Rockchip RT-Thread SPI2APB

文件标识：RK-KF-YF-498

发布版本：V1.2.0

日期：2024-05-25

文件密级：□绝密 □秘密 □内部资料 ■公开

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址： 福建省福州市铜盘路软件园A区18号

网址： www.rock-chips.com

客户服务电话： +86-4007-700-590

客户服务传真： +86-591-83951833

客户服务邮箱： fae@rock-chips.com

前言

概述

本文主要描述了 ROCKCHIP RT-Thread SPI 驱动的使用方法。

产品版本

芯片名称	内核版本
所有使用 RK RT-Thread SDK 的芯片产品	RT-Thread

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

日期	版本	作者	修改说明
2024-03-28	V1.0.0	林鼎强	初始版本
2024-04-26	V1.1.0	林鼎强	增加常见问题
2024-05-25	V1.2.0	林鼎强	增加 flash_obj 业务支持

目录

1. 简介

1.1 须知

SPI2APB 控制器在标准的 SPI 通讯协议上叠加 RK 私有协议，不同于无叠加协议、更通用的的 RK SPI 模块，建议客户审图时关注该情况。

1.2 控制器特性

- 支持2种SPI模式，SPI mode 0/2，推荐使用 mode 0
- 支持8bits 传输
- 支持中断
- 支持读写控制器内部寄存器
- 支持 LSB/MSB 可配
- 支持 Little Endian/Big Endian 可配
- 仅支持单工传输
- 仅支持 slave mode

SPI master 应遵从 SPI2APB 协议发起特定数据的传输，协议基础框架 `<CMD-32bits> [ADDR-32bits] [DUMMY-32bits] [READ_BEGING-32bits] [DATA]`

支持以下具体数据传输协议：

简介	协议简称	协议（**CMD-32bits** 小端传输）
读数据	read data	`<0x00000077> <ADDR> <DUMMY> <READ_BEGING> <DATA>`
写数据	write data	`<0x00000011> <ADDR> <DATA>`
写控制器内部 REG0 寄存器	write msg reg0	`<0x00010011> <DATA-32bits>`
写控制器内部 REG1 寄存器，触发中断	write msg reg1	`<0x00020011> <DATA-32bits>`
读取前一笔传输状态	query state	`<0x000000ff> <DATA-32bits>`
读控制器内部 REG2 寄存器	query msg reg2	`<0x000001ff> <DATA-32bits>`

说明：

- [ADDR-32bits] 为 SPI2APB 支持的内存地址，通常为 sram/dram
- [DUMMY-32bits] 值无要求，可以直接填 0
- [READ_BEGING-32bits] 值为 0x000000AA
- query state 返回值参考 TRM SPI2APB 章节 “ State Register Information” 说明

1.3 接口特性

SOC	接口最高速率
RK2118	50MHz

说明：

- 接口最高速率为理论速率，受设备走线 PCB 质量影响，以实测为准

2. 软件

2.1 代码路径

框架代码：

```
components/drivers/include/drivers/spi.h
components/drivers/spi/spi_core.c
components/drivers/spi/spi_dev.c
```

驱动适配层：

```
bsp/rockchip/common/drivers/drv_spi2apb.c
bsp/rockchip/common/hal/lib/hal/src/hal_spi2apb.c
```

测试命令，用户程序完全可以参照以下驱动：

```
bsp/rockchip-common/tests/spi2apb_test.c
```

2.2 配置

- 打开配置带有 SPI2APB 字样的选项

```
RT-Thread bsp drivers  --->
    RT-Thread rockchip "project" drivers  --->
        [*] Enable SPI
        [*]   Enable SPI0 (SPI2APB)
        [ ]   Enable SPI1
        [ ]   Enable SPI2
```

- 板级配置 iomux。

2.3 使用配置

默认配置：

- LSB、Little Endian

drv_spi2apb.c 源码修改配置参考：

```
diff --git a/bsp/rockchip/common/drivers/drv_spi2apb.c
b/bsp/rockchip/common/drivers/drv_spi2apb.c
index 15045270fe..ef055d8d03 100644
--- a/bsp/rockchip/common/drivers/drv_spi2apb.c
+++ b/bsp/rockchip/common/drivers/drv_spi2apb.c
@@ -45,7 +45,7 @@ struct rockchip_spi2apb

 static struct rockchip_spi2apb rk_spi2apb =
 {
-    .device.config.mode = (SPI2APB_LITTLE_ENDIAN | SPI2APB_LSB),
+    .device.config.mode = (SPI2APB_LITTLE_ENDIAN | SPI2APB_MSB),
     .device.config.clock_polarity = SPI2APB_TXCP_INVERT,
     .base = SPI2APB,
     .irq = SPISLV0_IRQn,
```

说明：

- 建议只针对 LSB/MSB 做修改，其余配置默认配置

3. 测试

3.1 配置

```
RT-Thread bsp test case  --->
    [*] RT-Thread Common Test case  --->
        [*] Enable BSP Common TEST
        [*]  Enable BSP Common SPI2APB TEST
```

3.2 测试命令

```
spi2apb_test config spi2apb 1 2          # <mode> <polarity>
spi2apb_test cb spi2apb                  # 注册 SPI2APB->REG1 中断回调
spi2apb_test read spi2apb 0              # 读取 SPI2APB->REG0 信息，建议调试方式，先主
设备端参考 'write msg reg0' 协议写，后从设备端执行该命令读
spi2apb_test read spi2apb 1              # 读取 SPI2APB->REG1 信息，建议调试方式，先主
端参考 'write msg reg1' 协议写，后从端执行该命令读
spi2apb_test query spi2apb               # 读取 SPI2APB->SR 信息
spi2apb_test write spi2apb 0x12345678    # 写入 SPI2APB->REG2 信息，建议调试方式，先从
端执行该命令，后主端参考 'query msg reg2' 协议读
```

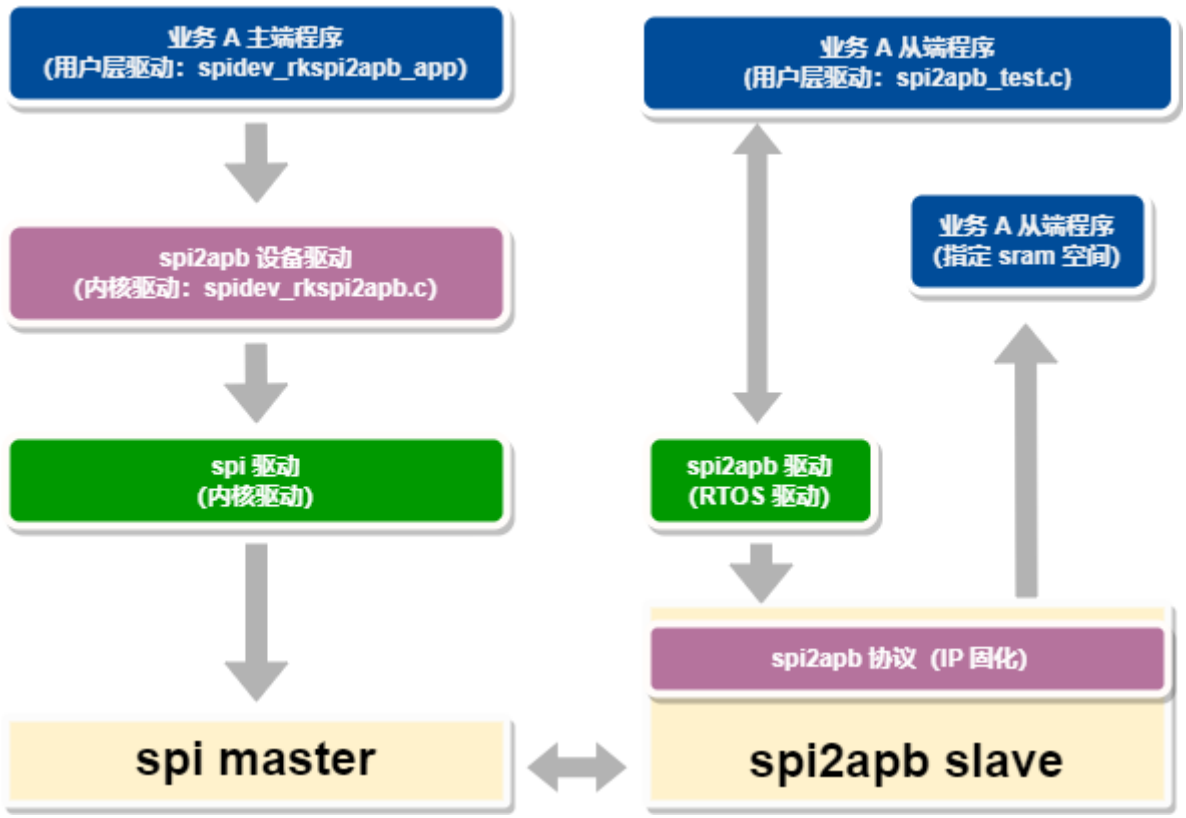
说明：

- 主设备端 - SPI master，从端 - RK SPI2APB slave
- mode，支持可配：
  - bit0：0-LSB 1-MSB
  - bit1：0-LittleEndian 1-BigEndian
- polarity，支持可配（仅做调试使用，实际请保持 TX invert/RX nornal 的默认配置，即 polarity = 2）：
  - bit0：0-RX normal 1-RX invert
  - bit1：0-TX normal 1-TX invert

4. 业务搭建说明

SPI2APB 业务要求主从两端基于 SPI2APB 协议原理搭建数据流、控制信息等业务行为，其中运行在 SPI master 主设备端的业务代码、设备驱动可以基于 RK 参考代码进一步开发，源码获取：[SPI开发资料 - FAE 项目 - Rockchip Redmine (rock-chips.com)]

4.1 业务基础原理



以 RV1106 SPI master 对接 RK2118 SPI2APB 为例，从源码认识业务实现：

所属层次	主设备（**Linux 5.10**）	从设备（**HAL + RTOS**）
用户层业务	spidev_rkspi2apb_app c 程序，调用 spi2apb 设备节点的 ioctl/read/write 接口	test_spi2apb.c 程序，获取 REG 寄存器信息
内核态设备驱动	spidev_rkspi2apb.c 源码，依旧 spi2apb 协议实现 ioctl/read/write 接口	
控制器驱动	spi-rockchip.c 源码	drv_spi2apb.c hal_spi2apb.c
控制器	spi master	spi2apb slave

说明：

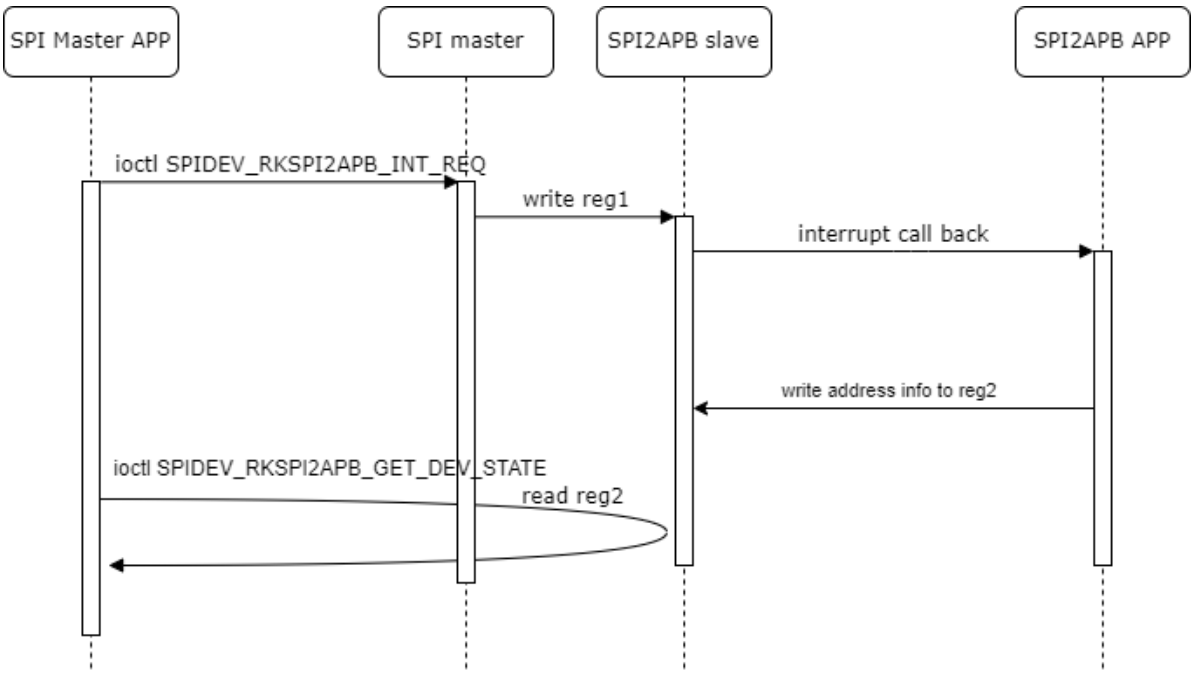
- spidev_rkspi2apb.c ioctl 接口说明：

```
#define SPIDEV_RKSPI2APB_GET_DEV_STATE      _IOR(SPIDEV_RKSPI2APB_BASE, 0, __u32)
/* 参考 'query msg reg2' 协议 */
#define SPIDEV_RKSPI2APB_SET_DST_ADDR       _IOW(SPIDEV_RKSPI2APB_BASE, 1, __u32)
/* 设置传输的目标内存地址，指向从设备端内存地址 */
#define SPIDEV_RKSPI2APB_SEND_MSG       _IOW(SPIDEV_RKSPI2APB_BASE, 2, __u32) /*
参考 'write msg reg0' 协议 */
#define SPIDEV_RKSPI2APB_INT_REQ        _IOW(SPIDEV_RKSPI2APB_BASE, 3, __u32) /*
参考 'write msg reg1' 协议 */
```

- spidev_rkspi2apbc.c 读写说明：

```
write(spidev_fd, buffer_w, size); /* 参考 'write data' 协议，addr 由 ioctl
SPIDEV_RKSPI2APB_SET_DST_ADDR 设定 */
read(spidev_fd, buffer_r, size); /* 参考 'read data' 协议，addr 由 ioctl
SPIDEV_RKSPI2APB_SET_DST_ADDR 设定 */
```

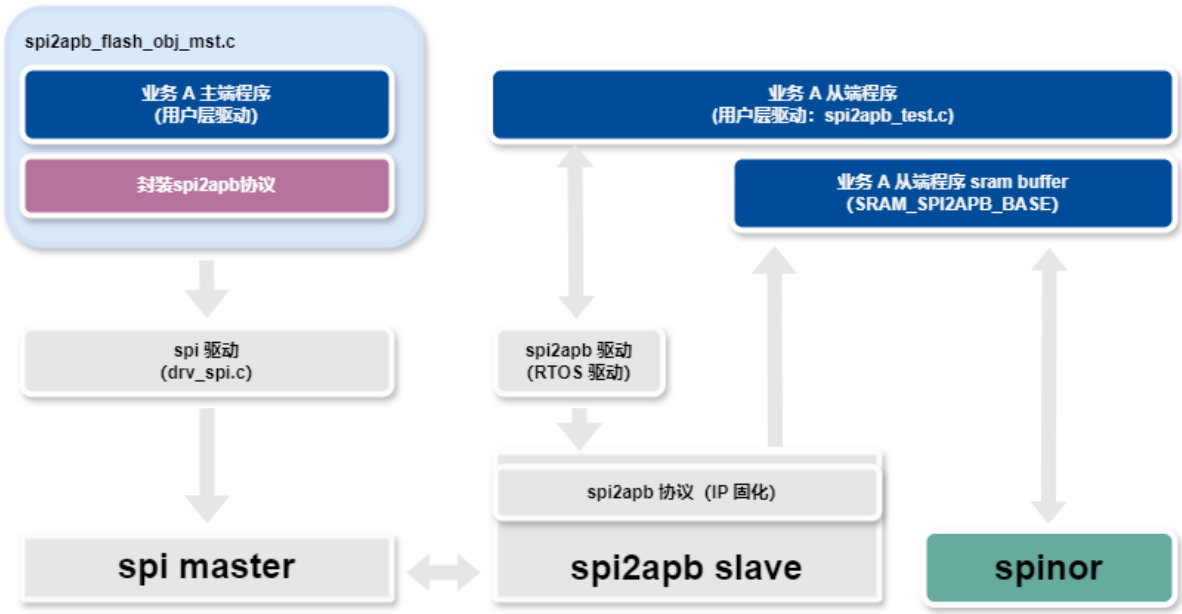
- 如何协定 SPI2APB 传输的目标内存地址：
  - 预设定（推荐使用该方案），通常需要在从设备预留一块已知的目标 sram 地址作为传输地址，主设备端使用 ioctl SPIDEV_RKSPI2APB_SET_DST_ADDR 设定，通常 RT-Thread 方案会在连接脚本中通过宏 __spi2apb_buffer_start__ 预留该内存
  - 自定义，从设备端申请一段 sram 地址空间，两端自定义业务，将地址信息最终传输到主设备端，例如以下参考流程：



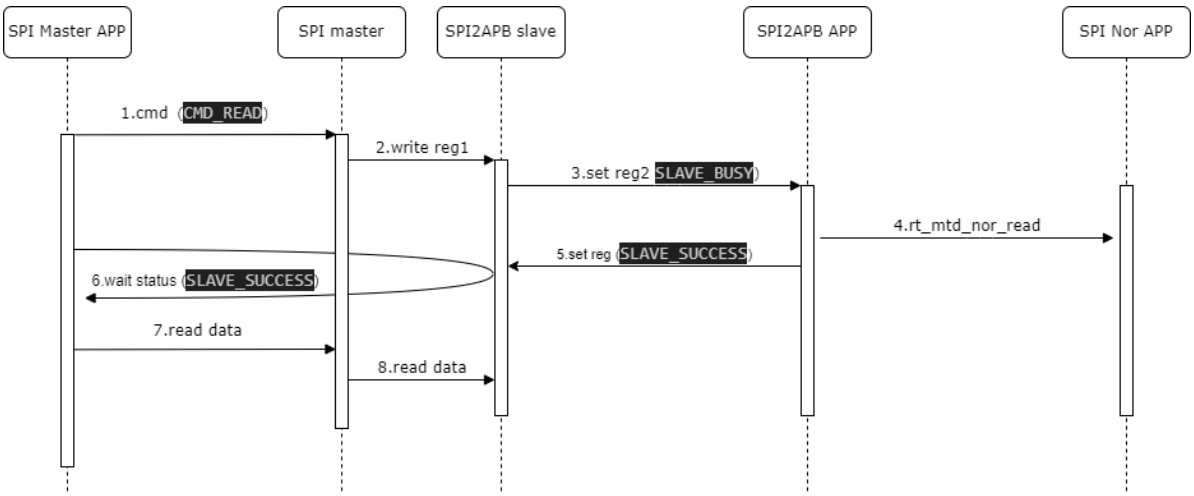
4.2 Flash Object 业务

Flash object 业务通过 spi/spi2apb 接口传输数据请求最终实现 spi 主设备端读写 spi2apb 从设备端设备上 spinor。

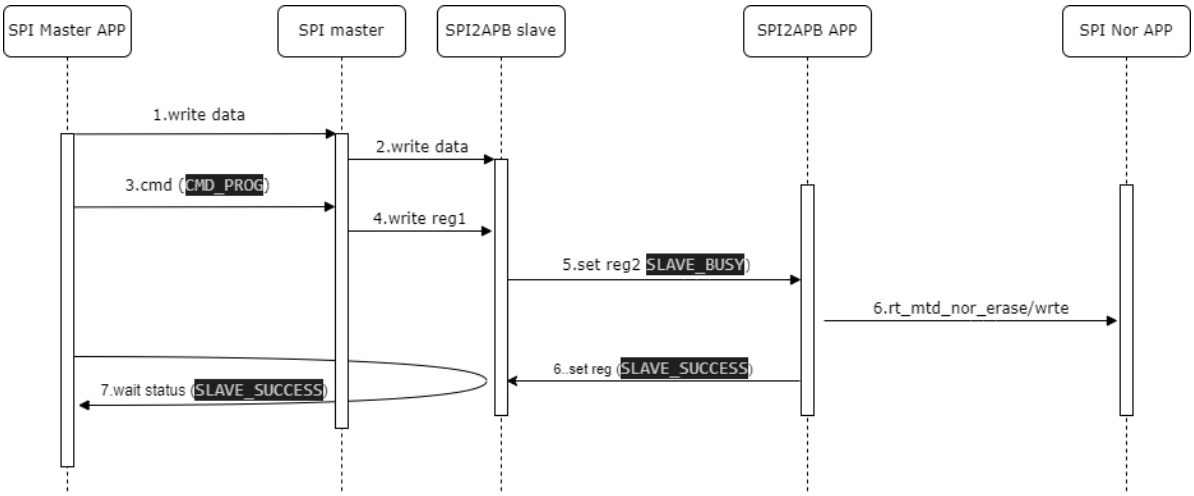
4.2.1 基础原理



读 nor flash 流程：



写 nor flash 流程：



4.2.2 接口

flash_obj master 端接口：

```
rt_err_t spi2apb_flash_obj_mst_init(struct rt_spi_device *spi_device);
/**
  * @brief  Write sector from remote spinor device.
  * @param  sec: pointer to spinor 4KB sector offset.
  * @param  buffer: data buffer.
  * @return rt_err_t
  */
rt_err_t spi2apb_flash_obj_mst_write_sector(rt_uint32_t sec, void *buffer);
/**
  * @brief  Read sector from remote spinor device.
  * @param  sec: pointer to spinor 4KB sector offset.
  * @param  buffer: data buffer.
  * @return rt_err_t
  */
rt_err_t spi2apb_flash_obj_mst_read_sector(rt_uint32_t sec, void *buffer);
```

flash_obj master 端 spi2apb 封装接口：

```

/**
```

```
  * @brief  Write data to remote sram by spi2apb interface.
  * @param  addr: pointer to remote sram address.
  * @param  buffer: data buffer.
  * @param  size: data size.
  * @param  trigger: if valid write spi2apb reg1 to trigger a interrupt.
  * @return rt_err_t
  */
static rt_err_t spi2apb_flash_obj_mst_spi2apb_progsram(uint32_t addr, uint8_t
*buffer, uint32_t size, uint32_t trigger);
/**
  * @brief  Read data from remote sram by spi2apb interface.
  * @param  addr: pointer to remote sram address.
  * @param  buffer: data buffer.
  * @param  size: data size.
  * @param  trigger: if valid write spi2apb reg1 to trigger a interrupt.
  * @return rt_err_t
  */
static rt_err_t spi2apb_flash_obj_mst_spi2apb_readsram(uint32_t addr, uint8_t
*buffer, uint32_t size, uint32_t trigger);
/**
  * @brief  Write msg reg in spi2apb.
  * @param  index: reg index.
  * @param  val: reg value.
  * @param  status: if valid check write success.
  * @return rt_err_t
  */
static rt_err_t spi2apb_flash_obj_mst_spi2apb_writemsg(uint8_t index, uint32_t
value, bool status);
/**
  * @brief  Query msg reg2 in spi2apb.
  * @return rt_err_t
  */
static uint32_t spi2apb_flash_obj_mst_spi2apb_querymsg(void);
/**
  * @brief  Query state abort last transmission.
  * @return uint32_t status.
  */
static uint32_t spi2apb_flash_obj_mst_spi2apb_querystatus(void);
```

flash_obj slave 端接口：

```
rt_err_t spi_test_spi2apb_flash_obj_init(void);
static void spi_test_spi2apb_flash_obj_work(struct rt_work *work, void
*work_data);
rt_err_t spi_test_spi2apb_flash_obj_write(rt_uint8_t index, rt_uint16_t addr,
void *buffer);
rt_err_t spi_test_spi2apb_flash_obj_read(rt_uint8_t index, rt_uint16_t addr, void
*buffer);
rt_err_t spi_test_spi2apb_flash_obj_set_status(rt_uint8_t index, rt_uint8_t
status);
```

说明：

- 完成初始化后注册 spi2apb 中断 callback，reg1 下半文 workqueue 解析事务，完成 spinor read/write 和状态机状态位设置

4.2.3 Flash obj 配置

```
CONFIG_RT_USING_COMMON_TEST_SPI2APB_FLASH_OBJ = y
```

4.2.4 Flash obj 测试命令

flash_obj slave 端命令：

```
spi2apb_test flash_obj spi2apb
```

flash_obj master 端命令：

```
1. spi2apb_flash_obj_mst spi2apb_progsram <spi_dev> <size>
2. spi2apb_flash_obj_mst spi2apb_readsram <spi_dev> <size>
3. spi2apb_flash_obj_mst spi2apb_writemsg <spi_dev> <reg> <val>
4. spi2apb_flash_obj_mst spi2apb_querymsg <spi_dev>
5. spi2apb_flash_obj_mst write_sector <spi_dev> <sector> <val> <loops>
6. spi2apb_flash_obj_mst read_sector <spi_dev> <sector> <loops>
7. spi2apb_flash_obj_mst stress <spi_dev> <sector> <n_sec>
like:
        spi2apb_flash_obj_mst spi2apb_progsram spi1_0 32
        spi2apb_flash_obj_mst spi2apb_readsram spi1_0 32
        spi2apb_flash_obj_mst spi2apb_writemsg spi1_0 1 55
        spi2apb_flash_obj_mst spi2apb_querymsg spi1_0
        spi2apb_flash_obj_mst write_sector spi1_0 512 1 10
        spi2apb_flash_obj_mst read_sector spi1_0 512 10
        spi2apb_flash_obj_mst stress spi1_0 512 32
```

说明：

- spi2apb_progsram 对应 spi2apb 协议 write data，完成写后 write msg reg1 触发 spi2apb 中断
- spi2apb_readsram 对应 spi2apb 协议 read data，读之前先等待 reg2 置位为 status SPI2APB_FLASH_OBJ_CMD_READ_SRAM_READY
- spi2apb_writemsg 对应 spi2apb 协议 write msg reg0/1
- spi2apb_querymsg 对应 spi2apb 协议 read msg reg2
- write_sector 为 flash_obj 写 nor flash 操作，单位 sector 4KB
- read_sector 为 flash_obj 读 nor flash 操作，单位 sector 4KB
- stress 为 flash_obj 写回读拷机测试，单位 sector 4KB

5. 常见问题

5.1 SPI2APB 传输失败

建议参考 “测试” 章节，配置控制器为 SPI mode 0，lsb，小端，注册回调函数，命令如下：

```
spi2apb_test config spi2apb 1 2
spi2apb_test cb spi2apb
```

说明：

- master 参考 “数据传输协议” 说明，发起传输，建议先尝试 "write reg1"，触发 SPI2APB 中断，常规调试流程：
    - 互联设备是否已经共地
    - 示波器 master 设备测量 csn/clock 是否有有效信号
    - slave 硬件线路是否连接正确，线路上是否有 0Ω 电阻未贴片
- master 传输后，slave 或 master 接收到错误的数据，常规调试流程：
    - 降速
    - 确认数据是否存在规律：
        - 是否匹配 LSB/MSB 或者大小端配置问题
        - 是否仅接收到部分有效数据，且数据是否与 cache line size 有关，如果是，大概率为 slave 操作数据前后未添加 cache 一致性操作
            - 通常为在读传输后添加 cache invalid，在写传输前添加 cache flush
            - 如果读传输所用 buffer 有清零、写数据等动作，建议 传输前添加 cache flush