上次发了LCD调试笔记，大家很感兴趣，所以这次再来一篇：六轴传感器ICM20608驱动移植笔记，大家还需要什么移植笔记？可以留言。我们尽量满足。

移植思路：

先找到驱动：也许内核里已经有，也许需要去网上查找。

打开bing.com，搜“ICM20608 linux driver”，发现这个网址：

<https://github.com/torvalds/linux/blob/master/drivers/iio/imu/inv_mpu6050/inv_mpu_spi.c>

大胆假设，在linux内核中应该也有驱动，找到：

$ cd drivers/iio/imu/inv\_mpu6050

$ grep "20608" \* -nr

果然找到一堆驱动，比如：

inv\_mpu\_i2c.c:173: {"icm20608", INV\_ICM20608},

inv\_mpu\_iio.h:75: INV\_ICM20608,

inv\_mpu\_iio.h:232:#define INV\_ICM20608\_WHOAMI\_VALUE 0xAF

inv\_mpu\_spi.c:85: {"icm20608", INV\_ICM20608},

100ASK\_IMX6ULL上使用SPI接口连接ICM20608，所以我们要找的驱动就是inv\_mpu\_spi.c。

这样事情就好办了。

接下来就是配置设备树，

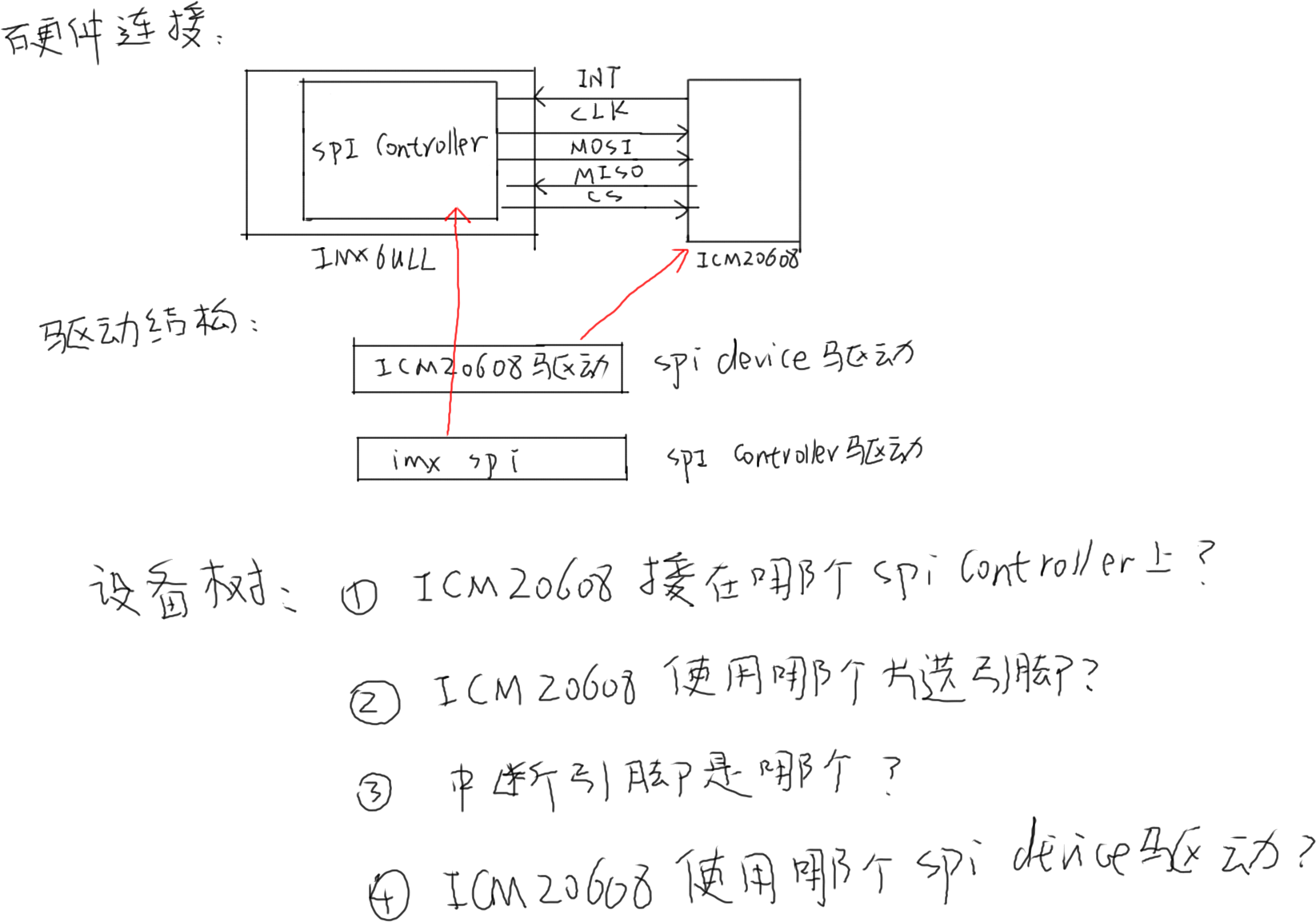
最后测试。

1.1 SPI驱动程序框架

我们需要弄清楚SPI驱动程序的结构，才能够理解驱动程序，添加设备树信息。

特别是对于ICM20608，在内核的设备树文档Documentation/devicetree/bindings目录下，找不到任何有用的信息。

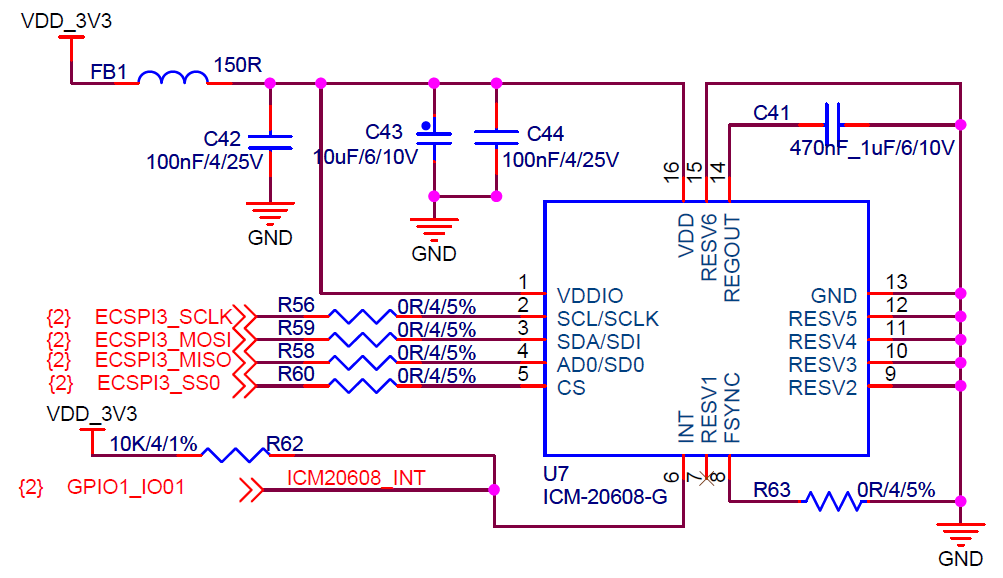
那我们只能看源码了，看源码之前先弄清结构。



我们需要阅读源码来确定设备树的上述4点内容。

1.2 ICM20608设备树

100ASK\_IMX6ULL开发板上，ICM20608接在哪一个SPI控制器上？



如上图，接在ECSPI3这个SPI控制器上。

打开设备树文件arch/arm/boot/dts/100ask\_imx6ull-14x14.dts，可以看到如下代码：

&ecspi3 {

pinctrl-names = "default";

pinctrl-0 = <&pinctrl\_ecspi3>;

cs-gpios = <&gpio1 20 GPIO\_ACTIVE\_LOW>;

status = "okay";

spidev: icm20608@0{

compatible = "invensense,icm20608";

interrupt-parent = <&gpio1>;

interrupts = <1 1>;

spi-max-frequency = <8000000>;

reg = <0>;

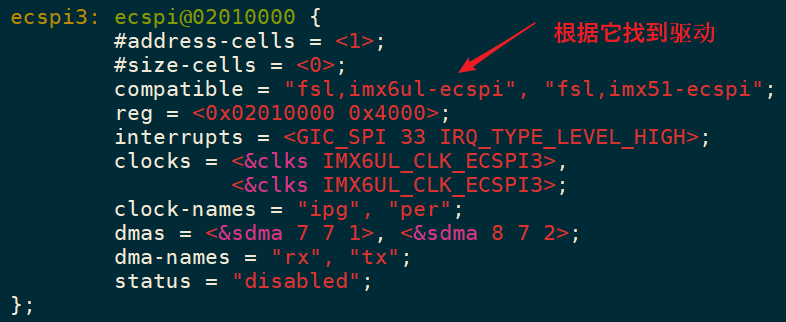
};

};

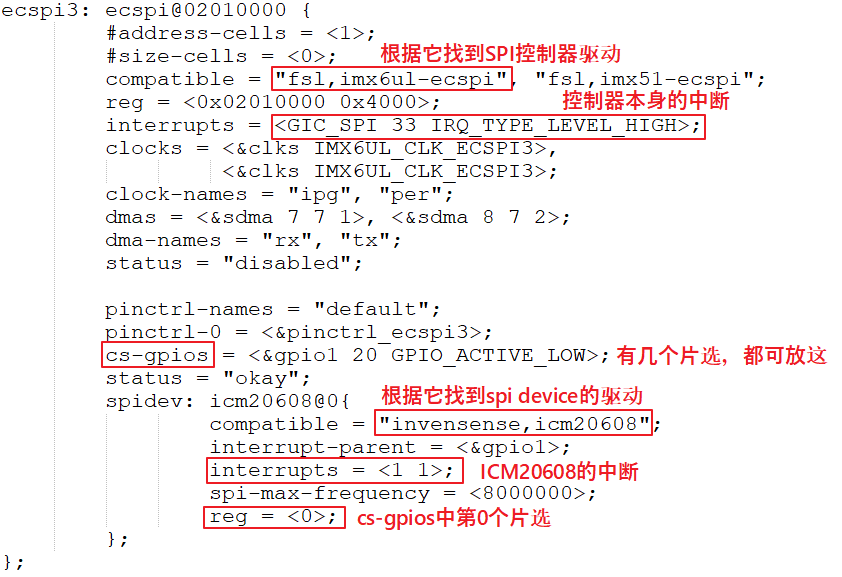
这是我们同事写出来的，具体过程就省略掉，看着挺简单，写的时候花了不少时间。

我们来研究一下，它位于ecspi3节点之下，ecspi3节点肯定就是SPI控制器之一。要找到它的compatible属性，才能找到SPI控制器的驱动程序。

打开 imx6ull.dtsi，果然有：



把ecspi3节点和ICM20608节点合并起来，内容如下：



1.3 SPI控制器驱动程序

1.3.1 找到驱动程序

根据上述设备树信息，在Linux内核源码目录下搜"fsl,imx6ul-ecspi"，就可以找到SPI控制器的驱动程序：

book@100ask:~/100ask\_imx6ull-sdk/Linux-4.9.88/drivers$ grep "fsl,imx6ul-ecspi" \* -nr

Binary file built-in.o matches

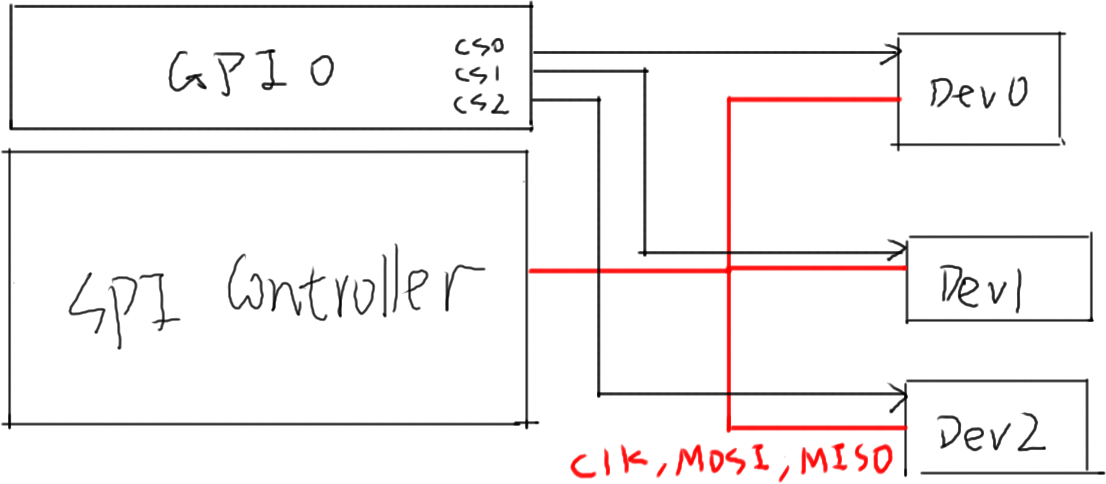
spi/spi-imx.c:782: { .compatible = "fsl,imx6ul-ecspi", .data = &imx6ul\_ecspi\_devtype\_data, },

可见，spi/spi-imx.c就是我们要找的SPI控制器驱动程序。

1.3.2 我们能做的不多，只能在设备树中指定片选

设备树已经在前面列出来了。

一个SPI控制器可以连接多个SPI设备，每个SPI设备使用都有单独的片选信号，如下图：

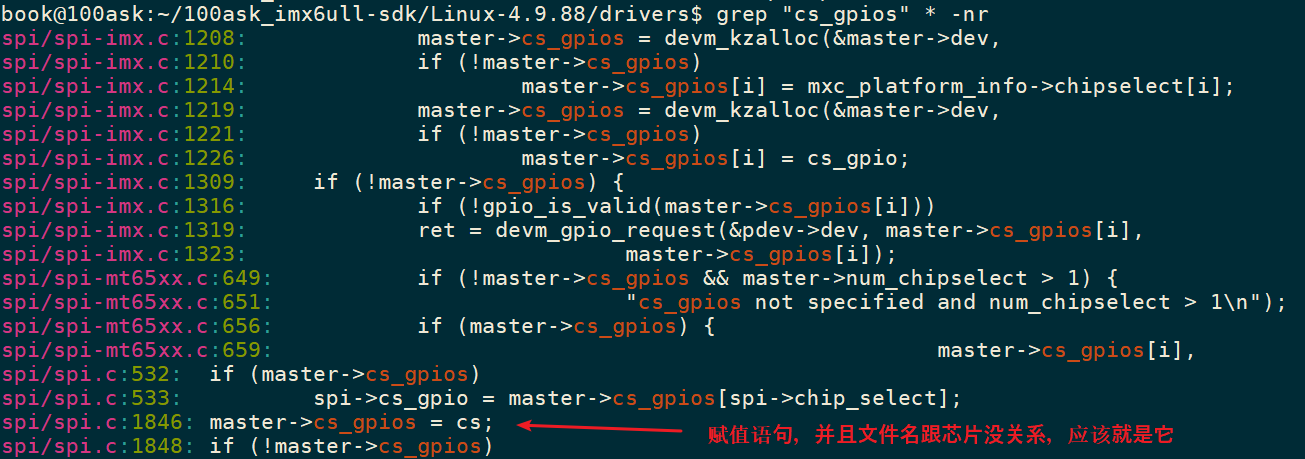


在SPI控制器驱动和设备树中，我们最关心的是片选信号，其他信号我们无法修改。

阅读spi-imx.c的spi\_imx\_probe函数：



我们得找到设备树的处理代码，看看它是怎么从设备树中设置cs\_gpios的，搜“cs\_gpios”，得到：



打开drivers/spi/spi.c第1846行，它确实是用来处理设备树的：



of\_spi\_register\_master函数的调用流程：

spi\_imx\_probe (drivers/spi/spi-imx.c)

master = spi\_alloc\_master(&pdev->dev, sizeof(struct spi\_imx\_data));

ret = spi\_bitbang\_start(&spi\_imx->bitbang); (drivers/spi/spi-bitbang.c)

ret = spi\_register\_master(spi\_master\_get(master)); (drivers/spi/spi.c)

status = of\_spi\_register\_master(master); (drivers/spi/spi.c,就是它处理cs-gpios)

1.3 SPI设备驱动程序

设备树已经在前面列出来了。

SPI设备跟SPI控制器之间的硬件连接，能确定的也就3点，我们需要在设备树中指定：

a. 接到哪个SPI控制器去？

IMX6ULL中有多个SPI控制器，在设备树里，把SPI设备的节点放到某个SPI控制器节点之下就可以。

b. SPI设备有没有中断？用哪一个中断？

我们的ICM20608如下设置：

interrupt-parent = <&gpio1>;

interrupts = <1 1>;

b. SPI设备使用哪个片选？

在SPI控制器节点里有cs-gpios属性，里面定义有1个或多个片选。

在SPI设备的节点中，用reg属性指定使用cs-gpios中的哪个片选(从0开始)，如下：

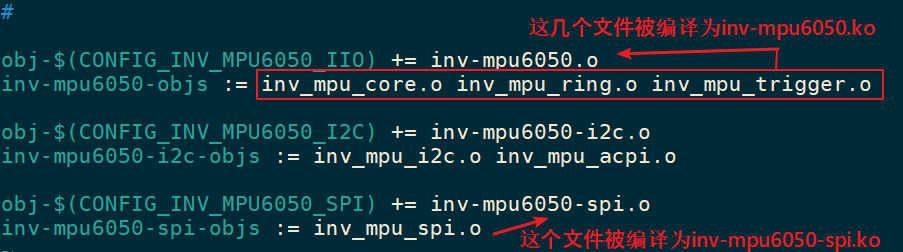
reg = <0>;

你看，我们只关心设备树，似乎没怎么看驱动程序啊。

根据设备节点的compatible属性可找到ICM20608的驱动程序为：

drivers\iio\imu\inv\_mpu6050\inv\_mpu\_spi.c

打开drivers/iio/imu/inv\_mpu6050/Makefile，其内容如下：



上图中的2个ko文件，都需要安装。前一个要先安装，它为后一个ko提供一些函数。

1.4 测试

ICM20608的驱动程序是基于IIO驱动来编写的，我们还没深入研究IIO。

所以本节只是简单地讲讲怎么测试ICM20608，以后再深入研究。

首先，请确保你的设备树文件arch/arm/boot/dts/100ask\_imx6ull-14x14.dts中，ICM20608节点的属性中含有cs-gpios，注意：不是cs-gpio。(我们曾经提供一个补丁，它处理的是cs-gpio属性，最新版本的内核已经去除了这个补丁，使用drivers/spi/spi.c处理的是cs-gpios属性)

然后在开发板上安装驱动程序：

[root@imx6ull:~]# insmod inv-mpu6050.ko

[root@imx6ull:~]# insmod inv-mpu6050-spi.ko

[ 56.892312] inv-mpu6000-spi spi2.0: mounting matrix not found: using identity...

你就可以看到设备节点了：

[root@imx6ull:~]# ls /dev/iio\*

/dev/iio:device0 /dev/iio:device1

也可以看到/sys下创建了一些文件：

[root@imx6ull:~]# ls /sys/bus/iio/devices

iio:device0 iio:device1 trigger0

是iio:device0还是iio:device1对应ICM20608？可以cat上述目录里的name文件：

[root@imx6ull:~]# cat /sys/bus/iio/devices/iio\:device1/name

icm20608

然后就可以进入/sys/bus/iio/devices/iio\:device1目录，读取里面的文件，同时转动开发板，可以观察到值在变化：

[root@imx6ull:~]# cd /sys/bus/iio/devices/iio\:device1

[root@imx6ull:/sys/bus/iio/devices/iio:device1]# ls

buffer in\_anglvel\_y\_calibbias

current\_timestamp\_clock in\_anglvel\_y\_raw

dev in\_anglvel\_z\_calibbias

in\_accel\_matrix in\_anglvel\_z\_raw

in\_accel\_mount\_matrix in\_gyro\_matrix

in\_accel\_scale in\_temp\_offset

in\_accel\_scale\_available in\_temp\_raw

in\_accel\_x\_calibbias in\_temp\_scale

in\_accel\_x\_raw name

in\_accel\_y\_calibbias of\_node

in\_accel\_y\_raw power

in\_accel\_z\_calibbias sampling\_frequency

in\_accel\_z\_raw sampling\_frequency\_available

in\_anglvel\_mount\_matrix scan\_elements

in\_anglvel\_scale subsystem

in\_anglvel\_scale\_available trigger

in\_anglvel\_x\_calibbias uevent

in\_anglvel\_x\_raw

[root@imx6ull:/sys/bus/iio/devices/iio:device1]# cat in\_accel\_x\_raw

-141

[root@imx6ull:/sys/bus/iio/devices/iio:device1]# cat in\_accel\_x\_raw

-4652

[root@imx6ull:/sys/bus/iio/devices/iio:device1]# cat in\_accel\_x\_raw

844

**注意**：我们对ICM20608并无深入研究，上述/sys目录中各值有什么含义，留待你们去发现。