码龄5年 暂无认证
240
原创
6482
积分
私信 关注

搜博主文章

热门文章

OPC UA 的本质 32179
使用C++ 编写嵌入式应用程序 14769
OPC UA 的发布/订阅（PUB/SUB）模式 12555
漫谈 Mbed OS（1） 12538
C++ strptime 和strptime函数的使用方式 11810

最新评论

如何构建一个基于IEC61499 的“云化PLC“
吕拓键: 拜读了作者关于IEC 61499的许多文章，很佩服作者对于这个协议的一些深...
基于MQTT的RPC协议
金三亲: ROS不就是嘛？
IEC61499 功能块调用C++ 动态程序库的...
qq_34234922: 我理解是不是先在4diac IDE里面创建一个Simple FB，然后导出成C+...
IEC61499 功能块调用C++ 动态程序库的...
姚家湾: 我是在自己开发的运行时中实现动态库。在运行时内部，功能块对应的是C...
IEC61499 功能块调用C++ 动态程序库的...
qq_34234922: 您好，我看了您的文章《IEC61499 功能块调用C++ 动态程序库的实...

您愿意向朋友推荐“博客详情页”吗？

强烈不推荐 不推荐 一般般 推荐 强烈推荐

最新文章

实时IEC61499 系统-网络篇
RT-Thread ： IEEE1588/PTP 协议的实现（2）
如何实现IEEE1588 高精度时间同步

2021

11月 4篇 | 10月 4篇 | 09月 4篇 | 08月 5篇

原创 姚家湾 2020-05-21 08:49:22 875 收藏 9 版权
分类专栏： linux C++

linux 同时被 2 个专栏收录 订阅专栏

在单片机上异常简单的LED 闪灯程序在Linux OS 下居然会异常麻烦，网络上介绍linux 控制GPIO的文章，各种说法都有。而且大都是shell 命令控制，而不是C语言如何控制。 在树莓PI 中，有一个wiringpi 库，而且资料非常清楚。但是其它厂商的东西就凌乱了。我同时使用瑞芯微的RK3399 PC 板和一块全志低成本的H6 板。方式各不相同。终于搞得明白了点，将内容分享给大家。

Linux 下的设备驱动模型主要包含：类（class）、总线（bus）、设备（device）、驱动（driver）。SysFs文件系统用来表示设备的结构.将设备的层次结构形象的反应到用户空间中。应用程序通过访问SysFs中的文件来控制硬件设备。例如GPIO，SPI和I2C 等等。

GPIO 端口操作

gpio 的文件存放在/sys/class/gpio 中。不过不同的SOC 有些不同，在RK3399 的ubuntu OS 中，gpio 下的目录为：

gpiochip0

gpiochip32

gpiochip64

gpiochip96

gpiochip128

/sys/class/gpio/gpiochipX目录保存系统中GPIO寄存器的信息，包括每个寄存器控制引脚的起始编号 base，寄存器名称，引脚总数

计算引脚编号

在SysFs 中是采用gpio+GPIO 引脚编号来控制GPIO 端口的，首先需要将一个物理gpio的名称，转换为一个GPIO 的编号。在RK3399 中，gpio的物理编号为 GPIO0_B2_d 或者GPIO0_A3_d。其中0 是gpiochip 的编号。计算此引脚编号的方法如下：

引脚编号 = 控制引脚的寄存器基数 + 控制引脚寄存器位数

例如：GPIO0_A3_d 的编号为 3

在全志H6 SOC 中，GPIO 的物理名称为PC13，PH3 这样的方式。GPIO 引脚的编号方法为：

以PC13 为例

P=Pin 可以或略

C=端口字母，该字母在字母表中的位置（从0开始）称为端口号

13=硬件号

GPIO 编号=端口号*32+引脚号

PC13=2*32+13=77

例如PH7=17*32+7=231

PL02=11*32+2=354

产生gpio3 文件

获得了引脚编号之后，要向/sys/class/gpio/export写入此编号，比如12号引脚，在Linux shell中可以通过以下命令实现，
echo 3 > /sys/class/gpio/export
命令成功后生成/sys/class/gpio/gpio3目录，如果没有出现相应的目录，说明此引脚不可导出：

分类专栏

设置GPIO 的方向

direction文件，定义输入输入方向，可以通过下面命令定义为输出

echo out > /sys/class/gpio/gpio12/direction
direction接受的参数：in, out, high, low。high/low同时设置方向为输出。

设置GPIO 的输出值

echo 1 >/sys/class/gpio/gpio3/value

echo 0 >/sys/class/gpio/gpio3/value

C 语言程序

上面介绍的方法是采用shell 命令来实现的，下面介绍C 语言的程序实现。在RK3399 PC 的GPIO 扩展槽2脚是GPIO0_A3_d ,其引脚编号位 3。在上面连接了一个LED 串联一个150欧姆的电阻到地（扩展槽3脚）

下面的程序是在RK3399 PC 板上调试通过的，请安心引用。

gpiolib.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/select.h>
#include <sys/stat.h>
#include "gpiolib.h"

int gpio_direction(int gpio, int dir)
{
    int ret = 0;
    char buf[50];
    sprintf(buf, "/sys/class/gpio/gpio%d/direction", gpio);
    int gpiofd = open(buf, O_WRONLY);
    if(gpiofd < 0) {
        perror("Couldn't open IRQ file");
    r e t   =  -1;
    }

    if(dir == 2 && gpiofd){
        if (3 != write(gpiofd, "high", 3)) {
            perror("Couldn't set GPIO direction to out");
      r e t   =  -2;
        }
    }

    if(dir == 1 && gpiofd){
        if (3 != write(gpiofd, "out", 3)) {
            perror("Couldn't set GPIO direction to out");
      r e t   =  -3;
        }
    }
    else if(gpiofd) {
        if(2 != write(gpiofd, "in", 2)) {
            perror("Couldn't set GPIO directio to in");
      r e t   =  -4;
        }
    }

    close(gpiofd);
    return ret;
}

int gpio_setedge(int gpio, int rising, int falling)
```

```
{
    int ret = 0;
    char buf[50];
    sprintf(buf, "/sys/class/gpio/gpio%d/edge", gpio);
    int gpiofd = open(buf, O_WRONLY);
    if(gpiofd < 0) {
        perror("Couldn't open IRQ file");
    r e t   =  -1;
    }

    if(gpiofd && rising && falling) {
        if(4 != write(gpiofd, "both", 4)) {
            perror("Failed to set IRQ to both falling & rising");
      r e t   =  -2;
        }
  }  else {
        if(rising && gpiofd) {
            if(6 != write(gpiofd, "rising", 6)) {
                perror("Failed to set IRQ to rising");
         r e t   =  -2;
            }
      }   else if(falling && gpiofd) {
            if(7 != write(gpiofd, "falling", 7)) {
                perror("Failed to set IRQ to falling");
         r e t   =  -3;
            }
        }
    }

    close(gpiofd);

    return ret;
}

int gpio_export(int gpio)
{
    int efd;
    char buf[50];
    int gpiofd, ret;

    /* Quick test if it has already been exported */
    sprintf(buf, "/sys/class/gpio/gpio%d/value", gpio);
 e f d  = open(buf, O_WRONLY);
    if(efd != -1) {
        close(efd);
        return 0;
    }

 e f d  = open("/sys/class/gpio/export", O_WRONLY);

    if(efd != -1) {
        sprintf(buf, "%d", gpio);
    r e t   =  write(efd, buf, strlen(buf));
        if(ret < 0) {
            perror("Export failed");
            return -2;
        }
        close(efd);
  }  else {
        // If we can't open the export file, we probably
        // dont have any gpio permissions
        return -1;
    }
    return 0;
}

void gpio_unexport(int gpio)
{
```

```
    int gpiofd, ret;
    char buf[50];
 gpiofd = open("/sys/class/gpio/unexport", O_WRONLY);
    sprintf(buf, "%d", gpio);
 ret = write(gpiofd, buf, strlen(buf));
    close(gpiofd);
}

int gpio_getfd(int gpio)
{
    char in[3] = {0, 0, 0};
    char buf[50];
    int gpiofd;
    sprintf(buf, "/sys/class/gpio/gpio%d/value", gpio);
 gpiofd = open(buf, O_RDWR);
    if(gpiofd < 0) {
        fprintf(stderr, "Failed to open gpio %d value\n", gpio);
        perror("gpio failed");
    }

    return gpiofd;
}

int gpio_read(int gpio)
{
    char in[3] = {0, 0, 0};
    char buf[50];
    int nread, gpiofd;
    sprintf(buf, "/sys/class/gpio/gpio%d/value", gpio);
 gpiofd = open(buf, O_RDWR);
    if(gpiofd < 0) {
        fprintf(stderr, "Failed to open gpio %d value\n", gpio);
        perror("gpio failed");
    }

    do {
   nread = read(gpiofd, in, 1);
  }  while (nread == 0);
    if(nread == -1){
        perror("GPIO Read failed");
        return -1;
    }

    close(gpiofd);
    return atoi(in);
}

int gpio_write(int gpio, int val)
{
    char buf[50];
    int nread, ret, gpiofd;
    sprintf(buf, "/sys/class/gpio/gpio%d/value", gpio);
 gpiofd = open(buf, O_RDWR);
    if(gpiofd > 0) {
        snprintf(buf, 2, "%d", val);
   ret = write(gpiofd, buf, 2);
        if(ret < 0) {
            perror("failed to set gpio");
            return 1;
        }

        close(gpiofd);
        if(ret == 2) return 0;
    }
    return 1;
}

```

```c

int gpio_select(int gpio)
{
    char gpio_irq[64];
    int ret = 0, buf, irqfd;
 fd_set fds;
    FD_ZERO(&fds);

    snprintf(gpio_irq, sizeof(gpio_irq), "/sys/class/gpio/gpio%d/value", gpio);
 irqfd = open(gpio_irq, O_RDONLY, S_IREAD);
    if(irqfd < 1) {
        perror("Couldn't open the value file");
        return -13;
    }

    // Read first since there is always an initial status
 ret = read(irqfd, &buf, sizeof(buf));

    while(1) {
        FD_SET(irqfd, &fds);
    ret = select(irqfd + 1, NULL, NULL, &fds, NULL);
        if(FD_ISSET(irqfd, &fds))
        {
            FD_CLR(irqfd, &fds);  //Remove the filedes from set
            // Clear the junk data in the IRQ file
       ret = read(irqfd, &buf, sizeof(buf));
            return 1;
        }
    }
}

```

gpio.h

```c
#ifndef _GPIOLIB_H_

/* returns -1 or the file descriptor of the gpio value file */
int gpio_export(int gpio);
/* Set direction to 2 = high output, 1 low output, 0 input */
int gpio_direction(int gpio, int dir);
/* Release the GPIO to be claimed by other processes or a kernel driver */
void gpio_unexport(int gpio);
/* Single GPIO read */
int gpio_read(int gpio);
/* Set GPIO to val (1 = high) */
int gpio_write(int gpio, int val);
/* Set which edge(s) causes the value select to return */
int gpio_setedge(int gpio, int rising, int falling);
/* Blocks on select until GPIO toggles on edge */
int gpio_select(int gpio);

/* Return the GPIO file descriptor */
int gpio_getfd(int gpio);

#endif //_GPIOLIB_H_
```

main.c

```c
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include "gpiolib.h"

int main(int argc, char **argv)  {
    int gpio_pin = 3;
```

```c

    gpio_export(gpio_pin);
    gpio_direction(gpio_pin, 1);

    for(int i = 0; i < 5; i++) {
        printf(">> GPIO %d ON\n", gpio_pin);
        gpio_write(gpio_pin, 1);

        sleep(1);

        printf(">> GPIO %d OFF\n", gpio_pin);
        gpio_write(gpio_pin, 0);

        sleep(1);
    }

    return 0;
}
```

C语言编程操作GPIO 05-14
包含一个C语言库， 适用于在Linux下操作GPIO（读、写、设置等）

GPIO按键控制接口 09-15
GPIO接口的实例，实现方案，乘风破浪会有时

抢沙发
请发表有价值的评论， 博客评论不欢迎灌水，良好的社区氛围需大家一起维护。
评论

GPIO操作之C语言版 06-29
本实验实现使用C语言来控制LED灯闪烁。包含ARM汇编调用C语言的方法及传参方法。另外，含有ARM指令集和THUMB指令集过程调用的规范。

主板GPIO控制软件 11-20
本文描述了如何操作工业计算机主板GPIO端口的编程实例

通过sysfs方式控制GPIO_wb3134的博客 10-26
通过sysfs方式控制GPIO,先访问/sys/class/gpio目录,向export文件写入GPIO编号,使得该GPIO的操作接口从内核空间暴露到用户空间,GPIO的操作接口包括…

RK3399平台开发系列讲解(内核入门篇)1.1、通过sysfs清... 10-28
SysFs方式下C语言控制GPIO(RK3399) 姚家湾博客 815 在单片机上异常简单的LED 闪灯程序在Linux OS 下居然会异常麻烦,网络上介绍linux 控制GPIO的…

RK3399 GPIO控制_RK3399_源码 最新发布 10-04
RK3399 GPIO的操作使用方式與相關說明

用C控制GPIO NetRouter专栏 1235
http://forum.cubietech.com/forum.php?mod=viewthread&tid=405&reltid=876&pre_thread_id=0&pre_pos=6&ext=CB 用C控制GPIO的代码[可以直接看CP…

Mysql rk3399_RK3399 友善NanoPC-T4开发板使用sysfs方... 11-18
1 wiringPi for Python简介 wiringPi for Python是wiringPi的Python语言扩展,用于在Python程序中操作GPIO/I2C/SPI库/UART/PWM ... 友善RK3399/NanoPC…

linux下使用gpio控制代码,zynq linux 下控制gpio的c代... 10-19
// gcc gpio.c -o gpio // The kernel needs the following configuration to make this work. // // CONFIG_GPIO_SYSFS=y // CONFIG_SYSFS=y // CONFIG_…

RK3399平台开发系列讲解（应用开发篇）1.9、GPIO 编程-操作/sys/class/gpio/目录下文件方式 内核笔记 977
平台 内核版本 安卓版本 RK3399 Linux4.4 Android7.1 文章目录GPIO 编程：操作/sys/class/gpio/目录下文件方式导出GPIO设置GPIO方向GPIO输入电平…

树莓派GPIO控制--C语言篇 热门推荐 hu7850的博客 3万+
一. 常用开源工程简介树莓派内核中已经编译自带了gpio的驱动，我们常通过一些第三方写好的库函数来完成具体的操作，比较常见的操作库函数有： 1. P…

RK3399平台开发系列讲解(内核入门篇)1.22、sysfs文件系... 10-25
RK3399平台开发入门到精通系列专栏同时被 2 个专栏收录 447 篇文章921 订阅¥59.90¥99.00 订阅专栏 子类__内核入门篇 42 篇文章5 订阅 订阅专栏 =>…

关于/sys/class/gpio 简介 - cjsycyl的专栏 - CSDN博客 8-28
如果是在已经适配好的linux内核上,那么相信已经有了完成的gpiochip,可以在用户空间/sys/class/gpio目录下看到,如:exportgpiochip0/gpiochip32/gpiochip6…

我的内核学习笔记11：linux leds-gpio驱动应用实例 李迟的专栏 9808
linux内核的leds-gpio是使用GPIO控制LED的驱动，只要将板子上LED灯对接的GPIO引脚号进行适当的配置，就能使用这个驱动了，十分方便。网上有很…

Android系统APK操作GPIO读写及添加权限 特立独行的博客 1533
说明： 1、控制 GPIO 的目录位于 /sys/class/gpio。 2、/sys/class/gpio/export 文件用于通知系统需要导出控制的 GPIO 引脚编号。 3、/sys/class/gpio/un…

RK3399平台开发系列讲解(内核入门篇)1.16、 Sysfs设备驱动管理 9-29
RK3399Linux4.4Android7.1 文章目录 1、Sysfs设备驱动管理 1、Sysfs设备驱动管理 Linux系统中一切皆文件。 设备文件在哪里呢?它在/dev目录下,也在/…

Linux内核驱动学习（七）应用层直接操作GPIO 技术联盟 143
文章目录 简介 原理图 节点 设置为输出 设置为输入 映射关系 debugfs pwm demo 简介 前面通过libgpio的方式介绍了内核空间对GPIO进行操作的接口，其…

linux gpio export lq496387202的博客 6998
更多资料翻阅：https://book.2cto.com/201304/20887.html下文转自：http://blog.csdn.net/xgbing/article/details/51009384点击打开链接在嵌入式设备中对…

Linux应用层直接操作GPIO hfyutdg的博客 599
Linux应用层直接操作GPIO

Linux GPIO - gpio.c(GPIO各个接口的实现) Bloom and Grow 7989
先说说gpio_request，其原型是int gpio_request(unsigned gpio, const char *label)其参数gpio为你要申请的哪一个管脚，label则是为其取一个名字。 其…

linux下c语言控制gpio暴露,linux下对/sys/class/gpio中的gpio的控制 weixin_35897007的博客 123
关闭GPIO：在终端输入：#echo "38" > /sys/class/gpio/unexport，即删除GPIO配置文件，可以看到目录gpio38已经被删除。下面是C语言编写的GPIO控…

RK3399平台开发系列讲解（内核入门篇）1.22、sysfs文件系统的数据结构的抽象 内核笔记 1096
文章目录1、底层数据结构1.1、kobject1.2、kset2、中间层数据结构2.1、bus_type2.2、bus_type的注册：bus_register2.3、device2.4、device 向设备…

RK3399的GPIO序号计算方式 joyopirate的博客 1621
根据资料 https://blog.csdn.net/zhuyong006/article/details/80907718 https://www.cnblogs.com/barfoot/p/10954917.html 得知，关于芯片RK3399的io控制…

linux的CH438Q驱动程序 iamyhw的博客 1363
CH438Q的linux驱动程序

linux 驱动笔记(七) catemo的博客 1244
第十六章 输入子系统模型 1 什么是输入子系统模型 1.1 什么是输入子系统 学过的模型： 普通的字符设备模型cdev 混杂设备模型miscdevice 平台模型p…