HAL_V2 TIMER

开发指南

版本号: 1.0

发布日期: 2025.7.15

版本历史

版本号	日期	制/修订人	内容描述
1.0	2025.7.15	AWA2351	初始版本

目　录

1 前言

1.1 文档简介

介绍 HAL_V2 中 TIMER 驱动的接口及使用方法，为 TIMER 使用者提供参考。

1.2 目标读者

TIMER 驱动的开发/维护人员。

1.3 适用范围

Table: 适用产品列表

表 1-1: 适用产品列表

产品名称	内核版本	驱动文件
T536	FreeRTOS	hal_timer.c
T153	FreeRTOS	hal_timer.c

1.4 文档约定

1.4.1 标志说明

⚠ 注意

- **提醒操作中应注意的事项。不当的操作可能会损坏器件，影响可靠性、降低性能等。**

说明

助准确理解文中指令、正确实施操作而提供的补充或强调信息。

技巧

一些容易忽视的小功能、技巧。了解这些功能或技巧能帮助解决特定问题或者节省操作时间。

1.5 相关术语介绍

1.5.1 硬件术语

表 1-2: 硬件术语

术语	解释说明
TIMER	定时器

1.5.2 软件术语

表 1-3: 软件术语

术语	解释说明
HAL	Hardware Abstraction Layer，硬件抽象层

2 模块介绍

2.1 模块功能介绍

Timer 模块可以用于实现计时、计数功能。其具体规格如下所示：

- 计数时钟可配置:LOSC
- 可配置 8 种预分频系统
- 可编程 32bit 减法定时器
- 两种工作模式：循环模式和单次计数模式
- 当计数值减到 0 时可以产生中断

2.2 模块配置介绍

menuconfig 配置项

```
DRIVERS_V2_TIMER //打开驱动
HAL_TEST_TIMER   //打开测试，可不开
```

2.3 模块源码结构

1. 驱动源码

```
hal_v2/hal/source/TIMER/
├──── hal_timer.c
├──── common_timer.h
├──── Kconfig
├──── Makefile
├──── platform
│   ├──── timer_sun8iw22.h
│   └──── timer_sun55iw6.h
└──── platform_timer.h
```

2. 驱动 APIs 声明头文件

```
hal_v2/include/hal/
└──── hal_sunxi_timer.h
```

3. 驱动 APIs 测试代码

```
hal_v2/hal/examples/TIMER/
├── timer_get_cntval
│   ├── timer_get_cntval.c
│   └── Makefile
├── timer_set_oneshot
│   ├── timer_set_oneshot.c
│   └── Makefile
└── timer_set_periodic
    ├── timer_set_periodic.c
    └── Makefile
```

3 模块接口说明

3.1 接口列表

表 3-1: 模块接口列表

API	解释说明
hal_timer_init	TIMER 模块初始化
hal_timer_deinit	TIMER 模块反初始化
hal_timer_start	TIMER 模块启动
hal_timer_stop	TIMER 模块停止
hal_timer_read_cntlow	TIMER 模块读取计数值低 32 位
hal_timer_read_cnthigh	TIMER 模块读取计数值高 32 位
hal_timer_read_intval_low	TIMER 模块读取溢出值低 32 位
hal_timer_read_intval_high	TIMER 模块读取溢出值高 32 位
hal_timer_get_counter	TIMER 获取计数值
hal_timer_set_counter	TIMER 设置计数值
hal_timer_set_oneshot	TIMER 设置为单次定时模式
hal_timer_set_periodic	TIMER 设置为周期定时模式

接口使用说明

3.2.1 hal_timer_init

- 作用：TIMER 模块初始化
- 参数：
  - timer_id 代表 TIMER 端口号

- timer 代表 TIMER 配置参数结构体

值：无

3.2.2 hal_timer_deinit

- 作用：TIMER 模块解初始化
- 参数：
  - timer_id 代表 TIMER 端口号
- 返回值：无

3.2.3 hal_timer_start

- 作用：启动 TIMER 模块
- 参数：
  - timer_id 代表 TIMER 端口号
- 返回值：无

4 hal_timer_stop

- 作用：停止 TIMER 模块
- 参数：
  - timer_id 代表 TIMER 端口号
- 返回值：无

3.2.5 hal_timer_read_cntlow

：TIMER 模块读取计数值低

- 参数：
  - timer_id 代表 TIMER 端口号
- 返回值：计数值

3.2.6 hal_timer_read_cnthigh

- 作用：TIMER 模块读取计数值高 32 位
- 参数：
  - timer_id 代表 TIMER 端口号
- 返回值：计数值

3.2.7 hal_timer_read_intval_low

- 作用：TIMER 模块读取溢出值低 32 位
  - timer_id 代表 TIMER 端口号
- 返回值：溢出值

3.2.8 hal_timer_read_intval_high

- 作用：TIMER 模块读取溢出值高 32 位
- 参数：
  - timer_id 代表 TIMER 端口号

3.2.9 hal_timer_get_counter

- 作用：TIMER 模块读取计数值
- 参数：
  - timer_id 代表 TIMER 端口号
- 返回值：计数值

3.2.10 hal_timer_set_counter

- 作用：TIMER 模块设置计数值
- 参数：
  - timer_id 代表 TIMER 端口号
  - val 代表计数值
- 返回值：无

3.2.11 hal_timer_set_oneshot

- 作用：TIMER 模块设置为单次定时模式
- 参数：
  - timer_id 代表 TIMER 端口号
  - interval_lo 代表溢出值低 32 位
  - interval_hi 代表溢出值高 32 位
- 返回值：
  - 0 代表成功
  - -1 代表失败

3.2.12 hal_timer_set_periodic

- 作用：TIMER 模块设置为周期定时模式
- 参数：
  - timer_id 代表 TIMER 端口号
  - interval_lo 代表溢出值低 32 位
  - interval_hi 代表溢出值高 32 位
- 返回值：
  - 0 代表成功
  - -1 代表失败

4 模块使用范例

可参考驱动 APIs 测试代码（hal_v2/hal/examples/TIMER/timer_set_periodic/timer_set_periodic.c），具体实现如下：

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

de <hal_clk.h>

#include <hal_reset.h>
#include <hal_interrupt.h>
#include <hal_sunxi_timer.h>

#if defined(CONFIG_KERNEL_FREERTOS)
#include <hal_log.h>
#include <hal_cmd.h>
#include <hal_thread.h>
#include <hal_timer.h>
#elif defined(CONFIG_KERNEL_BAREMETAL)
#include "shell.h"
#endif

#if defined(CONFIG_ARCH_SUN55IW6)
#define TEST_RST_BUS_TIMER RST_BUS_TIMER
e TEST_CLK_TIMER_BUS CLK_BUS_TIMER
#elif defined(CONFIG_ARCH_SUN8IW22)
#define TEST_RST_BUS_TIMER RST_BUS_TIMER0
#define TEST_CLK_TIMER_BUS CLK_BUS_TIMER0
#endif
#define TEST_CLK_TIMER CLK_TIMER3
#define TIMER_TEST_IRQ SUNXI_IRQ_TIMER3
#define TIMER_TEST_BASE HAL_TIMER3

static uint32_t test_timerx = TIMER_TEST_BASE;


static int timer_clk_init(void)
{
    int ret;
    hal_rst_t rst;
    rst_number_t rst_bus_num;
    hal_clk_t clk_bus;
    hal_clk_t clk_timer;
    clk_number_t clk_bus_num;
    clk_number_t clk_timer_num;

    /*TIMER0 BUS RST */
    rst_bus_num = MAKE_RSTn(AW_SYS_CCU, TEST_RST_BUS_TIMER);

    /*TIMER0 AHB CLK ENABLE*/
    clk_bus_num = MAKE_CLKn(AW_SYS_CCU, TEST_CLK_TIMER_BUS);
```

```
    /*TIMER0_3 CLK ENABLE*/
    clk_timer_num = MAKE_CLKn(AW_SYS_CCU, TEST_CLK_TIMER);

    printf("reset info,  rst_num: %u, rc_id: %u, rst_id: %u\n", clk_bus_num, RC_ID(rst_bus_num), RST_ID(rst_bus_num
      ));
    printf("clk info,  clk_num: %u, cc_id: %u, clk_id: %u\n", clk_bus_num, CC_ID(clk_bus_num), CLK_ID(clk_bus_num)
      );
    printf("clk info,  clk_num: %u, cc_id: %u, clk_id: %u\n", clk_timer_num, CC_ID(clk_timer_num), CLK_ID(
      clk_timer_num));

    ret = hal_rst_get_by_rst_num(rst_bus_num, &rst);
    if (ret) {
        printf("get reset handle failed,  rst_num: %u, ret: %d\n", rst_bus_num, ret);
        return -1;
    }

    ret = hal_n_clk_get_by_clk_num(clk_bus_num, &clk_bus);
(ret) {
        printf("get clk(%u) failed,  ret:  %d\n", clk_bus_num, ret);
        return -1;
    }

    ret = hal_n_clk_get_by_clk_num(clk_timer_num, &clk_timer);
    if (ret) {
        printf("get clk(%u) failed,  ret:  %d\n", clk_timer_num, ret);
        return -1;
    }

    ret = hal_rst_deassert(rst);
    if (ret) {
        printf("deassert reset failed，rst_num: %u, ret: %d\n", rst_bus_num, ret);
        ret = -2;
        goto exit_with_put_rst;


    ret = hal_n_clk_enable(clk_bus);
    if (ret) {
        printf("enable clk(%u) failed,  ret:  %d\n", clk_bus_num, ret);
        ret = -2;
        goto exit_with_put_clk_bus;
    }

    ret = hal_n_clk_enable(clk_timer);
    if (ret) {
        printf("enable clk(%u) failed,  ret:  %d\n", clk_timer_num, ret);
        ret = -2;
        goto exit_with_put_clk;
    }

    printf("deassert reset success, rst_bus_num);
    printf("enable clk_bus(%u) success\n", clk_bus_num);
    printf("enable clk(%u) success\n", clk_timer_num);

    ret = 0;

exit_with_put_clk:
    hal_n_clk_put(clk_timer);

exit_with_put_clk_bus:
    hal_n_clk_put(clk_bus);
```

```
exit_with_put_rst:
    hal_rst_put(rst);
    return ret;
}

static hal_irqreturn_t hal_timer_irq_handler(void *data)
{
    printf("timer3 1s irq periodic\n");

    unsigned int val = 0;
    u32 test_timer = *(u32 *)data;
    val = PENDING_BIT(test_timer);
    val &= readl(TIMER_IRQ_ST_REG(test_timer));
    if (val != PENDING_BIT(test_timer)) {
        return -1;
    }

    writel(PENDING_BIT(test_timer), TIMER_IRQ_ST_REG(test_timer));
    return 0;
}

void hal_v2_timer_set_periodic(void)
{
    int ret = 0;

    ret = timer_clk_init();
    if (ret) {
        printf("timer_clk_init failed, ret: %d\n", ret);
        return;
    }

    /* set time 1s */
    timer_initstruct.interval_lo = 24000000;/* defalut clk source: 24M, so 1s = 24000000 */
    timer_initstruct.interval_hi = 0;
    timer_initstruct.mode = CONTINUE_MODE;

    hal_timer_init(test_timerx, &timer_initstruct);

    hal_irq_init_t intc_initstruct ;
    intc_initstruct .irqn = TIMER_TEST_IRQ;
    intc_initstruct .preemptionpriority = 0;
    intc_initstruct .subpriority = 0;
    intc_initstruct .cmd = HAL_INIT_CMD_ENABLE;
    intc_initstruct .isr_info.func = hal_timer_irq_handler;
    intc_initstruct .isr_info.parg = &test_timerx;
    hal_v2_intc_irq_config(&intc_initstruct);

    hal_timer_start(test_timerx);

}

#if defined(CONFIG_KERNEL_FREERTOS)
FINSH_FUNCTION_EXPORT_CMD(hal_v2_timer_set_periodic, timer_set_periodic_test, timer hal_v2 APIs tests)
#elif defined(CONFIG_KERNEL_BAREMETAL)
SHELL_EXPORT_CMD(
SHELL_CMD_PERMISSION(0)|SHELL_CMD_TYPE(SHELL_TYPE_CMD_FUNC)|SHELL_CMD_DISABLE_RETURN,
timer_set_periodic_test, hal_v2_timer_set_periodic, test for timer set periodic);
#endif
```

5 FAQ

无

声明

版权所有 ©2025 珠海全志科技股份有限公司。保留一切权利。

本文档及内容受著作权法保护，其著作权由珠海全志科技股份有限公司（“全志”）拥有并保留一切权利。

本文档是全志的原创作品和版权财产，未经全志书面许可，任何单位和个人不得擅自摘抄、复制、修改、发表或传播本文档内容的部分或全部，且不得以任何形式传播。

商标声明

、 、 、 （不完全列举）均为珠海全志科技股份有限公司的商标或者注册商标。在本文档描述的产品中出现的其它商标，产品名称，和服务名称，均由其各自所有人拥有。

免责声明

您购买的产品、服务或特性应受您与珠海全志科技股份有限公司（“全志”）之间签署的商业合同和条款的约束。本文档中描述的全部或部分产品、服务或特性可能不在您所购买或使用的范围内。使用前请认真阅读合同条款和相关说明，并严格遵循本文档的使用说明。您将自行承担任何不当使用行为（包括但不限于如超压，超频，超温使用）造成的不利后果，全志概不负责。

本文档作为使用指导仅供参考。由于产品版本升级或其他原因，本文档内容有可能修改，如有变不另行通知。全志尽全力在本文档中提供准确的信息，但并不确保内容完全没有错误，因使用本文档而发生损害（包括但不限于间接的、偶然的、特殊的损失）或发生侵犯第三方权利事件，全志概不负责。本文档中的所有陈述、信息和建议并不构成任何明示或暗示的保证或承诺。

本文档未以明示或暗示或其他方式授予全志的任何专利或知识产权。在您实施方案或使用产品的过程中，可能需要获得第三方的权利许可。请您自行向第三方权利人获取相关的许可。全志不承担也不代为支付任何关于获取第三方许可的许可费或版税（专利税）。全志不对您所使用的第三方许可技术做出任何保证、赔偿或承担其他义务。