User Manual

User Manual

Customer Support

MT6757

Doc No:	CS6000-D8C-UMD-V1.0EN
Version:	V1.0
Release date:	2016-12-30
Classification:	Internal

`

Keywords

User Manual

MediaTek Inc.

Postal address

No. 1, Dusing 1st Rd. , Hsinchu Science Park, Hsinchu City, Taiwan 30078

MTK support office address

No. 1, Dusing 1st Rd. , Hsinchu Science Park, Hsinchu City, Taiwan 30078

Internet

http://www.mediatek.com/

Document Revision History

Revision	Date	Author	Description
V1.0	2016-08-29	Ting-Hao. Lin	The first release.

Table of Contents

Lists of Tables

Lists of Figures

1 Introduction

1.1 Purpose

This document provides the user guidelines for the PWM and associated modules. It describes how to bring up the PWM on the Android platform. This manual also elaborates the mechanism required to use the PWM.

We will provide the following in this document.

1. What is PWM
2. How to set up PWM to send wave form.

1.2 Scope

The document provide the customization details of the PWM.

Table 1-1 presents the reference information of the modules which are used but beyond the scope.

Table 1-1. Reference Information beyond Scope

Modules	Reference information
PWM driver	The PWM driver.

This document applyling plateform is MT6757. In this version, the memory mode is different from other chips. Kernel version is 4.4.

1.3 Who Should Read This Document

This document is primarily intended for:

- Customers who integrate the PWM with user-defined applications

1.4 How to Use This Manual

This segment explains how information is distributed in this document, and presents some cues and examples to simplify finding and understanding information in this document. Table 1-2 presents an overview of the chapters and appendices in this document.

Table 1-2. Chapter Overview

#	Chapter	Contents
1	Introduction	Describes the scope and layout of this document.
2	References	References of this document
3	Definitions	The definitions of terms in this document
4	Abbreviations	The abbreviations of terms in this document
5	Process Overview	Gives a brief description of the modules of the system
6	Function	Explain common API in pwm driver

#	Chapter	Contents
7	PWM configuration Guideline	Gives a detail description of PWM configuration data structure
8	Workflow	Explains how to configure PWM for sending wave in different mode and gives examples
9	Trouble shooting	Explains why pwm may not work

1.4.1 Terms and Conventions

This document uses special terms and typographical conventions to help you easily identify various information types in this document. These cues are designed to simply finding and understanding the information this document contains.

Table 1-3. Conventions

Convention	Usage	Example
N/A	N/A	N/A

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

[1] MTK Company Profile, http://brandclips.mediatek.inc/uploads/Company-profile-1H-2016_0418-Lite-final.pptx

[2] PWM wiki, https://en.wikipedia.org/wiki/Pulse-width_modulation

3 Definitions

For the purposes of the present document, the following terms and definitions apply:

Pulse-width modulation (PWM) : is a modulation technique used to encode a message into a pulsing signal. Although this modulation technique can be used to encode information for transmission, its main use is to allow the control of the power supplied to electrical devices, especially to inertial loads such as motors. In addition, PWM is one of the two principal algorithms used in photovoltaic solar battery chargers, the other being maximum power point tracking [2]

4 Abbreviations

Table 4-1. Abbreviations

Abbreviations	Explanation
MTK	MediaTek, Asia's largest fabless IC design company.
PWM	Pulse Width Modulation

5 Process Overview

This chapter first gives a brief description of the modules of the system and the relationship of the modules.

5.1 Process

Three generic pulse-width modulators are implemented to generate pulse sequences with programmable frequency and duration for LCD backlight, charging or other purposes. Before enabling PWM, the pulse sequences must be prepared in the memory or registers. Then PWM will read the pulse sequences to generate random waveform to meet all kinds of applications (see **Error! Reference source not found.**).

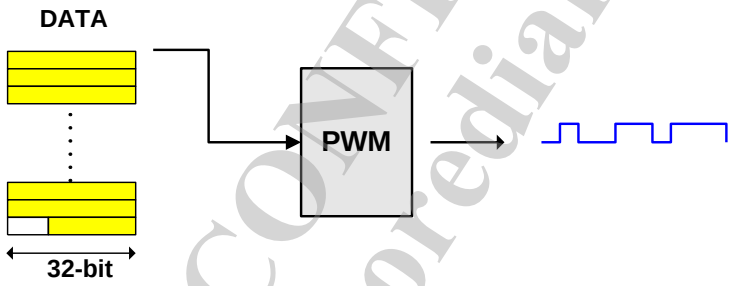


Figure 1. Generation procedure of PWM

5.2 Key Features

☞ List the key features or key workflows and give some introduction about the features or workflows.

PWM can support 3 modes waveform, which are OLD, FIFO and MEMORY. You can choose one to send waveform. The FIFO and MEMORY are frequently used modes. If the change of transmitting wave is not very larger in general, we can directly use FIFO MODE.

1. OLD mode : send the same waveform repeatly. It is easy to set up, but difficult to change the shap of waveform.
2. FIFO mode : send the wave according to the FIFO data. The data 1 means ON (HIGH), 0 means OFF (LOW). The data length is at most 64 bit.
3. MEMORY mode : moving the specific memory data to FIFO, sending the wave according to the data. You can specify the length of memory.

6 Function

6.1 Clock source

The frequency of waveform depends on the clock source, you can config the clock as following.

Bclk can be selected as 26MHz or 62MHz(bus clock) by PWM_CK_26_SEL.

Mode	OLD_MODE	CLKSEL_OLD	CLKSEL	CLKSRC
Old mode	1	Don't care	0	bclk
Old mode	1	1	1	32kHz (used in old mode only)
Old mode	1	0	1	bclk/1625
FIFO mode	0	Don't care	0	bclk
FIFO mode	0	Don't care	1	bclk/1625

Table 2.PWM clock source configuration

NOTE：Even in the situation of OLD MODE，CLKSRC = 32 KHz, PWM still not work in the sleep mode of system.

NOTE：OLD mode won't use 32K Hz to be CLKSRC for performance.

NOTE：When choose Block clock/1625, i.e. 26M/1625 = 16K Hz

7 PWM Configuration Guideline

In this chapter, we will gives customer how to configure the PWM. There are two mainly data struncture, pwm_easy_config and pwm_spec_config. pwm_easy config is only for old mode. We suggest you use pwm_spec_config and we will give examples.

7.1 Interface function

The function to control the CLK is provided as follows:

void mt_pwm_power_on(U32 pwm_no, BOOL pmic_pad)

void mt_pwm_power_off(U32 pwm_no, BOOL pmic_pad)

pwm_no: pwm channel index (enum PWM1~PWM4)

pmic_pad: 6757 don't support this. Just ignore.

NOTE: This function will turn off the PWM CLK, user is best not to modify this function because the misuse will influences the normal work of other PWM waveform.

7.2 pwm_set_spec_config

7.2.1 Interface function

This function is the basic call to PWM driver. The following example will analyze the use of this function.

Example:

```
int pwm_set_spec_example()
{
    struct pwm_spec_config pwm_setting; // reference pwm_spec_config structure
    /*initialize the member of pwm_spec_config structure*/
     pwm_setting.pwm_no = PWM1;
    pwm_setting.mode = PWM_MODE_FIFO;
    pwm_setting.clk_div = CLK_DIV1;
     pwm_setting.clk_src = PWM_CLK_NEW_MODE_BLOCK;
```

```
    pwm_setting.pmic_pad = false;
    pwm_setting.PWM_MODE_FIFO_REGS.IDLE_VALUE = FALSE;
    pwm_setting.PWM_MODE_FIFO_REGS.GUARD_VALUE = FALSE;
    pwm_setting.PWM_MODE_FIFO_REGS.STOP_BITPOS_VALUE = 63;
    pwm_setting.PWM_MODE_FIFO_REGS.HDURATION = 1;
    pwm_setting.PWM_MODE_FIFO_REGS.LDURATION = 1;
    pwm_setting.PWM_MODE_FIFO_REGS.GDURATION = 0;
    pwm_setting.PWM_MODE_FIFO_REGS.WAVE_NUM = 0;
    /*call pwm_set_spec_config() function, the function defined in mt_pwm.c */
    pwm_set_spec_config(&pwm_setting);
}
```

Firstly, initialize the pwm_spec_config structure, then call the pwm_set_spec_config() function. It will have a detailed explanation about pwm_spec_config structure as follows.

7.2.2 Data structure configuration (pwm_spec_config)

```
struct pwm_spec_config {
    U32 pwm_no;        //PWM1~PWM4
    U32 mode;          //the mode of PWM
    U32 clk_div;       //the frequency divider of clock source
    U32 clk_src;       //clock source
    /*the following are the specific definitions of PWM MODE*/
      union {
          struct _PWM_OLDMODE_REGS {
              U16 IDLE_VALUE;
              U16 GUARD_VALID;
              U16 GDURATION;
              U16 WAVE_NUM;
              U16 DATA_WIDTH;
              U16 THRESH;
          }PWM_MODE_OLD_REGS;
```

```
        struct _PWM_MODE_FIFO_REGS {
                U16 IDLE_VALUE;
                U16 GUARD_VALID;
                U16 STOP_BITPOS_VALUE;
                U16 HDURATION;
                U16 LDURATION;
                U32 GDURATION;
                U32 SEND_DATA0;
                U32 SEND_DATA1;
                U32 WAVE_NUM;
        }PWM_MODE_FIFO_REGS;
    }
        struct _PWM_MODE_MEMORY_REGS {
                u32 IDLE_VALUE;
                u32 GUARD_VALUE;
                u32 STOP_BITPOS_VALUE;
                u16 HDURATION;
                u16 LDURATION;
                u16 GDURATION;
                dma_addr_t BUF0_BASE_ADDR;
                u32 BUF0_SIZE;
                u16 WAVE_NUM;
        } PWM_MODE_MEMORY_REGS;
}
```

Detail introduction of the initialization process as follows

8 Workflow

This chapter gives a detail description about the every PWM mode configuration.

8.1 Old mode

```
U32 pwm_no;
U32 mode;
U32 clk_div;
U32 clk_src;
BOOL pmic_pad; // not used in MT6572
  union {
      struct _PWM_OLDMODE_REGS {
          U16 IDLE_VALUE;
          U16 GUARD_VALID;
          U16 GDURATION;
          U16 WAVE_NUM;
          U16 DATA_WIDTH;
          U16 THRESH;
      }PWM_MODE_OLD_REGS;
      ..
```

1：pwm_no: selecting the required PWM number, there are PWM1~ PWM4 for user to select;

2：mode：choose PWM_MODE_OLD；

3：clk_src：PWM_CLK_OLD_MODE_BLOCK,PWM_CLK_OLD_MODE_32K；

4： clk_div: CLK_DIV1, CLK_DIV2, CLK_DIV4, CLK_DIV8, CLK_DIV16, CLK_DIV32, CLK_DIV64, CLK_DIV128；

5：pmic_pad：Just ignore in MT6757;

6：IDLE_VALUE：PWM output when in idle time, it has 2 values: FALSE or TRUE. FALSE: The level of GPIO is low after the end of waveform transmission. TURE: The level of GPIO is high after the end of waveform transmission;

7：GUARD_VALID: FALSE or TRUE. FALSE: there is no guarding interval between 2 complete waveforms. TURE: it has guarding interval between two complete waveforms, guarding interval decided by the value of DATA_WIDTH;

（when in old mode, the value of GDURATION is invalid, which is controlled by DATA_WIDTH）

8：WAVE_NUM: number of waveform repeat, if WAVE_NUM =0, the waveform generate will not stop until the PWM is disabled;

9：DATA_WIDTH: control the time of a complete waveform, the time calculation formula:

T = 1/(clk_src/clk_div) * (DATA_WIDTH + 1)

Ex:clk_src choose 66 MHz，clk_div choose 1，DATA_WIDTH = 99

T = 1/(66/1) * (99 + 1) = 1.52 usec

10：THRESH: control the high/low time in a complete waveform, also called duty ration, calculation formula:

T = 1/(clk_src/clk_div) * (THRESH + 1)

Ex:clk_src choose 66 MHz，clk_div choose 1，THRESH = 49

T = 1/(66/1) * (49 + 1) = 0.76 usec

After the completion of all parameters' setting, user can call pwm_set_spec_config() to write the value of parameter into corresponding register.

8.2 FIFO mode

FIFO MODE is a very commonly used PWM mode, when the size of transmission data is not greater than 64 bits, we can directly write the data into send_data0 and send_data1, thus it can transmit waveform quickly.

```
U32 pwm_no;
U32 mode;
U32 clk_div;
U32 clk_src;
....
struct _PWM_MODE_FIFO_REGS {
    U16 IDLE_VALUE;
    U16 GUARD_VALID;
    U16 STOP_BITPOS_VALUE;
    U16 HDURATION;
    U16 LDURATION;
```

U32 GDURATION;

U32 SEND_DATA0;

U32 SEND_DATA1;

U32 WAVE_NUM;

}PWM_MODE_FIFO_REGS;

1：pwm_no: selecting the required PWM number, there are PWM1 ~ PWM4 for user to select.

2：mode：choose PWM_MODE_FIFO；

3：clk_src ： PWM_CLK_NEW_MODE_BLOCK, PWM_CLK_NEW_MODE_BLOCK_DIV_BY_1625;

4 ： clk_div ： CLK_DIV1, CLK_DIV2, CLK_DIV4, CLK_DIV8, CLK_DIV16, CLK_DIV32, CLK_DIV64, CLK_DIV128；

5：IDLE_VALUE：PWM output when in idle time, it has 2 values: FALSE or TRUE. FALSE: The level of GPIO is low after the end of waveform transmission. TURE: The level of GPIO is high after the end of waveform transmission;

6：GUARD_VALID: FALSE or TRUE. FALSE: there is no guarding interval between 2 complete waveforms. TURE: it has guarding interval between 2 complete waveforms, guarding interval decided by the value of GDURATION;

7：GDURATION：when GUARD_VALID is set to 1，GDURATION is valid. It expresses the interval between the two completed waveforms. The time calculation formula:

T = 1/(clk_src/clk_div) * (GDURATION + 1)

Ex:clk_src choose 66 MHz，clk_div choose 1，GDURATION = 1

T = 1/(66/1) * (1 + 1) = 30 nsec

8：HDURATION and LDURATION: the duration of the high/low ,calculation formula:

T = 1/(clk_src/clk_div) * (HDURATION + 1)

Ex:clk_src choose 66MHz, clk_div choose 1, HDURATION = 1

T = 1/(66/1) * (1 + 1) = 30 nsec

T = 1/(clk_src/clk_div) * (LDURATION + 1)

Ex:clk_src choose 66MHz,clk_div choose 1, LDURATION = 2

T = 1/(66/1) * (2 + 1) = 45.45 nsec

NOTE：The value of HDURATION and LDURATION could not be -1 or less than 0. And they could be different value.

9：STOP_BITPOS_VALUE：In FIFO MODE, it is used to indicate the stop bit position of send_data0 and send_data1, its maximum value is 63 (because the maximum of transfer data is 64bits). The value of STOP_BITPOS_VALUE can determine the transfer data:

Ex： Senddata0：0x0000_FF11

Senddata1：0xFFFF_FFFF

STOP_BITPOS_VALUE = 63, all bits have been transferred. As shown:

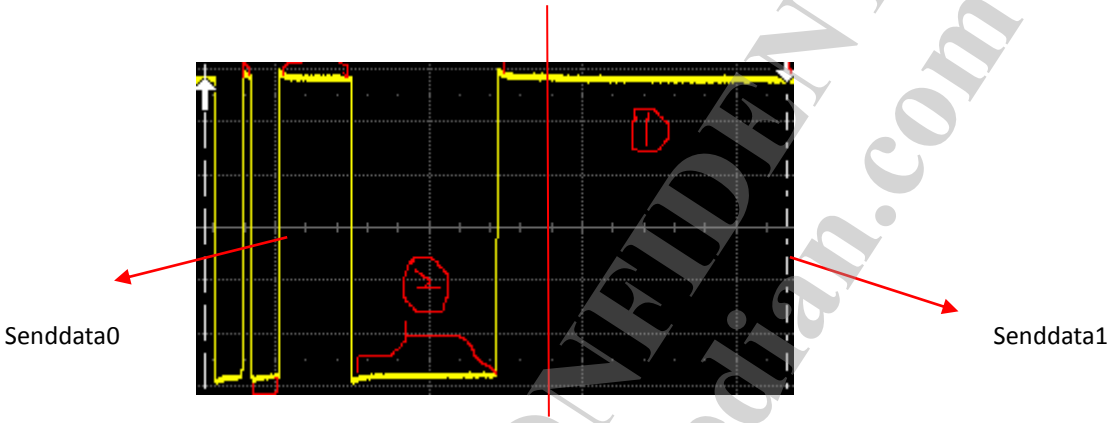


Figure 2. PWM waveform example

STOP_BITPOS_VALUE = 31, only 32 bits (Senddata0) have been transferred. As shown:

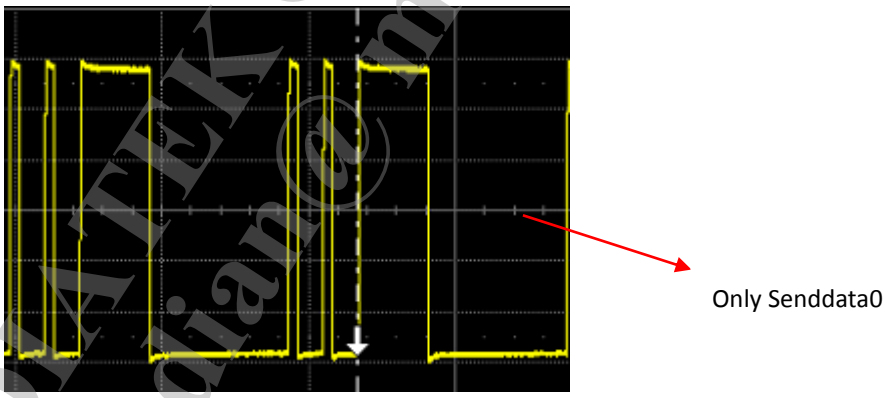


Figure 3. PWM waveform example

STOP_BITPOS_VALUE = 15, only 16 bits have been transferred. As shown:

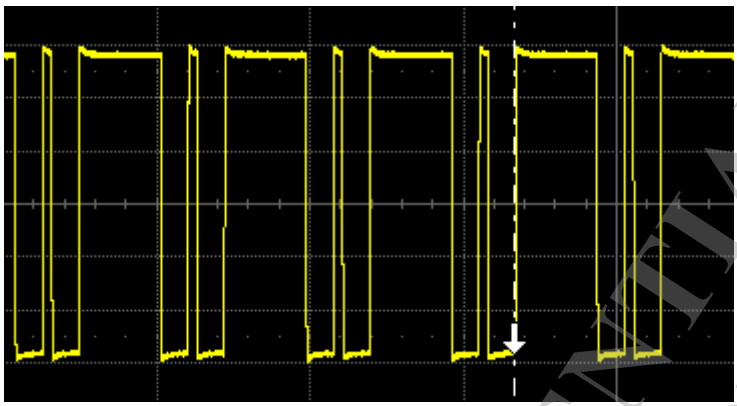


Figure 4. PWM waveform example

STOP_BITPOS_VALUE = 7, only 8 bits have been transferred. As shown:

Figure 5. PWM waveform example

10: Send_data0 and Send_data1: Local buffer of pulse sequence data to be generated, 1: high, 0: low.

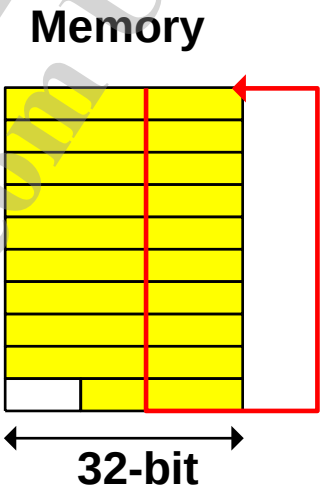
11: wave_num: Number by which PWM will generate from pulse data repeatedly. If wave_num=0, the waveform generation will not stop until it is disabled.

After the completion of all parameters' setting, user can call pwm_set_spec_config() to write the value of parameter into corresponding register.

8.3 Memory mode

Memory MODE is also a very commonly used PWM mode, we can directly write the data in specific memory. Memory mode and FIFO mode are roughly the same. The differences are that memory mode will move the data from memory to FIFO and the length of data can be larger.

```
U32 pwm_no;
U32 mode;
U32 clk_div;
U32 clk_src;
BOOL pmic_pad;
....
struct _PWM_MODE_MEMORY_REGS {
    u32 IDLE_VALUE;
    u32 GUARD_VALUE;
    u32 STOP_BITPOS_VALUE;
    u16 HDURATION;
    u16 LDURATION;
    u16 GDURATION;
    dma_addr_t BUF0_BASE_ADDR;
    u32 BUF0_SIZE;
    u16 WAVE_NUM;
} PWM_MODE_MEMORY_REGS;
```



1：pwm_no: selecting the required PWM number, there are PWM1 ~ PWM4 for user to select.

2：mode：choose PWM_MODE_FIFO；

3：clk_src ： PWM_CLK_NEW_MODE_BLOCK, PWM_CLK_NEW_MODE_BLOCK_DIV_BY_1625;

4 ： clk_div ： CLK_DIV1, CLK_DIV2, CLK_DIV4, CLK_DIV8, CLK_DIV16, CLK_DIV32, CLK_DIV64, CLK_DIV128；

5：IDLE_VALUE：PWM output when in idle time, it has 2 values: FALSE or TRUE. FALSE: The level of GPIO is low after the end of waveform transmission. TURE: The level of GPIO is high after the end of waveform transmission;

6：GUARD_VALID: FALSE or TRUE. FALSE: there is no guarding interval between 2 complete waveforms. TURE: it has guarding interval between 2 complete waveforms, guarding interval decided by the value of GDURATION;

7：GDURATION：when GUARD_VALID is set to 1，GDURATION is valid. It expresses the interval between the two completed waveforms. The time calculation formula:

T = 1/(clk_src/clk_div) * (GDURATION + 1)

Ex:clk_src choose 66 MHz，clk_div choose 1，GDURATION = 1

T = 1/(66/1) * (1 + 1) = 30 nsec

8：HDURATION and LDURATION: the duration of the high/low ,calculation formula:

T = 1/(clk_src/clk_div) * (HDURATION + 1)

Ex:clk_src choose 66MHz, clk_div choose 1, HDURATION = 1

T = 1/(66/1) * (1 + 1) = 30 nsec

T = 1/(clk_src/clk_div) * (LDURATION + 1)

Ex:clk_src choose 66MHz,clk_div choose 1, LDURATION = 2

T = 1/(66/1) * (2 + 1) = 45.45 nsec

NOTE：The value of HDURATION and LDURATION could not be -1 or less than 0. And they could be different value.

9：STOP_BITPOS_VALUE：The stop position of data in the last word. Just like the last word of the picture above.

10：BUF0_BASE_ADDR : the specific physical memory address，it should be below 4G (0xffff_ffff).

NOTE: using Flag “GFP_DMA” for allocating low-end memory (< 0xffff_ffff)。

Example,

Using dma_alloc_coherent.

wave_vir = dma_alloc_coherent(dev, buf_size, &wave_phy, GFP_KERNEL | GFP_DMA);

wave_phy is the physical memory which below 4g.

11：BUF0_SIZE : the size of memory in word (32 bit)

12：wave_num : the wave number. If 0, only disable pwm will stop sending wave.

After the completion of all parameters’ setting, user can call pwm_set_spec_config() to write the value of parameter into corresponding register.

9 Troubleshooting

9.1 Why the PWM can not work normally?

When PWM can't sent correct waveform, please:

1. Check the clocks are enable
2. Check the gpio setting is correct. Make sure the gpio is switched to PWM mode.

10 Frequently Asked Questions

10.1 N/A