GM8139

AUDIO SYSTEM PROGRAMMING GUIDE

User Guide

Rev.: 1.0

Issue Date: September 2014

REVISION HISTORY

GM8139 audio system programming guide

Date	Rev.	From	To
Jan. 2014	1.0	-	Original

Printed in Taiwan 2013

All information contained in this document is subject to change without notice. The products described in this document are NOT intended for use in implantation or other life support application where malfunction may result in injury or death to persons. The information contained in this document does not affect or change Grain Media's product specification or warranties. Nothing in this document shall operate as an express or implied license or indemnity under the intellectual property rights of Grain Media or third parties. All information contained in this document was obtained in specific environments, and is presented as an illustration. The results obtained in other operating environments may vary.

THE INFORMATION CONTAINED IN THIS DOCUMENT IS PROVIDED ON AN "AS IS" BASIS. In no event will Grain Media be liable for damages arising directly or indirectly from any use of the information contained in this document.

Grain Media, Inc.
5F, No. 5, Li-Hsin Road III, Hsinchu Science Park, Hsinchu City, Taiwan 300, R.O.C.

Grain Media's home page can be found at:
http://www.grain-media.com

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

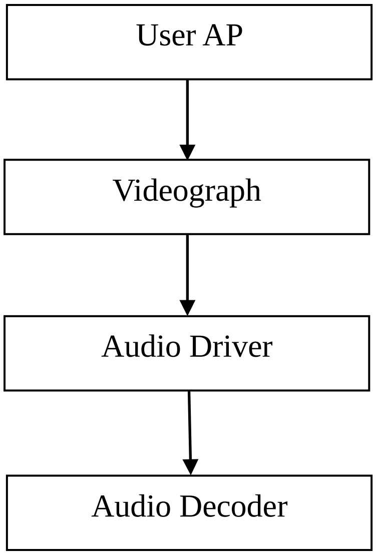
This chapter contains the following sections:

- 1.1 Overview
- 1.2 Block Diagram

1.1 Overview

Audio system porting has four parts to do, first is I2S PMU setting, then audio decoder register setting, the third is bypass function and /proc information for user, and audio driver programming.

1.2 Block Diagram



Block Diagram of audio system

Chapter 2

Audio decoder driver module

In the GM8139 SDK release package, users can find the source code and kernel module of audio decoder from “/module/front_end”.

The audio decoder driver module contains the following part:

- fe_common.ko

  This is the front end core. It includes the SSP pmu setting.

- decoder.ko

  This is the decoder core. It includes the video and audio register setting.

2.1 fe_common.ko

plat_8139.c is used to configure SSP’s main clock. At GM8139, we have 2 SSP (SSP0/SSP1) to receive audio data, both SSP0 and SSP1 are used for audio decoders. Usually, SSP’s default main clock is 12MHz, we define all SSP’s default value at PMU register 0x28 bit[15:14], in some special case, for example, if SSP1’s main clock need to change to 24MHz, then we have to recalculate the divide value from PLL2, here, we provide a specific API (pmu_cal_clk()) to accomplish it.

2.2 decoder.ko

decoder.ko has two parts to do :

A. Decoder's audio part : please refer to decoder's datasheet to configure audio CODEC.

B. /proc Node information :

The decoder module provides the /proc nodes. User can retrieve the decode information or configure the decoder through these nodes. A sample of these nodes are listed as below:

```
/ # ls /proc/adda302
ALCMAX    ALC_mode   ISMASTER  LHM  RADV    RHM     SPV
ALCMIN    BTL_mode   LADV      LIM  RDAV    RIM     input_mode
ALCNGTH   DAMIXER    LDAV      LIV  REBACK  RIV     single_end
```

Figure 2-1. Proc Nodes of decoder Module

User can use the /proc/adda302/SPV node to control the volume.

Usage:

- Get the current value

cat /proc/adda302/SPV

```
/ # cat /proc/adda302/SPV
SPV should be between -40~6dB
current SPV = 0
```

- Set the value to configure volume

echo [value] > /proc/adda302/SPV

Chapter 3

Audio driver programming

This chapter contains the following sections:

- 3.1 Overview
- 3.2 Usage of audio_drv.ko
- 3.3 Debug Audio and Conclusion

3.1 Overview

The Audio Driver is a Linux device driver linked with Videograph (it will be called as VG in the following sections) and controlling SSP and DMA for encoding low level audio raw data and decoding compressed audio data. It includes three main parts: audio_vg.c (linked with VG), audio_flow.c(audio process), ssp_dma.c(ssp/dma driver) and codec folder(encoder/decoder).

- **audio_vg.c**
    - Get job from VG and put it back when finishing audio process.
    - Construct audio_job data structure
- **audio_flow.c**
    - Main process flow of audio including recording/playing raw data and encoding/decoding audio.
- **ssp_dma.c**
    - SSP and DMA settings are in this c source code.

After compiling, you will find audio_drv.ko and codec.ko(in audio_drv/codec). The insertion order of codec.ko is previous than audio_drv.ko.

3.2 Usage of audio_drv.ko

There are 5 module parameters which will be assigned commonly when inserting audio driver. We use ADDA302 as an example. The GM8139 platform has an embedded audio decoder ADDA302, which is connected with SSP0, and user can attach an external decoder which will be connected with SSP1.

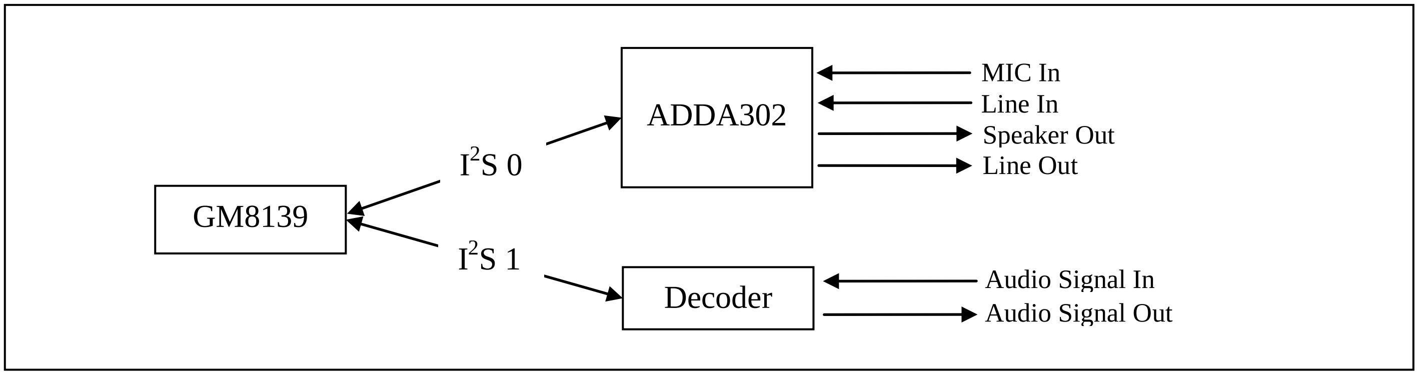


Figure 1-1 GM8139 platform with ADDA302

The following instructions illustrate how to use module parameters of audio_drv.ko.

1. *audio_ssp_num*[SSP_MAX_USED]: the SSP index number currently will used
2. *audio_ssp_chan*[SSP_MAX_USED]: the maximum audio channels for each SSP
3. *bitclock*[SSP_MAX_USED]: the bit-clock frequency of each SSP
4. *sample_rate*[SSP_MAX_USED]: the sample rate of each SSP
5. *audio_out_enable*[SSP_MAX_USED]: used to control whether the SSP can be output audio signal or not

Conclude the above illustrations, if we want to insert audio_drv.ko for GM8139 use the following command:

```
    insmod audio_drv.ko audio_ssp_num=0,1 audio_ssp_chan=1,1 bit_clock=400000,400000
sample_rate=8000,8000 audio_out_enable=1,0
```

Besides these 5 parameters, audio driver also support such as *audio_nr_enable*[SSP_MAX_USED] and *audio_is_stereo*[SSP_MAX_USED] parameters can be assigned when insert driver. These two value will leave as default in regular SDK from GM. You can add it to module list anytime if needed.

3.3 Debug Audio and Conclusion

When encountering the wrong behaviors of audio record/playback (ex. audio data error), first use the scope to check the frame sync and bit clock are set correctly (ex. ADDA302's bit clock=400000 Hz, frame sync = 8000 Hz). Second, check whether the I^{2}S settings are matched the settings of front end decoder. Third, check the module parameters of audio_drv.ko are correct or not.