**NanoPI2-LCD驱动移植**

2018.3.22

周星宇

## 2.LCD设备

**2.1 LCD设备描述**

NanoPi2支持的所有LCD型号使用一个结构体来存储nanopi2\_lcd\_config

arch\arm\plat-s5p4418\nanopi2\lcds.c

1. /\* Try to guess LCD panel by kernel command line, or
2. \* using \*HD101\* as default
3. \* \linux-3.4.y-nanopi2-lollipop-mr1\arch\arm\plat-s5p4418\nanopi2\lcds.c
4. \*/
5. **static** **struct** {
6. **char** \*name;
7. **struct** nxp\_lcd \*lcd;
8. **int** ctp;
9. } nanopi2\_lcd\_config[] = {
10. { "HD101",  &wxga\_hd101, 1 },
11. { "HD101B", &wxga\_hd101, CTP\_GOODIX  },
12. { "HD700",  &wxga\_hd700, 1 },
13. { "HD702",  &wxga\_hd700, CTP\_GOODIX  },
14. { "S70",    &wvga\_s70,   1 },
15. { "S702",   &wvga\_s702,  1 },
16. { "S70D",   &wvga\_s70d,  0 },
17. { "X710",   &wsvga\_x710, CTP\_ITE7260 },
18. { "S430",   &wvga\_s430,  CTP\_HIMAX   },
19. #ifndef CONFIG\_ANDROID
20. { "H43",    &hvga\_h43,   0 },
21. { "P43",    &hvga\_p43,   0 },
22. { "W35",    &qvga\_w35,   0 },
23. #endif
24. /\* TODO: Testing \*/
25. { "W50",    &wvga\_w50,   0 },
26. { "W101",   &wsvga\_w101, 1 },
27. { "A97",    &xga\_a97,    0 },
28. { "LQ150",  &xga\_lq150,  1 },
29. { "L80",    &vga\_l80,    1 },
30. { "BP101",  &wxga\_bp101, 1 },
31. { "HDMI",   &hdmi\_def,   0 },   /\* Pls keep it at last \*/
32. };

其中存储了每种LCD的名称与驱动参数。

第一列：LCD名字

第二列：LCD参数结构体

第三列：使用触摸芯片型号

下面用wsvga\_x710型号的LCD举例。

1. /\*
2. \* \linux-3.4.y-nanopi2-lollipop-mr1\arch\arm\plat-s5p4418\nanopi2\lcds.c
3. \*/
4. **static** **struct** nxp\_lcd wsvga\_x710 = {
5. .width= 1024,
6. .height = 600,
7. .p\_width = 154,
8. .p\_height = 90,
9. .bpp = 24,
10. .freq = 61,
11. .timing = {
12. .h\_fp = 84,
13. .h\_bp = 84,
14. .h\_sw = 88,
15. .v\_fp = 10,
16. .v\_fpe = 1,
17. .v\_bp = 10,
18. .v\_bpe = 1,
19. .v\_sw = 20,
20. },
21. .polarity = {
22. .rise\_vclk = 0,
23. .inv\_hsync = 1,
24. .inv\_vsync = 1,
25. .inv\_vden = 0,
26. },
27. .gpio\_init = hd101\_gpio\_init,
28. };

其中各参数的含义如下：

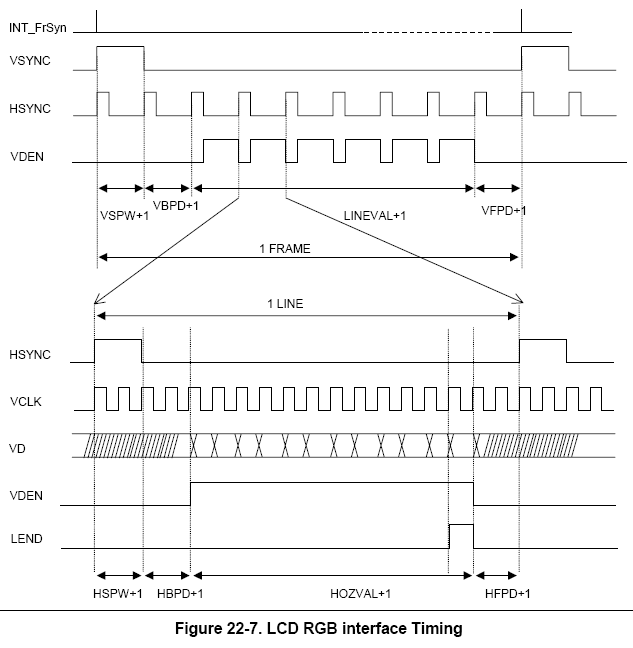
1. /\*
2. \* \linux-3.4.y-nanopi2-lollipop-mr1\arch\arm\plat-s5p4418\nanopi2\include
3. \* struct nxp\_lcd
4. \* @width:      horizontal resolution <水平分辨率>
5. \* @height:     vertical resolution <垂直分辨率>
6. \* @p\_width:    width of lcd in mm <水平物理尺寸>
7. \* @p\_height:   height of lcd in mm <垂直物理尺寸>
8. \* @bpp:        bits per pixel <每个像素要用多少位(24)>
9. \* @freq:       vframe frequency <刷新帧率(61)>
10. \* @timing:     timing values <存储时序参数的结构体>
11. \* @polarity:   polarity settings <结构体，存储信号线极性>
12. \* @gpio\_init:  pointer to GPIO init function <GPIO初始化函数>
13. \*
14. \*/
15. **struct** nxp\_lcd {
16. **int** width;
17. **int** height;
18. **int** p\_width;
19. **int** p\_height;
20. **int** bpp;
21. **int** freq;
22. **struct**  nxp\_lcd\_timing timing;
23. **struct**  nxp\_lcd\_polarity polarity;
24. **void**    (\*gpio\_init)(**void**);
25. };

LCD时序参数结构体如下：

1. /\*
2. \* struct nxp\_lcd\_timing
3. \* @h\_fp:   horizontal front porch
4. \* @h\_bp:   horizontal back porch
5. \* @h\_sw:   horizontal sync width
6. \* @v\_fp:   vertical front porch
7. \* @v\_fpe:  vertical front porch for even field
8. \* @v\_bp:   vertical back porch
9. \* @v\_bpe:  vertical back porch for even field
10. \*/
11. **struct** nxp\_lcd\_timing {
12. **int** h\_fp;
13. **int** h\_bp;
14. **int** h\_sw;
15. **int** v\_fp;
16. **int** v\_fpe;
17. **int** v\_bp;
18. **int** v\_bpe;
19. **int** v\_sw;
20. };

时序参数的含义参考LCD控制时序。

参数确定方法见《LCD驱动时序参数的确定.docx》



信号线极性结构体定义了个别信号线是否反转。

是否反转也可以在LCD时序图中看出来

1. /\*
2. \* struct nxp\_lcd\_polarity
3. \* @rise\_vclk:  if 1, video data is fetched at rising edge
4. \* @inv\_hsync:  if HSYNC polarity is inversed
5. \* @inv\_vsync:  if VSYNC polarity is inversed
6. \* @inv\_vden:   if VDEN polarity is inversed
7. \*/
8. **struct** nxp\_lcd\_polarity {
9. **int** rise\_vclk;
10. **int** inv\_hsync;
11. **int** inv\_vsync;
12. **int** inv\_vden;
13. };

GPIO初始化函数中定义了引脚的输出强度

## 3.触摸

**3.1 编译驱动**

修改.config文件，添加触摸屏驱动支持

1. # CONFIG\_TOUCHSCREEN\_TOUCHIT213 is not set
2. # CONFIG\_TOUCHSCREEN\_TSC\_SERIO is not set
3. # CONFIG\_TOUCHSCREEN\_TSC2005 is not set
4. # CONFIG\_TOUCHSCREEN\_TSC2007 is not set
5. # CONFIG\_TOUCHSCREEN\_ST1232 is not set
6. # CONFIG\_TOUCHSCREEN\_TPS6507X is not set
7. CONFIG\_TOUCHSCREEN\_FT5X0X=y

**3.2 定义I2c设备**

触摸屏相关的I2C设备在

linux-3.4.y-nanopi2-lollipop-mr1\arch\arm\plat-s5p4418\nanopi2\device.c

中定义。

1. #if defined(CONFIG\_TOUCHSCREEN\_IT7260)
2. #define IT7260\_I2C\_BUS      (2)
4. **static** **struct** i2c\_board\_info \_\_initdata it7260\_i2c\_bdi = {
5. I2C\_BOARD\_INFO("IT7260", (0x8C>>1)),
6. .irq = PB\_PIO\_IRQ(CFG\_IO\_TOUCH\_IRQ),
7. };
8. #endif

其中需要注意：

1.I2C设备的地址需要改成触摸芯片手册中的(例子中为0x8C>>1)

2."IT7260"为I2C设备名称，需要和驱动IT7260设备驱动程序中的驱动名称一致。

3. 不同触摸芯片型号需要填充I2c\_board\_info结构体中的不同成员，(如tsc2007需要填充device成员)。

驱动程序定义如下所示：

1. //linux-3.4.y-nanopi2-lollipop-mr1\drivers\input\touchscreen\it7260\_mts.c
2. **static** **const** **struct** i2c\_device\_id it7260\_ts\_id[] = {
3. {"IT7260", 0},
4. {}          /\* should not omitted \*/
5. };
6. MODULE\_DEVICE\_TABLE(i2c, it7260\_ts\_id);
8. **static** **struct** i2c\_driver it7260\_ts\_driver = {
9. .driver = {
10. .name = "IT7260-ts",
11. },
12. .probe = it7260\_ts\_probe,
13. .remove = \_\_devexit\_p(it7260\_ts\_remove),
14. .suspend = it7260\_ts\_suspend,
15. .resume = it7260\_ts\_resume,
16. .id\_table = it7260\_ts\_id,
17. };

设备驱动程序一般情况下linux内核的drivers目录中会自带，也很少需要我们去修改。只要修改.config文件将它们编译进内核就可以了。

下面介绍一个移植设备驱动的诀窍

诀窍：如何编写设备注册程序可参考linux内核源码中未编译进内核的其他平台的实现(如linux-3.4.y-nanopi2-lollipop-mr1\arch\arm\mach-imx\mach-cpuimx51sd.c中就有tsc2007设备的注册，如下图所示)

1. //linux-3.4.y-nanopi2-lollipop-mr1\arch\arm\mach-imx\mach-cpuimx51sd.c
2. **static** **struct** tsc2007\_platform\_data tsc2007\_info = {
3. .model          = 2007,
4. .x\_plate\_ohms       = 180,
5. .get\_pendown\_state  = ts\_get\_pendown\_state,
6. .init\_platform\_hw   = ts\_init,
7. };
9. **static** **struct** i2c\_board\_info tsc\_device = {
10. I2C\_BOARD\_INFO("tsc2007", 0x48),
11. .type       = "tsc2007",
12. .platform\_data  = &tsc2007\_info,
13. /\*.irq is selected on ap4evb\_init \*/
14. };

这样一来器件I2C地址也不需要去芯片手册找了。

**3.3 触摸屏设备加载**

在开机初始化函数nxp\_board\_devices\_register(**void**)中加载3.2中定义的触摸触摸屏设备。

1. //linux-3.4.y-nanopi2-lollipop-mr1\arch\arm\plat-s5p4418\nanopi2\device.c
2. **void** \_\_init nxp\_board\_devices\_register(**void**){
3. //.................
4. #if defined(CONFIG\_TOUCHSCREEN\_FT5X0X)
5. printk("plat: add touch(ft5x0x) device\n");
6. ft5x0x\_pdata.screen\_max\_x = lcd->width;
7. ft5x0x\_pdata.screen\_max\_y = lcd->height;
8. i2c\_register\_board\_info(FT5X0X\_I2C\_BUS, &ft5x0x\_i2c\_bdi, 1);
9. #endif
11. #if defined(CONFIG\_TOUCHSCREEN\_GOODIX)
12. printk("plat: add touch(goodix) device\n");
13. goodix\_pdata.screen\_max\_x = lcd->width;
14. goodix\_pdata.screen\_max\_y = lcd->height;
15. i2c\_register\_board\_info(GOODIX\_I2C\_BUS, &goodix\_i2c\_bdi, 1);
16. printk("plat: goodix: irq=%d (%d)\n", PB\_PIO\_IRQ(CFG\_IO\_TOUCH\_IRQ), CFG\_IO\_TOUCH\_IRQ);
17. #endif
19. #if defined(CONFIG\_TOUCHSCREEN\_HIMAX)
20. printk("plat: add touch(himax) device\n");
21. i2c\_register\_board\_info(HIMAX\_I2C\_BUS, &himax\_i2c\_bdi, 1);
22. #endif
24. #if defined(CONFIG\_TOUCHSCREEN\_IT7260)
25. printk("plat: add touch(it7260) device\n");
26. i2c\_register\_board\_info(IT7260\_I2C\_BUS, &it7260\_i2c\_bdi, 1);
27. #endif
28. //.......................

**3.4 修改IDC文件**

如果是linux系统，上面的工作做完之后已经ok。

但是android系统上还需要修改.idc文件，如果不修改的话会变成触摸鼠标

参考：

<http://blog.chinaunix.net/uid-149881-id-4347448.html>

## 4.背光驱动