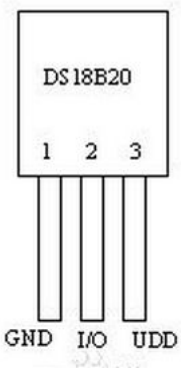
iTOP-4412-驱动-DS18B20 温度传感器使用文档

本文档主要介绍 DS18B20 温度传感器在 iTOP-4412 开发板上的使用过程。主要介绍硬件连接、内核配置以及驱动和应用程序的编译运行测试。

1.1 硬件连接

DS18B20 引脚定义：



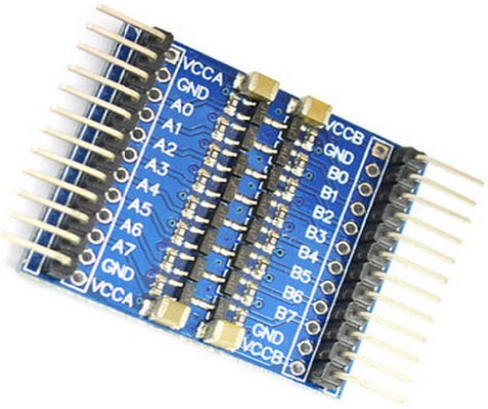
① DQ 为数字信号输入/输出端（电压范围：3.0～5.5V）；

② GND 为电源地；

③ VDD 为外接供电电源输入端（在寄生电源接线方式时接地）。

由于 iTOP-4412 开发板的引脚输出电压是 1.8V，因此需要接一个电平转换模块。

电平转换模块如下图所示。



想实现 1.8v 到 3.3 的电平转换，可以按照如下描述进行连接。

① VCCA 接 1.8V 电源

② VCCB 接 3.3V 电源

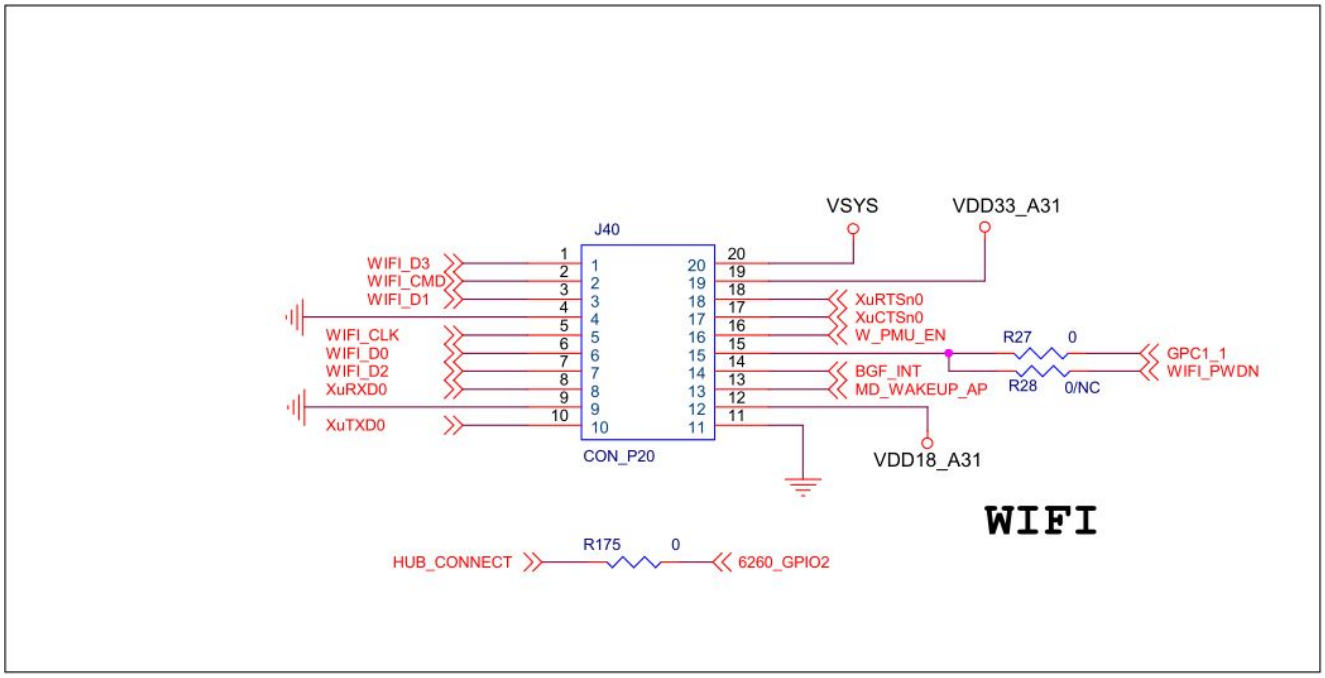
③ GND 接电源负极，两个电源共地。

④ Ax 输入 1.8VTTL 电平，Bx 将输出 3.3V 传感器模块 TTL 电平。

所以，现在要在 iTOP-4412 开发板上找到一个 1.8V 电源引脚、两个 3.3V 电源引脚、三个共地引脚以及一个 1.8V TTL 输入（针对电平转换模块来说）引脚（和驱动程序中对应）。

查看 iTOP-4412 底板原理图，找到如下所示接口。可以被我们使用。

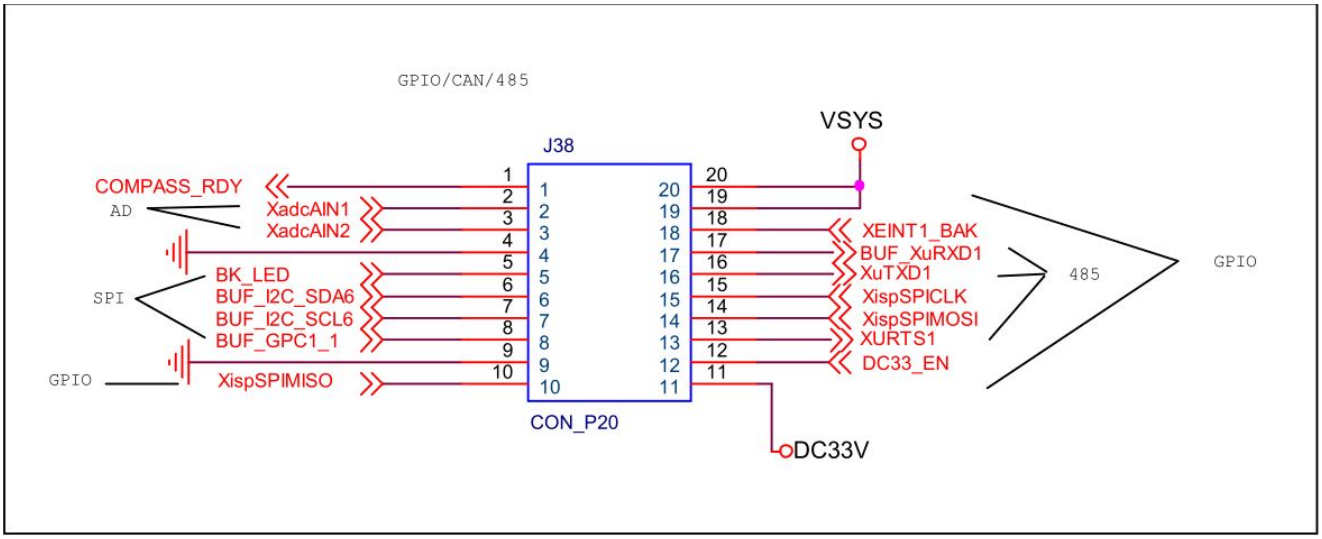
J40 接口（WIFI 接口）。



J40 接口我们使用 4 号共地引脚，9 号共地引脚，12 号 1.8V 电源引脚，19 号 3.3V 电源引脚。连接情况为：

4 号和 9 号引脚接电平转换模块的 GND。12 号接电平转换模块的 VCCA、19 号引脚接电平转换的 VCCB。

J38 接口（GPIO 接口）。



J38 接口我们使用 4 号共地引脚，11 号 3.3V 电源引脚，13 号 1.8V TTL 输入引脚。

连接情况为：4 号引脚接 DS18B20 模块的 GND，11 号接 DS18B20 模块的 VDD。13 号引脚接电平转换模块的 A3 引脚，然后 B3 引脚引出接到 DS18B20 模块的 DQ 引脚。

1.2 配置平台文件

在内核源码目录，使用“vi arch/arm/mach-exynos/mach-itop4412.c”命令打开平台文件。搜索关键词“struct platform_device s3c_device_buzzer_ctl”找到 buzzer 配置。然后在它的上面添加如下信息。

```
#ifdef CONFIG_DS18B20_CTL
struct platform_device s3c_device_ds18b20_ctl = {
        .name     = "ds18b20",
        .id                 = -1,
};
#endif
```

如下图。

```
#ifdef CONFIG_LEDS_CTL
struct platform_device s3c_device_leds_ctl = {
        .name   = "leds",
        .id             = -1,
};
#endif

#ifdef CONFIG_DS18B20_CTL
struct platform_device s3c_device_ds18b20_ctl = {
        .name   = "ds18b20",
        .id             = -1,
};
#endif



#ifdef CONFIG_BUZZER_CTL
struct platform_device s3c_device_buzzer_ctl = {
        .name   = "buzzer_ctl",
        .id             = -1,
```

保存，退出。

再次使用“vi arch/arm/mach-exynos/mach-itop4412.c”命令打开平台文件。搜索关键词“&s3c_device_buzzer_ctl”，在这一行上面添加：

```
#ifdef CONFIG_DS18B20_CTL
        &s3c_device_ds18b20_ctl,
#endif
```

如下图。

```
#ifdef CONFIG_LEDS_CTL
        &s3c_device_leds_ctl,
#endif

#ifdef CONFIG_DS18B20_CTL
        &s3c_device_ds18b20_ctl,
#endif

#ifdef CONFIG_BUZZER_CTL
        &s3c_device_buzzer_ctl,
#endif
```

保存，退出。

使用“vi drivers/char/Kconfig ”命令打开 Kconfig 配置文件。搜索关键词“BUZZER_CTL”，在该段的上面添加如下信息。

```
config DS18B20_CTL
        bool "Enable DS18B20 config"
        default y
        help
          Enable DS18B20 config
```

如下图。

```
config LEDS_CTL
        bool "Enable LEDS config"
        default y
        help
          Enable LEDS config

#add by neo 20180127
config DS18B20_CTL
        bool "Enable DS18B20 config"
        default y
        help
          Enable DS18B20 config

config BUZZER_CTL
        bool "Enable BUZZER config"
        default n
```

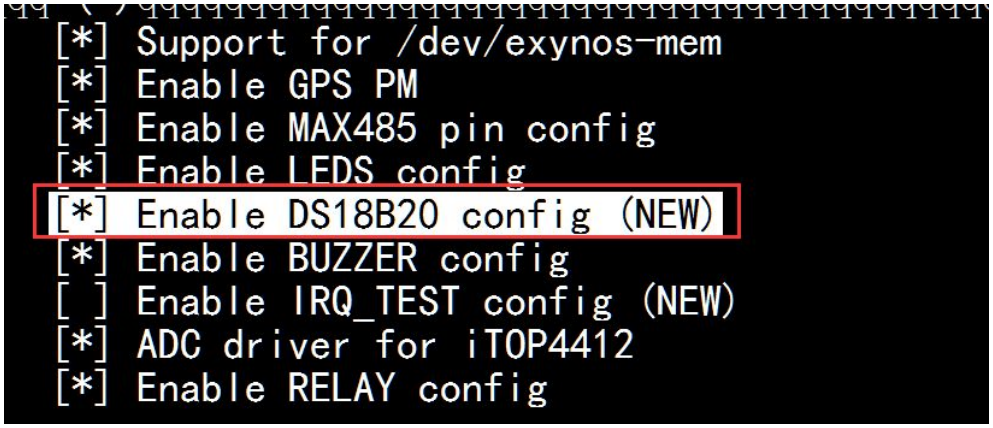
保存，退出。

在内核源码目录使用“cp config_for_linux_scp_elite .config”命令配置缺省信息。（用户要根据自己板子修改配置命令）

在内核目录下使用“make menuconfig”命令打开内核配置界面，如下图。

```
/android4.0/iTop4412_Kernel_3.0# cp config_for_linux_scp_elite .config
/android4.0/iTop4412_Kernel_3.0# make menuconfig
```

进入到 Device Drivers --->Character devices 目录。如下图。



可以看到 DS18B20 设备已经被选中。

然后保存退出内核配置界面。使用“make zImage”命令编译内核。如下图。

```
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0#
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0# make zImage
```

编译完成后如下图。

```
  KSYM    .tmp_kallsyms1.S
  AS      .tmp_kallsyms1.o
  LD      .tmp_vmlinux2
  KSYM    .tmp_kallsyms2.S
  AS      .tmp_kallsyms2.o
  LD      vmlinux
  SYSMAP  System.map
  SYSMAP  .tmp_System.map
  OBJCOPY arch/arm/boot/Image
  Kernel: arch/arm/boot/Image is ready
  GZIP    arch/arm/boot/compressed/piggy.gzip
  AS      arch/arm/boot/compressed/piggy.gzip.o
  SHIPPED arch/arm/boot/compressed/lib1funcs.S
  AS      arch/arm/boot/compressed/lib1funcs.o
  LD      arch/arm/boot/compressed/vmlinux
  OBJCOPY arch/arm/boot/zImage
  Kernel: arch/arm/boot/zImage is ready
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0#
```

然后把编译生成的 zImage （在 arch/arm/boot 目录下）烧写到 iTOP-4412 开发板上，烧写完成后启动开发板。

开发板启动之后，使用命令“ls /sys/devices/platform/”可以查看到新注册的 ds18b20 设备，如下图所示。

```
[root@iTOP-4412]# ls /sys/devices/platform/
adc_ctl                 relay_ctl               s5pv210-uart.2
alarm                   s3c-pl330.1             s5pv210-uart.3
android_pmem.0          s3c-pl330.2             samsung-audio
android_pmem.1          s3c-sdhci.2             samsung-audio-idma
arm-pmu.0               s3c-sdhci.3             samsung-i2s.0
bt-sysfs                s3c-usbgadget           samsung-i2s.4
buzzer_ctl              s3c2410-wdt             samsung-keypad
ds18b20                 s3c2440-i2c.1           samsung-kmsg
dw_mmc                  s3c2440-i2c.3           samsung-pd.0
exynos-busfreq          s3c2440-i2c.4           samsung-pd.1
exynos-usb-switch       s3c2440-i2c.5           samsung-pd.2
exynos4412-adc          s3c2440-i2c.7           samsung-pd.5
gpio-keys               s3c24xx-pwm.1           samsung-pd.6
i2c-gpio.0              s3c64xx-rtc             samsung-pd.7
ion-exynos              s3c64xx-spi.2           samsung-rp
irq_test                s5p-ehci                serial8250
```

设备注册完成。

1.3 驱动应用程序

驱动程序：

```
#include <linux/init.h>
#include <linux/module.h>
#include <linux/delay.h>
#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/slab.h>
#include <linux/input.h>
#include <linux/errno.h>
#include <asm/uaccess.h>
#include <linux/sched.h>
#include <linux/platform_device.h>
#include <linux/miscdevice.h>
#include <linux/fs.h>
#include <asm/irq.h>
#include <linux/gpio.h>
#include <plat/gpio-cfg.h>
#include <mach/gpio.h>
#include <mach/gpio-exynos4.h>

#define DS18B20_DEBUG
#ifdef DS18B20_DEBUG
#define DPRINTK(x...) printk("DS18B20_CTL DEBUG:" x)
#else
#define DPRINTK(x...)
```

```
#endif

#define DRIVER_NAME "ds18b20"
#define DEVICE_NAME "ds18b20"
#define DS18B20_DQ              EXYNOS4_GPA0(7)   //J38 接口 13 号引脚
#define OUTPUT                    S3C_GPIO_OUTPUT
#define INPUT                     S3C_GPIO_INPUT

MODULE_LICENSE("Dual BSD/GPL");
MODULE_AUTHOR("TOPEET");

unsigned char init_ds(void)//ds18b20 复位，返回 0 成功，返回 1 失败
{
        unsigned char ret = 1;
        int i =0;
        s3c_gpio_cfgpin(DS18B20_DQ,OUTPUT);
        s3c_gpio_setpull(DS18B20_DQ, S3C_GPIO_PULL_DOWN);
        gpio_direction_output(DS18B20_DQ,0);
        udelay(250);
        gpio_direction_output(DS18B20_DQ,0);//发送复位脉冲
        udelay(500);//延时( >480us )
        gpio_direction_output(DS18B20_DQ,1);//拉高数据线
        s3c_gpio_cfgpin(DS18B20_DQ, INPUT); //用返回的值来判断初始化有没有成功，18B20 存在的话
=0，否则=1

    while((ret==1)&&(i<10)){
        ret = gpio_get_value(DS18B20_DQ);
        udelay(10);
        i++;
    }
    if(ret==0){
        return 0;
    }
    else{
            return -1;
    }
}
```

```
void write_byte(char data)//向 18b20 写一个字节
{                                   //数据线从高电平拉至低电平，产生写起始信号。15us 之内将所需写的位送到数
据线上，
    int i = 0;
    s3c_gpio_cfgpin(DS18B20_DQ,OUTPUT);
    s3c_gpio_setpull(DS18B20_DQ, S3C_GPIO_PULL_UP);
    for(i=0;i<8;i++)
        {
            gpio_direction_output(DS18B20_DQ,0);
          udelay(10);
            gpio_direction_output(DS18B20_DQ,1);
            gpio_direction_output(DS18B20_DQ, data&0x01);
            udelay(40);
            gpio_direction_output(DS18B20_DQ,1);
            udelay(2);
            data >>= 1;
    }
}
unsigned char read_byte(void)//从 18b20 读一个字节
{                                           //主机数据线先从高拉至低电平 1us 以上，再使数据线升为高电平，
从而产生读信号
        unsigned char i;
        unsigned char data=0;
        s3c_gpio_cfgpin(DS18B20_DQ,OUTPUT);
        for(i=0;i<8;i++)
    {
            data >>= 1;
          gpio_direction_output(DS18B20_DQ,0);
          udelay(1);
          gpio_direction_output(DS18B20_DQ,1);
          s3c_gpio_cfgpin(DS18B20_DQ,INPUT);
            udelay(10);
          if(gpio_get_value(DS18B20_DQ))
          data |= 0x80;
          udelay(50);
            s3c_gpio_cfgpin(DS18B20_DQ,OUTPUT);
          gpio_direction_output(DS18B20_DQ,0);
          gpio_direction_output(DS18B20_DQ,1);
    }
```

```
        return data;
}

static ssize_t ds18b20_ctl_read(struct file *files, unsigned int *buffer, size_t count, loff_t *ppos)
{
            unsigned int tmp;
            unsigned int ret;
            unsigned int th,tl;
            th=tl=0;
            init_ds( );
            udelay(500);
            write_byte(0xcc);//跳过读序号列号的操作
            write_byte(0x44); //启动温度转换
            init_ds( );
            udelay(500);
            write_byte(0xcc);    //跳过读序号列号的操作
            write_byte(0xbe);    //准备读温度
            tl= read_byte( );    //读出温度的低位 LSB
            th= read_byte( );    //读出温度的高位 MSB
            th<<=8;
            tl|=th;                 //获取温度
            tmp=tl;
            ret=copy_to_user(buffer, &tmp, sizeof(tmp));

                if(ret>0)
                {
                        return 0;
                }
                    return -1;
}

static int ds18b20_ctl_close(struct inode *inode, struct file *file){
    printk(" %s   ! ! !\n",__FUNCTION__);
    DPRINTK("Device Closed Success!\n");
    return 0;
}

static int ds18b20_ctl_open(struct inode *inode, struct file *file){
    printk(" %s   ! ! !\n",__FUNCTION__);
    DPRINTK("Device Opened Success!\n");
```

```
	return nonseekable_open(inode,file);
}

int ds18b20_pm(bool enable)
{
	int ret = 0;
	printk(" %s    ! ! !\n",__FUNCTION__);
	printk("debug: DS18B20 PM return %d\r\n" , ret);
	return ret;
};

static struct file_operations ds18b20_ctl_ops = {
	.owner = THIS_MODULE,
	.open = ds18b20_ctl_open,
	.release = ds18b20_ctl_close,
	.read        = ds18b20_ctl_read,
};

static    struct miscdevice ds18b20_ctl_dev = {
	.minor = MISC_DYNAMIC_MINOR,
	.name = DEVICE_NAME,
	.fops = &ds18b20_ctl_ops,
};


static int ds18b20_ctl_probe(struct platform_device *pdv){
	int ret;
	char *banner = "ds18b20 Initialize\n";
	printk(banner);
	printk(" %s    ! ! !\n",__FUNCTION__);
	ret = gpio_request(DS18B20_DQ,"GPA0_7");
	if(ret){
		DPRINTK( "gpio_request DS18B20_DQ failed!\n");
		return ret;
	}
	ret=init_ds();
	if(ret==0){
		DPRINTK( "DS18B20 initialized ok!\n");
	}
	else
```

```
                DPRINTK( "DS18B20 initialized fail!\n");

        s3c_gpio_cfgpin(DS18B20_DQ,OUTPUT);
        gpio_direction_output(DS18B20_DQ, 1);
        ret = misc_register(&ds18b20_ctl_dev);
        if(ret<0)
        {
                printk("leds:register device failed!\n");
                goto exit;
        }
        return 0;

exit:
        misc_deregister(&ds18b20_ctl_dev);
        return ret;
}

static int ds18b20_ctl_remove(struct platform_device *pdv){
        printk(" %s    ! ! !\n",__FUNCTION__);
        printk(KERN_EMERG "\tremove\n");
        gpio_free(DS18B20_DQ);
        misc_deregister(&ds18b20_ctl_dev);
        return 0;
}

static void ds18b20_ctl_shutdown(struct platform_device *pdv){
        printk(" %s    ! ! !\n",__FUNCTION__);

}

static int ds18b20_ctl_suspend(struct platform_device *pdv,pm_message_t pmt){
        printk(" %s    ! ! !\n",__FUNCTION__);
        DPRINTK("ds18b20 suspend:power off!\n");
        return 0;
}

static int ds18b20_ctl_resume(struct platform_device *pdv){
        printk(" %s    ! ! !\n",__FUNCTION__);
        DPRINTK("ds18b20 resume:power on!\n");
        return 0;
```

```
}

struct platform_driver ds18b20_ctl_driver = {
	.probe = ds18b20_ctl_probe,
	.remove = ds18b20_ctl_remove,
	.shutdown = ds18b20_ctl_shutdown,
	.suspend = ds18b20_ctl_suspend,
	.resume = ds18b20_ctl_resume,
	.driver = {
		.name = DRIVER_NAME,
		.owner = THIS_MODULE,
	}
};

static int ds18b20_ctl_init(void)
{
	int DriverState;
	printk(" %s   ! ! !\n",__FUNCTION__);
	DriverState = platform_driver_register(&ds18b20_ctl_driver);
	printk(KERN_EMERG "\tDriverState is %d\n",DriverState);
	return 0;
}



static void ds18b20_ctl_exit(void)
{
	printk(" %s   ! ! !\n",__FUNCTION__);
	platform_driver_unregister(&ds18b20_ctl_driver);
}

module_init(ds18b20_ctl_init);
module_exit(ds18b20_ctl_exit);
```

应用程序：

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>
```

```
#include <sys/ioctl.h>
#include <string.h>
#include <errno.h>
#include <stdint.h>
#include <termios.h>
#define K 0.0625
int main(int argc , char **argv){
	int fd,i=5;
	char count = 5;
	unsigned int tmp;
	float res=0;
	char *hello_node = "/dev/ds18b20";
	if((fd = open(hello_node,O_RDWR|O_NOCTTY))<0){
		printf("APP open %s failed",hello_node);
	}
	else{
			printf("open ds18b20 success \n" );
	}
	while(i--){
		read(fd, &tmp , sizeof(tmp));
		printf("read");
			sleep(1);
	}
	res=tmp*K;
	printf("the currently temperature is %f\n",res);
	close(fd);
		return 0;
}
```

1.4 编译运行测试

1.4.1 驱动程序编译

把驱动程序“itop4412-ds18b20.c”和 Makefile 文件上传到同一目录，执行“make”命令。如下图。

```
root@ubuntu:~/neo/ds18b20# make
make -C /home/topeet/android4.0/iTop4412_Kernel_3.0 M=/root/neo/ds18b20 modules
make[1]: Entering directory `/home/topeet/android4.0/iTop4412_Kernel_3.0'
  CC [M]  /root/neo/ds18b20/itop4412-ds18b20.o
/root/neo/ds18b20/itop4412-ds18b20.c:158: warning: initialization from incompatible pointer type
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /root/neo/ds18b20/itop4412-ds18b20.mod.o
  LD [M]  /root/neo/ds18b20/itop4412-ds18b20.ko
make[1]: Leaving directory `/home/topeet/android4.0/iTop4412_Kernel_3.0'
root@ubuntu:~/neo/ds18b20#
```

通过 U 盘挂载、tftp 或者 nfs 功能把“itop4412-ds18b20.ko”文件上传到开发板。

1.4.2 应用程序编译

把应用程序“test-itop4412-ds18b20.c”上传到 ubuntu 系统。使用“ arm-none-linux-gnueabi-gcc -o test-itop4412-ds18b20 test-itop4412-ds18b20.c -static”命令来静态编译应用程序。如下图。

```
root@ubuntu:~/neo/ds18b20# ls
test-itop4412-ds18b20.c
root@ubuntu:~/neo/ds18b20# arm-none-linux-gnueabi-gcc -o test-itop4412-ds18b20 test-itop4412-ds18b20.c -
static
root@ubuntu:~/neo/ds18b20# ls
test-itop4412-ds18b20  test-itop4412-ds18b20.c
root@ubuntu:~/neo/ds18b20#
```

通过 U 盘挂载、tftp 或者 nfs 功能把“test-itop4412-ds18b20”文件上传到开发板。如下图。

```
[root@iTOP-4412]# ls
bin                     linuxrc                 tcp_client
dev                     mnt                     test-itop4412-ds18b20
etc                     proc                    tmp
itop4412-ds18b20.ko     sbin                    usr
lib                     sys                     var
[root@iTOP-4412]#
```

1.4.3 运行测试

使用“insmod itop4412-ds18b20.ko”命令来加载驱动程序。如下图。

```
lib                     sys                     var
[root@iTOP-4412]# insmod itop4412-ds18b20.ko
[ 1551.508265]  ds18b20_ctl_init  ! ! !
[ 1551.510770] ds18b20 Initialize
[ 1551.513439]  ds18b20_ctl_probe  ! ! !
[ 1551.526010] DS18B20_CTL DEBUG:DS18B20 initialized ok!
[ 1551.540252]  DriverState is 0
[root@iTOP-4412]#
```

由上图可知。进入 probe，并且 ds18b20 初始化成功。

使用“ ./test-itop4412-ds18b20”运行应用程序，提示权限不够。使用“chmod 777 test-itop4412-ds18b20”命令修改权限后，继续运行应用程序，如下图。

```
[ 1551.540252]  DriverState is 0
[root@iTOP-4412]# ./test-itop4412-ds18b20
-/bin/sh: ./test-itop4412-ds18b20: Permission denied
[root@iTOP-4412]# chmod 777 test-itop4412-ds18b20
[root@iTOP-4412]# ./test-itop4412-ds18b20
[ 2018.629167]  ds18b20_ctl_open  ! ! !
[ 2018.631377] DS18B20_CTL DEBUG:Device Opened Success!
open ds18b20 success
readreadreadreadr[ 2023.667820]  ds18b20_ctl_close  ! ! !
[ 2023.671444] DS18B20_CTL DEBUG:Device Closed Success!
eadthe currently temperature is 22.687500
[root@iTOP-4412]#
```

程序运行大约 5、6 秒之后，自动停止，并返回温度值。由上图可知当前室内温度为 22.68℃。

使用“rmmod itop4412-ds18b20”命令卸载驱动。如下图。

```
eadthe currently temperature is 22.687500
[root@iTOP-4412]# rmmod itop4412-ds18b20
[ 2166.425375]  ds18b20_ctl_exit  ! ! !
[ 2166.427581]  ds18b20_ctl_remove  ! ! !
[ 2166.431313]  remove
[root@iTOP-4412]#
```

由上图可知卸载成功，测试完成。

联系方式

北京迅为电子有限公司致力于嵌入式软硬件设计，是高端开发平台以及移动设备方案提供商；基于多年的技术积累，在工控、仪表、教育、医疗、车载等领域通过 OEM/ODM 方式为客户创造价值。

iTOP-4412 开发板是迅为电子基于三星最新四核处理器 Exynos4412 研制的一款实验开发平台，可以通过该产品评估 Exynos 4412 处理器相关性能，并以此为基础开发出用户需要的特定产品。

本手册主要介绍 iTOP-4412 开发板的使用方法，旨在帮助用户快速掌握该产品的应用特点，通过对开发板进行后续软硬件开发，衍生出符合特定需求的应用系统。

如需平板电脑案支持，请访问迅为平板方案网“http://www.topeet.com”，我司将有能力为您提供全方位的技术服务，保证您产品设计无忧！

本手册将持续更新，并通过多种方式发布给新老用户，希望迅为电子的努力能给您的学习和开发带来帮助。

迅为电子

2018 年 1 月