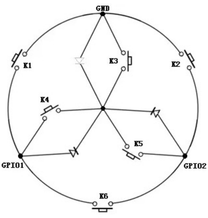
2IO检测6按键原理图分析

数码 2020-01-17 21:50 644阅读 · 25喜欢 · 23评论

--电子工作室-- UP
粉丝：1.6万 文章：33

2IO6按键原理图

特别说明（此电路不能检测多个按键同时按下）



上图为本文的原理图，2个GPIO即可检测6个按键，电路需要2个GPIO，6个按键，2个二极管（普通的1N4148即可）

检测原理分析：K1,K2是两个独立按键，如果GPIO1==0，K1被按下，同样GPIO2==0,K2被按下。我们再看K3，如果K3被按下，GPIO1,GPIO2均为0，所以，如果检测到这三种情况，我们可以通过GPIO1,GPIO2的状态去分析。如果GPIO1,GPIO2,均为1，证明没有按键按下，那么K1,K2,K3按键的检测，天生的第一优先级。

在GPIO1,GPIO2均为高电平情况下：

我们可以检测K4,K5,K6。这时候的检测可以叫主动检测。

我们先让GPIO1=0；此时记录GPIO2的状态，如果为0，证明K6,或者K4被按下，

检测后，我们让GPIO恢复高电平，避免影响下次检测。

下一步，我们让GPIO2=0，记录GPIO1的状态，如果GPIO1为0，可能是K5按下或者K6按下。

上边的两次主动拉低去检测另一个，如果两次都检测到低电平，一定是K6被按下，如果只有一次是低电平，那就是K4,或者K5被按下了，所以，我们要保存检测信息。

通过上述原理分析可以看出，这个电路是不支持两个按键同时被按下的那样会产生错误的信息，如，K4,K5同时按下，会被检测为K6。K1,K2同时被按下的话，会被检测为K3。

所以不支持同时按下两个按键，当然，同时按下两个按键貌似是不可能的，一定会存在时间差，只要这个时间差大于10ms（按键程序处理时间），那么，第二个按键程序就不会进行响应，唯有两个按键均处于释放状态，系统才会进行下次按键的检测处理。

下边，我将对程序进行分析。

```
sbit k1=P2^0;
sbit k2=P2^1;
sbit k0=P3^3;
u8 bdata kv=0x0f;

sbit kv1=kv^0;
sbit kv2=kv^1;
sbit kv3=kv^2;
sbit kv4=kv^3;
```

首先是变量定义部分，K1,K2代表GPIO1,GPIO2，同时定义了一个可以位寻址的unsigned char变量kv，用来存储按键检测的数据

```
void key_weiread()
{
    if(k1&&k2)
    {
        kv1=1;
        kv2=1;

        k1=0;
        _nop_();_nop_();//_nop_();_nop_();
        kv3=k2;
        k1=1;

        k2=0;
        _nop_();_nop_();//_nop_();_nop_();
        kv4=k1;
        k2=1;
    }
    else
    {
        kv3=1;
        kv4=1;
        kv1=k1;//
        kv2=k2;//
    }
}
```

void key_weiread()函数，是读取按键信息的函数，我们可以看到分为两个部分，首先我们看else部分，此时K1,K2至少有一个IO口被拉低，那么这个就是我们电路中前3个按键，

我们把检测的信息保存在KV1,KV2中，

我们再看复杂的if部分，这部分我们是先让K1=0，把K2的信息保存在KV3，然后K2=0，把K1的信息保存在KV4。这样，我们就保存了所有需要的信息，下一步就是要对KV变量进行处理。

```
void Key_scan()
{
    static bit flag=1;
    static u8 count=0;
    key_weiread();
    if(kv!=15&&flag)
    {
        count++;
        if(count>10)
        {
            count=0;
            switch(kv)
            {
                case 0x0c:key_data=1;break;
                case 0x0d:key_data=2;break;
                case 0x0e:key_data=3;break;

                case 0x0b:key_data=4;break;
                case 0x07:key_data=5;break;
                case 0x03:key_data=6;break;
            }
            flag=0;
        }
    }
    if(kv==15)//没有按键按下
    {
        count=0;
        flag=1;
    }
}
```

void Key_scan()这个函数是放在1ms一次的定时器中断里，首先，当KV==15时，是没有按键按下的（每次进入函数，程序都会读取改变KV），那么这个时候，我们允许对按键进行处理，也就是falg=1；这个也算是一个自锁操作，防止程序多次处理按键。当有按键按下时，KVI=15,同时，因为我们是从没有按键按下的情况来的，所以，flag=1；进入if，此时count++，只有加够10次，才能进入处理函数。如果连续10次都有按键按下，证明按键一定按下，那么我们让count=0；（可以不写），然后进入switch，KV一定在那几个情况里边，我们可以得到一个对应的key_data，然后我们可以对key_data进行switch语句。当然，也可以直接在switch（kv）进行处理。处理后，flag=0；就禁止再次进入了，保证一次按下不会多次触发，只有松开按键，才会再次进行处理检测。

这个代码暂时分析到这里，切记，一定要把Key_scan()放在定时器中断里（1KHz），保证检测的稳定性。当然也可以增加count数，不过检测时间长度就不确定了

如有疑惑，可加入群-电子制作/单片机编程交流群-见主页

欢迎关注BILIBILI-UP主：电子工作室

代码视频讲解地址https://www.bilibili.com/video/av83693426