实验 4 ARM 裸机 LED 点亮程序

实验目的

- 认识 tiny6410 开发板的硬件环境
- 学会在 tiny6410 开发板编写、下载、运行 ARM 裸机程序的方法。

实验设备

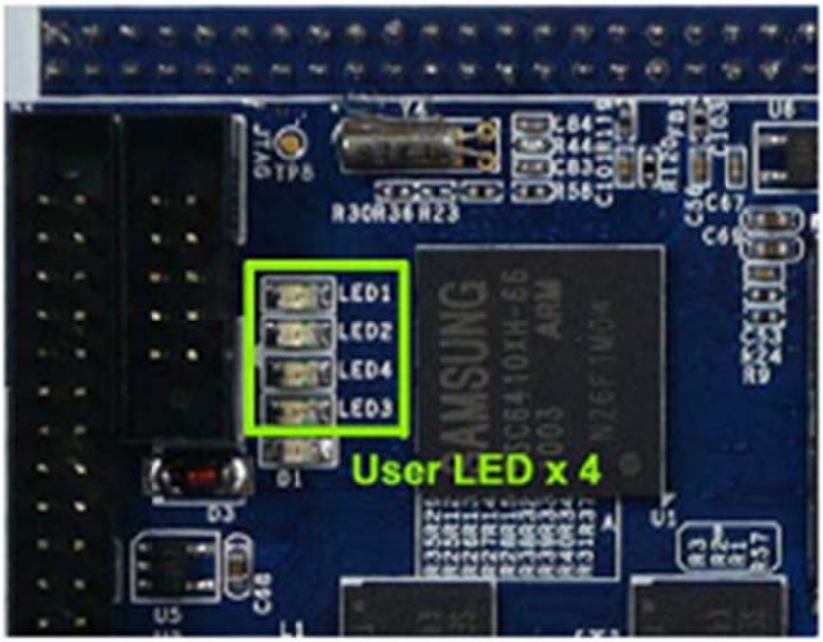
- 硬件：PC 机+tiny6410 开发板+SD 卡
- 软件：WINXP+VMWARE 虚拟机+UBUNTU10.04-32bit+arm-linux-gcc。

实验内容

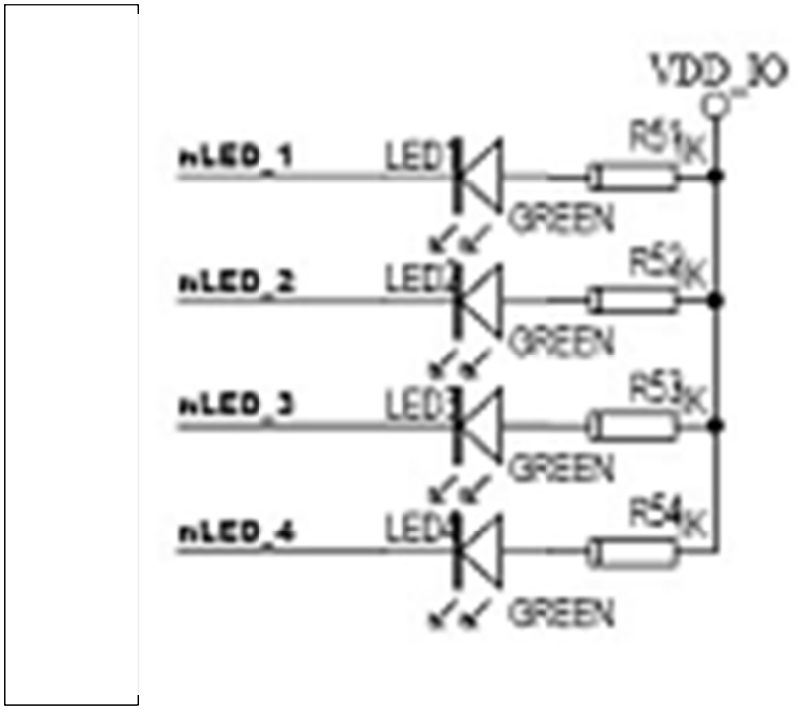
- 1.使用汇编语言编写 LED 灯点亮程序，并使用 makefile 编译生成 bin 文件，下载到 tiny6410 开发板执行，观测核心板上 LED 灯的点亮情况；
- 2. 使用 C 语言编写 LED 灯点亮程序，并使用 makefile 编译生成 bin 文件，下载到 tiny6410 开发板执行，观测核心板上 LED 灯的点亮情况；

实验原理

1. LED 硬件连接



2. 4 个 LED 指示灯直接与 CPU 的 GPIO 相连



3. LED 详细占用资源

	LED4	LED3	LED2	LED1
GPIO	GPK7	GPK6	GPK5	GPK4

4. GPIO 寄存器

Register	Address	R/W	Description	Reset Value
GPKCON0	0x7F008800	R/W	Port K Configuration Register 0	0x22222222
GPKCON1	0x7F008804	R/W	Port K Configuration Register 1	0x22222222
GPKDAT	0x7F008808	R/W	Port K Data Register	Undefined
GPKPUD	0x7F00880C	R/W	Port K Pull-up/down Register	0x55555555

5. GPKCON0：GPK4~7

GPK4	[19:16]	0000 = Input 0010 = Host I/F DATA[4] 0100 = Reserved 0110 = Reserved	0001 = Output 0011 = HSI TX READY 0101 = DATA_CF[4] 0111 = Reserved	0010
GPK5	[23:20]	0000 = Input 0010 = Host I/F DATA[5] 0100 = Reserved 0110 = Reserved	0001 = Output 0011 = HSI TX WAKE 0101 = DATA_CF[5] 0111 = Reserved	0010
GPK6	[27:24]	0000 = Input 0010 = Host I/F DATA[6] 0100 = Reserved 0110 = Reserved	0001 = Output 0011 = HSI TX FLAG 0101 = DATA_CF[6] 0111 = Reserved	0010
GPK7	[31:28]	0000 = Input 0010 = Host I/F DATA[7] 0100 = Reserved 0110 = Reserved	0001 = Output 0011 = HSI TX DATA 0101 = DATA_CF[7] 0111 = Reserved	0010

6. GPKDATA

GPKDAT	Bit	Description
GPK[15:0]	[15:0]	When the port is configured as input port, the corresponding bit is the pin state. When the port is configured as output port, the pin state is the same as the corresponding bit. When the port is configured as functional pin, the undefined value will be read.

7. **GPKPUD**

GPKPUD	Bit	Description
GPK[n]	[2n+1:2n] n = 0~15	00 = pull-up/down disabled 01 = pull-down enabled 10 = pull-up enabled 11 = Reserved.

8.LED 灯亮：配置相应的寄存器输出低电平；LED 灯灭：配置相应寄存器输出高电平。

实验步骤

实验内容一：编写 ARM 汇编程序，点亮 LED 灯。

1.在虚拟机 LINUX 交叉开发环境下编写、编译源程序。

1) 编辑源程序 leds.S （注意：这里的 ARM 汇编程序采用的是 GNU 风格，使用的是 GNU 编译器）

2) 编写 Makefile 文件（注意：编辑 Makefile 文件时，不要把本实验指导书的代码复制过去，这涉及到 WINDOWS 和 LINUX 文本编码的问题，所以直接复制可能会出现错误。）

3) 执行 Make，对 leds.S 编译，生成可执行的 leds.bin 文件

2.硬件开发板连线

1）把开发板的 USB 口和 PC 机的 USB 口用开发板附带的 USB 线缆连接起来。

2）把电源线连接到开发板（连接后先不要开机），把开发板的 S2 开关拨到 SDBOOT 一侧。

3.把 leds.bin 文件烧写到 tiny6410 开发板

1）安装 minitools 烧写工具（MiniToolsSetup-Windows-20140317.exe），安装完成 minitools 后，上电启动开发板，这时主机会提示安装驱动程序，按照提示安装好驱动即可。参考“03-Tiny6410 刷机指南.pdf”的 P28~P29 的 2.1.1 节。

2）把 leds.bin 文件从虚拟 LINUX 系统拷贝到 WINDWOS 系统。在 leds.bin 所在目录下执行如下指令：

cp leds.bin /mnt/hgfs/share

执行成功后，就可以在 WINDOWS 系统“D:\VM_OS\ub100432bit\share”看到 leds.bin 文件。

3）然后使用 minitools 工具烧写 leds.bin 到开发板，参考“Linux 平台下 Tiny6410 裸机程序开发指南”的 P10，“烧写运行”的“方式一”。注意：绝对不要使用“方式二”!!!!!!!!

注意：有时候 minitools 会发现有卡死的现象。解决方法是：第一安装完 minitools 和驱动之后，先把 USB 口线拔下来重插一次，并且要先启动

开发板，后启动 minitools。先启动 minitools，后启动开发板，容易造成 minitools 卡死。

4.leds.bin 运行后，观察 LED 灯的运行情况，并记录。

实验内容二：编写 C 程序，点亮 LED 灯。实验步骤参考实验内容 1。该内容生成的 bin 文件名为 ledc.bin。

参考源程序

实验内容 1：

leds.S 源代码：

```
.global _start

_start:
    // 把外设的基地址告诉 CPU
    ldr r0, =0x70000000                    // 对于 6410 来说，内存(0x00000000～0x60000000),外设(0x70000000-0x7fffffff)
    orr r0, r0, #0x13                      //外设大小:256M
    mcr p15,0,r0,c15,c2,4                  //把 r0 的值(包括了外设基地址+外设大小)告诉 cpu

    // 关看门狗
    ldr r0, =0x7E004000
    mov r1, #0
    str r1, [r0]

    // 设置 GPKCON0
    ldr r1, =0x7F008800
    ldr r0, =0x11110000

    str r0, [r1]

    mov r2, #0x1000
led_blink:
    // 设置 GPKDAT，使 GPK_4/5/6/7 引脚输出低电平，LED 亮
    ldr r1, =0x7F008808
    mov r0, #0
    str r0, [r1]

    // 延时
```

```
    bl delay

    // 设置 GPKDAT，使 GPK_4/5/6/7 引脚输出高电平，LED 灭
    ldr r1, =0x7F008808
    mov r0, #0xf0
    str r0, [r1]

    // 延时
    bl delay

    sub r2, r2, #1
    cmp r2,#0
    bne led_blink

halt:
    b halt


delay:
    mov r0, #0x1000000
delay_loop:
    cmp r0, #0
    sub r0, r0, #1
    bne delay_loop
    mov pc, lr
```

Makefile 文件：

```
leds.bin: leds.o
    arm-linux-ld -Ttext 0x50000000 -o leds.elf $^
    arm-linux-objcopy -O binary leds.elf leds.bin
    arm-linux-objdump -D leds.elf > leds_elf.dis
%.o : %.S
    arm-linux-gcc -o $@ $< -c

%.o : %.c
    arm-linux-gcc -o $@ $< -c

clean:
    rm *.o *.elf *.bin *.dis -rf
```

实验内容 2：

Start.S 源代码：（该程序负责开发板硬件初始化工作）

```
// 启动代码
.global _start

_start:

    // 把外设的基地址告诉 CPU
    ldr r0, =0x70000000                    // 对 于  6410  来 说 , 内 存 (0x00000000 ～ 0x60000000),外设(0x70000000-0x7fffffff)
    orr r0, r0, #0x13                      //外设大小:256M
    mcr p15,0,r0,c15,c2,4                  //把 r0 的值(包括了外设基地址+外设大小)写给 cpu

    // 关看门狗
    ldr r0, =0x7E004000
    mov r1, #0
    str r1, [r0]

    // 设置栈
    ldr sp, =0x0c002000

    // 调用 C 函数点灯
    bl main

halt:
    b halt
```

ledc.c 源程序:（该程序负责点亮 LED 灯）

```
// 功能:c 实现流水灯

// 延时
void delay()
{
    volatile int i = 0x500000;
    while (i--);
}

int main()
{
    // 配置引脚
    volatile unsigned long *gpkcon0 = (volatile unsigned long *)0x7F008800;
    volatile unsigned long *gpkdat = (volatile unsigned long *)0x7F008808;
```

```
    *gpkcon0 = 0x11110000;      //配置 GPK4/5/6/7 为输出功能

    // 跑马灯
    while (1)
    {
        *gpkdat=0x000000f0;      //GPKDAT=0x000000f0,GPKDAT[7:4]=1111,4 个 LED 全灭
        delay();
        *gpkdat=0x000000e0;      //GPKDAT=0x000000e0,GPKDAT[7:4]=1110,LED1 点亮
        delay();
        *gpkdat=0x000000d0;      //GPKDAT=0x000000d0,GPKDAT[7:4]=1101,LED2 点亮
        delay();
        *gpkdat=0x000000b0;      //GPKDAT=0x000000b0,GPKDAT[7:4]=1011,LED3 点亮
    delay();
        *gpkdat=0x00000070;      //GPKDAT=0x00000070,GPKDAT[7:4]=0111,LED4 点亮
       delay();
        *gpkdat=0x00000000;      //GPKDAT=0x00000000,GPKDAT[7:4]=0000,4 个 LED 全
亮
    delay();
    }

    return 0;
}
```

Makefile：

```
ledc.bin: start.o  ledc.o
    arm-linux-ld  -Ttext  0x50000000  -o  ledc.elf  $^
    arm-linux-objcopy  -O  binary  ledc.elf  ledc.bin
    arm-linux-objdump  -D  ledc.elf  >  ledc_elf.dis
%.o : %.S
    arm-linux-gcc  -o  $@  $<  -c

%.o : %.c
    arm-linux-gcc  -o  $@  $<  -c

clean:
    rm  *.o  *.elf  *.bin  *.dis  -rf
```

练习题：

修改程序，实现如下功能：实现 4 个 LED 灯依次点亮，并循环往复。