MT793X IoT SDK for PWM User Guide

Version: 2.0

Release date: 2021-11-10

Version History

Version	Date	Author	Description
0.1	2021-07-01	Guodong.Liu	Initial draft
1.0	2021-07-30	Guodong.Liu	Official release
2.0	2021-11-10	Guodong.Liu	Update description for PWM not work in sleep mode

Table of Contents

List of Figures

List of Tables

1 Overview

This document introduces the architecture of the Pulse-Width Modulation (PWM) driver. The PWM driver is a software module that is responsible for PWM output for desired behavior.

The PWM driver prepares API for PWM output desired behavior. It should be noted that PWM HW cannot work in sleep mode.

Figure 1. PWM Driver ArchitectureFigure 1 is the architecture of the PWM driver.

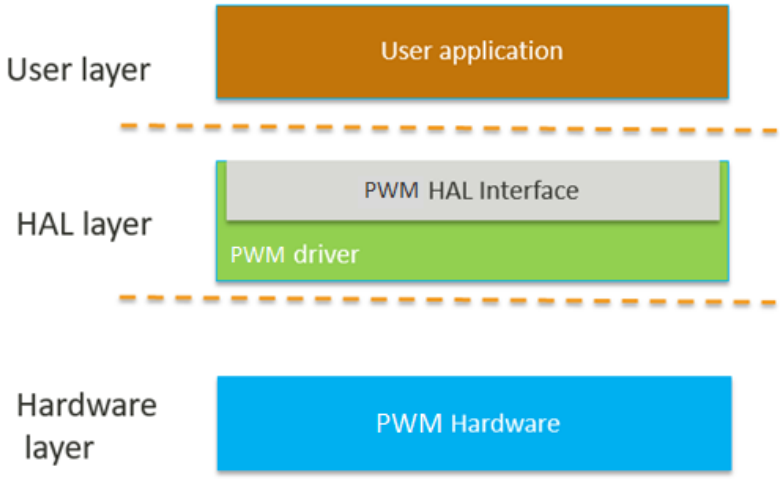

Figure 1. PWM Driver Architecture

2 PWM Driver Introduction

2.1 PWM Driver API Reference

2.1.1 API List

Table 2-1. API List

API	Description
hal_pwm_init (hal_pwm_source_clock_t source_clock)	This function initializes the PWM hardware source clock.
hal_pwm_deinit (void)	This function deinitializes the PWM hardware.
hal_pwm_set_frequency (hal_pwm_channel_t pwm_channel, uint32_t frequency, uint32_t *total_count)	This function sets the PWM frequency and retrieves total count of the PWM hardware at the specified frequency.
hal_pwm_set_duty_cycle (hal_pwm_channel_t pwm_channel, uint32_t duty_cycle)	This function sets the PWM duty cycle.
hal_pwm_start (hal_pwm_channel_t pwm_channel)	This function starts the PWM execution.
hal_pwm_stop (hal_pwm_channel_t pwm_channel)	This function stops the PWM execution.
hal_pwm_get_duty_cycle (hal_pwm_channel_t pwm_channel, uint32_t *duty_cycle)	This function gets the current duty cycle of the PWM module.
hal_pwm_get_frequency (hal_pwm_channel_t pwm_channel, uint32_t *frequency)	This function gets current frequency of the PWM module; the unit of frequency is Hz.
hal_pwm_get_running_status (hal_pwm_channel_t pwm_channel, hal_pwm_running_status_t *running_status)	This function gets the current status of the PWM module.

2.1.2 API Introduction

2.1.3 API Control Flow

2.1.3.1 Initialization Control Flow

Call the hal_pwm_init() function to initialize the PWM source clock, and then call the hal_pwm_set_frequency() function to set the PWM frequency and get the total counter value of the hardware. The total counter value is the PWM counter internal value at specified frequency. The duty cycle is calculated as the product of the application's duty ratio and total counter value. The duty ratio is the ratio of duty counter value over the total counter value.

Call the hal_pwm_set_duty_cycle() function to set the PWM duty cycle. Call the hal_pwm_start() function to trigger PWM execution.

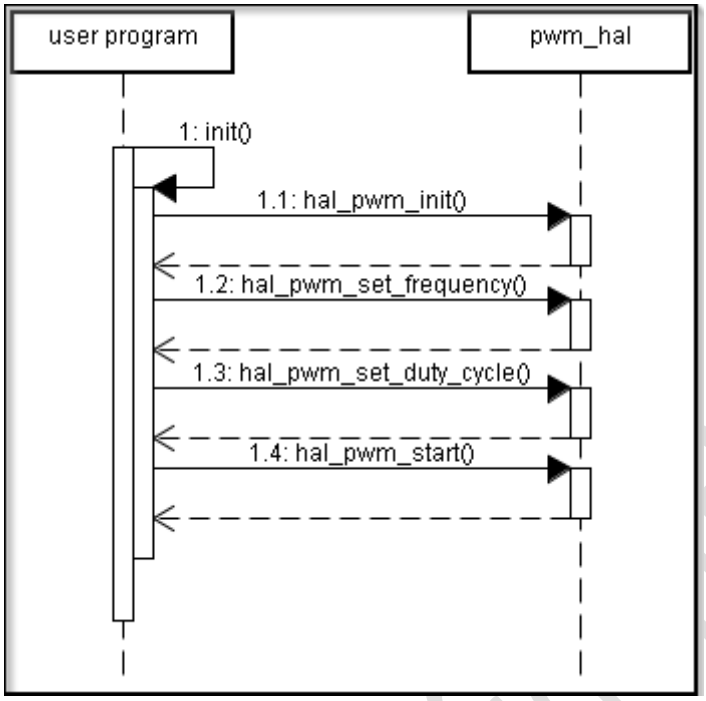

Figure 2. PWM Driver Initialization Flow

2.1.3.2 Deinit Control Flow

Call the hal_pwm_stop() function to stop the PWM module, and call the hal_pwm_deinit() function to deinitialize the PWM hardware.

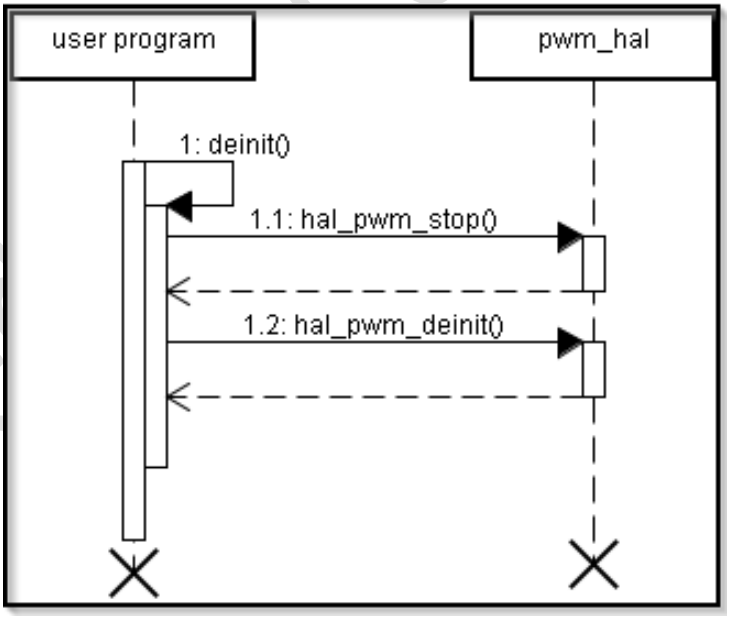

Figure 3. PWM Driver Deinitialization Flow

2.1.3.3 API Usage Notes

When using the API, you need to focus on the frequency and duty cycle of function parameters. The examples are as follows:

Assuming the crystal (XTAL) oscillator runs at 26MHz, the application needs a waveform at 2kHz and 50% duty ratio. The limits of frequency and duty cycle are as shown below.

1. The maximum supported frequency is half of the source clock frequency. When the 26MHz XTAL oscillator is used as the clock source, the maximum supported frequency is 13MHz.
2. PWM duty_cycle, resolution is 0.1, for example duty_ratio is 50%, duty_cycle = (50/0.1) = 500.
3. total_count = source_clock/frequency, and the value of total_count should be greater than 2. This value is optional and can be used as a reference.

2.2 PWM Driver Usage

2.2.1 Sample Code

```
uint32_t duty_ratio = 50; //The range from 0 to 100.
hal_pwm_source_clock_t source_clock = HAL_PWM_CLOCK_26MHZ;
uint32_t  frequency = 2000;
hal_gpio_init(HAL_GPIO_50);
//function index = MT7933_PIN_50_FUNC_PWM_5; for more information about pinmux please reference hal_
pinmux_define.h
hal_pinmux_set_function(HAL_GPIO_50, function_index);//Set the GPIO pinmux.
//Initialize the PWM source clock.
if(HAL_PWM_STATUS_OK != hal_pwm_init(source_clock)) {
    //error handle
}
//Sets frequency and gets total count of the PWM hardware at the specific frequency.
if(HAL_PWM_STATUS_OK != hal_pwm_set_frequency(HAL_PWM_5, frequency, &total_count)) {
    //error handle
}
duty_cycle = duty_ratio * 10; //duty_cycle is calculated as a product of application's duty_ratio.
//Enable PWM to start the timer.
if(HAL_PWM_STATUS_OK != hal_pwm_set_duty_cycle(HAL_PWM_5, duty_cycle)) {
    //error handle
}
hal_pwm_start(HAL_PWM_5);  //Trigger PWM execution.
//...
hal_pwm_get_frequency(HAL_PWM_5, &frequency);
hal_pwm_get_duty_cycle(HAL_PWM_5, &duty_cycle);
//...
hal_pwm_stop(HAL_PWM_5); //Stop the PWM.
hal_pwm_deinit();   //Deinitialize the PWM.
```

Exhibit 1 Terms and Conditions

Your access to and use of this document and the information contained herein (collectively this "Document") is subject to your (including the corporation or other legal entity you represent, collectively "You") acceptance of the terms and conditions set forth below ("T&C"). By using, accessing or downloading this Document, You are accepting the T&C and agree to be bound by the T&C. If You don't agree to the T&C, You may not use this Document and shall immediately destroy any copy thereof.

This Document contains information that is confidential and proprietary to MediaTek Inc. and/or its affiliates (collectively "MediaTek") or its licensors and is provided solely for Your internal use with MediaTek's chipset(s) described in this Document and shall not be used for any other purposes (including but not limited to identifying or providing evidence to support any potential patent infringement claim against MediaTek or any of MediaTek's suppliers and/or direct or indirect customers). Unauthorized use or disclosure of the information contained herein is prohibited. You agree to indemnify MediaTek for any loss or damages suffered by MediaTek for Your unauthorized use or disclosure of this Document, in whole or in part.

MediaTek and its licensors retain titles and all ownership rights in and to this Document and no license (express or implied, by estoppels or otherwise) to any intellectual propriety rights is granted hereunder. This Document is subject to change without further notification. MediaTek does not assume any responsibility arising out of or in connection with any use of, or reliance on, this Document, and specifically disclaims any and all liability, including, without limitation, consequential or incidental damages.

THIS DOCUMENT AND ANY OTHER MATERIALS OR TECHNICAL SUPPORT PROVIDED BY MEDIATEK IN CONNECTION WITH THIS DOCUMENT, IF ANY, ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE. MEDIATEK SPECIFICALLY DISCLAIMS ALL WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR A PARTICULAR PURPOSE, COMPLETENESS OR ACCURACY AND ALL WARRANTIES ARISING OUT OF TRADE USAGE OR OUT OF A COURSE OF DEALING OR COURSE OF PERFORMANCE. MEDIATEK SHALL NOT BE RESPONSIBLE FOR ANY MEDIATEK DELIVERABLES MADE TO MEET YOUR SPECIFICATIONS OR TO CONFORM TO A PARTICULAR STANDARD OR OPEN FORUM.

Without limiting the generality of the foregoing, MediaTek makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does MediaTek assume any liability arising out of the application or use of any product, circuit or software. You agree that You are solely responsible for the designing, validating and testing Your product incorporating MediaTek's product and ensure such product meets applicable standards and any safety, security or other requirements.

The above T&C and all acts in connection with the T&C or this Document shall be governed, construed and interpreted in accordance with the laws of Taiwan, without giving effect to the principles of conflicts of law.