ANALOG DEVICES

AN-526
APPLICATION NOTE

One Technology Way • P.O. Box 9106 • Norwood, MA 02062-9106 • 781/329-4700 • World Wide Web Site: http://www.analog.com

ADV601 Video Codec Frequently Asked Questions

by David Starr

IS THE ADV601 AVAILABLE?

Yes: It is in stock in the warehouse and at our distributors. Not all the distributor people know this yet, but we did ship a good deal of product to them. Prices are per 10,000-piece lots.

WHAT TECHNICAL INFORMATION IS AVAILABLE ON THE WORLD WIDE WEB?

Point your web browser to www.analog.com, the main Analog Devices page. Drop "ADV601" into the search engine to travel to the special ADV601 page. Here you can download the data sheet.

The Data Sheet

The ADV601 data sheet is available from the Norwood Literature Center in the United States (800-262-5643, press 2); in the UK (01932 266 000). It is also available on the ADV601 web page in Adobe Acrobat form.

The FTP Site

Find a hot-link (pointer) to the ftp site where the following ADV601 data is posted.

Application Notes

Ontheair.doc

Hints and kinks learned (while getting the Videolab board to run).

Dspbinwd.doc

Control ADV601 bit rate with an ADSP-21xx family DSP.

Binwidth.zip

Theory and practice of bit rate control and bin width calculation.

Schematics and Pal Code for the Videolab Evaluation Board

A 32-bit wide PCI bus design.

Software Simulator

A software-only demo in file icm.zip.

Download it, unzip it on a Windows® 95 machine and run the setup.exe. This will give you a nice software compression and decompression program that works on still images. It lets you compress as much as you like and shows a side by side, before and after to evaluate quality of the compressed image. Source code for the demo program is included. It is based on a Video-for-Windows software Codec compatible with standard Windows video programs such as Mediaplayer and Adobe Premiere. Use Wavelet image compression with your favorite Video-for-Windows tool.

Hardware diagnostic program 601test.

MS-DOS application ideal for debugging new hardware. Easily adapted to new hardware designs.

Windows 95 Drivers 601rpman and 601cman.

[We have] a Windows 95 Plug and Play interrupt service routine (601rpman.vxd) and a bit rate control program (601cman.dll). This is a matched set of low level drivers. Source code is posted.

Raw2avi file converter.

A Windows 95 tool that converts raw ADV601 video to the "audio-visual-interaction" .avi video acceptable to standard video-for-windows programs. Capture video with the Videolab board and edit with your favorite Windows application.

IS THERE AN EVALUATION BOARD?

Yes. Call Momentum Data Systems (714-557-6884) to place an order. In addition, a nonlinear editing board for the PC based on the ADV601 is available. Call David Brott at Quadrant International (610-251-9999 x215).

HOW DOES THE VIDEO LOOK AFTER COMPRESSION?

At 10:1 it looks as good as you get at home on good program material (say a ball game on a bright sunny afternoon). At 30:1 it looks as good as a VHS rental tape. At 200:1 you can still read the numbers on a football player's shirt. Download the software simulator from the ftp site and see for yourself.

WHAT HAPPENS WHEN I PUT THE VIDEOLAB CD-ROM INTO MY PC?

The CD-ROM Videolab 97 install smooths over the "low level" vs. "high-level" install confusion. Windows 95 distinguishes between "low-level" (hardware driver) software that needs a reboot afterwards, and high-level (.exe files) software that does not. Videolab contains both low level drivers and higher level application programs. A complete installation (or reinstallation of upgraded software) requires both a high level and a low level install. The CD-ROM automates this.

We care about this because it is easy to do the high level install and fail to do the low level install. Low level install (file complete.inf) is supposed to ask if you want to reboot, and you must say "YES" for the install to "take." Low level install causes the high level install (file setup.exe) to start right after the reboot. If by chance windows fails to offer a reboot, do it by hand. You have a better chance to kick off the low level install automatically, (Method 1) if you INSTALL THE HARDWARE FIRST. If this fails (incautious mouse clicking can cause Windows 95 to skip the low level install, or fail to find the CD-ROM drive) then, Method 2 ,"Add New Hardware," gives you a second chance.

Low level install is started one of three ways:

Windows 95 Detects the Hardware

You will see the "Windows has detected new hardware" message. Once Windows 95 sees the hardware, it remembers the card forever, and won't ask again. You have one only shot at #1, which explains why they have #2 and #3. In all three cases, ALWAYS hit the "have disk" tab. Otherwise, Windows 95 will try to find existing drivers on your hard drive. It sometimes finds old drivers, or drivers for some other product. You don't want that. Watch the activity LED on the CD-ROM drive. It ought to flash when Windows loads drivers from it. Never let Windows bypass the driver software installation because it will be very difficult to install the drivers later. Windows must offer a choice between "Complete.inf" and "sim_only.inf". Always select "complete.inf".

Click On "Add New Hardware" in the Control Panel

Click On "Change Driver" in "Device Manager"

Clicking on the Start button will put you into device manager, then "Settings," then "System" (icon of computer with a blue screen) and then the Device Manager tab. Look for "Video Capture and Compression" device. If present, you can do "Change Driver." If not present, it means you never reached low level install. Try the "Add New Hardware" applet on the "Control Panel" instead.

How Do You Tell Which Install[s] Have Been Done?

The low level install is quick and almost invisible. It puts "ADV601 Video Lab Card" into device manager upon completion. If this is missing, a low level install is required. A yellow exclamation point (!) on the ADV601 Video Lab Card entry means the low level install ran, but there is a resource conflict. (See below.)

The high level install puts up a flashy blue screen with "ADV601 Videolab 97" in large white 20-point type, with shadows showing under the letters. It also puts the "Videolab" icon on the Start button's program menu and a shortcut "To the Videolab" on your desktop.

WHAT DO YELLOW EXCLAMATION POINTS IN DEVICE MANAGER MEAN?

Configuration Manager in Windows 95 tries to dole out limited hardware resources to a herd of picky plug and play cards. The four resources are:

1. Memory Addresses
2. IO Port Addresses
3. DMA Channels
4. Interrupt Request Lines (IRQs)

Video lab needs only some memory and ONE (1) IRQ. The average PC has lots of memory, but only 15 interrupt (IRQ) lines, so it's the IRQ's that run out. In Windows 95-speak this is called a "resource conflict." Device Manager will show a yellow exclamation point on the offending device[s] and the Properties Sheet will say "This device is causing a resource conflict," or "This device is not working," or other similar comments.

IRQ resource conflicts are fixed either by removing one of the conflictor's (IRQ hogs) from the machine or moving one of them to a spare IRQ. If you click on the "Computer" icon at the top of the Device manager tree you can see all the IRQ's. If 0 through 15 (all of them) are in use, something will have to be removed, you are flat out of IRQ's and there is no way to make any more. On my machine for example, I removed my parallel port (LPT1) to free up IRQ 7 and the 2nd IDE disk controller to free up IRQ 15.

The LPT port and the IDE disk controller are usually integrated into the motherboard and you "remove" them in software. If they are real plug-in cards, you can really remove them by pulling them out of the machine. In the software "removal" case, you have to get into the "CMOS Setup Routine," which is part of BIOS. On late model Gateway's, hold down the F1 key at power-up time. Other machines work about the same, but you hold down a different key. The "CMOS Setup Routine" is menu driven and you should be able to disable the unneeded peripherals.

Once you have some wiggle room (a free IRQ or two) start up Windows 95. Get into device manager (Click *Start*, Click *Settings*, Click *Control Panel*, Click *System*, Click on Device Manager Tab). Check the devices you just removed. If they still show in the Device Manager window, Device Manager was too smart for its own good. It "saw" the disabled device at boot time and loaded the driver for it, consuming the resource you were trying to

free up. Let Device Manager know who is in charge. "Remove" the device. He will whine about it, "Warning: You are about to remove this device from your system," but be firm with him and click OK.

Now, from the start button click on "Help." In the help menu click on the "Contents" tab, then Open the Troubleshooting book. Click on "If you have a hardware conflict." This starts the "Wizard," which gives you step-by-step coaching to move conflicting IRQs around until they no longer clash. IRQ assignments are changed on the "Resource" tab of the device properties sheet. You often have to change the "Setting Based On" box from "Basic Configuration 0001" to something else like, "Basic Configuration 0002," before Device Manager will let you make any changes.

Device Manager should offer to reboot when you are done. Take him up on this and reboot. If Device Manager forgets to ask you to reboot, do it anyhow to make the changes stick. Upon reboot, device manager may "discover" the hardware you just "removed" and will load the drivers you just removed. This doesn't hurt because the devices you care about (Videolab) have a priority on resources and newly discovered devices lose out. On some machines the unused parallel port shows up with a resource conflict exclamation point. (Last card into the machine is a loser). My sound card, fax modem and Video lab card show "OK" and work properly.

CAN I JUST COPY THE .EXE FILES TO C: AND RUN THE VIDEOLAB CARD?

No. Under Windows 95, no ordinary program (.exe file) is allowed to perform I/O instructions or change the memory management hardware of the CPU. The Videolab card is memory mapped and requires memory management changes before a program can "see" the card in address space.

Only specially constructed (privileged) programs ("vxd's" in Windows 95-speak) are allowed to do the heavy lifting necessary to operate the Videolab card. Videolab comes with such a vxd (601rpman.vxd), and it must be installed in a special way or it won't work. The "vxd" and Windows 95 have to be formally introduced to each other. The script for this introduction is a file on the install diskette named Complete.inf. The script says

1. Delete the old version of the vxd and the DLL's.
2. Make some entries in the Registry connecting hardware device 0601 to driver 601rpman.vxd.
3. Copy the new version onto hard disk from floppy disc.

This script runs when Windows 95 detects new hardware FOR THE FIRST TIME ONLY, or when you select "Add New Hardware" on the "Control Panel" from the "Setting" button on the Start Menu, or when you select the "Change Driver" button in Device manager.

In all cases, you want Windows 95 to get the driver and the install script from CD-ROM. Always choose the "Have Disk" option. Be careful that Windows 95 is loading new drivers from your floppy disk and not reloading the old drivers from hard disk. Windows 95 saves the install script (and perhaps the old drivers) in hidden locations on the hard drive.

After a driver install (using complete.inf), you must reboot Windows 95 before the new vxd becomes effective. The old vxd has been loaded into memory at boot time and remains in control until you reboot. Windows 95 SOMETIMES prompts you to reboot, and SOMETIMES fails to prompt you. ALWAYS REBOOT.

The complete.inf script ONLY loads the vxd onto your system. The application programs (601Test, SOFTVCR and the rest) are loaded by running setup.exe on the install diskette. The CD-ROM will automatically run setup.exe when you stick it in the drive if the "auto insert notification" feature of Windows 95 is active. Alternatively, you can start setup.exe from the "ADD/Remove Programs" applet in "Control Panel," or you can click on setup.exe in Explorer or you can Run D:setup.exe from the start button.

Bottom line. To run VideoLab on a new PC,

1. Get the latest Videolab Software Diskette. We improve the software day by day, so it pays to get the very best.
2. Power down the CPU, and insert the VideoLab card and power up.
3. Windows will report "New hardware detected" and ask for a diskette.
4. After the complete.inf script ends, run the setup.exe program from the delivery CD-ROM to install the application programs. After both the vxd and the application program installs are done, then reboot.

WHAT IS THE SOFTWARE SIMULATOR?

We have a software simulator customers can use to evaluate quality of the compressed image. It runs on Windows 95 machines and can be downloaded from the FTP site (file icm.zip).

Copy icm.zip onto your Windows 95 machine, decompress it with pkunzip and then execute the setup.exe program that comes out of the zip file. This will install the software ADV601 Codec and a test program that gives you a side by side comparison of any image before or after compression. It can handle windows .bmp files of the right size. You must have Windows 95 on your machine.

The simulator package consists of a Video-for-Windows compliant software Codec (adv601.dll) and a simple application program (icmapp.exe). You have to run the setup file to install the software Codec. Once you do this

the Codec is available to all Video-for-Windows applications such as Mediaplayer and Adobe Premiere. You can use the Codec to compress an .avi file and then play it back using the Videolab hardware.

WHAT IMAGE SIZES WILL THE VIDEOLAB SOFTWARE CODEC HANDLE?

The ADV601 hardware can only handle

Full Resolution	720 by 486
CIF	360 by 243
QCIF	180 by 122

We limited the software Codec to the sizes the hardware can handle. You have to find .bmp files of the right size, or use a graphics program like Adobe Premiere to "stretch" nonstandard images to one of the standard sizes.

WHY ISN'T "ANALOG DEVICES WAVELET COMPRESSION" IN THE SOFTWARE SIMULATOR EDIT MENU?

Your .bmp file is the wrong size. If it isn't one of the sizes above, the Wavelet Codec falls off the Codec pick list inside icmapp.exe.

HOW LONG IS VIDEO DELAYED GOING THROUGH THE ADV601?

The latency through the ADV601 is one field time (16 milliseconds for NTSC video and 20 milliseconds for PAL). The ADV601 has a one field buffer in the DRAM. The Wavelet transform output is buffered in the DRAM and statistics (sum of squares) are computed on the transformed image. Huffman coding does not start until the host has had time to update the Bin Width Registers to control the amount of compression that will occur. Statistics are ready one line time after v sync starts, and the host has the remaining 19 line times (about 1.2 ms) to compute and load new bin widths. The Huffman encoder then starts up on the Wavelet transformed video in the DRAM and the Wavelet transformer starts doing the next field. This is faster than the eye can see. Persistence of vision is such that the eye cannot see events in this short a time.

You have to push the next field into the part to force out the previous field. The transformed field does not come out of the DATA bus until the NEXT field starts pushing in the VDATA bus. Each field is buffered in DRAM after the Wavelet transform and before the run length and Huffman encoder. Buffering gives the host time to look at the statistics and set the quantizing before the encoding to control the bit rate. When capturing live video, you don't really care. You want to keep compressing until the camera man decides he has enough and takes his finger off the camera trigger. When using the part to compress and decompress captured video from disk for nonlinear editing, you just have to push an extra field into the part at the end of each clip to force out the last desired field.

To compress one still frame (two fields) you would have to push field one and two into the part. Compressed field 1 would come out as field 2 was going in. You would then have to push in one dummy field to force out compressed field 2.

WHAT KIND OF PC IS REQUIRED TO RUN THE VIDEOLAB CARD?

To run successfully, the ADV601 VideoLab board requires a computer with the following equipment. Anyone buying a new machine might want to make sure it at least has the following components.

1. Pentium® CPU 133 MHz
2. 16 Meg RAM
3. SCSI Hard drive. You want a gigabyte free for capturing video, so a 2-gigabyte drive is a good starting place.
4. PCI bus with one free bus mastering slot.
5. Windows 95
6. A CD-ROM to load the software.
7. Intel Triton Chipset. These are a pair of big quad flatpack chips on the motherboard, marked Intel PCI Chipset SB82471 and SB82371.

We have been using Gateway P5-133s with success. The Adaptec 2940 SCSI board works for us. This hardware will keep up with the video for a compression ratio as low as 8:1. Slower hard drives like IDE work, but you must compress harder to slow down the video rate to something the hard drive can handle.

HOW DO YOU USE THE FIFO?

Transfer data as long as FIFO Service Request (FIFO_SRQ) is active. Don't use FIFO_STOP to shut down transfers. It comes active too late. You will inevitably do one too many transfers because FIFO_STOP does not go active soon enough to stop the transfers in time.

Leave two words in the FIFO. If you completely drain the FIFO (read every word out of it) the part gets into trouble. For instance, if FIFO service request was programmed for half full, you might read 256 words out of the FIFO. Instead read 254 words, leaving two behind.

Or, transfer until FIFO service request goes away, which avoids the problem completely, since FIFO service request always goes away while 16 words or more are in the FIFO.

The FIFO_SRQ bit in the Interrupt Mask and Status register is no longer sticky. It will simply follow the HCD_FF_SRQ status signal from the Huffman FIFO. This implies that a read of the Interrupt Mask and Status register will NOT reset this bit. All other sticky bits will be reset as usual, but the FIFO_SRQ bit will not. Simply, it will always reflect the state of the HCD_FF_SRQ signal from the

Pentium is a registered trademark of Intel Corporation.

FIFO. Likewise, if the HCD_FF_SRQ signal goes away before a sticky register read, the FIFO_SRQ bit will also go away.

This change also has a corresponding effect on the ADV601 HIRQ* output. Since the srq bit is not reset by a sticky register read, HIRQ* WILL NOT NECESSARILY GO AWAY AFTER A STICKY REGISTER READ. If the HCD_FF_SRQ status signal from the FIFO remains high, then HIRQ* will stay asserted after a sticky register read, assuming that bit is not masked. Likewise, if the HCD_FF_SRQ status signal goes away, then HIRQ* will immediately go away (assuming no other interrupts are pending) without waiting for a sticky register read.

WHAT DOES "RAW2AVI" DO?

Raw2avi converts the raw video data captured by the ADV601 into the "audio-visual-interactive" .avi format required by Windows video programs like Mediaplayer or Adobe Premiere. The ADV601 capture driver doesn't bother to compute and insert the special windows start of field markers into the video data stream on the fly, hence the Raw2avi program. You must run Raw2avi on each video clip you capture to use that clip in windows programs. Basically, Raw2avi picks through the raw video file and inserts fancy frame headers between each video frame. Raw2avi is a bidirectional program. It can convert raw video files into .avi files and can do the reverse, take an .avi file and make it suitable for playback with the ADV601. You need this if you have edited the video as an .avi file and then want to display it with the Videolab board.

What Do the Statistics Mean?

Raw video data is checked for the proper kind and location of fields and sub-bands (Mallat blocks) of video within each field. Data errors or data loss during capture can result in missing escape sequences. The legal escape sequences (start of field start, of block, VITC, etc. are defined in Table XVI of the ADV601 data sheet. Raw2avi reads the entire captured video file (that's why he is sluggish) and counts all the escape sequences. If the start of field headers read "Field 1, Field 2, Field 1, Field 2, . . . all the way through the file, the field sequence is "Correct." If Raw2avi finds a pair of field 1 in a row, a pair of field 2 in a row, or a file that starts with field 2, the field sequence is "Reverse," which means the video file may not play back properly. Each field must contain 41 start-of-block headers. If one is missing, or extras are detected, the field is tagged "bad," a message box appears to this effect, and it will be edited out of the .avi file. Bad fields may not play back properly. Each escape sequence "leadin" (the 0xFFFFFFFF marker) must be followed by one of the codes listed in Table XVII of the data sheet. If an unknown code should appear, it is counted and displayed. There should be no unknown escape sequences.

While Raw2avi is running, it prints status. As of now you can see the following:
Number of field 1s seen
Number of field 2s seen
Number of bytes read from file
Number of bad fields
Number of Mallat blocks inside each field
Number of bytes in each field

Errors in the .601 file suggest that video data was lost somewhere between the ADV601 and the disk file.

Each frame has a field 1 and a field 2, so the field 1 and field 2 counts should match. If they don't, a field is missing. Each field should contain 42 Mallat blocks. The number of bytes per field is updated on each field as a measure of the compression of each field. In an ideal world, the bin width calculator would keep the bit rate constant and each compressed field would be the same size. In the real world, the bit rate will vary somewhat from field to field. The jumpiness of the bytes-per-field display is a measure of the bin width calculator's effectiveness.

Statistics displayed by Raw2avi are intended to warn of video capture problems. Raw2avi will always create an .avi file, even if the raw video has problems, to permit viewing of the video clip and perhaps learning why the clip is defective.

How Do I Know If Raw2avi Is Working?

Convert a .601 file to a .avi file. The resulting .avi file must be slightly longer than the input file. You can then convert the .avi file back to raw (.601 is the extension for a raw video file). The new raw file will exactly match the original. Use the DOS fc command to make sure. Naturally, the .601 file should be good, with no bad fields. If Raw2Avi finds bad fields it will edit them out of the .avi file, and the final .601 file will be shorter than the original.

HOW DO I RUN "PHILLIPS MODE" VIDEO INTO THE ADV601?

To run from composite sync, put the part into Multiplexed Phillips Video Mode via the mode control register video interface format bits. Timing diagram, Figure 27 in the data sheet, shows where H and V sync must occur. To elaborate, HSYNC must be good to one video clock, in other words it must come up every 720 pixels in NTSC or PAL. If it sometimes counts extra pixels or drops a pixel, the ADV601 will get confused. 10 ns Setup time and 3 ns hold time relative to video clock are required by Table VIII on page 49. Horizontal sync should only last one pixel time. Vertical sync should be asserted on the leading edge of HSYNC and remain asserted for at least five line times. The setup and hold times are relative to video clock and the same as for HSYNC. On encode, HSYNC and VSYNC "deasserted" mark pixel 0,0 of the field. HSYNC polarity is programmable via the PHSYNC bit in the mode control register, so HSYNC asserted

(Horizontal retrace time) can be a "go low" or a "go high," whichever is more convenient. The video area registers are "cropping" registers. Video outside the active video area is set to black (0), which encodes very compactly. They do not shift the image nor change its size. They just save bits that would otherwise be expended transmitting information of no interest. They affect only encode. They do not interact with HSYNC and VSYNC timing.

HOW DO I OPERATE THE ADV601 IN SLAVE PHILLIPS MODE?

Does the device have any real-world knowledge of the expected video format when operating in slave mode; i.e., in my application, the video input is not direct from a video device but does need to be processed in real time? To this end, I am generating logic to fake the Hsyncs and Vsyncs into the ADV601. In this case, do the syncs need to meet the "classical" format signal timing requirements or can the syncs just be toggled in the correct sequence with respect to each other (i.e., do they purely reset internal counters)?

You have to keep the VCLK rate pretty close to the numbers given in the data sheet, page 32. The part runs on VCLK and uses a pll to create DRAM timing, so you cannot just change VCLK. That is the reason for the 1 ns jitter specification on VCLK. You have to have 720 samples per line, no more, no less, or the part gets very confused and the video loses sync. You can fudge the horizontal blanking. Normal blanking is 138 pixels long.

You can make it a little longer without trouble. The part counts total pixels per line (active and inactive) with a counter that rolls over at 1024. As long as you keep the blanking interval less than 304 (= 1024–720) pixels the part should work well. There is some end of line processing, so I would not shorten the horizontal blanking interval by much. Also, we designed and tested the part on video running at the regular speed, so we think you are all set.

HOW LONG IS INTERRUPT LATENCY UNDER WINDOWS 95?

Occasionally interrupt latencies as bad as 127 μs have been seen. Normally, you see FIFO servicing (data transfer under program control) start 5 μs after the interrupt is asserted, and the FIFO stuffed full in 15 μs. Assume we are seeing the Windows 95 clock tick and scheduler interrupt service routine butting in ahead of us, at interrupt, and we just don't get control until the timer tick and scheduler does his thing and does an IRET instruction.

HOW CAN I IMPROVE WINDOWS 95 REAL-TIME PERFORMANCE?

Two things in Windows 95 do a lot of good. Turn off the CD ROM auto insertion detection feature. Do this from the Windows 95 start button/settings/system/device manager. Click on the CD-ROM drive, click Properties, click settings tab, and clear the check mark in the box marked "Auto insert notification." This suppresses a periodic and lengthy check of the CD-ROM drive to see if a CD-ROM became stuck in the drive. The check code is a CPU hog, locking out interrupts for too long.

On the setting page get into "Add/Remove Programs," Windows Setup, and remove the "accessibility options" (the wheelchair icon) and turn OFF the "automatic reset" feature. This suppressed some other CPU hog that runs after 24 seconds to kill some kind of disabled user support feature. In both cases, Windows 95 features were basically grabbing the CPU and holding it so long the FIFO either backed up or dried out, before out interrupt handler could gain control of the CPU to service (stuff) the ADV601 FIFO. Once the CPU hogs were turned off, we achieved a 5 μs interrupt latency and could fill the FIFO in less than 25 μs using PCI bus burst mode.

WHAT WILL THE HARDWARE TEST PROGRAM (601test) DO FOR ME?

The following options are on the main menu of 601test:

Select Looping Options

You can make 601test repeat any one, any group or all the tests. This was intended to give a good bright trace on a standard analog oscilloscope. It is also useful when looking for intermittent faults.

Toggle Log-to-Disk Flag

Log-to-disk writes everything displayed on the screen to a disk file named 601test.dat. This is useful when error messages scroll off the top of the screen too fast to read, or you wish to include a printed test data sheet with each board shipped. Log-to-disk defaults to OFF. Be careful when looping with log-to-disk active. Sooner or later the disk fills up.

Toggle Video Format [NTSC/PAL]

This sets a global flag that all three chip initialization routines (ADV7175, ADV601 and SAA7111) look at. The flag is sticky and defaults to NTSC. As of this writing, color ramp is NTSC only, and only NTSC compressed data is available to playback. Attempts to play back NTSC compressed data when set to PAL and vice versa will cause unpredictable performance. If strange things happen, check the video format setting.

Toggle Video Direction [Record/Playback]

This sets a second global flag for the chip initialization routines. Normally, this flag controls direction EXCEPT in obvious cases like color ramp, which MUST be playback. If you select color ramp or memory playback, the chip initialization routines are set to PLAYBACK regardless of the setting the R/P flag.

Set Logic Analyzer Trigger Point

This not-so-obvious setting is useful only in special cases. The FIFO test does not know it is in trouble until after it reads back faulty data. When doing full FIFO loads, it can

be quite some time between when the program sees the error and when it happened, and the logic analyzer cannot look very far back before the trigger. This option is a method to tell the program to test only a small portion of the FIFO, so the trigger point comes up sooner after the fault. If you set the trigger point early, the whole FIFO is not tested and, in some cases, the program may call defective boards good. It is good practice to set the trigger point back to full (7FFFH) if you are not using this special feature.

Suppress Feel-Good Messages for Better Looping Speed
This option turns off nonessential screen writing to make the loops run faster. Failure messages always print; success messages are turned off. Everything always prints to the log file, if the log file is active. Of course, writing to disk is slower than writing to the screen, so you ought to toggle Log-to-Disk off if you desire the greatest looping speed.

Bin Width Register Control
The ADV601 test program can now set all the bin width and reciprocal bin width registers to the same value. This will not give very good compression, but it will be useful in isolating chip faults. The final bin width scheme requires the host to read the statistics and compute a scale factor (bin width) for each of the 42 Wavelet sub-bands. This scheme is complex, and a fault in the statistics hardware, the statistics register interface to host, a bug in the host program, or defective bin width or reciprocal bin width registers will result in either bad video, tearing, or no picture. In this case, the 601 test program allows you to set all the bin width and reciprocal bin width registers to the same number. Giving the program a "1" will load 100 Hex into the bin width registers and 400 hex into reciprocal bin width registers. The program inputs floating point (fractions are allowed) and computes the hex bin width scaled 8.8 and reciprocal bin width scaled 6.10. This scheme will not give good compression, but it should permit some kind of viewable picture. If we do not have watchable video, something is broken, and it is not the statistics registers, computation or the host bin width calculation.

Option 2 lets you load one of three canned bin width sets that have worked well on test pictures. These should give reasonable compression and good looking video, without the complexities of a constant bit rate adjustment from frame to frame. The bit rate may vary as the scene changes, but we should be able to see video.

Option 3 reads back and displays the bin width register settings in hex. You can use this option to check that all the bits were loaded into the 601 registers and no bits are stuck.

WHAT IS STALL MODE?
The "Stall Mode," or "virtual FIFO mode," on the ADV601 works thusly. Should the FIFO become full, the ADV601 will hold video up in the DRAM until room appears in the FIFO. The FIFO ERR bit still sets when the FIFO is full (or empty), but the part will hold video in DRAM as well. Video will not be lost until the Memory Error bit sets.

Stall mode will help should data transfer be momentarily delayed (perhaps by an unexpected disk seek). On decode, the Huffman coder reads the compressed data out of the FIFO, decodes it and places the raw (but Wavelet-transformed) video into the DRAM. The Wavelet pipe takes the transformed video out of DRAM and inverse-transforms it back to original video. The Wavelet pipe will keep running as long as there is any data in DRAM. It only outputs a black field when the DRAM runs dry. FIFO_ERR on decode means the input to the Huffman decoder has run out, and is a warning that action is needed; it does not mean that video has been lost. If your card reacts to FIFO_ERR by quickly loading the FIFO (doing a 512-long word burst write), the Huffman decoder will pick up, the Wavelet pipe will not run out of data and the output video will be perfect. If, after FIFO_ERR is asserted, your card cannot fill the FIFO in time, the Huffman decoder will stay stalled, the DRAM will run dry and the ADV601 will output ONE black field. This will consume 25 milliseconds in PAL, giving your card plenty of time to load the rest of the field into the FIFO. The delayed field will be output next. This causes a flickering on the output video monitor, not a solid black screen. Our card does this here in Norwood.

You might need to change your driver to ignore the FIFO error bit and press on until memory error occurs. I assume your driver is checking all the status bits and reporting errors when it sees them. If you just comment out the FIFO ERR check code, you should be able to go faster with good video.

The stall mode feature is tested on the chip tester and it does work. There are no extra control bits required to enable the mode.

WHAT DOES THE END-OF-SEQUENCE CODE DO?
On decode, the ADV601 will output a black screen upon end-of-data. The EOS code serves to prompt the ADV601 to start outputting black. It is most likely smart enough to start doing black when the FIFO runs dry, but I might add the EOS code just as a safety precaution.

BIN WIDTH CALCULATION IN A NUTSHELL

The ADV601 has two main parts: a Wavelet transformer and a run length/Huffman encoder. On compression (encoding), the video is transformed into the Wavelet domain and then run length/Huffman encoded. The Wavelet transform is analogous to the Fourier transform in that it is reversible and the Wavelet domain divides the input signal into "sub-bands." The signal in each sub-band looks like the input signal after band pass filtering. Think of the Wavelet transform as a bank of band pass filters.

To reverse the Wavelet transform and recover the original signal, simply add all the sub-bands back together. If you save all the bits in all the sub-bands, you have a lossless transformation and recover the original signal intact. This is "visually lossless" compression and gives about 3:1 compression, which is not really enough for many applications.

If you look at the signal in the highest frequency sub-band, you find small amplitude wiggles that are mostly high frequency noise and a few high amplitude spikes that represent sharp edges in the picture. We obtain more compression if we just dump the low order bits in the high frequency sub-bands. This gives longer runs of zeros, which the run length encoder eats right up. It can actually improve the picture by clipping off the grass.

We clip the grass by multiplying every data point in each sub-band by a fraction, called the reciprocal bin width. This is the adaptive quantizer located in between the Wavelet transformer and the run length/Huffman encoder. There are 42 different bins and each one has a reciprocal bin width register, permitting independent quantization of each sub-band. On decode, we restore the quantized signal to its proper amplitude by multiplying each data point by a factor called the bin width. Again, there are 42 bin width registers.

The duty of the "Bin Width Calculator" program is to set all 42 bin width and 42 reciprocal bin width registers with the proper fraction to achieve the desired compression ratio. We have two bin width calculator programs at the current time. One of them looks at the sum of the squares registers for each sub-band and selects a set of bin widths that quantize hard enough to achieve the target bit rate. It is host-based and written in C. You can find the source on the Internet at ftp.analog.com/pub/dsp/adv601/software/601cman.

The other program looks at the size of the compressed field and adjusts the bin width registers up or down (negative feedback) to achieve the target bit rate. It is ADSP-2104 based and written in assembler. It's source is also on the Internet at tp.analog.com/pub/dsp/adv601/software/dspbinwd.

CAN I READ AND WRITE THE ADV601 MODE CONTROL REGISTER FROM THE DSP SERIAL PORT?

No. The DSP serial port gives the DSP access ONLY to the statistics and bin width registers. You cannot access the mode control register, the FIFO control register or the cropping registers. If the DSP is the only processor in the system, interface it via the host port (parallel port) not the DSP serial port.

CAN I DO CIF AND QCIF IMAGES WITH THE ADV601?

Yes, but not quite the way you think you would. The CIF acronym stands for "Common Image Format" and means 352 by 240 (NSTC) . This is essentially a half size image. Full CCIR 601 resolution is 720 by 486. The CIF image contains only one quarter of the pixels of a full sized image and so stores in one quarter of the disk space or transmits four times faster than full resolution, at the cost of reduced image quality.

To encode a CIF image, we have to discard half of the lines and half of the pixels in the original full-sized image. Getting rid of half the lines is easily accomplished by discarding field 2 of the interlaced video. The controlling microcomputer can simply direct compressed video field 2 into the bit bucket and only transmit or store compressed field 1. We can then discard half the pixels in field 1 by setting the ADV601's reciprocal bin width registers for the highest sub-band (Luma, CR and CB) to zero, discarding all the bits in those three sub-bands. The high sub-band runs from 6.75 MHz down to 3.375 MHz, or the top half of the video spectrum. In effect, we reduce the highest video frequency by half, saving all the bits needed to encode this information. By Shannon's sampling theorem, this is equivalent to discarding one-half the pixels. Voila! A CIF compressed image out of the ADV601 with no additional hardware. We achieve the savings of a CIF image plus the ADV601's wavelet compression of the remaining image.

To play back a CIF image on the ADV601, we simply push each field into the decode chip twice, and the ADV601 will produce a CIF resolution image at 60 fields/sec for playback on a regular TV set. The picture will play back full size (covers the entire TV Screen), but the resolution is only CIF. In other words, you will have a full size, but somewhat fuzzier picture.

To create a smaller image (which will look sharper) on a computer monitor, just paint the 243 lines of field 1 and drop every other pixel horizontally. This can be done in software or hardware.

QCIF is "Quarter CIF" or 180 by 122. This can be done by extending the technique of forming a CIF image. Discard field 2, and zero out the top two sub-bands instead of just the top one. This will reduce the video frequency by half again, or down to 1.75 MHz.

E3172–.5–3/98

PRINTED IN U.S.A.