Rockchip SPI Developer Guide

ID: RK-KF-YF-075

Release Version: V3.5.0

Release Date: 2024-09-05

Security Level: □Top-Secret □Secret □Internal ■Public

Rockchip Electronics Co., Ltd.

No.18 Building, A District, No.89, software Boulevard Fuzhou, Fujian,PRC

Website: www.rock-chips.com

Customer service Tel: +86-4007-700-590

Customer service Fax: +86-591-83951833

Customer service e-Mail: fae@rock-chips.com

Preface

Overview

This article introduces the Linux SPI driver principle and basic debugging methods.

Product Version

Chipset	Kernel Version
All chips develop in linux4.4	Linux 4.4
All chips develop in linux4.19 and above	Linux 4.19 and above

Intended Audience

This document (this guide) is mainly intended for:

Technical support engineers
Software development engineers

Revision History

Version	Author	Date	Change Description
V1.0.0	Huibin Hong	2016-06-29	Initial version
V2.0.0	Jon Lin	2019-12-09	Support Linux 4.19
V2.1.0	Jon Lin	2020-02-13	Adjust SPI slave configuration
V2.2.0	Jon Lin	2020-07-14	Linux 4.19 DTS configuration change, Optimize document layout
V2.3.0	Jon Lin	2020-11-02	Add comment for supporting spi-bus cs-gpios property
V2.3.1	Jon Lin	2020-12-11	Update Linux 4.4 SPI slave description
V2.3.2	Jon Lin	2021-07-06	Add more add configuration description、Add more cs-gpios description
V2.4.0	Jon Lin	2021-08-31	Add FAQs and reduce redundant configurations
V2.5.0	Jon Lin	2021-12-27	Support Linux 5.10
V2.6.0	Jon Lin	2023-06-22	Added SPI Slave Software In Kernel、rockchip,poll-only support and explanation of common problems
V2.7.0	Jon Lin	2023-08-15	Explanation of the optimization direction for increasing SPI transmission rate and high CPU usage
V2.8.0	Jon Lin	2023-10-23	Add SPI interface speed description
V2.8.1	Jon Lin	2023-10-24	Modifying incorrect dts node property
V2.9.0	Jon Lin	2023-12-03	Support spi-rockchip-slave source code、Add SPI Slave Notice for customers
V3.0.0	Jon Lin	2023-12-03	Update SPI Slave instructions, remove sram buffer support.
V3.1.0	Jon Lin	2024-03-04	Add RK3576.
V3.2.0	Jon Lin	2024-04-26	Modify SPI clock rate
V3.2.1	Jon Lin	2024-04-26	Modify SPI slave rockchip object structure diagram
V3.3.0	Jon Lin	2024-06-20	Update RK3528/RK3576 interface rate

Version	Author	Date	Change Description
V3.4.0	Jon Lin/Xuhui Lin	2024-07-26	Add RV1103B/RK3506
V3.5.0	Jon Lin	2024-09-05	Add RV1106B

Contents

Rockchip SPI Developer Guide

1. Feature of Rockchip SPI
   - 1.1 SPI interface rate
2. Kernel Software
   - 2.1 Code Path
   - 2.2 SPI Device Configuration: RK SPI As Master Port
   - 2.3 SPI Device Configuration: RK SPI As Slave Port
   - 2.4 SPI Slave Notice
     - 2.4.1 Suggest Performance Mode
     - 2.4.2 Suggest 16bits Width
     - 2.4.3 Other Notes
   - 2.5 SPI Device Driver
   - 2.6 User mode SPI device Configuration
   - 2.7 Support cs-gpios
     - 2.7.1 Configuration of Linux 4.4
     - 2.7.2 Configuration of Linux 4.19 and above
3. SPI Testing Driver in Kernel
   - 3.1 Code Path
   - 3.2 SPI Testing Device Configuration
   - 3.3 Test Command
4. SPI Slave Software In Kernel
   - 4.1 Introduction
   - 4.2 SPI Slave Testing Device Configuration
   - 4.3 Test Command
5. FAQ
   - 5.1 SPI no signal
   - 5.2 How to design application code in SPI
   - 5.3 Delay sampling clock configuration
   - 5.4 SPI transmission method description
   - 5.5 SPI Transfer Rate and CPU Usage Optimization Directions

1. Feature of Rockchip SPI

The serial peripheral interface is called SPI, the following are some of the features supported by the Linux 4.4 SPI driver:

- Motorola SPI protocol is used by default
- Supports 8-bit and 16-bit
- Software programmable clock frequency
- Support 4 transfer mode configurations of SPI
- One or two chips selects per SPI controller
- Only holding SPI slave mode, with and only SPI-CS0N as the CS input pin:
    - Switching to GPIO function is not allowed during the input process
    - CS1N substitution not supported

the following are some of the new features supported by the Linux 4.19 SPI driver:

- Support both slave and master mode

1.1 SPI interface rate

SOC	Master Mode Interface maximum speed	Slave Mode Interface maximum speed
RK3506	50MHz	50MHz
RV1106B/RV1103B	50MHz	33MHz
RK3576	50MHz	33MHz
RK3562	50MHz	33MHz
RK3528	50MHz	33MHz
RV1106/RV1103	50MHz	33MHz
RK3588	50MHz	33MHz
RV1126/RV1109	50MHz	16MHz
RK3568	50MHz	33MHz
RK1808	50MHz	16MHz
RK3308	50MHz	16MHz
Others	50MHz	16MHz

Note:

- The maximum speed of the interface is the theoretical rate, which is affected by the quality of the device's PCB wiring. The actual measurement shall prevail
- Due to PLL strategy reasons, some platforms are unable to accurately divide the frequency to the upper limit value. In reality, the maximum frequency division value shall prevail

2. Kernel Software

2.1 Code Path

```
drivers/spi/spi.c               /* SPI Driver framework */
drivers/spi/spi-rockchip.c      /* RK SPI Slave implement of interface */
drivers/spi/spi-rockchip-slave.c      /* RK SPI Slave implement of interface */
drivers/spi/spidev.c            /* Create SPI device node for using */
drivers/spi/spi-rockchip-test.c /* SPI test driver, it needs to add to Makefile
compiler manually. */
Documentation/spi/spidev_test.c /* SPI test tool in user state */
```

2.2 SPI Device Configuration: RK SPI As Master Port

Kernel Configuration

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

DTS Node Configuration

```
&spi1 {                                 //Quote SPI controller node
    status = "okay";
    //assigned-clocks = <CLK_SPI1>;      //Not configured by default, depend on
soc dtsi
    //assigned-clock-rates = <200000000>;//Not configured by default, spi
controller work clock
    //dma-names;                        //Not configured by default, turn off
DMA support, only supports IRQ transmission
    //rockchip,poll-only;               //Not configured by default,  turn to
use CPU transmission, only master mode supported
    //rx-sample-delay-ns = <10>;        //Not configured by default, Read
sampling delay. Please refer to "FAQ" and "Delay sampling clock configuration"
for details
    //rockchip, autosuspend-delay-ms = <500>; //Not configured by default,
Runtime PM autosuspend delay, refer to "SPI Transfer Rate and CPU Usage
Optimization Directions" for details.
    //rockchip,rt;                      //Not configured by default,Place SPI
data transfer process into SCHED_FIFO, its priority is 50
    spi_test@10 {
        compatible ="rockchip,spi_test_bus1_cs0";   //The name corresponding to
the driver
        reg = <0>;                      //Chip select 0 or 1
```

```
        spi-cpha;                           //If configure it, cpha is 1
        spi-cpol;                           //If configure it,cpol is 1, the clk
pin remains high level.
        spi-lsb-first;                      //IO firstly transfer lsb
        status = "okay";                    //Enable device node
        spi-max-frequency = <24000000>;  //This is clock frequency of SPI clk
output,witch does not exceed 50M.
    };
};
```

Configuration instructions for spiclk assigned-clock-rates and spi-max-frequency:

- spi-max-frequency is the output clock of SPI. spi-max-frequency is output after internal frequency division of SPI working clock spiclk in assigned-clock-rates. Since there are at least 2 internal frequency divisions, the relationship is that SPI assigned clock rates > = 2 * SPI Max frequency;
- Assume that we want 50MHz SPI IO rate，the configuration can be set as: spiclk assigned-clock-rates = <100000000>，spi-max-frequency = <50000000>.
- spiclk assigned-clock-rates should not be lower than 24M, otherwise there may be problems.

2.3 SPI Device Configuration: RK SPI As Slave Port

Key Patch

Recommended to use the SPI slave source code spi-rockchip-slave.c, as the SDK version issue, please confirm that the SDK has the following patch:

```
commit 10cbf3c2c93fca6e5ec6c99b5bdb319ca0494d45
Author: Jon Lin <jon.lin@rock-chips.com>
Date:   Tue Nov 21 10:58:57 2023 +0800

    spi: rockchip-slave: Add code

    1.Implement one msg mechanism
    2.Support SRAM extension by dts rockchip,sram property

    Change-Id: I0fccc5d4347294488b5382ad3ba5ae72b35610f2
    Signed-Off-By: Jon Lin <jon.lin@rock-chips.com>
```

Instructions

- If no such patch, the customer can directly refer to Redmine FAE Project -> Document -> Development Configuration -> SPI Path.

Kernel Configuration

```
Device Drivers  --->
    [*] SPI support  --->
        [*]   SPI slave protocol handlers
        [*]   Rockchip SPI Slave controller driver
```

DTS Node Configuration

```
&spi1 {
```

```
    compatible = "rockchip,spi-slave";       //priority use SPI slave dedicated
driver
    status = "okay";
    //ready-gpios = <&gpio1 RK_PD2 GPIO_ACTIVE_LOW>;//recommend configuring, SPI
slave complete transmission flag, refer to "kernel SPI Slave Software" chapter
    //rockchip,cs-inactive-disable;                 //default do not configure,
when SPI master timing tod_cs (Clock Rise To CS Rise Time) is over multiple io
clock cycles, should open config to detect cs release action
    slave {                                         //according to framework
requirement, SPI slave sub node's name should start with "slave"
        compatible ="rockchip,spi_test_bus1_cs0";
        reg = <0>;                                  //only support single chip
select
        spi-cpha;                                   //set CPHA = 1, do not
configure otherwise
        spi-cpol;                                   //set CPOL = 1, do not
configure otherwise
        spi-lsb-first;                              //IO Input lsb first
        status = "okay";                            //enable device node
    };
};
```

Note:

- RK SPI enables DMA transmission by default, and slave mode does not recommend turning off DMA transmission. When a transfer exceeds the controller's cache count, the software will configure it for DMA transfer to avoid interrupting the corresponding transmission in a timely manner.

2.4 SPI Slave Notice

2.4.1 Suggest Performance Mode

When the master rate exceeds a certain frequency, it is recommended to set the transfer process in performance mode to avoid DRAM frequency conversion causing controller cache overflow:

- bits_per_word = 8btis, master io rate exceeds 5MHz
- bits_per_word = 16btis, master io rate exceeds 10MHz

Reference code:

```
diff --git a/drivers/spi/spi-rockchip-test.c b/drivers/spi/spi-rockchip-test.c
index 544d6038919a..c1037153ff86 100644
--- a/drivers/spi/spi-rockchip-test.c
+++ b/drivers/spi/spi-rockchip-test.c
@@ -36,6 +36,8 @@
 #include <linux/platform_data/spi-rockchip.h>
 #include <linux/uaccess.h>
 #include <linux/syscalls.h>
+#include <soc/rockchip/rockchip-system-status.h>
+#include <dt-bindings/soc/rockchip-system-status.h>

 #define MAX_SPI_DEV_NUM 10
 #define SPI_MAX_SPEED_HZ       12000000
@@ -242,8 +244,10 @@ static ssize_t spi_test_write(struct file *file,
                }
```

```
                start_time = ktime_get();
+               rockchip_set_system_status(SYS_STATUS_PERFORMANCE);
                for (i = 0; i < times; i++)
                        spi_read_slt(id, rxbuf, size);
+               rockchip_clear_system_status(SYS_STATUS_PERFORMANCE);
                end_time = ktime_get();
                cost_time = ktime_sub(end_time, start_time);
                us = ktime_to_us(cost_time);
```

Notes:

- It is recommended that all slave mode transfer behavior be run in performance mode.
- The set/clear performance interface has some time overhead, so it is recommended to set it at the business level to avoid frequent calls.
- If there is a cache overflow, the slave cannot complete DMA transmission and will block and exit, which can be confirmed by printing the SPI->SPI_RISR register.

2.4.2 Suggest 16bits Width

Make full use of slave fifo capacity to accelerate and minimize burst 2, which can accelerate the slave end DMA transmission rate and avoid fifo being stacked due to insufficient movement.

2.4.3 Other Notes

SPI Slave Test Notes

Spi is a slave, first start the slave read, then start the master write, otherwise the slave will not have read yet and the master has already written.

Slave write, master read also needs to start the slave write first, because only after the master sends clk, the slave will work, and the master will immediately send or receive data.

Example: On the basis of Chapter 3:

First slave: `echo write 0 1 16 > /dev/spi_misc_test`

Then master: `echo read 0 1 16 > /dev/spi_misc_test`

2.5 SPI Device Driver

Register device driver:

```
#include <linux/init.h>
#include <linux/module.h>
#include <linux/platform_device.h>
#include <linux/of.h>
#include <linux/spi/spi.h>

static int spi_test_probe(struct spi_device *spi)
{
    int ret;
```

```
    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 = {
        .name  = "spi_test",
        .owner = THIS_MODULE,
        .of_match_table = of_match_ptr(spi_test_dt_match),
    },
    .probe = spi_test_probe,
    .remove = spi_test_remove,
};

static int __init spi_test_init(void)
{
    int ret = 0;
    ret = spi_register_driver(&spi_test_driver);
    return ret;
}
module_init(spi_test_init);

static void __exit spi_test_exit(void)
{
    return spi_unregister_driver(&spi_test_driver);
}
module_exit(spi_test_exit);
```

For SPI read and write operations, please refer to `include/linux/spi/spi.h`.

```
static inline int
spi_write(struct spi_device *spi,const void *buf, size_t len)
static inline int
spi_read(struct spi_device *spi,void *buf, size_t len)
static inline int
spi_write_and_read(structspi_device *spi, const void *tx_buf, void *rx_buf,
size_t len)
```

2.6 User mode SPI device Configuration

User mode SPI device means operating the SPI interface in user space directly, which makes it convenient for many SPI peripheral drivers run in user space.

There is no need to change the kernel to facilitate driver development.

Kernel Configuration

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

DTS Configuration

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

Using Instruction

After the driver device is successfully registered, a device like this name will be displayed: /dev/spidev1.1

For the demo of spidev operation, please refer to:

- Kernel 4.4 Documentation/spi/spidev_test.c
- Kernel 4.19 and later tools/spi/spidev_test.c
- After the kernel project is compiled, enter the corresponding path and enter the following command to directly compile the standard SPI app program:

```
make CROSS_COMPILE=~/path-to-toolchain/gcc-xxxxx-toolchain/bin/xxxx-linux-gnu-
    # Choose kernel toolchain
```

- It supports the configuration of SPI slave devices. Refer to "SPI Device Configuration: RK SPI As Slave Port", in which the DTS configuration sub node should remain "rockchip, spidev"

2.7 Support cs-gpios

Users can use the cs-gpios attribute of spi-bus to implement gpio simulation cs to extend SPI chip selection signal. Users can refer to the kernel document `Documentation/devicetree/bindings/spi/spi-bus.txt` to learn more about cs-gpios.

2.7.1 Configuration of Linux 4.4

This support needs more support patches. Please contact RK Engineer for the corresponding patches.

2.7.2 Configuration of Linux 4.19 and above

Take spi1_cs2n in GPIO0_C4 for example:

Set the cs-gpio pin and reference it in the SPI node

```
diff --git a/arch/arm/boot/dts/rv1126-evb-v10.dtsi b/arch/arm/boot/dts/rv1126-
evb-v10.dtsi
index 144e9edf1831..c17ac362289e 100644
--- a/arch/arm/boot/dts/rv1126-evb-v10.dtsi
+++ b/arch/arm/boot/dts/rv1126-evb-v10.dtsi

&pinctrl {
        ...
+
+       spi1 {
+               spi1_cs0n: spi1-cs1n {
+                       rockchip,pins =
+                               <0 RK_PC2 RK_FUNC_GPIO
&pcfg_pull_up_drv_level_0>;
+               };
+               spi1_cs1n: spi1-cs1n {
+                       rockchip,pins =
+                               <0 RK_PC3 RK_FUNC_GPIO
&pcfg_pull_up_drv_level_0>;
+               };
+               spi1_cs2n: spi1-cs2n {
+                       rockchip,pins =
+                               <0 RK_PC4 RK_FUNC_GPIO
&pcfg_pull_up_drv_level_0>;
+               };
+       };
};

diff --git a/arch/arm/boot/dts/rv1126.dtsi b/arch/arm/boot/dts/rv1126.dtsi
index 351bc668ea42..986a85f13832 100644
--- a/arch/arm/boot/dts/rv1126.dtsi
+++ b/arch/arm/boot/dts/rv1126.dtsi

spi1: spi@ff5b0000 {
        compatible = "rockchip,rv1126-spi", "rockchip,rk3066-spi";
        reg = <0xff5b0000 0x1000>;
        interrupts = <GIC_SPI 11 IRQ_TYPE_LEVEL_HIGH>;
        #address-cells = <1>;
        #size-cells = <0>;
        clocks = <&cru CLK_SPI1>, <&cru PCLK_SPI1>;
```

```
        clock-names = "spiclk", "apb_pclk";
        dmas = <&dmac 3>, <&dmac 2>;
        dma-names = "tx", "rx";
        pinctrl-names = "default", "high_speed";
-       pinctrl-0 = <&spi1m0_clk &spi1m0_cs0n &spi1m0_cs1n &spi1m0_miso
&spi1m0_mosi>;
-       pinctrl-1 = <&spi1m0_clk_hs &spi1m0_cs0n &spi1m0_cs1n &spi1m0_miso_hs
&spi1m0_mosi_hs>;
+       pinctrl-0 = <&spi1m0_clk &spi1_cs0n &spi1_cs1n &spi1_cs2n &spi1m0_miso
&spi1m0_mosi>;
+       pinctrl-1 = <&spi1m0_clk_hs &spi1_cs0n &spi1_cs1n &spi1_cs2n
&spi1m0_miso_hs &spi1m0_mosi_hs>
        status = "disabled";
};
```

SPI node reassigns CS pin

```
+&spi1 {
+       status = "okay";
+       max-freq = <48000000>;
+       cs-gpios = <&gpio0 RK_PC2 GPIO_ACTIVE_LOW>, <&gpio0 RK_PC3
GPIO_ACTIVE_LOW>, <&gpio0 RK_PC4 GPIO_ACTIVE_LOW>;
        spi_test@0 {
                compatible = "rockchip,spi_test_bus1_cs0";
...
+       spi_test@2 {
+               compatible = "rockchip,spi_test_bus1_cs2";
+               id = <2>;
+               reg = <0x2>;
+               spi-cpha;
+               spi-cpol;
+               spi-lsb-first;
+               spi-max-frequency = <16000000>;
+       };
};
```

Note:

- If you want to extend cs with gpio, all cs should be converted to gpio function and supported by cs-gpios property.

3. SPI Testing Driver in Kernel

3.1 Code Path

drivers/spi/spi-rockchip-test.c

3.2 SPI Testing Device Configuration

Kernel Path

```
drivers/spi/Makefile
+obj-y                                  += spi-rockchip-test.o
```

DTS Configuraion

```
&spi0 {
    status = "okay";
    spi_test@0 {
        compatible = "rockchip,spi_test_bus0_cs0";
        id = <0>;       //This attribute is used to distinguish different SPI
slave devices in "spi-rockchip-test.c".
        reg = <0>;      //chip select  0:cs0  1:cs1
        spi-max-frequency = <24000000>;                 //spi output clock
    };
    spi_test@1 {
        compatible = "rockchip,spi_test_bus0_cs1";
        id = <1>;
        reg = <1>;
        spi-max-frequency = <24000000>;
    };
};
```

Driver log

```
[    0.457137]
rockchip_spi_test_probe:name=spi_test_bus0_cs0,bus_num=0,cs=0,mode=11,speed=1600
0000
[    0.457308]
rockchip_spi_test_probe:name=spi_test_bus0_cs1,bus_num=0,cs=1,mode=11,speed=1600
0000
```

3.3 Test Command

```
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
```

The above means:

Echo type id number of loops transfer length > /dev/spi_misc_test

Echo setspeed id frequency (in Hz) > /dev/spi_misc_test

You can modify the test case by yourself if you want.

4. SPI Slave Software In Kernel

4.1 Introduction

Background

The transmission between SPI masters and slaves usually follows specific protocols, such as SPI No compatible with JEDEC SDFP protocol, and RK SPI slave, as a device side transmission, should also follow specific protocols. As the protocol has no paradigm, RK provides customized transmission protocols and device drivers for customer reference.

Linux SPI slave driver framework limitations:

- Using a transmission queue, although the thread priority after queue wake-up is higher, real-time performance cannot be fully guaranteed due to scheduling constraints

RK SPI slave mode restrictions:

- Each transmission requires a restart of the SPI controller configuration. Therefore, to ensure that the SPI master can know that the RK SPI slave has completed the transmission configuration and initiated data transmission, the RK SPI slave end needs to add a side band signal as a ready status bit

Transport Protocol

RK SPI slave transmission protocol:

- The RK SPI slave transmission requires specifying ready-gpios to notify the SPI master. The basic process is as follows:

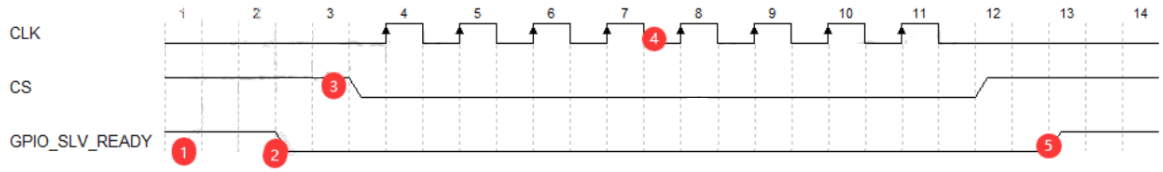
  step1：slave start spi_sync

  step2：slave ready, output GPIO_SLV_READY signal

  step3：master confirm slave ready, then begin to transfer

  step4：The slave receives sufficient clks from the master to complete the transmission

  step5：slave idle，release GPIO_SLV_READY signal



- Define two types of packages:
  Ctrl packet: 2B cmd, 2B addr (application buffer offset address defined by RK slave), 4B data (usually used to specify the transmission length of subsequent data packets)

  data packet

- Define two types of transmission:
  Ctrl transmission, only containing 1 ctrl packet

  Data transfer, including two SPI transfers of 1 ctrl packet and 1 data packet

- spidev_rkslv support SPI_OBJ_APP_RAM_SIZE Bytes application buffer for transmission buffer， The data transmission 1 ctrl packet 2B addr initiated by the SPI master points to the cache offset address

Device Driver

Key Patch:

```
commit d2fef34977c1a7aab3837d29ac8dc3b5378a2754 (HEAD -> develop-4.19)
Author: Jon Lin <jon.lin@rock-chips.com>
Date:   Wed Dec 20 12:02:14 2023 +0800

    spi: spidev_rkslv: Support dynamic adjustment of system performance
```

```
    If the DRAM frequency conversion jitters during the transmission process,
    it will cause the DMA to be unable to transport SPI FIFO data in a timely
    manner, resulting in FIFO overflow/underflow.

    Clear performance status for short cmd packet and Set the performance
    status for data packet.

    Change-Id: I65532ba309677a8d98c8277875a3bd358ca44e44
    Signed-off-by: Jon Lin <jon.lin@rock-chips.com>
```

Note:

- If no such patch, the customer can directly refer to Redmine FAE Project -> Document -> Development Configuration -> SPI Path.

Driver source code:

```
drivers/spi/spidev-rkslv.c
drivers/spi/spidev-rkmst.c
```

Source code introduction:

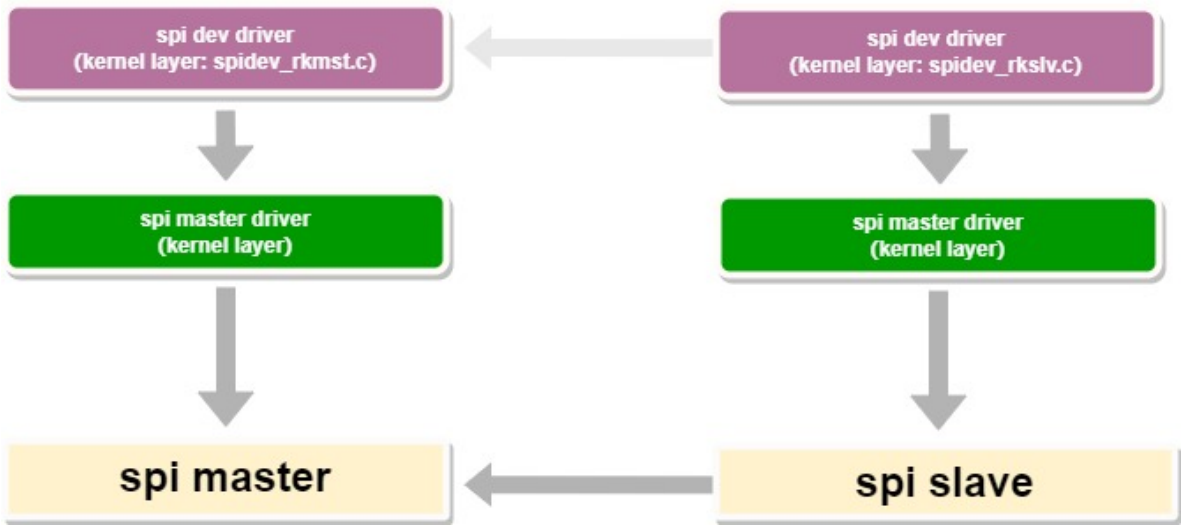
drivers/spi/spidev-rkslv.c：

```
static int spidev_rkslv_ctrl_receiver_thread(void *p)
//Establish a thread and repeatedly initiate transfers within the thread
{
    while (1)
        spidev_rkslv_xfer(spidev);
}

static int spidev_rkslv_xfer(struct spidev_rkslv_data *spidev)
 //Transmission entrance
{
    spidev_slv_read(spidev, spidev->ctrlbuf, SPI_OBJ_CTRL_MSG_SIZE);    //1 ctrl
packet to obtain and parse the transmission type
    switch (ctrl->cmd) {                                                //1 data
packet, define data packets based on the transmission type and complete sending
and receiving
    case SPI_OBJ_CTRL_CMD_INIT:
            /* to-do */
    case SPI_OBJ_CTRL_CMD_READ:
            /* to-do */
    case SPI_OBJ_CTRL_CMD_WRITE:
            /* to-do */
    case SPI_OBJ_CTRL_CMD_DUPLEX:
            /* to-do */
    }
}

static const struct file_operations spidev_rkslv_misc_fops = {}
 //Register misc device test interface
```

drivers/spi/spidev-rkmst.c：

```
static int spidev_rkmst_xfer(struct spidev_rkmst_data *spidev, void *tx, void
*rx, u16 addr, u32 len)  //Transmission entrance
{
    spidev_rkmst_ctrl(spidev, cmd, addr, len);                          //1
ctrl packet, defining the transmission type
    switch (cmd) {                                                      //1
data packet, define data packets based on the transmission type and complete
sending and receiving
    case SPI_OBJ_CTRL_CMD_READ:
            /* to-do */
    case SPI_OBJ_CTRL_CMD_WRITE:
            /* to-do */
    case SPI_OBJ_CTRL_CMD_DUPLEX:
            /* to-do */
    }
}

static const struct file_operations spidev_rkmst_misc_fops = {}
 //Register misc device test interface
```

Implement business



The purpose of providing "SPI Slave Software In Kernel" is to provide reference for protocol and device drivers, and the end customer should also define their product requirements on the application buffer of the slave end to achieve business.

4.2 SPI Slave Testing Device Configuration

Defconfig configuration:

```
CONFIG_SPI_SLAVE_ROCKCHIP_OBJ=y
```

RK SPI slave end dts reference configuration:

```
&spi1 {
    status = "okay";
    spi-slave;
```

```
    rockchip,cs-inactive-disable;                               //RK internal
interconnection using RK Linux SPI master driver, and the tod_cs takes a long
time
    ready-gpios = <&gpio1 RK_PD3 GPIO_ACTIVE_LOW>;              //Please set to
the actual GPIO used
    slave {
        compatible = "rockchip,spi-obj-slave";
        reg = <0x0>;
        spi-cpha;
        spi-cpol;
        spi-lsb-first;
        spi-max-frequency = <50000000>;
    };
};
```

RK SPI master dts reference configuration:

```
&spi0 {
    status = "okay";
    spi_test@00 {
        compatible = "rockchip,spi-obj-master";
        reg = <0x0>;
        spi-cpha;
        spi-cpol;
        spi-lsb-first;
        spi-max-frequency = <16000000>;
        ready-gpios = <&gpio1 RK_PD2 GPIO_ACTIVE_LOW>;
    };
};
```

4.3 Test Command

SPI master initiates single packet data transmission testing

```
echo cmd addr length > /dev/spidev_rkmst_misc
```

Note:

- cmd: Support read/write/duplex
- addr: Refers to the offset of the peer slave application buffer, in Bytes, which only supports decimal input
- length: The length of the data packet, in Bytes, only supports decimal input
- The example is as follows:

```
echo write 128 128 > /dev/spidev_rkmst_misc
echo read 128 128 > /dev/spidev_rkmst_misc
echo duplex 128 128 > /dev/spidev_rkmst_misc
```

SPI master initiates Test automation

```
echo autotest length loops compare > /dev/spidev_rkmst_misc
```

Note:

- autotest: Fixed input, test full duplex data transmission first, then test read and write data transmission, and output rate results
- The test defaults to using the opposite slave application buffer offset address 0
- length: The length of the data packet, in Bytes, only supports decimal input
- loops: Set the number of pressure testing cycles
- compare: 1 - enable verification, 0 - disable verification(support specific scenarios, such as continuous output data for signal testing)
- The strength is as follows:

```
echo autotest 1024 64 1 > /dev/spidev_rkmst_misc
```

SPI slave testing

```
echo appmem 0 256 > ./dev/spidev_rkslv_misc      #Print application buffer data
echo verbose 1 > ./dev/spidev_rkslv_misc         #Enable the transmission
process debug log, echo verbose 0, and close printing
```

5. FAQ

5.1 SPI no signal

- Confirm that the driver is running before debugging
- Ensure that the IOMUX configuration of the SPI 4 pins is correct .
- Confirm that during the TX sending, the TX pin has a normal waveform, CLK has a normal CLOCK signal, and the CS signal is pulled low.
- If the clock frequency is high, considering increasing the drive strength to improve the signal.
- How to simply judge whether SPI DMA is enabled or not? If the serial port printing does not have the following keywords, DMA is enabled successfully:

```
[    0.457137] Failed to request TX DMA channel
[    0.457237] Failed to request RX DMA channel
```

5.2 How to design application code in SPI

Please select the appropriate object function interface before writing the driver.

Custom SPI device driver

Refer to "SPI Device Driver", for example: drivers/spi/spi-rockchip-test.c.

Application program based on spidev standard device node

Refer to "User mode SPI device Configuration"

5.3 Delay sampling clock configuration

In the case of high SPI IO rate, the normal SPI mode may still not match the output delay of external devices, and RK SPI master read may not be able to sample valid data. SPI RSD logic needs to be enabled to delay the sampling clock.

RK SPI RSD (read sample delay) control logic has the following characteristics:

- The assignable values are 0, 1, 2, 3
- The delay unit is 1 spi_clk cycle, i.e. the working clock of the controller, see "SPI Device Configuration" for details

rx-sample-delay actual delay is the RSD effective value closest to the DTS set value, subject to spi_clk 200MHz, cycle 5ns, for example:

The actual configurable delay of RSD is 0, 5ns, 10ns and 15ns. RX sample delay is set to 12ns, which is close to the effective value of 10ns, so the final delay is 10ns.

5.4 SPI transmission method description

Default Transfer Mode

Master mode supports IRQ, DMA, and CPU transfers, while slave mode supports IRQ and DMA transfers, with the default combination of IRQ/DMA transfer modes:

- When the transfer length is less than the FIFO depth, IRQ transfer is used. By default, for SOCs using kernel versions 4.19 and above, the FIFO depth is 64.
- When the transfer length is greater than or equal to the FIFO depth, DMA transfer is used.

Modifying Transfer Mode

Master mode supports:

- The default combination of IRQ/DMA transfer mode.
- Refer to the instructions "Disable DMA support, only support IRQ transfer" to disable DMA, after which only IRQ transfer is supported.
- Refer to the instructions for "rockchip,poll-only" to configure it to only support CPU transfer.

Slave mode does not support modifying the transfer mode.

IRQ Transfer Characteristics

- When the data is less than the FIFO depth, a single transfer triggers 1 interrupt.
- When the data is greater than or equal to the FIFO depth and using IRQ transfer, the FIFO watermark is set to half the FIFO, typically 32 items, and a transfer roughly triggers items / 32 interrupts.

DMA Transfer Characteristics

- Does not trigger an interrupt for the SPI controller, using DMA transfer finished call back callback.

5.5 SPI Transfer Rate and CPU Usage Optimization Directions

The reasons for slow SPI transfer rates and high CPU usage under heavy IO load are usually related to small transfer granularity and frequent transfer requests, which involve the following:

- SPI thread scheduling
- Interrupt scheduling (refer to the "SPI Transfer Mode Explanation" section to confirm if interrupt transfers are used)

- CPU idle scheduling

Recommended optimization directions:

1. Enable auto runtime suspend with a delay of 500ms (specific value may vary based on actual testing). Modify the DTS node by adding the `rockchip,autosuspend-delay-ms` property.
2. Reduce CPU load: Switch to IRQ-based transfers, which may have advantages over DMA. Refer to the "Switch to IRQ Transfers" subsection for patch reference.
3. Reduce CPU load: If using DMA transfers, modify the TX DMA waterline to reduce the time CPU spends waiting for FIFO transfers to complete in the DMA callback function. Refer to the "Modify SPI Waterline" patch.

Patch references:

Switch to IRQ Transfers

```
diff复制代码diff --git a/arch/arm/boot/dts/rv1126-evb-v10.dtsi
b/arch/arm/boot/dts/rv1126-evb-v10.dtsi
index 86dd23482d97..2cea93d2423f 100644
--- a/arch/arm/boot/dts/rv1126-evb-v10.dtsi
+++ b/arch/arm/boot/dts/rv1126-evb-v10.dtsi
@@ -1367,6 +1367,7 @@
        status = "okay";
        max-freq = <48000000>;
        cs-gpios = <0>, <0>, <&gpio0 RK_PC4 GPIO_ACTIVE_LOW>;
+       dma-names;
        spi_test@00 {
```

Modify SPI Waterline

```
diff复制代码diff --git a/drivers/spi/spi-rockchip.c b/drivers/spi/spi-rockchip.c
index 27fd6f671b12..bd0fa8c5f8c3 100644
--- a/drivers/spi/spi-rockchip.c
+++ b/drivers/spi/spi-rockchip.c
@@ -616,7 +616,8 @@ static void rockchip_spi_config(struct rockchip_spi *rs,
        else
                writel_relaxed(rs->fifo_len / 2 - 1, rs->regs +
ROCKCHIP_SPI_RXFTLR);

-       writel_relaxed(rs->fifo_len / 2 - 1, rs->regs + ROCKCHIP_SPI_DMATDLR);
+       // writel_relaxed(rs->fifo_len / 2 - 1, rs->regs +
ROCKCHIP_SPI_DMATDLR);
+       writel_relaxed(11, rs->regs + ROCKCHIP_SPI_DMATDLR);
        writel_relaxed(rockchip_spi_calc_burst_size(xfer->len / rs->n_bytes) -
1,
                       rs->regs + ROCKCHIP_SPI_DMARDLR);
        writel_relaxed(dmacr, rs->regs + ROCKCHIP_SPI_DMACR);
```

Note: The provided patch references are specific to the Rockchip SPI driver and RV1126-EVB-V10 DTS file. Adjustments may be needed based on your specific hardware and software configurations.