To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

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

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   “Specific”: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

RENESAS

Application Note

Multimedia Processor for Mobile Applications

Audio/Voice and PWM Interface

EMMA Mobile1

Document No. S19902EJ1V0AN00
Date Published Aug. 2009

PREFACE

Purpose

The purpose of this document is to specify the usage of the Audio/Voice and PWM interface.

Organization

This document includes the following:

- Introduction
- Usage of Audio/Voice Interface
- Usage of PWM Interface
- Example of Audio/Voice Operation
- Example of PWM Operation
- Audio/Voice Driver Function
- PWM Driver Function

Notation

Here explains the meaning of following words in text:

Note	Explanation of item indicated in the text
Caution	Information to which user should afford special attention
Remark	Supplementary information

Related document

The following tables list related documents.

Reference Document

Document Name	Version/date	Author	Description
S19265EJ1V0UM00_ASMUGIO.pdf	1st edition	NECEL	ASMU/GIO User's Manual
S19268EJ1V0UM00_1chip.pdf	1st edition	NECEL	1 Chip User's Manual
S19256EJ1V0UM00_I2C.pdf	1st edition	NECEL	I2C User's Manual
S19253EJ1V0UM00_AUDIO.pdf	1st edition	NECEL	Audio/Voice and PWM User's Manual
S19255EJ1V0UM00_DMA.pdf		NECEL	DMA User's Manual
ak4648_f01e.pdf	1st edition	AKM	AK4648 User's Manual
S19907EJ1V0AN00_GD.pdf f	1st edition	NECEL	GD Spec

Disclaimers

- **The information contained in this document is subject to change without prior notice in the future. Refer to the latest applicable data sheet(s) and user's manual when designing a product for mass production.**
- No part of this document may be copied or reproduced in any form or by any means without the prior written consent of NEC Electronics. NEC Electronics assumes no responsibility for any errors that may appear in this document.
- NEC Electronics does not assume any liability for infringement of patents, copyrights or other intellectual property rights of third parties by or arising from the use of NEC Electronics products listed in this documents or any other liability arising from the use of such products. No license, express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC Electronics or others.
- Descriptions of circuits, software and other related information in this document are provided for illustrative purposes in semiconductor product operation and application examples. The incorporation of these circuits, software and information in the design of a customers' equipment shall be done under the full responsibility of the customer. NEC Electronics assume no responsibility for any losses incurred by customers or third parties arising from the use of these circuits, software and information.
- While NEC Electronics endeavors to enhance the quality, reliability and safety of NEC Electronics products, customers agree and acknowledge that possibility of defects thereof cannot be eliminated entirely. To minimize risks of damage to property or injury (including death) to persons arising from defects in NEC Electronics products, customers must incorporate sufficient safety measures in their design, such as redundancy, fire-containment and anti-failure features.

Note)

1. "NEC Electronics" as used in this document means NEC Electronics Corporation and also includes its majority-owned subsidiaries.
2. "NEC Electronics products" means any product developed or manufactured by or for NEC Electronics (as defined above)
3. All trademarks or registered trademarks are the property of their respective owners. Registered trademarks ® and trademarks™ are not noted in this document.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1 Introduction

1.1 Outline

This document introduces how to use the Audio/Voice and PWM interface of EMMA Mobile1, including the following functions:

- How to send and receive data with PCM interface.
- How to output pulse-width-modulated signals with PWM interface.

The Audio/Voice interface can output digital data to CODEC (for example AK4648) or input digital data from CODEC.

The PWM interface outputs pulse-width-modulated signals. It is mainly used for the beep sound generator, motor control, and LCD modulation control.

About the details of functions please refer to "**CHAPTER 1 OVERVIEW**" of Audio/Voice and PWM user's manual.

1.2 Development Environment

- Hardware environment of this project is listed as below.

Table 1-1 Hardware Environment

Name	Version	Maker
EMMA Mobile1 evaluation board (PSKCH2Y-S-0016-01)	-	NEC Electronics
PARTNER-Jet ICE ARM	M20	Kyoto Microcomputer Co. Ltd

- Software used in this project is listed as below.

Table 1-2 Software Environment

Name	Version	Maker
GNUARM Toolchain	V4.3.2	GNU
WJETSET-ARM	V5.10a	Kyoto Microcomputer Co. Ltd

Chapter 2 Usage of Audio/Voice Interface

2.1 Audio/Voice Overview

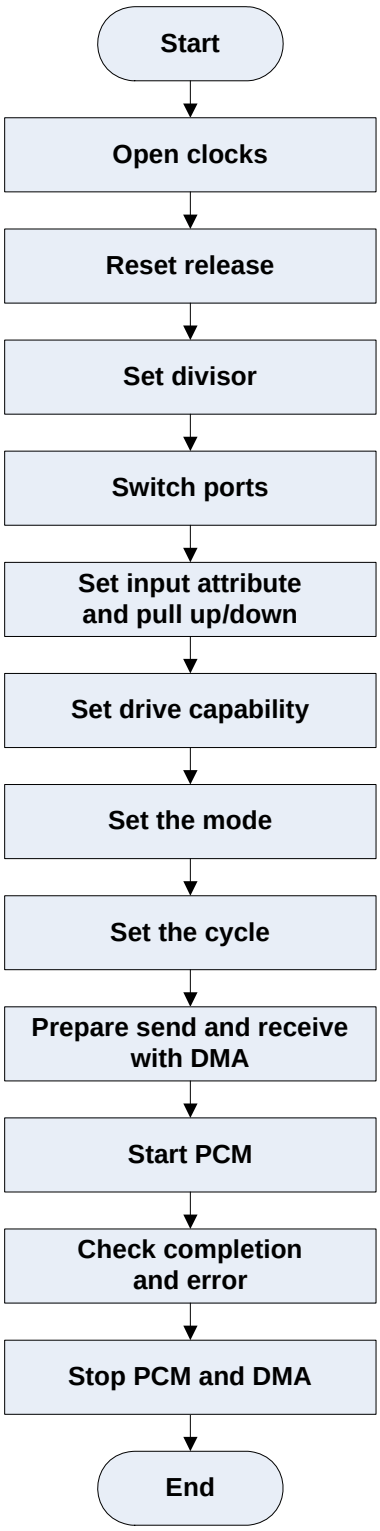

Figure 2-1 Audio/Voice Operation Overview

2.2 Audio/Voice Details

(1). Open clocks.

The related registers are as follow:

ASMU_GCLKCTRL2ENA;

ASMU_GCLKCTRL2;

ASMU_GCLKCTRL4ENA;

ASMU_GCLKCTRL4;

(2). Reset release.

The related registers are as follow:

ASMU_RESETREQ1ENA;

ASMU_RESETREQ1;

(3). Set divisor.

The related registers are as follow:

ASMU_DIVPM0SCLK;

ASMU_DIVPM1SCLK;

(4). Switch ports.

The related registers are as follow:

CHG_PINSEL_G80;

CHG_PINSEL_G64;

(5). Set input attribute and pull up/down.

The related registers are as follow:

CHG_PULL_G72;

CHG_PULL_G80;

CHG_PULL2;

(6). Set drive capability.

The related registers are as follow:

CHG_DRIVE1;

(7). Set the mode.

This step sets the communicate mode (from mode 0 to mode 6), master mode or slave mode, clock polarity. Reference the CODEC chip data sheet to decide how to set this value.

The related registers are as follow:

PM0_FUNC_SEL;

PM1_FUNC_SEL;

(8). Set the cycle.

In mode 0 to 4, this step sets the transmit bits, receive bits and frame length. In mode 5 and 6, this step sets the word numbers. Reference the CODEC chip data sheet to decide how to set this value.

The related registers are as follow:

PM0_CYCLE;
PM0_CYCLE2;
PM1_CYCLE;
PM1_CYCLE2;

(9). Prepare send and receive with DMA.

The PCM interface uses the DMA to transmit and receive data. Please refer to the DMA user's manual about how to use the DMA.

The related registers for DMA are as follow:

ASMU_GCLKCTRL0ENA;
ASMU_GCLKCTRL0;
ASMU_RESETREQ0ENA;
ASMU_RESETREQ0;

DMA_M2P_LCH9_AADD;
DMA_M2P_LCH9_AOFF;
DMA_M2P_LCH9_ASIZE;
DMA_M2P_LCH9_ASIZE_COUNT;
DMA_M2P_LCH9_BADD;
DMA_M2P_LCH9_LENG;
DMA_M2P_LCH9_MODE;
DMA_M2P_CONT;

DMA_P2M_LCH9_AADD;
DMA_P2M_LCH9_BOFF;
DMA_P2M_LCH9_BSIZE;
DMA_P2M_LCH9_BSIZE_COUNT;
DMA_P2M_LCH9_BADD;
DMA_P2M_LCH9_LENG;
DMA_P2M_LCH9_MODE;
DMA_P2M_CONT;

The related registers for PCM are as follow:

PM0_TXQ;
PM0_RXQ;
PM1_TXQ;
PM1_RXQ;

(10). Start PCM.

The related registers are as follow:

PM0_TXRX_EN;

PM1_TXRX_EN;

(11). Check completion and error.

To check the completion and error, use the registers about interrupt.

The related registers are as follow:

PM0_RAW;

PM0_STATUS;

PM0_ENSET;

PM0_ENCLR;

PM0_CLEAR;

PM1_RAW;

PM1_STATUS;

PM1_ENSET;

PM1_ENCLR;

PM1_CLEAR;

(12). Stop PCM and DMA.

The related registers are as follow:

PM0_TXRX_DIS;

PM1_TXRX_DIS;

DMA_M2P_END;

DMA_P2M_END;

Chapter 3 Usage of PWM Interface

3.1 PWM Overview

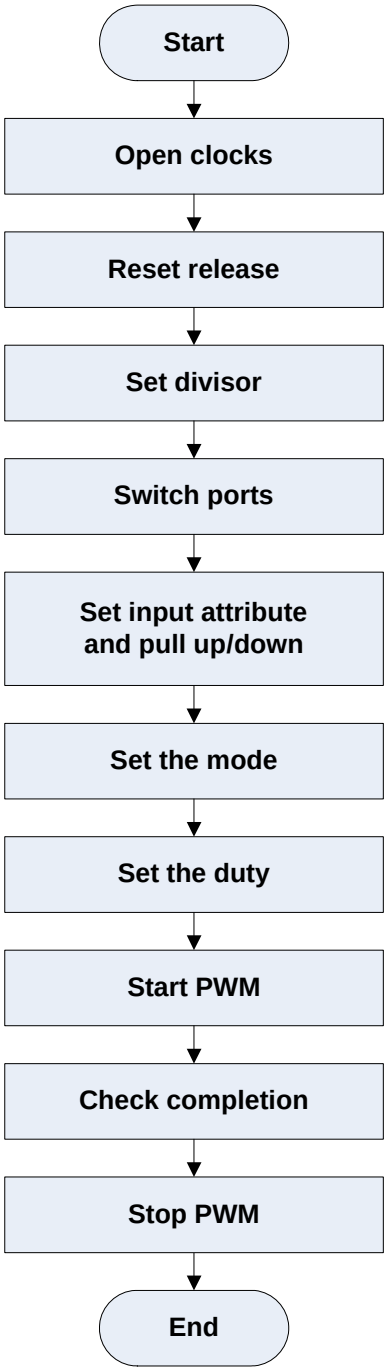

Figure 3-1 PWM Operation Overview

3.2 PWM Details

(1). Open clocks.
The related registers are as follow:
ASMU_GCLKCTRL3ENA;
ASMU_GCLKCTRL3;

(2). Reset release.
The related registers are as follow:
ASMU_RESETREQ2ENA;
ASMU_RESETREQ2;

(3). Set divisor.
The related registers are as follow:
ASMU_DIVPWMPWCLK;

(4). Switch ports.
The related registers are as follow:
CHG_PINSEL_G80;

(5). Set input attribute and pull up/down.
The related registers are as follow:
CHG_PULL_G88;

(6). Set the mode.
This step sets the operation mode (AND, OR and XOR) and CMP number (CMP0, CMP1 and CMP2).
The related registers are as follow:
PWM_CH0_MODE;
PWM_CH1_MODE;

(7). Set the duty.
Details about how to set the duty please refer to the "**Chapter 4.2 PWM Interface**" of Audio/Voice and PWM user's manual.
The related registers are as follow:
PWM_CH0_DELAY0;
PWM_CH0_LEDGE0;
PWM_CH0_TEDGE0;
PWM_CH0_TOTAL0;
PWM_CH0_LOOP0;
PWM_CH0_DELAY1;
PWM_CH0_LEDGE1;
PWM_CH0_TEDGE1;

PWM_CH0_TOTAL1;
PWM_CH0_LOOP1;
PWM_CH0_DELAY2;
PWM_CH0_LEDGE2;
PWM_CH0_TEDGE2;
PWM_CH0_TOTAL2;
PWM_CH0_LOOP2;

PWM_CH1_DELAY0;
PWM_CH1_LEDGE0;
PWM_CH1_TEDGE0;
PWM_CH1_TOTAL0;
PWM_CH1_LOOP0;
PWM_CH1_DELAY1;
PWM_CH1_LEDGE1;
PWM_CH1_TEDGE1;
PWM_CH1_TOTAL1;
PWM_CH1_LOOP1;
PWM_CH1_DELAY2;
PWM_CH1_LEDGE2;
PWM_CH1_TEDGE2;
PWM_CH1_TOTAL2;
PWM_CH1_LOOP2;

(8). Start PWM.
The related registers are as follow:
PWM_CH0_CTRL;
PWM_CH1_CTRL;

(9). Check completion.
The related registers are as follow:
PWM_INTSTATUS;
PWM_INTRAWSTATUS;
PWM_INTENSET;
PWM_INTENCLR;
PWM_INTFFCLR;

(10). Stop PWM.
The related registers are as follow:
PWM_CH0_CTRL;
PWM_CH1_CTRL;

Chapter 4 Example of Audio/Voice Operation

The following contents show 2 examples about Audio/Voice interface:

- How to output sine wave as master.
- How to input and output data as slave.

4.1 Example of Output as Master

The hardware connection of EMMA Mobile1 and AK4648 for output is as follow figure.

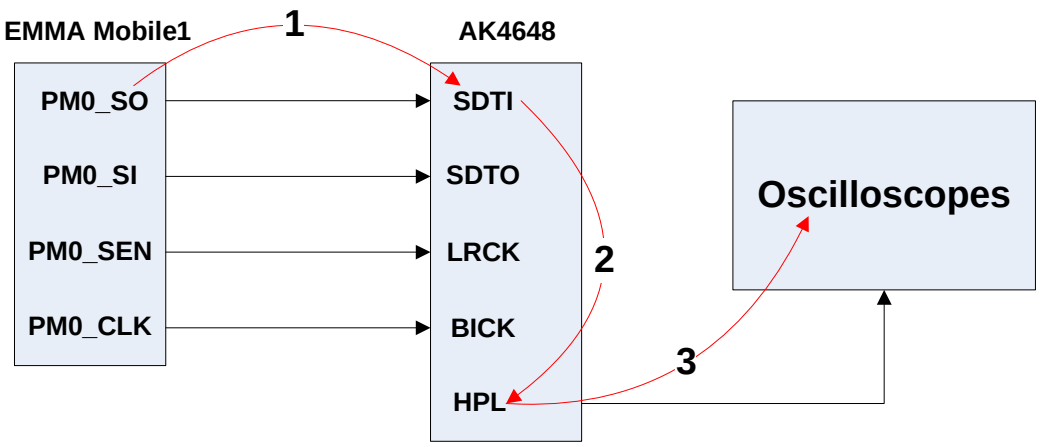

Figure 4-1 Hardware Connection of PCM and AK4648 for Output

This example has the follow steps:

1. EMMA Mobile1 outputs digital sine wave from PM0_SO to SDTI of AK4648.
2. The AK4648 converts the digital signal to analog signal.
3. Oscilloscopes can measure HPL signal of the AK4648. If the signal is sine wave, it means PCM output of EMMA1 Mobile1 is correct.

The AK4648 is a stereo CODEC. It supports the PCM interface. About the details of AK4648 please refer to the AK4648 user's manual.

4.1.1 Operation Flow

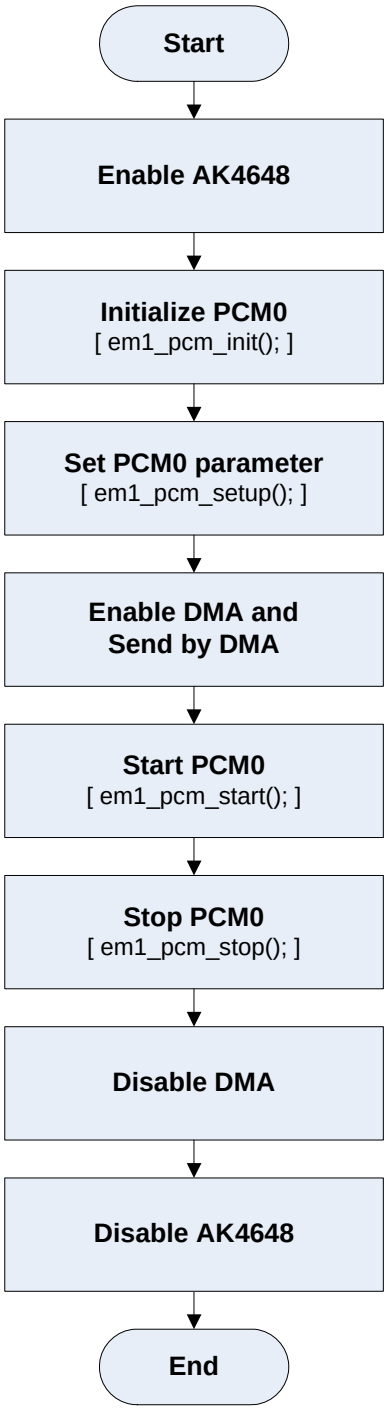

Figure 4-2 Example of Output with PCM0

About the PCM function, please refer to the "**Appendix A Audio/Voice Driver Function**".

4.1.2 Operation Detail

(1). Enable AK4648.
This process sets the following three functions: the AK4648 reference clock; AK4648 address and power on; the AK4648 control register.
About how to set AK4648 address and power on, how to set the AK4648 control register, please refer to AK4648 user's manual.

The following contents set the AK4648 reference clock. About the reference clock frequency value, please refer to the AK4648 user's manual.

- Set PLL2 clock frequency (32.768 KHz * 125 * 0x79 = 499.712MHz).
  ASMU_PLL2CTRL1 = 0xff;
  ASMU_PLL2CTRL0 = 0x79;

- Set reference clock divisor (499.712MHz / (0x29 + 1) = 12.188MHz).
  ASMU_DIVREFCLK = 0x29;
  ASMU_GCLKCTRL2ENA[20] = 1;
  ASMU_GCLKCTRL2[20] = 1;
  ASMU_GCLKCTRL2ENA[20] = 0;
  CHG_PINSEL_REFCLKO[0] = 1;
  ASMU_PLL2CTRL1 = 0x0;

(2). Initialize PCM0.
The process calls the "em1_pcm_init()" function.
The "em1_pcm_init()" function finishes the following functions for PCM0:

- Open PM0_SCLK.
  ASMU_GCLKCTRL2ENA[13] = 1;
  ASMU_GCLKCTRL2[13] = 1;
  ASMU_GCLKCTRL2ENA[13] = 0;

- Open PM0_PCLK.
  ASMU_GCLKCTRL2ENA[12] = 1;
  ASMU_GCLKCTRL2[12] = 1;
  ASMU_GCLKCTRL2ENA[12] = 0;
  ASMU_GCLKCTRL4ENA[8] = 1;
  ASMU_GCLKCTRL4[8] = 1;
  ASMU_GCLKCTRL4ENA[8] = 0;

- Reset PCM0.
  ASMU_RESETREQ1ENA[16] = 1;
  ASMU_RESETREQ1[16] = 1;

ASMU_RESETREQ1ENA[16] = 0;

- Reset release PCM0.

ASMU_RESETREQ1ENA[16] = 1;
ASMU_RESETREQ1[16] = 0;
ASMU_RESETREQ1ENA[16] = 0;

- Set the divisor.

ASMU_DIVPM0SCLK = Divisor;

Divisor is an input value and it should be set according to AK4648 timing. Details about how to set the Divisor please refer to the "**Chapter 3.2.55 PM0_SCLK frequency division setting register**" of ASMU/GIO user's manual.

- Switch port function.

CHG_PINSEL_G80[15:14] = 1; // GIO_P87 -- > PM0_SI;

- Set input attribute and pull up/down.

CHG_PULL_G80[30:28] = 5; // Enable PM0_SI inputs to pass and disable pull up/down.
CHG_PULL2[26:24] = 5; // Enable PM0_CK inputs to pass and disable pull up/down.
CHG_PULL2[30:28] = 5; // Enable PM0_EN inputs to pass and disable pull up/down.

- Set drive capability.

CHG_DRIVE1[9:8] = 1; // Set the PM0_CLK, PM0_SEN, PM0_SI, and PM0_SO drive capability: 4mA

(3). Set PCM0 parameter.

```
PCM_SETUP_ST pcm_st;

pcm_st.channel = 0;                 // PCM channel 0;
pcm_st.num_TX_FIFO = 3;             // Trigger number is 4;
pcm_st.mode_m_s = 1;                // Master mode;
pcm_st.mode_sel = 2;                // The data format is I2S;
pcm_st.TX_padding = 0;              // Disable padding;
pcm_st.SOB = 0xf;                   // Transmit data length is 16 bits;
pcm_st.cyc_val = 0x1f;              // The frame length is 32 PM0_CLK;
em1_pcm_setup(&pcm_st);
```

The em1_pcm_setup() finishes the following functions for PCM0. About the I2S transmission format, data length and frame length please refer to the AK4648 user's manual.

- Set master mode and I2S data format; the trigger number is 4.

PM0_FUNC_SEL = 0x6a;

- The frame length is 32 PM0_CLK, the transmit data length is 16 bits.
  PM0_CYCLE = 0x0f0f1f;

The EMMA Mobile1 supports 6 transmission modes:

Table 4-1 PCM Mode List

Mode	Function
Mode 0	Default
Mode 1	Reference Mode 0
Mode 2	I2S format
Mode 3	MSB justified
Mode 4	LSB justified
Mode 5	Multi-channel mode
Mode 6	Multi-channel mode

About the mode details, please refer to the "**CHAPTER 4 DESCRIPTION OF FUNCTIONS**" of Audio/Voice user's manual.

(4). Enable DMA and send by DMA.
The process enables the DMA to send data.
About the DMA interface, please refer to the DMA user's manual.

- Send data from PCM0.
  PM0_TXQ = data;

(5). Start PCM0.
The process calls the em1_pcm_start();
The em1_pcm_start() finishes the following functions for PCM0:

- Disable PCM0.
  PM0_TXRX_DIS = 0x1;

- Enable PCM0.
  PM0_TXRX_EN = 0x1;

(6). Stop PCM0.
The process calls the em1_pcm_stop();
The em1_pcm_stop() finishes the following functions for PCM0:

- Disable PCM0.
  PM0_TXRX_DIS = 0x1;

(7). Disable DMA.
The process disables the DMA to send data.
About the DMA interface, please refer to the DMA user's manual.

(8). Disable AK4648.

About the details, please refer to the AK4648 user's manual.

4.2 Example of Input and Output as Salve

The hardware connection of EMMA Mobile1 and AK4648 for input and output is as follow figure.

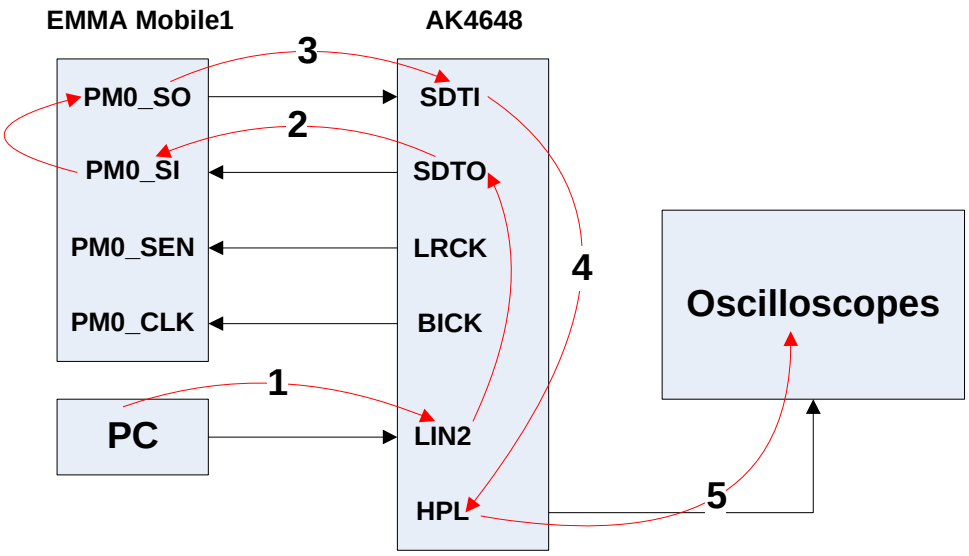

Figure 4-3 Hardware Connection of PCM and AKI4648 for Input and Output

This example has the follow steps:

1. PC plays specified audio file for outputting analog sine wave to the LIN2 of AK4648.
2. The AK4648 converts the analog signal to digital signal, and then outputs the digital signal from SDTO to PM0_SI of EMMA Mobile1.
3. EMMA Mobile1 receives the digital signal at step2, and then outputs the signal from PM0_SO to SDTI of AK4648.
4. The AK4648 receives the digital signal at step3, and then converts to the analog signal to HPL.
5. Oscilloscopes can measure HPL signal of the AK4648. If the signal is sine wave, it means PCM output of EMMA1 Mobile1 is correct.

The AK4648 is a stereo CODEC. It supports the PCM interface. About the details of AK4648, please refer to the AK4648 user's manual.

4.2.1 Operation Flow

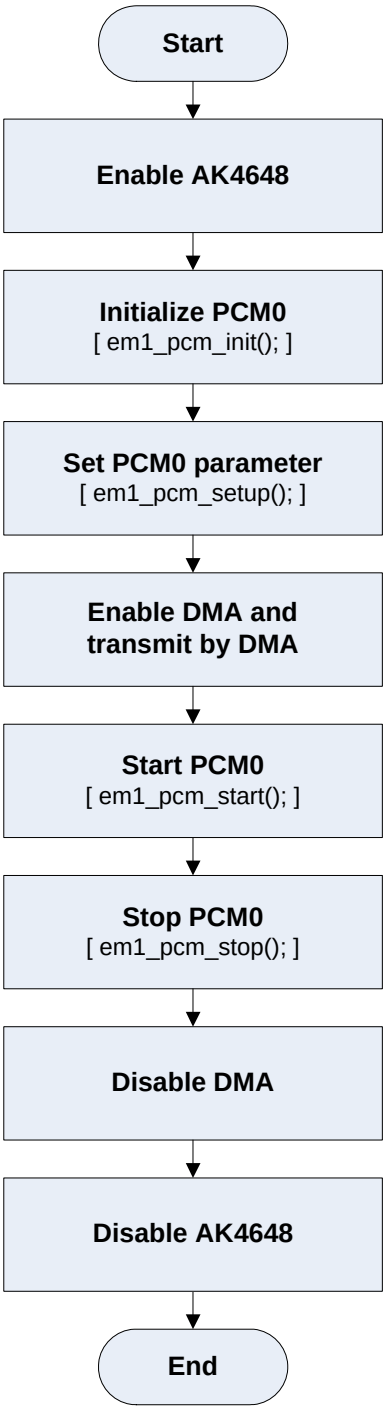

Figure 4-4 Example of Input and Output with PCM0

About the PCM function, please refer to the "**Appendix A Audio/Voice Driver Function**".

4.2.2 Operation Detail

(1). Enable AK4648.
Please refer to the "**Chapter 4.1.2 Operation Detail**" of this document.

(2). Initialize PCM0.
Please refer to the "**Chapter 4.1.2 Operation Detail**" of this document.

(3). Set PCM0 parameter.
PCM_SETUP_ST pcm_st;

```
pcm_st.channel = 0;                 // PCM channel 0;
pcm_st.num_TX_FIFO = 3;             // Trigger number is 4;
pcm_st.mode_m_s = 2;                // Slave mode;
pcm_st.mode_sel = 2;                // The data format is I2S;
pcm_st.TX_padding = 0;              // Disable padding;
pcm_st.SOB = 0xf;                   // Transmit data length is 16 bits;
pcm_st.RX_padding = 0;              // Disable padding;
pcm_st.SIB = 0xf;                   // Receive data length is 16 bits;
pcm_st.cyc_val = 0x1f;              // The number of PM0_CLK is 32 clock;
em1_pcm_setup(&pcm_st);
```

The em1_pcm_setup() finishes the following functions for PCM0. About the I2S transmission format, data length and frame length please refer to the AK4648 user's manual.

- Set slave mode and I2S data format; the trigger number is 4.
  PM0_FUNC_SEL = 0x72;

- The number of PM0_CLK is 32; the transmit data length is 16 bits.
  PM0_CYCLE = 0x0f0f1f;

(4). Enable DMA and transmit by DMA.
The process enables the DMA to send and receive data.
About the DMA interface, please refer to the DMA user's manual.

- Send and receive data
  PM0_TXQ = data1;
  Data2 = PM0_RXQ;

(5). Start PCM0.
The process calls the em1_pcm_start();
The em1_pcm_start() finishes the following functions for PCM0:

- Disable PCM0.

PM0_TXRX_DIS = 0x3;

- Enable PCM0.
  PM0_TXRX_EN = 0x3;

(6). Stop PCM0.
The process calls the em1_pcm_stop();
The em1_pcm_stop() finishes the following functions for PCM0:

- Disable PCM0.
  PM0_TXRX_DIS = 0x3;

(7). Disable DMA.
The process disables the DMA to send data.
About the DMA interface, please refer to the DMA user's manual.

(8). Disable AK4648.
About the details, please refer to the AK4648 user's manual.

Chapter 5 Example of PWM Operation

The following contents show 4 examples about PWM interface:

- Sample 1: counter 0 of channel 0, normal output and repeated
- Sample 2: counter 0 of channel 1, inverted output and repeated
- Sample 3: counter 0 of channel 0, normal output and automatic stop
- Sample 4: counter 0 to 2 of channel 1, normal output and repeated

About the API details, please refer to the “**Appendix B PWM Driver Function**”.

5.1 Initialization

5.1.1 Operation Flow

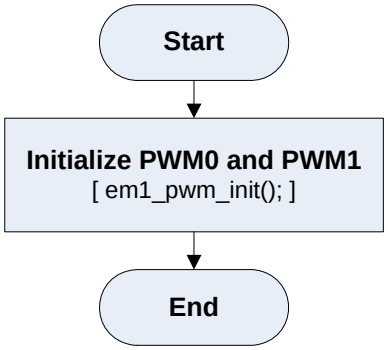

Figure 5-1 Example of PWM Initialization

About the PWM function, please refer to the "**Appendix B PWM Driver Function**".

5.1.2 Operation Detail

(1). Initialize the PWM.

The process calls the em1_pwm_init().

The em1_pwm_init() finishes the following functions for PWM:

- Open PWM_PCLK.

For PWM0 and PWM1:

ASMU_GCLKCTRL3ENA[27] = 1;
ASMU_GCLKCTRL3[27] = 1;
ASMU_GCLKCTRL3ENA[27] = 0;

- Open PWM_PWCLK0 and PWM_PWMCLK1.

For PWM0:

ASMU_GCLKCTRL3ENA[28] = 1;
ASMU_GCLKCTRL3[28] = 1;
ASMU_GCLKCTRL3ENA[28] = 0;

For PWM1:

ASMU_GCLKCTRL3ENA[29] = 1;
ASMU_GCLKCTRL3[29] = 1;
ASMU_GCLKCTRL3ENA[29] = 0;

- Reset PWM.

For PWM0 and PWM1:

ASMU_RESETREQ2ENA[9] = 1;
ASMU_RESETREQ2[9] = 1;
ASMU_RESETREQ2ENA[9] = 0;

- Reset release PWM.

For PWM0 and PWM1:

ASMU_RESETREQ2ENA[9] = 1;
ASMU_RESETREQ2[9] = 0;
ASMU_RESETREQ2ENA[9] = 0;

- Set divisor.

For PWM0:

ASMU_DIVPWMPWCLK[7:0] = Divisor;

For PWM1:

ASMU_DIVPWMPWCLK[23:16] = Divisor;

Divisor is an input value, about how to set the divisor, please refers to the "**Chapter 3.2.58 PWMPWCLK frequency division setting register**" of ASMU/GIO user's manual.

- Switch port function.

For PWM0:

```
CHG_PINSEL_G80[29:28] = 1;          // GIO_P94 -- > PWM0;
```

For PWM1:

```
CHG_PINSEL_G80[31:30] = 1;          // GIO_P95 -- > PWM1;
```

- Set input attribute and pull up/down.

For PWM0:

```
CHG_PULL_G88[26:24] = 1;            // Disable input and pull up/down;
```

For PWM1:

```
CHG_PULL_G88[30:28] = 1;            // Disable input and pull up/down;
```

5.2 Calculation method of PWM interface Parameters

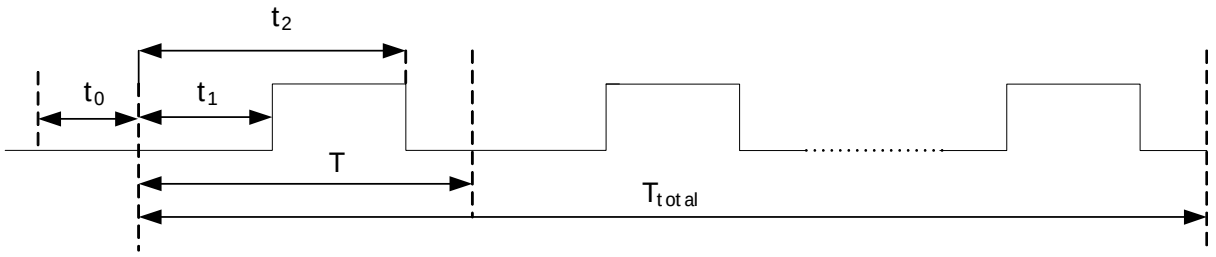

Figure 5-2 Output Waveform

PWM0 interface parameters can be obtain with the following calculation method. And it is the same to PWM1.

LEDGE0= t_1 / t_{pwclk}

TEDGE0 = (t_2-t_1) / t_{pwclk}

TOTAL0 = T / t_{pwclk}

LOOP0 = T_{total} / T

DELAY0 = t_0 / t_{pwclk}

Note: T is one PWM output cycle; T_{total} is “loop number” * T.

Frequency of PWM output signal is PWM_PWCLK0 / TOTAL0. And the duty of PWM output signal is TEDGE0 / TOTAL0.

Note :

1. PWM_PWCLK0 = PLL3 / divisor.
2. The PLL3 is fixed 229.376MHz and divisor is set by ASMU_PWMPWCLK.
3. t_{pwclk} = 1/PWM_PWMCLK0.

The register setting is as following.

```
PWM_CH0_TOTAL0 = TOTAL0;
PWM_CH0_DELAY0 = DELAY0;
PWM_CH0_LEDGE0 = LEDGE0;
PWM_CH0_TEDGE0 = TEDGE0;
PWM_CH0_LOOP0 = LOOP0;
PWM_CH0_MODE = 0x3;
```

If the output waveform is infinite, set CMP_ATST0 to 0.

```
PWM_CH0_MODE = 0x1;
```

If the output waveform is inverted with above figure, set CMP_INV0 to 1.

```
PWM_CH0_MODE = 0x7;        // loop output, but no infinite
PWM_CH0_MODE = 0x5;        // output is infinite
```

5.3 Sample1 (Enable one counter)

5.3.1 Operation Flow

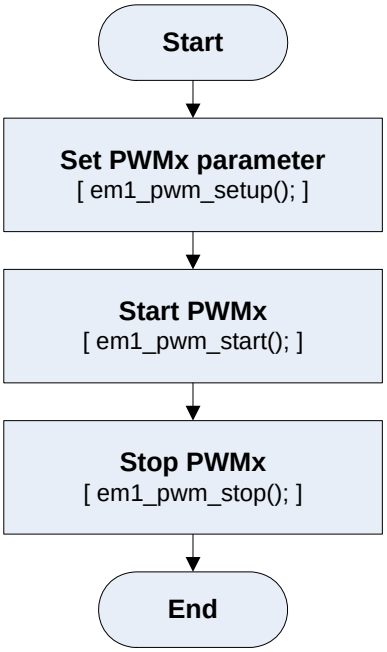

Figure 5-3 Example of PWM

Note: x is 0 or 1.

About the PWM function, please refer to the "**Appendix B PWM Driver Function**".

5.3.2 Operation Detail

(1). Set PWM0 parameter.
PWM_SETUP_ST pwm_st;

```
pwm_st.channel = 0;                 // Select the channel 0;
pwm_st.mode.MOD = 0;                // Select the OR mode;
pwm_st.mode.CMP_EN[0] = 1;          // Enable counter 0;
pwm_st.TOTAL[0] = 0x00000009;
pwm_st.DELAY[0] = 0x00000005;
pwm_st.LEDGE[0] = 0x00000004;
pwm_st.TEDGE[0] = 0x00000007;
em1_pwm_setup(&pwm_st);
```

The em1_pwm_setup() finishes the following functions for PWM0:

- Disable automatic stop and inversion of PWM_CMP0.
  PWM_CH0_MODE = 0x1;

- Set delay value PWM_PWCLK x 5 cycles.
  PWM_CH0_DELAY0 = 0x5;

- Set leading setting 4 cycles.
  PWM_CH0_LEDGE0 = 0x4;

- Set trailing setting 7 cycles
  PWM_CH0_TEDGE0 = 0x7;

- Set total cycle PWM_PWCLK x 9 cycles
  PWM_CH0_TOTAL0 = 0x9;

(2). Start PWM0.
The process calls the em1_pwm_start(0).
The em1_pwm_start() finishes the following functions for PWM0:

- Start PWM0.
  PWM_CH0_CTRL = 0x1;

(3). Stop PWM0.

The process calls the em1_pwm_stop(0).

The em1_pwm_stop() finishes the following functions for PWM0:

- Stop PWM0.
  PWM_CH0_CTRL = 0x0;

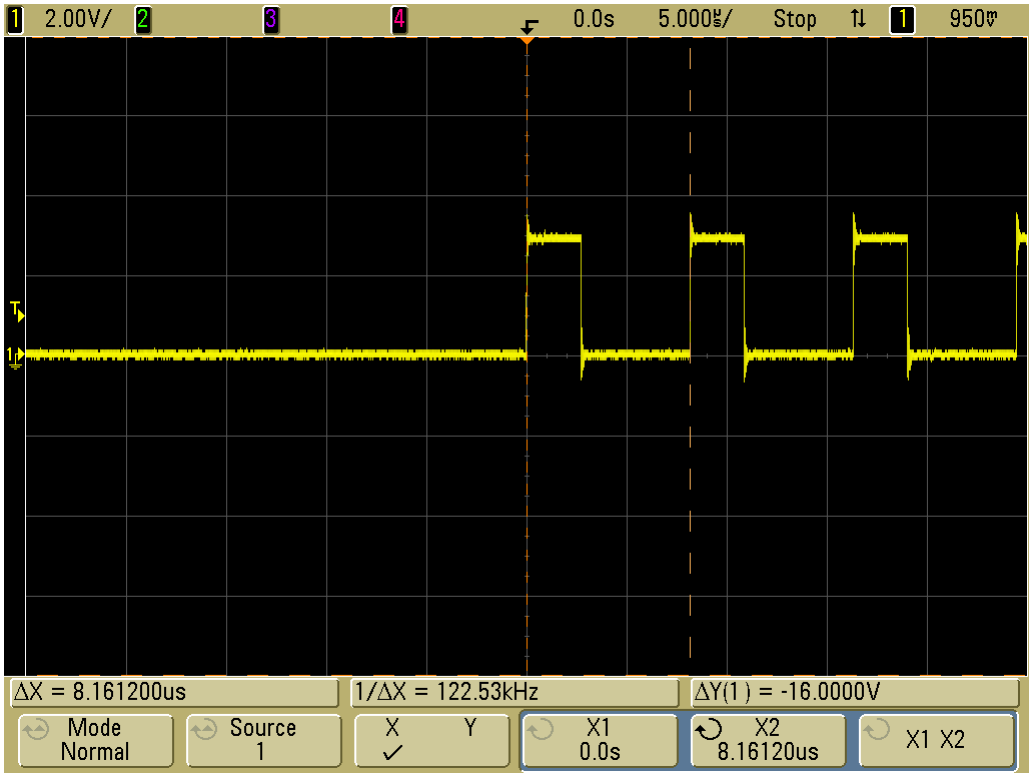

Figure 5-4 PWM Example1-1

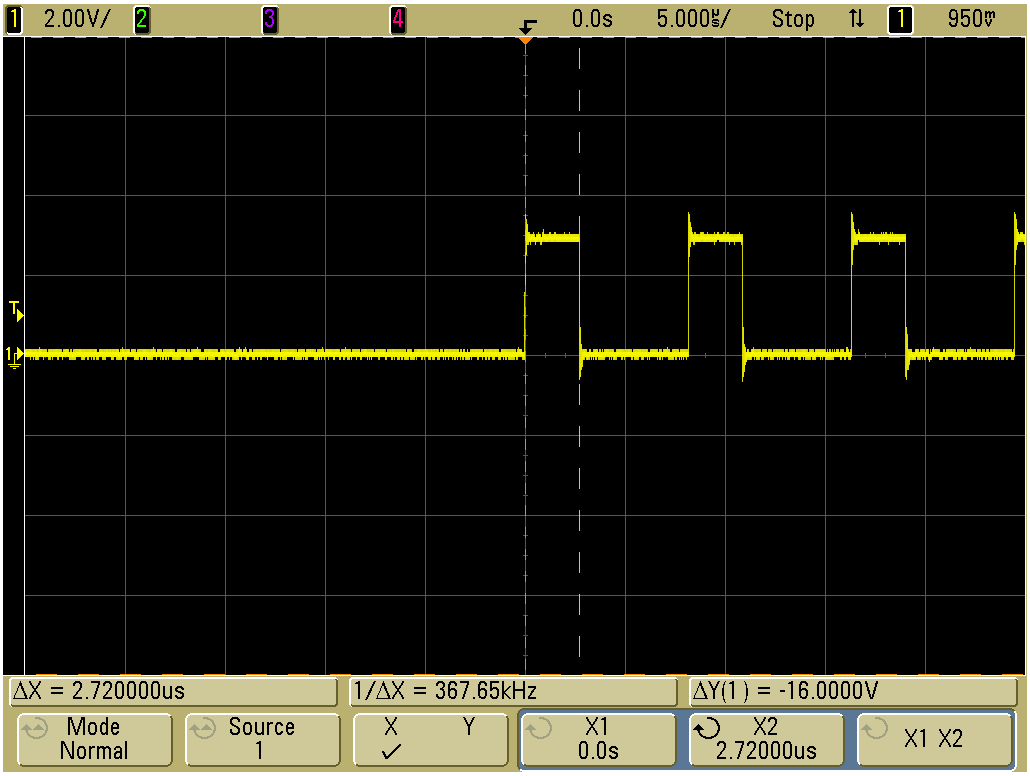

Figure 5-5 PWM Example1-2

In this sample, PWM_PWCLK0 is come from PLL3. The frequency of PLL3 is 229.376MHz and the divisor is 208 set by ASMU_DIVPWMPWCLK, so the PWM_PWCLK0 is:

PWM_PWCLK0 = PLL3 / divisor = 229.376MHz / 208 = 1.10MHz;

As the figure 5-4, the PWM0 output frequency is:

PLL3 / divisor / PWM_CH0_TOTAL0 = 229.376 / 208 / 9 = 122.53 KHz;

PWM0 output cycle is:

1 PWM_PWCLK0 cycle * PWM_CH0_TOTAL0 = (208/229.376) * 9 = 8.16us;

As the figure 5-5, the PWM0 output duty is:

(PWM_CH0_TEDGE0 - PWM_CH0_LEDGE0) / PWM_CH0_TOTAL0 = (7 – 4) / 9 = 33.3%;

The high level width of PWM0 is:

(PWM_CH0_TEDGE0 - PWM_CH0_LEDGE0) / PWM_CH0_TOTAL0 * 1 PWM0 output cycle = (7-4)/9 * 8.16 = 2.72us;

5.4 Sample2 (Enable one counter and inversion)

5.4.1 Operation Flow

About the figure, please refer to the figure 5-3.

5.4.2 Operation Detail

(1). Set PWM1 parameter.
PWM_SETUP_ST pwm_st;

```
pwm_st.channel = 1;                 // Select the channel 1;
pwm_st.mode.MOD = 0;                // Select the OR mode;
pwm_st.mode.CMP_EN[0] = 1;          // Enable counter 0;
pwm_st.mode.CMP_INTV[0] = 1;        // Enable inversion;
pwm_st.TOTAL[0] = 0x00000009;
pwm_st.DELAY[0] = 0x00000005;
pwm_st.LEDGE[0] = 0x00000003;
pwm_st.TEDGE[0] = 0x00000007;
em1_pwm_setup(&pwm_st);
```

The em1_pwm_setup() finishes the following functions for PWM1:

- Disable automatic stop and enable inversion of PWM_CMP0.
  PWM_CH1_MODE = 0x5;

- Set delay value PWM_PWCLK x 5 cycles.
  PWM_CH1_DELAY0 = 0x5;

- Set leading setting 3 cycles.
  PWM_CH1_LEDGE0 = 0x3;

- Set trailing setting 7 cycles
  PWM_CH1_TEDGE0 = 0x7;

- Set total cycle PWM_PWCLK x 9 cycles
  PWM_CH1_TOTAL0 = 0x9;

(2). Start PWM1.

The process calls the em1_pwm_start(1).

The em1_pwm_start() finishes the following functions for PWM1:

- Start PWM1.
  PWM_CH1_CTRL = 0x1;

(3). Stop PWM1.

The process calls the em1_pwm_stop(1).

The em1_pwm_stop() finishes the following functions for PWM1:

- Stop PWM1.
  PWM_CH1_CTRL = 0x0;

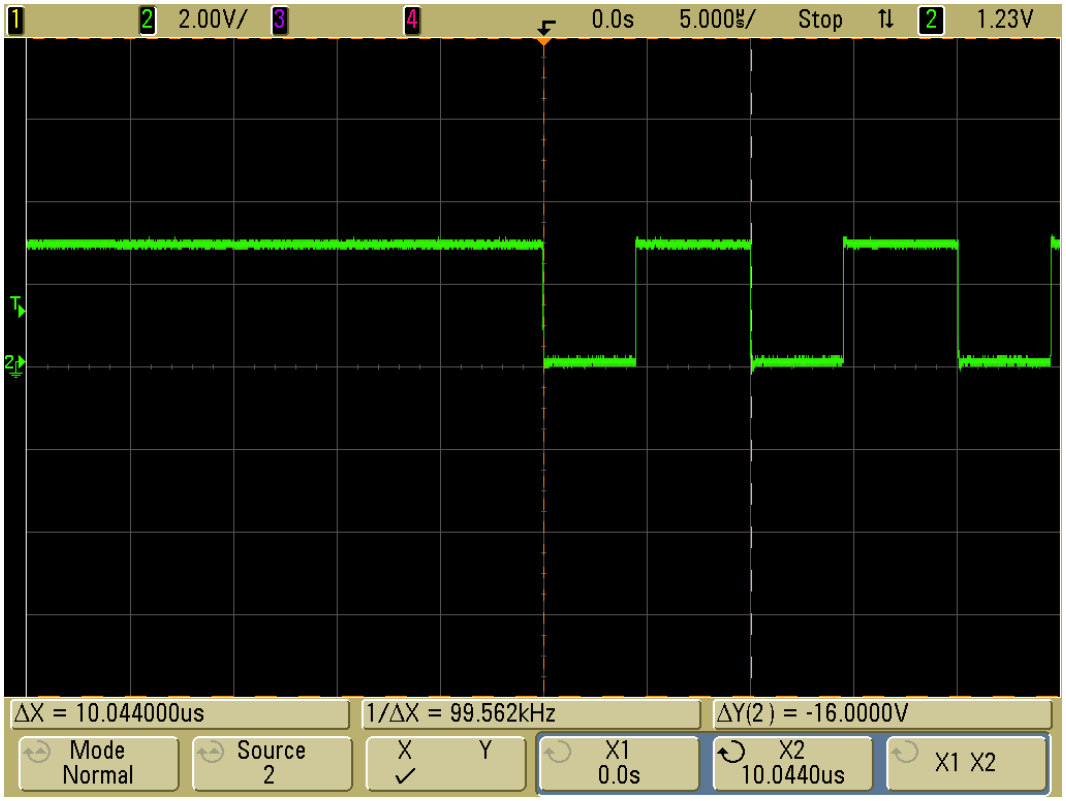

Figure 5-6 PWM Example2-1

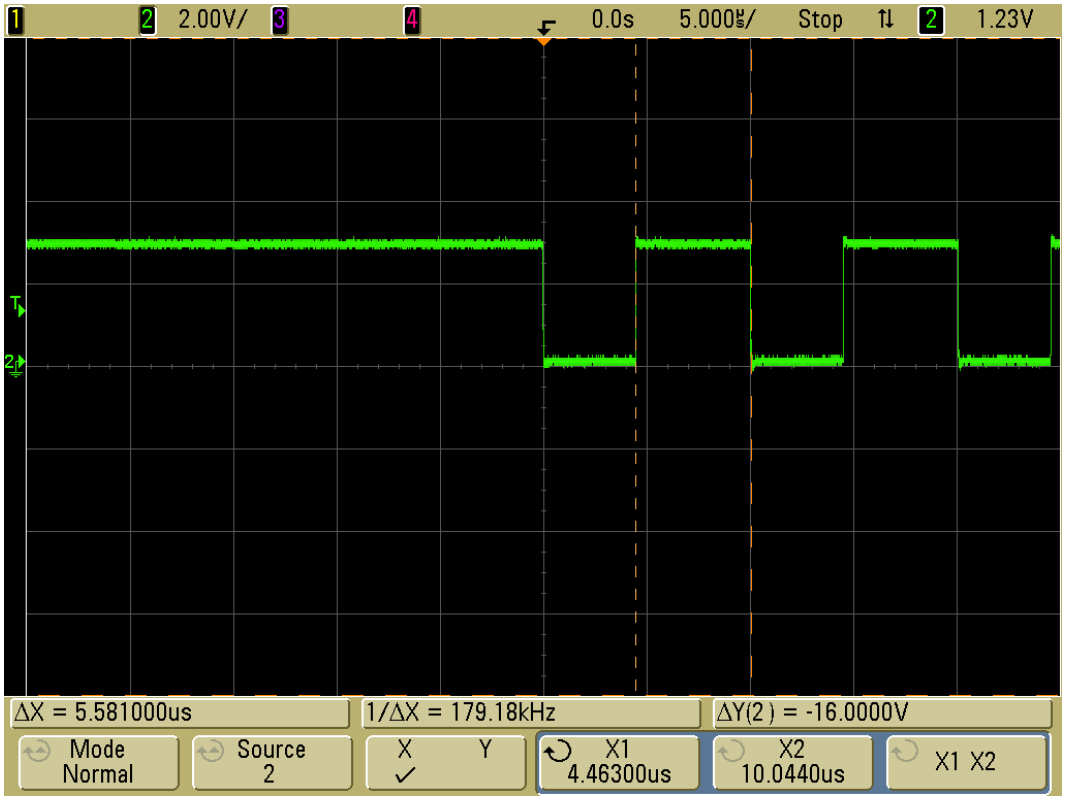

Figure 5-7 PWM Example2-2

In this sample, PWM_PWCLK1 is come from PLL3. The frequency of PLL3 is 229.376MHz and the divisor is 256 set by ASMU_DIVPWMPWCLK, so the PWM_PWCLK1 is:
PWM_PWCLK1 = PLL3 / divisor = 229.376MHz / 256 = 0.896MHz;

As the figure 5-6, the PWM1 output frequency is:
PLL3 / divisor / PWM_CH1_TOTAL0 = 229.376 / 256 / 9 = 99.56 KHz;
PWM1 output cycle is:
1 PWM_PWCLK1 cycle * PWM_CH1_TOTAL0 = (256/229.376) * 9 = 10.04us;

As the figure 5-7, the PWM1 output duty is:
(PWM_CH1_TOTAL0 - (PWM_CH1_TEDGE0 - PWM_CH1_LEDGE0)) / PWM_CH1_TOTAL0 = (9 - (7 – 3)) / 9 = 55.6%;
The high level width of PWM1 is:
(PWM_CH1_TOTAL0 - (PWM_CH1_TEDGE0 - PWM_CH1_LEDGE0)) / PWM_CH1_TOTAL0 * 1 PWM1 cycle = (9 – (7 - 3)) / 9 * 10.04 = 5.58us

5.5 Sample3 (Enable automatic stop)

5.5.1 Operation Flow

About the figure, please refer to the figure 5-3

5.5.2 Operation Detail

(1). Set PWM0 parameter.

```
PWM_SETUP_ST pwm_st;

pwm_st.channel = 0;                 // Select the channel 0;
pwm_st.mode.MOD = 0;                // Select the OR mode;
pwm_st.mode.CMP_EN[0] = 1;          // Enable counter 0;
pwm_st.mode.CMP_ATST[0] = 1;        // Enable automatic stop;
pwm_st.TOTAL[0] = 0x00000009;
pwm_st.DELAY[0] = 0x00000006;
pwm_st.LEDGE[0] = 0x00000004;
pwm_st.TEDGE[0] = 0x00000008;
pwm_st.LOOP[0] = 0x3;
em1_pwm_setup(&pwm_st);
```

The em1_pwm_setup() finishes the following functions for PWM0:

- Enable automatic stop and disable inversion of PWM_CMP0.
  PWM_CH0_MODE = 0x3;

- Set delay value PWM_PWCLK x 6 cycles.
  PWM_CH0_DELAY0 = 0x6;

- Set leading setting 4 cycles.
  PWM_CH0_LEDGE0 = 0x4;

- Set trailing setting 8 cycles.
  PWM_CH0_TEDGE0 = 0x8;

- Set total cycle PWM_PWCLK x 9 cycles.
  PWM_CH0_TOTAL0 = 0x9;

- Set loop number 3.
  PWM_CH0_LOOP = 0x3;

(2). Start PWM0.

The process calls the em1_pwm_start(0).

The em1_pwm_start() finishes the following functions for PWM0:

- Start PWM0.
  PWM_CH0_CTRL = 0x1;

(3). Stop PWM0.

The process calls the em1_pwm_stop(0).

The em1_pwm_stop() finishes the following functions for PWM0:

- Stop PWM0.
  PWM_CH0_CTRL = 0x0;

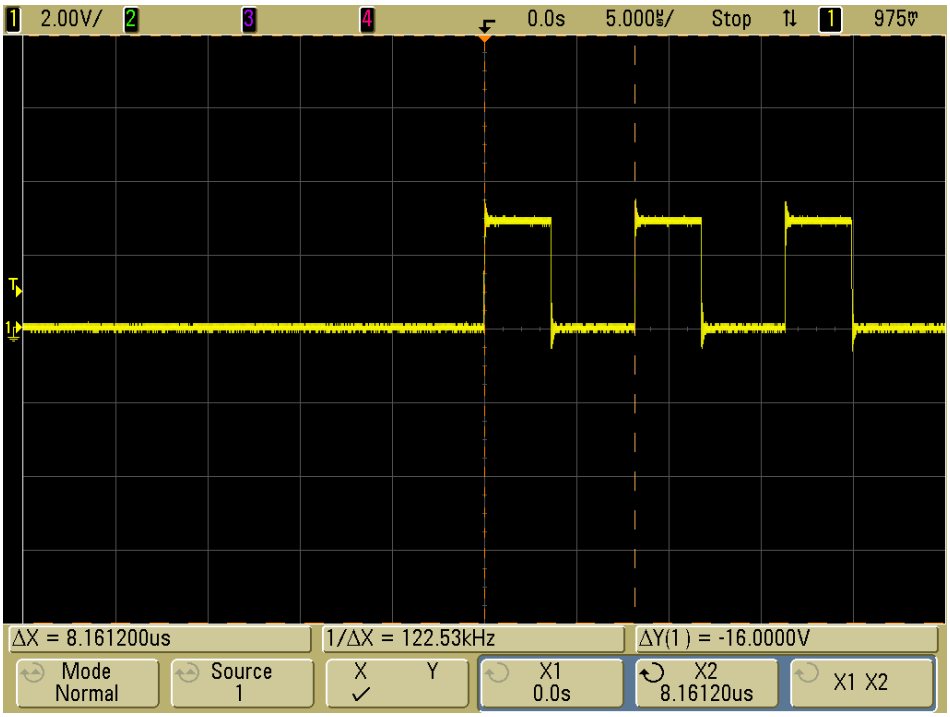

Figure 5-8 PWM Example3-1

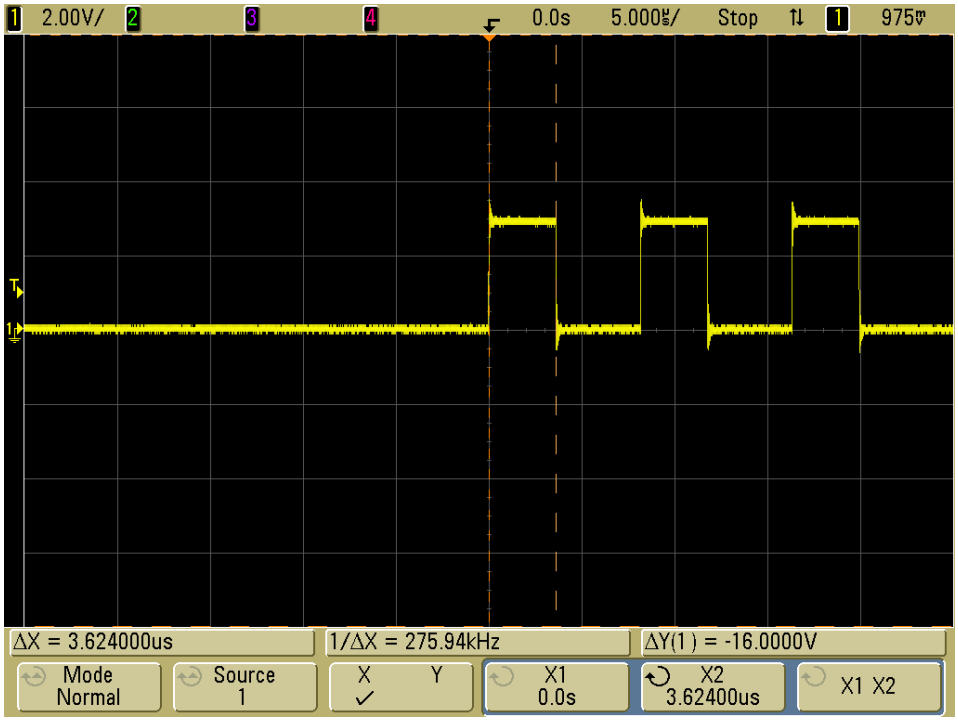

Figure 5-9 PWM Example3-2

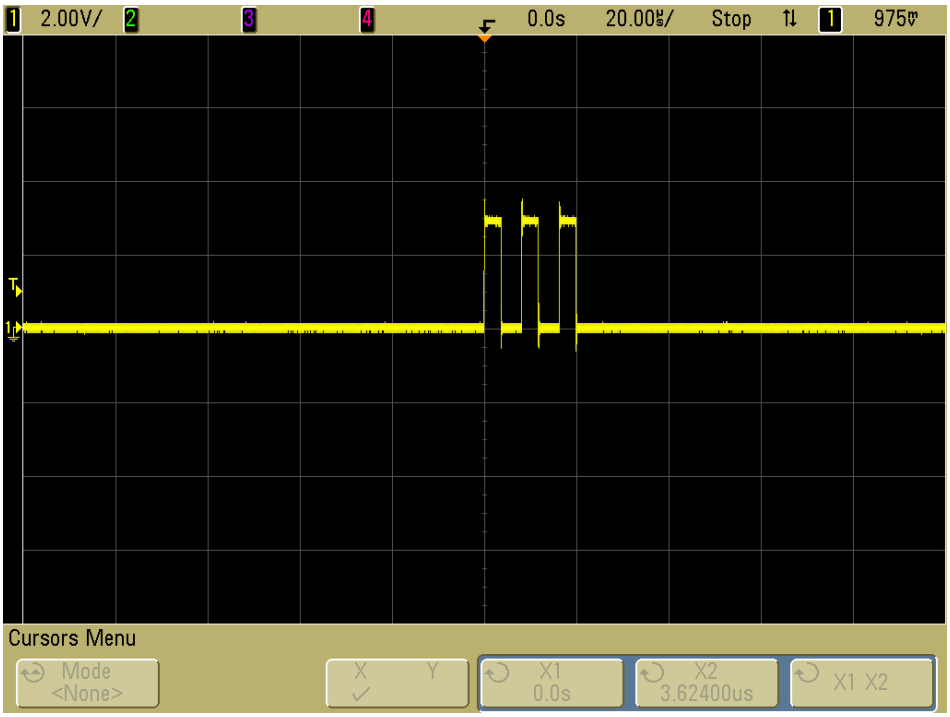

Figure 5-10 PWM Example3-3

In this sample, PWM_PWCLK0 is come from PLL3. The frequency of PLL3 is 229.376MHz and the divisor is 208 set by ASMU_DIVPWMPWCLK, so the PWM_PWCLK0 is:
PWM_PWCLK0 = PLL3 / divisor = 229.376MHz / 208 = 1.10MHz;

As the figure 5-8, the PWM0 output frequency is:
PLL3 / divisor / PWM_CH0_TOTAL0 = 229.376 / 208 / 9 = 122.53 KHz;
PWM0 output cycle is:
1 PWM_PWCLK0 cycle * PWM_CH0_TOTAL0 = (208/229.376) * 9 = 8.16us;

As the figure 5-9, the PWM0 output duty is:
(PWM_CH0_TEDGE0 - PWM_CH0_LEDGE0) / PWM_CH0_TOTAL0 = (8 – 4) / 9 = 44.4%;
The high level width of PWM0 is:
(PWM_CH0_TEDGE0 - PWM_CH0_LEDGE0) / PWM_CH0_TOTAL0 * 1 PWM0 cycle
= (8-4)/9 * 8.16 = 3.62us;

As the figure 5-10, the loop number is 3 set by PWM_CH0_LOOP.

5.6 Sample4 (Enable three counters)

5.6.1 Operation Flow

About the figure, please refer to the figure 5-3.

5.6.2 Operation Detail

(1). Set PWM1 parameter.

```
PWM_SETUP_ST pwm_st;

pwm_st.channel = 1;                 // Select the channel 1;
pwm_st.mode.MOD = 1;                // Select the AND mode;
pwm_st.mode.CMP_EN[0] = 1;          // Enable counter 0;
pwm_st.mode.CMP_ATST[0] = 0;        // Disable automatic stop;
pwm_st.mode.CMP_INTV[0] = 0;        // Disable inversion;
pwm_st.mode.CMP_EN[1] = 1;          // Enable counter 1;
pwm_st.mode.CMP_ATST[1] = 0;        // Disable automatic stop;
pwm_st.mode.CMP_INTV[1] = 0;        // Disable inversion;
pwm_st.mode.CMP_EN[2] = 1;          // Enable counter 2;
pwm_st.mode.CMP_ATST[2] = 0;        // Disable automatic stop;
pwm_st.mode.CMP_INTV[2] = 0;        // Disable inversion;
pwm_st.TOTAL[0] = 0x00000006;
pwm_st.DELAY[0] = 0x00000001;
pwm_st.LEDGE[0] = 0x00000002;
pwm_st.TEDGE[0] = 0x00000005;
pwm_st.TOTAL[1] = 0x00000003;
pwm_st.DELAY[1] = 0x00000001;
pwm_st.LEDGE[1] = 0x00000001;
pwm_st.TEDGE[1] = 0x00000003;
pwm_st.TOTAL[2] = 0x0000000C;
pwm_st.DELAY[2] = 0x00000001;
pwm_st.LEDGE[2] = 0x00000003;
pwm_st.TEDGE[2] = 0x00000006;
em1_pwm_setup(&pwm_st);
```

The em1_pwm_setup() finishes the following functions for PWM1:

- Disable automatic stop and inversion of PWM_CMP0, PWM_CMP1 and PWM_CMP2.
  PWM_CH1_MODE = 0x17;

- Set delay value PWM_PWCLK x 1 cycle.
  PWM_CH1_DELAY0 = 0x1;

- Set leading setting 2 cycles.
  PWM_CH1_LEDGE0 = 0x2;

- Set trailing setting 5 cycles
  PWM_CH1_TEDGE0 = 0x5;

- Set total cycle PWM_PWCLK x 6 cycles
  PWM_CH1_TOTAL0 = 0x6;

- Set delay value PWM_PWCLK x 1 cycle.
  PWM_CH1_DELAY1 = 0x1;

- Set leading setting 1 cycle.
  PWM_CH1_LEDGE1 = 0x1;

- Set trailing setting 3 cycles
  PWM_CH1_TEDGE1 = 0x3;

- Set total cycle PWM_PWCLK x 3 cycles
  PWM_CH1_TOTAL1 = 0x3;

- Set delay value PWM_PWCLK x 1 cycle.
  PWM_CH1_DELAY2 = 0x1;

- Set leading setting 3 cycles.
  PWM_CH1_LEDGE2 = 0x3;

- Set trailing setting 6 cycles
  PWM_CH1_TEDGE2 = 0x6;

- Set total cycle PWM_PWCLK x 12 cycles
  PWM_CH1_TOTAL2 = 0xC;

(2). Start PWM1.
The process calls the em1_pwm_start(1).
The em1_pwm_start() finishes the following functions for PWM1:

- Start PWM1.
  PWM_CH1_CTRL = 0x1;

(3). Stop PWM1.

The process calls the em1_pwm_stop(1).

The em1_pwm_stop() finishes the following functions for PWM1:

- Stop PWM1.
  PWM_CH1_CTRL = 0x0;

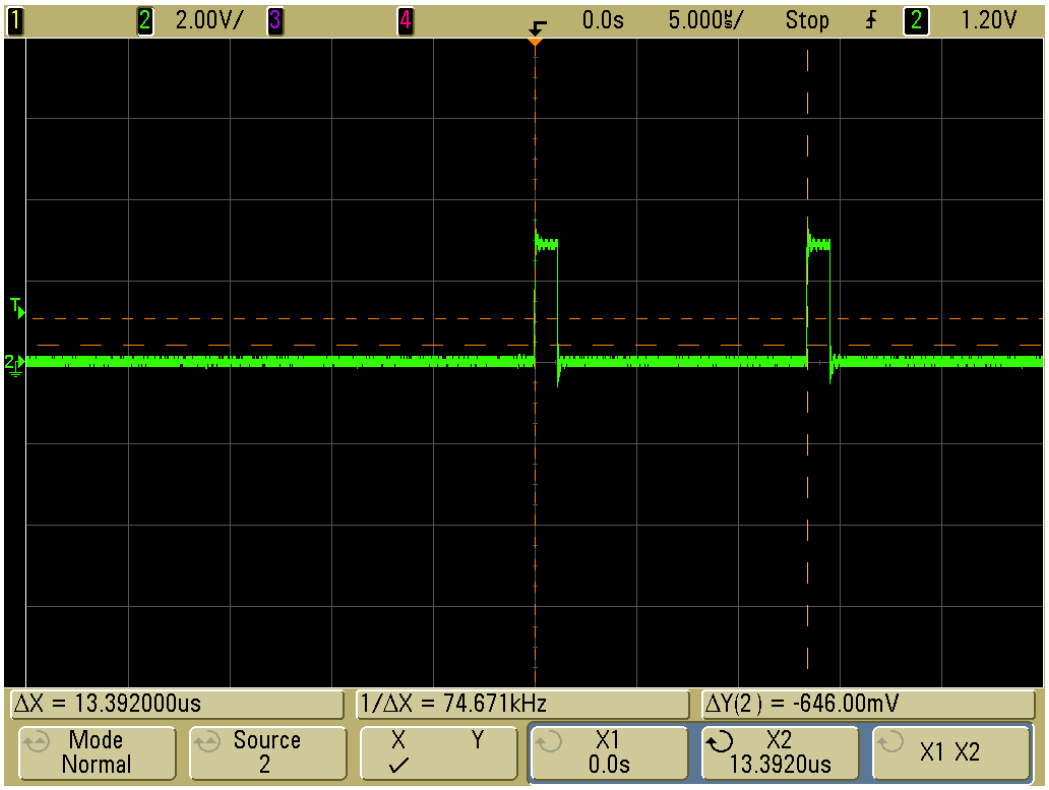

Figure 5-11 PWM Example4-1

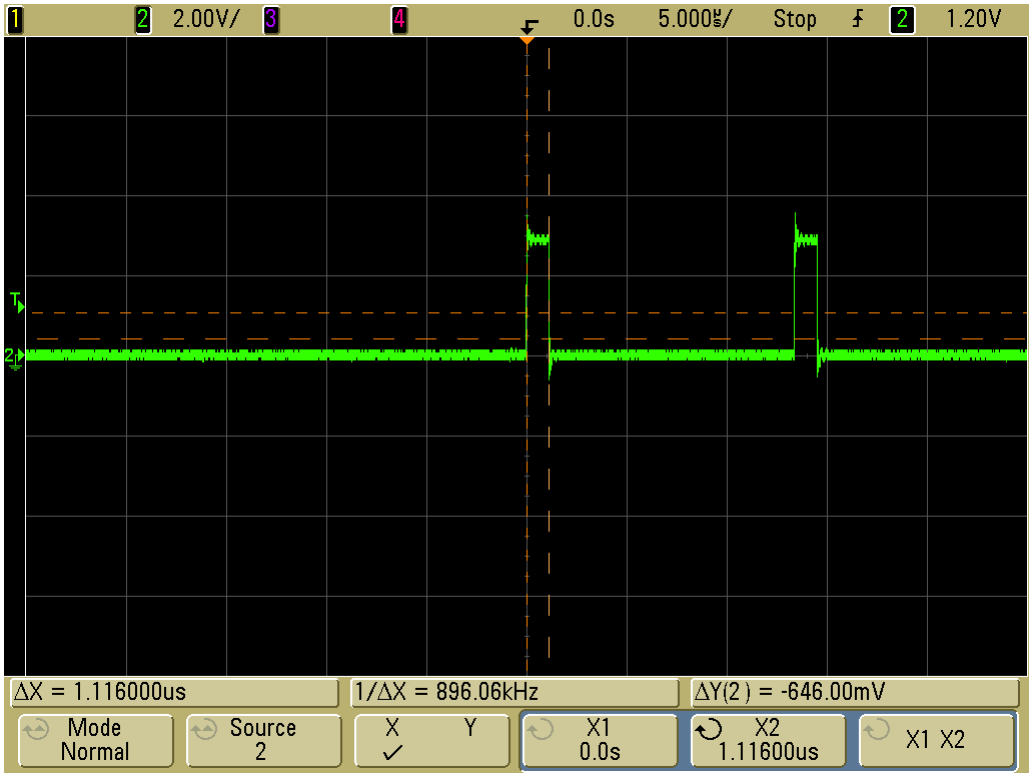

Figure 5-12 PWM Example4-2

In this sample, PWM_PWCLK1 is come from PLL3. The frequency of PLL3 is 229.376MHz and the divisor is 256 set by ASMU_DIVPWMPWCLK, so the PWM_PWCLK1 is:
PWM_PWCLK1 = PLL3 / divisor = 229.376MHz / 256 = 0.896MHz;

As the figure 5-11, the PWM1 output frequency is:

PLL3 / divisor / PWM_CH1_TOTALx = 229.376 / 256 / 12 = 74.67 KHz;

PWM1 output cycle is:

1 PWM_PWCLK1 cycle * PWM_CH1_TOTALx = (256/229.376) * 12 = 13.39us;

Note: PWM_CH1_TOTALx is the smallest common multiple of PWM_CH1_TOTAL0, PWM_CH1_TOTAL1 and PWM_CH1_TOTAL2.

As the figure 5-12, the high level width of PWM1 is:

High level proportion * 1 PWM1 cycle = 1/12 * 13.39 = 1.11us;

Note: the “High level proportion” is the common high level part of PWM_CMP0, PWM_CMP1 and PWM_CMP2.

About the details, please refer to the “**Chapter 4.2.1 PWM operation**” of Audio/Voice and PWM user’s manual.

Appendix A Audio/Voice Driver Function

A.1 Audio/Voice Driver Function List

The following table shows the Audio/Voice driver interface functions:

Table A-1 Audio/Voice Driver Function List

Class	Function Name	Function Detail
Driver Function	em1_pcm_init;	Initialize the PCM;
	em1_pcm_setup;	Set the PCM parameter;
	em1_pcm_start;	Start the PCM;
	em1_pcm_stop;	Stop the PCM;

A.2 Audio/Voice Structure Define

The following table shows the Audio/Voice structure define:

Table A-2 Structure Define

Structure Name	Detail
st_PCM_SETTING	PCM configure information.

A.2.1 st_PCM_SETTING

Table A-3 Structure of st_PCM_SETTING

Member	Type	Detail
num_TX_FIFO	uchar	The number of valid data items in the TX data FIFO.
mode_m_s	uchar	Master mode/slave mode.
mode_sel	uchar	Transmission format.
TX_padding	uchar	Transmit data padding.
RX_padding	uchar	Receive data padding.
SOB	uchar	Transmit bits.
SIB	uchar	Receive bits.
cyc_val	uchar	The frame length or words.
SOB2	uchar	Transmit bits.
SIB2	uchar	Receive bits.
cyc_val2	uchar	The words of phase 2 per frame.
channel	uchar	PCM number.

A.3 Audio/Voice Driver Function Detail

A.3.1 Initialize PCM

[Function Name]

em1_pcm_init

[Format]

DRV_RESULT em1_pcm_init(uchar num, uint div);

[Argument]

Parameter	Type	I/O	Detail
num	uchar	I	PCM number 0: PCM0 device 1: PCM1 device
div	uint	I	Clock divisor

[Function Return]

DRV_OK: The function executes successfully.

DRV_ERR_PARAM: The input parameter is error.

[Flow Chart]

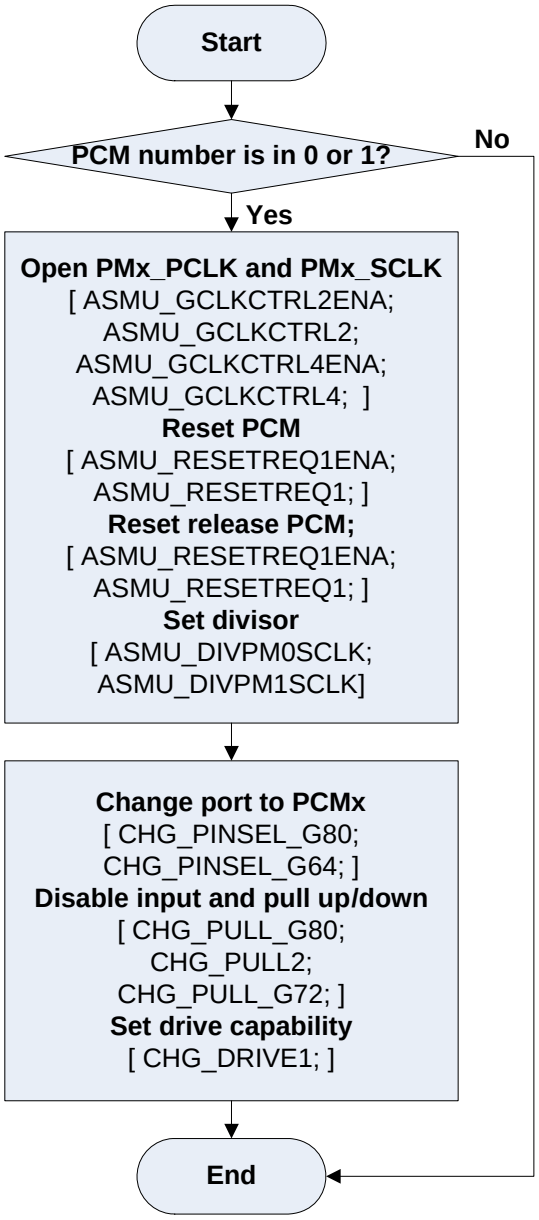

Figure A-1 Initialize PCM

A.3.2 Setup PCM

[Function Name]

em1_pcm_setup

[Format]

DRV_RESULT em1_pcm_setup(st_PCM_SETTING * setup_st);

[Argument]s

Parameter	Type	I/O	Detail
setup_st	st_PCM_SETTING	I	PCM information

[Function Return]

DRV_OK: The function executes successfully.

DRV_ERR_PARAM: The input parameter is error.

[Flow Chart]

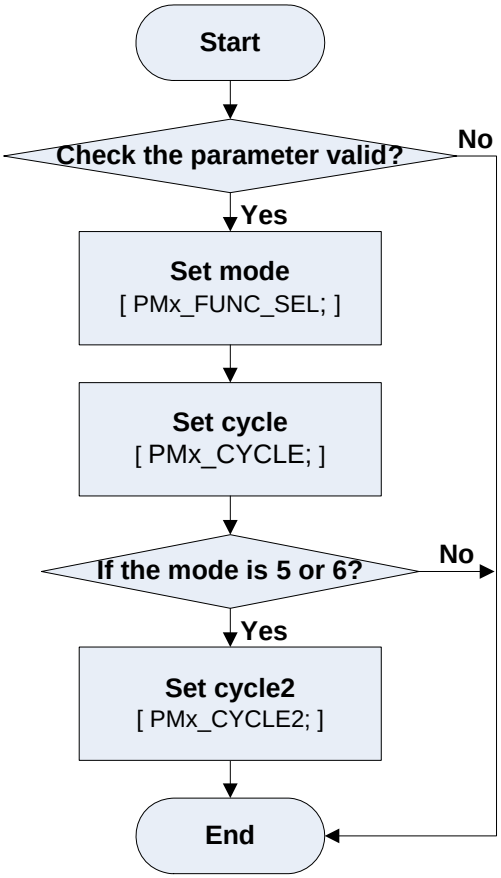

Figure A-2 Setup PCM

Note: x is 0 or 1.

[Note]

(1). Check the parameter valid.

The input parameter for register should not exceed the register scope.

A.3.3 Start PCM

[Function Name]

em1_pcm_start

[Format]

DRV_RESULT em1_pcm_start(uchar channel, uchar tx_rx);

[Argument]

Parameter	Type	I/O	Detail
channel	uchar	I	PCM number 0: PCM0 device 1: PCM1 device
tx_rx	uchar	I	Transmit or receive enable

[Function Return]

DRV_OK: The function executes successfully.

DRV_ERR_PARAM: The input parameter is error.

[Flow Chart]

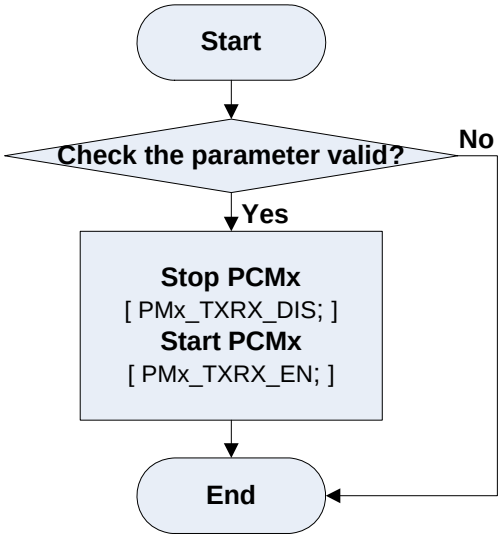

Figure A-3 Start PCM

Note: x is 0 or 1.

[Note]

(1). Check the parameter valid.

The input parameter for register should not exceed the register scope.

A.3.4 Stop PCM

[Function Name]

em1_pcm_stop

[Format]

DRV_RESULT em1_pcm_stop(uchar channel, uchar tx_rx);

[Argument]

Parameter	Type	I/O	Detail
channel	uchar	I	PCM number 0: PCM0 device 1: PCM1 device
tx_rx	uchar	I	Transmit or receive disable

[Function Return]

DRV_OK: The function executes successfully.

DRV_ERR_PARAM: The input parameter is error.

[Flow Chart]

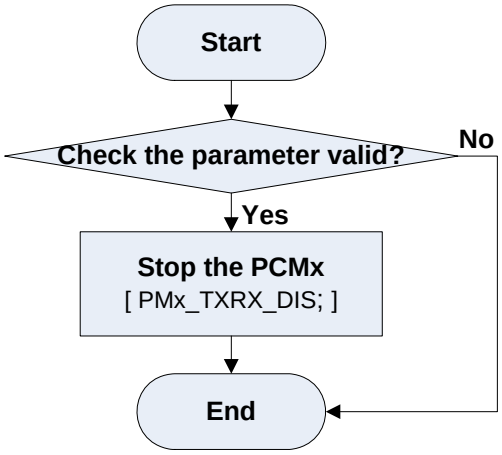

Figure A-4 Stop PCM

Note: x is 0 or 1.

[Note]

(1). Check the parameter valid.

The input parameter for register should not exceed the register scope.

Appendix B PWM Driver Function

B.1 PWM Driver Function List

The following table shows the PWM driver interface functions:

Table B-1 PWM Driver Function List

Class	Function Name	Function Detail
Driver Function	em1_pwm_init;	Initialize the PWM;
	em1_pwm_setup;	Set the PWM parameter;
	em1_pwm_start;	Start the PWM;
	em1_pwm_stop;	Stop the PWM;

B.2 PWM Structure Define

The following table shows the PWM structure define:

Table B-2 Structure Define

Structure Name	Detail
st_PWM_MODE	PWM mode information.
st_PWM_SETTING	PWM cycle information.

B.2.1 st_PWM_MODE

Table B-3 Structure of st_PWM_MODE

Member	Type	Detail
MOD	uchar	Output mode.
CMP_INTV[3]	uchar	Output level.
CMP_ATST[3]	uchar	Automatically termination method.
CMP_EN[3]	uchar	Enable CMP0, CMP1 and CMP2.

B.2.2 st_PWM_SETTING

Table B-4 Structure of st_PWM_SETTING

Member	Type	Detail
channel	uchar	PWM channel.
mode	st_PWM_MODE	PWM operation mode.
DELAY[3]	uchar	Delay value.
LEDGE[3]	uchar	Leading edge period.
TEDGE[3]	uchar	Trailing edge period.
TOTAL[3]	uchar	Total cycle.
LOOP[3]	uchar	Counter loop count.

B.3 PWM Driver Function Detail

B.3.1 Initialize PWM

[Function Name]

em1_pwm_init

[Format]

DRV_RESULT em1_pwm_init(uchar num, uint div);

[Argument]

Parameter	Type	I/O	Detail
num	uchar	I	PWM number 0: PWM0 device 1: PWM1 device
div	uint	I	Clock divisor

[Function Return]

DRV_OK: The function executes successfully.

DRV_ERR_PARAM: The input parameter is error.

[Flow Chart]

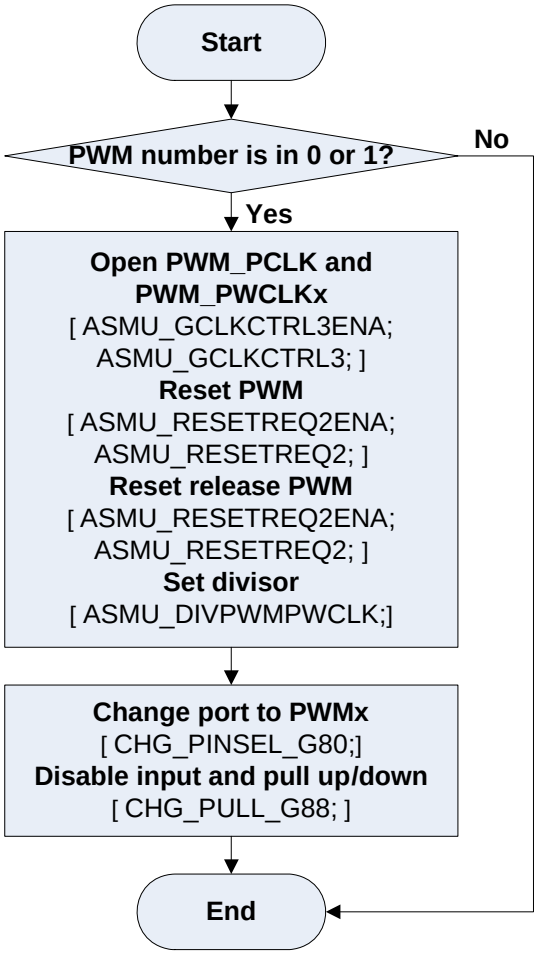

Figure B-1 Initialize PWM

B.3.2 Setup PWM

[Function Name]

em1_pwm_setup

[Format]

DRV_RESULT em1_pwm_setup(st_PWM_SETTING * setup_st);

[Argument]

Parameter	Type	I/O	Detail
setup_st	st_PWM_SETTING *	I/O	PWM information

[Function Return]

DRV_OK: The function executes successfully.

DRV_ERR_PARAM: The input parameter is error.

[Flow Chart]

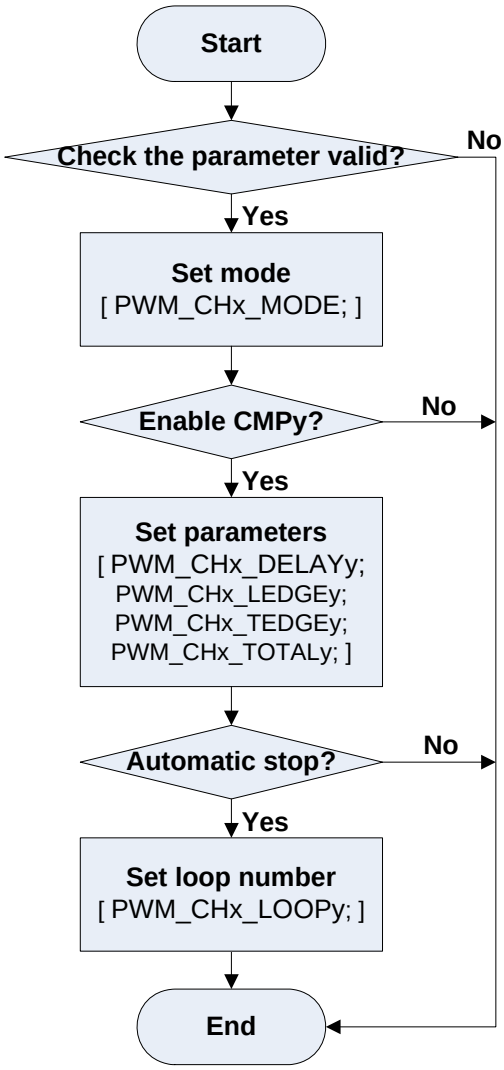

Figure B-2 Setup PWM

Note: x is 0 or 1; y is 0, 1 or 2.

[Note]

(1). Check the parameter valid.

The input parameter for register should not exceed the register scope.

B.3.3 Start PWM

[Function Name]

em1_pwm_start

[Format]

DRV_RESULT em1_pwm_start(uchar channel);

[Argument]

Parameter	Type	I/O	Detail
channel	uchar	I	PWM number 0: PWM0 device 1: PWM1 device

[Function Return]

DRV_OK: The function executes successfully.

DRV_ERR_PARAM: The input parameter is error.

[Flow Chart]

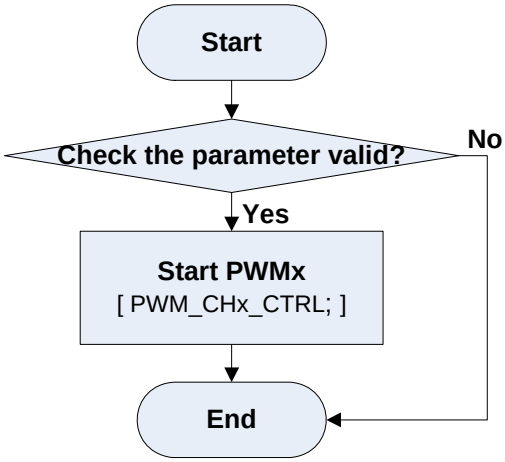

Figure B-3 Start PWM

Note: x is 0 or 1.

[Note]

(1). Check the parameter valid.

The input parameter for register should not exceed the register scope.

B.3.4 Stop PWM

[Function Name]

em1_pwm_stop

[Format]

DRV_RESULT em1_pwm_stop(uchar channel);

[Argument]

Parameter	Type	I/O	Detail
channel	uchar	I	PWM number 0: PWM0 device 1: PWM1 device

[Function Return]

DRV_OK: The function executes successfully.

DRV_ERR_PARAM: The input parameter is error.

[Flow Chart]

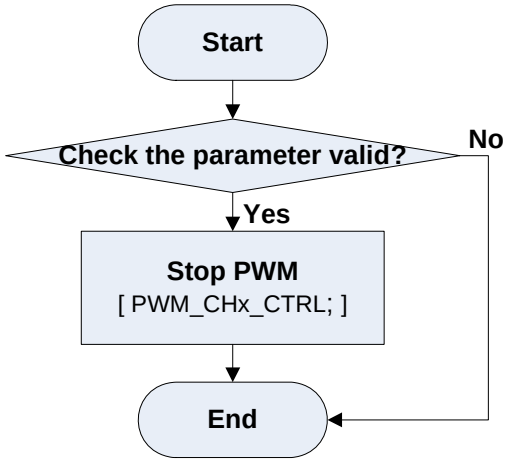

Figure B-4 Stop PWM

Note: x is 0 or 1.

[Note]

(1). Check the parameter valid.

The input parameter for register should not exceed the register scope.

ANNEX Modification History

Number	Modification Contents	Author	Date
Ver 1.00	New version		Aug, 4, 2009