

内核驱动LED

内核驱动LED

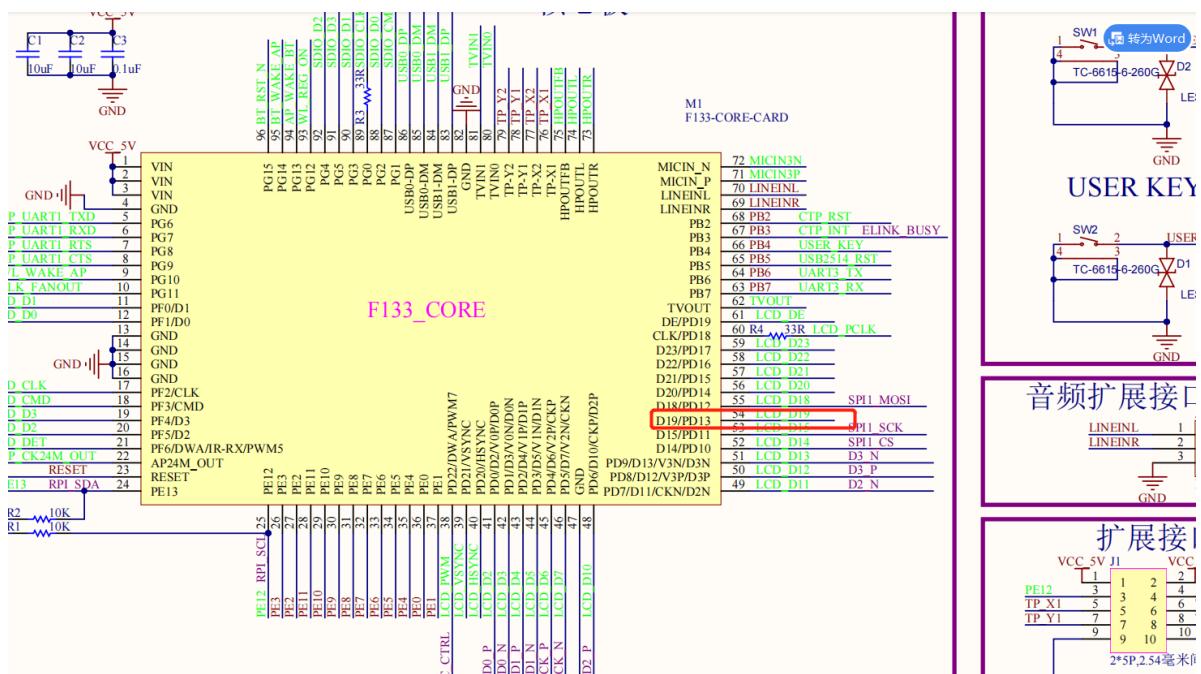
- 1.确定使用的IO口并准备硬件
 - 2.修改设备树
 - 2.1.什么是设备树？为什么引入设备树？
 - 2.2.修改设备树实操：
 - 3.裁剪LINUX内核，编译并烧写
 - 3.1.裁剪内核
 - 3.2.编译内核
 - 3.3.烧写SD卡
 - 4.验证设备挂载设备树成功
 - 5.命令点亮LED，编写最简单的应用程序

实现步骤：

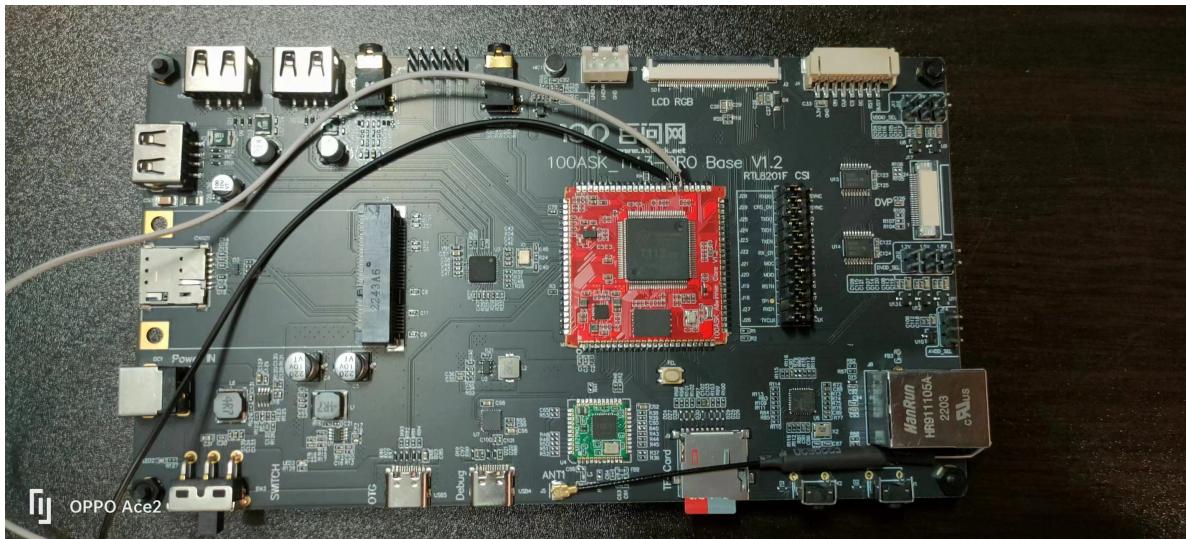
- 1.确定使用的IO口并准备硬件
 - 2.修改设备树
 - 3.裁剪LINUX内核，编译并烧写
 - 4.验证设备挂载设备树成功(该步骤后续将不再进行)
 - 5.命令点亮LED，编写最简单的应用程序

1. 确定使用的IO口并准备硬件

因为使用的开发板上并没有专门留一个IO给LED（就没有LED），这里选用PD13作为本次现象的引脚，为第54号引脚，直接数芯片引脚号就行。



飞线如下(LED用的是白色的那根线，黑色的线是后续用于DHT11的数据总线，可以一并飞了):



2.修改设备树

2.1.什么是设备树？为什么引入设备树？

对于一个设备(如LED)的驱动厂商而言，一个完善的设备驱动应支持多种不同的单板，而同一款芯片可能有成千上万款不同的单板，不同的单板所板载的外设不同，如单板ALED对应引脚为A1，单板BLED对应引脚为A2，单板A可以在boardA.c中对使用的引脚进行指定，而单板B可以在boardB.c中对使用的引脚进行指定，这就引发了两个问题：

1.引脚配置信息是定义于boardX.c文件中的，每次更换使用的引脚都需要对.c文件进行编译，并且需要重新安装驱动程序

2.大量厂家进入市场且厂家代码编写水平层次不齐，他们使用某款芯片对他们自己家的单板进行驱动开发，将可能产生一个含有大量屎山代码的boardX.c文件，越来越多的这种boardX.c堆积，这使得整个linux内核变得非常臃肿。

于是设备树文件便被引入，设备树(dts)实质其实是一个配置文件，为描述设备所形成的“树”而起名设备树文件。

2.2.修改设备树实操：

Buildroot 是一个开源工具，用于为嵌入式系统自动化构建 Linux 系统。它能够自动构建所有必要的组件，包括交叉编译工具链、根文件系统、内核镜像和引导程序等。在构建过程中，Buildroot 提供了一套配置系统，使得用户能够方便地指定所需的软件包、配置选项、目标平台等参数。Buildroot 的优点包括构建速度快、易于使用、支持多种嵌入式平台、支持定制化配置等。它可以在 Linux、macOS 和 Windows 系统上运行。

防止看这个文档的人没有**buildroot**框架，这里就给出代码的git获取命令：

```

git clone https://gitee.com/weidongshan/buildroot-100ask_t113-pro

cd buildroot-100ask_t113-pro/

git submodule update --init --recursive

git submodule update --recursive --remote

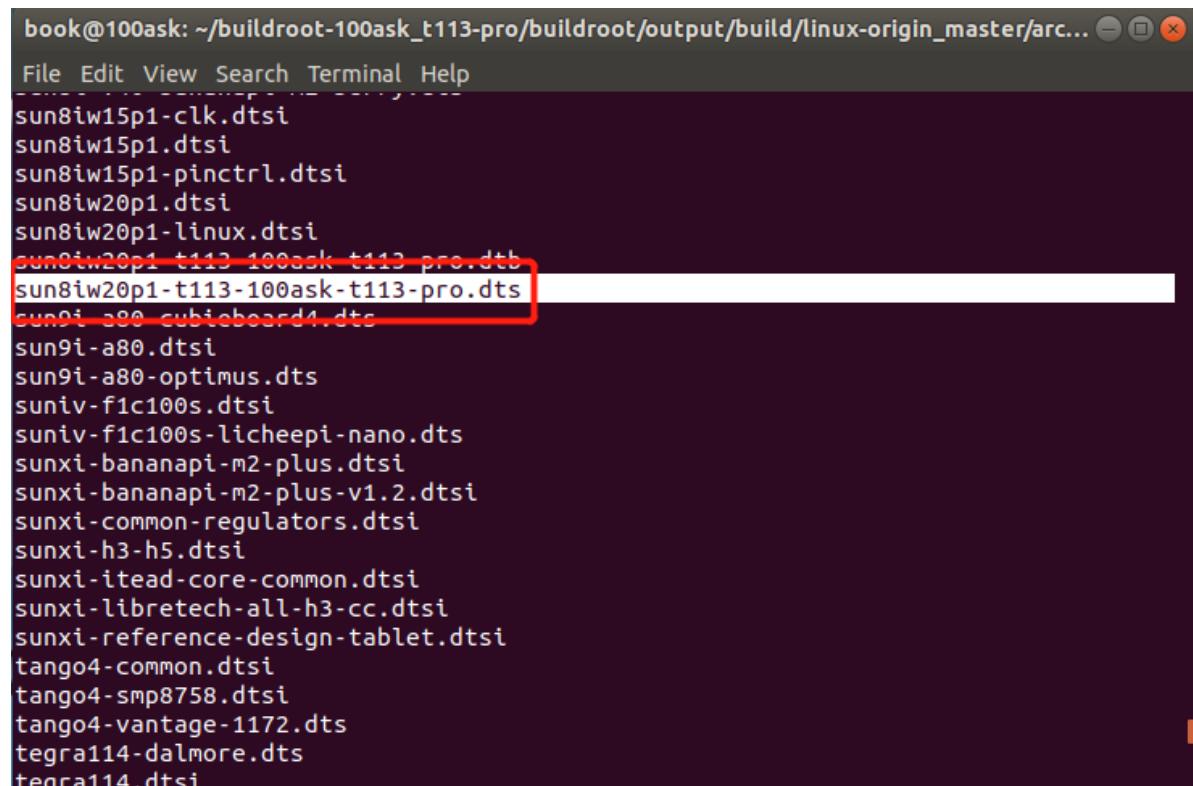
cd buildroot/

git submodule update --init --recursive

```

进入到**/buildroot-100ask_t113-pro/buildroot/output/build/linux-origin_master/arch/arm/boot/dts**目录下，需要修改的是以下的这个设备树：sudo gedit **sun8iw20p1-t113-100ask-t113-pro.dts**进行修改

```
sudo gedit sun8iw20p1-t113-100ask-t113-pro.dts
```



如果你进去了这个设备树文件中，就会看到这部分设备树文本，这部分文本的意思如注释所示：

```

/ { /*表示根节点设备*/
    model = "sun8iw20"; /*表示使用sun8iw20的源码进行初始化*/
    compatible = "allwinner,r528", "arm,sun8iw20p1"; /*兼容列表，表示如果没有找到
model的源码，则会使用另外这两种源码*/
    reg_vdd_cpu: vdd-cpu { /*这段设备树文档描述的是一个用于 CPU 的电压稳压器*/
        compatible = "sunxi-pwm-regulator"; /*表示该电压稳压器使用 sunxi-pwm-
regulator 驱动程序*/
        pwms = <&pwm 3 5000 0>; /*表示该电压稳压器使用ID为3的PWM控制器来进行控制，频率为
5000Hz，初始占空比为0。*/
        regulator-name = "vdd_cpu"; /*该电压稳压器的名称为 "vdd_cpu"*/
        regulator-min-microvolt = <810000>; /*该电压稳压器的最小输出电压为 810000 微伏
*/
    }
}

```

```

regulator-max-microvolt = <1160000>; /*该电压稳压器的最大输出电压为 1160000 微
伏*/
regulator-ramp-delay = <25>; /*该电压稳压器的输出电压从低到高或从高到低的过渡时间为 25 微秒*/
regulator-always-on; /*该电压稳压器一直处于打开状态*/
regulator-boot-on; /*该电压稳压器在系统启动时自动打开*/
status = "okay"; /*该电压稳压器状态正常*/
};

reg_usb1_vbus: usb1-vbus { /*这段设备树文档描述的是一个用于控制 USB1 的 VBUS 电压的电压稳压器*/
    compatible = "regulator-fixed"; /*表示该电压稳压器是一个固定输出电压的电压稳压器*/
    regulator-name = "usb1-vbus"; /*表示该电压稳压器的名称为 "usb1-vbus"*/
    regulator-min-microvolt = <5000000>; /*表示该电压稳压器的输出电压为 5V*/
    regulator-max-microvolt = <5000000>; /*表示该电压稳压器的最大输出电压也是5V*/
    regulator-enable-ramp-delay = <1000>; /*表示该电压稳压器的使能电平由低到高或从高到低的过渡时间为 1000 微秒*/
    gpio = <&pio PB 3 GPIO_ACTIVE_HIGH>; /*表示使用 PB3 引脚作为 GPIO 控制引脚*/
    enable-active-high; /*表示稳压器使能引脚的活动电平是高电平，也就是高电平时该电压稳压器处于工作状态。*/
};


```



```

sun8iw20p1-t113-100ask-t113-pro.dts
~/buildroot-100ask_t113-pro/build/origin_master/arch/arm/boot/dts

/{
    model = "sun8iw20";
    compatible = "allwinner,r528", "arm,sun8iw20p1";

    reg_vdd_cpu: vdd-cpu {
        compatible = "sunxi-pwm-regulator";
        pwms = <&pwm 3 5000 0>;
        regulator-name = "vdd_cpu";
        regulator-min-microvolt = <810000>;
        regulator-max-microvolt = <1160000>;
        regulator-ramp-delay = <25>;
        regulator-always-on;
        regulator-boot-on;
        status = "okay";
    };

    reg_usb1_vbus: usb1-vbus {
        compatible = "regulator-fixed";
        regulator-name = "usb1-vbus";
        regulator-min-microvolt = <5000000>;
        regulator-max-microvolt = <5000000>;
        regulator-enable-ramp-delay = <1000>;
        gpio = <&pio PB 3 GPIO_ACTIVE_HIGH>;
        enable-active-high;
    };
};


```

在根节点下我们添加以下的设备树节点：(这个就是我们LED所需要的设备树节点)

```

dtsleds {
    compatible = "gpio-leds";
    led0 {
        label = "red";
        gpios = <&pio PD 13 GPIO_ACTIVE_HIGH>;
        default-state = "off";
    };
};


```

贴出代码：(缩减要对齐，只能用TAB进行缩进)

```
dtsleds {  
    compatible = "gpio-leds"; /*表示该设备使用 GPIO 控制 LED。*/  
    led0 { /*表示该设备中的一个 LED, 可以使用多个 ledN 来表示不同的 LED*/  
        label = "red"; /*"red" 表示该 LED 的标签为 "red"*/  
        gpios = <&pio PD 13 GPIO_ACTIVE_HIGH>; /*表示使用 PD13 引脚作为 GPIO 控制引脚, 并且高电平时 LED 点亮*/  
        default-state = "off"; /*表示该 LED 默认状态为关闭(灭), 也就是启动时 LED 处于关闭状态。可以使用 "on" 来设置 LED 默认状态为点亮*/  
    };  
};
```

实在不知道怎么改就直接下载我百度网盘改好的：

链接：<https://pan.baidu.com/s/1NHV6zQgA-kk3e7b3dwjjyA>

提取码：1234

然后扔到/buildroot-100ask_t113-pro/buildroot/output/build/linux-origin_master/arch/arm/boot/dts目录下

3.裁剪LINUX内核，编译并烧写

3.1.裁剪内核

进入到/buildroot-100ask_t113-pro/buildroot/output/build/linux-origin_master目录下

```
make menuconfig
```

内核裁剪菜单路径参考：(关于内核裁剪操作我默认你们会了，Y选中拉入内核，N去除出内核，按Esc可以一直按到退出)

```
Device Drivers -> LED LED Support -> LED Support for GPIO connected LEDs(*)
```

3.2.编译内核

进入到/buildroot目录下，输入以下两条命令即可在/output/images目录下找到生成的sd卡镜像，该命令用到了buildroot框架。

```
make linux-rebuild v=1  
make v=1
```

3.3.烧写SD卡

准备一张SD卡，16G-32G都可

链接：<https://pan.baidu.com/s/1jHU54YqkzqjH-YEwCnh5tw>

提取码：1234

格式化工具和烧写工具都准备好了，至于教程默认你们都会烧写SD卡了

4. 验证设备挂载设备树成功

```
cd /sys/firmware/devicetree  
cd base  
ls led0
```

可以看到设备树目录(devicetree)下挂载了一个dtsleds设备节点，该设备节点内有led0设备

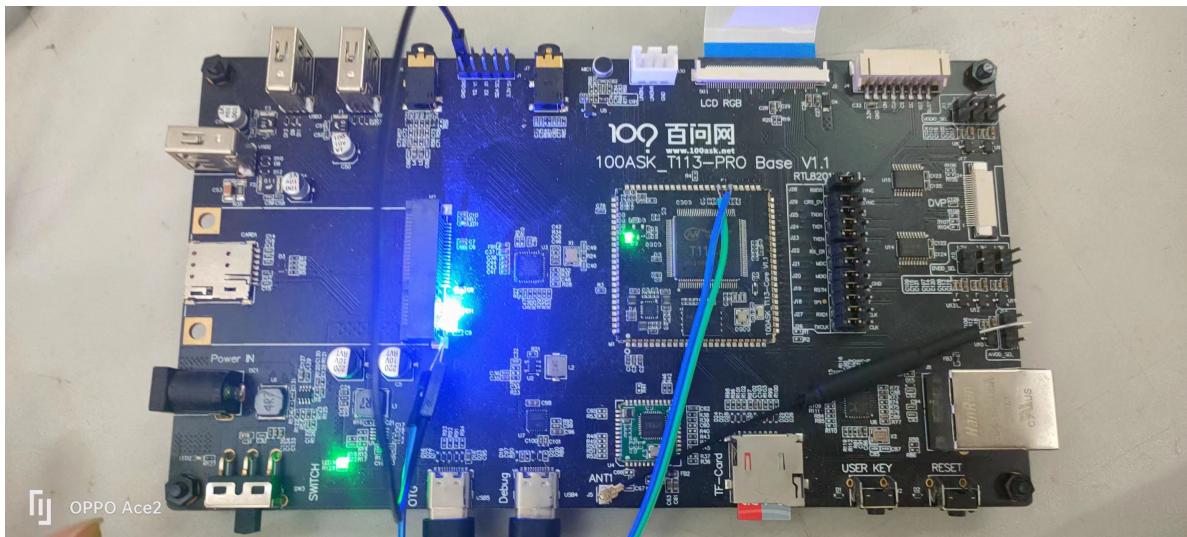
5. 命令点亮LED，编写最简单的应用程序

LED采用内核点灯的方式驱动：

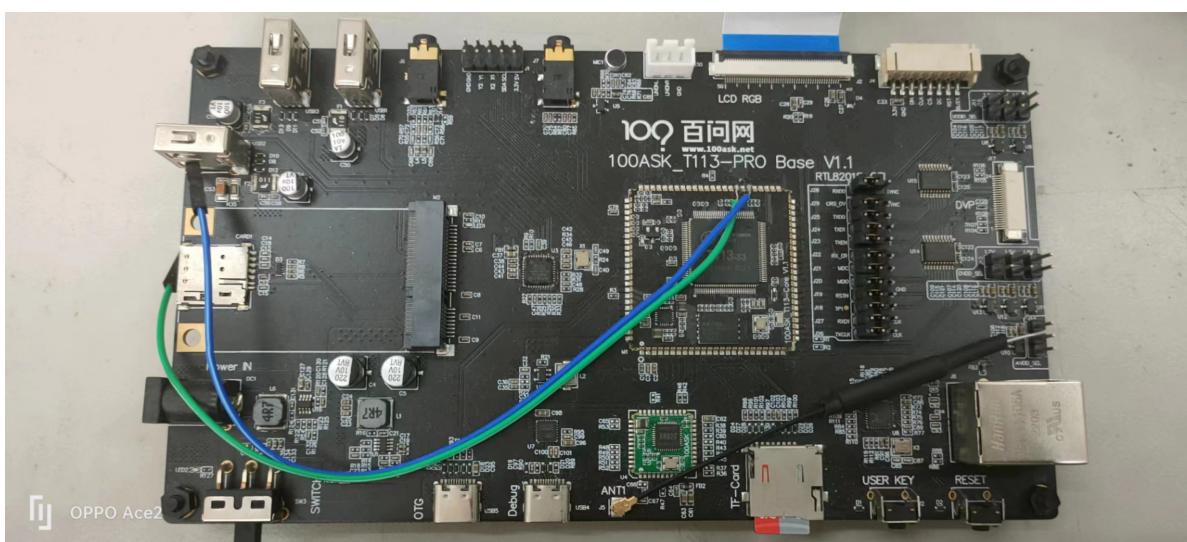
这个命令是用来控制Linux系统中的LED灯的亮度。具体来说，它将数字1写入/sys/class/leds/red/brightness文件中，从而将与此文件相关联的LED灯的亮度设置为最大值（通常100%）。

在Linux系统中，LED灯通常被用于指示设备的状态，如网络活动、电源状态等。通过更改brightness文件中的值，可以控制LED灯的亮度，从而改变状态指示。

```
输出高电平点亮: echo 1 > /sys/class/leds/red/brightness
```



```
输出低电平熄灭: echo 0 > /sys/class/leds/red/brightness
```



既然是用命令来驱动LED，那写应用不就简单了，直接给个示例：

LED.c

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <unistd.h>

void LED_LIGHTING(void)
{
    struct timespec req;
    req.tv_sec = 1;           /* 0s */
    req.tv_nsec = 20000000; /* 20000000ns = 20ms */

    char command[256];
    int ret;

    /*1. 点灯*/
    sprintf(command, "echo 1 > /sys/class/leds/red/brightness");
    ret = system(command);
    nanosleep(&req, NULL); //延时1s加20ms
    /*2. 灭灯*/
}
```

```
    sprintf(command, "echo 0 > /sys/class/leds/red/brightness");
    ret = system(command);
    nanosleep(&req, NULL);延时1s加20ms
}
```