EC2X&-QuecOpen SPI Guidelines

LTE Module Series

Rev. EC2X-QuecOpen_SPI_Guidelines

Date: 2018-03-09

Status: Preliminary

www.quectel.com

Our aim is to provide customers with timely and comprehensive service. For any assistance, please contact our company headquarters:

Quectel Wireless Solutions Co., Ltd.
7th Floor, Hongye Building, No.1801 Hongmei Road, Xuhui District, Shanghai 200233, China
Tel: +86 21 5108 6236
Email: info@quectel.com

Or our local office. For more information, please visit:
http://quectel.com/support/sales.htm

For technical support, or to report documentation errors, please visit:
http://quectel.com/support/technical.htm
Or email to: support@quectel.com

GENERAL NOTES

COPYRIGHT

About the Document

History

Revision	Date	Author	Description
1.0	2018-03-09	Gale	Initial

Contents

Table Index

Figure Index

1 Introduction

From the perspective of user development, this document introduces circuit design, software drive layer, software application layer, etc. It can help customers develop easily and quickly.

2 EC20 R2.1-QuecOpen SPI Introdcution

(1) The module provides one SPI interface by default, only supports the main mode, and supports DMA by default;
(2) The maximum clock frequency supported is 50MHz;
(3) Each SPI controller supports up to four chip selection signals (CS);
(4) The following figure shows the SPI framework of EC20 R2.1-Quecopen module.

In which, The SPI adapter layer is the device tree configuration and driver of SPI controller.

SPI slave provides SPI device driver for QuecOpen, which is divided into standard 4-line and extended 6-line.

4-line: Usually used to connect SPI flash, LCD, etc., requests are initiated by the module.

6-line: Usually used to communicate with MCU, modules, MCU can all initiate requests, compared with serial communication is also faster.

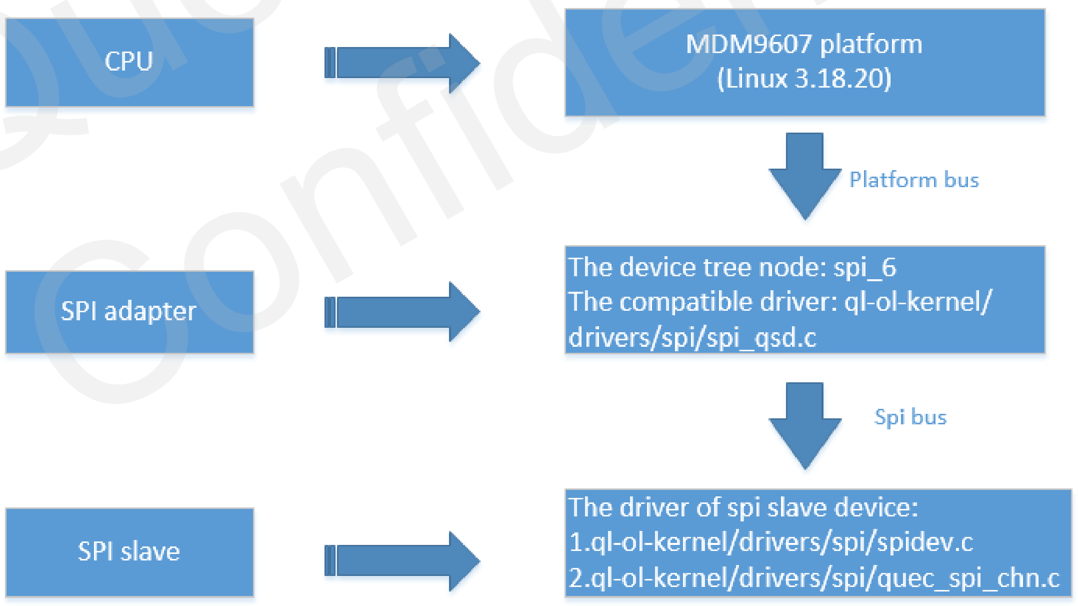

CPU
MDM9607 platform
(Linux 3.18.20)
Platform bus
SPI adapter
The device tree node: spi_6
The compatible driver: ql-ol-kernel/drivers/spi/spi_qsd.c
Spi bus
SPI slave
The driver of spi slave device:
1.ql-ol-kernel/drivers/spi/spidev.c
2.ql-ol-kernel/drivers/spi/quec_spi_chn.c

Figure 1: SPI framework of EC20 R2.1-Quecopen module

3 Recommended Hardware Circuit Design

3.1. Standard 4-line SPI External Flash Reference Design

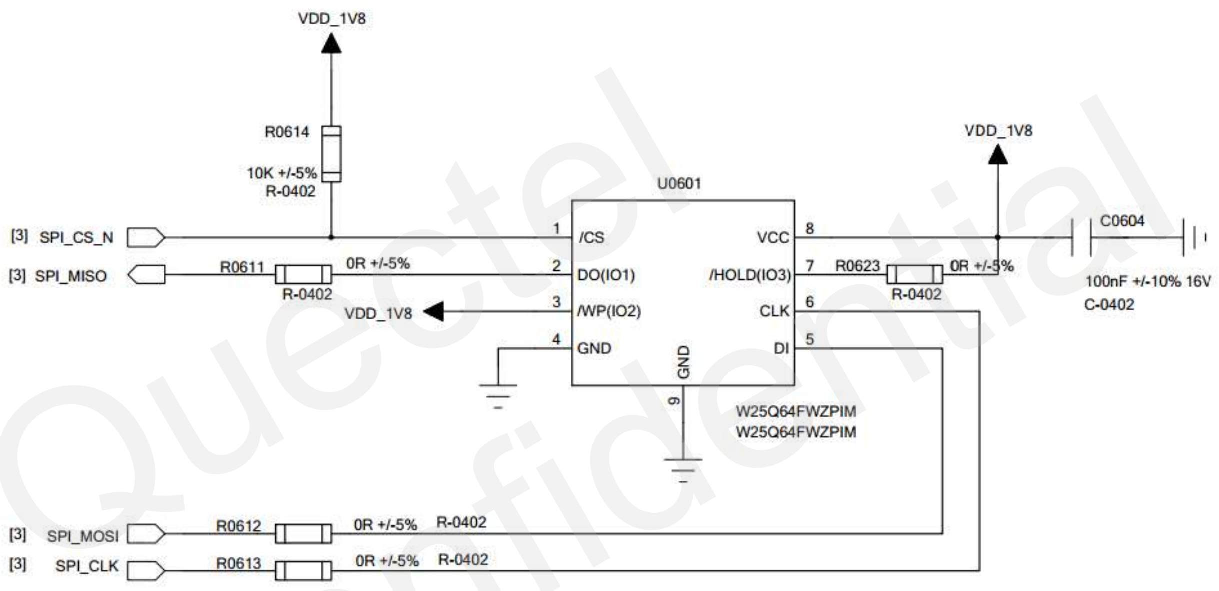


Figure 2: Referenced Design of Standard 4-line SPI External Flash

3.2. Extended line-6 SPI External MCU Circuit Reference Design

The following figure shows the SPI external 1.8V MCU circuit reference design.
If the MCU is on 3.3v, the level conversion chip shall be added.

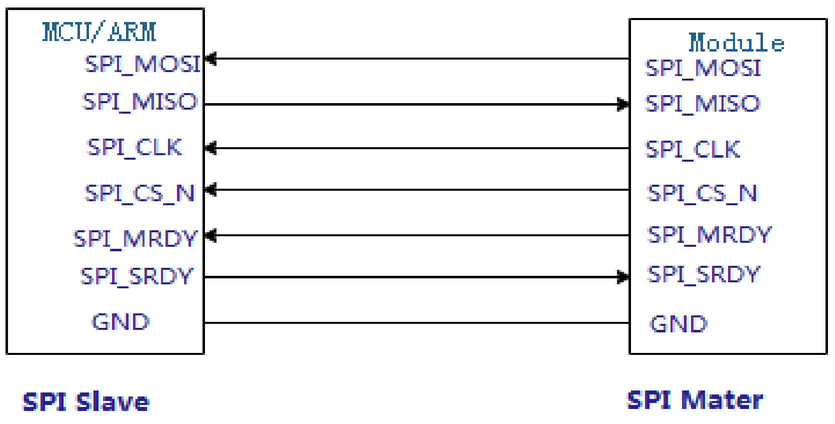


Figure 3: Extended line-6 SPI External MCU Circuit Reference Design

4 Driver Layer and Device Tree Software Adaptation

4.1. SPI Pin

(1) In the table below, the non-default multiplexing feature needs to be effective only after the software configuration, please refer to the corresponding feature chapter for software configuration.
(2) For the specific use of the pin please refer to the following document:
Quectel_EC20 R2.1_QuecOpen_GPIO_Assignment_Speadsheet

Table 1: Pin Function Multiplexing

Pin Name	Pin Number	Mode 1 (Default)	Mode 2	Mode 3	Reset State[1)]	Interrupt Wakeup[2)]	Remark
SPI_CS_N	37	SPI_CS_N_BLSP6	GPIO_22	UART_RTS_BLSP6	B-PD,L	YES	
SPI_MOSI	38	SPI_MOSI_BLSP6	GPIO_20	UART_TXD_BLSP6	B-PD,L	YES	
SPI_MISO	39	SPI_MISO_BLSP6	GPIO_21	UART_RXD_BLSP6	B-PD,L	YES	
SPI_CLK	40	SPI_CLK_BLSP6	GPIO_23	UART_CTS_BLSP6	B-PU,H	NO	BOOT_CONFIG_4

4.1.1. Standard 4-line SPI Pin Use

Table 2: Standard 4-line SPI Pin Use

Pin Name	Pin Number	I/O	Description	Remark
SPI_CS_N	37	DO	SPI chip selection signal	1.8V power domains, if not used, it is suspended.
SPI_MOSI	38	DO	SPI data output	1.8V power domains, if not used, it is suspended.

SPI_MISO	39	DI	SPI data Input	1.8V power domains, if not used, it is suspended.
SPI_CLK	40	DO	SPI Clock	1.8V power domains, if not used, it is suspended.

4.1.2. Extended 6-line SPI Pin Use

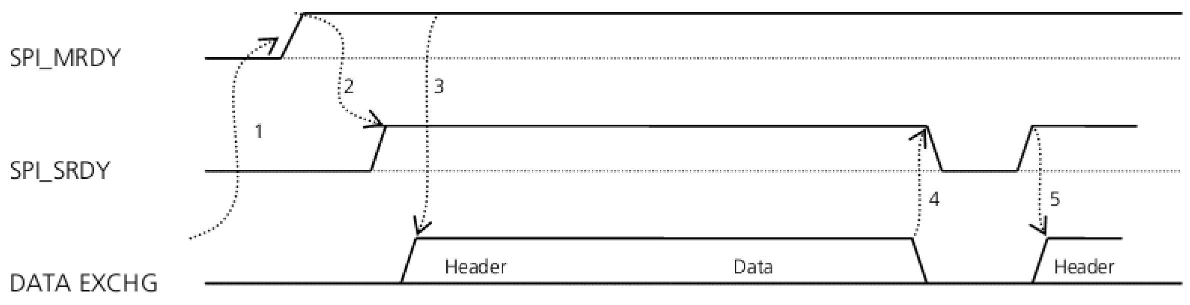
4.1.2.1. The Use of Pin

Table 3: Extended 6-line SPI Pin Use

Pin Name	Pin Number	I/O	Description	Remark
SPI_CS_N	37	DO	SPI chip selection signal	1.8V power domains, if not used, it is suspended.
SPI_MOSI	38	DO	SPI data Output	1.8V power domains, if not used, it is suspended.
SPI_MISO	39	DI	SPI data Input	1.8V power domains, if not used, it is suspended.
SPI_CLK	40	DO	SPI Clock	1.8V power domains, if not used, it is suspended.
SPI_MRDY	Selected by Users	DO	Module output signal, when idle the level is low. If the module wants to output data, the driver will automatically pull high this PIN.	
SPI_SRDY	Selected by Users	DI	SPI Slave ready signal, when idle the level is low. If SPI Slave is ready to receive/send data, pulls high this PIN.	

4.1.2.2. The Process by which the 4G Module and the slave device Initiate the Request Separately

4G Module Initiates a Request:

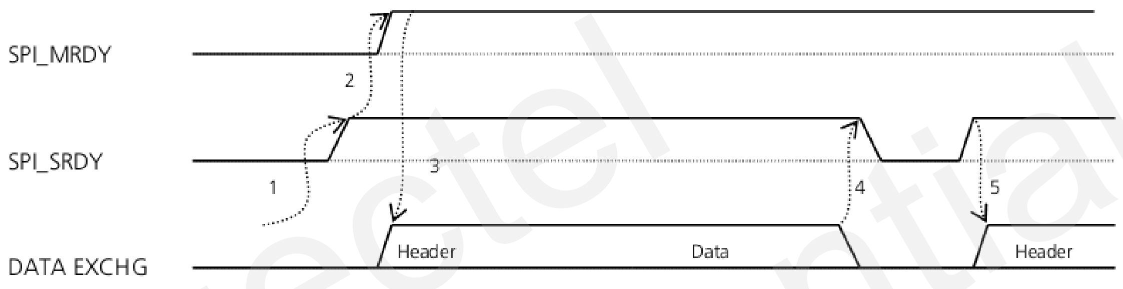


The Process for 4G Module:

(1) Driver auto pulls high SPI_MRDY to inform SPI Slave.
(2) Judge if SPI_SRDY is high-level, if not, wait for the SPI_SRDY "rising" edge interrupt.
(3) Receive slave “rising” edge, start SPI transmission.
(4) After the transmission is completed, if continuing sending data is required, keep SPI_MRDY high and back to the Step (2), otherwise pull down SPI_MRDY.

The process for SPI Slave:
(1) When receive SPI_MRDY "rising" edge interrupt, indicating that the 4G module needs to send data.
(2) Get ready for SPI transport and pull up the SPI_SRDY to notify 4G modules to start SPI transmission.
(3) Wait for SPI transmission to end and pull down SPI_SRDY.
(4) If SPI_MRDY is high-level, back to Step (2).

SPI Slave Initiates a Request:



The process for SPI Slave:
(1) Get ready for SPI transport and pull up SPI_SRDY.
(2) Wait for SPI transmission to end up and pull down SPI_SRDY
(3) If continuing sending data is required, back to step 1.

The process for 4G module:
(1) When receive SPI_SRDY "rising" edge interrupt, indicating that the slave device needs to send data.
(2) Pull high SPI_MRDY and start SPI transmission.
(3) Wait for SPI transmission to end and pull down SPI_MRDY

4.2. SPI Software Configuration Method

4.2.1. SPI Controller Configuration Introduction

The SPI architecture of Linux is divided into three parts:
(1) SPI kernel: SPI kernel provides registration of SPI bus driver and device driver, cancellation method, SPI communication method, code irrelevant to specific controller, detection device, upper code of detection device address, etc.
(2) SPI bus (controller) driver: it is the implementation of SPI hardware system controller, which is controlled by CPU and can be directly integrated into CPU.
(3) SPI device driver: that is, the customer's SPI slave device driver, is the implementation of the device

side in the SPI hardware architecture. The device is typically attached to the CPU-controlled SPI controller and exchanges data with the CPU through the SPI controller.

NOTE

Users generally only need to care about and modify the SPI device driver, will introduce more details in Chapter 4.2.2

SPI bus driver: that is SPI controller, the device tree node on mdm9607 platform is USES SPI -qup-v2. For its hardware parameter configuration, such as compatible driver, pin selection, register address, CLK, interrupt number, as well as the system sleep and working pin configuration, etc, QuecOpen has been completed, users do not need to care and modify.

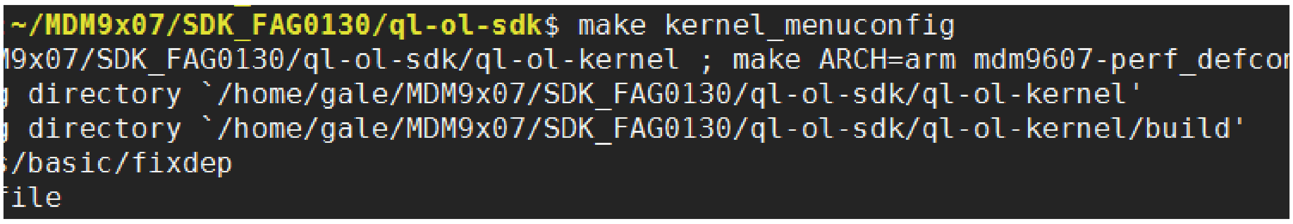
```
	spi_6: spi@78ba000 {
		compatible = "qcom,spi-qup-v2";
		#address-cells = <1>;
		#size-cells = <0>;
		reg-names = "spi_physical", "spi_bam_physical";
		reg = <0x78ba000 0x600>,
		      <0x7884000 0x2b000>;
		interrupt-names = "spi_irq", "spi_bam_irq";
		interrupts = <0 100 0>, <0 238 0>;
		spi-max-frequency = <19200000>;
		pinctrl-names = "spi_default", "spi_sleep";
		pinctrl-0 = <&spi6_default &spi6_cs0_active>;
		pinctrl-1 = <&spi6_sleep &spi6_cs0_sleep>;
		clocks = <&clock_gcc clk_gcc_blsp1_ahb_clk>,
			 <&clock_gcc clk_gcc_blsp1_qup6_spi_apps_clk>;
		clock-names = "iface_clk", "core_clk";
		qcom,infinite-mode = <0>;
		qcom,use-bam;
		qcom,use-pinctrl;
		qcom,ver-reg-exists;
		qcom,bam-consumer-pipe-index = <22>;
		qcom,bam-producer-pipe-index = <23>;
		qcom,master-id = <86>;
```

In addition, unless the user does not use the SPI controller at all on the mdm9607 platform, the user can turn off the SPI controller in the following ways. The following method performs at least one.

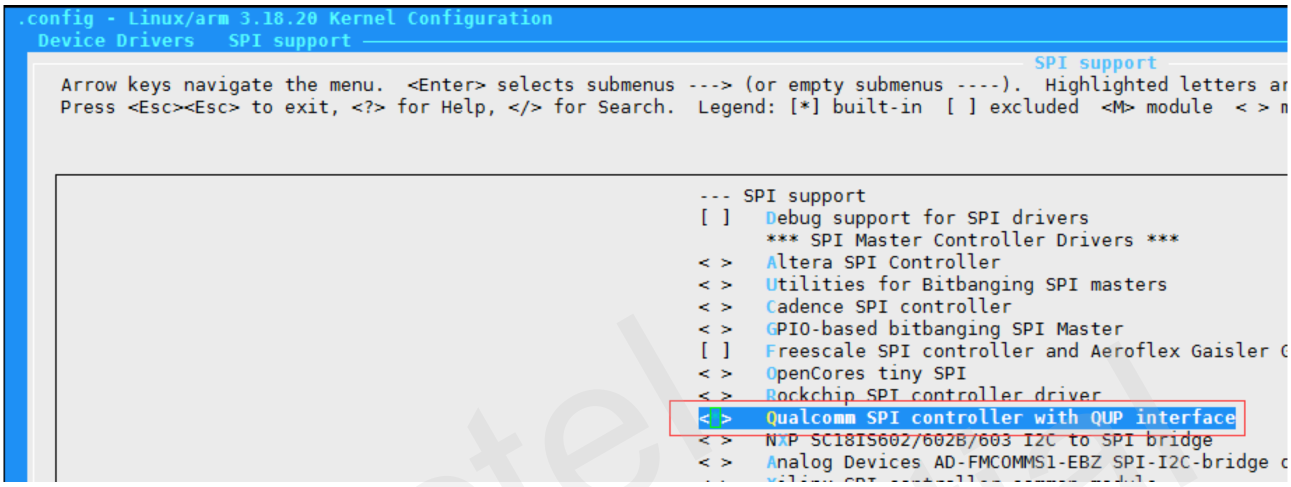
- Turn off controller device node.

```
--- a/ql-ol-kernel/arch/arm/boot/dts/qcom/mdm9607-mtp.dtsi
+++ b/ql-ol-kernel/arch/arm/boot/dts/qcom/mdm9607-mtp.dtsi
@@ -48,7 +48,7 @@
 //2016-01-19, comment out by jun.wu, remove UART3 && spi_1 from device tree

 &spi_6 {
-	status = "ok";
+	status = "disabled";
 };
```

- make kernel_menuconfig Delete SPI_QUP kernel option



```
~/MDM9x07/SDK_FAG0130/ql-ol-sdk$ make kernel_menuconfig
19x07/SDK_FAG0130/ql-ol-sdk/ql-ol-kernel ; make ARCH=arm mdm9607-perf_defcor
g directory `/home/gale/MDM9x07/SDK_FAG0130/ql-ol-sdk/ql-ol-kernel'
g directory `/home/gale/MDM9x07/SDK_FAG0130/ql-ol-sdk/ql-ol-kernel/build'
;/basic/fixdep
File
```



```
.config - Linux/arm 3.18.20 Kernel Configuration
 Device Drivers    SPI support
                                                                  SPI support
   Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus ----).  Highlighted letters ar
   Press <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < > m

                                            --- SPI support
                                            [ ]   Debug support for SPI drivers
                                                  *** SPI Master Controller Drivers ***
                                            < >   Altera SPI Controller
                                            < >   Utilities for Bitbanging SPI masters
                                            < >   Cadence SPI controller
                                            < >   GPIO-based bitbanging SPI Master
                                            [ ]   Freescale SPI controller and Aeroflex Gaisler C
                                            < >   OpenCores tiny SPI
                                            < >   Rockchip SPI controller driver
                                            < >   Qualcomm SPI controller with QUP interface
                                            < >   NXP SC18IS602/602B/603 I2C to SPI bridge
                                            < >   Analog Devices AD-FMCOMMS1-EBZ SPI-I2C-bridge c
```

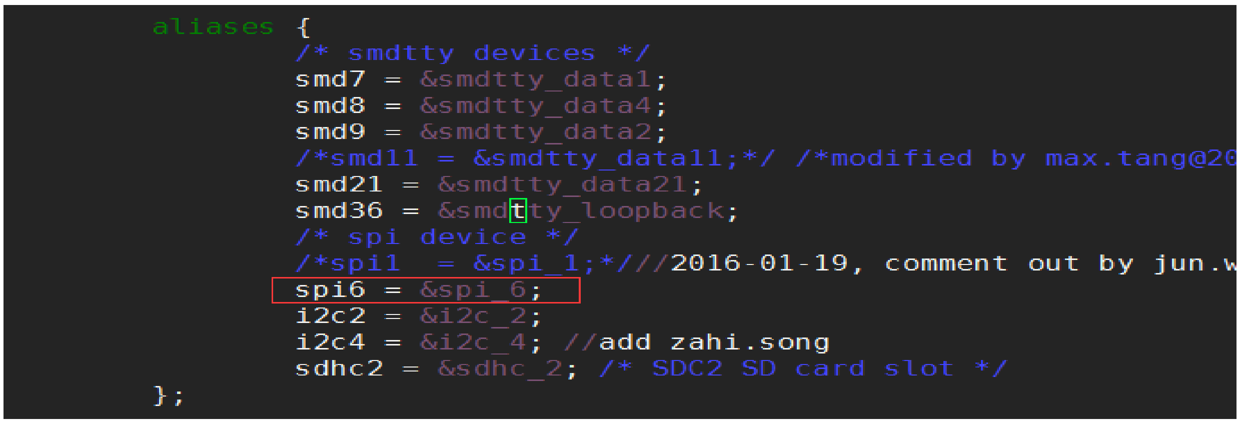
4.2.2. The Use of SPI Device Driver

QuecOpen module offers two SPI device drivers, divided into 4-line and 6-line, the default to the kernel module way to compile and stored in rootfs/usr/lib/modules / 3.18.20 / kernel/drivers/spi/directory, the user needs to insmod.

4.2.2.1. The Introduction of Standard 4-Line SPI Device Driver

4-line: The driver is located in ql-ol-kernel/drivers/spi/spidev.c, which is usually used to connect to spi flash, lcd, etc., and the module initiates a request. This driver does not use the device tree to pass the parameter, directly insmod is more flexible.

Parameters supported when the kernel module is loaded.
Busnum: Spi controller number is 6, this parameter must be passed in, otherwise the SPI slave device will not find the controller and the load will fail.



```
        aliases {
                /* smdtty devices */
                smd7 = &smdtty_data1;
                smd8 = &smdtty_data4;
                smd9 = &smdtty_data2;
                /*smd11 = &smdtty_data11;*/ /*modified by max.tang@20
                smd21 = &smdtty_data21;
                smd36 = &smdtty_loopback;
                /* spi device */
                /*spi1  = &spi_1;*///2016-01-19, comment out by jun.w
                spi6 = &spi_6;
                i2c2 = &i2c_2;
                i2c4 = &i2c_4; //add zahi.song
                sdhc2 = &sdhc_2; /* SDC2 SD card slot */
        };
```

chipselect: Chip selection supports 0, 1, 2, 3, this parameter must be passed, otherwise the SPI device will fail to register.

spimode: Four working modes are supported, values are the bitwise-or of clock phase (CPHA 0x01) and clock polarity (CPOL 0x02), the driver code defaults to SPI_MODE_3 mode, which users can modify when insmod.

Clock polarity (CPOL): that is, when SPI is idle, the level of clock signal SCLK. (1: when idle, is high-level; 0: when idle, is low-high)

Clock phase (CPHA): that is, SPI starts sampling at which edge of SCLK. (1: Start from the first edge; 0: Start from the second edge.)

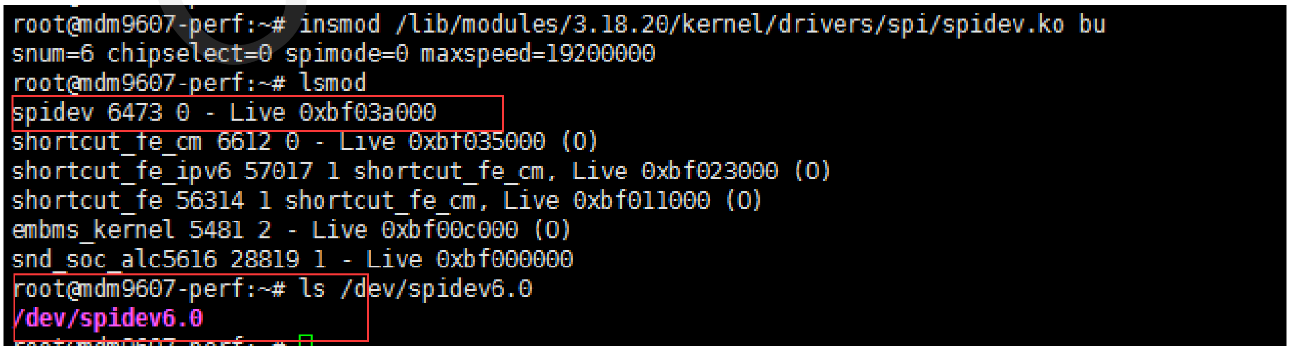
maxspeed: Optional parameter, driver is 9.6Mhz by default, The actual maximum supported is determined by spi controller configuration and does not conflict with the theoretical maximum. Optional values are supported: 960000, 4800000, 9600000, 16000000, 19200000, 25000000, 50000000.

bufsiz: Optional parameter, to set the size of each transfer in the spi transfer queue, default is 4096Bytes, users can set it according to the size of each data transfer.

Load Command

```
insmod /lib/modules/3.18.20/kernel/drivers/spi/spidev.ko busnum=6 chipselect=0 spimode=0 maxspeed=19200000
```

Successful confirmation:



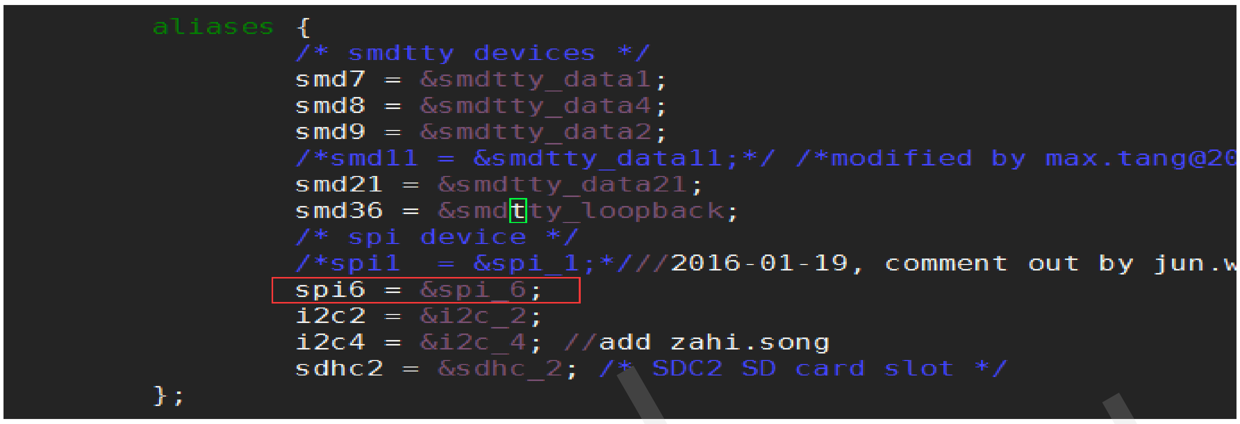
```
root@mdm9607-perf:~# insmod /lib/modules/3.18.20/kernel/drivers/spi/spidev.ko bu
snum=6 chipselect=0 spimode=0 maxspeed=19200000
root@mdm9607-perf:~# lsmod
spidev 6473 0 - Live 0xbf03a000
shortcut_fe_cm 6612 0 - Live 0xbf035000 (O)
shortcut_fe_ipv6 57017 1 shortcut_fe_cm, Live 0xbf023000 (O)
shortcut_fe 56314 1 shortcut_fe_cm, Live 0xbf011000 (O)
embms_kernel 5481 2 - Live 0xbf00c000 (O)
snd_soc_alc5616 28819 1 - Live 0xbf000000
root@mdm9607-perf:~# ls /dev/spidev6.0
/dev/spidev6.0
```

4.2.2.2. The Introduction of Extended 6-Line SPI Device Driver

6-line: The driver is located in ql-ol-kernel/drivers/spi/quec_chn_spi.c, which is usually used to connect to MCU, and both the module and the MCU can initiate a request. This driver does not use the device tree to pass the parameter, directly insmod is more flexible.

Parameters supported when the kernel module is loaded.

busnum: Spi controller number is 6, determined by the configuration shown below. It's the optional parameter and default value is 6.



```
        aliases {
                /* smdtty devices */
                smd7 = &smdtty_data1;
                smd8 = &smdtty_data4;
                smd9 = &smdtty_data2;
                /*smd11 = &smdtty_data11;*/ /*modified by max.tang@20
                smd21 = &smdtty_data21;
                smd36 = &smdtty_loopback;
                /* spi device */
                /*spi1  = &spi_1;*///2016-01-19, comment out by jun.w
                spi6 = &spi_6;
                i2c2 = &i2c_2;
                i2c4 = &i2c_4; //add zahi.song
                sdhc2 = &sdhc_2; /* SDC2 SD card slot */
        };
```

chipselect: Chip selection supports 0, 1, 2, 3, it's the optional parameter and default value is 0.

spi_mode: Four working modes are supported, values are the bitwise-or of clock phase (CPHA 0x01) and clock polarity (CPOL 0x02), the driver code defaults to SPI_MODE_3 mode, which users can modify when insmod.

Clock polarity (CPOL): that is, when SPI is idle, the level of clock signal SCLK. (1: when idle, is high-level; 0: when idle, is low-high)

Clock phase (CPHA): that is, SPI starts sampling at which edge of SCLK. (1: Start from the first edge; 0: Start from the second edge.)

speed_hz: Optional parameter, driver is 9.6Mhz by default, The actual maximum supported is determined by spi controller configuration and does not conflict with the theoretical maximum. Optional values are supported: 960000, 4800000, 9600000, 16000000, 19200000, 25000000, 50000000.

frame_size: Optional parameter, to set the size of each transfer in the spi transfer queue, default is 4096Bytes, users can set it according to the size of each data transfer.

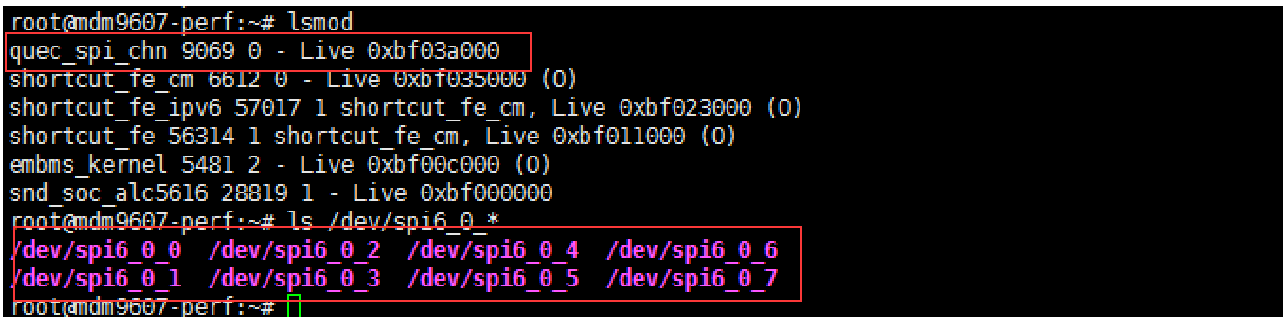
gpiomodemready: To set SPI_MRDY pin, driver code default use gpio34, can be modified by passing parameters.

Gpiomcuready: To set SPI_SRDY pin, driver code default use gpio52, can be modified by passing parameters.

Load Command

```
Insmod /lib/modules/3.18.20/kernel/drivers/spi/quec_spi_chn.ko speed_hz=19200000 gpiomodemready=38 gpiomcuready=34
```

Successful confirmation:



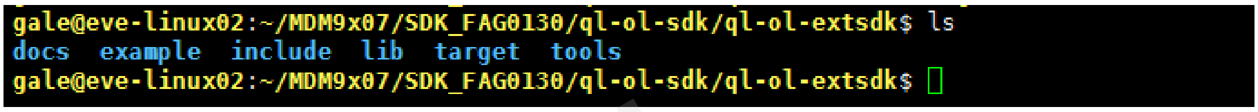
```
root@mdm9607-perf:~# lsmod
quec_spi_chn 9069 0 - Live 0xbf03a000
shortcut_fe_cm 6612 0 - Live 0xbf035000 (O)
shortcut_fe_ipv6 57017 1 shortcut_fe_cm, Live 0xbf023000 (O)
shortcut_fe 56314 1 shortcut_fe_cm, Live 0xbf011000 (O)
embms_kernel 5481 2 - Live 0xbf00c000 (O)
snd_soc_alc5616 28819 1 - Live 0xbf000000
root@mdm9607-perf:~# ls /dev/spi6_0_*
/dev/spi6_0_0  /dev/spi6_0_2  /dev/spi6_0_4  /dev/spi6_0_6
/dev/spi6_0_1  /dev/spi6_0_3  /dev/spi6_0_5  /dev/spi6_0_7
root@mdm9607-perf:~#
```

The extended 6-line SPI driver provided here virtualizes 8 data channels for use, and the client MCU can negotiate the purpose of each channel with the 4G module.

5 QuecOpen Application Layer API

5.1. User Programming Introduction

The SDK of QuecOpen project provides a complete set of user programming interface.
Reference path: ql-ol-sdk/ql-ol-extsdk/



```
gale@eve-linux02:~/MDM9x07/SDK_FAG0130/ql-ol-sdk/ql-ol-extsdk$ ls
docs  example  include  lib  target  tools
gale@eve-linux02:~/MDM9x07/SDK_FAG0130/ql-ol-sdk/ql-ol-extsdk$
```

The lib directory shown in the figure contains the API interface library provided by quectel; the include directory is the header file of all APIs; the example directory is an API usage example divided by function.

5.2. SPI API Introduction

As explained in Chapter 4, SPI device nodes have been successfully registered and can be operated at the application layer directly using the API below.

5.2.1. The Operation of Standard 4-Line API

The written of standard 4-line SPI application needs to be dependent on the library libql_peripheral.a.
Head file: ql_spi.h

SPI mode enumeration: working mode supported by SPI

```
typedef enum
{
    SPIMODE0  =   SPI_MODE_0,
    SPIMODE1  =   SPI_MODE_1,
    SPIMODE2  =   SPI_MODE_2,
    SPIMODE3  =   SPI_MODE_3,
}SPI_MODE;
```

SPI clock enumeration: the clock size supported by SPI by default.

```
typedef enum
{
    S_960K    =   960000,
    S_4_8M    =   4800000,
    S_9_6M    =   9600000,
    S_16M          =    16000000,
    S_19_2M        =    19200000,
    S_25M          =    25000000,
    S_50M          =    50000000,
}SPI_SPEED;
```

```
int Ql_SPI_Init(char *dev_name,SPI_MODE mode,uint8_t bits, SPI_SPEED speed);
```

Feature: Open the SPI device and configure the corresponding parameters.
Parameter: dev_name: SPI device, spidev.ko needs to be loaded manually.
SPI_MODE: Four working modes, SPI_MODE enumeration value.
bits: Number of bits of data word sent, support 4,8,16,32.
speed: SPI controller output clock, SPI_SPEED enumeration value.
Returned Value: Current opened device file descriptor.

```
int Ql_SPI_Write_Read(int fd,uint8_t* write_buf,uint8_t* read_buf,uint32_t len);
```

Feature: Read and write SPI data
Parameter: fd: SPI device file descriptor.
write_buf: SPI write data pointer.
read_buf: SPI read data pointer.
len: Read and write data length.
SPI communication is full-duplex, write_buf content can be configured to be 0 when read-only and read_buf content can be discarded when write-only.
Since standard SPI reads and writes in a transfer, all operations are full-duplex. Pass a NULL to read_buf, which is a write-only operation, and discard the data on the MISO line. Also, pass a NULL to write_buf, which is a read-only operation.
Returned valule: Success returns 0, otherwise returns a negative value.

```
int Ql_SPI_Deinit(int fd);
```

Feature: Turn off SPI device.
Parameter: fd: SPI device file descriptor.

Reference: ql-ol-extsdk/example/spi/std_spi

5.2.2. The Operation of Extended 6-Line API

The extended 6-line SPI driver provided here virtualizes 8 data channels for use, and the client MCU can negotiate the purpose of each channel with the 4G module.
Directly use open, read, write to read and write spi devices, and use select listener device to implement asynchronous notification.

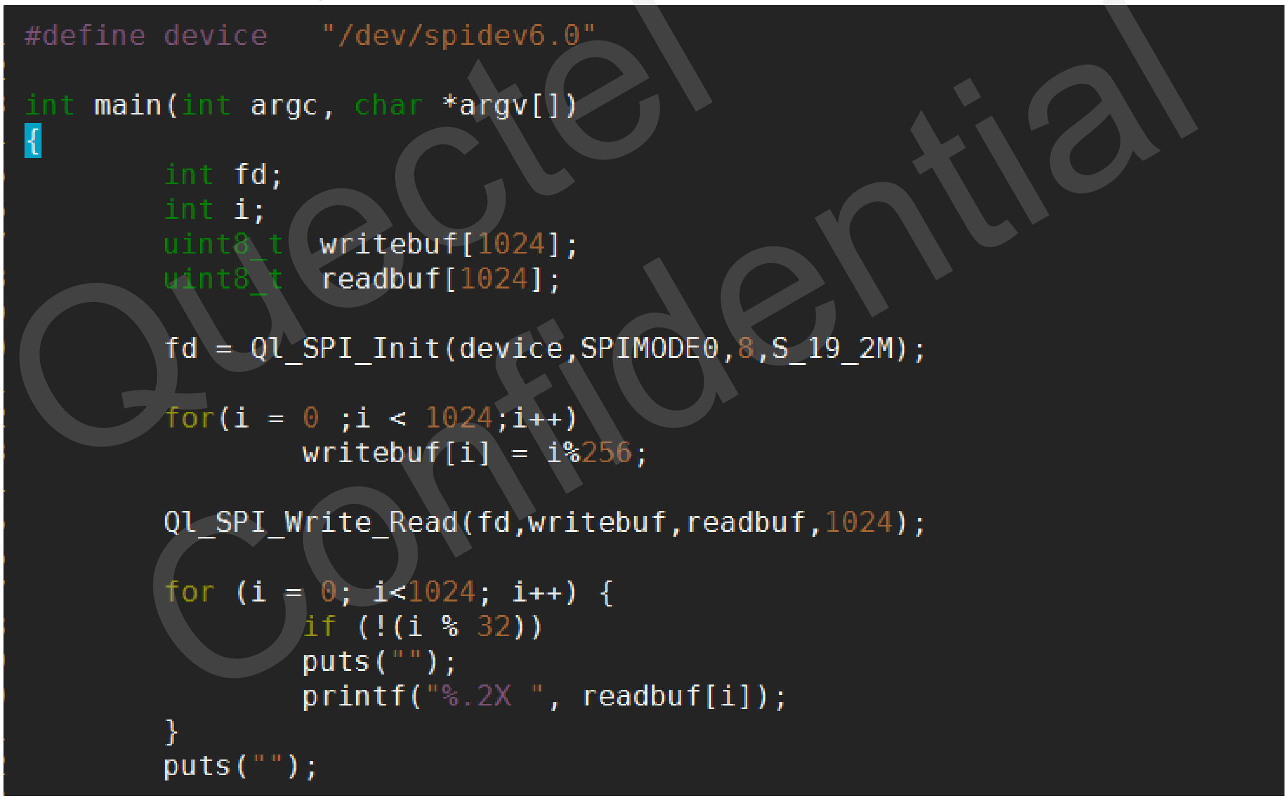
Reference: ql-ol-extsdk/example/spi/six_line

6 SPI Feature Test Verification

6.1. Example Introduction and Compilation

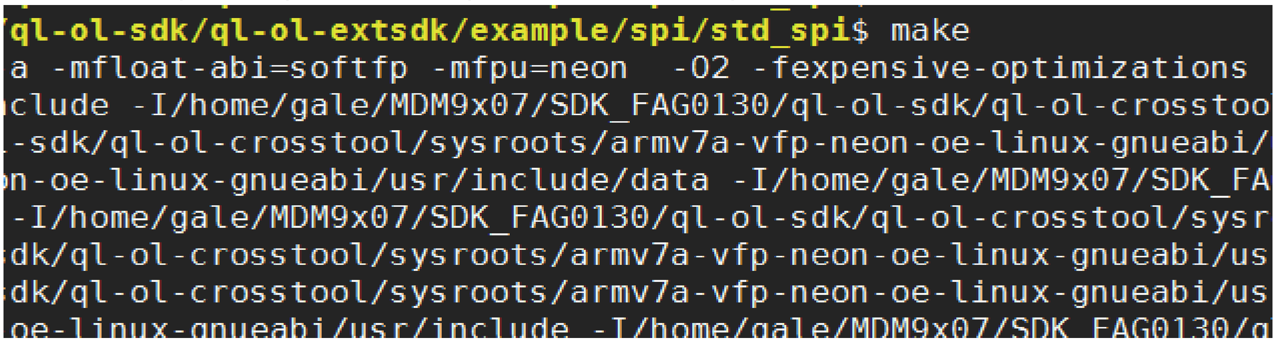
6.1.1. Standard 4-line SPI Introduction

ql-ol-extsdk/example/spi/std_spi example:
The Example initializes the device with SPI_MODE_0, 8bits/word, 19.2M speed, writes 1024 bytes to the device, and reads 1024 bytes back at the same time;



```
#define device   "/dev/spidev6.0"

int main(int argc, char *argv[])
{
        int fd;
        int i;
        uint8_t  writebuf[1024];
        uint8_t  readbuf[1024];

        fd = Ql_SPI_Init(device,SPIMODE0,8,S_19_2M);

        for(i = 0 ;i < 1024;i++)
                writebuf[i] = i%256;

        Ql_SPI_Write_Read(fd,writebuf,readbuf,1024);

        for (i = 0; i<1024; i++) {
                if (!(i % 32))
                puts("");
                printf("%.2X ", readbuf[i]);
        }
        puts("");
```

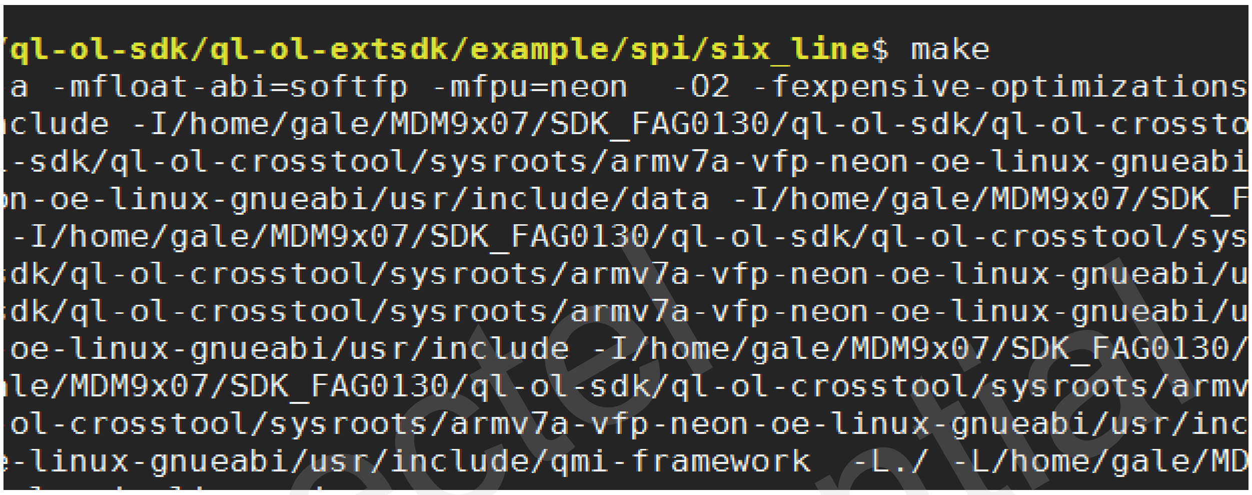
Enter the directory ql-ol-sdk/ql-ol-extsdk/example/spi/std_spi, make generates the example_spi executable program, the prerequisite for compiling must be the initialization of the cross-compilation environment. (source ql-ol-crosstool/ql-ol-crosstool-env-init)



```
/ql-ol-sdk/ql-ol-extsdk/example/spi/std_spi$ make
a -mfloat-abi=softfp -mfpu=neon  -O2 -fexpensive-optimizations
clude -I/home/gale/MDM9x07/SDK_FAG0130/ql-ol-sdk/ql-ol-crosstoo
-sdk/ql-ol-crosstool/sysroots/armv7a-vfp-neon-oe-linux-gnueabi/
n-oe-linux-gnueabi/usr/include/data -I/home/gale/MDM9x07/SDK_FA
-I/home/gale/MDM9x07/SDK_FAG0130/ql-ol-sdk/ql-ol-crosstool/sysr
dk/ql-ol-crosstool/sysroots/armv7a-vfp-neon-oe-linux-gnueabi/us
dk/ql-ol-crosstool/sysroots/armv7a-vfp-neon-oe-linux-gnueabi/us
oe-linux-gnueabi/usr/include -I/home/gale/MDM9x07/SDK_FAG0130/q
```

6.1.2. Extended 6-line SPI Introduction

ql-ol-extsdk/example/spi/six_line example:
The example main thread sends data to the SPI device, and the child thread listens at the same time to read whether it is readable.
Enter the directory ql-ol-sdk/ql-ol-extsdk/example/spi/six_line, make generates the example_spi example_six_line_spi executable program, the prerequisite for compiling must be the initialization of the cross-compilation environment. (source ql-ol-crosstool/ql-ol-crosstool-env-init)


```
/ql-ol-sdk/ql-ol-extsdk/example/spi/six_line$ make
 a -mfloat-abi=softfp -mfpu=neon  -O2 -fexpensive-optimizations
nclude -I/home/gale/MDM9x07/SDK_FAG0130/ql-ol-sdk/ql-ol-crossto
l-sdk/ql-ol-crosstool/sysroots/armv7a-vfp-neon-oe-linux-gnueabi
on-oe-linux-gnueabi/usr/include/data -I/home/gale/MDM9x07/SDK_F
 -I/home/gale/MDM9x07/SDK_FAG0130/ql-ol-sdk/ql-ol-crosstool/sys
sdk/ql-ol-crosstool/sysroots/armv7a-vfp-neon-oe-linux-gnueabi/u
sdk/ql-ol-crosstool/sysroots/armv7a-vfp-neon-oe-linux-gnueabi/u
 oe-linux-gnueabi/usr/include -I/home/gale/MDM9x07/SDK_FAG0130/
ale/MDM9x07/SDK_FAG0130/ql-ol-sdk/ql-ol-crosstool/sysroots/armv
 ol-crosstool/sysroots/armv7a-vfp-neon-oe-linux-gnueabi/usr/inc
e-linux-gnueabi/usr/include/qmi-framework  -L./ -L/home/gale/MD
```

6.2. Feature Test

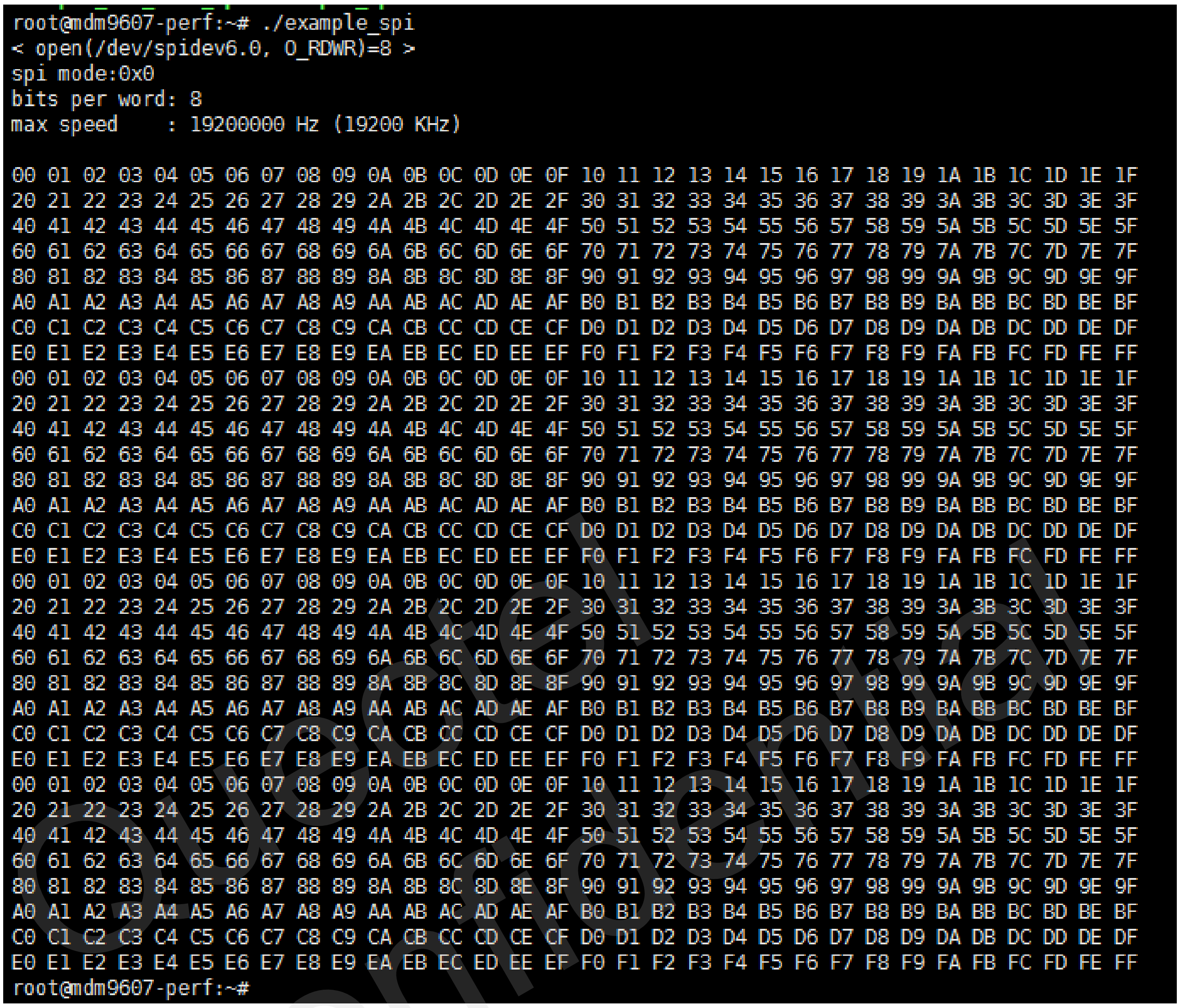
6.2.1. The Standard 4-Line SPI Test

Because the spi slave device is not connected, directly short-circuit GPIO_20, GPIO_21 for self-sending and self-receiving test.

(1) Load driver: insmod /lib/modules/3.18.20/kernel/drivers/spi/spidev.ko busnum=6 chipselect=0 spimode=0 maxspeed=19200000
(2) Compile and upload example_spi to module.

Use adb push < path of example_spi in the host computer> <path in the module, such as /usrdata>
Or upload using serial protocol rz.

(3) If use OPEN_EVB, need to connect the J0201 SPI pin with a jumper cap. GPIO_20 connect to GPIO_21
(4) Execute example_spi, as shown in the figure below, the received data is consistent with the sent data.

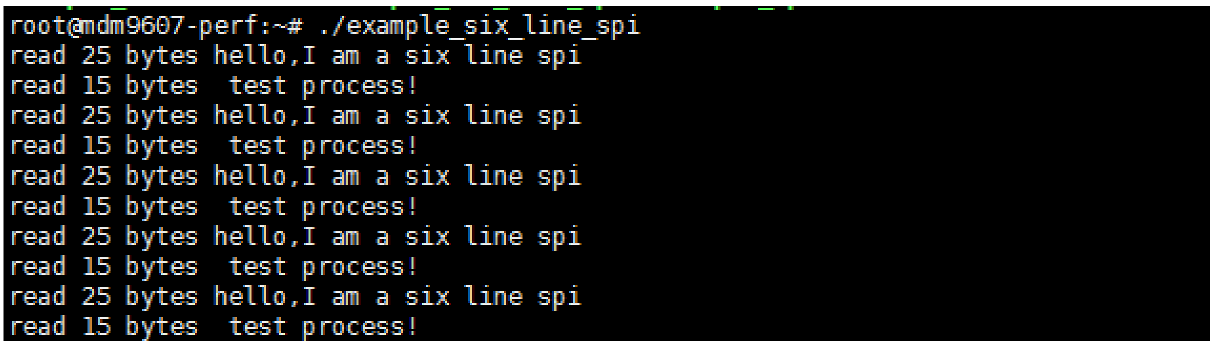


```
root@mdm9607-perf:~# ./example_spi
< open(/dev/spidev6.0, O_RDWR)=8 >
spi mode:0x0
bits per word: 8
max speed     : 19200000 Hz (19200 KHz)

00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F
60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F
80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F 90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF
C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF
E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 EA EB EC ED EE EF F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F
60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F
80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F 90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF
C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF
E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 EA EB EC ED EE EF F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F
60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F
80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F 90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF
C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF
E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 EA EB EC ED EE EF F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F
60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F
80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F 90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF
C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF
E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 EA EB EC ED EE EF F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF
root@mdm9607-perf:~#
```

6.2.2. The Extended 6-Line SPI Test

Because the spi slave device is not connected, directly short-circuit GPIO_20, GPIO_21, MRDY (gpiomodemready) and SRDY (gpiocmuready) for self-sending and self-receiving test.

(1) Load driver: insmod /lib/modules/3.18.20/kernel/drivers/spi/quec_spi_chn.ko speed_hz=19200000 gpiomodemready=38 gpiomcuready=34
(2) Compile and upload example_six_line_spi to module.
Use adb push < path of example_six_line_spi in the host computer> <path in the module, such as /usrdata>
Or upload using serial protocol rz.
(3) If use OPEN_EVB, need to connect the J0201 SPI pin with a jumper cap. GPIO_20 connects to GPIO_21, MRDY connects to SRDY
(4) Executeexample_six_line_spi, as shown in the figure below, the received data is consistent with the sent data.



```
root@mdm9607-perf:~# ./example_six_line_spi
read 25 bytes hello,I am a six line spi
read 15 bytes  test process!
read 25 bytes hello,I am a six line spi
read 15 bytes  test process!
read 25 bytes hello,I am a six line spi
read 15 bytes  test process!
read 25 bytes hello,I am a six line spi
read 15 bytes  test process!
read 25 bytes hello,I am a six line spi
read 15 bytes  test process!
```

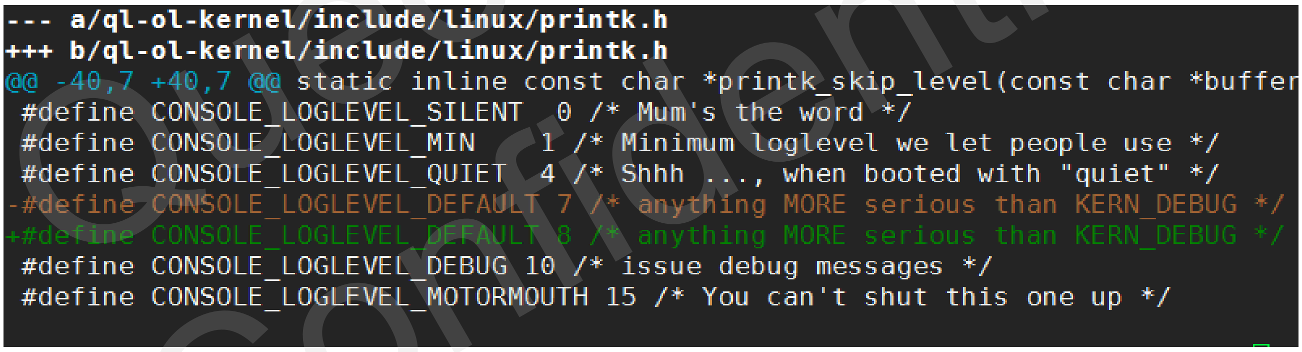
7 SPI Driver Debug Method

7.1. General Debugging Method

(1) In the SDK provided by QuecOpen, the default message level of the kernel log is 4 (KERN_WARNING), that is, if the kernel does not specify the message level when calling printk(), the default is 4.

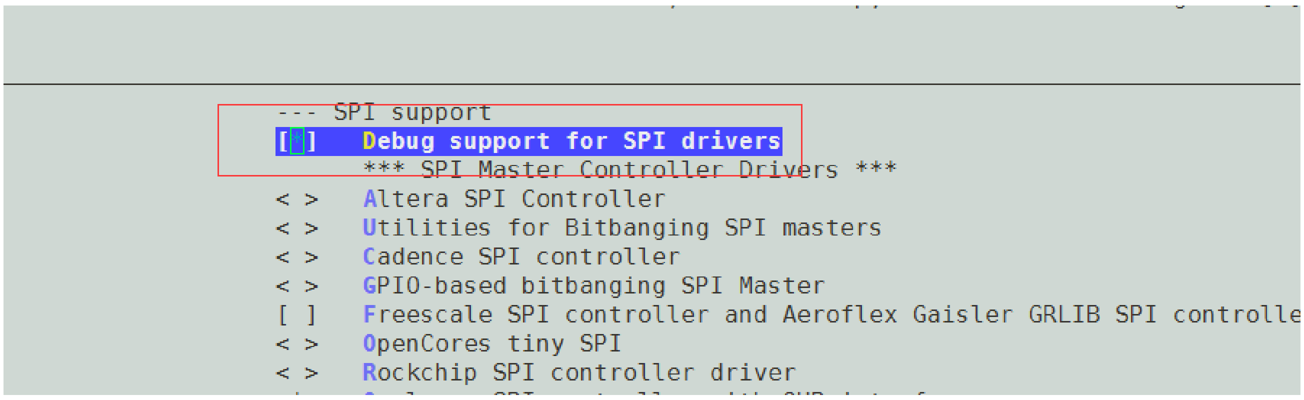
(2) The default print level of the console is 7 (KERN_DEBUG), that is, the kernel log less than 7 will be executed by the kernel code. Although it is executed, it is stored in the kernel log_buffer. When we use dmesg, the log output of the buffer will be output.

Then if user want to open the debug log that has been compiled into the kernel, modify the command line dmesg –n 8 directly;
Or modify the default value in the code.



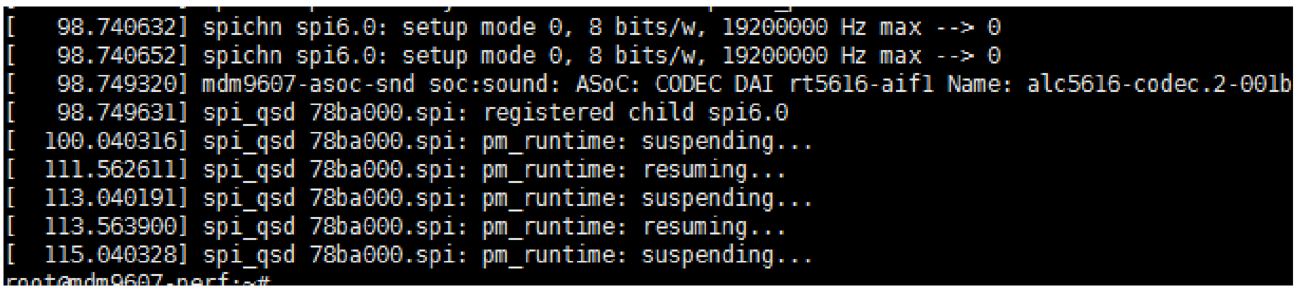
```
--- a/ql-ol-kernel/include/linux/printk.h
+++ b/ql-ol-kernel/include/linux/printk.h
@@ -40,7 +40,7 @@ static inline const char *printk_skip_level(const char *buffer
 #define CONSOLE_LOGLEVEL_SILENT  0 /* Mum's the word */
 #define CONSOLE_LOGLEVEL_MIN     1 /* Minimum loglevel we let people use */
 #define CONSOLE_LOGLEVEL_QUIET   4 /* Shhh ..., when booted with "quiet" */
-#define CONSOLE_LOGLEVEL_DEFAULT 7 /* anything MORE serious than KERN_DEBUG */
+#define CONSOLE_LOGLEVEL_DEFAULT 8 /* anything MORE serious than KERN_DEBUG */
 #define CONSOLE_LOGLEVEL_DEBUG 10 /* issue debug messages */
 #define CONSOLE_LOGLEVEL_MOTORMOUTH 15 /* You can't shut this one up */
```

However, in many driver modules, it will define own DEBUG compilation macro. If do not open this macro, even the printk code (KERN_DEBUG) will not be compiled.
make kernel_menuconfig, select the below SPI debug option, compile and download it.



```
--- SPI support
[*]    Debug support for SPI drivers
       *** SPI Master Controller Drivers ***
< >    Altera SPI Controller
< >    Utilities for Bitbanging SPI masters
< >    Cadence SPI controller
< >    GPIO-based bitbanging SPI Master
[ ]    Freescale SPI controller and Aeroflex Gaisler GRLIB SPI controlle
< >    OpenCores tiny SPI
< >    Rockchip SPI controller driver
```

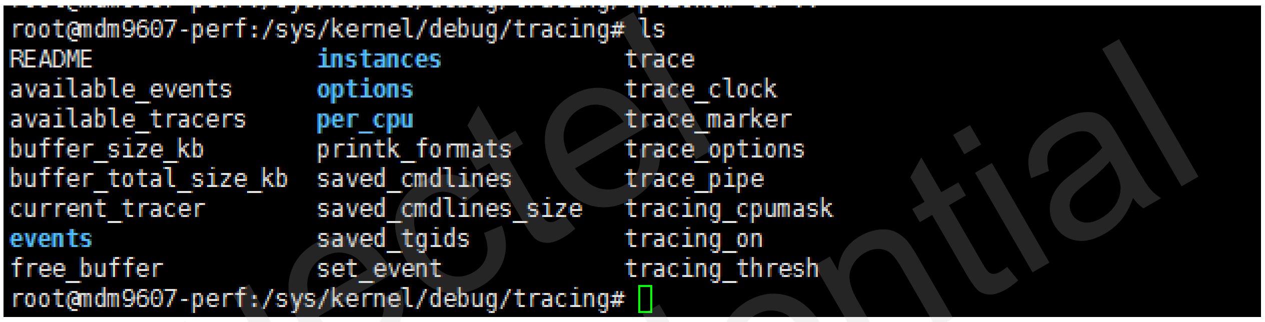
Then will show below message.



```
[   98.740632] spichn spi6.0: setup mode 0, 8 bits/w, 19200000 Hz max --> 0
[   98.740652] spichn spi6.0: setup mode 0, 8 bits/w, 19200000 Hz max --> 0
[   98.749320] mdm9607-asoc-snd soc:sound: ASoC: CODEC DAI rt5616-aif1 Name: alc5616-codec.2-001b
[   98.749631] spi_qsd 78ba000.spi: registered child spi6.0
[  100.040316] spi_qsd 78ba000.spi: pm_runtime: suspending...
[  111.562611] spi_qsd 78ba000.spi: pm_runtime: resuming...
[  113.040191] spi_qsd 78ba000.spi: pm_runtime: suspending...
[  113.563900] spi_qsd 78ba000.spi: pm_runtime: resuming...
[  115.040328] spi_qsd 78ba000.spi: pm_runtime: suspending...
```

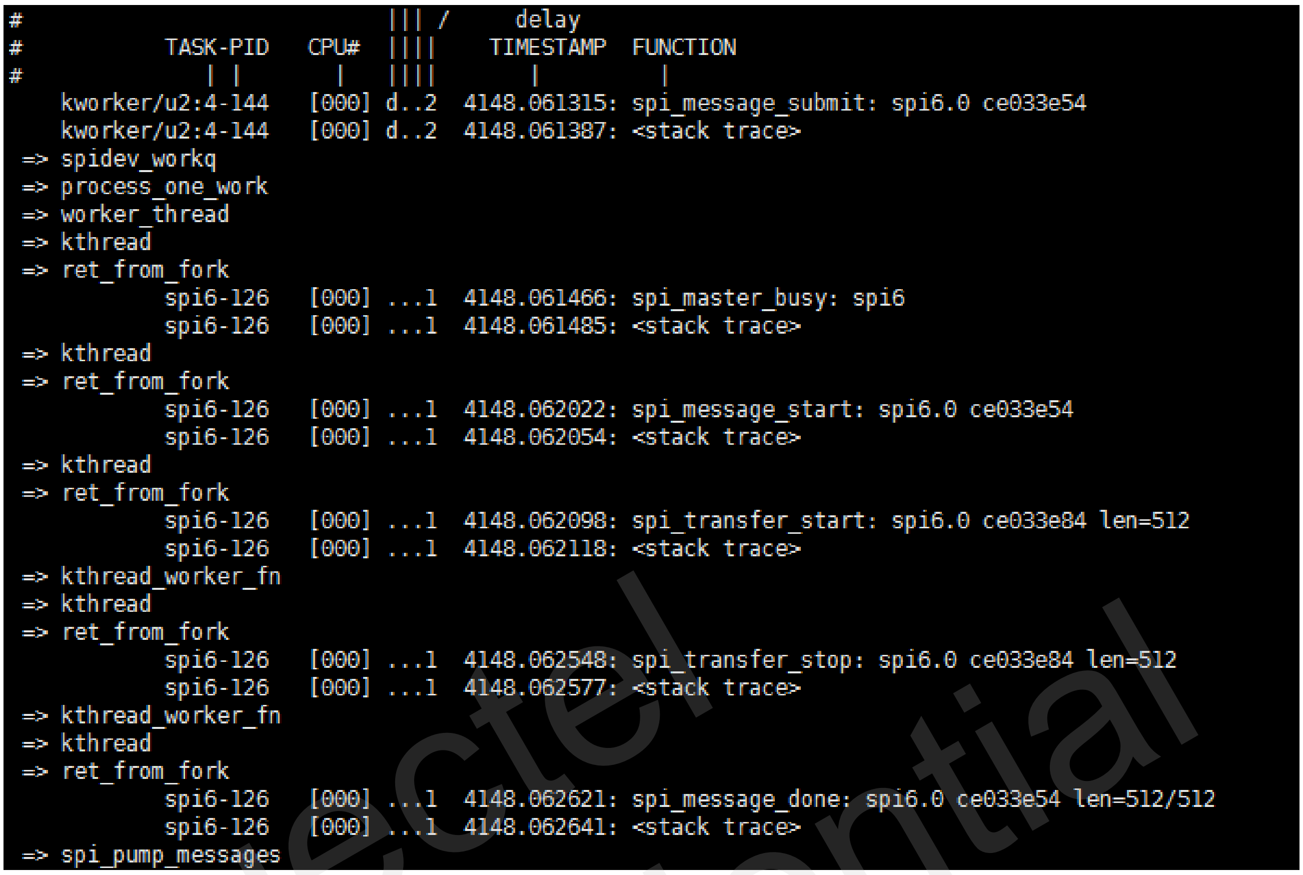
7.2. Debug with Kernel Tracer

The QuecOpen SDK opens the kernel debugfs and kernel tracer features in kernel hacking by default, kernel tracer is often used to debug the kernel.



```
root@mdm9607-perf:/sys/kernel/debug/tracing# ls
README                 instances              trace
available_events       options                trace_clock
available_tracers      per_cpu                trace_marker
buffer_size_kb         printk_formats         trace_options
buffer_total_size_kb   saved_cmdlines         trace_pipe
current_tracer         saved_cmdlines_size    tracing_cpumask
events                 saved_tgids            tracing_on
free_buffer            set_event              tracing_thresh
root@mdm9607-perf:/sys/kernel/debug/tracing#
```

(1) Open kernel call stack trace
    echo 1 > options/stacktrace
(2) Open the log of the printk output
    echo 1 > events/printk/enable
(3) Open spi event debug
    echo 1 > events/spi/enable
(4) Initiate a SPI access
(5) Call tatus of the SPI interface in the kerneL can be checked via cat trace.



```
#                              ||| /     delay
#          TASK-PID   CPU#  ||||    TIMESTAMP  FUNCTION
#             | |       |   ||||       |          |
    kworker/u2:4-144   [000] d..2  4148.061315: spi_message_submit: spi6.0 ce033e54
    kworker/u2:4-144   [000] d..2  4148.061387: <stack trace>
 => spidev_workq
 => process_one_work
 => worker_thread
 => kthread
 => ret_from_fork
            spi6-126   [000] ...1  4148.061466: spi_master_busy: spi6
            spi6-126   [000] ...1  4148.061485: <stack trace>
 => kthread
 => ret_from_fork
            spi6-126   [000] ...1  4148.062022: spi_message_start: spi6.0 ce033e54
            spi6-126   [000] ...1  4148.062054: <stack trace>
 => kthread
 => ret_from_fork
            spi6-126   [000] ...1  4148.062098: spi_transfer_start: spi6.0 ce033e84 len=512
            spi6-126   [000] ...1  4148.062118: <stack trace>
 => kthread_worker_fn
 => kthread
 => ret_from_fork
            spi6-126   [000] ...1  4148.062548: spi_transfer_stop: spi6.0 ce033e84 len=512
            spi6-126   [000] ...1  4148.062577: <stack trace>
 => kthread_worker_fn
 => kthread
 => ret_from_fork
            spi6-126   [000] ...1  4148.062621: spi_message_done: spi6.0 ce033e54 len=512/512
            spi6-126   [000] ...1  4148.062641: <stack trace>
 => spi_pump_messages
```