【Linux 学习系列六：操作 GPIO】

2019-05-25

www.topsemic.com

版 本 历 史

版本	作者	参与者	日期	备注
V1.0	Topsemic		2019/05/25	创建

www.topsemic.com

目录

www.topsemic.com

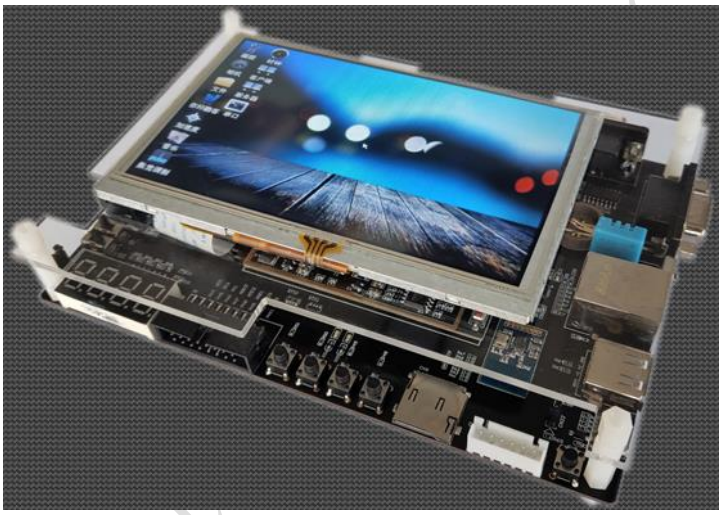
1.引言

学习单片机的第一个例子通常都是点亮 LED 灯，对于 Linux 应用，我们也从 LED 入手，我就记得自己刚开始学的时候查了好多资料才勉强能控制一个灯亮，当时就感受到了 Linux 和单片机裸机有很大的差异。这里做个总结，希望对大家有所帮助。

2.环境介绍

2.1.硬件

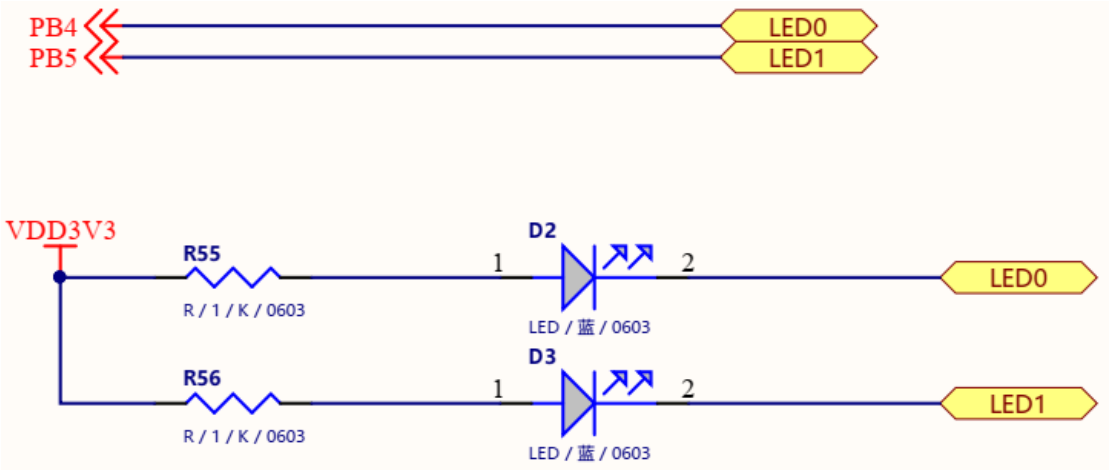
网上的一个第三方做的 NUC972 开发板：



有兴趣购买的朋友，大家去他们的淘宝店购买即可：

https://s.click.taobao.com/t?e=m%3D2%26s%3DVqeqPgNPc7IcQipKwQzePOeEDrYVVa64LKpWJ%2Bin0XJRAdhuF14FMYcqQp9jM3JMRitN3%2FurF3weYrs2z1V%2BWARwPYjsuD9IQ67nx0X4I%2FwbK4NckI8ZycEnxKDasWxncCadkoABCrnmL8IX2r%2Bngfx81NZbGhxUxiXvDf8DaRs%3D

要控制的是板子底板上的 D2 和 D3 两个 LED 指示灯



对应 NUC972 的 PB4 和 PB5 引脚。

2.2.软件

1）需要在上一篇《Linux 学习系列五：Nand Flash 根文件系统制作》的基础上改动下 Linux 内核配置，生成新的 970uimage 并烧写到板子里。

2）uboot、rootfs 使用板子里默认的，板子厂家发货时里面已经烧录过了。

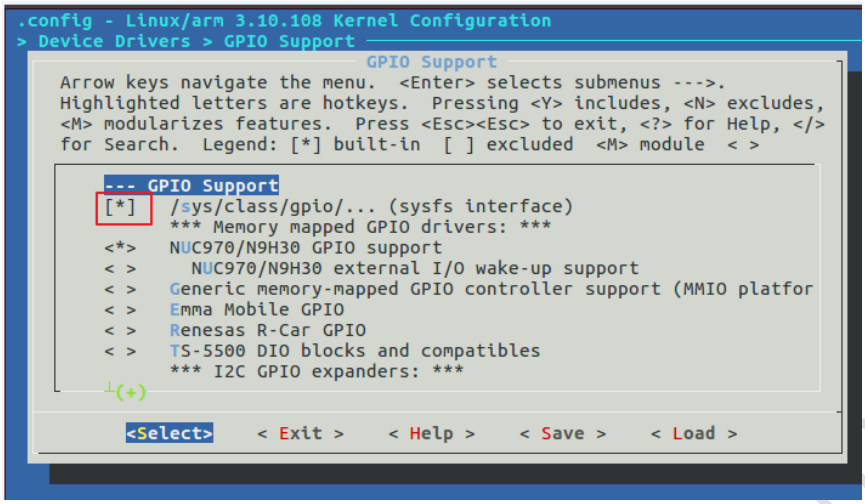
3）板子厂家使用的交叉工具链 arm_linux_4.8.tar.gz，在百度网盘里，见 5 结束语部分。

3.内核配置

1)为了让 NUC972 芯片支持 GPIO 控制，需要在内核中使能”/sys/class/gpio…”，如下所示

```
Device Drivers --->
  -*- GPIO Support --->
    [*] /sys/class/gpio/... (sysfs interface)unzip
```

www.topsemic.com



保存生成新的.config 文件。

2）make uImage，生成 970uimage 文件，将其单独下载到板子里即可，其他 Uboot、Rootfs 等就不需要再下载了。

4.GPIO 操作

4.1.使用命令行操作 GPIO

GPIO 驱动程序将 NUC970 芯片的 IO 口，从 GPIOA~GPIOJ 每组 IO 都保留 32 个号码，所以 GPIOA 编号 0x000~0x01F，GPIOB 编号 0x020~0x03F，GPIOC 编号 0x040~0x05F, GPIOD 编号 0x060~0x07F, GPIOE 编号 0x080~0x09F，GPIOF 编号 0x0A0~0x0BF，GPIOG 编号 0x0C0~0x0DF， GPIOH 编号 0x0E0~0x0FF, GPIOI 编号 0x100~0x11F，GPIOJ 编号 0x120~0x13F。

用户可以通过文件系统/sys/class/gpio/…路径下的文件节点来操作对应的 IO 口

/sys/class/gpio/export :来告诉系统需要控制哪个 GPIO

/sys/class/gpio/unexport:可以取消相应的 GPIO 控制

/sys/class/gpio/gpio0/direction:控制 GPIO in 或 out

/sys/class/gpio/gpio0/value: 控制 GPIO 输出 1 或 0，也可查看输入状态下当前 GPIO 的输入值。

LED D2 对应 GPIO PB4，编号 0x24，对应十进制 36，要控制它的话可在命令行中依次输入下面指令：

```
echo 36 > /sys/class/gpio/export
```

www.topsemic.com

```
echo out > /sys/class/gpio/gpio36/direction
echo 0 > /sys/class/gpio/gpio36/value
echo 1 > /sys/class/gpio/gpio36/value
```

4.2.使用 shell 脚本操作 GPIO

一句句的敲打上述代码显然是比较麻烦的，方便的办法是写到一个 shell 脚本里，这样的话只要执行 shell 脚本就可以控制 LED 了。实际产品中这个也是比较有用的，比如可以在系统启动后，实现 LED D2 亮灭各 1s 循环，这样 LED 就可以作为系统状态指示灯，我们可以通过判断 LED D2 是否正常闪烁来判断系统是否在正常工作。实现方法如下：

1）在板子的/opt 目录（默认没有此目录，自己新建下即可）下，新建一个 gpio.sh 脚本，里面内容如下：

```
#!/bin/sh
echo 36 > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio36/direction
while true
do
    echo 0 > /sys/class/gpio/gpio36/value
    sleep 1
    echo 1 > /sys/class/gpio/gpio36/value
done
```

其中第一行#!/bin/sh 是指此脚本使用/bin/sh 来解释执行。

其中，#!是一个特殊的表示符，其后，跟着解释此脚本的 shell 路径。

sh 只是 shell 的一种，还有很多其它 shell，如：bash,csh,zsh,tcsh,...

这里用到一个 while 循环语句，大家注意下格式即可。

2）利用这个脚本，实现开机自启动，方法是 vi /etc/init.d/rcS

在里面添加一句话

```
./opt/gpio.sh &
```

```
Serial-COM16
#!/bin/sh
/bin/mount -t proc none /proc
/bin/mount -t sysfs sysfs /sys
/bin/mount -t ramfs /tmp
/bin/mount -t ramfs /mnt
#/bin/mkdir /mnt/pts
#/bin/mount -t devpts devpts /dev/pts
/bin/echo > /dev/mdev.seq
/bin/echo /bin/mdev > /proc/sys/kernel/hotplug
/bin/mdev -s

insmod /usr/drivers_demo/mpu6050d.ko

./opt/gpio.sh &

source /etc/profile
./usr/qt_demo/desktop -qws  &
```

因为在 Linux 系统起来后会执行/etc/init.d/rcS 这个文件，所以我们在这里添加上面那句话就可以实现开机后自动控制 LED 了。

大家可以执行 reboot 指令或者断电再重新上电看一下效果。shell 脚本因为不需要编译即可执行，所以使用非常方便，大家平时可以多用用。

4.3.C 语言代码里操作 GPIO

实际项目中，你可能得在 C 代码里去控制 GPIO，这里实现的功能是让 LED D3 循环闪烁，具体步骤如下：

1）编写 gpio_demo.c ，代码如下：

```
/************************************************
 * @{
 * @file    : gpio_demo.c
 * @brief :
 * @author: Wenxue Wang
 * @email : topsemic@sina.com
 * @wechat : wangwenxue1989
 * @date    : 2019-05-25
************************************************/
//-----------------------------------------------
// Copyright (c) Topsemic
//-----------------------------------------------
```

```
#include <stdio.h>

// Operation of LED D2
#define EXPORT_GPIO    "echo    37 > /sys/class/gpio/export"
#define GPIO_OUTPUT    "echo    out > /sys/class/gpio/gpio37/direction"
#define TURNON_LED    "echo    0 > /sys/class/gpio/gpio37/value"
#define TURNOFF_LED "echo    1 > /sys/class/gpio/gpio37/value"

void InitLED()
{
      system(EXPORT_GPIO);
      system(GPIO_OUTPUT);
}

void TurnOnLED()
{
      system(TURNON_LED);
}

void TurnOffLED()
{
      system(TURNOFF_LED);
}

int main(void)
{
      InitLED();

       while(1)
```

```
        {
            TurnOnLED();
            sleep(1);
            TurnOffLED();
            sleep(1);
        }
        return 0;
    }
     done
```

上述代码里说明两处：

i） system()函数，它是调用“/bin/sh -c command”执行特定的命令，阻塞当前进程直到 command 命令执行完毕

ii） sleep()函数，单位是 s，还有个 usleep()函数，它的单位 us。在这里用来做延时。当执行这个函数后，程序就不往下走了，当指定时间结束后，继续执行下面的语句。

2）先使用我们之前的交叉工具链去编译上述代码

```
arm-none-linux-gnueabi-gcc gpio_demo.c -o gpio_demo
```

将生成的 gpio_demo 放到板子里去运行，提示如下错误

```
/opt # ./gpio_demo
-/bin/sh: ./gpio_demo: not found
/opt #
```

原因是因为板子里文件系统和我们用的交叉工具链不匹配。

解决方法是用和板子里默认文件系统对应的交叉工具链（在百度网盘）来编译，具体操作就不详细介绍了。

3）使用和板子里自带文件系统匹配的交叉工具链来重新编译

```
arm-linux-gcc gpio_demo.c -o gpio_demo
```

这时它会报一个错误：

```
topsemic@topsemic-virtual-machine:~/nuc972/examples/gpio$ arm-linux-gcc gpio_dem
o.c -o gpio_demo
arm-linux-gcc: error while loading shared libraries: libstdc++.so.6: cannot oper
 shared object file: No such file or directory
```

解决方法是需要安装 32 位兼容包

```
sudo apt-get install lib32stdc++6 libc6:i386
```

之后就可以编译成功了。

再次将 gpio_demo 放到板子上，运行，就不再报错了，可以看到 LED D3 灯在不断的闪烁。

5.结束语

本篇为大家介绍了 Linux 下 GPIO 的使用，同时也穿插着介绍 shell 脚本的些许知识。

大家有任何想法，欢迎给我留言反馈（微信：wangwenxue1989），发邮件也可以，邮箱：Topsemic@sina.com ，微信公众号如下，欢迎关注：

本期相关的资料在百度网盘，链接:

https://pan.baidu.com/s/1pN5Jc8L2FbrD8YgX6yHWhQ 提取码：lke3；（06 Lesson6 操作 GPIO）

里面包含了如下内容：