关于4.3寸rgb屏触摸ic-GT1151的裸机例程改动

4.3寸的800x480的屏幕更换了一个新的触摸IC，原来使用的是GT9147，现在使用GT1151，这两个芯片在使用上并无太大的区别，甚至在驱动上有很多共同的地方，此次改动并非重写代码，而是在原来的基础上增加对两者不同的处理还有一些使用上的优化。

触摸芯片大概分两种：一种是内部flash比较大能够存储固件，这种芯片基本上只要上电，复位后就能工作，这种的驱动编写可以简单很多，还有一种是内部的flash比较小只能够满足其自身的初始化，固件需要其初始化之后由驱动加载下去。我们使用的GT1151和GT9147都属于前者，当然固件更新也是可以的，不过一般芯片内部默认的固件都足以使用了。

1151和9147都是使用i2c进行通信，其从机地址和大部分寄存器的地址都是相同的。从机地址都是0x14，控制寄存器都是0x8040基本的控制命令都是一样，版本号寄存器1151是0x8050而9147是0x8047。产品ID寄存器都是0x8140。五个触摸点的坐标寄存器都是一样的分别是如下的5个0x8150, 0x8158, 0x8160, 0x8168, 0x8170。

下面我们就基于之前的例程代码进行修改：

我们先在头文件bsp\_gt9147.h修改（文件路径：例程源码\bsp\touchscreen\bsp\_gt9147.h），只需要要添加一个宏定义。

|  |
| --- |
| #define GT1151\_CFGS\_REG 0X8050 /\* GT1151配置起始地址寄存器 \*/ |

接下来我们修改同文件夹下的bsp\_gt9147.c源码文件。

因为1151和9147有很多相同的地方所以这次我们需要改动的地方也很小。

首先我们先拉到gt9147\_init这个函数，我们先看看原本的。

|  |
| --- |
| void gt9147\_init**(**void**)**  **{**  unsigned char temp**[**5**];**  unsigned char regvalue**;**  gpio\_pin\_config\_t ctintpin\_config**;**  gpio\_pin\_config\_t ctretpin\_config**;**  gt9147\_dev**.**initfalg **=** GT9147\_INIT\_NOTFINISHED**;**  /\* 1、初始化IIC2 IO  \* I2C2\_SCL -> UART5\_TXD  \* I2C2\_SDA -> UART5\_RXD  \*/  IOMUXC\_SetPinMux**(**IOMUXC\_UART5\_TX\_DATA\_I2C2\_SCL**,**1**);**  IOMUXC\_SetPinMux**(**IOMUXC\_UART5\_RX\_DATA\_I2C2\_SDA**,**1**);** |

这里给gt9147\_dev这个结构体的initfalg属性赋值。接下来看看增加的部分。

|  |
| --- |
| void gt9147\_init**(**void**)**  **{**  unsigned char temp**[**5**];**  unsigned char regvalue**;**  gpio\_pin\_config\_t ctintpin\_config**;**  gpio\_pin\_config\_t ctretpin\_config**;**  gt9147\_dev**.**initfalg **=** GT9147\_INIT\_NOTFINISHED**;**  int i;  for(i = 0; i < 5; i++)  {  gt9147\_dev.x[i] = 0; //避免编译器自动赋值  gt9147\_dev.y[i] = 0;  }  /\* 1、初始化IIC2 IO  \* I2C2\_SCL -> UART5\_TXD  \* I2C2\_SDA -> UART5\_RXD  \*/  IOMUXC\_SetPinMux**(**IOMUXC\_UART5\_TX\_DATA\_I2C2\_SCL**,**1**);**  IOMUXC\_SetPinMux**(**IOMUXC\_UART5\_RX\_DATA\_I2C2\_SDA**,**1**);** |

增加的部分是为了避免编译器优化自动赋值导致上电程序运行屏幕输出奇怪的值。

往下看

|  |
| --- |
| gt9147\_read\_len**(**GT9147\_ADDR**,** GT\_PID\_REG**,** 4**,** temp**);**/\* 读取产品ID \*/  temp**[**4**]** **=** 0**;**  printf**(**"CTP ID:%s\r\n"**,** temp**);** /\* 打印ID \*/  gt9147\_write\_byte**(**GT9147\_ADDR**,** GT\_CTRL\_REG**,** 0x02**);**  regvalue **=** gt9147\_read\_byte**(**GT9147\_ADDR**,** GT\_CFGS\_REG**);**  printf**(**"Default Ver:%#x\r\n"**,**regvalue**);**  gt9147\_send\_cfg**(**0**);**/\* 配置GT9147 \*/  delayms**(**10**);**  gt9147\_write\_byte**(**GT9147\_ADDR**,** GT\_CTRL\_REG**,** 0**);** |

1151和9147的产品id寄存器是相同的所以我们可以不用动，但是它们存放固件版本号的寄存器是不一样所以我们要做处理。在裸机例程这里是有给9147下发新固件的，但是由于flash内部的版本号更高所以并不会生效，对于1151，我们直接使用其内部的固件即可。

|  |
| --- |
| gt9147\_read\_len**(**GT9147\_ADDR**,** GT\_PID\_REG**,** 4**,** temp**);**/\* 读取产品ID \*/  temp**[**4**]** **=** 0**;**  printf**(**"CTP ID:%s\r\n"**,** temp**);** /\* 打印ID \*/  gt9147\_write\_byte**(**GT9147\_ADDR**,** GT\_CTRL\_REG**,** 0x02**);**  **if(**temp**[**0**]==**'1'**)**  regvalue **=** gt9147\_read\_byte**(**GT9147\_ADDR**,** GT1151\_CFGS\_REG**);**  **else**  **{**  regvalue **=** gt9147\_read\_byte**(**GT9147\_ADDR**,** GT\_CFGS\_REG**);**  gt9147\_send\_cfg**(**0**);**/\* 配置GT9147 \*/  delayms**(**10**);**  **}**  printf**(**"Default Ver:%#x\r\n"**,**regvalue**);**  gt9147\_write\_byte**(**GT9147\_ADDR**,** GT\_CTRL\_REG**,** 0**);** |

针对两个不同型号分别进行处理。最后给控制寄存器写0是让触摸芯片进去读取坐标工作模式。这样gt9147\_init函数就改完了。接下来我们对gt9147\_read\_tpcoord函数进行一些优化。

|  |
| --- |
| void gt9147\_read\_tpcoord**(**void**)**  **{**  u8 buf**[**4**];**  u8 i **=** 0**;**  u8 regvalue **=** 0**;**  regvalue **=** gt9147\_read\_byte**(**GT9147\_ADDR**,** GT\_GSTID\_REG**);**  gt9147\_write\_byte**(**GT9147\_ADDR**,** GT\_GSTID\_REG**,** 0x00**);**  /\* 读取5点触摸坐标值 \*/  **for(**i **=** 0**;** i **<** 5**;** i**++)**  **{**  gt9147\_read\_len**(**GT9147\_ADDR**,** GT9147\_TPX\_TBL**[**i**],** 4**,** buf**);** /\* 读取坐标值 \*/  gt9147\_dev**.**x**[**i**]** **=** **((**u16**)**buf**[**1**]** **<<** 8**)** **+** buf**[**0**];**  gt9147\_dev**.**y**[**i**]** **=** **(((**u16**)**buf**[**3**]** **<<** 8**)** **+** buf**[**2**]);**  **}**  gt9147\_dev**.**point\_num **=** regvalue **&** 0XF**;**  **}** |

regvalue这个值是读取芯片检测到了几个触摸点，下面的gt9147\_write\_byte写一个0值是为了让坐标寄存器的坐标值能更新，不写0的话坐标寄存器的值就不会更新，我在借鉴Linux下的驱动的时候发现，这句写在读完寄存器后面会比较恰当一点，但测试并没有影响。原本读坐标值是直接循环读取5个，但是测试发现这样在1151会有一些问题。正确的做法应该是读取到几个就读取多少个。改动后如下。

|  |
| --- |
| void gt9147\_read\_tpcoord**(**void**)**  **{**  u8 buf**[**4**];**  u8 i **=** 0**;**  u8 regvalue **=** 0**;**  regvalue **=** gt9147\_read\_byte**(**GT9147\_ADDR**,** GT\_GSTID\_REG**);**  gt9147\_write\_byte**(**GT9147\_ADDR**,** GT\_GSTID\_REG**,** 0x00**);**  gt9147\_dev**.**point\_num **=** regvalue **&** 0XF**;**  /\* 读取5点触摸坐标值 \*/  **for(**i **=** 0**;** i **<** gt9147\_dev**.**point\_num**;** i**++)**  **{**  gt9147\_read\_len**(**GT9147\_ADDR**,** GT9147\_TPX\_TBL**[**i**],** 4**,** buf**);** /\* 读取坐标值 \*/  gt9147\_dev**.**x**[**i**]** **=** **((**u16**)**buf**[**1**]** **<<** 8**)** **+** buf**[**0**];**  gt9147\_dev**.**y**[**i**]** **=** **(((**u16**)**buf**[**3**]** **<<** 8**)** **+** buf**[**2**]);**  **}**    **}** |

# Linux下的改动

Linux下建议将例程代码里的更新固件代码注释掉，因为出厂的屏幕自带的驱动已经满足使用了，并且驱动代码里面的固件版本比芯片内部的旧即使下载下去也并不会生效。注释掉，可以防止其对1151的干扰。

注释更新固件代码操作：源码文件路径（资料A盘\1、例程源码\2、Linux驱动例程\23\_multitouch\gt9147.c）

代码段位置在gt9147\_probe函数里面

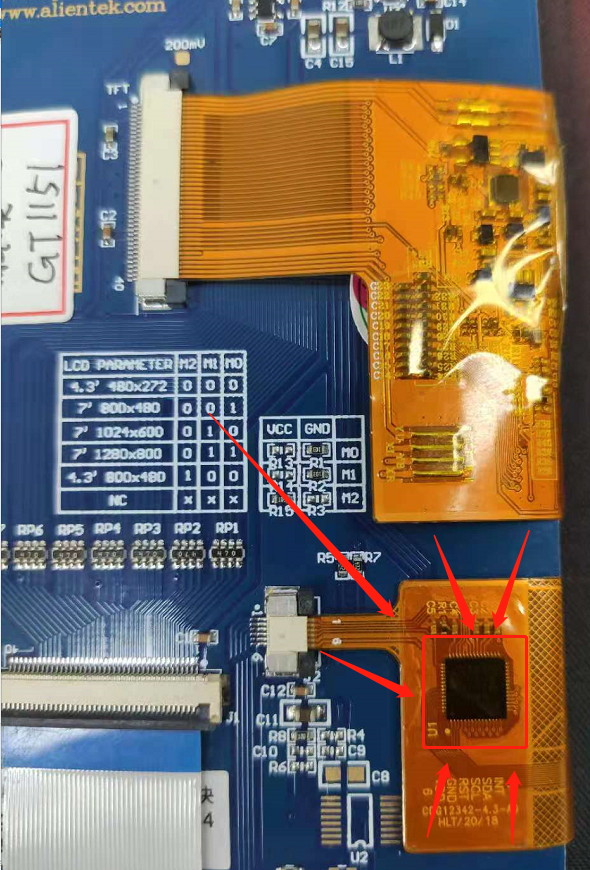
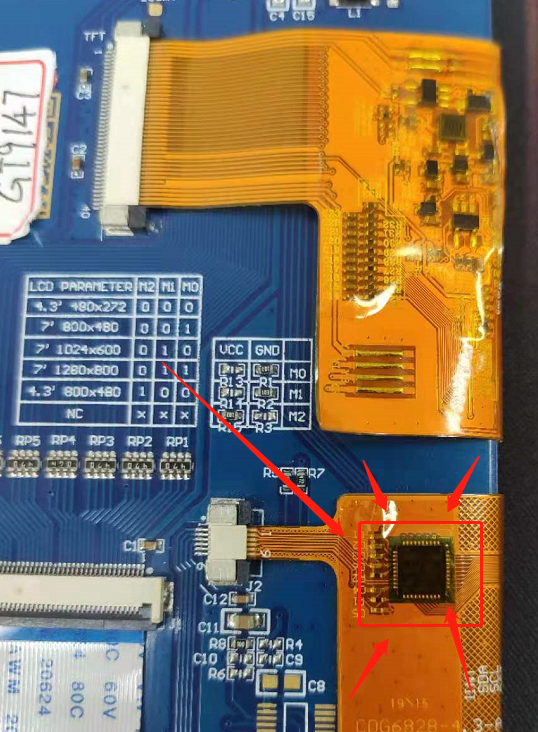
|  |
| --- |
| /\* 4,初始化GT9147，烧写固件 \*/  gt9147\_read\_regs**(&**gt9147**,** GT\_CFGS\_REG**,** **&**data**,** 1**);**  printk**(**"GT9147 ID =%#X\r\n"**,** data**);**  **if(**data **<** GT9147\_CT**[**0**])** **{**  gt9147\_send\_cfg**(&**gt9147**,** 0**);**  **}** |

假设注释后驱动加载成功触摸屏幕hexdump /dev/input/event2（eventX这个X是你自己板子上对应的触摸事件，根据自己的实际情况进行调整）。并无数据生成的话，修改一下设备树，教程文档上将GPIO1-09的引脚的IO配置配成**0xF080**，我们只需要将0xF080改成**0x49**即可成功驱动触摸芯片。



# 4.3寸屏触摸IC的区分

4.3寸屏的触摸IC的区分有两个方法：第一个看屏幕背面排线部分，具体如下面两图，触摸IC在屏幕排线的右边的那个小方块。左边GT9147的芯片大小较小在芯片周边会有一点点的绿色的包围，右边GT1151的芯片大小较大，不会有绿色的包围。



第二个方法就是在出厂系统启动的时候看内核的启动日志。内核在加载触摸芯片的驱动的时候会去读取触摸芯片内部的型号id，如下图当读取到的ID是9147那触摸芯片的型号就是9147，当读取到的是115X的时候触摸芯片就是1151。

