第3节　单片机 C语言程序设计举例

一、概述

二、举例

例1.　发光二极管的控制

例2.　节日彩灯的控制

例3.　用按键控制节日彩灯显示的变化

一、概述

1. 编写C语言程序步骤

2. 判断程序质量的标准

3. 几种程序结构

1. 编写程序步骤

- 分析实际问题，抽象描述问题的模型
- 确定解决模型的算法
- 按算法画出程序流程图
- 按流程图编写程序
- 上机调试、运行程序

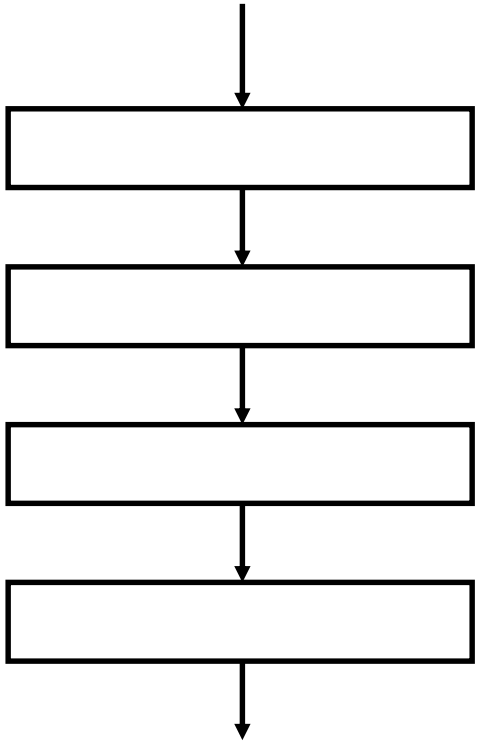
2. 判断程序质量的标准

- 程序的正确性
- 程序的可读性
- 程序的执行时间
- 程序所占内存大小

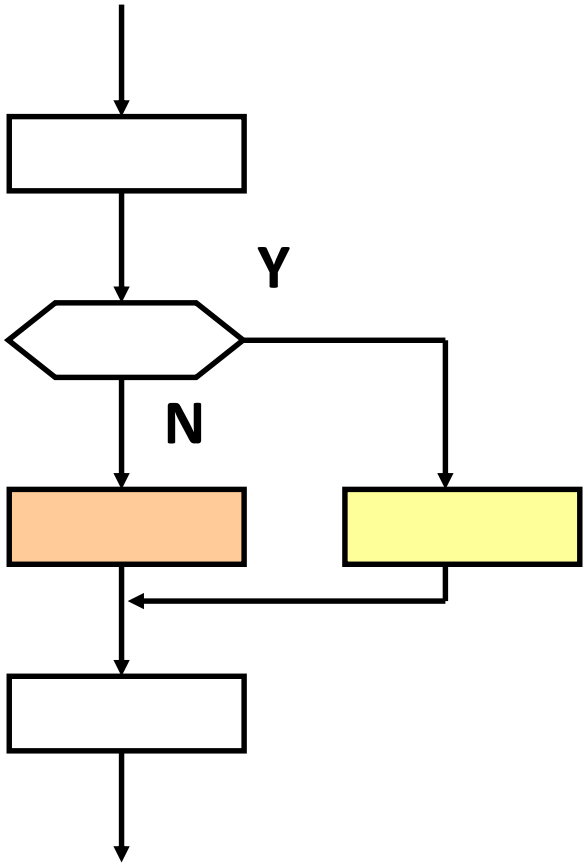
3. 几种程序结构

- 顺序结构
- 分支结构
- 循环结构
- 子程结构

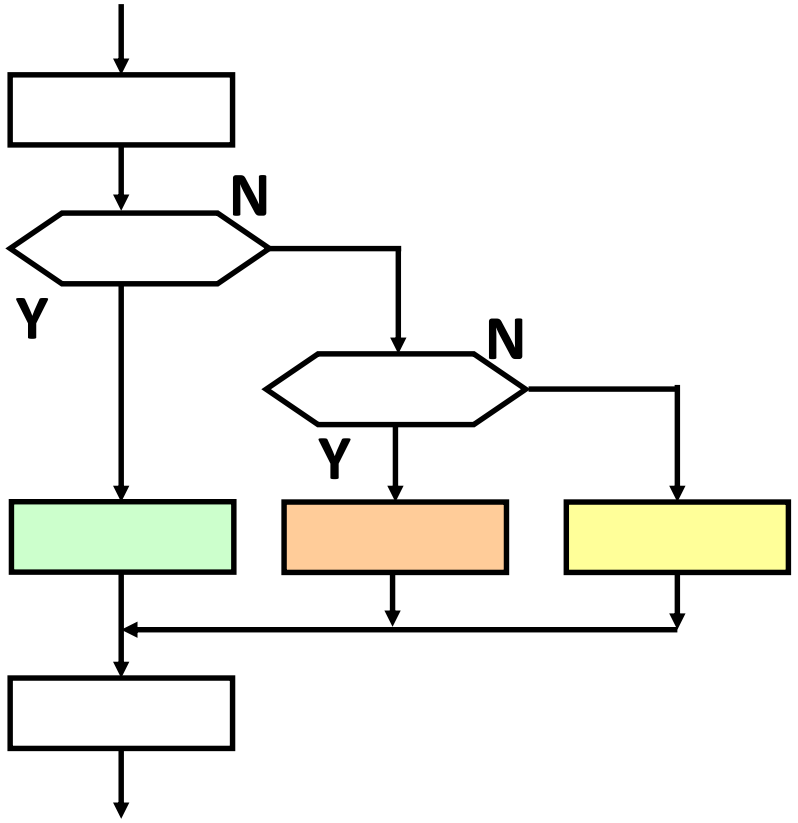
顺序结构



分支结构



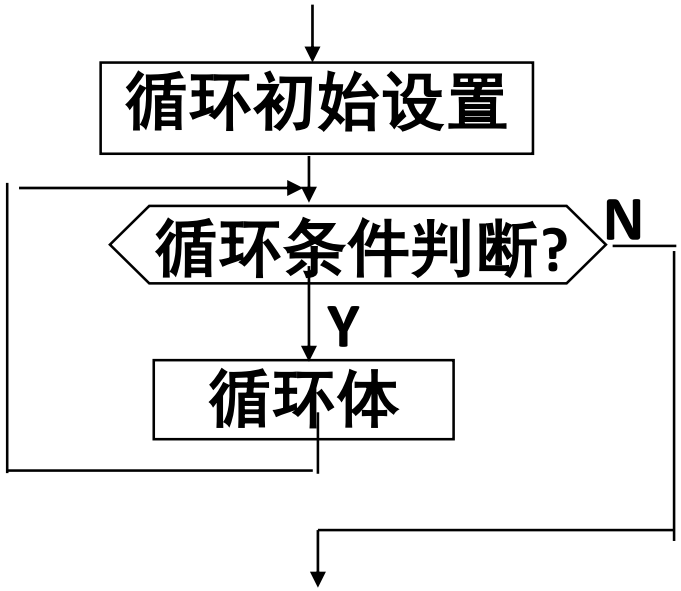
两个分支



三个分支

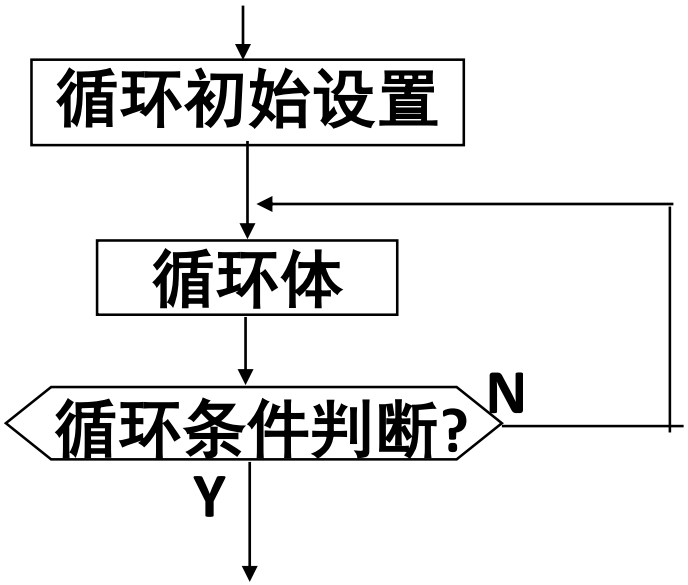
if 语句、swich 语句等

循环结构



当型循环

(当条件成立进入循环)



直到型循环

(直到条件成立退出循环)

for 语句、while 语句 、do... while语句等

函数调用

1. 多处调用完成同一功能的函数：

```
            ……
int  main(void)
{           ……
            delay();
            ……
             delay();
            ……
             delay();
            ……
}

void    delay( )
{           ……
            ……
}
```

2．利用函数进行模块化程序设计：

```
            ……
int  main(void)
{           ……
            IOinit();
            TAinit();
            ADCinit();
            ……
}
void    IOinit( )
{           …….       }

void    TAinit( )
{           …….       }

void    ADCinit( )
{           …….       }
```

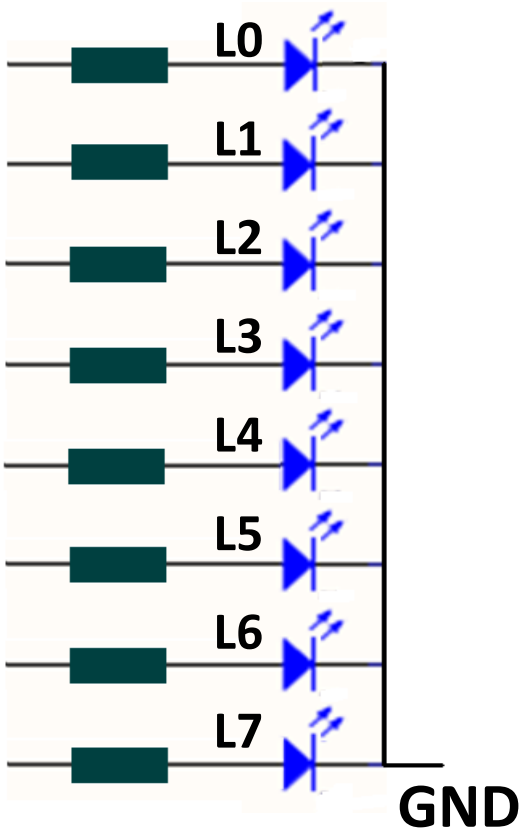
二、C语言程序设计举例

例1. 发光二极管的控制

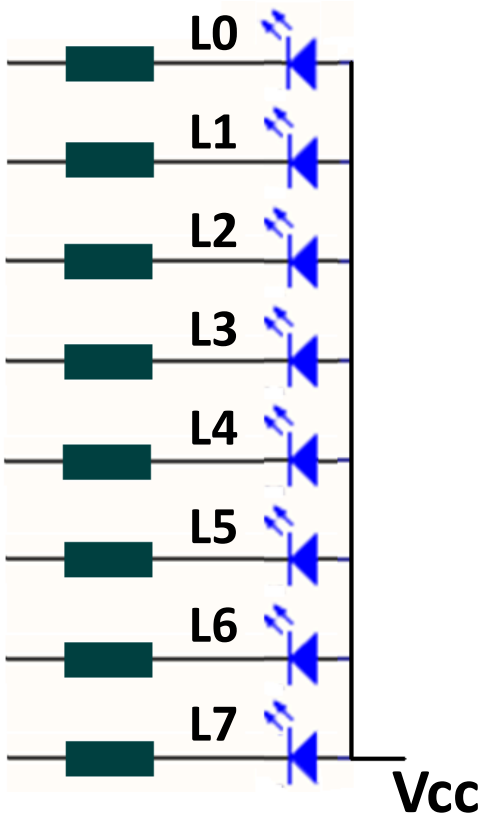
例2. 节日彩灯的控制

例3. 用按键控制节日彩灯显示的变化

发光二极管电路图和工作原理



共阴连接



共阳连接

灯的三种工作状态：亮、灭、闪

例1.

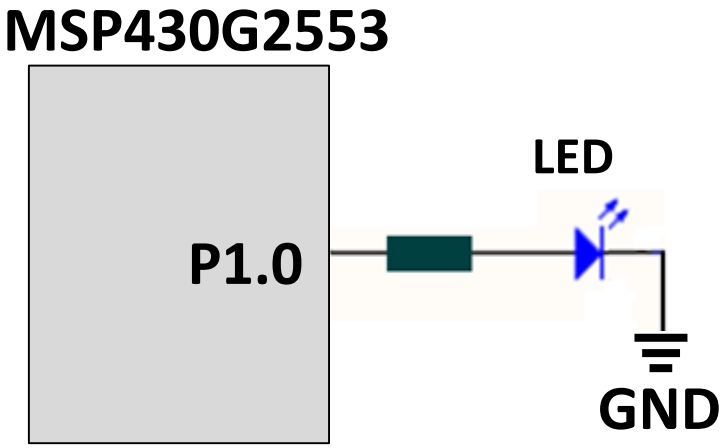
连接如右图，

编程控制发光二极管

从亮 → 灭 → 闪 →灭

. . .

如此不断反复



注意：硬件电路的连线设计及单片机底层相关硬件的软件设置关系

分析：

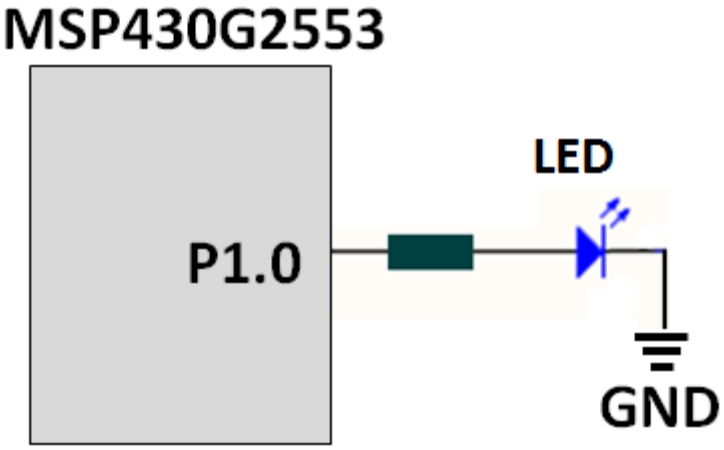
发光二极管LED经限流电阻

与单片机的引脚P1.0相连;

通过控制在该引脚，

输出高电平，LED亮；

输出低电平，LED灭。



编程时:

使P1OUT的D0位为1，可控制引脚P1.0输出高电平, 则LED亮；

为0，低电平, 则LED灭。

问题：

如何编程控制P1OUT的D0位的值？

有关的端口寄存器：P1SEL, P1SEL2, P1DIR，P1OUT

msp430G2553.h中的定义例

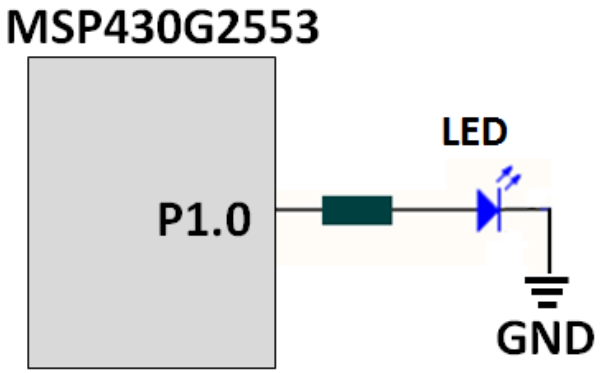
```
extern volatile unsigned char P1IN;
extern volatile unsigned char P1OUT;
extern volatile unsigned char P1DIR;
extern volatile unsigned char P1SEL;
extern volatile unsigned char P1SEL2;
extern volatile unsigned char P1REN;

extern volatile unsigned char P2IN;
extern volatile unsigned char P2OUT;
extern volatile unsigned char P2DIR;
extern volatile unsigned char P2SEL;
extern volatile unsigned char P2SEL2;
extern volatile unsigned char P2REN;
```

方法1： 使用for循环做延时

C3_e1_1.c

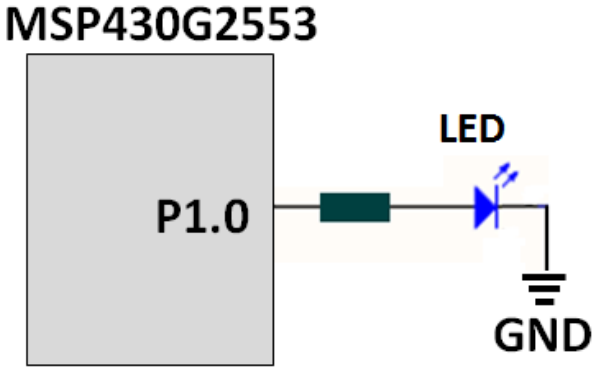
```
#include "msp430.h"
//unsigned long  j;                    //定义全局变
int main ( void )
{ unsigned long  j;                    //定义局部变量
   char  i;                            // 默认是 unsigned char
   WDTCTL = WDTPW + WDTHOLD;           //关闭看门狗
   P1SEL &= ~BIT0;                     //设置引脚P1.0为基本输入输出功能
   P1SEL2 &= ~BIT0;
   P1OUT &= ~BIT0;                     //设置引脚P1.0输出的初值为0
   P1DIR |= BIT0;                      //设置引脚P1.0为输出方向
   while (1)                           //主循环
    { P1OUT |= BIT0;                   //亮
      for (j=0; j<0xf000; j++);        //亮延时
      P1OUT &= ~BIT0;                  //灭
     for (j=0; j<0xf000*2; j++);       //灭延时
     for (i=0 ;i<4; i++)               //闪4下
     {  P1OUT |= BIT0;                 //闪亮
        for (j=0; j<0x4000; j++);      //闪亮延时
        P1OUT &= ~BIT0;                //闪灭
        for (j=0; j<0x4000; j++);      //闪灭延时
      }
      P1OUT &= ~BIT0;                  //灭
     for (j=0; j<0xf000*2; j++);       //灭延时
    };
}
```



方法2：使用延时函数

C3_e1_2.c

```
#include "msp430.h"
void delay(unsigned long numLoop);
int main ( void )
{   char  i;                                  //定义局部变量
    WDTCTL = WDTPW + WDTHOLD;   //关闭看门狗
    P1SEL &= ~BIT0;             //设置引脚P1.0为基本输入输出功能
    P1SEL2 &= ~BIT0;
    P1OUT &= ~BIT0;             //设置引脚P1.0输出的初值为0
    P1DIR |= BIT0;              //设置引脚P1.0为输出方向
    while (1)                   //主循环
    { P1OUT |= BIT0;            //亮
      delay(0xf000);            //亮延时
      P1OUT &= ~BIT0;           //灭
      delay(0xf000*2);          //灭延时
      for (i=0;i<4;i++)         //闪4下
      { P1OUT |= BIT0;          //闪亮
        delay(0x4000);          //闪亮延时
        P1OUT &= ~BIT0;          //闪灭
        delay(0x4000);          //闪灭延时
      }
      P1OUT &= ~ BIT0;          //灭
      delay(0xf000*2);          //灭延时
    };
}
void delay(unsigned long numLoop)
{   unsigned long  j;
    for (j=0; j<numLoop; j++);  //for 循环延时
}
```



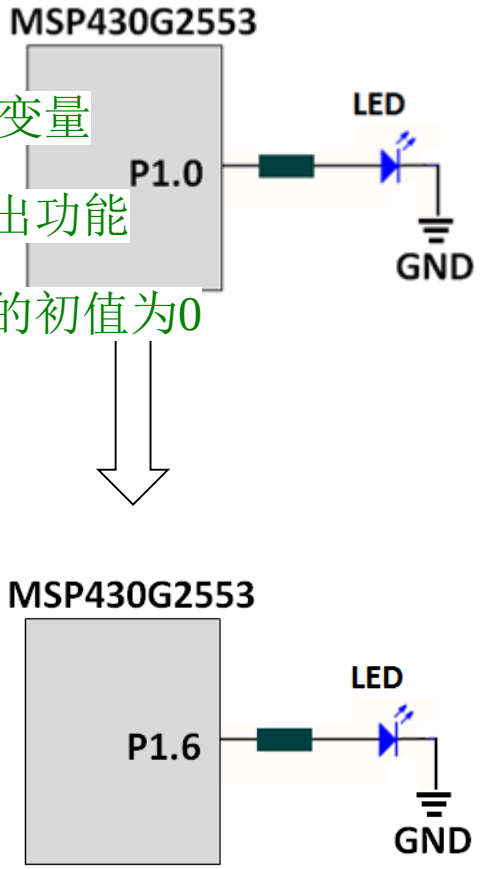
思考:

如何调整亮、灭、闪的时间长短?

方法3：自定义延时符号常量

C3_e1_3.c

```
#include "msp430.h"
#define  bright  0xf000     //亮延时数
#define  black   0xf000*2   //灭延时数
#define  blink   0x4000     //闪延时数
void delay(unsigned long numLoop);
int main ( void )
{   char  i;                                          //定义局部变量
    WDTCTL = WDTPW + WDTHOLD;  //关闭看门狗
    P1SEL &= ~BIT0;                                   //设置引脚P1.0为基本输入输出功能
    P1SEL2 &= ~BIT0;
    P1OUT &= ~BIT0;                                   //设置引脚P1.0输出的初值为0
    P1DIR |= BIT0;                                    //设置引脚P1.0为输出方向
    while (1)                          //主循环
    { P1OUT |=BIT0;                    //亮
      delay(bright);                                  //亮延时
      P1OUT &= ~BIT0;                  //灭
      delay(black);                                   //灭延时
      for (i=0;i<4;i++)                //闪4下
      { P1OUT |= BIT0;                 //闪亮
        delay(blink);                //闪亮延时
        P1OUT &= ~BIT0;                //闪灭
        delay(blink);;                                //闪灭延时
      }
      P1OUT &= ~BIT0;                                 //灭
      delay(black);                                   //灭延时
    };
}
void delay(unsigned long numLoop)（略）
```



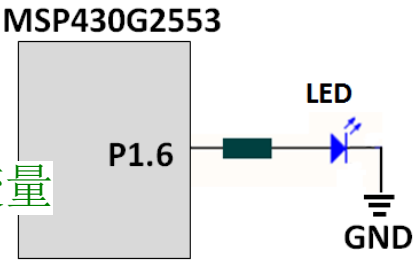
思考:

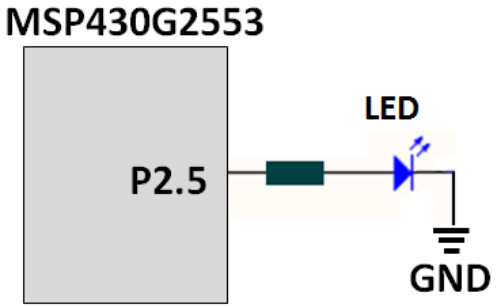
若LED不是连接在P1.0而是连接在P1.6上，如何修改程序?

方法4：自定义引脚符号

C3_e1_4.c

```
#include "msp430.h"
#define  bright  0xf000     //亮延时数
#define  black   0xf000*2   //灭延时数
#define  blink   0x4000     //闪延时数
#define  ledPin   BIT6      // LED引脚连接的位
void delay(unsigned long numLoop);
int main ( void )
{   char  i;                                          //定义局部变量
    WDTCTL = WDTPW + WDTHOLD;   //关闭看门狗
    P1SEL &= ~ledPin;           //设置引脚P1.0为基本输入输出功能
    P1SEL2 &= ~ledPin;
    P1OUT &= ~ledPin;           //设置引脚P1.0输出的初值为0
    P1DIR |= ledPin;            //设置引脚P1.0为输出方向
    while (1)                   //主循环
    { P1OUT |= ledPin;          //亮
      delay(bright);                          //亮延时
      P1OUT &= ~ledPin;                       //灭
      delay(black);                           //灭延时
      for (i=0; i<4; i++)       //闪4下
      {  P1OUT |= ledPin;       //闪亮
         delay(blink);          //闪亮延时
         P1OUT &= ~ledPin;      //闪灭
         delay(blink);;                       //闪灭延时
      }
      P1OUT &= ~ledPin;         //灭
      delay(black);                           //灭延时
    };
}
void delay(unsigned long numLoop)（略）
```





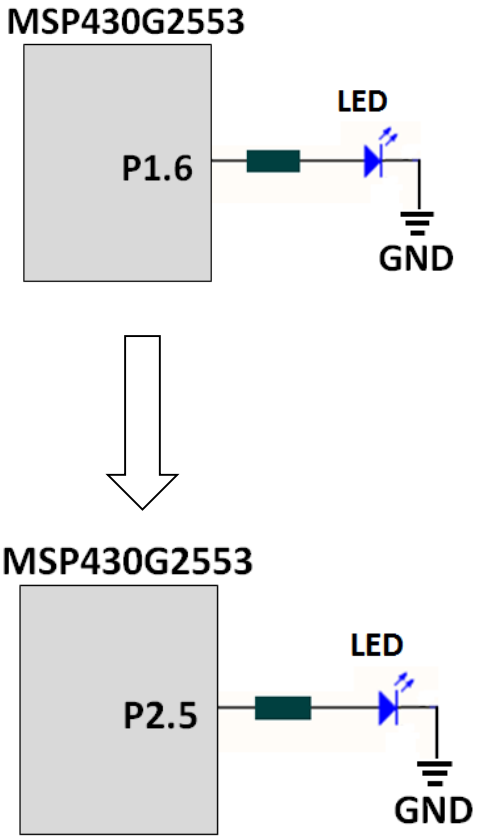
思考:

若LED不是连接在P1.6而是连接在P2.5上，如何修改程序?

方法5：定义端口符号

C3_e1_5.c

```
#include "msp430.h"
#define  bright    0xf000      //亮延时数
#define  black     0xf000*2    //灭延时数
#define  blink     0x4000      //闪延时数
#define  ledPin    BIT5        // LED引脚连接的位
#define  PledSEL   P2SEL       //LED引脚所对应的选择寄存器
#define  PledSEL2  P2SEL2      //LED引脚所对应的选择寄存器2
#define  PledOUT   P2OUT       //LED引脚所对应的输出寄存器
#define  PledDIR   P2DIR       //LED引脚所对应的方向寄存器
void delay(unsigned long numLoop);
int main ( void )
{   char  i;   //定义局部变量
    WDTCTL = WDTPW + WDTHOLD;  //关闭看门狗
    PledSEL &= ~ledPin;        //设置引脚P1.0为基本输入输出功能
    PledSEL2 &= ~ledPin;
    PledOUT &= ~ledPin;            //设置引脚P1.0输出的初值为0
    PledDIR |= ledPin;         //设置引脚P1.0为输出方向
    while (1)                  //主循环
    { PledOUT|= ledPin;        //亮
      delay(bright);           //亮延时
      PledOUT &= ~ ledPin;     //灭
      delay(black);                //灭延时
      for (i=0;i<4;i++)            //闪4下
      { PledOUT |= ledPin;     //闪亮
        delay(blink);          //闪亮延时
        PledOUT &= ~ledPin;    //闪灭
        delay(blink);                          //闪灭延时
      }
      PledOUT &= ~ledPin;      //灭
      delay(black);                            //灭延时
    };
}
void delay(unsigned long numLoop)（略）
```

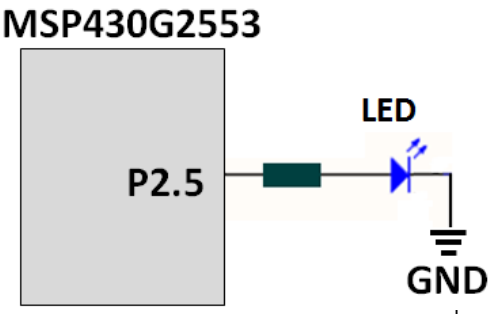


思考:

比较符号PledOUT的定义与P2OUT的定义有什么本质上的不同？

自定义硬件相关头文件 led.h

把底层与硬件关联的引脚、i/o寄存器，用符号预定义在一个.h头文件里。

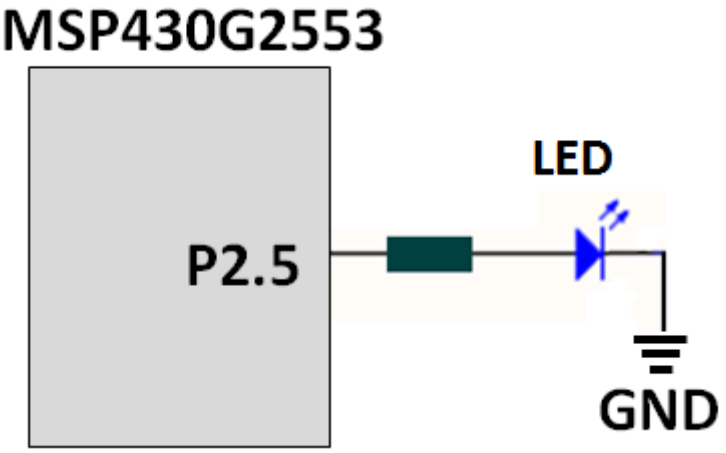



```
#define  bright      0xf000      //亮延时数
#define  black       0xf000*2    //灭延时数
#define  blink       0x4000      //闪延时数
#define  ledPin      BIT5        //LED 连接的引脚
#define  PledSEL     P2SEL       //LED引脚所对应的选择寄存器
#define  PledSEL2    P2SEL2      //LED引脚所对应的选择寄存器2
#define  PledOUT     P2OUT       //LED引脚所对应的输出寄存器
#define  PledDIR     P2DIR       //LED引脚所对应的方向寄存器
```

方法6：应用自定义硬件相关头文件

C3_e1_6.c

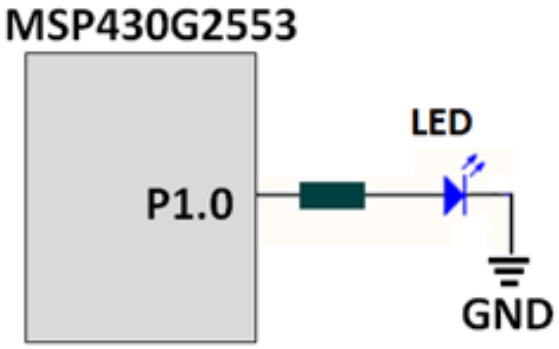
```
#include "msp430.h"
#include "led.h"
void delay(unsigned long numLoop);
int main ( void )
{   char  i;                                          //定义局部变量
    WDTCTL = WDTPW + WDTHOLD;  //关闭看门狗
    PledSEL &= ~ ledPin;       //设置led引脚为基本输入输出功能
    PledSEL2 &= ~ ledPin;
    PledOUT &= ~ ledPin;       //设置led引脚输出的初值为0
    PledDIR |= ledPin;         //设置led引脚为输出方向
    while (1)                  //主循环
    { PledOUT |=ledPin;        //亮
      delay(bright);                          //亮延时
      PledOUT &= ~ ledPin;     //灭
      delay(black);                           //灭延时
      for (i=0;i<4;i++)        //闪4下
      { PledOUT |= ledPin;     //闪亮
        delay(blink);          //闪亮延时
        PledOUT &= ~ ledPin;   //闪灭
        delay(blink);                         //闪灭延时
      }
      PledOUT &= ~ ledPin;     //灭
      delay(black);                           //灭延时
    };
}
void delay(unsigned long numLoop)（略）
```

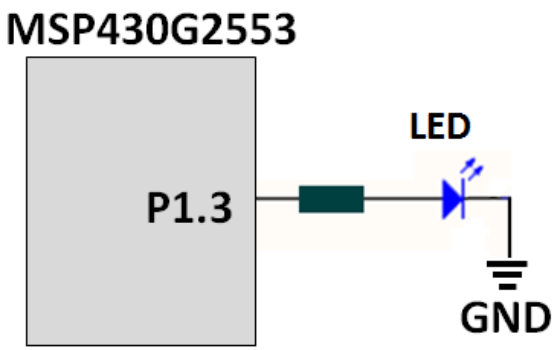


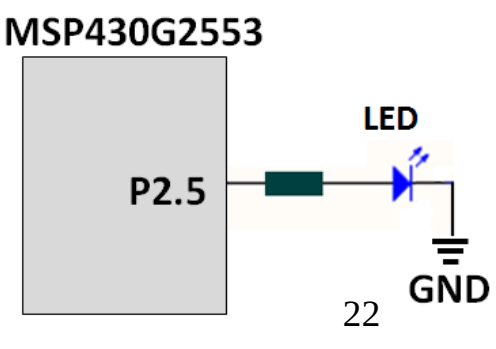
使用自定义硬件相关头文件 led.h

只需要根据实际的连线，改变该头文件中与硬件连线相关的定义，而不必修改 .c 程序中的语句。

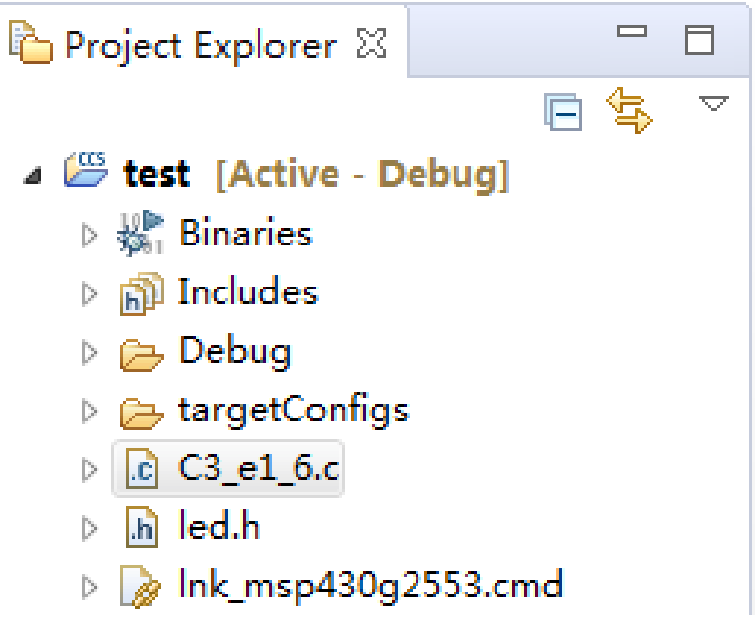
```
#define  bright  0xf000        //亮延时数
#define  black  0xf000*2       //灭延时数
#define  blink   0x4000        //闪延时数
#define  ledPin     BIT5       //LED 连接的引脚
#define  PledSEL    P2SEL       //LED引脚所对应的选择寄存器
#define  PledSEL2  P2SEL2       //LED引脚所对应的选择寄存器2
#define  PledOUT  P2OUT        //LED引脚所对应的输出寄存器
#define  PledDIR    P2DIR       //LED引脚所对应的方向寄存器
```







在CCS工程项目中, 需要把自定义头文件包含进来。

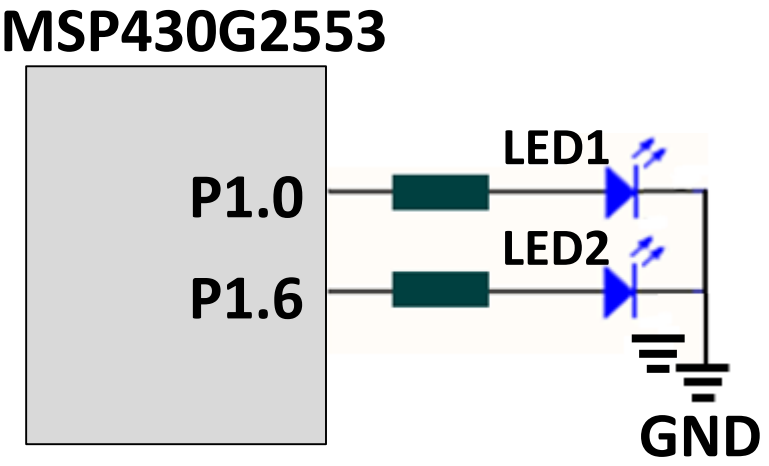


C3_e1_6.c

```
1  //例1.编程控制发光二极管从亮→ 灭→ 闪→灭， ...,如此不断反复
2  //方法6：自定义硬件相关头文件
3  #include "msp430.h"
4  #include "led.h"
5  void delay(unsigned long numLoop);
6  int main ( void )
7  {       char  i;
8      WDTCTL = WDTPW + WDTHOLD;  //关闭看门狗
9      PledSEL &=~ledPin;      //设置引脚P1.0为基本输入输出功能
10     PledSEL2 &=~ledPin;
11     PledOUT &=~ledPin;    //设置引脚P1.0输出的初值为0
12         PledDIR |=ledPin;    //设置引脚P1.0为输出方向
```

思考:

如图用P1.0和P1.6分别连接了两盏LED,编程控制两灯同时按亮、灭、闪、灭循环变化。



在例6 自定义硬件相关头文件 led.h的基础上，做如下修改，

例6其他部分的源程序不变。

```
#define  bright     0xf000        //亮延时数
#define  black      0xf000*2            //灭延时数
#define  blink      0x4000        //闪延时数
#define  ledPin     (BIT0+BIT6)  //LED 连接的引脚,注意加括号
#define  PledSEL    P1SEL         //LED引脚所对应的选择寄存器
#define  PledSEL2   P1SEL2        //LED引脚所对应的选择寄存器2
#define  PledOUT    P1OUT         //LED引脚所对应的输出寄存器
#define  PledDIR    P1DIR         //LED引脚所对应的方向寄存器
```


比较(BIT0+BIT6),有括号和无括号的不同？

例2.

连接如右图，

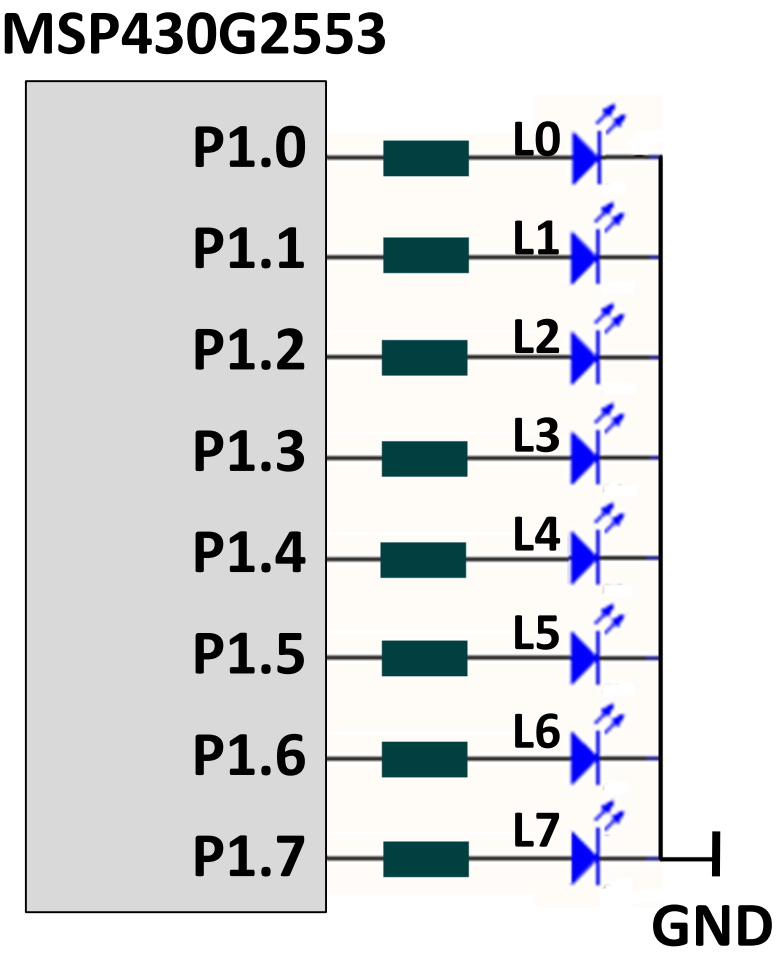
编程控制发光二极管

从L0 → L1... →L7 → L0...

一盏一盏点亮，

每次只有一盏亮，其他灯灭。

如此不断反复



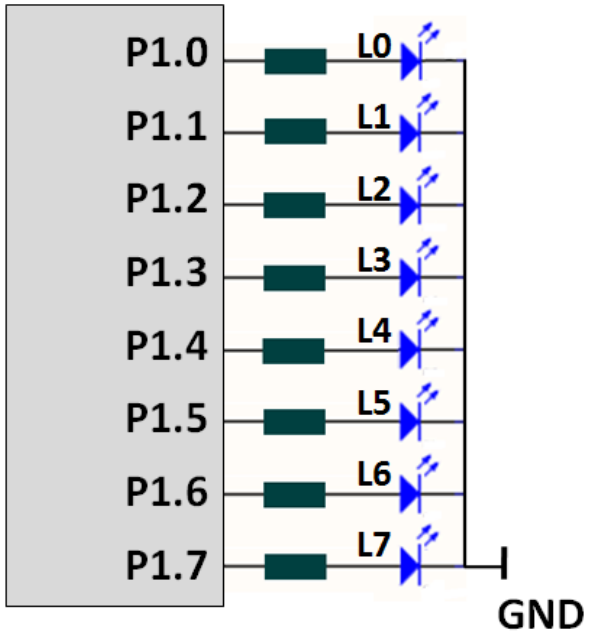
注意：硬件电路的连线设计及单片机底层相关硬件的软件设置关系

分析：

8个发光二极管与单片机的端口1相连;

通过控制与二极管相连接的引脚，

输出高/低不同的电平，可使二极管亮/灭。



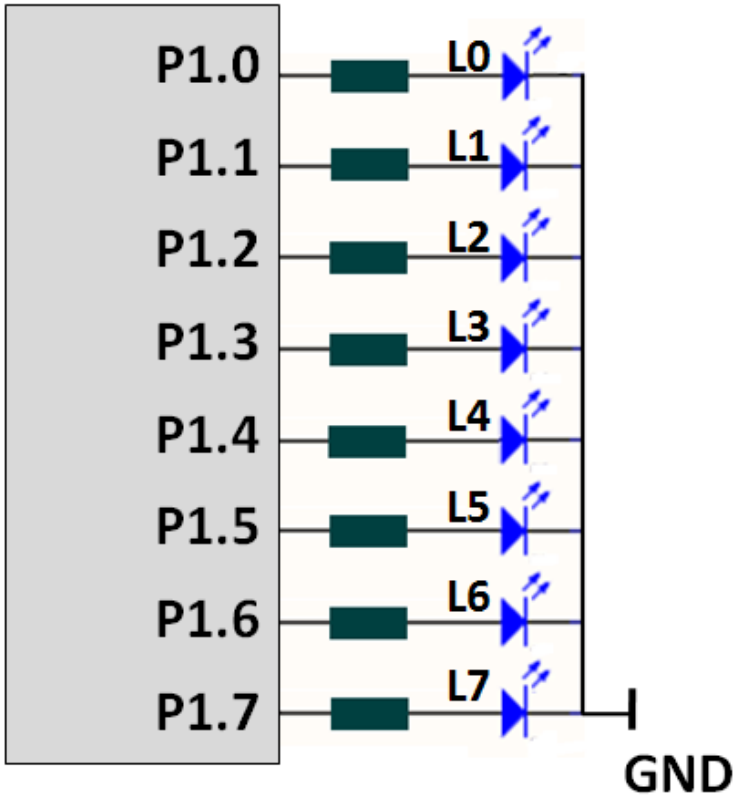
问题：

如何编程控制 端口引脚P1.0~P1.7的输出？

有关的端口寄存器：P1SEL, P1SEL2, P1DIR，P1OUT

分析8个发光二极管的变化规律

L7	L6	L5	L4	L3	L2	L1	L0	P1OUT
○	○	○	○	○	○	○	●	0x01
○	○	○	○	○	○	●	○	0x02
○	○	○	○	○	●	○	○	0x04
○	○	○	○	●	○	○	○	0x08
○	○	○	●	○	○	○	○	0x10
○	○	●	○	○	○	○	○	0x20
○	●	○	○	○	○	○	○	0x40
●	○	○	○	○	○	○	○	0x80



用数组存放循环显示的显示值

```
char LEDdata[8]={0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80};
```

与写成 `char LEDdata[8]={1, 2, 4, 8,0x10,0x20,0x40,0x80};`

有何不同?

例2 程序清单 C3_e2_1.c

```
#include "msp430.h"
void delay(unsigned long numLoop);
int  main( void )
{   char LEDdata[8]={0x01, 0x02, 0x04, 0x08,0x10,0x20,0x40,0x80};
    signed char i;
    WDTCTL = WDTPW + WDTHOLD;  //关闭看门狗
    P1SEL=0x00;                          //设置P1的8个引脚为基本I/O
    P1SEL2=0x00;                                  //
    P1OUT=0x00;                                   //使8个LED全灭
    P1DIR=0xFF;                                   //设置P1的8个引脚为输出端口
    while(1)                             //主循环
    {   for ( i=0; i<8; i++ ) //8个LED依次点亮
        {  P1OUT=LEDdata[i];   //
           delay(0xf000);     //调用延时子程
        }
        };
}
```

思考:

1. 若发光二极管不是连接在P1而是连接在P2上,如何修改程序?
2. 若要求是L7→ L6.... →L0 →L7循环点亮,如何修改程序?
3. 相邻两个亮L0 L1→ L1 L2.... →L6 L7 →L7 L0 ,如何修改程序?
4. 若发光二极管不是共阴而是共阳连接,如何修改程序?
5. 如何改写成用硬件抽象层的.h头文件的形式?


```
void delay(unsigned long numLoop)（略）
```

利用关键字 const 将常量数组存放到ROM中，省出RAM空间存放其他变量，程序清单C3_e2_2.c

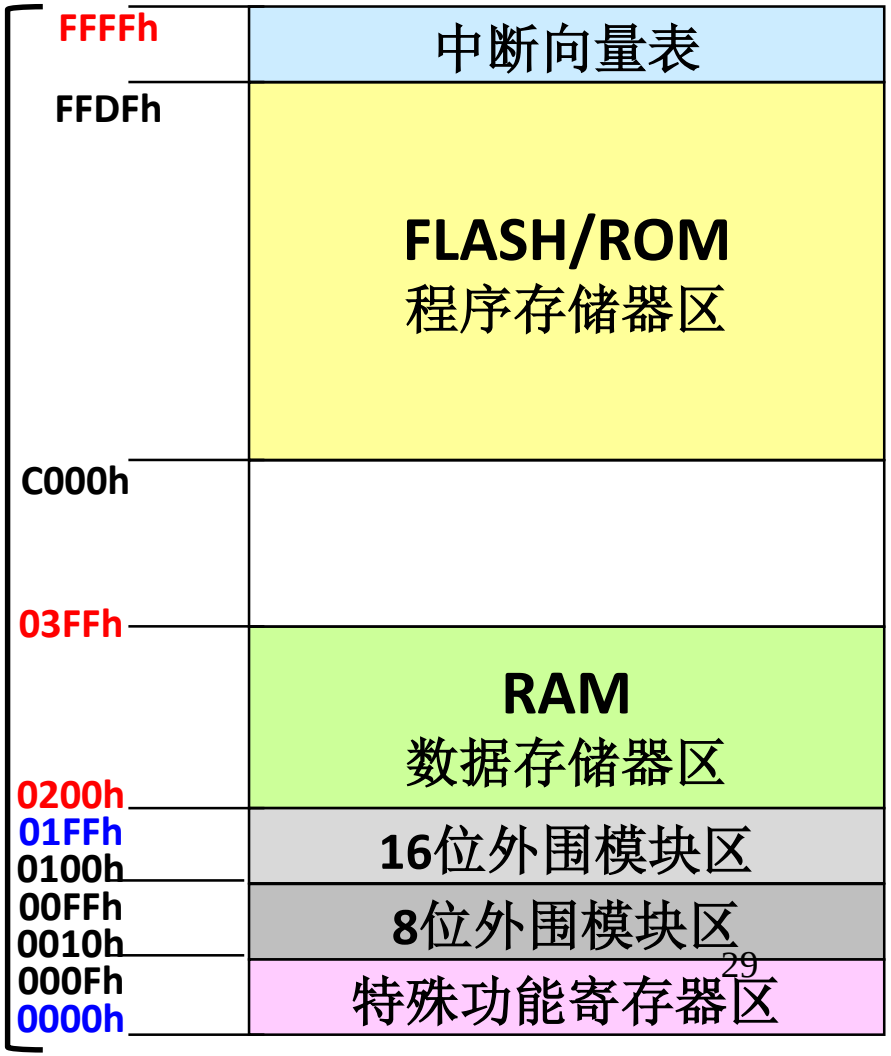
```
#include "msp430.h"                          //常量型变量定义
const char LEDdata[8 ]={0x01, 0x02, 0x04,0x08,
                        0x10,0x20,0x40,0x80};

        ……
int     main( void )
{       ……

}
```

msp430G2553

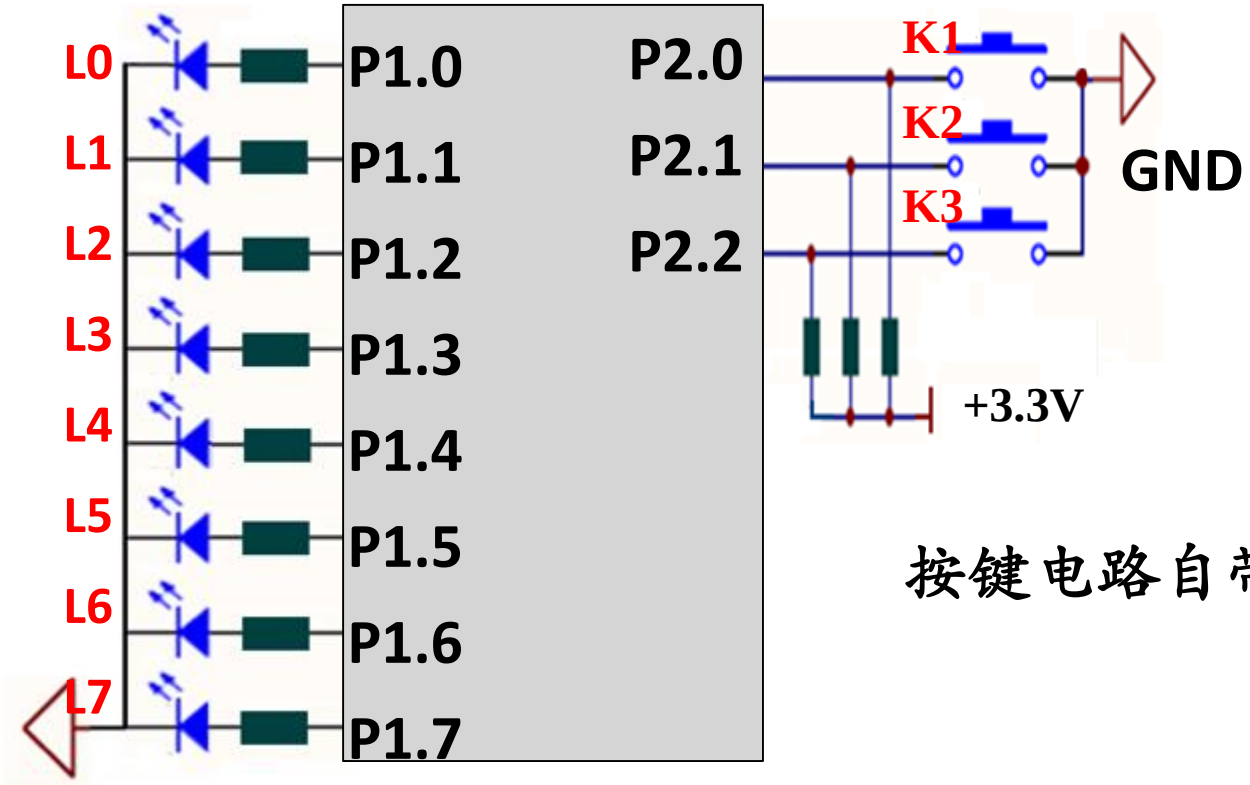
的存储器结构



方法2：用指针取数组元素(C3_e2_3.c)

```
#include "msp430.h"
void delay(unsigned long numLoop);
int          main( void )
{            unsigned char LEDdata[8 ]={0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80};
             unsigned  int  i;
             unsigned char  *p;
             WDTCTL = WDTPW + WDTHOLD;  //关闭看门狗
             P1SEL=0x00;                //设置P1的8个引脚为基本I/O
             P1SEL2=0x00;               //
             P1OUT=0x00;                //使8个LED全灭
             P1DIR=0xFF;                //设置P11的8个引脚为输出
             while(1)                   /主循环
             {            p=&LEDdata[0];          //指向数组首地址
                          for (i=0; i<8; i++)     //8个LED依次点亮
                          {   P1OUT=*p;
                              p++;
                              delay(0xf000);      //调用延时子程
                          }
             };
}
void delay(unsigned long numLoop)（略）
```

例3. 用按键控制节日彩灯显示的变化

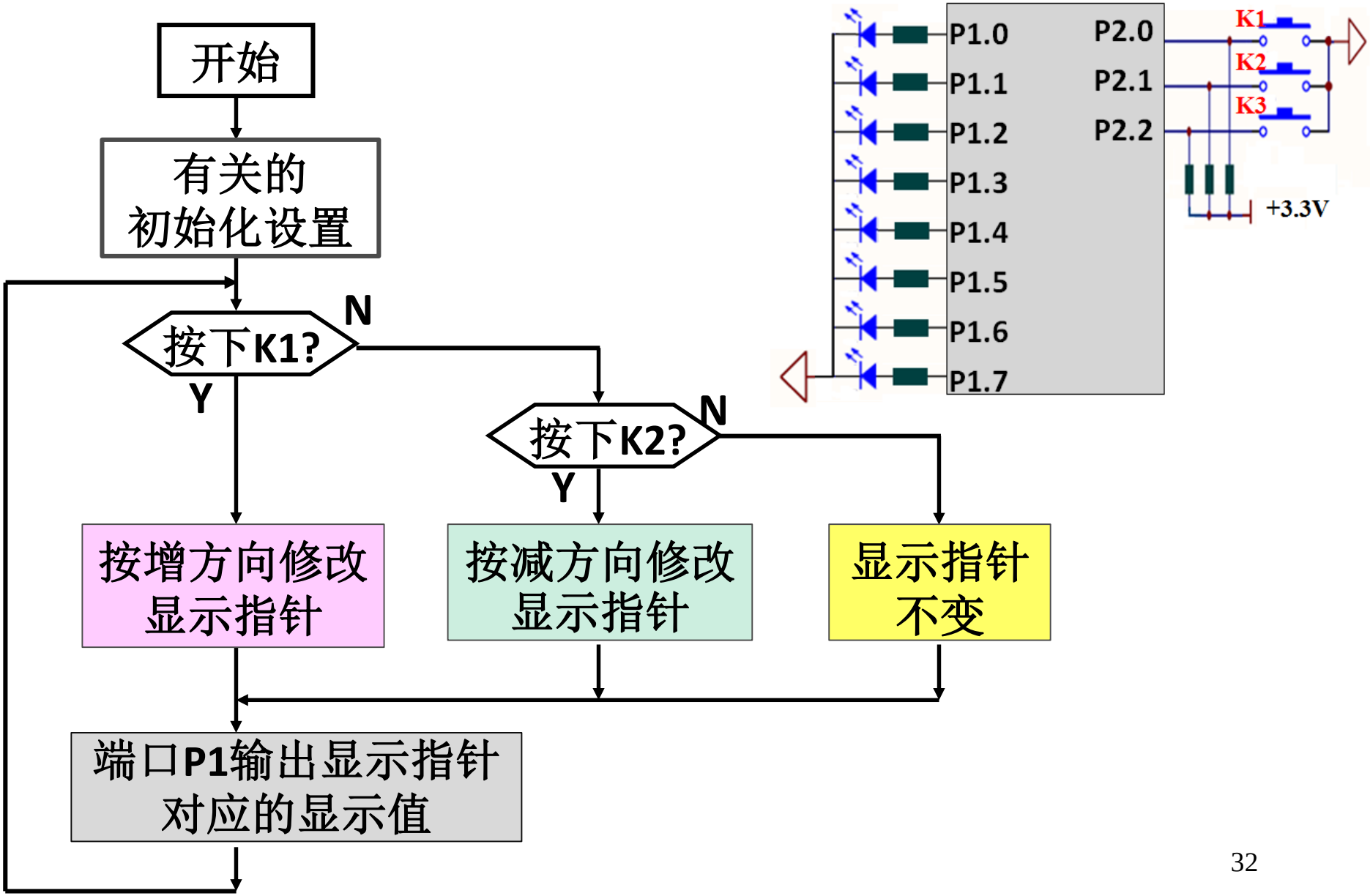


按键电路自带上拉电阻

当按下K1键时，灯从L0→ L1…→L7 →L0…循环点亮，
当按下K2键时，灯从L7→ L6… →L0→L7…循环点亮，
当按下K3键时，灯的状态保持不变，停止变化
每次只有一个灯点亮

根据硬件连线，确定显示表格

LEDtab[8] = {0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80}



例3程序清单 C3_e3_1_1.c

```
#include "msp430.h"
const char LEDdata[8]={0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80};
int          main( void )
{   unsigned int j;                                    //延时控制变量
    signed char  i=0;                                  //置显示码数组初始下标
    WDTCTL = WDTPW + WDTHOLD;                          //关闭看门狗
    P1SEL=0x00;                                        //设置P1的8个引脚为基本I/O
    P1SEL2=0x00;
    P1OUT=0x00;                                //使8个LED全灭
    P1DIR=0xFF;                                //设置P1的8个引脚为输出端口
    P2SEL  &= ~(BIT0+BIT1+BIT2);               //设置引脚P2.0~2为基本I/O
    P2SEL2 &= ~(BIT0+BIT1+BIT2);
   P2DIR  &= ~(BIT0+BIT1+BIT2);                //设置引脚P2.0~2为输入
   while(1)                                                          //主循环
   {  if ( (P2IN&BIT0) ==0)                    //判断K1是否按下
      {      i=i+1;
             if  (  i==8 )  i=0;
       }                                        //到表尾，回表首
       else if  ( (P2IN&BIT1) ==0)             //判断K2是否按下
       {     i=i-1;
            if  ( i==-1 )  i=7;
        }                                       //到表首，回表尾
       P1OUT=LEDdata[i];                        //取显示码输出
      for ( j=0;  j<0xf000; j++);               //延时
   };
}
```

注意关系运算符==的优先级高于逻辑运算符&, 比较：
((P2IN & 0x01) ==0) 与 (P2IN & 0x01 ==0) 一样吗?

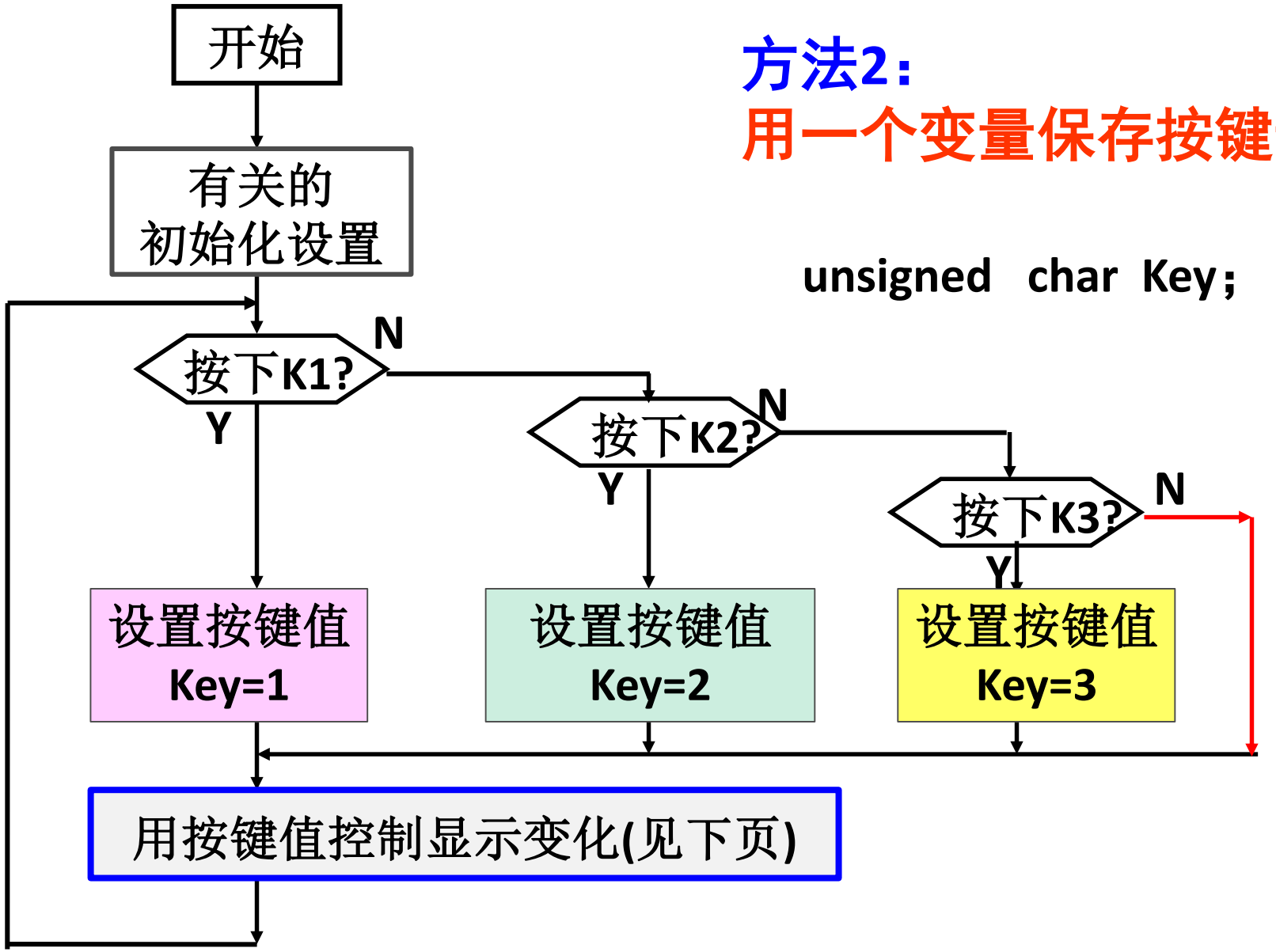
思考：

前面编写的程序，需要一直按着键，灯才显示变化，

可否修改程序，不需一直按键，

灯按最近一次的按键进行显示？

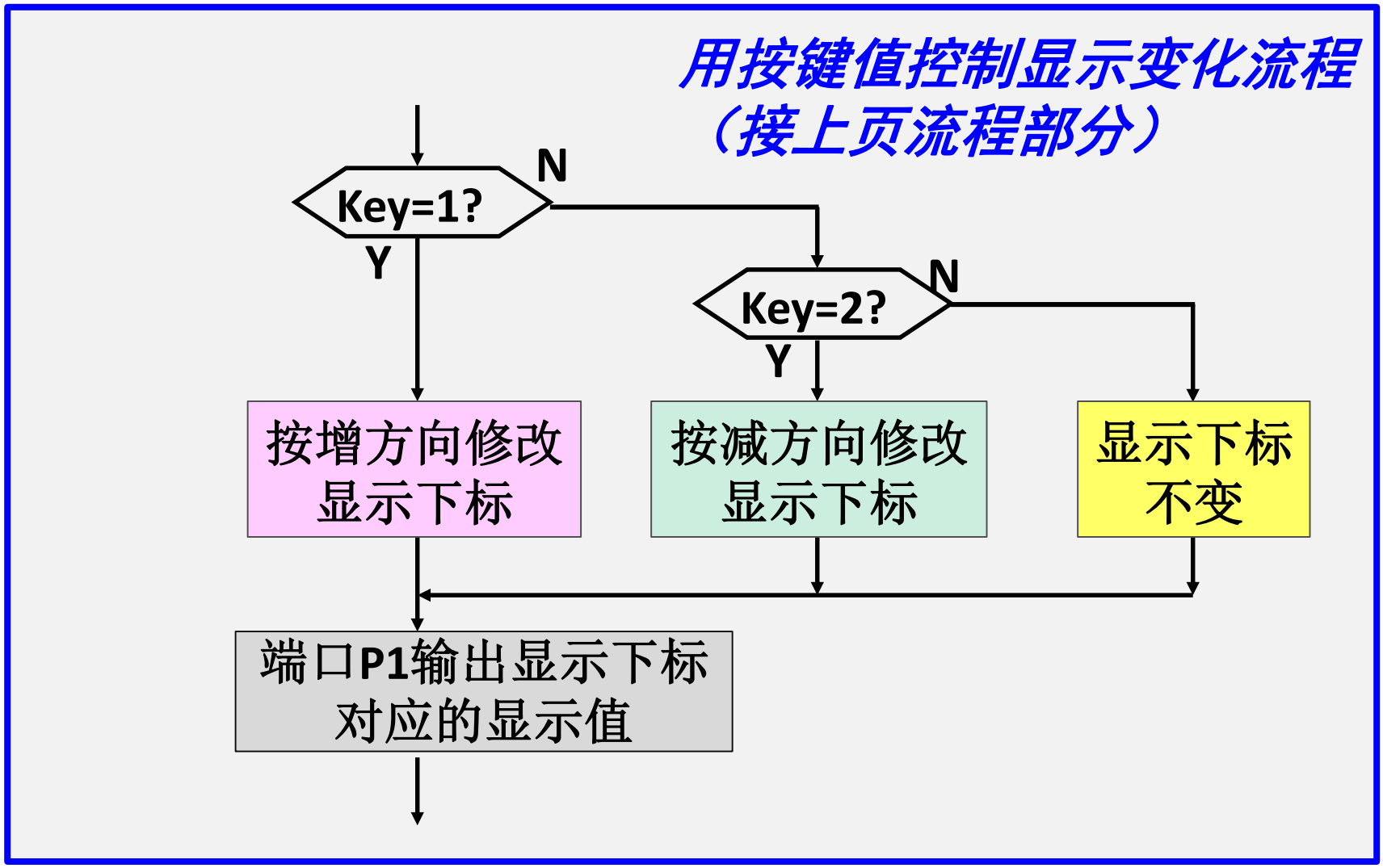
```
        ……
        while(1)                                    //主循环
        {       if   ( (P2IN&BIT0) ==0)             //判断K1是否按下
                  {       i=i+1;
                          if  (  i==8 )  i=0;  }    //到表尾，回表首
                else if  ( (P2IN&BIT1) ==0)         //判断K2是否按下
                  {       i=i-1;
                          if  ( i==-1 )  i=7;  }    //到表首，回表尾
              P1OUT=LEDdata[i];                     //取显示码输出
                for ( j=0;  j<0xffff;  j++);                //延时
        };
}
```



方法2：
用一个变量保存按键动作

```
unsigned  char  Key;
```

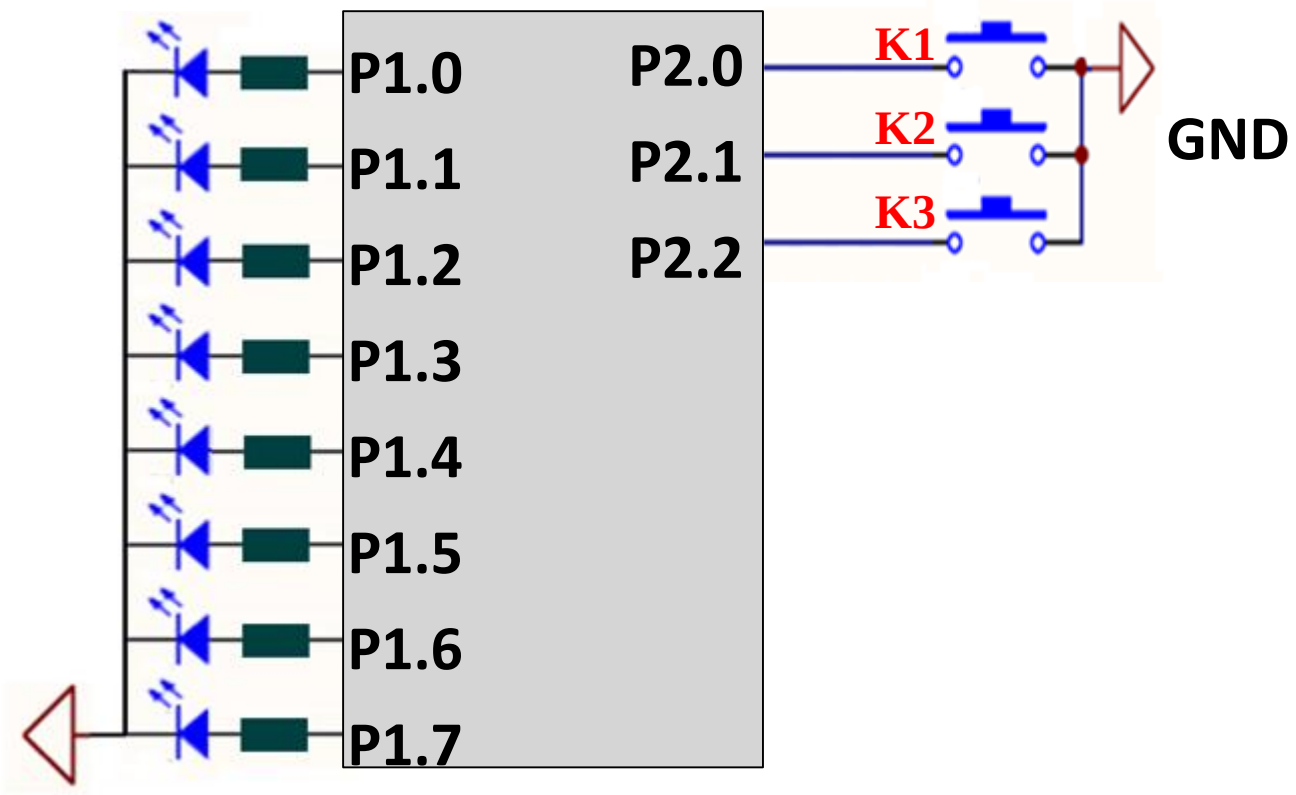
显示表格 LEDdata[8]={0x01, 0x02, 0x04, 0x08,0x10,0x20,0x40,0x80}



方法2: 用变量保存按键值 (C3_e3_2_1.c)

```
#include "msp430.h"
const unsigned char LEDdata[8]={0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80};
int main( void )
{   unsigned int j ;                                          //延时控制变量
    signed char i=0;                    //置显示码数组初始下标
    unsigned char Key=0;                //保存按键状态的变量
    WDTCTL = WDTPW + WDTHOLD;     //关闭看门狗
    P1SEL=0x00;                         //设置P1为基本I/O
    P1SEL2=0x00;                                                  //
    P1OUT=0x00;                         //使8个LED全灭
    P1DIR=0xFF;                         //设置P1为输出端口
    P2SEL  &= ~(BIT0+BIT1+BIT2);        //设置P2.0~2为基本I/O
    P2SEL2  &= ~(BIT0+BIT1+BIT2);
    P2DIR  &= ~(BIT0+BIT1+BIT2);        //设置P2.0~2为输入
    while(1)                                                      //主循环
    {   if   ((P2IN&BIT0)==0)   { Key=1; } //判断K1是否按下
       else if  ((P2IN&BIT1)==0) { Key=2; }                       //判断K2是否按下
       else if  ((P2IN&BIT2)==0) { Key=3; }                       //判断K3是否按下
      if   (Key==1)                     //据Key值控制显示码数组下标
      {  i=i+1;
         if  (  i==8 )  i=0;
       }
       else if  (Key==2)
        {  i=i-1;
          if  ( i==-1 )  i=7;
         }
        P1OUT=LEDdata[i];                   //取显示码输出
       for ( j=0;  j<0xf000;  j++);         //延时
      };
}
```

如图，如果线路中按键部分没有上拉电阻

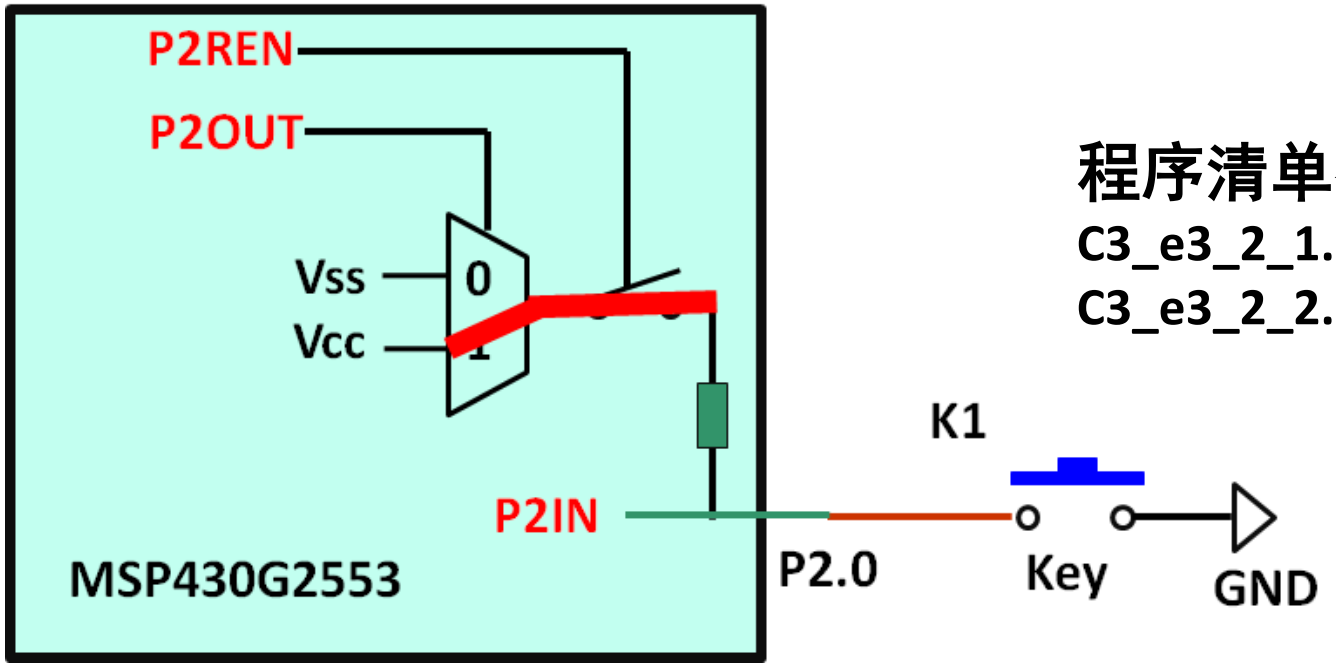


如何修改程序？

输入引脚需要使用内部上拉电阻，

需在初始化时，增加对输入引脚 置内部上拉电阻允许

```
P2SEL  &= ~(BIT0+BIT1+BIT2);    //置P2.0~2为基本I/O
P2SEL2  &= ~(BIT0+BIT1+BIT2);   //置P2.0~2为基本I/O
P2DIR  &= ~(BIT0+BIT1+BIT2);    //置P2.0~2为输入
P2OUT  |= BIT0+BIT1+BIT2;       //置P2.0~2内部电阻为上拉电阻
P2REN  |= BIT0+BIT1+BIT2;       //置P2.0~2内部电阻允许
```



程序清单参考：

C3_e3_2_1.c

C3_e3_2_2.c