MT793X IoT SDK for GPT User Guide

Version: 1.0

Release date: 2021-08-02

Version History

Version	Date	Description
1.0	2021-08-02	Official release

Table of Contents

1 Overview

Terms and Acronyms

Terms	Details
GPT	General Purpose Timer (GPT) is used as an alarm clock for timing control.
NVIC	The Nested Vectored Interrupt Controller (NVIC) is the interrupt controller of the ARM Cortex-M processor. For more details, please refer to NVIC introduction in ARM Cortex-M4 Processor Technical Reference Manual.

GPT has a generic design to support various combinations of the timer functionality.

Support oneshot, repeat and free run modes.

A timer can be configured in one of the modes, oneshot mode, repeat mode and free run mode. The timer mode is configured through the mode configuration.

- **Oneshot mode.** In this mode, the interrupt is triggered only once when the timer expires.
- **Repeat mode.** In this mode, the interrupt is triggered when the timer expires. At the same time, the timer is reloaded again with the same value and starts ticking till the next expiration. This pattern repeats until the timer is cancelled. This mode is useful for handling functions that are executed periodically.
- **Free run mode.** In this mode, the timer simply becomes a counter. There is no interrupt triggered in this mode. The counter keeps running once hal_gpt_get_free_run_count() is used, and never stops. Call hal_gpt_get_free_run_count() to get the stamp of counter ticks. This mode is useful when a delay or counting operations are performed.

Support delay function.

The GPT driver supports delays in milliseconds hal_gpt_delay_ms() and microseconds hal_gpt_delay_us(). It utilizes the free run mode of the timer and the polling mechanism to determine if the timeout for the timer is reached. The delay function is a hardware delay. It is for some programs to delay for a specified period without the need for the context switch.

Support callback function registration.

A callback function must be registered using hal_gpt_register_callback() function, and then hal_gpt_start_timer_ms() is called to set the oneshot mode or the repeat mode to start the timer.
The callback function is called after the GPT triggers an interrupt in the GPT ISR routine.

1.1 Using Oneshot or Repeat Mode

1. Call hal_gpt_get_running_status() function to manually check the GPT status before using the GPT driver in oneshot or repeat mode. If the status is HAL_GPT_RUNNING, you should wait till the status is HAL_GPT_STOPPED.

2. Call hal_gpt_init() and then hal_gpt_start_timer_ms() functions to set the GPT mode and expiration time.

The timer then starts ticking. Once the pre-set time expires in the oneshot mode, the GPT triggers an interrupt, stops the timer, and calls the callback function you give. In the repeat mode, when the set time expires, the timer is reloaded with the set time and the callback function you give is invoked. To ensure precise timing, do not overload the callback function. Let the callback return as fast as possible.

Step1: Call hal_gpt_get_running_status() to get the current running status.
Step2: Call hal_gpt_init() to configure the basic environment.
Step3: Call hal_gpt_register_callback() to register a callback function.
Step4: Call hal_gpt_start_timer_ms() to set the timer mode and the expiration time and start the timer.
Step5: Call hal_gpt_stop_timer() to stop the timer.
Step6: Call hal_gpt_deinit() to de-initialize the GPT module if it is no longer in use.

Sample Code: doc_internal/ MTK 7933 API Reference Manual/group___g_p_t.html

1.2 Using Free Run Mode

1. Call hal_gpt_get_free_run_count() to get the first stamp of timer ticks.
2. Call hal_gpt_get_free_run_count() again to get the second stamp of timer ticks.

If the HAL_GPT_CLOCK_SOURCE_32K parameter is in use, the unit of tick is 1/32768 seconds. The difference between the first and the second stamps is the counting ticks. In the free run mode, the GPT runs continuously and never stops. It does not provide interrupt feature. The GPT driver also uses this function implementation similar to the delay function hal_gpt_delay_ms().

Note: when the GPT works in the free run mode, you do not need to call hal_gpt_init() and hal_gpt_deinit() to initialize and deinitialize the GPT.

Step1: Call hal_gpt_get_free_run_count() to get the first tick count.
Step2: Call hal_gpt_get_free_run_count() to get the second tick count.
Step3: Call hal_gpt_get_duration_count() to get the duration time.

Sample Code: doc_internal/ MTK 7933 API Reference Manual/group___g_p_t.html

1.3 Using GPT the Delay Function

Call the hal_gpt_delay_ms() function to set the delay time. When in a delay, the GPT driver uses free run mode and polls register value until it expires.

Sample Code: doc_internal/ MTK 7933 API Reference Manual/group___g_p_t.html

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.