Rockchip SPI 开发指南

发布版本：1.00

作者邮箱：hhb@rock-chips.com

日期：2016.06

文件密级：公开资料

前言

概述

产品版本

芯片名称	内核版本
采用linxu4.4的所有芯片	Linux4.4

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
2016-06-29	V1.0	洪慧斌	

---

1 Rockchip SPI功能特点

SPI（serial peripheral interface），以下是linux 4.4 spi驱动支持的一些特性：

- 默认采用摩托罗拉 SPI协议
- 支持8位和16位
- 软件可编程时钟频率和传输速率高达50MHz
- 支持SPI 4种传输模式配置
- 每个SPI控制器支持一个到两个片选

2 内核软件

2.1 代码路径

```
drivers/spi/spi.c             spi驱动框架
drivers/spi/spi-rockchip.c    rk spi各接口实现
drivers/spi/spidev.c          创建spi设备节点，用户态使用。
drivers/spi/spi-rockchip-test.c  spi测试驱动，需要自己手动添加到Makefile编译
Documentation/spi/spidev_test.c  用户态spi测试工具
```

2.2 内核配置

```
Device Drivers  --->
    [*] SPI support  --->
        <*>   Rockchip SPI controller driver
```

2.3 DTS节点配置

```
&spi1 {                                引用spi 控制器节点
status = "okay";
max-freq = <48000000>;                 spi内部工作时钟
dma-names = "tx","rx";                 使能DMA模式，一般通讯字节少于32字节的不建议用
    spi_test@10 {
        compatible ="rockchip,spi_test_bus1_cs0";  与驱动对应的名字
        reg = <0>;                     片选0或者1
        spi-max-frequency = <24000000>;   spi clk输出的时钟频率，不超过50M
        spi-cpha ;                     如果有配，cpha为1

        spi-cpol ;                     如果有配，cpol为1,clk脚保持高电平
```

```
        spi-cs-high;                如果有配，每传完一个数据，cs都会被拉高，再拉低
        status = "okay";            使能设备节点
    };
};
```

一般只需配置以下几个属性就能工作了。

```
        spi_test@11 {
                compatible ="rockchip,spi_test_bus1_cs1";
                reg = <1>;
                spi-max-frequency = <24000000>;
                status = "okay";
        };
```

max-freq 和 spi-max-frequency的配置说明：

- spi-max-frequency 是SPI的输出时钟，是max-freq分频后输出的，关系是max-freq >= 2*spi-max-frequency。
- max-freq 不要低于24M，否则可能有问题。
- 如果需要配置spi-cpha的话， max-freq <= 6M，1M <= spi-max-frequency >= 3M。

2.3 SPI设备驱动

设备驱动注册:

```
static int spi_test_probe(struct spi_device *spi)
{
        int ret;
        int id = 0;
        if(!spi)
            return -ENOMEM;
        spi->bits_per_word= 8;
        ret= spi_setup(spi);
        if(ret < 0) {
            dev_err(&spi->dev,"ERR: fail to setup spi\n");
            return-1;
        }
        return ret;
}
static int spi_test_remove(struct spi_device *spi)
{
        printk("%s\n",__func__);
        return 0;
}
static const struct of_device_id spi_test_dt_match[]= {
        {.compatible = "rockchip,spi_test_bus1_cs0", },
        {.compatible = "rockchip,spi_test_bus1_cs1", },
        {},
};
MODULE_DEVICE_TABLE(of,spi_test_dt_match);
static struct spi_driver spi_test_driver = {
        .driver = {
```

```
28              .name  = "spi_test",
29              .owner = THIS_MODULE,
30              .of_match_table = of_match_ptr(spi_test_dt_match),
31          },
32          .probe = spi_test_probe,
33          .remove = spi_test_remove,
34  };
35  static int __init spi_test_init(void)
36  {
37          int ret = 0;
38          ret = spi_register_driver(&spi_test_driver);
39          return ret;
40  }
41  device_initcall(spi_test_init);
42  static void __exit spi_test_exit(void)
43  {
44          return spi_unregister_driver(&spi_test_driver);
45  }
46  module_exit(spi_test_exit);
```

对spi读写操作请参考include/linux/spi/spi.h，以下简单列出几个

```
1  static inline int
2  spi_write(struct spi_device *spi,const void *buf, size_t len)
3  static inline int
4  spi_read(struct spi_device *spi,void *buf, size_t len)
5  static inline int
6  spi_write_and_read(structspi_device *spi, const void *tx_buf, void *rx_buf, size_t len)
```

2.4 User mode SPI device配置说明

User mode SPI device 指的是用户空间直接操作SPI接口，这样方便众多的SPI外设驱动跑在用户空间，

不需要改到内核，方便驱动移植开发。

2.4.1 内核配置

```
1  Device Drivers  --->
2      [*] SPI support  --->
3          [*]   User mode SPI device driver support
```

2.4.2 DTS配置

```
&spi0 {
    status = "okay";
    max-freq = <50000000>;
    spi_test@00 {
        compatible = "rockchip,spidev";
        reg = <0>;
        spi-max-frequency = <5000000>;
    };
};
```

2.4.3 内核补丁

```
diff --git a/drivers/spi/spidev.c b/drivers/spi/spidev.c
index d0e7dfc..b388c32 100644
--- a/drivers/spi/spidev.c
+++ b/drivers/spi/spidev.c
@@ -695,6 +695,7 @@ static struct class *spidev_class;
static const struct of_device_id spidev_dt_ids[] = {
        { .compatible = "rohm,dh2228fv" },
        { .compatible = "lineartechnology,ltc2488" },
+       { .compatible = "rockchip,spidev" },
        {},
};
MODULE_DEVICE_TABLE(of, spidev_dt_ids);
```

说明：较旧的内核可能没有2.4.1 和2.4.3 ，需要手动添加，如果已经包含这两个的内核，只要添加2.4.2即可。

2.4.4 使用说明

驱动设备加载注册成功后，会出现类似这个名字的设备：/dev/spidev1.1

请参照Documentation/spi/spidev_test.c

3 SPI 内核测试驱动

3.1 内核驱动

```
drivers/spi/spi-rockchip-test.c  需要手动添加编译
```

3.2 DTS配置

```
&spi0 {
        status = "okay";
        max-freq = <48000000>;   //spi internal clk, don't modify
        //dma-names = "tx", "rx";   //enable dma
        pinctrl-names = "default";  //pinctrl according to you board
        pinctrl-0 = <&spi0_clk &spi0_tx &spi0_rx &spi0_cs0 &spi0_cs1>;
        spi_test@00 {
                compatible = "rockchip,spi_test_bus0_cs0";
```

```
                id = <0>;        //这个属性spi-rockchip-test.c用来区分不同的spi从设备的
                reg = <0>;   //chip select  0:cs0  1:cs1
                spi-max-frequency = <24000000>;   //spi output clock
                //spi-cpha;      //not support
                //spi-cpol;     //if the property is here it is 1:clk is high, else 0:clk is low  when idle
        };

        spi_test@01 {
                compatible = "rockchip,spi_test_bus0_cs1";
                id = <1>;
                reg = <1>;
                spi-max-frequency = <24000000>;
                spi-cpha;
                spi-cpol;
        };
};
```

3.3 驱动log

```
[    0.530204] spi_test spi32766.0: fail to get poll_mode, default set 0
[    0.530774] spi_test spi32766.0: fail to get type, default set 0
[    0.531342] spi_test spi32766.0: fail to get enable_dma, default set 0
以上这几个没配的话，不用管
[    0.531929] rockchip_spi_test_probe:name=spi_test_bus1_cs0,bus_num=32766,cs=0,mode=0,speed=5000000
[    0.532711] rockchip_spi_test_probe:poll_mode=0, type=0, enable_dma=0
这是驱动注册成功的标志
```

3.4 测试命令

```
echo write 0 10 255 > /dev/spi_misc_test
echo write 0 10 255 init.rc > /dev/spi_misc_test
echo read 0 10 255 > /dev/spi_misc_test
echo loop 0 10 255 > /dev/spi_misc_test
echo setspeed 0 1000000 > /dev/spi_misc_test
```

echo 类型 id 循环次数 传输长度 > /dev/spi_misc_test

echo setspeed id 频率（单位Hz） > /dev/spi_misc_test

如果需要，可以自己修改测试case。

4 常见问题

- 调试前确认驱动有跑起来
- 确保SPI 4个引脚的IOMUX配置无误
- 确认TX送时，TX引脚有正常的波形，CLK 有正常的CLOCK信号，CS信号有拉低
- 如果clk频率较高，可以考虑提高驱动强度来改善信号