SH7724

GStreamer Multimedia Framework for SH7724

R01AN0511EU0100
Rev.1.00
Jan 26, 2011

Introduction

This document provides an introduction to the GStreamer framework for the Linux operating system as it pertains to the SH7724 platform. There are also many examples for reference.

Target Device

SH7724

Contents

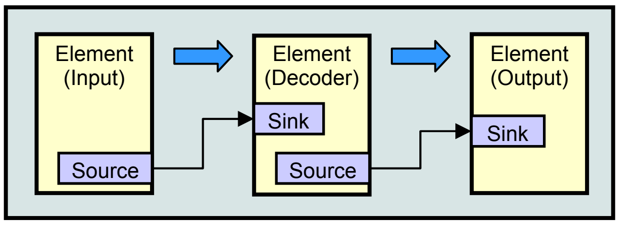
1. About GStreamer

GStreamer (http://gstreamer.freedesktop.org/) is a framework for combining multiple components in order to encode or decode a media stream. It provides a quick and easy way to add audio & video support to your product for all kinds of media formats without you ever having to dig up and digest a bunch of media specifications. It's also free.

Totem (default media player for Ubuntu) for example utilizes the GStreamer framework. Compared to Windows Media Player, GStreamer applications can play many more media formats because it has more codecs at its disposal.

One thing to note is that while GStreamer can play all type of media formats, it's not really the GStreamer software doing the decoding. It's mostly made up of other open source (and closed source) codecs. GStreamer is essentially a 'wrapper' around all those different codecs and video/sound manipulators making it easy to integrate into a product and switch elements in and out as you needed.

The general idea is that elements (grouped together in libraries called 'plug-ins') are pipelined together to create the designed encode or decode stream. Knowing which elements /plug-ins are available and which need to be 'piped' together and how is the trick about learning GStreamer.



To take advantage of the Video Processing Unit (VPU) in the SH7724, a GStreamer plugin was created by Renesas to be a drop in replacement for the standard software based H.264 codecs and other elements (like camera capture, resizing and others)

Filters are used to change the default format of data as they flow from element to element. You can think of them as like a type cast in programming language. In Linux, the term "Sink" is also used to define the final stage of an output. A screen sink for example would be a video frame buffer. An audio sink could be ALSA (a common Linux audio player)

Each plug-in can define as many options as they want. Some plug-ins (like the one Renesas made) even take a configuration file location as an argument because the number of options are greater than you really want to specify as individual parameters.

GStreamer comes with a default set of command-line utilities that can help in application development. The application 'gst-inspect' can be used to inspect all properties, signals, dynamic parameters and the object hierarchy of an element. This can be very useful to see which GObject properties or which signals (and using what arguments) an element supports. Run "gst-inspect fakesrc" to get an idea of what it does. The app "gst-launch" is a simple script-like command line application that can be used to test pipelines. For example, the command "gst-launch audiotestsrc ! audioconvert ! audio/x-raw-int,channels=2 ! alsasink" will run a pipeline which generates a sine-wave audio stream and plays it to your ALSA audio output. This is very useful in prototyping and creating demos quickly. Note that the command line of gst-launch uses exclamation points ('!') to separate the plug-ins, kind of like the pipe symbol (|) on a shell command line.

The components of GStreamer are grouped together in separate packages for download. The 'GStreamer' and 'Base' packages are essentially your starting point for the overall framework.

For the codec support, there are 3 main plugin packages; Good, Bad and Ugly. The Good contains elements that are well supported, have good licenses and good documentation. The Bad contains less supported code that needs testing or fixing. The Ugly have elements that are well supported and good code, but there may be some license issues you'll have to look into before using in your product.

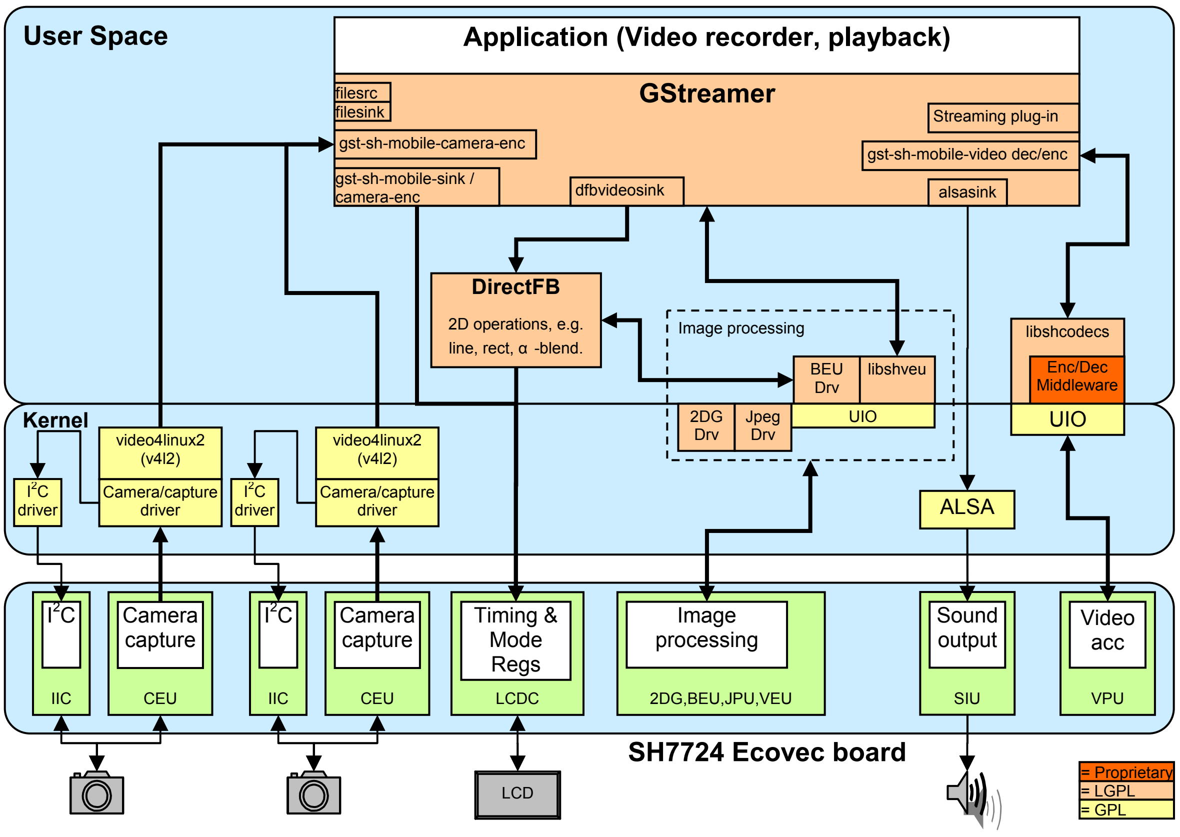
There are other download packages available on the GStreamer website. Some codes (like FFmpeg) or other specialty utilities have a module package completely for themselves.

There is a master list on the gstreamer web site that show what media support is located in which package so you know what to download. By putting the software in these categories, it at least gives you a general idea of what type of software you're getting. For the 'Ugly' for example, you may not have been aware of the licensing issues for some of the media formats, so this gives you a good heads up in that sense.

The license associated with GStreamer is a GNU LGPL. With the GNU LGPL license, even though the code that you get is free and open source, you can still use it as part of a project that will not be free/open source. Compared to GPL where any software that uses GPL code will automatically be required to also be open source. The importance of this

LGPL license is that you get to use free open source code, but it doesn't hinder you from making a closed source product. It should be noted that while the software for encoding and decoding various types of media are freely available, it does not mean the technology behind the codecs are royalty free. For example, MP3 files are very widely used, but the technology is still patented and licensed (until 2012 anyway). Before releasing a product, you should check with the patent holders for licensing costs.

Below is a representation of the interaction between GStreamer elements, the Linux kernel and the SH7724 hardware.



2. Getting Gstreamer for SH7724

While the complete source for the GStreamer is available on their website, setting up the make files for cross compiling it takes some know how. So, here are two methods of getting GStreamer for your SH7724

2.1 Prebuilt Binaries in Debian Repository

Renesas maintains a complete set of binaries for the SH4 for the Debian distribution. You can download a Debian root file system for the SH7724 from the Renesas OSS website (oss.renesas.com) and then configure and run 'apt-get' on the SH7724 board itself to get the GStreamer binaries.

Alternatively, you could download the install package (.deb file) directly from the SH4 Debian repository online (essentially, the sample place that 'apt-get' gets them from). If you browse here (http://ftp.debian-ports.org/debian/pool-sh4/main/g/), you can look up and download the GStreamer packages (.deb files) you want and copy them to the board and install them using 'dpkg -i /tmp/gst-xxxx.deb' after you've booted back into Linux. It's also nice to have the .deb files so you can archive them in case you ever want to recreate your final file system.

2.2 Download and building from source inside a 'Buildroot'

This method is if you don't desire to use the full Debian root file system and would like to make a smaller, more embedded type file system for your product.

First, 'Buildroot' (http://buildroot.uclibc.org/) is a build environment that specializes in cross compiling for embedded processors. Since many Linux projects out there have makefiles that were only intended to be build natively (not cross compiled) and are x86 centric, Buildroot attempts to ease that process by creating an environment that allows those makefiles to build for an embedded target. GStreamer is just one of the many packages that have been added to

Buildroot environment. Buildroot also takes care of downloading the package source and applying any patches needed to enable cross compiling. You simply choose what you want to build, and it goes out and downloads it and then builds it. In the end, it creates a complete file system that you can copy and use on your product.

Additionally, Renesas has provided a branch of the standard Buildroot package that is specific to the SH7724 platform. This can be found here: https://github.com/pedwo/buildroot/tree/renesas-2010.08 . This branch includes a default config as well as patches to the vanilla u-boot and Linux Kernel that are specific to the SH7724. Patches for application packages such as DirectFB are also included in order to take advantage of the SH7724 hardware. Also, specific custom Renesas applications have been added to make downloading and building simple.

Follow the simple instructions on the Wiki page (https://github.com/pedwo/buildroot/wiki) to get started. By default there is a lot of stuff selected in the MS7724_av_defconfig file so you might want to go through the current selected list (by using 'make menuconfig' after you've done 'make MS7724_av_defconfig') and remove some things that you aren't interested in quite yet, otherwise it will take a while to download and build everything. Some examples of things that are in there by default that you might want to remove are u-boot, the Linux kernel, Qt and DirectFB. While these are all good things, it's easy enough to go back and add them back in later, and for now it will save you a good amount of time.

3. About the Renesas GStreamer Elements

The following explains about the elements that contained in the Renesas GStreamer plugin

3.1 gst-sh-mobile-enc

This element is used to encode a given YCbCr source into a H.264 stream using the SH7724 VPU.

3.2 gst-sh-mobile-camera-enc

This element is used to both capture from a V4L2 input (ie, the cameras) and H.264 encode it using the SH7724 VPU. The reason why these two functions were combined into a single element is due to performance as we can use the standard v4l2src elements with gst-sh-mobile-enc, but the capture buffers are allocated by v4l2. This means our encoder element has to copy the captured frames to a new UIOMux allocated buffer before encoding them. Combining the two elements into a single new element eliminates this needless processing.

This element requires a control file to be specified. The main thing you really need to be concerned about in the file is the image size you wish to capture. You have to make sure this matches up with the video/x-h264 element filter that usually follow this element in the pipeline. Sample control files can be found in /usr/share/libshcodecs in the SH7724's file system.

When the element option "preview=1" is specified, the element will also dump the current captured image to the Linux Frame Buffer (display images being captured) as it also encodes it.

3.3 gst-sh-mobile-dec

This element decodes a given H.264 Elementary Stream (ES) into YCbCr data.

3.4 gst-sh-mobile-mixer

This element can blend 2 or more images together. It uses the libshbeu API to use the SH7724's BEU to do the blending. The xpos/ypos is the location and the alpha is the level of transparency. This element pretty much does what the standard videomixer element does, but we support a wider range of inputs and outputs.

3.5 gst-sh-mobile-resize

This element can be used as a direct replacement for the standard videoscale element and ffmpegcolorspace (some formats aren't supported, but the most common ones are).

3.6 gst-sh-mobile-sink

The gst-sh-mobile-sink element outputs images directly to the Linux Frame Buffer (eg, LCD screen). This element knows that the frame buffer is double buffered hence uses this feature. Also, the framebuffer memory needs to be registered with UIOMux so that it can dump images straight into the framebuffer.

4. Example gst-launch Scripts

Please remember that the commands are single command lines. But, in Linux, you can extend a single command line to multiple lines by adding a backslash '\' to the end of each line, just like in 'C' language coding. We will be using '\' in the long scripts to make them look nice and to allow you to copy/paste the text directly into a terminal or text editor for further editing.

Note that an exclamation sign '!' is used in the gst-launch command to separate the elements. Don't get confused and put in a pipe '|' symbol, otherwise you will get all kinds of odd errors.

The text in **bold** are portions that you might need to edit yourself since they will be specific to the specific media you are trying to play/record.

If you want to log the messages, replace gst-launch with gst-launch 2>debug.log --gst-debug-no-color --gst-debug=gst-sh-*:5

When encoding, use the -e with gst-launch. This forces End Of Stream down the pipeline when Ctrl+C is pressed to terminate the pipeline. This is often needed to add the headers to particular file formats (e.g. mp4)

4.1 Record

4.1.1 Camera to Encoding to a File (H.264 ES)

```
gst-launch gst-sh-mobile-camera-enc cntl_file=/tmp/480p-stream.ctl preview=1 ! \
  video/x-h264, width=640, height=480, framerate=30/1 ! filesink
location=/tmp/mycoolvideo.264
```

or

```
gst-launch -e \
 v4l2src device=/dev/video0 ! "video/x-raw-yuv, format=(fourcc)NV12, width=320,
height=240, framerate=15/1" \
 ! queue ! gst-sh-mobile-enc cntl_file=ctl/h264-video0-qvga-stream.ctl \
 ! filesink location=tmp.264
```

This capture from a camera and save an encoded stream to a file.

The MS7724 board has 2 camera connectors and 1 NTSC input jack. Note that the 2nd camera and the NTCS share a CEU (Capture Engine Unit) so dip switches need to be set to select one or the other. In the control file, you specify the device you want to capture from (input_yuv_file = video0;). If you have 2 cameras attached, you can choose which camera to capture from (/dev/video0 or /dev/video1). If you boot your SH7724 Ecovec board without any camera plugged in, and the switches are set to use the NTSC jack, then the NTSC input jack will be video0 and you can capture from that instead of a camera.

4.1.2 Camera to Resize to Encode to File (H.264 ES)

```
gst-launch -e \
 v4l2src device=/dev/video0 ! "video/x-raw-yuv, format=(fourcc)NV12, width=640,
height=480, framerate=30/1" \
 ! gst-sh-mobile-resize \
 ! "video/x-raw-yuv, width=320, height=240, framerate=15/1" \
 ! gst-sh-mobile-enc cntl_file=ctl/h264-video0-qvga-stream.ctl \
 ! filesink location=tmp.264
```

4.1.3 Record PCM Audio to File (AVI)

```
gst-launch -e \
 alsasrc ! audio/x-raw-int, width=24,rate=44100,channels=2 \
 ! queue ! audioconvert ! avimux ! filesink location=file.avi
```

4.1.4 Record PCM Audio to Encode to File (MP4 AAC)

```
gst-launch -e \
 alsasrc ! audio/x-raw-int, rate=16000, channels=2 \
 ! queue ! audioconvert ! faac outputformat=1 profile=LC bitrate=192000 \
 ! queue ! mux. \
   mp4mux name=mux ! filesink location=./test.m4a
```

4.1.5 Camera to File (YCbCr) (no encoding)

```
gst-launch -e \
 v4l2src device=/dev/video0 ! "video/x-raw-yuv, format=(fourcc)NV12, width=320,
height=240, framerate=15/1" \
 ! filesink location=tmp.yuv
```

4.1.6 Camera (VGA) to Encode to File (H.264 ES) (with LCD preview)

Note that the ceu capture size specified in the cntl_file is used for capture, then scaled to the size specified here.

```
gst-launch \
   gst-sh-mobile-camera-enc cntl_file=ctl/h264-video0-vga-stream.ctl preview=1 \
 ! video/x-h264, width=640, height=480, framerate=10/1 \
 ! filesink location=encoded_video.h264
```

4.1.7 [File(H.264 ES) to Decoder] and Camera to Mixer to Display

```
gst-launch \
 filesrc location=/tmp/advert-h264-vga-25fps-2mbps.264 \
 ! "video/x-h264, width=640, height=480, framerate=30/1" \
 ! gst-sh-mobile-dec \
 ! mix. v4l2src device=/dev/video0 \
 ! "video/x-raw-yuv, format=(fourcc)NV12, width=320, height=240, framerate=15/1" \
 ! mix. \
   gst-sh-mobile-mixer name=mix sink_1::alpha=0.8 sink_1::xpos=100 sink_1::ypos=0 \
 ! gst-sh-mobile-sink
```

This takes 2 video streams (1 from our video decoder, 1 from a cameras using v4l2src) and blends them using the gst-sh-mobile-mixer element. This uses libshbeu to use the BEU to do the blending. The xpos/ypos is the location and the alpha is the level of transparency.

4.1.8 [File(H.264 ES) to Decoder] and Camera0 and Camera1 to Mixer to Display

```
gst-launch \
 filesrc location=/tmp/advert-h264-vga-25fps-2mbps.264 \
 ! "video/x-h264, width=640, height=480, framerate=30/1" \
 ! gst-sh-mobile-dec \
 ! mix. v4l2src device=/dev/video0 \
 ! "video/x-raw-yuv, format=(fourcc)NV12, width=320, height=240, framerate=15/1" \
 ! mix. v4l2src device=/dev/video2 \
 ! "video/x-raw-yuv, format=(fourcc)NV12, width=320, height=240, framerate=15/1" \
 ! mix. \
   gst-sh-mobile-mixer name=mix sink_1::alpha=0.8 sink_1::xpos=160 sink_1::ypos=120
sink_2::alpha=0.8 \
 ! gst-sh-mobile-sink
```

This takes 3 video streams (1 from our video decoder, 2 from separate cameras using v4l2src) and blends them using the gst-sh-mobile-mixer element. This uses libshbeu to use the BEU to do the blending. The xpos/ypos is the location and the alpha is the level of transparency.

4.2 Playback

4.2.1 Play a H.264 Elementary Stream (ES) File

```
gst-launch filesrc  location=/tmp/my_movie_720p.h264 ! \
  video/x-h264,width=1280,height=720,framerate=24/1 ! gst-sh-mobile-dec ! gst-sh-
mobile-sink
```

or

```
gst-launch \
 filesrc location=qvga.264 ! "video/x-h264, width=320, height=240, framerate=15/1" \
 ! gst-sh-mobile-dec \
 ! gst-sh-mobile-sink
```

Change “location=/tmp/my_movie_720p.h264” to where you H.264 ES file is located.

An Elementary Stream (ES) is essentially a raw H.264 data stream. Because there is no stream information in a raw H.264 ES stream, we need to use the “video/x-h264” filter to explain that it is H.264 data and what its dimensions and frame rate are before passing it to the gst-sh-mobile-dec element.

4.2.2 Play MP4 file

```
gst-launch filesrc  location=/root/media/serenity.mp4 ! qtdemux ! gst-sh-mobile-dec !
gst-sh-mobile-sink
```

The qtdemux element extracts video from MP4 files so that it may become an input for decode elements. It also supports other ‘QuickTime’ file formats.

4.2.3 Hardware vs. Software H.264 Decoding Comparison

This will shows you the difference that the internal VPU makes when you use it to decode H.264 frames as opposed to using the CPU and using a software codec. This example uses a H.264 ES file.

There is the CPU version:

```
gst-launch filesrc  location=/tmp/my_movie_720p.h264 ! video/x-h264, width=1280,
height=720, framerate=24/1 ! \
  ffdec_h264 ! ffmpegcolorspace ! videoscale ! video/x-raw-rgb, height=440, width=800 !
fbdevsink
```

And here is the VPU version

```
gst-launch filesrc  location=/tmp/my_movie_720p.h264 ! \
  video/x-h264,width=1280,height=720,framerate=24/1 ! gst-sh-mobile-dec ! gst-sh-
mobile-sink
```

The output of ffdec_h264 is YCbCr, so we need to use ffmpegcolorspace in order to convert it to RGB before we can display it.

4.2.4 Play an MP3 Audio File

```
gst-launch \
 filesrc location=./my_song.mp3 \
 ! mad ! audioconvert ! audioresample ! autoaudiosink
```

This is simply a software decoding of the MP3 file.

The ‘mad’ plug-in is an implementation of the MPEG Audio Decoder (MAD) library of software.

4.2.5 Playback raw audio in avi container

```
gst-launch \
 filesrc location=file.avi ! avidemux name=demux demux.audio_00 \
 ! queue ! decodebin ! audioconvert ! audioresample ! autoaudiosink
```

4.2.6 Play MP4 AAC File

```
gst-launch \
 filesrc location=./test.m4a \
 ! queue ! faad ! audioconvert ! audioresample ! autoaudiosink
```

```
gst-launch \
 filesrc location=/sample_media/audio/RANewMovie1024_track2.aac \
 ! queue ! faad ! audioconvert ! audioresample ! autoaudiosink
```

4.2.7 Movie (audio & video) playback

```
gst-launch \
 filesrc location=/sample_media/movie/advert-h264-vga-25fps-2mbps_aac.avi \
 ! avidemux name=demux \
  demux.video_00 ! queue ! gst-sh-mobile-dec ! gst-sh-mobile-sink \
  demux.audio_00 ! queue ! decodebin ! audioconvert ! audioresample ! autoaudiosink
```

4.3 Transcoding

4.3.1 File (YCbCr) to Encoder to File (H.264 ES)

```
gst-launch \
 filesrc location=qvga.yuv ! "video/x-raw-yuv, format=(fourcc)NV12, width=320,
height=240, framerate=15/1" \
 ! gst-sh-mobile-enc stream-type=h264 bitrate=250000 \
 ! filesink location=tmp.264
```

4.3.2 File (H.264 ES) to Decoder to File (YCbCr)

```
gst-launch \
 filesrc location=qvga.264 ! "video/x-h264, width=320, height=240, framerate=15/1" \
 ! gst-sh-mobile-dec \
 ! filesink location=tmp.yuv
```

4.3.3 File(H.264 ES) to Decoder to Encoder File (H.264 ES)

```
gst-launch \
 filesrc location=qvga.264 ! "video/x-h264, width=320, height=240, framerate=15/1" \
 ! gst-sh-mobile-dec \
 ! gst-sh-mobile-enc stream-type=h264 bitrate=2000000 ratecontrol-skip-enable=0 \
 ! filesink location=tmp.264
```

4.3.4 File (H.264 ES) to Decoder to Encoder to File (H.264 ES)

```
gst-launch \
 filesrc location=qvga.264 ! "video/x-h264, width=320, height=240, framerate=30/1" \
 ! gst-sh-mobile-dec \
 ! "video/x-raw-yuv, width=320, height=240, framerate=30/1" \
 ! gst-sh-mobile-enc stream-type=h264 bitrate=1000000 ratecontrol-skip-enable=0 \
 ! filesink location=tmp.264
```

4.3.5 File (H.264 ES) to Decoder to Resize to Encoder to File (H.264 ES)

```
gst-launch \
 filesrc location=/tmp/advert-h264-vga-25fps-2mbps.264 \
 ! "video/x-h264, width=640, height=480, framerate=30/1" \
 ! gst-sh-mobile-dec \
 ! gst-sh-mobile-resize \
 ! "video/x-raw-yuv, width=320, height=240, framerate=30/1" \
 ! queue ! gst-sh-mobile-enc stream-type=h264 bitrate=1000000 ratecontrol-skip-enable=0 \
 ! filesink location=tmp.264
```

4.4 Testing

4.4.1 Testsrc (YCbCr) to Encoder to File (H.264 ES)

```
gst-launch -e \
 videotestsrc \
 ! "video/x-raw-yuv, format=(fourcc)NV12, width=320, height=240, framerate=15/1" \
 ! gst-sh-mobile-enc stream-type=h264 bitrate=250000 ratecontrol-skip-enable=0 \
 ! filesink location=tmp.264
```

```
gst-launch \
 videotestsrc \
 ! "video/x-raw-yuv, width=320, height=240" \
 ! gst-sh-mobile-enc stream-type=h264 bitrate=250000 \
 ! filesink location=tmp.264
```

4.4.2 Testsrc (RGB) to Resize to Encoder to File (H.264 ES)

```
gst-launch -e \
 videotestsrc \
 ! "video/x-raw-rgb, bpp=16, width=160, height=120" \
 ! gst-sh-mobile-resize \
 ! "video/x-raw-yuv, width=320, height=240, framerate=15/1" \
 ! gst-sh-mobile-enc stream-type=h264 bitrate=250000 ratecontrol-skip-enable=0 \
 ! filesink location=tmp.264
```

4.4.3 Testsrc (RGB) to Resize to File (YCbCr)

```
gst-launch \
 videotestsrc \
 ! "video/x-raw-rgb, bpp=16, width=160, height=120" \
 ! gst-sh-mobile-resize \
 ! "video/x-raw-yuv, width=320, height=240" \
 ! filesink location=tmp.yuv
```

4.4.4 Test ALSA audio output (single tone)

```
gst-launch audiotestsrc is-live=1 ! alsasink
```

4.4.5 Test ALSA audio input (WAV file)

```
gst-launch -e \
 alsasrc ! audio/x-raw-int, rate=44100, channels=2 \
 ! wavenc ! filesink location=test.wav
```

4.4.6 Test ALSA audio output (WAV file)

```
gst-launch filesrc location=./test.wav ! wavparse ! alsasink
```

4.4.7 Testsrc to Mixer to File (YCbCr)

# Testsrc =/

```
gst-launch \
 videotestsrc pattern=1 ! "video/x-raw-yuv, format=(fourcc)NV12,
framerate=(fraction)10/1, width=320, height=240" ! queue ! mix. \
 videotestsrc           ! "video/x-raw-yuv, format=(fourcc)NV12,
framerate=(fraction)5/1,  width=100, height=100" ! queue ! mix. \
 gst-sh-mobile-mixer name=mix sink_1::alpha=0.5 \
  ! "video/x-raw-yuv, format=(fourcc)NV12" \
  ! filesink location=tmp.yuv
```

4.4.8 Testsrc to Mixer to File (YCbCr)

# Testsrc =/

```
gst-launch \
 videotestsrc pattern=1 ! "video/x-raw-rgb, bpp=32, framerate=(fraction)10/1, width=320,
height=240" ! queue ! mix. \
 videotestsrc           ! "video/x-raw-yuv, format=(fourcc)NV12,
framerate=(fraction)5/1,  width=100, height=100" ! queue ! mix. \
 gst-sh-mobile-mixer name=mix sink_1::alpha=0.5 sink_1::xpos=100 sink_1::ypos=50\
  ! "video/x-raw-yuv, format=(fourcc)NV12" \
  ! filesink location=tmp.yuv
```

Website and Support

Renesas Electronics Website
    http://www.renesas.com/

Inquiries
    http://www.renesas.com/inquiry

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Jan.26.11	—	First edition issued

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this manual, refer to the relevant sections of the manual. If the descriptions under General Precautions in the Handling of MPU/MCU Products and in the body of the manual differ from each other, the description in the body of the manual takes precedence.

1. Handling of Unused Pins
   Handle unused pins in accord with the directions given under Handling of Unused Pins in the manual.
   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.
2. Processing at Power-on
   The state of the product is undefined at the moment when power is supplied.
   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.
3. Prohibition of Access to Reserved Addresses
   Access to reserved addresses is prohibited.
   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.
4. Clock Signals
   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.
   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.
5. Differences between Products
   Before changing from one product to another, i.e. to one with a different type number, confirm that the change will not lead to problems.
   — The characteristics of MPU/MCU in the same group but having different type numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different type numbers, implement a system-evaluation test for each of the products.

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
7F, No. 363 Fu Shing North Road Taipei, Taiwan, R.O.C.
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
1 harbourFront Avenue, #06-10, keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141