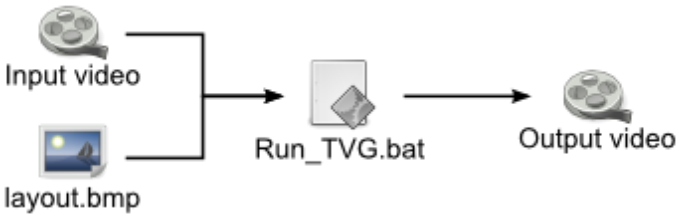
OptoFidelity Test Video Generator User Manual Version 1.0

OFTVG is an application that is used to add synchronization and frame id markers to video streams. It generates test videos where each video frame can be uniquely and automatically identified.

The implementation consists of a script, **Run_TVG.bat**, which contains the configuration parameters, and a bitmap image, **layout.bmp**, which defines the locations and sizes of the markers on the screen. The program opens an input video, which can be in various formats, optionally resizes it and then draws the markers and compresses the result into the selected output format.



The program uses GStreamer as a backend for processing the video, and includes a custom GStreamer plugin to add the markers. All the necessary software components are stored under the **bin** and **lib** folders.

1. Getting started

The distribution package includes a part of the Big Buck Bunny movie by the Blender Foundation as **big_buck_bunny_1080p_h264.mp4**. This can be used for testing the generator.

For a first test, just double-click **Run_TVG.bat**. This should open a command prompt and show encoding progress. The result is saved as **output.mov**.

2. Structure of the output video

The output video starts with an optional white sequence. This 5 second long sequence can be used to calibrate AV100, or to accurately start and stop Video Multimeter measurements.

After the white sequence, the actual test video begins. It consists of N first frames of the input video, each of which has a unique frame id marker along the bottom edge.

The length of the test sequence can be adjusted by editing the **NUM_BUFFERS** setting in the **Run_TVG.bat**.

3. Configuring the test video generator

The test video generator is configured by editing **Run_TVG.bat**. You can make multiple copies of this file under different names to keep different configuration sets. The script accesses files under **bin** and **lib** folders, so it must be in the same directory as they are.

The file is a regular Windows .bat; you can use :: to comment lines.

3.1. Input and output files

These three lines set the names of the input and output files:

SET INPUT=big_buck_bunny_1080p_h264.mov
SET LAYOUT=layout.bmp
SET OUTPUT=output.3gp

The input video can have any format supported by GStreamer, which includes most of the common formats such as AVI, MOV, MP4, 3GP and others. The format is detected automatically.

The layout image defines the location of the markers in the image. Specific colors in the image correspond to each marker. The layout image can have either BMP or PNG format. (JPEG is not recommended because the lossy compression may corrupt the markers.)

The output filename can be chosen freely. It should, however, have the correct file extension with respect to the selected video format. The list of the file extensions is given below in the description of the **CONTAINER** parameter.

3.2. Video output format

The format of the output video is configured using two parameters:

SET COMPRESSION=x264enc speed-preset=4
SET CONTAINER=gppmux

The **COMPRESSION** type defines how the video data is encoded, ie. which video codec is used. Common video codecs are H.264 and MPEG-2. Some encoders accept configuration parameters that can be specified after the codec name.

The **CONTAINER** type defines the file format used to store the encoded data. This does not affect the encoding of the video, but is merely the selection of file and frame headers that are added to the data. Common containers are AVI and MOV.

The compression and container types can be combined quite freely, but there is a small number of combinations that are not supported. Most notably, Motion JPEG is not supported in the 3GP container.

3.2.1. List of supported COMPRESSION types

Name	Encoder	Options	
Uncompressed	video/x-raw, format=…		Format is e.g. I420, YV12 or RGBA.
H.264 video	x264enc	bitrate	Bitrate in kbit/s. Default is 2048.
		speed-preset	Speed of the encoding process. 1 = fastest encoding, worst compression; 9 = slowest encoding, best compression
		profile	H.264 profile, i.e. which features of the standard are used. One of **baseline**, **main**, **high**. Set to 'baseline' for iPod and other mobile devices.
Motion-JPEG	jpegenc	bitrate	Bitrate in bit/s. Default is 300 000.
MPEG-4 part 2	ffenc_mpeg4	bitrate	Bitrate in bit/s. Default is 300 000.
		flags	Specify 'flags=global-headers' when using with .3gp or .mp4 formats.
MPEG-2 video	ffenc_mpeg2video	bitrate	Bitrate in bit/s. Default is 300 000.
Windows Media Video 8	ffenc_wmv2	bitrate	Bitrate in bit/s. Default is 300 000.
Flash video	ffenc_flv	bitrate	Bitrate in bit/s. Default is 300 000.

3.2.2. List of supported CONTAINER types

Name	Mux	File extension
MPEG-4 part 14	mp4mux	.MP4
3G mobile phone video container	gppmux	.3GP
Microsoft AVI	avimux	.AVI
QuickTime	qtmux	.MOV
Microsoft Active Streaming Format	asfmux	.ASF, .WMV
Flash Video	flvmux	.FLV

3.3. Preprocessing

The input video can be resized or the framerate can be adjusted before processing:

*Resize only: SET PREPROCESS=! videoscale ! video/x-raw-yuv,width=**640**,height=**480***
*Resize and change aspect ratio: SET PREPROCESS=! videoscale ! video/x-raw-yuv,width=**320**,height=**240**,pixel-aspect-ratio=**1/1***
*Adjust FPS: SET PREPROCESS=! videorate ! video/x-raw-yuv,framerate=**10**/1*
*Both: SET PREPROCESS=! videorate ! videoscale ! video/x-raw-yuv,framerate=**5**/1,width=**320**,height=**240***

The preprocessing is specified by uncommenting one of the example lines in **Run_TVG.bat** and editing the width, height and framerate parameters to match the application. If no preprocessing is needed, all **PREPROCESS**-lines should remain commented out.

Note that a non-integer framerate should be given as a fraction. For example 59.94 FPS equals 60000/1001.

The pixel aspect ratio is used by some video players to rescale the video at the playback time. For example, iOS devices have a 3:2 screen but expect 320x240 video (4:3). To make full-screen video for these devices, pixel-aspect-ratio must be set to (3:2)/(4:3) = 9:8.

3.4. Length of the test video

The test video consists of the first **NUM_BUFFERS** frames of the input:

SET NUM_BUFFERS=64

The number of buffers should not be larger than what can be expressed using the frame id bits defined in the layout image. For example, 8 bits can represent up to 2^8 = 256 frames. For AV10, you should always use a power of 2 as the number of frames, e.g. 8, 16, 32, 64, 128, 256, 512, 1024, 2048 or 4096 frames.

The repeat count can be 1 or larger.

3.5. Generation of white prefix/suffix sequence

The test video generator can add a white prefix or suffix to the video. In AV100, this is used to calibrate the marker locations. With Video Multimeter, the white period allows time for the user to start and stop the measurement, in order to obtain accurate frame counts.

SET CALIBRATION=prepend

Allowed values are **off**, **only**, **prepend, both, rgb6_prepend** and **rgb6_both**.

Off means that no calibration sequence is added to this video. For use with AV100, you have to have it as a separate video file. Such a separate video can be generated by settings **CALIBRATION=only**, which generates only the calibration sequence and no test video. Video Multimeter can also be used without the white sequence when accurate frame counts are not important.

The third choice, **prepend**, puts the calibration sequence in the beginning of the test video. The fourth choice, **both**, also appends white frames to the end of the test video. This is used with Video Multimeter to provide time to stop the measurement accurately.

The **rgb6_** options will place the white color only in the marker area, which is especially suitable for use with Video Multimeter. The normal **prepend** and **both** options will make the whole frame white, which works for both AV100 and Video Multimeter.

4. Editing the marker layout bitmap

The layout bitmap can be edited using any bitmap editor, such as Microsoft Paint or the GIMP. If you need pixel-accurate placement of the markers, the resolution of the layout bitmap should equal the video resolution. Otherwise the bitmap will be automatically resized in memory by TVG.

Note: If you resize the layout bitmap manually, use “nearest neighbor” interpolation mode in the image editor. Otherwise the edges of the markers may blur and get mixed with other markers, which will cause visual errors, such as flickering borders around markers.

The bitmap should contain rectangles with specific colors to define where the markers will be placed:

Color value (Red, Green, Blue)	**Meaning**
(255, 255, 255) (White)	Background color
(10, 10, 10)	1st frameid bit, ie. changes most often

(20, 20, 20)	2nd frameid bit
(30, 30, 30)	3rd frameid bit
...	... similarly, greyscale values in 10 step increments ...
(240, 240, 240)	24th frameid bit
(255, 0, 0) (Red)	Primary sync marker, changes every frame
(0, 255, 0) (Green)	Secondary sync marker, changes every other frame
(0, 0, 255) (Blue)	Multicolor marker (6-state: R, Y, G, C, B, P)

All markers do not need to be included. It is enough to include as many frame ids as are necessary to identify the wanted number of frames. The markers should be placed in numeric order, either vertically or horizontally in a line.

Any unknown color will be regarded as background. However, new marker types might be added in the future so the background should be left white to guarantee that it doesn't get mixed with the markers.

5. Audio support

The input audio is included in the output video. If the input video does not contain an audio track, silent audio is automatically generated.

For testing lip sync delay using Video Multimeter, additional sound markers have to be generated. Set LIPSYNC option to e.g. 2000 to generate one marker every two seconds:

SET LIPSYNC=2000

6. Alternative configuration method: .tvg files

Starting with version 0.4, you can store only the changed settings in a file named *something.tvg*. Drag this file over Run_TVG.bat or select "Open with.." to associate .tvg files with the program.

All the same commands work in .tvg files; they are simply included into the main Run_TVG.bat at runtime.

7. Analyzing the generated video files

Starting with version 1.0, there is an included tool to analyze both input video files and the generated test videos. Simply drag the video file over the **Analyzer.bat** or give it as an argument to the command. The script will analyze the file and show the details as text:

```
{
    "file":           "C:\...\output.mov",
    "demuxer":        "qtdemux",
    "video_decoder":  "avdec_h264",
    "audio_decoder":  "faad",
    "muxer":          "3gppmux",
    "video_encoder":  "x264enc",
    "audio_encoder":  "avenc_aac",
    "resolution":     [1920,1080],
    "framerate":       24.00,
```

```
    "total_frames":      493,
    "video_length":   20.542,
    "audio_length":   20.689,
    "markers_found":       1,
    "markers": [
      {"index":   0, "pos": [1674,   0], "size": [246,245],
       "crc": "b679be3d", "type": "RGB6"}
    ],
    "video_structure": {
      "header_frames":      121,
      "locator_frames":       0,
      "content_frames":     256,
      "trailer_frames":     116
    },
    "lipsync": {
      "audio_markers":           6,
      "video_markers":           6,
      "audio_delay_min_ms":    0.3,
      "audio_delay_max_ms":    0.4
    },
    "frame_data": "frames.txt",
    "warnings": [
    ]
}
```


The reported values are described in the table below:

Parameter	**Description**
file	Full path to the analyzed video file.
demuxer	Automatically detected ’demuxer’ element, i.e. the container format of the video and audio.
video_decoder	Automatically detected decoder element for the video content.
audio_decoder	Automatically detected decoder element for the audio content.
muxer	Value of the ‘CONTAINER’ setting for Run_TVG.bat to generate this format.
video_encoder	Value of the ‘COMPRESSION’ setting for Run_TVG.bat to match this video encoding.
audio_encoder	Value of the ‘AUDIOCOMPRESSION’ setting for Run_TVG.bat to match this audio encoding.
resolution	Pixel resolution of the video content.
framerate	Nominal frame rate of the video content (or ‘variable’ if the video has variable FPS).
total_frames	Total number of frames in the video file.
video_length	Length of the video content in seconds.
audio_length	Length of the audio content in seconds.
markers_found	Number of automatically detected test video markers (frame id, sync id, 6-color RGB marker) in the video.
markers	
index	0-based index of the marker found. Markers are indexed starting from upper left.
pos	Pixel position of the upper left corner of the marker: [x, y]
size	The pixel size of the marker: [width, height]

crc	Checksum of the marker colors during the video. Mostly for use in test cases.
type	Type of the marker. One of 'RGB6', 'FRAMEID', 'SYNCMARK' or 'UNKNOWN'. Any non-perfect sequence is marked as 'UNKNOWN'.
video_structure	
header_frames	Number of completely white frames in the beginning of the video.
locator_frames	Number of frames with the frameids in black (only with marker type 'FRAMEID'.
content_frames	Number of video content frames.
trailer_frames	Number of completely white frames at the end of the video.
lipsync	
audio_markers	Number of detected audio lipsync markers (beeps) in the sound track.
video_markers	Number of detected video lipsync markers (black color in RGB6) in the video content.
audio_delay_min_ms	Minimum lipsync value during the video, normally around 0.0 +- 1 ms.
audio_delay_max_ms	Maximum lipsync value during the video, normally around 0.0 +- 1 ms
frame_data	Notice that the frame data was saved to 'frames.txt' for more detailed analysis if needed.
warnings	Any warnings about incoherent timestamps if detected.

The analyzer tool can be used, for example, for the following purposes:

- To check the video codec used in a specific video, in order to configure TVG to produce same type of videos.
- To verify the lip sync after various encoding steps, e.g. if uploaded to an online video service or transcoded for a specific device.
- To get detailed timestamp information about the frames and markers in the video, for e.g. comparison with Video Multimeter results.

8. Details on the implementation

The **Run_TVG.bat** script uses the GStreamer **gst-launch** command for launching the custom TVG plugin and compressing the video. The actual command is at the end of the script, and uses variables defined earlier in the file. Normally editing the variables is enough, but below is some basic information about the structure to aid in development.

In **gst-launch**, a pipeline is defined as a set of elements separated by exclamation point: "**element1 ! element2 ! element3**". After each element, there can be any number of property values: "**element property1=value1 …**". The basic pipeline flows from a **filesrc** (file reader) up to **filesink** (file writer). The names of the compression and container elements come directly from the variables **%COMPRESSION%** and **%CONTAINER%**. Therefore any attributes for them can be added after the SET= statements.

More documentation on the relevant gstreamer parts can be found here:

http://linux.die.net/man/1/gst-launch-1.0 (gst-launch command)
http://gstreamer.freedesktop.org/data/doc/gstreamer/head/gst-plugins-ugly-plugins/html/gst-plugins-ugly-plugins-x264enc.html (x264enc encoder)

8.1. License notices

The source code of the program is available under GPL license from https://github.com/OptoFidelity/TVG . The distribution package includes third-party components under GPL, LGPL and other licenses.

9. Troubleshooting

Error messages are printed in the console window. Below are some common error causes.

Error message	**Reason**
WARNING: erroneous pipeline: could not link ffenc_mjpeg0 to gppmux0	The video encoder and container format are incompatible with each other. Use either different encoder or a different container.
streaming stopped, reason not-negotiated when using H264 encoder with AVI format.	Add **byte-stream=true** option after **x264enc.**

If the cause of the error is not found, please contact the software vendor and provide the following information:

1. The error message displayed on the screen.
2. The Run_TVG.bat you are using.
3. Contents of the **debug** folder.