HAL_V2 GPIO

开发指南

版本号: 1.0

发布日期: 2025.07.24

版本历史

版本号	日期	制/修订人	内容描述
1.0	2025.07.24	AWA2215	初始版本

目　录

1 前言 **1**
1.1 文档简介 . . . 1
1.2 目标读者 . . . 1
1.3 适用范围 . . . 1
1.4 文档约定 . . . 1
1.4.1 标志说明 . . . 1
1.5 相关术语介绍 . . . 2

2 模块介绍 **3**
2.1 模块功能介绍 . . . 3
2.2 模块配置介绍 . . . 3
2.3 模块源码结构 . . . 3

3 模块接口说明 **5**
3.1 数据结构 . . . 5
3.1.1 gpio_init_t . . . 5
3.1.2 pin_pull_e . . . 5
3.1.3 pin_multi_drive_e . . . 6
3.1.4 pin_int_trigger_mode_e . . . 6
3.1.5 pin_data_status_e . . . 6
3.2 接口使用说明 . . . 6
3.2.1 hal_gpio_init . . . 7
3.2.2 hal_gpio_deinit . . . 7
3.2.3 hal_gpio_set_drv . . . 8
3.2.4 hal_gpio_get_drv . . . 8
3.2.5 hal_gpio_set_pull . . . 8
3.2.6 hal_gpio_get_pull . . . 8
3.2.7 hal_gpio_set_data . . . 9
3.2.8 hal_gpio_get_data . . . 9
3.2.9 hal_gpio_clear_irq_status . . . 9

块使用范例 **10**

5 FAQ **11**

1 前言

1.1 文档简介

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

1.2 目标读者

GPIO 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
T536	FreeRTOS	hal_gpio.c
T153	FreeRTOS	hal_gpio.c

1.4 文档约定

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。

1.5 相关术语介绍

术语	解释说明
Sunxi	指 Allwinner 的一系列 SOC 硬件平台
GPIO	General Purpose Input/Output，通用输入输出

2 模块介绍

2.1 模块功能介绍

整个 GPIO 控制器由数字部分（GPIO 和外设接口）以及 IO 模拟部分（输出缓冲，双下拉，pad）组成。其中数字部分的输出可以通过 MUX 开关选择，模拟部分可以用来配置上下拉，驱动能力以
输出电压等等。具体的规格如下：

- 可以在软件上配置各个引脚的状态
- 每个引脚都可以触发中断
- 可以配置上拉/下拉/无上下拉三种状态
- 每个引脚都可以配置 4 种驱动能力
- 可以配置边缘中断触发
- 中断数量跟随平台变化

2.2 模块配置介绍

menuconfig 配置项

```
DRIVERS_V2_GPIO  //打开驱动
HAL_TEST_GPIO  //打开测试，可不开
```

2.3 模块源码结构

1. 驱动源码

/hal/source/gpio/

```
├── gpio.c
├── gpio.h
├── Kconfig
├── Makefile
├── platform
│   ├── gpio_sun55iw6.h
│   └── gpio_sun8iw22.h
└── platform_gpio.h
```

2. 驱动 APIs 声明头文件

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

3. 驱动 APIs 测试代码

```
hal_v2/hal/examples/pwm/
├── gpio_drv_test
│   ├── gpio_drv_test.c
│   └── Makefile
├── gpio_interrupt_test
│   ├── gpio_interrupt_test.c
│   └── Makefile
├── gpio_mux_test
│   ├── gpio_mux_test.c
│   └── Makefile
gpio_output_test
│   ├── gpio_output_test.c
│   └── Makefile
├── gpio_pull_test
│   ├── gpio_pull_test.c
│   └── Makefile
└── Makefile
```

3 模块接口说明

3.1 数据结构

由于 GPIO 需要配置每个引脚的引脚复用功能，中断类型，驱动能力，上下拉，输出/输入数据，输入/输出方向等等，所以对 GPIO 的这些配置都封装在一个 gpio_init_t 结构体里面，方便使用。

一些配置的定义，结构体，如果要了解更多的数据结构可以到

3.1.1 gpio_init_t

gpio 初始化用到的结构体，具体定义如下：

```
/* struct for gpio */
typedef struct
{
  unsigned char     port;
  unsigned int      port_num;
  signed char       mul_sel;
  signed char       pull;

  signed char       data;
  unsigned int      int_trriger_mode;
  unsigned char     reserved[2];
}gpio_init_t;
```

ed char drv_level;

- port：gpio 组，SUNXI_GPIO_A、SUNXI_GPIO_B 等
- port_num：gpio 引脚号
- mul_sel：gpio 复用功能
- drv_level：gpio 驱动能力
- data：复用为 gpio 输出时，gpio 电平
- int_trriger_mode：gpio 中断触发方式

erved：保留

3.1.2 pin_pull_e

该枚举定义了引脚的上下拉的值，具体定义如下：

```
typedef enum pin_pull {
  PIN_PULL_DISABLE  = 0x00,
  PIN_PULL_UP       = 0x01,
```

```
    PIN_PULL_DOWN    = 0x02,
    PIN_PULL_RESERVED = 0x03,
    PIN_PULL_DEFAULT  = 0xFF
} pin_pull_e;
```

3.1.3 pin_multi_drive_e

该枚举定义了引脚的驱动能力的值，具体定义如下：

```
typedef enum pin_multi_drive {
    PIN_MULTI_DRIVE_0    = 0x00,
    PIN_MULTI_DRIVE_1    = 0x01,
    PIN_MULTI_DRIVE_2    = 0x02,
    PIN_MULTI_DRIVE_3    = 0x03,
_MULTI_DRIVE_DEFAULT = 0xFF
} pin_multi_drive_e;
```

3.1.4 pin_int_trigger_mode_e

该枚举定义了引脚的中断模式，具体定义如下

```
typedef enum pin_int_trigger_mode {
    PIN_INT_POSITIVE_EDGE  = 0x0,
    PIN_INT_NEGATIVE_EDGE  = 0x1,
    PIN_INT_HIGN_LEVEL     = 0x2,
    PIN_INT_LOW_LEVEL      = 0x3,
_INT_DOUBLE_EDGE = 0x4
nt_trigger_mode_e;
```

3.1.5 pin_data_status_e

该枚举定义引脚的输入输出数据，具体定义如下：

```
typedef enum pin_data_status {
    PIN_DATA_LOW    = 0x00,
    PIN_DATA_HIGH   = 0x01,
    PIN_DATA_DEFAULT = 0XFF
} pin_data_status_e;
```

3.2 接口使用说明

表 3-1: 模块接口列表

API	解释说明
hal_gpio_init	gpio 初始化
hal_gpio_deinit	gpio 解初始化
hal_gpio_set_drv	设置 gpio 驱动能力
hal_gpio_get_drv	获取 gpio 驱动能力
hal_gpio_set_pull	设置 gpio 上下拉
hal_gpio_get_pull	获取 gpio 上下拉
hal_gpio_set_data	设置 gpio 输出电平
hal_gpio_get_data	获取 gpio 输入电平
hal_gpio_clear_irq_status	清除 gpio 中断状态

3.2.1 hal_gpio_init

- 原型：int hal_gpio_init(void *user_gpio_list);
- 作用：gpio 初始化
- 参数：
  - user_gpio_list：gpio_init_t 结构体指针
- 返回值：
  - 0 代表成功
  - -1 代表失败

3.2.2 hal_gpio_deinit

- 原型：void hal_gpio_deinit(void *user_gpio_list);
- 作用：gpio 解初始化
- 参数：
  - user_gpio_list：gpio_init_t 结构体指针
- 返回值：
  - void

3.2.3 hal_gpio_set_drv

- 原型：int hal_gpio_set_drv(u32 pin, u32 val);
- 作用：设置 gpio 驱动能力
- 参数：
  - pin：gpio 引脚
  - val：驱动能力
- 返回值：无

4 hal_gpio_get_drv

- 原型：int hal_gpio_get_drv(u32 pin);
- 作用：获取 gpio 驱动能力
- 参数：
  - pin：gpio 引脚
- 返回值：
  - int：驱动能力值

5 hal_gpio_set_pull

- 原型：int hal_gpio_set_pull(u32 pin, u32 val);
- 作用：设置 gpio 上下拉
- 参数：
  - pin：gpio 引脚
  - val：要设置的上下拉
- 返回值：
  - 0 代表成功

-1 代表失败

3.2.6 hal_gpio_get_pull

- 原型：int hal_gpio_get_pull(u32 pin);
- 作用：获取 gpio 上下拉
- 参数：
  - pin：gpio 引脚

- 返回值：

int：gpio 上下拉能力

3.2.7 hal_gpio_set_data

- 原型：void hal_gpio_set_data(u32 pin, u32 val);
- 作用：设置 gpio 输出电平
- 参数：
  - pin：gpio 引脚
  - val：要设置的 gpio 电平

值：

  - void

3.2.8 hal_gpio_get_data

- 原型：int hal_gpio_get_data(u32 pin);
- 作用：获取 gpio 电平
- 参数：

port：gpio 引脚

- 返回值：
  - int：gpio 电平

3.2.9 hal_gpio_clear_irq_status

- 原型：void hal_gpio_clear_irq_status(u32 port);
- 作用：清除 gpio 组的中断状态
- 参数：

port：gpio 组

- 返回值：
  - void

4 模块使用范例

可参考驱动 APIs 测试代码（hal_v2/hal/examples/gpio/），基本 gpio 复用操作方法如下：

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


#if defined(CONFIG_KERNEL_FREERTOS)
#include <console.h>
#elif defined(CONFIG_KERNEL_BAREMETAL)
#include "shell.h"
#endif

void gpio_mux_test(void)
{
    gpio_init_t gpio_initstruct;

    gpio_initstruct.port = SUNXI_GPIO_B;/* PB4 PB5: uart8_tx */
    gpio_initstruct.port_num = PIN_4 | PIN_5;
    gpio_initstruct.mul_sel = GPB_UART8;
    gpio_initstruct.pull = PIN_PULL_UP;
    gpio_initstruct.drv_level = PIN_MULTI_DRIVE_2;

    hal_gpio_init(&gpio_initstruct);

    printf("gpio mux test pass!\n");

    /* release gpio */
    hal_gpio_deinit(&gpio_initstruct);
}

#if defined(CONFIG_KERNEL_FREERTOS)
FINSH_FUNCTION_EXPORT_CMD(gpio_mux_test, gpio_mux_test, gpio 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,
gpio_mux_test, gpio_mux_test, test for gpio mux);
```

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。