor definitions

- Add MSWindows=1
- Add KHD if building for KHD
- Add KSD if building for KSD
- Add HDNTV if building for HD-NTV
- (Add only one of KHD, KSD, or HDNTV.)

C/C++ Tab Preprocessor settings-> Additional include directories

../Includes,../Classes,..\paintlib\paintlib\common,..\paintlib\paintlib\win\paintlib

Of course, you don't need to add the paintlib directories if you aren't using paintlib. If you are using paintlib you need to include the following libraries:

Link Tab -> Object/library modules

For Debug configurations:

..\PaintLib\lib\Debug\libtiff.lib ..\PaintLib\lib\Debug\libpng.lib ..\Paint-Lib\lib\Debug\libjpeg.lib ..\PaintLib\lib\Debug\paintlib.lib

For Release configurations:

..\PaintLib\lib\Release\libtiff.lib ..\PaintLib\lib\Release\libpng.lib ..\Paint-Lib\lib\Release\libjpeg.lib ..\PaintLib\lib\Release\paintlib.lib

As you add files from the Classes subdirectory you need to tell the compiler that it doesn't use precompiled header files.

For example if you add ntv2card.cpp to the project.

- Go to the C/C++ Tab Precompiled headers section, select ntv2card.cpp in the file pane, and choose "Not using precompiled headers'.

- Or add stdafx.h to top of .cpp files (to use precompiled headers).

SDK Visual Studio Workspace Tips

The NTV-2 SDK for Windows is provided with ".dsw" and ".dsp" files for use with Microsoft Visual C++. The projects are all included in one workspace (NTV2SDK.dsw), and the active project is selected under the **Project** menu, **Set Active Project**. Within each project, are multiple build configurations. The active configuration for a project is set in the **Build** menu, **Set Active Configuration**.

If you try to open the individual .dsp project files instead of using the NTV2DSK workspace, the project files will not build, because the *dependencies* (on Classes, WinGraphLib, etc.)are declared in the workspace file.

When working between multiple projects, it is important to always remember to set the 'active project' and 'active configuration'. Many operations, such as "Build", "Find in Files", and 'browse' commands (such as "Go to definition of..."), apply to the 'active project'.

Also note that, when the 'active project' is changed, Visual Studio may not necessarily update the 'active configuration' to a valid configuration for that project, and the configuration selection in the 'Build' toolbar may also show invalid configurations! It may be necessary, after changing the active project, to select **Build** menu, **Configurations** in order to refresh the configurations presented in the 'Build' toolbar.

With the ‘multiple projects, one workspace’ scheme, libraries (such as ‘**Classes**’ and ‘**WinGraphLib**’) are also built with multiple configurations, and the configuration for any libraries linked in will be the configuration matching the current active configuration for the active project. (Unless there is no matching configuration... in which case Visual Studio just picks one (the wrong one!) for you. There are matching library configurations for all project configurations in the SDK.) There are no such issues with the ‘**paintlib**’, since it is built seperately, outside of the ‘SDK’ workspace.

Restrictions

The Ntv2Demo sample project requires a hard-drive or RAID array with 40MB/second(KSD) or 160MB/second(KHD) throughput for Capture To Disk and Playback From Disk modes.

The NTV2 cards support an Auto-DMA mode for Audio, which lets the Audio hardware communicate with the DMA hardware to automatically keep a stream of data going into and/or out of the card. This mode is not supported by the 2.2 drivers.