



# **RTOS SPI 开发指南**

**版本号: 1.1  
发布日期: 2021.4.14**

## 版本历史

版本号	日期	制/修订人	内容描述
1.0	2020.7.16	AWA1636	1. 初版
1.1	2021.4.14	AWA1636	1. 增加 sys_config.fex



# 目 录

<b>1 前言</b>	<b>1</b>
1.1 文档简介	1
1.2 目标读者	1
1.3 适用范围	1
<b>2 模块介绍</b>	<b>2</b>
2.1 模块功能介绍	2
2.2 pin 引脚配置	2
2.3 相关术语介绍	3
2.4 模块配置介绍	3
2.5 模块源码结构	3
<b>3 模块接口说明</b>	<b>4</b>
3.1 接口列表	4
3.2 接口使用说明	4
3.2.1 SPI 初始化接口	4
3.2.2 SPI 硬件配置接口	5
3.2.3 SPI 发送数据接口	5
3.2.4 SPI 接收数据接口	5
3.2.5 SPI 数据传输接口	6
3.2.6 SPI 去初始化接口	6
<b>4 模块使用范例</b>	<b>7</b>
<b>5 FAQ</b>	<b>8</b>
5.1 如何修改 SPI 时钟	8

# 1 前言

## 1.1 文档简介

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

## 1.2 目标读者

SPI 驱动层/应用层开发/使用/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
F133	Melis	hal_spi.c
V833	Melis	hal_spi.c
F133	Melis	hal_spi.c
R328	FreeRTOS	hal_spi.c

## 2 模块介绍

### 2.1 模块功能介绍

SPI 是一种全双工同步串行接口，可以工作在 Master 模式和 Slave 模式，SPI 主要有以下特点：

- (1) 全双工同步串行接口；
- (2) 5 个时钟源；
- (3) Master/Slave 模式可配置；
- (4) 4 种片选模式，可支持多种外设；
- (5) 片选和时钟的极性和相位可配置；
- (6) 支持中断或 DMA 传输；
- (7) 支持 3 线/4 线 SPI；
- (8) 支持可编程帧长度：0~31bit；
- (9) 支持标准 SPI、双输出 SPI、双输入 SPI、双 I/O SPI、双输出/双输入 SPI。

### 2.2 pin 引脚配置

引脚配置在 source/project/方案/configs/sys\_config\_XXX.fex 中存放 (xx 是存储方案的标识，例如 sys\_config\_nor.cfg、sys\_config\_nand.cfg)。

```
[spi0]
spi0_sclk      = port:PC02<2><0><2><default>
spi0_cs        = port:PC03<2><1><2><default>
spi0_mosi      = port:PC04<2><0><2><default>
spi0_miso      = port:PC05<2><0><2><default>
spi0_wp        = port:PC06<2><0><2><default>
spi0_hold      = port:PC07<2><0><2><default>
```

#### 说明

引脚说明：**port:** 端口 < 复用功能 > < 上下拉 > < 驱动能力 > < 输出值 >

## 2.3 相关术语介绍

术语	解释说明
Sunxi	指 Allwinner 的一系列 SOC 硬件平台
SPI	Serial Peripheral Interface, 同步串行外设接口
SPI Master	SPI 主设备
SPI Device	SPI 从设备

## 2.4 模块配置介绍

配置路径如下：

```
Kernel Setup --->
  Drivers Setup --->
    SoC HAL Drivers --->
      SPI Devices --->
```

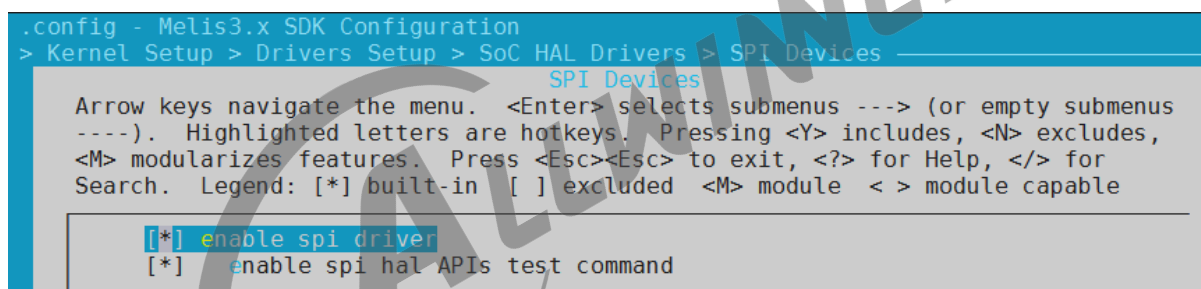


图 2-1: SPI menuconfig

## 2.5 模块源码结构

SPI 模块源码结构如下所示：

```
rtos-hal/
|--hal/source/spi/hal_spi.c //hal层接口代码
|--include/hal/sunxi_hal_spi.h //头文件
```

## 3 模块接口说明

需要包含头文件：

```
#include <hal/sunxi_hal_spi.h>
```

### 3.1 接口列表

SPI 提供的接口列表如下：

```
spi_master_status_t hal_spi_init(hal_spi_master_port_t port);  
spi_master_status_t hal_spi_write(hal_spi_master_port_t port, const void *buf, uint32_t  
    size);  
spi_master_status_t hal_spi_deinit(hal_spi_master_port_t port);  
spi_master_status_t hal_spi_read(hal_spi_master_port_t port, void *buf, uint32_t size);  
spi_master_status_t hal_spi_xfer(hal_spi_master_port_t port, hal_spi_master_transfer_t *  
    transfer);  
spi_master_status_t hal_spi_hw_config(hal_spi_master_port_t port, hal_spi_master_config_t *  
    spi_config);
```

### 3.2 接口使用说明

#### 3.2.1 SPI 初始化接口

- 原型：spi\_master\_status\_t hal\_spi\_init(hal\_spi\_master\_port\_t port)
- 功能：SPI 模块初始化，主要申请中断、pinctrl 初始化、clk 初始化等
- 参数：
  - port：SPI 端口号
- 返回值：
  - 0 代表成功
  - 负数代表失败

### 3.2.2 SPI 硬件配置接口

- 原型：spi\_master\_status\_t hal\_spi\_hw\_config(hal\_spi\_master\_port\_t port, hal\_spi\_master\_config\_t \*spi\_config)
- 功能：配置 SPI 模块，包括 SPI 总线最大传输速率、片选模式等
- 参数：
  - port: SPI 端口号
  - spi\_config: 配置信息
- 返回值：
  - 0 代表成功
  - 负数代表失败

### 3.2.3 SPI 发送数据接口

- 原型：spi\_master\_status\_t hal\_spi\_write(hal\_spi\_master\_port\_t port, const void \*buf, uint32\_t size)
- 功能：发送数据，调 hal\_spi\_xfer 接口
- 参数：
  - port: SPI 端口号
  - buf: 发送数据
  - size: 发送数据大小
- 返回值：
  - 0 代表成功
  - 负数代表失败

### 3.2.4 SPI 接收数据接口

- 原型：spi\_master\_status\_t hal\_spi\_read(hal\_spi\_master\_port\_t port, void \*buf, uint32\_t size)
- 功能：接收数据，调 hal\_spi\_xfer 接口
- 参数：
  - port: SPI 端口号
  - buf: 接收数据
  - size: 接收数据大小
- 返回值：
  - 0 代表成功
  - 负数代表失败



### 3.2.5 SPI 数据传输接口

- 原型：spi\_master\_status\_t hal\_spi\_xfer(hal\_spi\_master\_port\_t port, hal\_spi\_master\_transfer\_t \*transfer)
- 功能：发送或接收数据
- 参数：
  - port：SPI 端口号
  - transfer：传输配置信息
- 返回值：
  - 0 代表成功
  - 负数代表失败

### 3.2.6 SPI 去初始化接口

- 原型：spi\_master\_status\_t hal\_spi\_deinit(hal\_spi\_master\_port\_t port)
- 功能：SPI 模块去初始化
- 参数：
  - port：SPI 端口号
- 返回值：
  - 0 代表成功
  - 负数代表失败

## 4 模块使用范例

---

可参考驱动 APIs 测试代码（hal/test/spi/）。



## 5 FAQ

### 5.1 如何修改 SPI 时钟

1. 申请 SPI 初始化时，配置 `hal_spi_master_config_t` 结构体里 `clock_frequency` 参数，如：

```
static hal_spi_master_config_t cfg =
{
    .clock_frequency = 10*1000*1000,//10MHz
    .slave_port = HAL_SPI_MASTER_SLAVE_0,
    .cpha = HAL_SPI_MASTER_CLOCK_PHASE0,
    .cpol = HAL_SPI_MASTER_CLOCK_POLARITY0,
    //.bit_order = HAL_SPI_MASTER_LSB_FIRST,//LSB
};//SPI configure
```

2. 如果没有配置 `clock_frequency`，那么默认使用宏 `SPI_MOD_CLK` 配置，具体定义在：  
`lichee/rtos-hal/hal/source/spi/common_spi.h`




## 著作权声明

版权所有 © 2021 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

、 全志科技、（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。