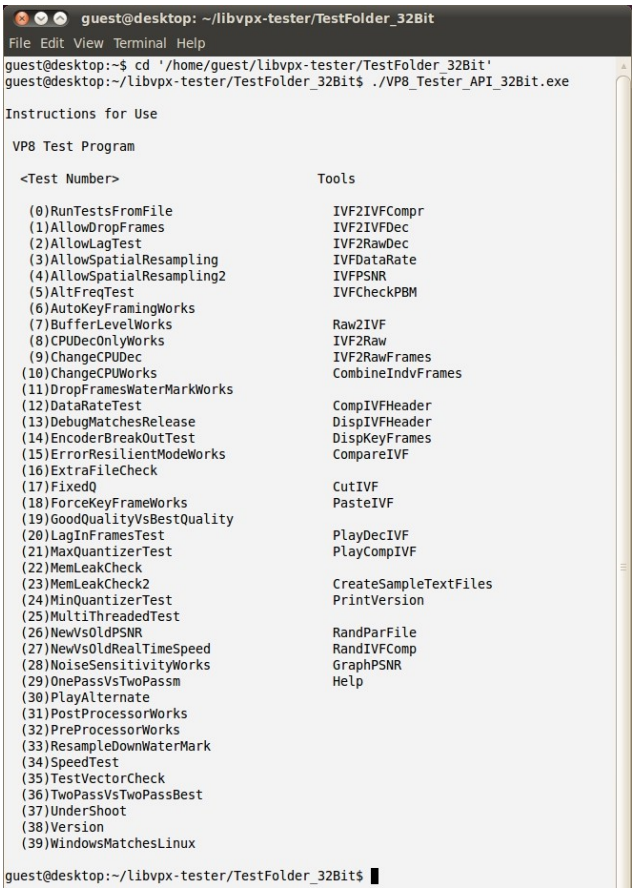
Libvpx CodecTester v0.9

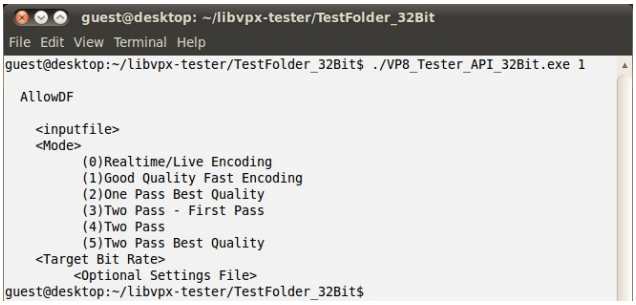
(06-11-2010)



Using the Tester

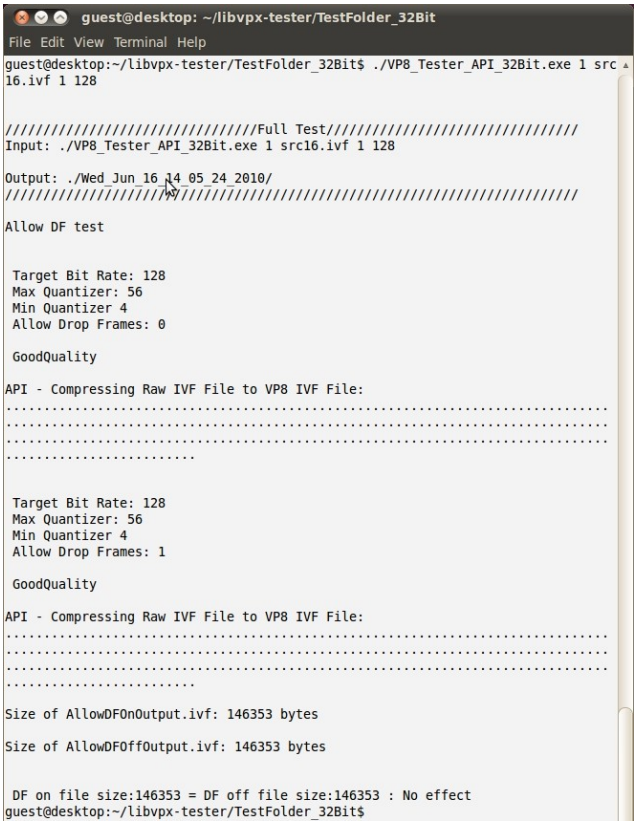
To run the executable open the command prompt, from there change the current directory to /libvpx-tester/TestFolder_32Bit or /libvpx-tester/TestFolder_64Bit depending on which tester you have built or wish to run. Run VP8_Tester_API_32Bit.exe or VP8_Tester_API_64Bit.exe.

When just the master executable is run, you will see the default screen shown on the right. The Tester is driven on an instance by instance basis. Tests are signified numerically with the number to the left of the test name representing the test. It is possible to run tests independently or from a properly formatted control text file. It is also possible to make use of tools built into the tester.



Running a Single Test

To run a single test run the executable with the number of the test you wish to run as the first command line argument. The Tester will output what further test specific input is necessary.



EXAMPLE- to run the test shown on the right type:

"./VP8_Tester_API_32Bit.exe 1 src16.ivf 1 128"

Where the first "1" is the test number, "src16.ivf" is the input file, the second "1" is the mode to run the compression in, and 128 is the target bit rate in KBps.

Running Multiple Tests

Multiple Tests are run by inputting commands from a user created text file specifying which tests are to be run and the conditions under which they are to be run. A sample control file can be created to the directory the executable is located in by using the Tester's "CreateSampleTextFiles" Tool (see "Using Tools" below). Multiple Tests may be run in four modes.

Mode 1: Run Compressions and Tests,

Mode 2: Run Compressions Only,

Mode 3: Run tests on preexisting compressions, or

Mode 4: Resume last mode.

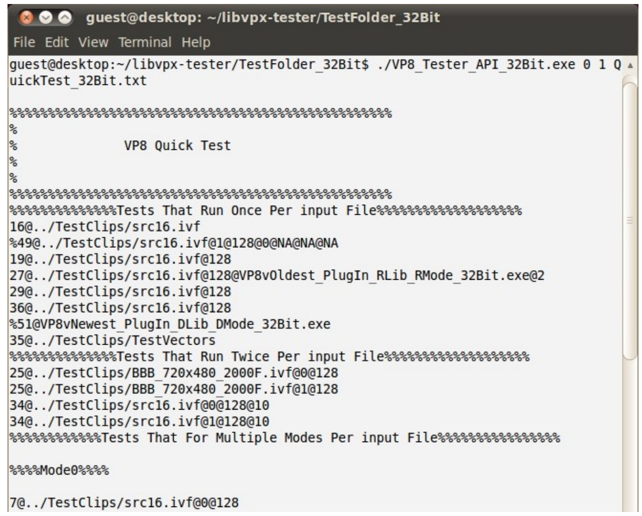
To run multiple tests run the executable with the command line argument 0, for "RunTestsFromFile", the mode you wish to run it in, and finally the controlling text file or in the case of Modes 3 and 4, the directory that compressions were being saved to or that tests were being run in.

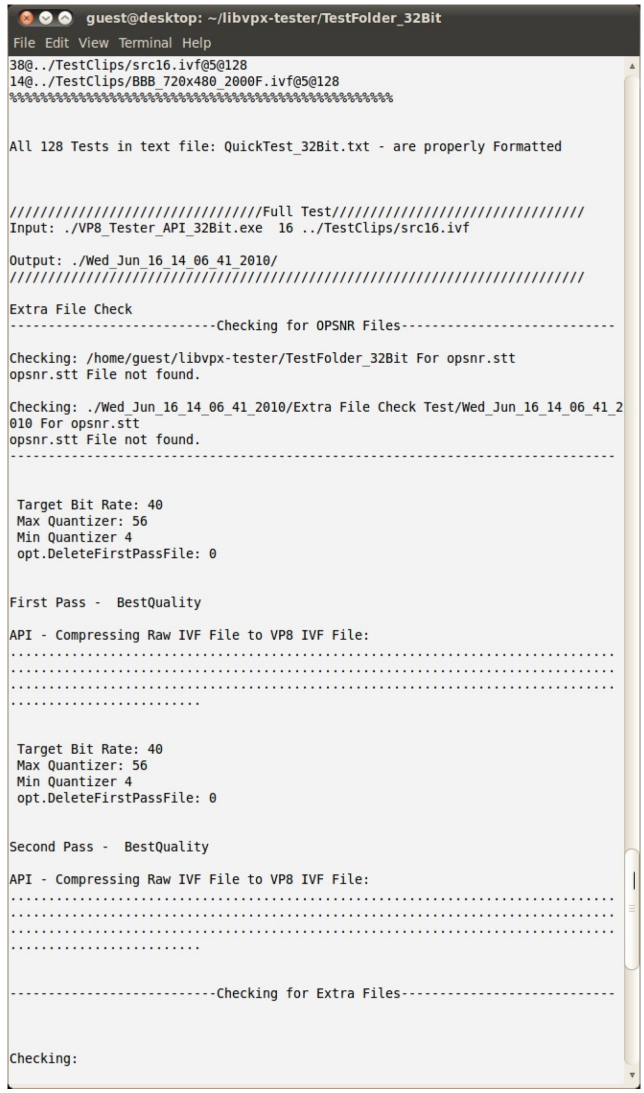
EXAMPLE- to run the multiple test shown on the left type:

"./VP8_Tester_API_32Bit.exe 0 1 QuickTest_32Bit.txt"

The sample control file provided by the tester is configured under the assumption that input test .ivf files are located in a directory named "TestClips" that exists one folder level above the executable .

The Raw video files used as input for the sample multiple test can be downloaded from "ftp.on2.com" Username: opensource, Password: opensource, Filename: vpx-sample-test-clips.rar





Using Tools

To use a tool input the name of the tool as the first command line parameter and press enter. If any further information is needed to run the tool it will be displayed in the same way test input is displayed.

EXAMPLE- to run the Create Sample Text Files Tool mentioned above type:

"./VP8_Tester_API_32Bit.exe CreateSampleTextFiles"

See the column named "Tools" under the testers default output or the section below named "Tools Overview" for a full list of supported tools.

Checking a Multiple Test's Progress

To check the status of a currently running multiple test you can check both "TestsRun.txt" and "Mode1Results.txt" (or "Mode2" or "Mode3Results" if running modes 2 or 3).

TestsRun.txt tracks the testers progress through the original input control text file. Tests that have been fully completed are marked with a "+" to their left. Tests that have had compressions only completed are marked with a "-" and tests that have not been completed at all are marked with a " " to their left.

Mode1Results.txt records the result of each test the tester has run along with the folder name it is saved to.

Mode2Results.txt records if compressions have been made properly and the folder names they were saved to.

Mode3Results.txt records the outcome of the final tests run on pre-compressed material and the folder names they were saved to.

Resuming a Stopped Multiple Test

To resume a stopped multiple test input the following:

"./VP8_Tester_API_32Bit.exe 0 4 TestFolderName"

Where "TestFolderName" is the name of the folder containing the test data you wish to restart.

Creating a Multiple Test Input Text File

To create a new multiple test input text file create a new text file. Any line that starts with a "%" will be considered to be commented out to allow for an identifying header at the beginning of the text file or notes through out the file. The "@" symbol serves as a delineator for text file arguments so to write a new test go to a new line and type the number of the test you wish to run followed by an "@" then type in the test specific input you wish to have run using an "@" between each new parameter. When finished run the file in the same way the sample file is run.

EXAMPLE– To create a new Allow Drop Frames Test that will run in Best Quality Mode for a target bit rate of 512 Kbps type:

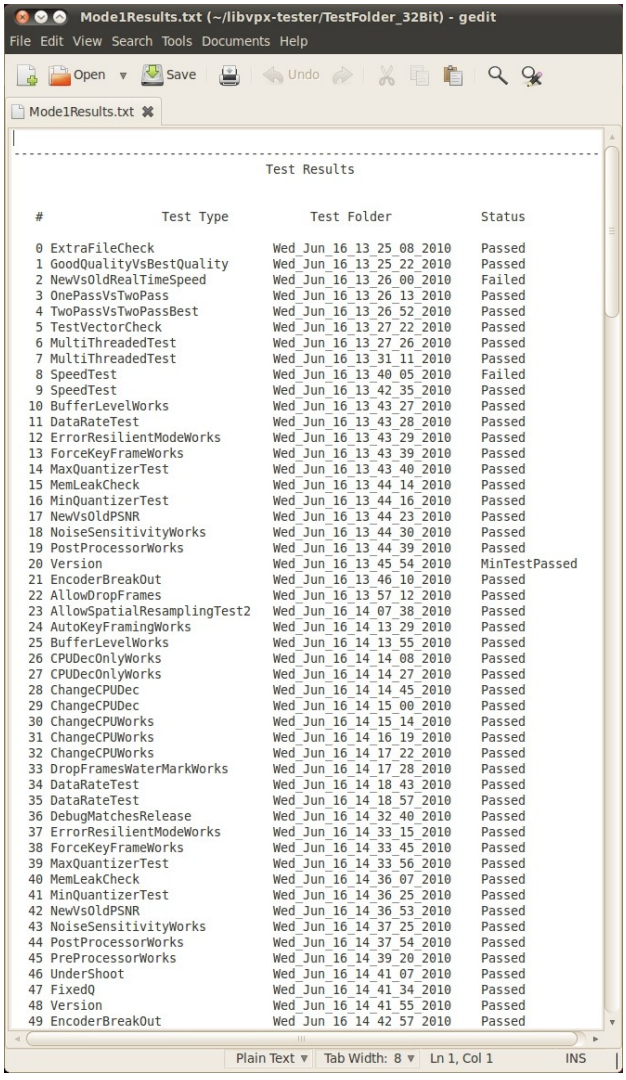
"1@FiletoTest.ivf@2@512"

Evaluating a Multiple Test's Results

When a Multiple Test finishes, the results summary will be saved in two formats. A condensed results file (Mode1Results.txt, Mode2Results.txt, Mode3Results.txt) and an expanded results file (Mode1Results_Expanded.txt, Mode2Results_Expanded.txt, Mode3Results_Expanded.txt).

The condensed results summary contains the type of test that was run, the order it was run in, the folder name of the in depth test output, and final Pass/Fail status of the test. This file lists test run in chronological order. (See picture below)

The expanded results file groups results for individual test types together. In addition to the information contained in the condensed results file the expanded file also lists the parameters under which the test was run and a summary of how many total passes and fails occur for each type of test.



Tests Overview

0	RunTestsFromFile	The test runs tests from an input file and summarizes the results. The test can be run in four modes: Mode 1 - Create Compressions and run tests, Mode 2 - Create Compressions Only, Mode 3 - Run Tests on Pre-existing Compressions, and Mode 4 - resume Tests in progress. To create a template driver text file use the command: CreateSampleTextFiles.
1	AllowDropFrames	The test creates two files; the first with Drop Frames on, the second with Drop Frames off. It then compares the sizes of the two files. If Drop Frames on is smaller than Drop Frames off the test passes.
2	AllowLagTest	The test creates two compressions; the first with Allow Lag equal to 0, the second with Allow Lag equal to 1. If the two files are not identical the test passes.
3	AllowSpatialResampling	The test creates two files; the first with Spatial Resampling off the second with Spatial Resampling on. The test then compares the sizes of the two files. If Spatial Resampling on is smaller than Spatial Resampling off the Test passes
4	AllowSpatialResampling2	The test creates a file with Allow Spatial Resampling on and Tests its PSNR. If its PSNR is greater than 15 the test passes.
5	AltFreqTest	The test creates two compressions each with user input Alternate Frequencies. It then compares the compressions. If the files are not identical the test passes.
6	AutoKeyFramingWorks	The test creates two files with identical parameters setting autokey frame equal to 1. The test then checks to make sure that the key frames for both files occur in identical locations if so the test passes.
7	BufferLevelWorks	The test creates a compression and runs CheckPBM on it if CheckPBM passes the Test Passes.
8	CPUDecOnlyWorks	The test creates a compression of the user input version and then decompresses it for CPUID values ranging from 0 to 11. The Test then compares them against one another. If all compressions are identical and the times to decompress them are not the test passes.
9	ChangeCPUDec	The test creates a compression of the user input version and then decompresses it for CPUID values 0 1 3 and 7. The test then compares them against one another. If all compressions are identical and the times to compress them are not the test passes.
10	ChangeCPUWorks	The test creates compressions of the user input version with CPUID's 0 1 3 7 and 15. If all compressions are identical and compression times are not the test passes.
11	DropFramesWaterMarkWorks	The test creates 6 compressions with DFWM values of 100, 80, 60, 40, 20 and 0 and records their sizes. If each successively lower DFWM compression has an equal or larger size than the previous the test passes.
12	DataRateTest	The test creates a compression and tests to make sure that it is with in 10% of its target. If it is the test passes.
13	DebugMatchesRelease	The test creates two compressions. The first compression is made from an executable built using the release library. The second compression is made from an executable built using the debug library. If the two files are identical the test is passed.
14	EncoderBreakOutTest	The Test creates four compressions. The first with an EncodeBreakout of 0, the second with an EncodeBreakout of 100, the thrid with an EncodeBreakout of 500 and the fourth with an EncodeBreakout of 1000. Decompressions of the Encoded files are then carried out and PSNR values are calculated. If the Decompressions run, and the PSNR values of each successive EncodeBreakout trial are with in 2 dB the test passes. If the PSNR's are greater than 2 dB but less than 5 dB the test is inconclusive. If the PSNR's have greater than a 5 dB difference the test fails.

15	ErrorResilientModeWorks	The test creates two compressions; the first with Error Resilient Mode off, the second on. If their PSNR's are with in 10% of one another the test passes.
16	ExtraFileCheck	The test creates a two pass compression and checks the current directory, the directory the executable is located in and the directory the output file is written to for extra files. If no extra files are found the test passes.
17	FixedQ	The test creates two compressions each with user input Fixed Quantizers. The test then compares the compressions. If the files are not identical the test passes.
18	ForceKeyFrameWorks	The test creates a compression using a user input value for Key Frame Frequency. The resulting compression's key frames are then checked to make sure that its key frames occur at least as frequently as Key Frame Frequency dictates. If they do the test passes.
19	GoodQualityVsBestQuality	The test creates a compression in Good Quality Mode and another in Best Quality Mode the test then records their respective PSNR and data rates. If Best Quality Mode has a higher PSNR or lower data rate than Good Quality Mode the test passes.
20	LagInFramesTest	The test creates three compressions; one with Allow Lag set to 0 the second and third with Allow Lag set to 1. The second compression uses the first user input Lag in Frames value for its Lag in Frames and the third uses the second user input value for its Lag in Frames. If none of the files are identical and the PSNR's of each successive file are within 10% of the last the test passes.
21	MaxQuantizerTest	The test creates nine compressions; the first with a WorstAllowedQ equal to 3 and each subsequent file with a WorstAllowedQ eight greater than the last until 63. If the PSNRs of each WorstAllowedQ compression from 3 to 63 increase as Worst AllowedQ decreases the test passes.
22	MemLeakCheck	The test creates a compression using the debug executable to check memory usage and records the results to an output file. If no memory leaks are found the test passes.
23	MemLeakCheck2	The test opens and closes 10,000 instances of the encoder and opens and closes 10,000 instance the decoder and then checks to make sure there are no memory leaks. If there are no leaks the test passes.
24	MinQuantizerTest	The test creates two compressions; the first with a MinQ equal to 10, the second with a MinQ equal to 60. If the first file has a higher PSNR than the second file the test passes.
25	MultiThreadedTest	The test creates two compressions; the first using a MultiThreaded equal to 2, the second using a MultiThreaded equal to 0. The test then compares the times to compress each. If MultiThreaded 2 is faster than 0 the test passes.
26	NewVsOldPSNR	The test creates two compressions; the first using the newest version of VP8 and the second using a separate executable built using an older version. It then tests the new vs. the old and passes if the new beats the olds PSNR or is at least within 1% of the old.
27	NewVsOldRealTimeSpeed	The test creates two compressions; the first using the newest version of VP8 and the second using a separate executable built using an older version. The test records the time that each compression took and if the first file compresses at least 10% faster than the second the test passes.
28	NoiseSensitivityWorks	The test compresses two compressions; the first with Noise Sensitivity equal to 0, the second with Noise Sensitivity equal to 6. If the first file has a higher PSNR than the second file the test passes.
29	OnePassVsTwoPass	The test compresses two files the first using Two Pass Good Quality Mode the second using Best Quality Mode. The test then records and evaluates the first and second files PSNR and Data Rates. If Two Pass Good Quality Mode Produces a lower bit rate and or a higher PSNR the test Passes

30	PlayAlternate	The test creates two compressions; the first with Play Alternate equal to 0, the second with Play Alternate equal to 1. The test then compares the resulting compressions if they are not equal the test passes.
31	PostProcessorWorks	The test creates a compression then creates a No Filtering decompression, and decompressions for Deblock and Noise levels ranging from 0 to 15. So long as all Deblock and Noise decompressions return a different PSNR than the No Filtering Decompression but are within 10% the test passes.
32	PreProcessorWorks	The test creates seven compressions using Noise Sensitivity values from 0 to 6 and records their PSNR's. If none of the compressions PSNR's are equal the test passes.
33	ResampleDownWaterMark	The Test creates two compressions; the first with Resample-Down-Watermark set to 90, the second with Resample-Down-Watermark set to 10. If the PSNR of the first compression is less than the PSNR of the second the test passes.
34	SpeedTest	The test works for Real Time Mode and Good Quality Mode. For Real Time Mode the test creates compressions for CpuUsed Values from - 1 to -16 and 0 to 16. If compression speed increases as CpuUsed increases and all PSNRs are within 10% of the previous the test passes. For Good Quality Mode the test creates compressions for CpuUsed Values from 0 to 5. If compression speed increases as CpuUsed increases and all PSNRs are within 10% of the previous the test passes.
35	TestVectorCheck	This test decodes each VP8 Test Vector and Checks its MD5 checksum against the expected value. If all Test Vectors decode properly the test passes.
36	TwoPassVsTwoPassBest	The test creates two compressions; the first using Two Pass Mode, the second Two Pass Best Mode. The test then records and evaluates the first and second files PSNR and Data Rates. If Second Pass Best Quality Mode Produces a lower bit rate and or a higher PSNR the test Passes
37	UnderShoot	The test creates two compressions; the first with an undershoot equal to 10 the second with an undershoot equal to 100. If the second file's bit rate is greater than the first's the test passes.
38	Version	The test creates four compressions; the first with Version equal to 0, the second with Version equal to 1, the third with Version equal to 2, the fourth with Version equal to 3. The test then decodes each and records the time it took to do so. If each successive Version takes less time than the prior to decode and has a lower PSNR the test passes.
39	WindowsMatchesLinux	The test can be run in two test modes. The first, mode 0, creates platform specific compressions and decompressions to be tested on another platform. The second mode creates platform specific compressions and decompressions and then compares them to previously encoded and decoded files created by test mode 0. If the files are identical the test passes.

Tools Overview

IVF2IVFCompr	This utility will take in a raw ivf file and produce an encoded ivf file using the given mode and bit rate. Default encode settings can be overridden by specifying a parameter file.
IVF2IVFDec	This utility will take in an encoded ivf file and output a decoded ivf file.
IVF2RawDec	This utility will take in an encoded ivf file and output a decoded raw file.
IVFDataRate	This utility will take in an ivf file and compute its average, min, max, and file data rates.
IVFPSNR	This utility will compute an encoded files psnr using the encoded file's ivf source file.
IVFCheckPBM	This utility will run CheckPBM to make sure a buffer under run wont occur.
Raw2IVF	This utility will take in a raw file and produce a raw ivf file.
IVF2Raw	This utility will take in an ivf file and produce a raw file.
IVF2RawFrames	This utility will take in an ivf file and produce individual raw frames for each frame that exists in the user specified directory.
CombineIndvFrames	This utility will combine all individual decoded frames in a folder into a single raw file in numerical order.
CompIVFHeader	This utility will compare the file and frame headers of two ivf files.
DispIVFHeader	This utility will display the file and frame headers of an ivf file.
DispKeyFrames	This utility will display the location of key frames within an ivf file.
CompareIVF	This utility will compare the video content of two ivf files and will display if they are identical, or if they differ the first frame they differ at.
CutIVF	This utility will cut a portion of an ivf file starting at Starting Frame and ending at Ending Frame to a new output file
PasteIVF	This utility will paste the contents of Inputfile2 into Inputfile1 starting at Inputfile1's First Paste Frame to a new Outputfile.
PlayDecIVF	This utility will convert an uncompressed ivf file to a raw yuv file and play it using tmnplay or mplayer.
PlayCompIVF	This utility will convert a compressed ivf file to a raw yuv file and play it using tmnplay or mplayer.
CreateSampleTextFiles	This utility will create sample text files.
PrintVersion	This utility will print the version of vp8 being used by the tester.
RandParFile	This utility will create a valid vp8 random parameter file.
RandIVFComp	This utility will create a compression using random parameters for an input ivf file.
GraphPSNR	The utility creates compressions from user input starting to user input ending bit rates at user input steps. The utility then computes and outputs the data rates, PSNRs, SSIMs, encode times and decode times for the resultant files.
Help	This utility displays tester help instructions.