HAL_V2 PWMCS

开发指南

版本号: 1.0

发布日期: 2025.07.28

版本历史

版本号	日期	制/修订人	内容描述
1.0	2025.07.28	AWA2215	初始版本

目　录

9

1 前言

1.1 文档简介

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

1.2 目标读者

PWMCS 驱动的开发/维护人员。

1.3 适用范围

表 1-1: 适用产品列表

产品名称	内核版本	驱动文件
T153	FreeRTOS	hal_pwmcs.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

为准确理解文中指令、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。

相关术语介绍

术语	解释说明
Sunxi	指 Allwinner 的一系列 SOC 硬件平台
PWMCS	Pulse Width Modulation Control System 即脉宽调制控制系统

2 模块介绍

2.1 模块功能介绍

脉宽调制控制系统（PWMCS）模块支持单端、AB 相、脉冲 + 方向、CW/CCW 等波形输出，支持输入波形捕获功能用于检测输入脉冲信号。系统域共有 2 套 PWMCS，最大支持 16 个独立输出通6 个独立捕获通道。

说明

halv2 pwmcs 驱动仅支持单端波形输出。

2.2 模块配置介绍

menuconfig 配置项

```
Drivers V2 Config  --->
  PWMCS Driver  --->
    [*] enable pwmcs driver
enable pwmcs hal APIs test command
```

2.3 模块源码结构

1. 驱动源码

```
hal_v2/hal/source/pwmcs/
├──── common_pwmcs.h
├──── hal_pwmcs.c
├──── Kconfig
├──── Makefile
├──── platform
│    └──── pwmcs_sun8iw22.h
└──── platform_pwmcs.h
```

2. 驱动 APIs 声明头文件

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

3. 驱动 APIs 测试代码

```
hal_v2/hal/examples/pwmcs/
├──── Makefile
└──── pwmcs_cycle_mode
  ├──── Makefile
  └──── pwmcs_cycle_mode.c
```

3 模块接口说明

3.1 hal_pwmcs_init

- 原 型：pwm_status_t hal_pwmcs_init(hal_pwmcs_port_t port, uint32_t channel, pwmcs_config_t *pwm_config);

能：pwmcs 初始化

- 参数：
  - port：pwm 端口号
  - channel：通道号
  - pwm_config：初始化结构体指针
- 返回值：
  - 0：成功
  - -1：失败

hal_pwmcs_channel_enable

- 原型：int hal_pwmcs_channel_enable(hal_pwmcs_port_t port, uint32_t channel);
- 功能：pwm 通道使能
- 参数：
  - port：pwm 端口号
  - channel：通道号
- 返回值：
  - 0：成功

- -1：失败

3.3 hal_pwmcs_channel_disable

- 原型：int hal_pwmcs_channel_disable(hal_pwmcs_port_t port, uint32_t channel);
- 功能：关闭 pwm 通道
- 参数：

  - port：pwm 端口号
  - channel：通道号

- 返回值：
  - 0：成功
  - -1：失败

4 模块使用范例

可参考驱动 APIs 测试代码（hal_v2/hal/examples/pwmcs/）：

```
#ifdef CONFIG_KERNEL_BAREMETAL
#include <include.h>
#else
#include <sunxi_hal_common.h>
#endif

de<hal_gpio.h>

#include <hal_reset.h>
#include <hal_clk.h>
#include <hal_pwmcs.h>

static int pwm_clk_init(void)
{
    int ret;
    hal_rst_t rst;
    rst_number_t rst_num;
    hal_clk_t clk;
    clk_number_t clk_num;

    rst_num = MAKE_RSTn(AW_SYS_CCU, RST_BUS_PWMCS0);
    clk_num = MAKE_CLKn(AW_SYS_CCU, CLK_PWM0);

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

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

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

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

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

```
    printf("deassert reset success, rst_num: %u\n", rst_num);
f("enable clk(%u) success\n", clk_num);

    ret = 0;

exit_with_put_clk:
    hal_n_clk_put(clk);

exit_with_put_rst:
    hal_rst_put(rst);
    return ret;
}

void pwmcs_cycle_mode(void)
{
    /* pwm clk init */
_clk_init();


    /* pwmcs0_0 PB8 init */
    gpio_init_t gpio_initstruct;

    gpio_initstruct.port = SUNXI_GPIO_B;
    gpio_initstruct.port_num = PIN_8;
    gpio_initstruct.mul_sel = GPB_PWM0;
    hal_gpio_init(&gpio_initstruct);

    pwmcs_config_t pwmcs_initstruct;
    pwmcs_initstruct.output_mode = PWM_MODE_SINGLE;
    pwmcs_initstruct.period_ns = 25000;
    pwmcs_initstruct.duty_ns = 7125;
    pwmcs_initstruct.pulse_num = 0;
    pwmcs_initstruct.polarity = PWM_POLARITY_NORMAL;
pwmcs_init(HAL_CPUX_PWMCS0, 0, &pwmcs_initstruct);

    /* enable pwm2_8 and ouput waveform */
    hal_pwmcs_channel_enable(HAL_CPUX_PWMCS0, PWM_CHANNEL_0);

    /* output 5s */
    udelay(5 * 1000 * 1000);

    /* disable pwm0_0 */
    hal_pwmcs_channel_disable(HAL_CPUX_PWMCS0, PWM_CHANNEL_0);
}
```

5 FAQ

无

声明

商标声明

免责声明