HAL_V2 UART

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 文档简介

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

1.2 目标读者

UART 驱动的开发/维护人员。

1.3 适用范围

表 1-1: 适用产品列表

内核版本	驱动文件	
T536	FreeRTOS	hal_uart.c
T153	FreeRTOS	hal_uart.c

1.4 文档约定

1.4.1 标志说明

- 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

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
UART	Universal Asynchronous Receiver/Transmitter
BaudRate	波特率，UART 通信中每秒钟传输符号的数量

2 模块介绍

2.1 模块配置介绍

menuconfig 配置项

```
Drivers V2 Config  --->
iver--->
        [*] enable uart driver
        [*]   enable uart hal APIs test command
        [ ]   support poll APIs
        (0)   cli uart port number
        (115200) cli uart port's baudrate
```

2.2 模块源码结构

1. 驱动源码

```
/source/uart/
├── hal_uart.c
├── Kconfig
├── Makefile
├── platform
│   ├── uart-sun55iw6.h
│   └── uart-sun8iw22.h
└── platform-uart.h
```

2. 驱动 APIs 声明头文件

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

3. 驱动 APIs 测试代码

```
hal_v2/hal/examples/uart/
├── Makefile
├── uart_cpu_loopback_mode
│   ├── Makefile
│   └── uart_cpu_loopback_mode.c
├── uart_hardware_rs485_mode
│   ├── Makefile
│   └── uart_hardware_rs485_mode.c
```

```
├── uart_interrupt_mode
│   ├── Makefile
│   └── uart_interrupt_mode.c
├── uart_rx_dma
│   ├── Makefile
│   └── uart_rx_dma.c
└── uart_tx_dma
    ├── Makefile
    └── uart_tx_dma.c
```

3 模块接口说明

3.1 接口使用说明

3.2 hal_uart_init

- 函数原型:int32_t hal_uart_init(uart_init_t *uart_config_t);
- 作用：初始化 UART 驱动
- 参数：
  - uart_config_t: uart_init_t 结构体指针
- 返回：
  - SUNXI_HAL_OK: 成功
  - HAL_UART_STATUS_ERROR: 失败

_uart_deinit

- 函数原型: int32_t hal_uart_deinit(int32_t uart_port);
- 作用：卸载 UART 驱动
- 参数：
  - uart_port:UART 端口号
- 返回：
  - SUNXI_HAL_OK: 成功

3.4 hal_uart_send

- 函数原型: int32_t hal_uart_send(int32_t dev, const uint8_t *data, uint32_t num);
- 作用：发送数据
- 参数：
  - dev:UART 端口号
  - data: 发送数据缓冲区

- num: 发送数据长度

- size: 成功发送的字节数

3.5 hal_uart_receive

- 函数原型: int32_t hal_uart_receive(int32_t dev, uint8_t *data, uint32_t num);
- 作用：接收数据
- 参数：
  - dev:UART 端口号
  - data: 接收数据缓冲区
  - num: 接收数据长度
- 返回：
  - size: 成功接收的字节数

3.6 hal_uart_receive_no_block

nt32_t hal_uart_receive_no_block(int32_t dev, uint8_t *data, uint32_t num, int32_t timeout);

- 作用：非阻塞接收
- 参数：
  - dev:UART 端口号
  - data: 接收数据缓冲区
  - num: 接收数据长度
  - timeout: 等待超时时长
- 返回：
  - size: 成功接收的字节数

3.7 hal_uart_receive_polling

- 函数原型: int32_t hal_uart_receive_polling(int32_t dev, uint8_t *data, uint32_t num);
- 作用：轮询接收
- 参数：
  - dev:UART 端口号
  - data: 接收数据缓冲区

- num: 接收数据长度

- size: 成功接收的字节数

3.8 hal_uart_set_hardware_flowcontrol

- 函数原型: void hal_uart_set_hardware_flowcontrol(uart_port_t uart_port);
- 作用：设置硬件流控
- 参数：

_port:UART 端口号

- 返回：

  - void

3.9 hal_uart_disable_flowcontrol

- 函数原型: void hal_uart_disable_flowcontrol(uart_port_t uart_port);
- 作用：禁止硬件流控

  - uart_port:UART 端口号

- 返回：

  - void

3.10 hal_uart_set_loopback

- 函数原型: void hal_uart_set_loopback(uart_port_t uart_port, bool enable);

置 uart 回环

  - uart_port:UART 端口号
  - enable: 是否开启

- 返回：

  - void

4 功能开发

4.1 功能概述

UART 驱动提供了以下核心功能：

- UART 硬件初始化;
- 数据发送和接收;
- 波特率设置;
- 数据位、停止位和校验位配置;
- 发送、接收缓冲区管理;

4.2 开发流程

4.2.1 初始化某个 uart port

步骤 1 定义_uart_init_t 类型 uart 参数变量，并配置波特率、停止位等信息。

步骤 2 调用 hal_uart_init 初始化指定 uart port。

步骤 3 调用 hal_uart_set_hardware_flowcontrol 或 hal_uart_disable_flowcontrol 设置流控。

4.2.2 卸载某个 uart port

步骤 1 调用 hal_uart_deinit 卸载

4.2.3 发送接收数据

步骤 1 调用 hal_uart_send 发送数据。

步骤 2 调用 hal_uart_receive 或 hal_uart_receive_polling 接收数据。

5 模块使用范例

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

```
#include <hal_uart.h>
#include <hal_gpio.h>
#include <hal_clk.h>
#include <hal_reset.h>

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

static int uart_clk_init()
{
    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_UART1);
    clk_num = MAKE_CLKn(AW_SYS_CCU, CLK_BUS_UART1);

    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;
    }

    ret = 0;
```

```
        exit_with_put_clk:
clk_put(clk);

        exit_with_put_rst:
            hal_rst_put(rst);
            return ret;
        }

        void uart_cpu_loopback_test(void)
        {
            u32 port = UART_1;

            int i = 0;
            uart_init_t uart_initstruct;
            gpio_init_t gpio_initstruct;
            uint8_t tbuf[5] = "hello";
trbuf[5]={0};

            uart_clk_init();

            /* init uart1 rx tx pin < -- > PG6 PG7 */
            gpio_initstruct.port = SUNXI_GPIO_G;
            gpio_initstruct.port_num = PIN_6 | PIN_7;
            gpio_initstruct.mul_sel = GPG_UART1;//uart1 tx rx
            hal_gpio_init(&gpio_initstruct);

            /* init uart1 format */
            uart_initstruct.uart_port = port;
            uart_initstruct.baudrate = UART_BAUDRATE_115200;
            uart_initstruct.word_length = UART_WORD_LENGTH_8;
            uart_initstruct.stop_bit = UART_STOP_BIT_1;
            uart_initstruct.parity = UART_PARITY_NONE;
nitstruct.function=UART_MODE;

            hal_uart_init(&uart_initstruct);
            hal_uart_disable_flowcontrol(port);
            hal_uart_set_loopback(port, 1);
            /* send */
            hal_uart_send(port, tbuf, 5);

            udelay(10 * 1000);
            /* loopback receive */

        #ifdef CONFIG_KERNEL_BAREMETAL
            hal_uart_receive_polling(port, rbuf, 5);
        #else
            hal_uart_receive(port, rbuf, 5);
        #endif

        #define UART_DEBUG
        #ifdef UART_DEBUG
            printf("Sending:");
            for (i = 0; i < 5; i++)
                printf("0x%x", tbuf[i]);
            printf(" \n");

            printf("Receiving:");
            for (i = 0; i < 5; i++)
                printf("0x%x", rbuf[i]);
```

```
    printf(" \n");
#endif

    /* verify data */
    for (i = 0; i < 5; i++) {
        if (tbuf[i] != rbuf[i]) {
            printf("uart cpu loopback test failed!\n");
            printf("tx != rx, tbuf[%d]:0x%x, rbuf[%d]:0x%x\n", i, tbuf[i], i, rbuf[i]);
            return;
        }
    }

    printf("uart cpu loopback test passed!\n");
}

#if defined(CONFIG_KERNEL_FREERTOS)
FINSH_FUNCTION_EXPORT_CMD(uart_cpu_loopback_test, uart_cpu_loopback_test, uart hal_v2 APIs

#elifdefined(CONFIG_KERNEL_BAREMETAL)
SHELL_EXPORT_CMD(
SHELL_CMD_PERMISSION(0)|SHELL_CMD_TYPE(SHELL_TYPE_CMD_FUNC)|SHELL_CMD_DISABLE_RETURN,
uart_cpu_loopback_test, uart_cpu_loopback_test, test for uart cpu loopback mode);
#endif
```

6 FAQ

无

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，如有变更，恕不另行通知。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。