i.MX Linux Multimedia Framework

User's Guide

Document Number: 924-76335
Rev. 1.7
12/2009

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:
Freescale 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 or install multimedia components on Ubuntu OS
Chapter 2	Explains how to test and use 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. For replaceable command parameters it will start with *$*.

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
APT	**A**dvanced **P**ackaging **T**ool
Gst	Gstreamer (open source multimedia framework)
gplay	Freescale command line player with Gstreamer backend

Chapter 1
Installing and Building the Plugins

This chapter describes how to build/install Freescale multimedia core libraries and Freescale Gstreamer plugins.

Freescale multimedia core libraries are released in binary only. Freescale Gstreamer plugins include source code.

Freescale provides two types of release packages; LTIB packages are for Freescale core libraries and Gstreamer plugins, while Debian packages are for Ubuntu system.

The LTIB packages contain Freescale multimedia core binary libraries and Freescale Gstreamer plugins source code.

Debian binary packages are used to install Freescale core libraries and Gstreamer plugins binaries into an i.MX51 Babbage board running Ubuntu OS. Debian source packages are used to build Freescale Gstreamer plugins on an i.MX51 Babbage board.

1.1 Building the Plugins with LTIB

1.1.1 BSP Requirements

Requirements:

- i.MX series board
- Compliant i.MX series Linux BSP 09.12.0 or above.
- Gstreamer
  - Gstreamer (version>=0.10.22)
  - Gstreamer-plugins-base (version>=0.10.22)
  - Gstreamer-plugins-good (version>=0.10.14)

NOTE

The Freescale Gstreamer plugins are dependent on the Gstreamer framework including the Gstreamer Core, Gst-Plugins-base, and Gst-Plugins-good.

1.1.2 Building the Plugins with LTIB

Following LTIB related procedures are running on a PC(x86) with a Linux OS.

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

1. Install LTIB on PC.

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

2. Obtain the following packages included in the release

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

   Standard packages:
   - gst-fsl-plugin-$VERSION.tar.gz is **gstreamer plugin source package** that contains source code for the multimedia Gstreamer-based plugin for the i.MX application processor.
   - fsl-mm-codeclib-$VERSION.tar.gz is **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. Copy these two standard packages to the LPP directory, which by default is set to `/opt/freescale/pkgs`.

   NOTE

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

To build the package, please follow these steps:

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

   The LTIB Configuration Menu is displayed (Figure 1).

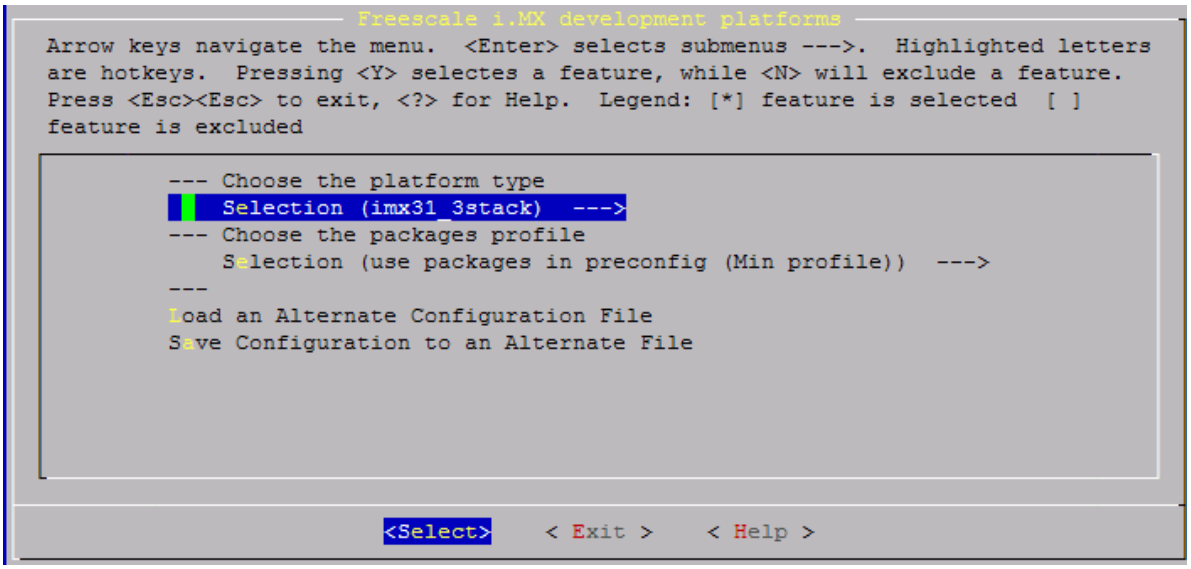

Figure 1 Configuration Menu

2. Select the platform.

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

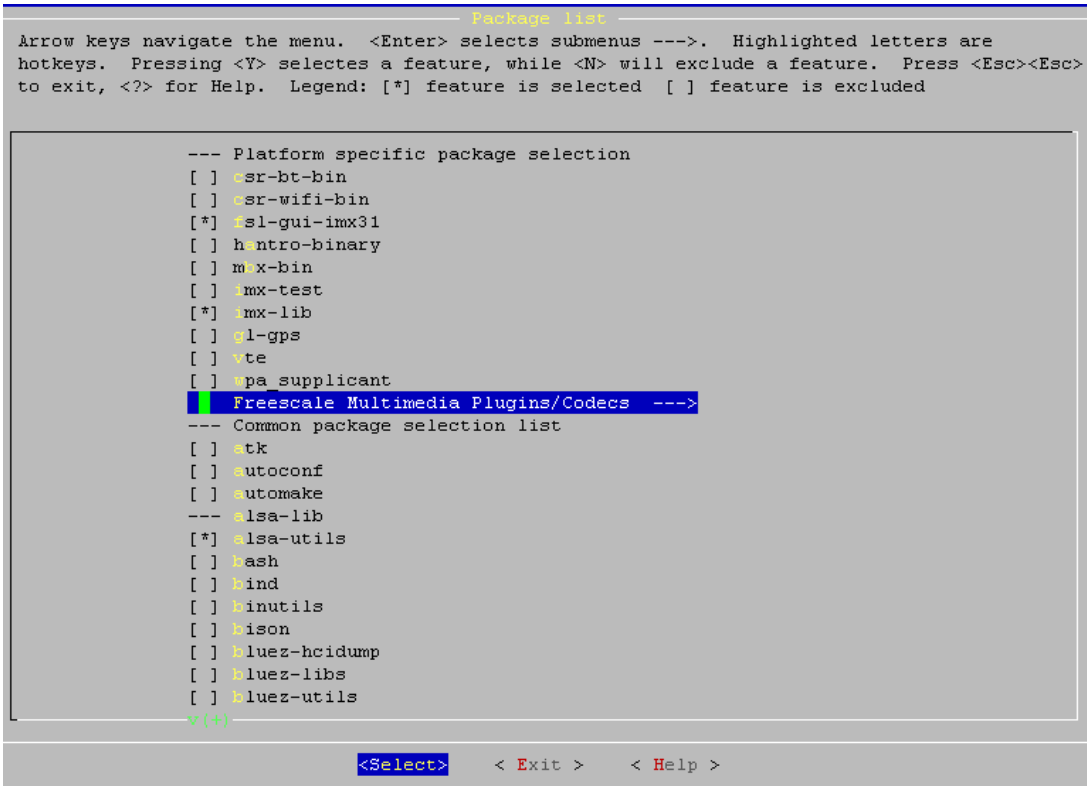

Figure 2 LTIB Package Selection Menu

.

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

4. Select the **gstreamer-fsl-plugins, the fsl-mm-codec-libs** will be automatically selected. (Figure 3).

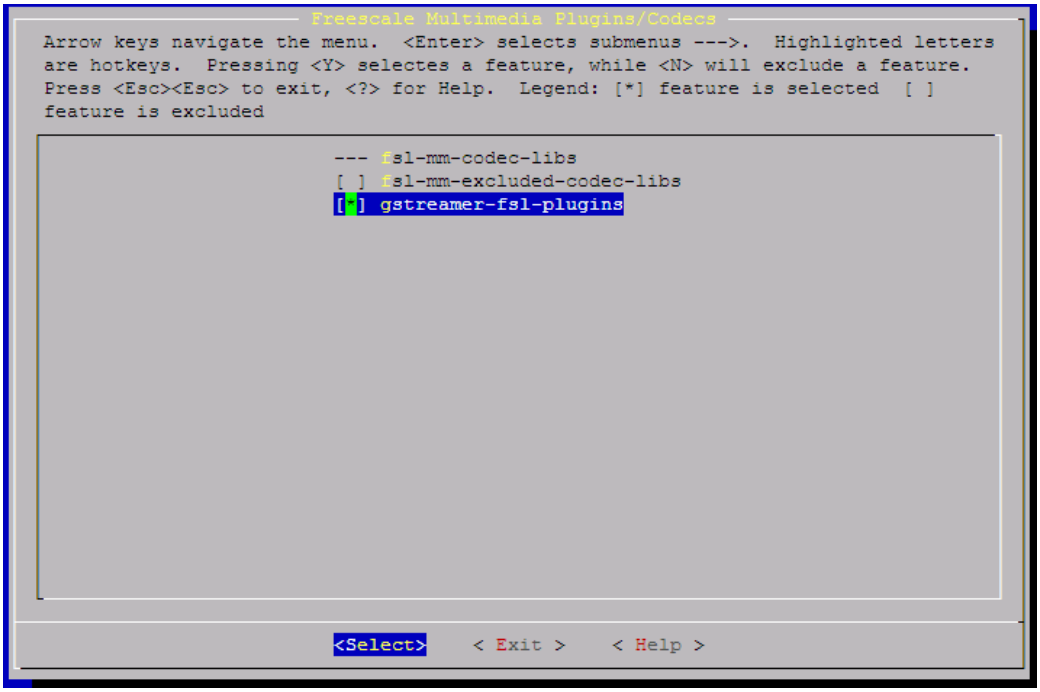

Figure 3 Selecting the Plugins

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

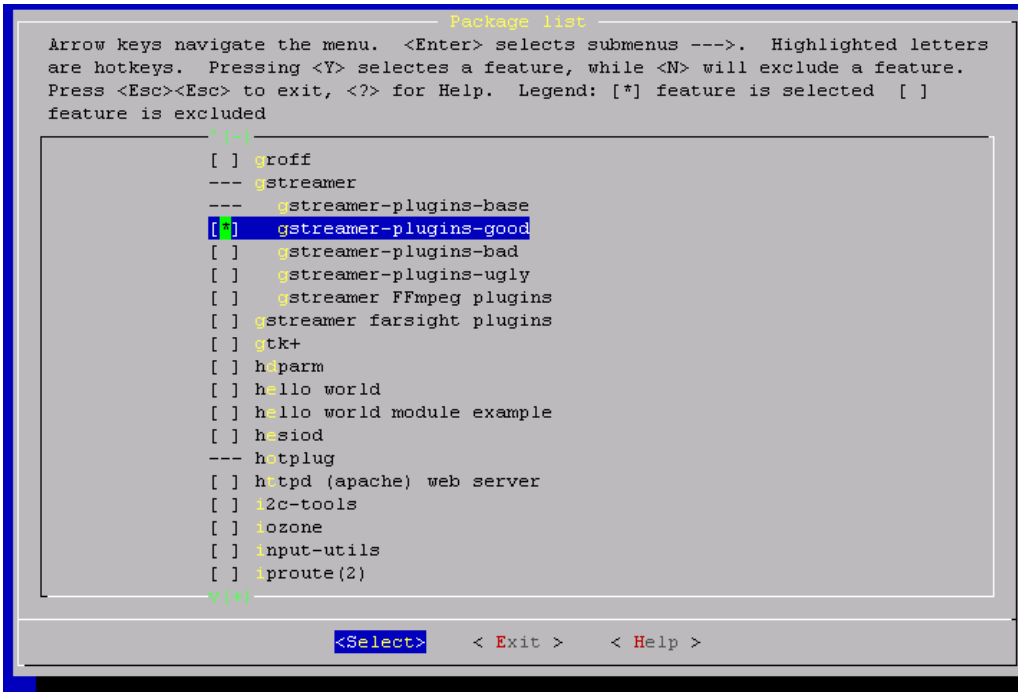

Figure 4 Selecting gstreamer-plugins-good package

6. Follow the LTIB compilation instructions.

After a successful build, zImage and rootfs will be created. The rootfs is located in LTIB directory. It includes the Freescale multimedia Linux codecs and Freescale Gstreamer plugins. The zImage is located in rootfs/boot in LTIB build directory.

1.2 Installing/Building the Plugins on Ubuntu

1.2.1 BSP Requirements

Requirements:

- i.MX51 Babbage board runs Ubuntu OS (9.10).
- imx-lib Debian package (version>=09.12.00).
- Gstreamer
  - Gstreamer (version=0.10.25 on Ubuntu 9.10)
  - Gstreamer-plugins-good (version=0.10.16 on Ubuntu 9.10)

1.2.2 Installing/Building the Plugins

The Freescale Multimedia core libraries and Freescale Gstreamer plugins are released in Debian release package format. Ubuntu OS uses APT to manage all these packages. For detail information about how to use/manage packages by APT, please refer to http://www.debian.org/doc/manuals/apt-howto/index.en.html.

Following steps illustrate how to install Freescale Multimedia Plugins on Ubuntu and also how to build a Debian package from the source.

1. Prepare an i.MX51 Babbage board running Ubuntu OS(9.10)
2. Install BSP support libraries package, the package is released with BSP.

   sudo dpkg -i imx-lib-$VERSION-$RELEASE.deb

3. Install dpkg-dev package by running command.

   sudo aptitude install dpkg-dev

4. Obtain the following packages, which are included in the Debian release.
   - libgstreamer-plugins-base0.10-0_$VERSION-$RELEASE_armel.deb is **Gst-plugins-base libraries** modified by Freescale
   - gstreamer0.10-plugins-base_$VERSION-$RELEASE_armel.deb is **Gst-plugins-base binary** modified by Freescale to enable playbin/playbin2 compatible with Freescale Gstreamer plugins. This package is needed for playback with a player such as gplay or totem.
   - libfsl-mm-core1_$VERSION-$RELEASE _armel.deb is **Freescale Multimedia core binaries** that contains the Freescale multimedia core codec/parser libraries for the i.MX application processor.

- libgst-fsl-mm-plugins_$VERSION-$RELEASE _armel.deb is **Freescale Multimedia gstreamer plugin binary package** which contains binaries for the multimedia Gstreamer-based plugin for the i.MX application processor.

To build the codec and plugin deb packages from source, following packages are also needed.

- libgstreamer-plugins-base0.10-dev_$VERSION-$RELEASE_armel.deb is the **Gst-plugins-base** development support package.
- libfsl-mm-core-dev_$VERSION-RELEASE_armel.deb is the **Freescale Multimedia core libraries** development support package.
- fsl-mm-codeclib-$VERSION.orig.tar.gz, fsl-mm-codeclib-$VERSION-$RELEASE.diff.gz, fsl-mm-codeclib-$VERSION-$RELEASE.dsc are **codec/parser binary packages** that contain the Freescale multimedia core codec/parser libraries for the i.MX application processor.
- gst-fsl-plugin-$VERSION.orig.tar.gz, gst-fsl-plugins_$VERSION-$RELEASE.diff.gz, gst-fsl-plugins_$VERSION-$RELEASE.dsc are **gstreamer plugin source packages** that contain source code of the multimedia Gstreamer-based plugin for the i.MX application processor.

5. Create a local directory /root/debs in the rootfs and copy all these packages to this directory.
6. Change directory to /root and run following commands to generate package/source list files for these packages.

```
sudo dpkg-scanpackages debs | gzip > debs/Packages.gz

sudo dpkg-scansources debs | gzip > debs/Sources.gz
```

7. Add following lines to /etc/apt/sources.list

```
deb file:/root debs/

deb-src file:/root debs/
```

8. Run following command to update local package list.

```
sudo aptitude update
```

9. Install **Gst-plugins-base** by following commands.

```
sudo aptitude install libgstreamer-plugins-base0.10-0

sudo aptitude install gstreamer0.10-plugins-base
```

10. Install **Gst-plugins-good** by following command.

```
sudo aptitude install gstreamer0.10-plugins-good
```

To build the codec and plugin deb packages from source, please jump to step 13.

11. Install Freescale Multimedia core libraries by following command

```
sudo aptitude install libfsl-mm-core1
```

12. Install Freescale Multimedia gstreamer plugins by following command

```
sudo aptitude install libgst-fsl-mm-plugins
```

NOTE

This version of multimedia debian package only compatible with GCC 4.4.1 related rootfs. In order to applying the multimedia plugins with other type of rootfs such as Ubuntu 9.04, these debian packages need be recompiled in Ubuntu 9.04 with GCC version 4.3.3

The Freescale Multimedia core libraries and Freescale Gstreamer plugins are now successfully installed into your i.MX Babbage Ubuntu rootfs. Ignore following steps if do not want to build Debian packages from source.

13. Install BSP development support package which is released with BSP.

```
sudo dpkg -i kernel_$VERSION-imx_$RELEASE_armel.deb
```

14. Install Gstreamer build environment by following command.

```
apt-get build-dep gstreamer0.10-plugins-good
```

15. Get Freescale multimedia core libraries source

```
apt-get source libfsl-mm-core1
```

After this command executed successfully, all source packages and also a patched directory named “fsl-mm-codeclib-$VERSION” are download to current directory. Go to the directory and build it by running following command

```
debian/rules binary
```

Go to the up level directory, there are two Debian packages(.deb files), install these packages by following commands:

```
sudo dpkg -i libfsl-mm-core1_$VERSION-RELEASE_armel.deb

sudo dpkg -i libfsl-mm-core-dev_$VERSION-RELEASE_armel.deb
```

16. Get Freescale multimedia gstreamer plugins source

```
apt-get source libgst-fsl-mm-plugins
```

After this command executed successfully, all source packages and also a patched directory named “gst-fsl-plugin-$VERSION” are download to current directory. Go to the directory and build it by running following command

```
debian/rules binary
```

Go to the up level directory, there is a Debian package(.deb file), install the package by following command:

```
sudo dpkg -i libgst-fsl-mm-plugins_$VERSION-RELEASE_armel.deb
```

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 certain set of codecs. Please refer to the Release Notes to determine which codecs are included in your BSP.

2.1 Testing the Codecs with Gstreamer

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

2.1.1 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_v4lsrc:  mfw_v4lsrc: Freescale Video Source plug-in
mfw_avidemuxer:  mfw_avidemuxer: FSL Avi Demuxer
mfw_deinterlacer:  mfw_deinterlacer: Mfw De-interlace
mfw_aacplusdecoder:  mfw_aacplusdecoder: Freescale AAC Decoder Plugin
mfw_aacdecoder:  mfw_aacdecoder: Freescale AAC Decoder Plugin
mfw_wma10decoder:  mfw_wma10decoder: Freescale's wma10 decoder
mfw_wmvdecoder:  mfw_wmvdecoder: Freescale wmv decoder
mfw_vpudecoder:  mfw_vpudecoder: Freescale: Hardware (VPU) Decoder
mfw_audio_pp:  mfw_audio_pp: Freescale Audio Post-process Filter
mfw_wma8encoder:  mfw_wma8encoder: freescale wma8 encoder
mfw_v4lsink:  mfw_v4lsink: Freescale: V4L Sink
mfw_vpuencoder:  mfw_vpuencoder: Freescale: Hardware (VPU) Encoder
mfw_mp4demuxer:  mfw_mp4demuxer: Freescale-Mp4 demuxer plugin
mfw_mpg2demuxer:  mfw_mpg2demuxer: Freescale MPEG demuxer
mfw_downmixer:  mfw_downmixer: Freescale Audio Down Mixer
mfw_mp3decoder:  mfw_mp3decoder: freescale mp3 decoder
mfw_asfdemuxer:  mfw_asfdemuxer: FSL Asf Demuxer
mfw_mp3encoder:  mfw_mp3encoder: freescale mp3 encoder
```

The elements contained in this list maybe different depend on the target platform

For example:

```
mfw_asfdemuxer:  mfw_asfdemuxer: FSL Asf Demuxer
```

The first “mfw_asfdemuxer” is plugin name, the second “mfw_asfdemuxer” is element name, “FSL Asf Demuxer” is long name of element.

Use following gst-inspect command to view the detail information of an element.

```
gst-inspect $ELEMENT_NAME
```

For example,

```
gst-inspect mfw_asfdemuxer
```

to display detail information of element mfw_asfdemuxer

2.1.2 gst-launch Tool

The **gst-launch** tool builds and runs the basic Gstreamer pipeline without trick mode support.

2.1.2.1 Playback with playbin/playbin2

Freescale recommends using Gstreamer playbin or playbin2 plugins to playback audio or/and video. Playbin and playbin2 are self-constructed pipeline elements which will auto connect all necessary elements to decode a media file/resource, including source, parser, decoder and sink, etc. The command is

```
gst-launch playbin uri=$URI
```

or

```
gst-launch playbin2 uri=$URI
```

The *$URI* is Universal Resource Identifier. For a local file, URI start with file://, for example, to play a local file test.avi locate in /media directory, please use

```
gst-launch playbin uri=file:///media/test.avi
```

NOTE

To make playbin/playbin2 compatible with Freescale multimedia Gstreamer plugins, a Freescale optimized gstreamer-plugins-base package needs to be installed. This package is released in LTIB package or provided in Multimedia Debian release packages.

2.1.2.2 Audio playback

Use the following commands 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 !
‘audio/x-raw-int, channels=2’ ! 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 !
‘audio/x-raw-int, channels=2’ ! 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 ! ‘audio/x-raw-int, channels=2’ ! 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 ! ‘audio/x-raw-int, channels=2’ ! alsasink
```

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

NOTE

For this test, the Gstreamer Good Plugin package must be installed.Due to hardware and opensource element limitation, for some combine configurations of specific channels and samplerate, the sound may not be heard.

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

2.1.2.3 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
```

2.1.2.4 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 ! ‘audio/x-raw-
int, channels=2’ ! alsasink
```

In VPU mode, change video_decoder_plugin to mfw_vpudecoder. 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 ! mfw_avidemuxer 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 ! ‘audio/x-raw-int,
channels=2’ ! 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 ! ‘audio/x-raw-int,
channels=2’ ! 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 ! ‘audio/x-raw-int,
channels=2’ ! 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 ! ‘audio/x-raw-int,
channels=2’ ! alsasink
```

MPEG2 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 ! ‘audio/x-raw-int,
channels=2’ ! 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 ! mfw_vpudecoder ! mfw_deinterlacer ! mfw_v4lsink
demux. ! queue max-size-buffers=0 max-size-time=0 ! audio_decoder_plugin ! audioconvert !
‘audio/x-raw-int, channels=2’ ! alsasink
```

NOTE

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

2.1.2.5 MPEG4 Hantro Encode Record

Use these steps:

17. 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
```

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

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

19. 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.

2.1.2.6 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 !
‘audio/x-raw-int, channels=2’ ! 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=$NUMBER blocksize=$SIZE ! mfw_wma8encoder ! filesink
location=output.wma
```

The time duration of recording equals $NUMBER*$SIZE*8/ (samplerate*channel*bitwidth)

For example, to record 60 second of mono channel sample with 44.1K sample rate and 16bit width, use

```
gst-launch alsasrc num-buffers=20 blocksize=4410 ! mfw_wma8encoder ! filesink location=output.wma
```

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 ! ‘audio/x-raw-int, channels=2’ ! alsasink
```

2.1.2.7 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 capture-width=$WIDTH capture-height=$HEIGHT ! 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, currently camera ov3640 driver only support 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).

2.1.2.8 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.

2.1.2.9 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
```

2.1.3 gplay Player

gplay is a command line based player. It is based on Gstreamer playbin element and provide full functions of playback, including trick mode, video display setting etc.

The command line is

```
gplay $MEDIA_FILE
```

For detail information of gplay tool, please refer to

Gstreamer_Command-line_Player_Application_Specification.pdf.

2.1.4 totem Player

Totem is the official movie player of the GNOME desktop environment based on GStreamer, it's a graphic UI based player running on Linux desktop system. Totem is default installed on i.MX51 Babbage running Ubuntu OS or Gnome mobile OS.

For detail information of using totem, please refer to totem help.

2.2 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.

2.3 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
```

Add this command to .gdbinit script as the default setting to debug the multimedia plugins.