HAL_V2 GPADC

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 GPADC 驱动的接口及使用方法，为 GPADC 使用者提供参考。

1.2 目标读者

GPADC 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_gpadc.c
T153	FreeRTOS	hal_gpadc.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
GPADC	高精度模数转换

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

GPADC 是模数转换模块，模拟输入范围 0～1.8V，最高采样率 1MHZ，通道数、采集精度以 datasheet 为准，并且支持数据比较，自校验功能，同时工作于可配置的四种工作模式：

Single mode：在指定的通道完成一次转换并将数据放在响应数据寄存器中；

（2）Single-cycle mode：在指定的通道完成一个周期转换并将数据放在响应数据寄存器中；

（3）Continuous mode：在指定的通道持续转换并将数据放在响应数据寄存器中；

（4）Burst mode：边采样边转换并将数据放入 32 字节的 FIFO，支持中断控制。

2.2 模块配置介绍

config 配置项

```
DRIVERS_V2_GPADC  //打开驱动
HAL_TEST_GPADC    //打开测试，可不开
```

2.3 模块源码结构

1. 驱动源码

common_gpadc.h

```
hal_v2/hal/source/GPADC/
├── hal_gpadc.c

├── Kconfig
├── Makefile
├── platform
│   ├── gpadc_sun8iw22.h
│   └── gpadc_sun55iw6.h
└── platform_gpadc.h
```

2. 驱动 APIs 声明头文件

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

3. 驱动 APIs 测试代码

```
hal_v2/hal/examples/GPADC/
└──── gpadc_read_vol
   ├──── gpadc_read_vol.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_gpadc_init	GPADC 模块初始化
hal_gpadc_deinit	GPADC 模块反初始化
hal_gpadc_start	GPADC 模块启动
hal_gpadc_stop	GPADC 模块停止
hal_gpadc_set_sample_rate	GPADC 模块设置采样率
hal_gpadc_read_channel_data	GPADC 模块通道数据读取

3.2 接口使用说明

3.2.1 hal_gpadc_init

- 作用：GPADC 模块初始化
- 参数：
  - gpadc_config 代表 gpadc 配置参数结构体

值：

  - 0 代表成功
  - -1 代表失败

3.2.2 hal_gpadc_deinit

- 作用：GPADC 模块解初始化

- 参数：

口号

port 代表 GPADC

- 返回值：
  - 0 代表成功
  - -1 代表失败

3.2.3 hal_gpadc_start

- 作用：启动 GPADC 模块

：

  - port 代表 GPADC 端口号
- 返回值：无

3.2.4 hal_gpadc_stop

- 作用：停止 GPADC 模块
- 参数：
  - port 代表 GPADC 端口号
- 返回值：无

3.2.5 hal_gpadc_set_sample_rate

- 作用：设计 GPADC 模块采样频率
- 参数：
  - port 代表 GPADC 端口号
  - freq 代表采样频率

值：无

3.2.6 hal_gpadc_read_channel_data

- 作用：读 GPADC 模块某个通道采集的电压值
- 参数：
  - port 代表 GPADC 端口号

— channal 代表通道号

值：

— 电压值

4 模块使用范例

可参考驱动 APIs 测试代码（hal_v2/hal/examples/gpadc/gpadc_read_vol/gpadc_read_vol.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_gpadc.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_GPADC     RST_BUS_GPADC1
e TEST_CLK_GPADC_AHB_BUS CLK_BUS_GPADC1
#define TEST_CLK_GPADC_BUS     CLK_GPADC1
#define GPADC_TEST_BASE      GPADC1
#define GPADC_TEST_CHANNEL     GP_CH_7
#elif  defined(CONFIG_ARCH_SUN8IW22)
#define TEST_RST_BUS_GPADC     RST_BUS_GPADC0
#define TEST_CLK_GPADC_AHB_BUS  CLK_BUS_GPADC0
#define TEST_CLK_GPADC_BUS     CLK_GPADC0
#define GPADC_TEST_BASE      GPADC0
#define GPADC_TEST_CHANNEL     GP_CH_0
#endif

static  int  gpadc_clk_init(void)
{
    int  ret;
    hal_rst_t  rst;
    rst_number_t rst_bus_num;
    hal_clk_t clk_bus;
    hal_clk_t clk_gpadc;
    clk_number_t clk_bus_num;
    clk_number_t clk_gpadc_num;

    /*GPADC1 BUS RST */
    rst_bus_num = MAKE_RSTn(AW_SYS_CCU, TEST_RST_BUS_GPADC);

    /*GPADC1 AHB CLK ENABLE*/
    clk_bus_num = MAKE_CLKn(AW_SYS_CCU, TEST_CLK_GPADC_AHB_BUS);
```

```
    /*GPADC1 CLK ENABLE*/
    clk_gpadc_num = MAKE_CLKn(AW_SYS_CCU, TEST_CLK_GPADC_BUS);

    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_gpadc_num, CC_ID(clk_gpadc_num), CLK_ID(
      clk_gpadc_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_gpadc_num, &clk_gpadc);
    if (ret) {
        printf("get clk(%u) failed,  ret:  %d\n", clk_gpadc_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_gpadc);
    if (ret) {
        printf("enable clk(%u) failed,  ret:  %d\n", clk_gpadc_num, ret);
        ret = -2;
        goto exit_with_put_clk;
    }

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

    ret = 0;

exit_with_put_clk:
    hal_n_clk_put(clk_gpadc);

exit_with_put_clk_bus:
    hal_n_clk_put(clk_bus);
```

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

void hal_v2_gpadc_read_vol(void)
{
    int ret = 0;
    uint32_t vol_data;
    int count = 10;
    int delay = 1 * 1000 * 1000;
    ret = gpadc_clk_init();
    if (ret) {
        printf("gpadc_clk_init failed, ret: %d\n", ret);
        return;
    }

    gpadc_init_t gpadc_initstruct;
    gpadc_initstruct.port = GPADC_TEST_BASE;
    gpadc_initstruct.channel_used = GPADC_TEST_CHANNEL;
    gpadc_initstruct.mode = GP_CONTINUOUS_MODE;
    gpadc_initstruct.sample_rate = 1000000;

    hal_gpadc_init(&gpadc_initstruct);
    hal_gpadc_start(GPADC_TEST_BASE);

    while(count > 0) {
        vol_data = gpadc_read_channel_data(GPADC_TEST_BASE, GPADC_TEST_CHANNEL);
        printf("GPADC[%d]:channel %d vol data is %d mV\n", GPADC_TEST_BASE, GPADC_TEST_CHANNEL, vol_data);

        count--;
        udelay(delay);

}

#if defined(CONFIG_KERNEL_FREERTOS)
FINSH_FUNCTION_EXPORT_CMD(hal_v2_gpadc_read_vol, gpadc_read_vol_test, gpadc 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,
gpadc_read_vol_test, hal_v2_gpadc_read_vol, test for gpadc read);
#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。