# 版权声明

**百问科技©2019**

深圳百问网科技有限公司版权所有，并保留对本手册及声明的一切权力。

未得到本公司的书面许可，任何单位和个人不得以任何方式或形式对本手册内的任何部分进行复制、摘录、备份、修改、传播、翻译成其他语言、将其全部或部分用于商业用途

# 更新记录

|  |  |
| --- | --- |
| **类别** | 100ask-imx6ull系列文档 |
| **文档名** | 100ask\_imx6ull Development Manual |
| **当前版本** | 1.0 |
| **适用型号** | MCIMX6Y2xxx08 |
| **编辑** | 百问科技文档编辑团队 |
| **审核** | 韦东山 |

|  |  |  |
| --- | --- | --- |
| **修改日志** | | |
| 版本 | 修改时间 | 更改说明 |
| 1.0 | 2020.03.04 | 初始版本 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# 目录

[版权声明 I](#_Toc68102530)

[更新记录 II](#_Toc68102531)

[目录 1](#_Toc68102532)

[在100ASK\_IMX6ULL板子上支持其他型号的屏幕 2](#_Toc68102533)

[1. 在100ASK\_IMX6ULL底板上如何接其他厂家的屏幕 2](#_Toc68102535)

[2. 移植思路 3](#_Toc68102536)

[3. 移植LCD驱动 3](#_Toc68102537)

[3.1 确定LCD参数 3](#_Toc68102538)

[3.2 修改设备树 5](#_Toc68102539)

[3.3 完善驱动 5](#_Toc68102540)

[4. 移植触摸屏驱动 8](#_Toc68102541)

[4.1 确定触摸屏型号 8](#_Toc68102542)

[4.2 在设备树中指定触摸IC信息 9](#_Toc68102543)

[4.3 重新配置内核添加驱动 11](#_Toc68102544)

[4.4 调试：找出问题 12](#_Toc68102545)

[4.5 解决方法 13](#_Toc68102546)

[4.6 其他情况 13](#_Toc68102547)

[4.7 gt9xx芯片固件更新 14](#_Toc68102548)

# 在100ASK\_IMX6ULL板子上支持其他型号的屏幕



1. 在100ASK\_IMX6ULL底板上如何接其他厂家的屏幕

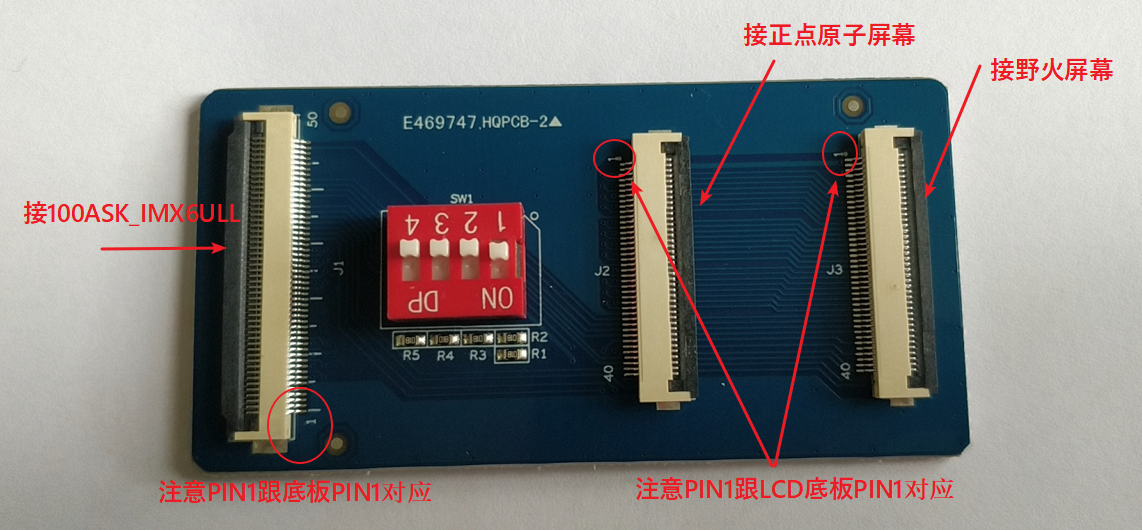
很多学员有过STM32的学习经验，他们手上的开发板很多，LCD也很多。

一个LCD还挺贵的，不能浪费。

各家的LCD引脚顺序都不一样，所以别家的LCD不能直接接到100ASK\_IMX6ULL开发板，需要转接板。

大部分单片机学员都是使用正点原子、野火的板子，有他们的屏。

针对这两家的屏，我们做了转接板，如下：



下面的文档是示例，教你们怎么自己移植。

2. 移植思路

LCD除了显示之外，它的表面通常还贴有一个触摸屏。

所以我们移植的是2个设备的驱动：LCD、触摸屏。

LCD驱动在内核中已经有了，并且很完善，我们只需要修改设备树就可以：修改时序等LCD参数，修改背光引脚等板子参数。

触摸屏的驱动在内核中一般也有了，各厂家用的触摸屏IC可能不同。需要配置内核把它加进去，同时修改设备树：指定触摸IC的信息(比如I2C地址)，指定中断引脚。

如果能拿到这块LCD在别的板子上的内核源码，就可以参考它的LCD参数、触摸IC信息。

再结合你用的开发板，把涉及的GPIO找出来写入设备树。

简单吧？

开工！

3. 移植LCD驱动

**注意**：100ASK\_IMX6ULL不能直接连接其他厂家的屏，需要转接板。

IMX6ULL跟LCD的连接框图如下：



IMX6ULL内部有LCD控制器，肯定是厂家对这个LCD控制器最熟悉了，所以他们为了卖芯片，一般都会在内核中做好LCD控制器的驱动程序。

而IMX6ULL可以接各种LCD，这些LCD参数各有不同。LCD控制器的驱动程序会去设备树中获得这些参数，并根据这些参数来设置LCD控制器。

所以，我们要做的事情从理论上讲很简单：根据LCD参数修改设备树。

但是，谁说厂家的驱动就没有BUG，就完美无缺了？

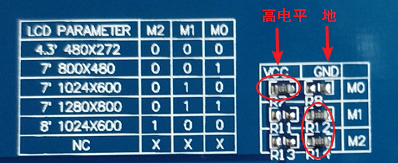
我们要做的事有3项：确定LCD参数，修改设备树，完善驱动。

3.1 确定LCD参数

如果还保留有LCD的芯片手册，这是最好的。可惜我碰到的大多学员早已经忘记这些资料在哪了。

野火的屏在板子背后直接写明分辨率是多少，这比较简单直接。

正点原子的屏在背后画有一个表格，如下：



上图例子中，M2、M1接地，M0接电源，所以M2M1M0为001；对比表格可知它的分辨率为800x480。

如果是从零开发，我们还需要找到芯片手册确定LCD的详细时序。

我们没有LCD的手册，即使有也懒得看。

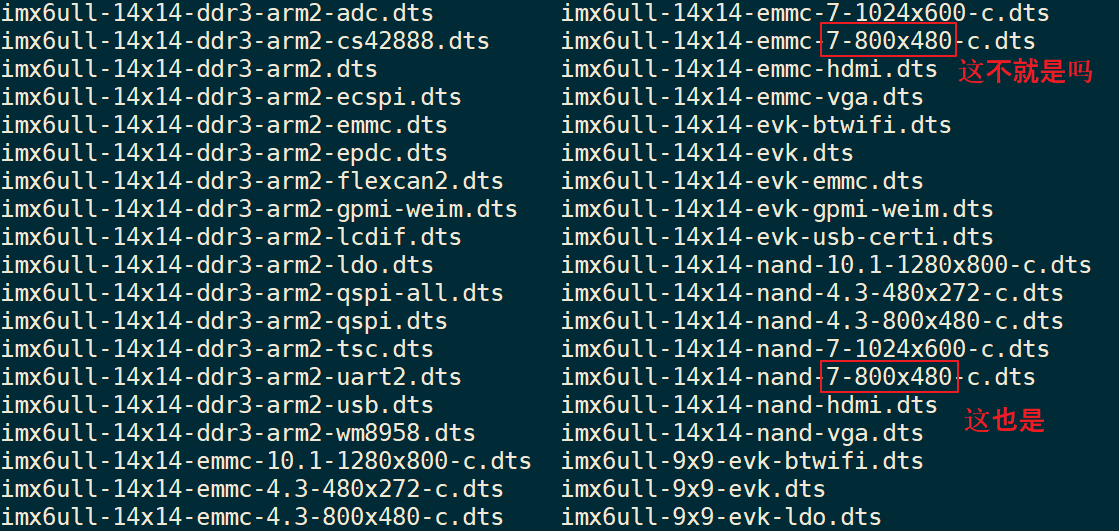
直接看厂家的源码不就行了？直接点，直接看它的设备树不就行了？

找到LCD厂家的IMX6ULL内核源码，执行以下命令：

$ cd arch/arm/boot/dts/

$ ls \*imx6ull\*.dts

结果如下：



可以找到2个设备树文件，我们只关心里面的LCD信息，打开任意一个看看：



3.2 修改设备树

100ASK\_IMX6ULL用的内核版本是4.9.88，版本高一点，但是IMX6ULL设备树的写法完全一样。

设备树文件是arch/arm/boot/dts/100ask\_imx6ull-14x14.dts，替换下图红框部分：



修改好设备树后，就可以编译了。

为方便你们操作，下面贴出编译命令，假设你已经位于我们提供的内核源码目录中：

$ export ARCH=arm

$ export CROSS\_COMPILE=arm-linux-gnueabihf-

$ export PATH=$PATH:/home/book/100ask\_imx6ull-sdk/ToolChain/gcc-linaro-6.2.1-2016.11-x86\_64\_arm-linux-gnueabihf/bin

$ make dtbs

新编译好的设备树文件为：arch/arm/boot/dts/100ask\_imx6ull-14x14.dtb。

怎么更新设备树？请看我们的开发板使用手册。

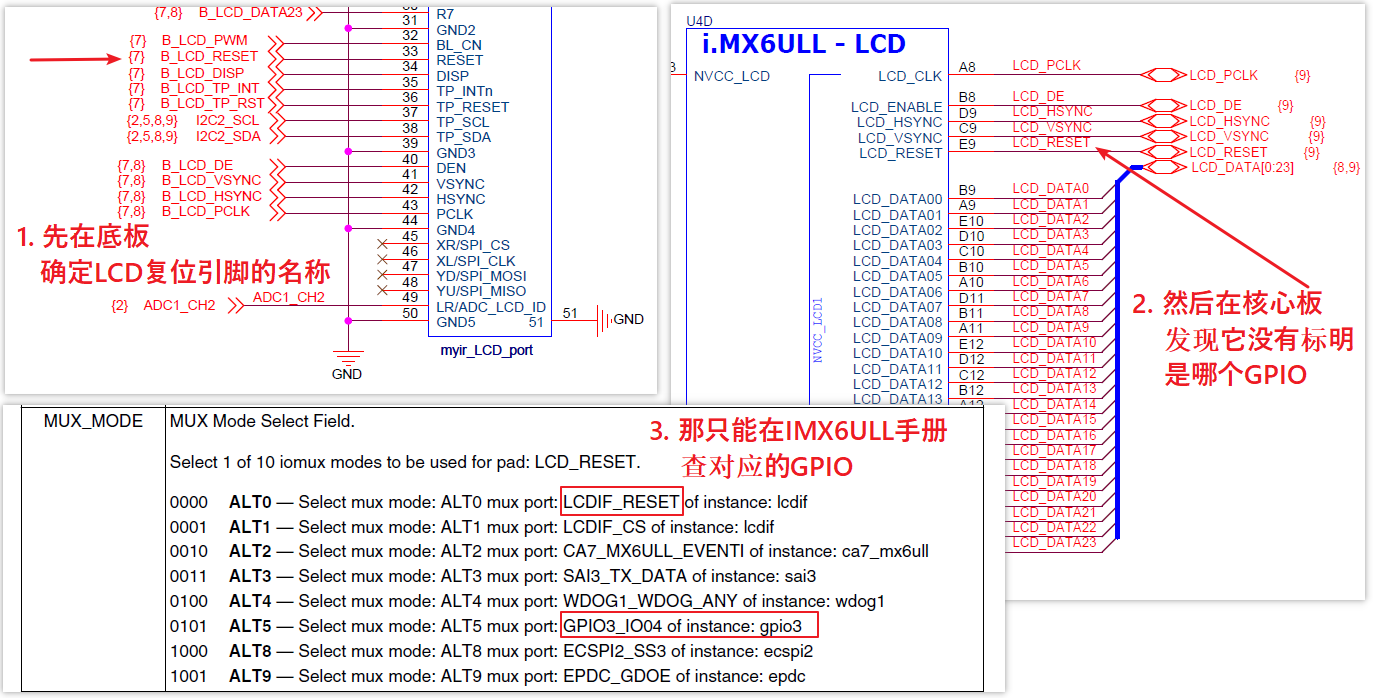
更新设备树后，重启板子观察效果。

3.3 完善驱动

使用新的设备树启动板子后，你会发现一个神奇的现象：LCD有时候有显示，有时候没有，不断地冷启动偶尔会有显示。

如果你经验丰富，可以判断这是复位问题。

为验证是否复位问题，我们可以执行命令手工发出复位信号，先确定LCD复位引脚是哪个GPIO：



从上图可以确定LCD的复位引脚用到GPIO3\_IO04，那我们可以使用GPIO子系统来验证。

执行某些命令让GPIO输出低电平，再输出高电平，这样就可以复位LCD了。

3.3.1 使用GPIO子系统复位LCD

GPIO3\_IO04在GPIO子系统中编号为：(3-1)\*32+4=68，它是第68号GPIO。

板子进入Linux后，执行以下命令：

$ fb-test // LCD上应该显示红绿蓝色块

$ echo 68 > /sys/class/gpio/export // 导出68号GPIO

$ echo out > /sys/class/gpio/gpio68/direction // 设置为输出引脚

$ echo 0 > /sys/class/gpio/gpio68/value // 让它输出0

$ echo 1 > /sys/class/gpio/gpio68/value // 让它输出1

$ echo 68 > /sys/class/gpio/unexport // unexport

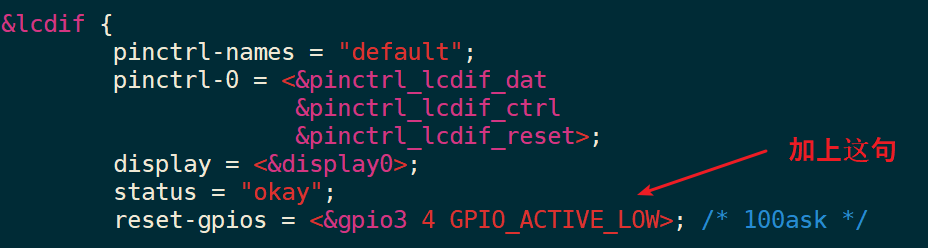
你会发现一旦执行上述命令，LCD立刻就有显示了。

所以，LCD驱动不完善，应该加上复位信号。

3.3.2 修改设备树：指定复位引脚

设备树文件为：arch/arm/boot/dts/100ask\_imx6ull-14x14.dts

如下图修改：



把添加的代码写出来：

reset-gpios = <&gpio3 4 GPIO\_ACTIVE\_LOW>; /\* 100ask \*/

3.3.3 修改驱动：复位LCD

LCD驱动程序是哪个？

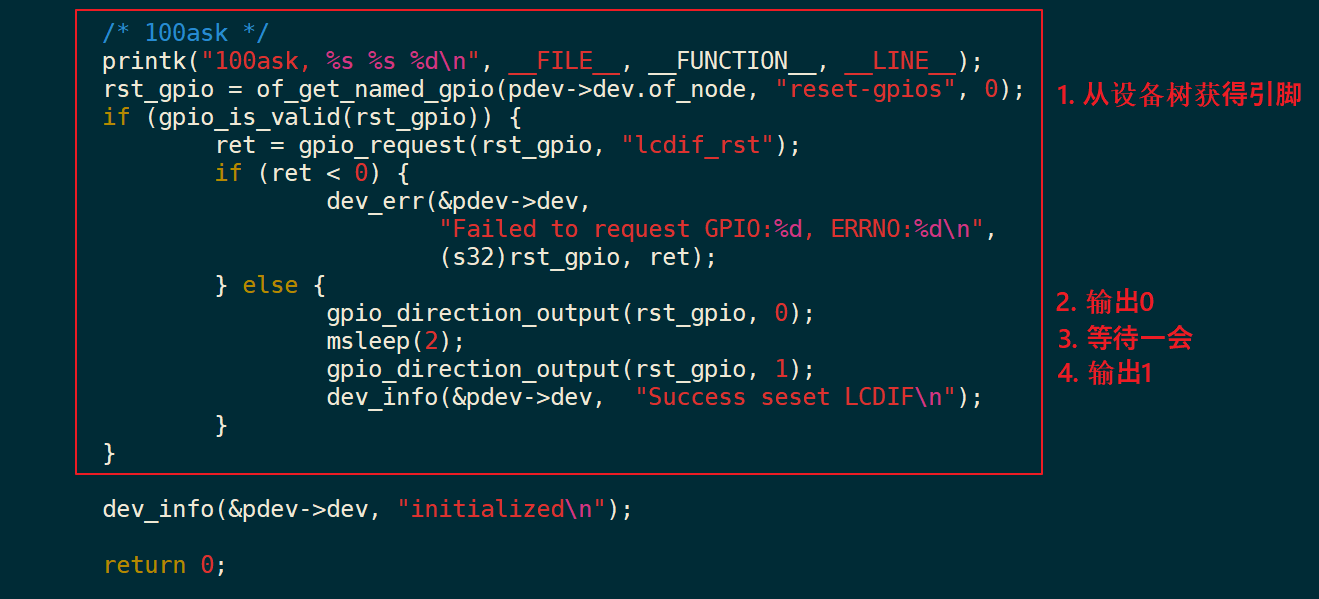
在Linux内核源码目录下执行命令：

$ ls drivers/video/fbdev/\*.o

drivers/video/fbdev/built-in.o drivers/video/fbdev/mx3fb.o drivers/video/fbdev/mxsfb.o

发现有2个.o文件：mx3fb.o、mxsfb.o。我们是imx6ull，应该是后者。

我们在mxsfb.c中mxsfb\_probe函数的后面添加复位代码，如下图修改：



把添加的代码写出来(在mxsfb.c中mxsfb\_probe函数的后面添加复位代码，)：

/\* 100ask \*/

printk("100ask, %s %s %d\n", \_\_FILE\_\_, \_\_FUNCTION\_\_, \_\_LINE\_\_);

rst\_gpio = of\_get\_named\_gpio(pdev->dev.of\_node, "reset-gpios", 0);

if (gpio\_is\_valid(rst\_gpio)) {

ret = gpio\_request(rst\_gpio, "lcdif\_rst");

if (ret < 0) {

dev\_err(&pdev->dev,

"Failed to request GPIO:%d, ERRNO:%d\n",

(s32)rst\_gpio, ret);

} else {

gpio\_direction\_output(rst\_gpio, 0);

msleep(2);

gpio\_direction\_output(rst\_gpio, 1);

dev\_info(&pdev->dev, "Success seset LCDIF\n");

}

}

修改完后，重新编译得到zImage和100ask\_imx6ull-14x14.dtb，更新开发板，重启，完美！

4. 移植触摸屏驱动

4.1 确定触摸屏型号

还是那句话，如果有触摸屏数据手册，看手册就好了。

如果没有手册，怎么办？

触摸屏的主控芯片一般都是I2C接口的，那么我们可以把屏接到板子上，用i2cdetect检测出I2C设备的地址，根据地址就可以知道它的型号。

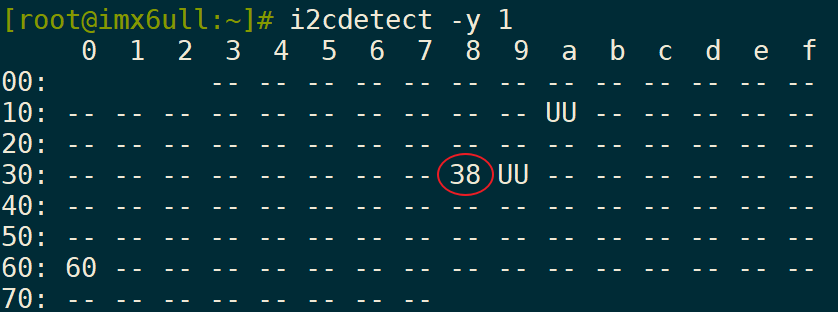
**注意**：100ASK\_IMX6ULL不能直接连接其他厂家的屏，需要转接板。

接上屏幕后，启动开发板进入Linux，执行如下命令：

[root@imx6ull:~]# i2cdetect -y 1

命令解析：“-y”表示 Disable interactive mode，简单地说就是“别让我确认了，赶紧执行”；“1”表示I2C总线1(从0开始)。

结果如下：



上图中，

“--”表示没有这个地址对应的I2C设备；

“UU”表示这个地址的I2C设备已经有驱动在使用占用它了，那这个I2C设备肯定是存在的；

其他数值表示该地址对应的I2C设备是存在的，并且还没有驱动程序跟它匹配。

根据上图，我们可以知道0x38就是触摸屏设备，为什么！为什么不是0x60？你可以把屏幕取下，再重新执行命令，就可以看到“38”消失了。

根据0x38，我们得找到对应的芯片型号，怎么找？去内核设备树目录里找。

$ cd arch/arm/boot/dts/

$ grep "@38" \* -nR

可以得到很多结果，比如：



地址为0x38的I2C芯片有不少，比如HDMI PHY，还有ft5306、ft5x06。你在百度搜一下“ft5306”，它确实就是触摸屏芯片。所以这款触摸屏的主控芯片就是ft5x06。x表示某些数字，可能有多个型号，我们暂时没不用去细分。

4.2 在设备树中指定触摸IC信息

IMX6ULL跟触摸IC的连接图如下：



所以，我们要确定的信息是：

a. 它接在哪个I2C控制器上？

b. 它的I2C地址是？

c. 复位引脚使用哪个GPIO？低电平有效还是高电平有效？

d. 中断引脚使用哪个GPIO？低电平有效还是高电平有效？

不知道怎么写？没关系，参考！

a. 对于ft5x06，设备树节点中有哪些内容？

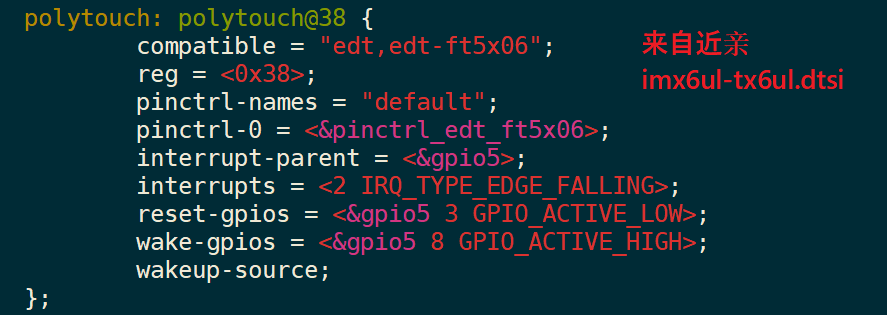
b. 那些内容怎么适配100ASM\_IMX6ULL板子？即怎么改成100ASK\_IMX6ULL所用的GPIO引脚

前面说过，根据I2C设备的地址0x38，执行如下命令：

$ cd arch/arm/boot/dts/

$ grep "@38" \* -nR

我们可以得到很多结果，打开跟imx6ull最相近的imx6ul-tx6ul.dtsi，可以看到如下代码：



我们把这个结点的内容先复制下来，粘贴到哪里去？

100ASK\_IMX6ULL也配有触摸屏，我们用的型号是gt9xx，把这个结点放到gt9xx结点相同位置去就可以了，如下图所示：

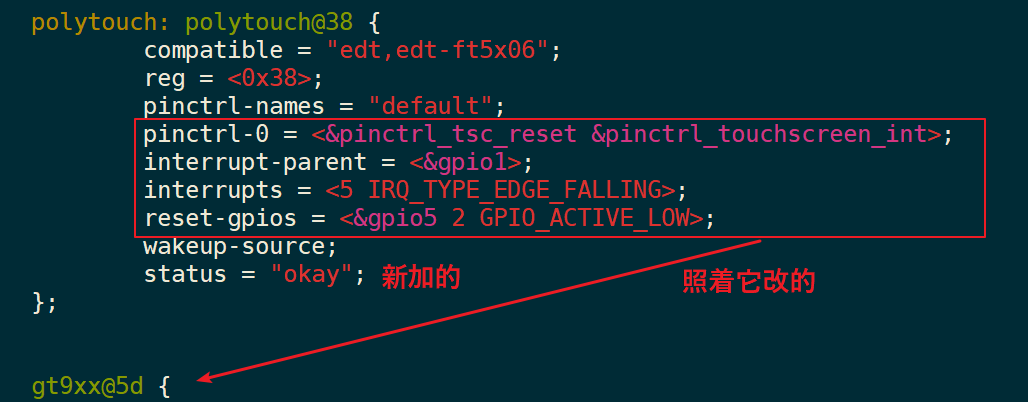


100ASK\_IMX6ULL接标配的LCD时，触摸IC是gt9xx，用的引脚假设是 AAA；

那么同一个底板接上另一块LCD时，虽然触摸IC型号不同，但是它仍然用的是同一个引脚AAA。

所以，新加的节点，其内容可以参考gt9xx节点的内容。

下图就是改好的样子：



重新编译设备树，更新到板子上，发现触摸屏还是不能用。

这还得往内核里加驱动。

4.3 重新配置内核添加驱动

这设备节点对应哪个驱动啊？它有这个属性：

compatible = "edt,edt-ft5x06"

在内核drivers/input/touchscreen目录下搜搜"edt,edt-ft5x06"：

$ cd drivers/input/touchscreen/

$ grep "edt,edt-ft5x06" \* -nr

什么都没搜到，再搜“edt-ft5x06”：

$ grep "edt-ft5x06" \* -nr

edt-ft5x06.c:1071: { .name = "edt-ft5x06", .driver\_data = (long)&edt\_ft5x06\_data },

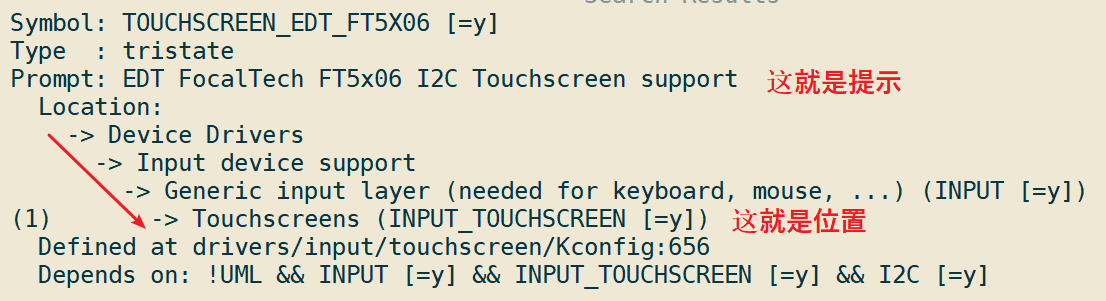
Kconfig:667: module will be called edt-ft5x06.

Makefile:31:obj-$(CONFIG\_TOUCHSCREEN\_EDT\_FT5X06) += edt-ft5x06.o

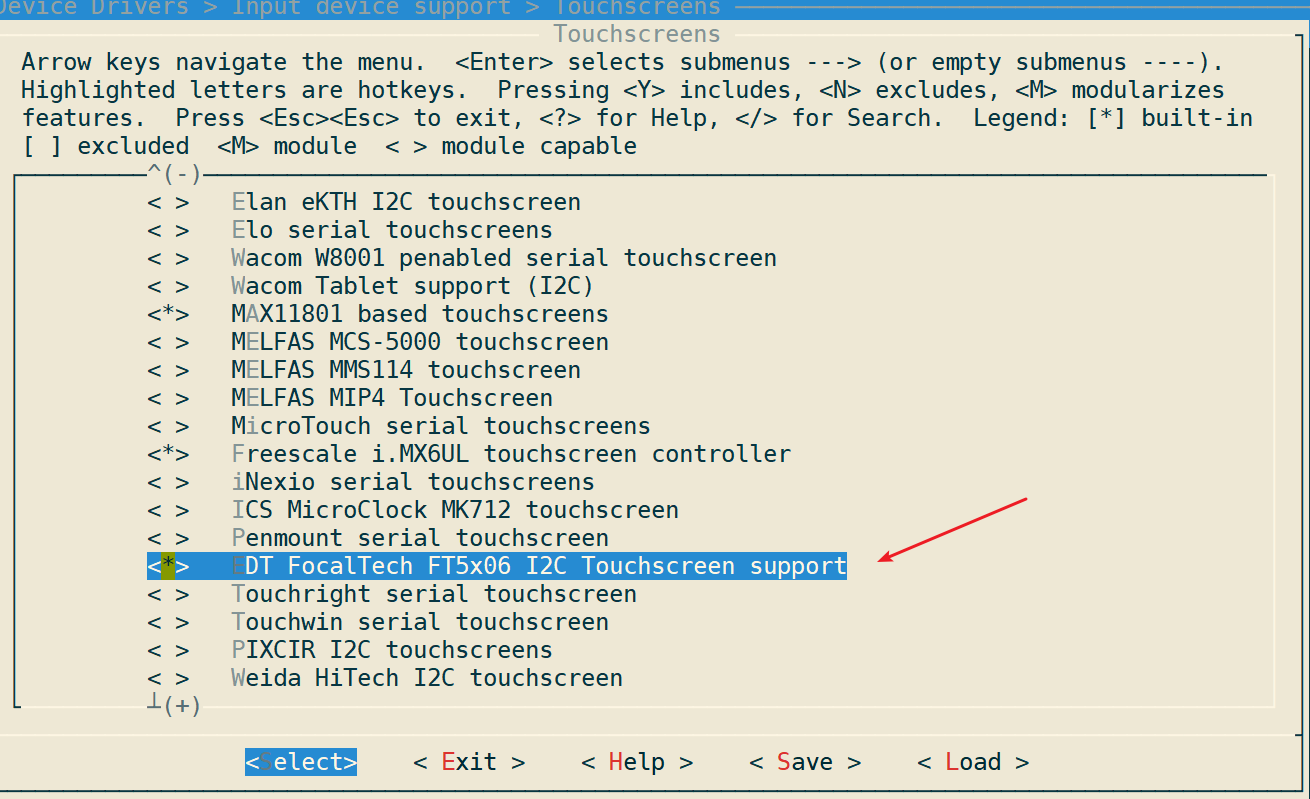
显然，驱动程序是edt-ft5x06.c，内核配置项是CONFIG\_TOUCHSCREEN\_EDT\_FT5X06。

我们需要配置内核，把CONFIG\_TOUCHSCREEN\_EDT\_FT5X06配置为y。

在内核目录下执行“make menuconfig”，然后搜“CONFIG\_TOUCHSCREEN\_EDT\_FT5X06”：



在菜单里找到它，把它配置为y，如下图：



重新编译内核zImage，更新到板子上，启动。

发现触摸屏有反应，但是点不准，还得调试。

4.4 调试：找出问题

Tslib是触摸屏的库，自带有很多工具：

a. ts\_print\_raw ：打印触摸屏原始数据

b. ts\_print ：打印经过较准的数据

c. ts\_test\_mt：测试电容屏，点击触摸屏，同时就会在LCD上显示触点位置。

我们先把系统自带的QT系统去掉，在开发板执行：

# mv /etc/init.d/S07hmi /root/

# reboot

然后设置环境变量，执行ts\_test\_mt：

export TSLIB\_TSDEVICE=/dev/input/event1

export TSLIB\_CONFFILE=/etc/ts.conf

export TSLIB\_CALIBFILE=/etc/pointercal

export TSLIB\_PLUGINDIR=/usr/lib/ts

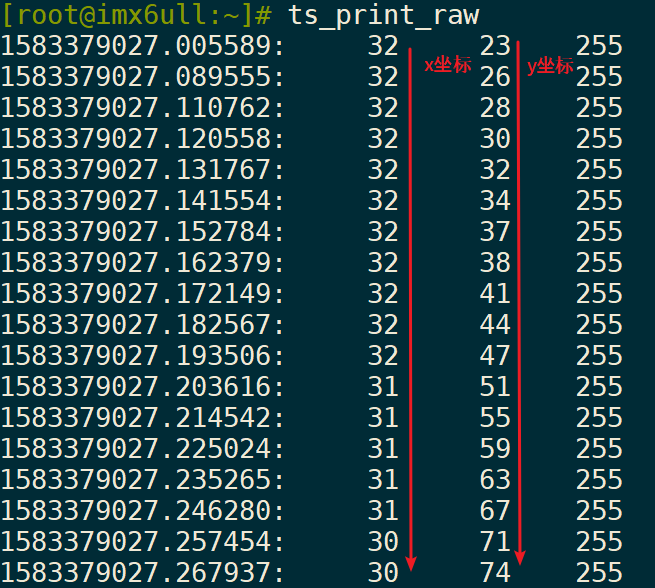
ts\_test\_mt

可以在LCD屏幕上看到提示，你点击某个位置，正常的话在该位置就会显示一个标号。

我们发现有意思的现象：从左往右点，标号从上边移动到下边；从上往下点，标号从左边移动到右边。

猜测：XY坐标对调了。

再试一下，执行 ts\_print\_raw，然后从左往右点，现象如下：



从左往右点，x坐标应该发生变化，y坐标保持不变；但是从上图看来，这是相反的。

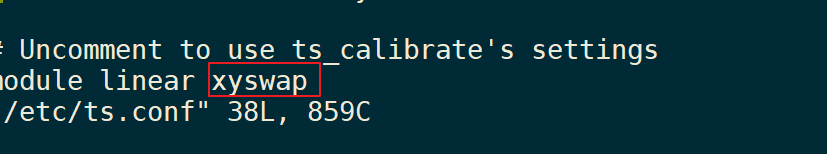
所以，确实是xy坐标对调了。

4.5 解决方法

可以从应用层面解决，也可以从驱动层面解决。二选一就好了，不要同时做。

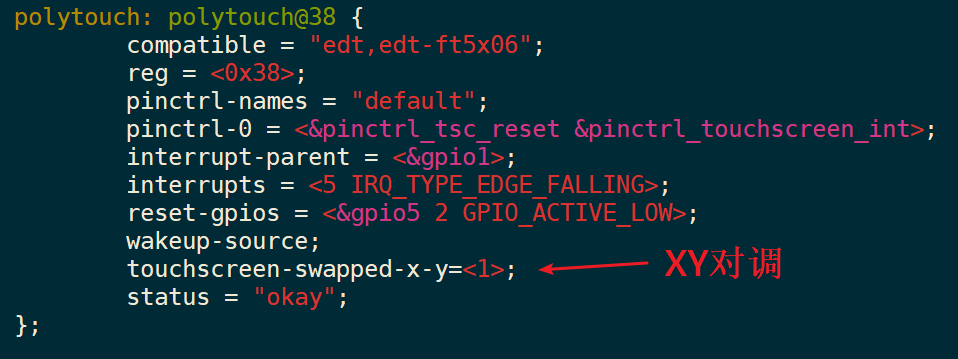
4.5.1 修改/etc/ts.conf

如下图加上xyswap就可以了：



4.5.2 修改设备树

有时候我们并不愿意、不能修改应用层的东西，那可以修改设备树：



4.6 其他情况

100ASK\_IMX6ULL标配的屏，带的触摸IC是gt9xx；但是我们发现别家的LCD即使同样使用gt9xx，但是它的xy值是反转的。

什么意思呢？

你从左往右点，正常来说x值是从小变大，但是有些屏是从大变小。

你从上往下点，正常来说y值是从小变大，但是有些屏是从大变小。

这时候，你同样可以修改设备树，或是修改/etc/ts.conf。

怎么修改设备树？

参考内核文档：Documentation/devicetree/bindings/input/touchscreen，该目录下有很多I2C触摸芯片的设备树说明，比如有goodix.txt，对应gt9xx芯片；有edt-ft5x06.txt。

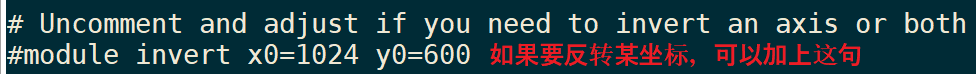
要让x反转，或是y反转，在设备节点中加入这样的属性值就可以：

touchscreen-inverted-x = <1>;

touchscreen-inverted-y = <1>;

有时为了测试方便，就是想临时改一下/etc/ts.conf，怎么做？

这个文件本身是有些注释的，可以参考：



“x0=1024”的意思就是：x坐标，0表示1024；

“y0=600”的意思是：y坐标，0表示600。

4.7 gt9xx芯片固件更新

gt9xx芯片功能强大，可以写入配置信息让它支持不同分辨率的触摸屏。

但是出厂的触摸屏IC一般都已经写好配置信息了，我们不应该让驱动程序去修改这些配置信息。

可以在设备树中加上这一句，禁止驱动去修改配置信息：

goodix,driver-send-cfg = <0>;

这是我们调试过程中碰到的一个坑。