实验内容.

GPIO实验(一).

1. 本小组为第12组.需在88-95引脚范围内进行选择。

最终选择90、91号引脚作为实验的输出端.操作如下:

① 因为该引脚所在寄存器编号为2,故先在头文件中设置编号2的各寄存器地址:

```
#define  GPDR2    (*(( volatile  ulong *)(0x40E00014)))
#define  GPLR2    (*(( volatile  ulong *)(0x40E00008)))
#define  GPSR2    (*(( volatile  ulong *)(0x40E00020)))
#define  GPCR2    (*(( volatile  ulong *)(0x40E0002C))).
```

② 设置90、91号引脚为输出.

编号2寄存器引脚范围是64-95,故90,91号引脚对应寄存器中的26、27比特位。

将26、27比特位置1,其余位置0生成32位数为 0x0C000000.

令该数值与GPDR2进行或运算: GPDR2 |= 0x0C000000.

即将2寄存器90、91号引脚状态/方向置为输出且不改变其他引脚方向.

2. 读取引脚电平状态,通过串口显示在超级终端中,并将电平状态在LED灯上显示出来.

设计思路:

在超级终端显示则直接将GPLR2的值输出,并观察对应比特位的值即可.

将电平状态表示在LED灯时,先将26、27比特位逻辑右移至0、1比特位,并将逻辑右移后的值传送给LED灯的地址(Led-Addr).观察左边两个灯的熄灭情况,指示灯亮对应低电平,不亮对应高电平.

程序设计:

```
int   led_value;          //定义变量用于接收GPLR2的值.
GPDR2 |=  0x0C000000;     //修改90、91引脚方向为输出.
led_value = GPLR2;        //读取GPLR2的值并赋给变量.
printf("\n\n led_value = %x \n", led_value);
                          //将GPLR2的值通过串口在超级终端上输出.
```

Led_Addr = led_value >>26; //将GPLR2的值逻辑右移26位，并传递给LED灯.
//使得90、91号引脚的值，对应LED灯的0、1比特位.

实验结果观察：

在超级终端上输出的值为 led_value = 13b7ffff

其中 88-91 对应的16进制数为3. 转换2进制为 0011

即 90、91号引脚均为低电平，此时左侧第1、2个LED灯预期结果为亮.

通过观察实验板LED灯状态，左侧1、2灯亮，与预期结果相同。

3. 选中91号引脚，先设置为高，再设置为低，并将结果显示到超级终端和LED灯上.

设计思路：使用 GPSR2 和 GPCR2 寄存器用来将91号引脚设置为高/低电平.

程序源码：

```
//设置90、91号引脚为向为输出，并观察初始条件下led_value的值.
int   led_value;
GPDR2 |=  0x0C000000;
led_value  =  GPLR2;
printf ("\n\n led_value = %x \n", led_value );
//将91号引脚置1，观察led_value变化，并通过逻辑右移将91号引脚值反馈到LED灯上.
GPSR2 |=  0x08000000;      //将91号引脚值置1.
led_value  =  GPLR2;
printf ("\n\n led_value = %x \n", led_value );
Led_Addr = led_value >>27;      //将91号引脚值反馈到LED灯0比特位上.
//将91号引脚清零，重复上述操作并观察.
GPCR2 |=  0x08000000;      //将91号引脚清0.
led_value  =  GPLR2;
printf ("\n\n Led_value = %x \n", led_value );
Led_Addr = led_value >>27;
```

实验结果观察：

Led_value 初始值为 13b7 ffff，对应 91 号引脚的比特 27 位值为 0.

进行 91 号引脚置 1 后，led_value 值变为 1bb7 ffff，对应 led 灯不亮.

进行 91 号引脚清 0 后，led_value 值变为 13b7 ffff，led 灯亮.

与预期结果一致。

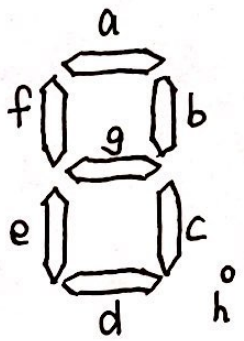
4. 选取 90 号引脚判断该引脚状态，如果高电平，数码管显示"引脚编号-H"，如果低电平显示"90-L".

设计：① 判断 90 号引脚电平状态：

设置 GPDR2 的比特 26 位为 1（输出），读取 GPLR2 的值，将其右移 26 位使 90 号引脚对应状态在 0 比特位上，对移位后的值除以 2 取余，结果即为电平状态。若值为 1，显示"90-H"，若为 0，显示"90-L".

② 将数码管显示对应符号.

根据数码管排列方式及性质，若管亮则将对应位置 0，否则置 1.



	显示数字/符号	对应 hgfedcba	转换成16进制
①	9	00010000	10H
②	0	01000000	40H.
③	-	00111111	3fH.
④	H	00001001	09H.
⑤	L	01000111	47H.

编码：

```
int led_value, a;          // a 用于存放 90 号引脚状态.
GPDR2 |= 0x04000000;       // 将 90 号引脚方向设为输出.
led_value = GPLR2;
printf("\n\n led_value =%x \n", led_value);
a = (led_value >>26) %2;   // 将 90 号引脚电平状态赋值为 a.
LED_CS2 = 0x4010;          // 让左侧数码管显示 90.
```

```
if(a){
    LED-CS3 = 0x093f;          //如果为高电平，显示"-H"
}
else{
    LED-CS3 = 0x473f;          //如果为低电平，显示"-L".
}
```

拓展设计：

```
GPSR2 |= 0x04000000;
        //将90号引脚值设为高电平重复上述操作.
```

(重复代码略).

```
GPCR2 |= 0x04000000;
        //将90号引脚值设为低电平重复操作.
```

实验结果分析：

初值 led_value = 13b7ffff，90号引脚对应低电平，数码管显示"90-自"，结果正确.

将90号引脚置1后，数码管显示"90-H"；清零后显示"90-L"，与预期设计一致。

GPIO实验(二).

设置3*3键盘、或4*4键盘，或者自选一种按键模式，分析记录各个按键值.

实验结果分析：

当未按下键盘时，KPAS值为ff；按下后值改变并显示 key-value，松开后恢复为ff。在键盘中从左至右列编号为0、1、2、5，从上至下行编号为0、1、2、6。故当按下键盘时，key-value的值是行、列编号(0行不显示).

实验结果与猜测相同.