Design and Implementation of Embedded WM8960 Audio Driver and Multi Thread Player

Liu Li,Wang Mingjiang, Zhao Boya,Yang Anli

Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, Guangdong, 518000, China
1263764592@qq.com

***Abstract*—This paper introduces the architecture and working principle of the audio system based on Linux operating system, which is composed of Cortex A8 microprocessor Tiny210 and WM8960 codec audio chip. Write the device driver of the audio chip, and successfully transplant it to the Linux kernel, and can work in the ARM board. The system uses I2S bus to transmit audio data and I2C bus to transmit control signal. At the same time, using ALSA framework to design a simple multi-threaded key player which can achieve the basic functions.**

Keywords-Linux; Audio Driver; Tiny210; WM8960; ALSA; I2S Bus; I2C Bus

I. INTRODUCTION

The application of the embedded system is more and more wide, and the audio structure is one of the important components, it is more attention than the traditional audio system. Embedded audio system consists of 3 parts composition including the structure of hardware, driver and upper layer of the calling program, and the driver in the entire audio system plays a connecting link between the preceding and the following. The WM8960 chip driver in many ARM development board are closed, so this paper will be the program design, and can achieve the capture and playback function. The use of hardware is mainly dependent on the I2S and I2C two buses, the kernel is Linux3.0.8, and the software part is based on the ALSA architecture.

II. WM8960 AUDIO DRIVER ARCHITECTURE

The embedded device's audio system can be divided into three parts, Machine, Platform, Codec, and the structure is shown in Figure 1.

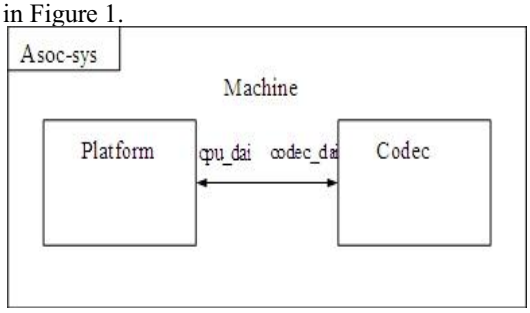

Figure 1. Audio driver structure

Machine driver is not reusable, hardware implementation of each machine are the same, CPU is not the same, codec driver is not the same, the audio input and output devices are not the same, machine driver for CPU, codec, input and output devices provide a carrier; platform driver is reusable and the interface chip provides some control functions; codec driver, including the I2S interface, D / A, a / D, mixer, PA, usually include multiple input and multiple output, through the I2C to control the internal registers, and it is reusable.

III. DESIGN AND IMPLEMENTATION OF AUDIO DRIVER

The snd_card is the alsa driver at the top of a structure, the sound of the logical structure of software began to this structure, almost all sound related logical devices are under the management of snd_card, and sound_card_driver for the first action is usually to create a snd_card structure. The WM8960 audio chip driver designed in this paper is related to the following file in Figure 2.

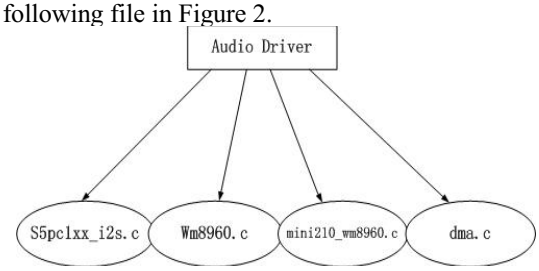

Figure 2. Driver file list

From the top of the chart, the task is divided into 4 parts, which will be allocated to platform driver, codec driver, machine driver. ASoC achieve the sound card as a platform device, and then use the platform_device structure of the dev field: dev.drvdata, which actually points to a snd_soc_device structure. It can be considered that the snd_soc_device is the basic data structure of the whole ASoC, from which it began to lead a series of data structures used to describe the various features and functions of audio.

The snd_soc_device structure leads to the structure of snd_soc_card and soc_codec_device two, and then the snd_soc_card and snd_soc_platform, snd_soc_dai_link and snd_soc_codec structure.

As mentioned above, the ASoC is divided into machine, platform and codec to three parts, if it appears from these data structures, snd_codec_device and snd_soc_card represents the machine drive, snd_soc_platform represents platform driver, snd_soc_codec and soc_codec_device represents the codec driver, and snd_soc_dai link is responsible for connecting platform and codec.

978-1-4673-9393-5/15 $31.00 © 2015 IEEE
DOI 10.1109/ISDEA.2015.80

A. Design I2S Bus

I2S is a serial digital audio bus transmission protocol, which makes the core processor S5PV210 and the external audio chip WM8960 for data transmission. As shown in Figure 3 below.

I2S has 3 clock lines, respectively, is the serial data provided by the bit clock, switch the left and right channels of the frame clock, as well as the system synchronization clock; 2 data lines for the serial data input and output lines. I2S register set to the driver has a great influence, only set the right to guarantee normal work.

The two most closely associated with the I2S register for the IISCON control register, IISMOD mode register. At the same time, it contains the interface function of register related settings, and provides the multiple relations between the system clock and the sampling rate of I2S.

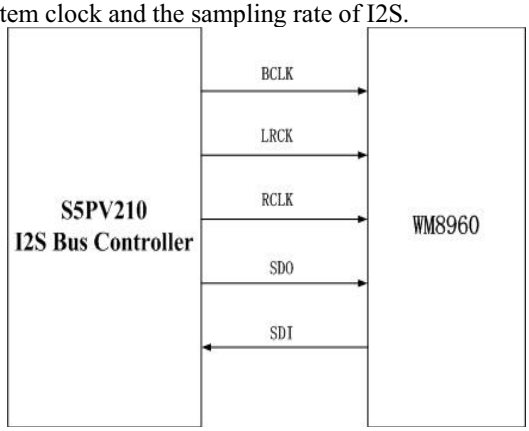

Figure 3. I2S interface connection diagram

The s5pc1xx_i2s_dai function plays a connection role. s5pc1xx-i2s.c, set up the interface function of the relevant registers will be implemented in the platform driver, the role of audio data transmission. Machine driver write process must be registered in the platform can be used, in the initialization function and the exit function to register and release platform devices. The relationship between several clock, audio subsystem clock source selection, kernel selection is EPLL, frequency of the clock as a clock source to the I2S subsystem, when enters I2S subsystem, select I2SCLK as the I2S module clock source, the frequency can be root clock(RCLK), root clock frequency can get the relationship between BCLK. IIS interface root clock and LRCLK clock is RFS (root clock) =256*44.1KHz=11.2896 MHz .In the mode register of RFS and BFS key configuration we use the default value, bit clock =32fs, root clock=256fs, fs=44.1 MHz, here is the codec frequency, according to WM8960 datasheet to check, from the top of the clock in relation calculation we know that fs is equals to LRCK, the left and right choice of the clock.

B. Design WM8960.c

The driver file is the codec driver. Mainly MIC or Line in and other analog signals A/D conversion into CPU to identify the processing of digital signal and the PCM and other signals for D/A conversion, audio access control. The main data structures of WM8960.c are described as snd_soc_codec,snd_soc_codec_drive,snd_soc_dai,snd_soc_dai_driver, the snd_soc_dai and snd_soc_dai_driver are also used in the platform driver, dai of platform and codec through the snd_soc_dai_link structure, the machine driver in the binding connection. The primary task of the WM8960.c driver is to determine the instance of snd_soc_codec and snd_soc_dai, and to register them in the system, the dai and codec after the registration can be used for machine driver.

The driver is important part of which is I2C hard problem. The wm8960_i2c_probe function, which first applied for a wm8960 variable. The variable is used as the I2C device driver data is used, as its sub-devices will be removed and use this driver data. Next, use regmap_init_i2c to initialize and obtain a regmap structure, the structure is mainly used for the follow-up of the regmap mechanism based on I/O. Finally, wm8960_device_init is added to the MFD subsystem.

The struct list_head devices records the list of all logical devices under the sound card.The struct list_head controls records the list of all control units under the sound card. The void *private_data is sound card's private data; you can specify the size of the data in the creation of a sound card. Alsa driver has provided a number of common components of the creation function, and do not need to directly call snd_device_new.

C. Design mini210-wm8960.c

The drive file represents machine driver. Through the snd_soc_card structure, two important data structures of Machine driver are introduced: snd_soc_ops and snd_soc_dai_link.

When we write the driver program, the first thing to do is sound card creation and registration created by the function snd_card_create. It contains the sound card ID and name, management of the device driver on the list of registered by the snd_card_register function to achieve, only registered to be outside security access.

The soc_probe_dai_link function calls codec, dai, and probe to the platform function, and it also calls the soc_new_pcm function to create a logical device. snd_pcm_set_ops function describes the operation of the recording and playback, including open, close, read and write, hardware parameters and control function.

The mini210_wm8960.c driver calls several external functions.

The snd_soc_dai_set_fmt function sets the format of the dai and loads it into the codec_dai and cpu_dai, and the two part of the drive is connected to the I2S, and the fmt mode is selected.

The snd_soc_dai_set_sysclk function is primarily responsible for setting the master clock of the dai, and setting the signal generated by the I2S and the clock signal to the motherboard.

The snd_soc_dai_set_clkdiv function is used to set the frequency dividing coefficients.

The snd_soc_dapm_new_controls function is used to create a widget for each widget memory allocation, initialization necessary field, then the this widget hanging in

on behalf of the sound card snd_soc_card widgets list field, set the playback and recording mode.

The snd_soc_dapm_add_routes function is used to create a path connection information for the machine level.

The snd_soc_dapm_nc_pin, snd_soc_dapm_disable_pin, snd_soc_dapm_enable_pin three API functions are used to set the pin state.

D. DMA transmission

Platform driver's main role is to complete audio data management, through the digital audio interface of the CPU to the audio data is transmitted to the codec processing and finally by the codec output drive headphones or horn signal is heard.

In the concrete implementation, ASoC has the platform driver is divided into two parts: snd_soc_platform_driver and snd_soc_dai_driver. Among them, platform driver is responsible for management of audio data, the audio data from DMA or other operation is transmitted to the CPU dai, dai driver mainly performed on one side of the CPU dai configuration parameters, but also through certain ways the necessary DMA parameters and snd_soc_platform_driver interact.

The main function of the soc-platform driver is to complete the transmission of audio data, in most cases, the audio data is accomplished by dma.

Because of the high requirement of real-time audio application, the data amount is large. So it is necessary to allocate resources reasonably and use appropriate algorithm. DMA's data buffer is consistent_alloc function application. CPU decoding and DMA transmission are not affected. The buf_full tag controls the DMA not to transmit an empty block. DMA transmission requires the source, destination physical address in each buffer block of data structure dma_desc.

The dma_new function for playback and capture were called preallocate_dma_buffer function, after obtaining a DMA buffer, that obtain the source address of the DMA operation.The snd_soc_dai_get_dma_data function to obtain the DMA dai information, which includes the DMA destination address information.

The DMA information usually also be saved in the Substream->runtime->private_data data, in order to substream throughout the life cycle of ready access to this information, complete the configuration and operation of the DMA.

IV. MULTI THREAD KEY AUDIO PLAYER

Audio player is based on the ALSA architecture. Its overall architecture is shown in Figure 4 below. In the kernel device driver layer alsa provides alsa-driver, also in the application layer alsa for us provides alsa lib, application as long as provided by calling the alsa lib API, on the bottom of the audio hardware control can be completed.

As it can be seen from the picture, the user space of alsa-lib provides a unified API interface, which can hide the implementation details of the driver, simplifying the application of the difficulty of the application.

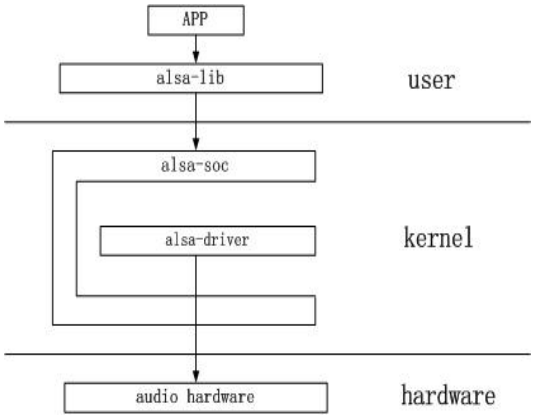

Figure 4. ALSA overall architecture

In the kernel space, alsa-soc is actually a further encapsulation of alsa-driver; it has provided some of the functions of the column for embedded devices. This series of Posts only discusses the alsa-soc and alsa-driver in the embedded system.

In order to test the performance of the WM8960 driver written in the Tiny210 development board, design a multi thread key audio player by transplanting madplay. Play flow chart as shown in Figure 5 below.

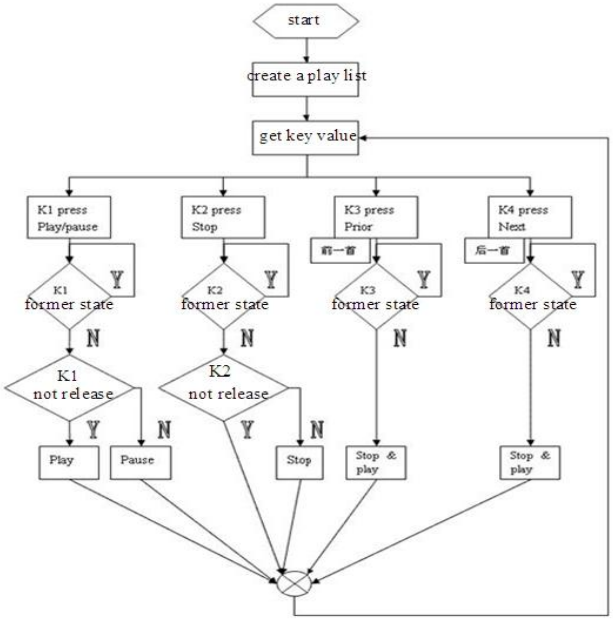

Figure 5. Playback flow chart

Based on the Tiny210 development board, the use of the board four key resources, K1~K4, respectively, the following functions:

(1) Play /Pause K1;
(2) Stop K2;
(3) Prev Song K3;
(4) Next Song K4;

The need for their implementation is a key driver; application layer are the main broadcast program and madplay player program to do a multi-threaded key player from layers of software, driver layer includes a key driver and sound card drivers.

V. CONCLUSIONS

In research based on the audio codec chip based on the Linux operating system audio system architecture and working principle of, and drive the design coder by cortex A8 microprocessor tiny210 and WM8960 the main work of this paper. It is able to achieve audio playback and acquisition operation. On the basis of the design and implementation of a simple player application, the driver's design and implementation of the correctness of the program is verified by calling the driver function.

REFERENCES

[1] Fox J J,Gormley T J.Informed maintenance for next generation reusable launches system s[J].Act an Austronautic,2001,48(5-12):338-345.

[2] Philips Semiconductor.I2S bus specification[S].

[3] Dave Phillips, State of the art: Linux audio 2008[M].USA: Specialized Systems Consulttants, Inc.2008.

[4] Mat Hans, Ronald Schafer W.Lossless Compression of Digital Audio [J].IEEE Sihnal Processing Magazine, 2001(7):13-18.

[5] Ahmed Amine Jerraya.Embedded Software for Soc.Kluwer Academic Publishers, 2003:23-26.

[6] Goodacre J, Sloss A N.Parallelisn and the ARM instruction set architecture [J].

[7] Minch R G.Give your heboot: using the operating system to boot the operating system [J].Cluster computing, 2004, 20(4):440-445.