i.MX Linux Multimedia Framework

User's Guide

Document Number: 924-76335
Rev. 1.6
08/2009

freescale™
semiconductor

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064, Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
reescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 010 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Contents

About This Book

This document describes the package contents and provides instructions for building the libraries that are based on the Gstreamer architecture. Gstreamer is a powerful, versatile framework for creating streaming media applications.

Audience

This document is intended for software, hardware, and system engineers who are planning to use the Multimedia codecs with Gstreamer architecture and for anyone who wants to understand more about the Multimedia codecs. You need to have a basic understanding of Gstreamer and LTIB architecture.

Organization

This document contains the following chapters.

Chapter 1 Identifies the BSP requirements, and explains how to build the multimedia components from LTIB.

Chapter 2 Explains how to test the multimedia codecs.

Conventions

This document uses the following conventions:

`Courier`	Is used to identify commands, explicit command parameters, code examples, expressions, data types, and directives.
Italic	Is used for emphasis, to identify new terms, and for replaceable command parameters.

References

The following documents were referenced to build this document.

1. i.MX Linux User’s Guide

2. i.MX Linux Multimedia Framework Release Notes

3. i.MX Advanced ToolKit Standard User’s Guide

Definitions, Acronyms, and Abbreviations

The following list defines the abbreviations used in this document.

FSL	**F**ree**s**ca**l**e
Codec	**co**der-**dec**oder
LTIB	**L**inux **T**arget **I**mage **B**uilder
ARM	**A**dvanced **R**ISC **M**achine
ASRC	**A**synchronous **S**ample **R**ate **C**onverter

Chapter 1
Installing and Building the Plugins

This chapter describes the BSP requirements, and explains how to use the LTIB to build the multimedia codec plugins. You will need to install LTIB, extract the package files, and build the package.

1.1 BSP Requirements

To use the Freescale Multimedia Linux codecs, you will need the following:

NOTE

The Freescale Multimedia Linux codecs, which are based on Gstreamer architecture, include the Gstreamer Core, Good Plugins, and Base Plugins.

Requirements:

- i.MX 3-Stack board
- Compliant i.MX 3-Stack Linux BSP v4.4.0 or above.
- Gstreamer
  - Gstreamer (version 0.10.22)
  - Gstreamer-plugins-base (version 0.10.22)
  - Gstreamer-plugins-good (version 0.10.14)

1.2 Building the Plugins

To install LTIB and extract the package files, use these steps:

1. Install LTIB.

   For instructions, see the *i.MX Linux User's Guide* for your platform.

2. Obtain the following packages, which are included in the release.

   There are two packages for building the Freescale multimedia Linux codecs.

   - `*codecs*$VERSION.tar.gz` is the **gstreamer plugin source package** that contains source code for the multimedia Gstreamer-based plugin for the i.MX application processor.
   - `*plugins*$VERSION.tar.gz` is the **codec/parser binary package** that contains the Freescale multimedia core codec/parser libraries for the i.MX application processor.

   NOTE

   These two packages MUST be compliant with LTIB specifications.

3. Extract these two packages.

   Each package contains a `.tar.gz` file and a `.spec` file.

4. Copy the extracted `.tar.gz` files to the LPP directory, which by default is set to `/opt/freescale/pkgs/`, and replace the `.spec` files in the `$LTIB_PATH/dist/lfs-5.1/fsl-mm/` with the extracted `.spec` files.

   NOTE

   For the first LTIB installation, you must create this directory manually.

To build the package, use these steps:

1. Go to LTIB setup directory and run `./ltib -c`.

   The LTIB Configuration Menu is displayed (Figure 1).

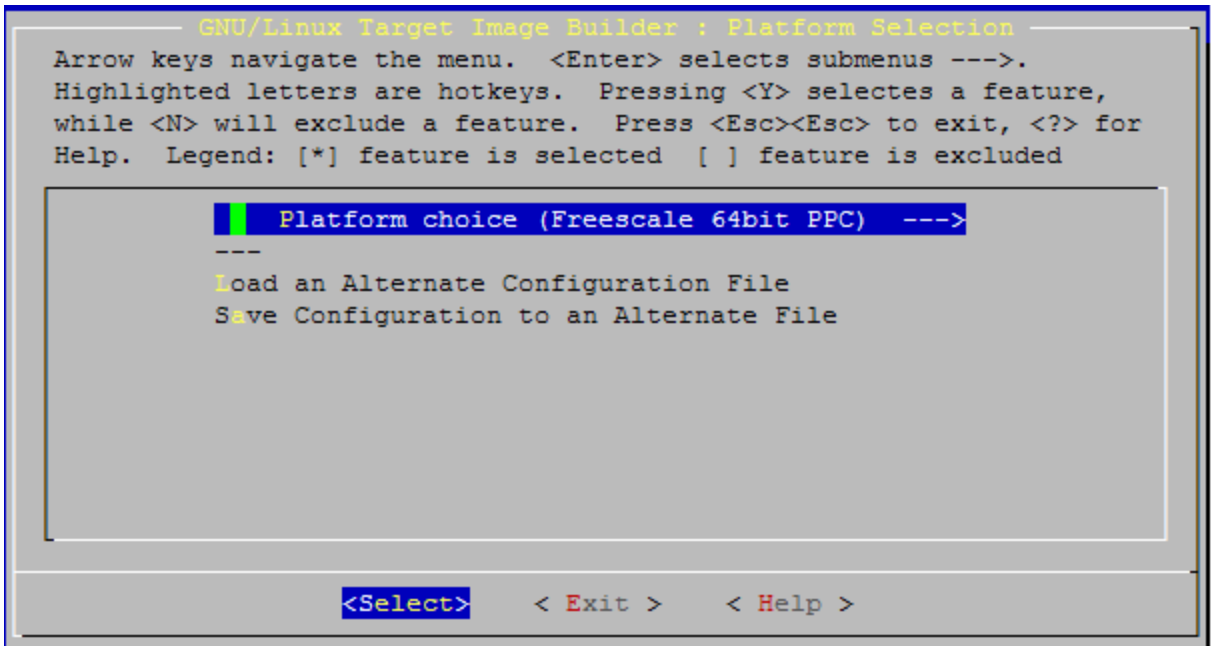

Figure 1 Configuration Menu

2. Select the platform.

   The Freescale board setup menu is displayed (Figure 2).

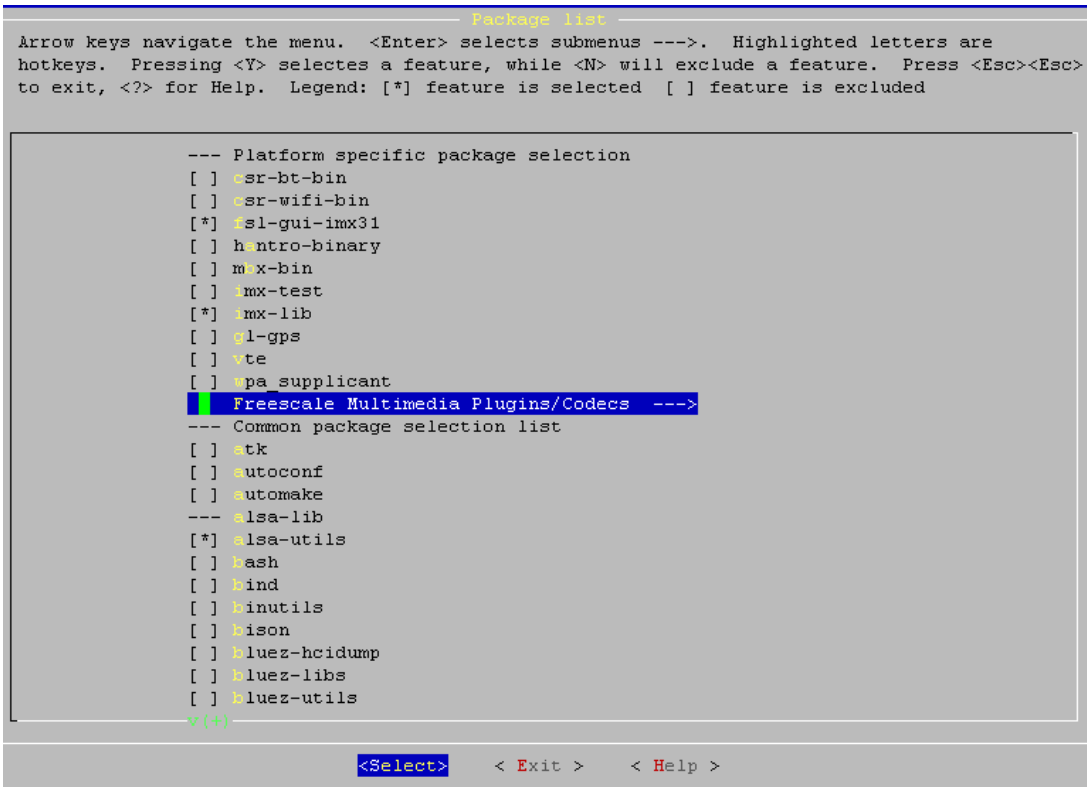

Figure 2 LTIB Package Selection Menu

.

3. Select **Package List > Freescale Multimedia Plugins/Codecs**.

4. Select the **fsl-mm-codec-libs** and **gstreamer-fsl-plugins** (Figure 3).

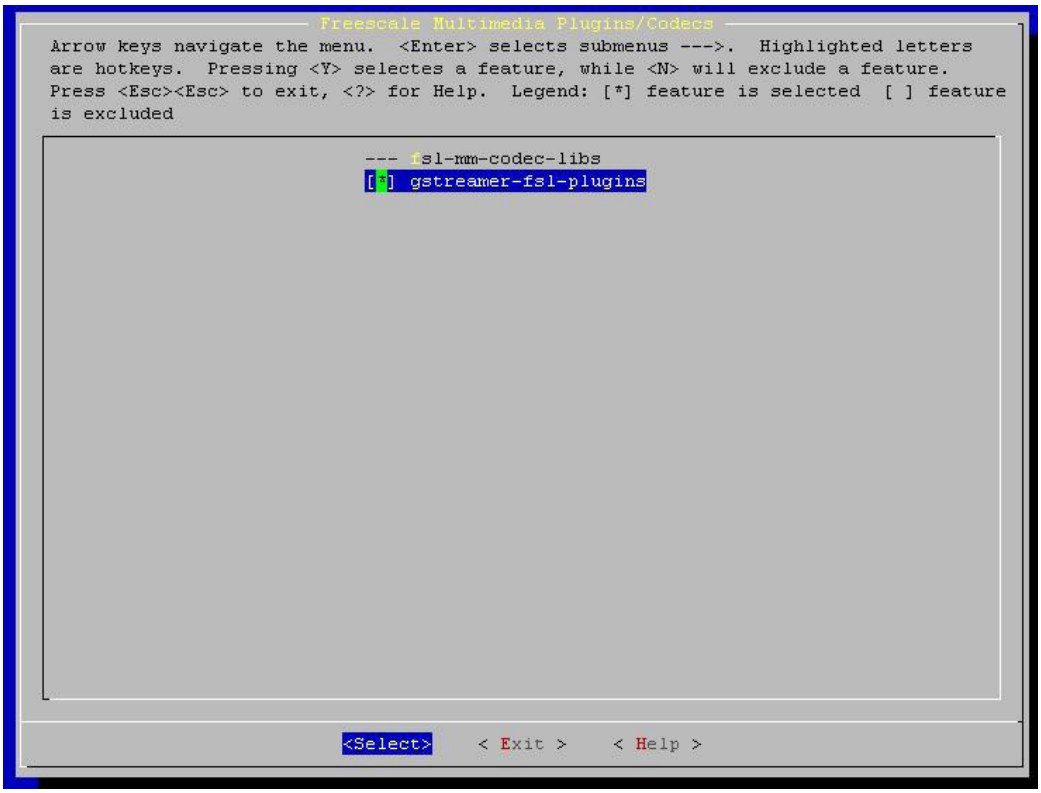

Figure 3 Selecting the Plugins

5. Select **gstreamer-plugins-good** package (Figure 4)

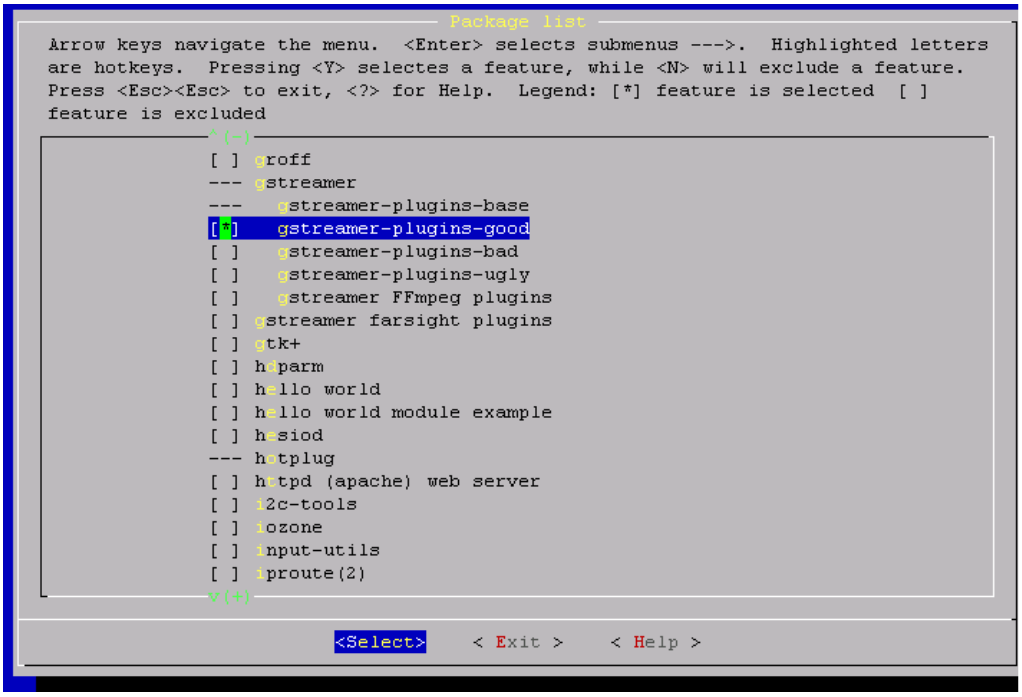

Figure 4 Selecting gstreamer-plugins-good package

6. Follow the LTIB compilation instructions.

   After a successful build, two binaries will be created: **zImage** and **rootfs**. Now **rootfs** includes the Freescale multimedia Linux codecs.

Chapter 2
Testing the Installation

This chapter explains how to check and test the multimedia codecs (audio decoder, audio encoder, video decoder and video encoder). It also explains how to enable the post-process filter to the pipeline that is being created in the Gstreamer architecture.

NOTE

Each platform provides a select set of codecs. To determine which codecs are included in your BSP, see the Release Notes.

2.1 Setting Up the Environment

To test the multimedia codecs, you need to use **rootfs** and **zImage**. If you have performed the instructions in Chapter 2, **rootfs** now includes the multimedia Linux codecs.

For instructions on using **rootfs** and **zImage**, see the *i.MX Linux User's Guide*.

2.2 Testing the Codecs with Gstreamer

Gstreamer provides two useful applications for testing multimedia codecs: **gst-inspect**, and **gst-launch**.

2.2.1 gst-inspect Tool

The **gst-inspect** tool can provide information about an available Gstreamer plugin, a particular plugin, or a particular element.

To view the list of installed multimedia codec plugins, type the following command in a shell:

```
gst-inspect | grep mfw
```

A list similar to the following is displayed.

```
mfw_asfdemuxer:  mfw_asfdemuxer: FSL Asf Demuxer
mfw_h264decoder:  mfw_h264decoder: Freescale    H264 decoder
mfw_mpeg4aspdecoder:  mfw_mpeg4aspdecoder: Freescale MPEG4 Decoder
mfw_mp4demuxer:  mfw_mp4demuxer: Freescale-Mp4 demuxer plugin
mfw_mpeg2decoder:  mfw_mpeg2decoder: Freescale MPEG2 Decoder
mfw_audio_pp:  mfw_audio_pp: Freescale Audio Post-process Filter
mfw_avidemuxer:  mfw_avidemuxer: FSL Avi Demuxer
mfw_aacdecoder:  mfw_aacdecoder: Freescale AAC Decoder Plugin
mfw_mpg2demuxer:  mfw_mpg2demuxer: Freescale MPEG demuxer
mfw_wmv9mpdecoder:  mfw_wmv9mpdecoder: Freescale wmv decoder
mfw_mp3encoder:  mfw_mp3encoder: freescale mp3 encoder
mfw_spdiftx:  mfw_spdiftx: Freescale SPDIF Transmit Converter
mfw_mp3decoder:  mfw_mp3decoder: freescale mp3 decoder
mfw_spdifrx:  mfw_spdifrx: Freescale SPDIF Receiver Converter
mfw_audiosrc:  mfw_audiosrc: Freescale Audio Sampling Rate Converter
mfw_downmixer:  mfw_downmixer: Freescale Audio Down Mixer
mfw_wma10decoder:  mfw_wma10decoder: Freescale's wma10 decoder
mfw_wmvdecoder:  mfw_wmvdecoder: Freescale wmv decoder
mfw_v4lsink:  mfw_v4lsink: Freescale: v4l_sink
mfw_v4lsrc:  mfw_v4lsrc: Freescale Video Source plug-in
mfw_aacplusdecoder:  mfw_aacplusdecoder: Freescale AAC Decoder Plugin
mfw_wma8encoder:  mfw_wma8encoder: freescale wma8 encoder
```

To view detailed information about a plugin, enter the following command in a shell:

```
gst-inspect plugin_name
```

2.2.2 gst-launch Tool

The **gst-launch** tool builds and runs the basic Gstreamer pipeline.

2.2.2.1 Audio playback

Use the commands that follow to test the MP3 playback, AAC playback, and WMA playback.

To test the MP3 audio playback, use the following command:

```
gst-launch filesrc location=test.mp3 ! queue max-size-time=0 ! mfw_mp3decoder ! audioconvert !
alsasink
```

To test the AAC audio playback, use the following command:

```
gst-launch filesrc location=test.aac ! queue max-size-time=0 ! mfw_aacdecoder ! audioconvert !
alsasink
```

To test the WMA audio playback, use the following command:

```
gst-launch filesrc location=test.wma ! mfw_asfdemuxer ! queue max-size-time=0 ! mfw_wma10decoder
! audioconvert ! alsasink
```

To test the M4A audio playback, use the following command:

```
gst-launch filesrc location=test.m4a ! mfw_mp4demuxer ! queue max-size-time=0 !
mfw_aacplusdecoder ! audioconvert ! alsasink
```

To test the WAV audio playback, use the following command:

NOTE

For this test, the Gstreamer Good Plugin package must be installed.

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

2.2.2.2 Video only playback

To create a video-only pipeline with the gst-launch tool, use these commands:

```
gst-launch filesrc location= test.video ! demuxer_plugin ! queue max-size-time=0 !
video_decoder_plugin ! mfw_v4lsink
```

For example, for an ASF(WMV only) file playback, use this command:

```
gst-launch filesrc location=test.asf ! mfw_asfdemuxer ! queue max-size-time=0 ! mfw_wmvdecoder !
mfw_v4lsink
```

2.2.2.3 AV file playback

To create a audio/video combined pipeline with the gst-launch tool, use these commands.

```
gst-launch filesrc location=test_file ! demuxer_plugin name=demux demux. !
queue max-size-buffers=0 max-size-time=0 ! video_decoder_plugin ! mfw_v4lsink demux. !
queue max-size-buffers=0 max-size-time=0 ! audio_decoder_plugin ! audioconvert ! alsasink
```

In VPU mode, change video_decoder_plugin to mfw_vpu_decoder. The VPU mode is only used for the Freescale i.MX SoC with embedded VPU.

The **max-size-time** in Queue element should be set because the playback could be not smoothly with default value one second.

Example commands

The following commands are examples for different parsers and codecs:

AVI(H264+MP3) video playback

```
gst-launch filesrc location=test.avi ! avidemux name=demux demux. !
queue max-size-buffers=0 max-size-time=0 ! mfw_h264decoder ! mfw_v4lsink demux. !
queue max-size-buffers=0 max-size-time=0 ! mfw_mp3decoder ! audioconvert ! alsasink
```

MP4(MPEG4+AAC) video playback

```
gst-launch filesrc location=test.mp4 ! mfw_mp4demuxer name=demux demux. !
queue max-size-buffers=0 max-size-time=0 ! mfw_mpeg4aspdecoder ! mfw_v4lsink demux. !
queue max-size-buffers=0 max-size-time=0 ! mfw_aacplusdecoder ! audioconvert ! alsasink
```

ASF(WMV9+WMA) video playback

```
gst-launch filesrc location=test.asf ! mfw_asfdemuxer name=demux demux. !
queue max-size-buffers=0 max-size-time=0 ! mfw_wmv9mpdecoder ! mfw_v4lsink demux. !
queue max-size-buffers=0 max-size-time=0 ! mfw_wma10decoder ! audioconvert ! alsasink
```

ASF(WMV7/WMV8+WMA) video playback

```
gst-launch filesrc location=test.asf ! mfw_asfdemuxer name=demux demux. !
queue max-size-buffers=0 max-size-time=0 ! mfw_wmvdecoder ! mfw_v4lsink demux. !
queue max-size-buffers=0 max-size-time=0 ! mfw_wma10decoder ! audioconvert ! alsasink
```

MPEG1 system stream video playback

```
gst-launch filesrc location=test.mpeg ! mfw_mpg2demuxer name=demux demux. !
queue max-size-buffers=0 max-size-time=0 ! mfw_mpeg2decoder ! mfw_v4lsink demux. !
queue max-size-buffers=0 max-size-time=0 ! mfw_mp3decoder ! audioconvert ! alsasink
```

VPU + Deinterlace video playback

```
gst-launch filesrc location=test_file ! demuxer_plugin name=demux demux. !
queue max-size- buffers=0 max-size-time=0 ! video_decoder_plugin ! mfw_deinterlace ! mfw_v4lsink
demux. ! queue max-size-buffers=0 max-size-time=0 ! audio_decoder_plugin ! audioconvert !
alsasink
```

NOTE

The VPU decoder is currently available only for the Freescale i.MX SoC with embedded VPU.

2.2.2.4 MPEG4 Hantro Encode Record

Use these steps:

7. To perform the initial setup,
   a) Insert the **memalloc.ko** kernel module with the **insmod** command.
   b) Check that the **memalloc** module is present using the **lsmod** command.

```
$ insmod memalloc.ko
$ lsmod
```

8. Create the `/dev/memalloc` device using the following commands:

```
$ cat /proc/devices | grep memalloc
$ mknod /dev/memalloc c 244 0
```

9. Run the encoder using the following commands:

```
gst-launch-0.10 filesrc blocksize=38016 location=yuv_file !
'video/x-raw-yuv,format=(fourcc)I420,width=176,height=144,framerate=(fraction)25/1' !
 mfw_mpeg4encoder bitrate=200000 scheme=0 ! filesink location=outstream.bits
```

NOTES

For i.MX31 3-Stack board, **memalloc.ko** will be built in **rootfs**. For more information, see the *i.MX Linux User's Guide*.

The **memalloc** device uses a dynamic major number. The first command displays the dynamic number used. For this example, the dynamic number generated was 244. You should use the dynamic number generated in the second command, rather than 244.

The **blocksize** property of the **filesrc** plugin depends on the resolution of the input image. For example:

`blocksize = inputwidth * inputheight * 1.5.`

You must change the width and height of the **mpeg4** encoder plugin to match the resolution of the mandatory input image.

2.2.2.5 Audio Encoder Record

This release provides two audio encoders: MP3 and WMA8. You may enable either or both.

MP3 Encoder Record

```
gst-launch filesrc location=test.wav ! wavparse ! mfw_mp3encoder ! filesink location=output.mp3
```

To verify that the MP3 output is correct, use the **mfw_mp3decoder**:

```
gst-launch filesrc location=output.mp3 ! queue max-size-time=0 ! mfw_mp3decoder ! audioconvert !
alsasink
```

WMA8 Encoder Record

Encoding from file:

```
gst-launch filesrc location=test.wav ! wavparse ! mfw_wma8encoder ! filesink location=output.wma
```

Recording:

```
gst-launch alsasrc num-buffers=100*time ! mfw_wma8encoder ! filesink location=output.wma
```

where: **Time** is the recording time in seconds.

To verify that the WMA output is correct, use the **mfw_wma10decoder**:

```
gst-launch filesrc location=output.wma ! mfw_asfdemuxer ! queue max-size-time=0 !
mfw_wma10decoder ! audioconvert ! alsasink
```

2.2.2.6 VPU based Video Encoder Record

NOTE

The VPU encoder is currently available only for some of the Freescale i.MX SoC with embedded VPU.

You need enable camera before can run video recoder, for the camera driver install, please refer BSP document.

Use “modprobe mxc_v4l2_capture” to enable v4l2 capture interface.

Encoding from file:

```
gst-launch filesrc location=test.yuv blocksize=115200 ! mfw_vpuencoder codec-type=0 ! avimux !
filesink location=output.avi sync=false
```

NOTE

The input file support I420 format YUV files.

The **blocksize** property of the **filesrc** plugin depends on the resolution of the input image. For example:

```
blocksize = inputwidth * inputheight * 1.5.
```

The **codec-type** property of the **mfw_vpuencoder** plugin control the target encode codec type. It could be 0(MPEG4), 1(H263), 2(H264) or 7(MJPG).

Recording:

```
gst-launch mfw_v4lsrc fps-n=15 ! queue ! mfw_vpuencoder codec-type=0 ! avimux ! filesink
location=output.avi sync=false
```

NOTE

The **fps-n** property of the **mfw_v4lsrc** plugin control the camera capture frame rate, it should be 15 or 30.

The **codec-type** property of the **mfw_vpuencoder** plugin control the target encode codec type. It could be 0(MPEG4), 1(H263), 2(H264) or 7(MJPG).

2.2.2.7 SPDIF Transmit and Receive Converter

The SPDIF supports both transmit and receive feature with PCM or Non-PCM data. With Non-PCM data, the **mfw_spdifrx** and **mfw_spdiftx** plugin convert data between the IEC958 format and raw data. In this version, only support AC3 data format.

To verify the SPDIF receive is correct, use the **mfw_spdifrx**:

```
gst-launch alsasrc device="plughw:1,0" ! mfw_spdifrx ! filesink location= test.bits
```

NOTES

The SPDIF feature is applied in i.MX35 platform. For more information, see the *i.MX Linux User's Guide* for your platform.

To verify the SPDIF transmit is correct, use the **mfw_spdiftx**:

```
gst-launch filesrc location= test.bits  ! mfw_spdiftx ! alsasink device="plughw:1,0"
```

NOTE

Insert the snd-spdif.ko kernel module with the **insmod** command. For i.MX35 3-Stack board, **snd-spdif** module will be built in **rootfs**. For more information, see the *i.MX Linux User's Guide*.

The "plughw" parameter depends on your system.

2.2.2.8 Audio Post-Process

To verify the Parametric EQ is correct, use the **mfw_audio_pp**:

```
gst-launch filesrc location=test.mp3 ! queue ! mfw_mp3decoder ! mfw_audio_pp enable=1 eqmode=2 !
alsasink
```

NOTE

The eqmode value 2 means the “bass booster” scene.

To verify the ASRC is correct, use the **mfw_audiosrc**:

```
gst-launch filesrc location= test.mp3 ! mfw_mp3decoder ! mfw_audiosrc use-ASRC=1 out-rate=32000
asrc-outclk=1 ! alsasink


gst-launch filesrc location=test.mp3 ! mfw_mp3decoder ! mfw_audiosrc use-ASRC=1 out-rate=48000
asrc_outclk=0 ! capsfilter caps="audio/x-raw-int, channels=2, samplerate=48000" ! wavenc !
filesink location= ./output.wav
```

NOTES

The ASRC is currently available only for i.MX35 Platform.

The supported input rates and output rates are 32000, 44100, 48000.

The **asrc-outclk** only support OUTCLK_SSI1_TX.

With the filesink, the **asrc-outclk** should be 0**.**

To verify the Downmixing is correct, use the **mfw_downmixer**:

```
gst-launch filesrc location= test.mp3 ! mfw_mp3decoder ! mfw_downmixer ochannels=2 ! alsasink
```

2.3 Testing the Core Codec Libraries

Some core codec libraries have no corresponding Gstreamer plugins, such as the **image** and some **audio encoders**. To view the list of Gstreamer plugins, see the *i.MX Multimedia Framework Linux Release Notes.*

To test those core codec libraries, you must use the Freescale proprietary test applications that are included in codec/parser binary package.

2.4 Debug exception in multimedia plugin

In the GDB debug mode, some multimedia plugin might generate exceptions on their system check initialization but are safe to continue since the exceptions are handled directly by the multimedia components. This might disturb your debug environment with processing these exceptions. The following step specifies how to configure your debugger so that these exceptions are handled automatically without user input needed.

```
$handle SIGBUS nostop
```

You also can add this command to .gdbinit script as the default setting for debug the multimedia plugins.