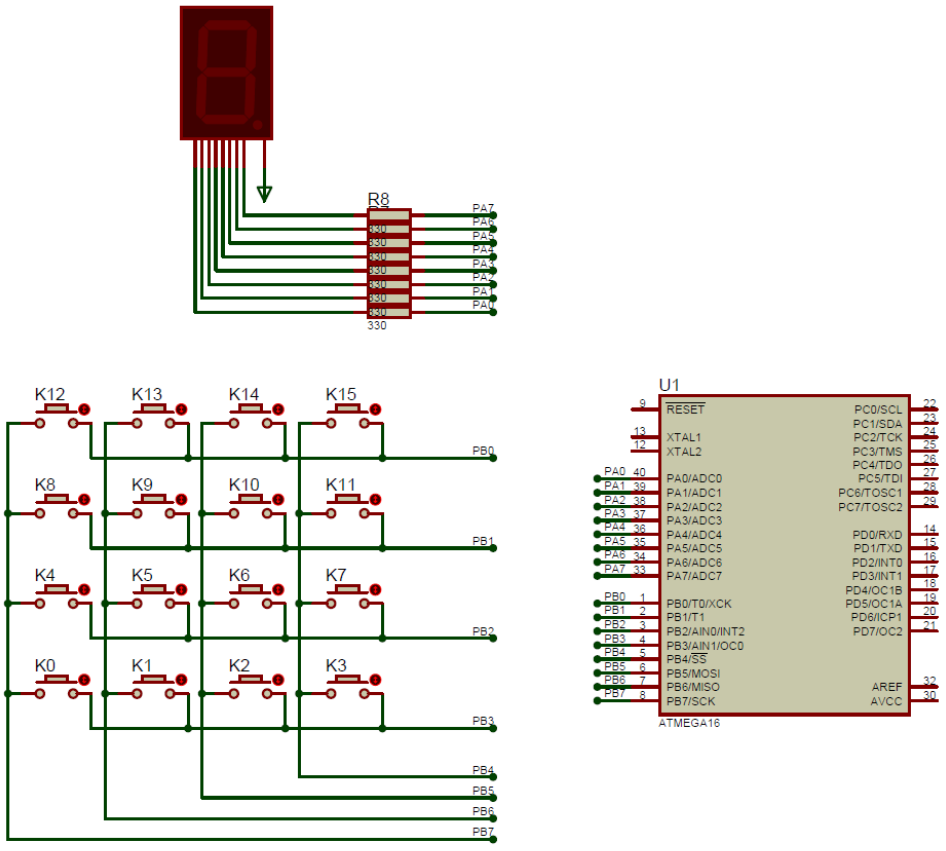
实验 7：矩阵按键识别

1. 试验描述：

矩阵按键部分由 16 个轻触按键按照 4 行 4 列排列，连接到 PB 口。将行线所接的单片机 IO 输出低电平，列线所接的 IO 口作为输入并拉高。无按键按下时，输入端均为高电平；有按键按下时，该按键对应的列线连接的 IO 输入低电平，因此可以获取按下按键的列位置。反之将行线所接的 IO 口作为输入并拉高，列线所接的单片机 IO 输出低电平，可以获取按下按键的行位置。此方法比较简单，但其局限在于同时按下多个按键时无法判断。

2. 系统框图：

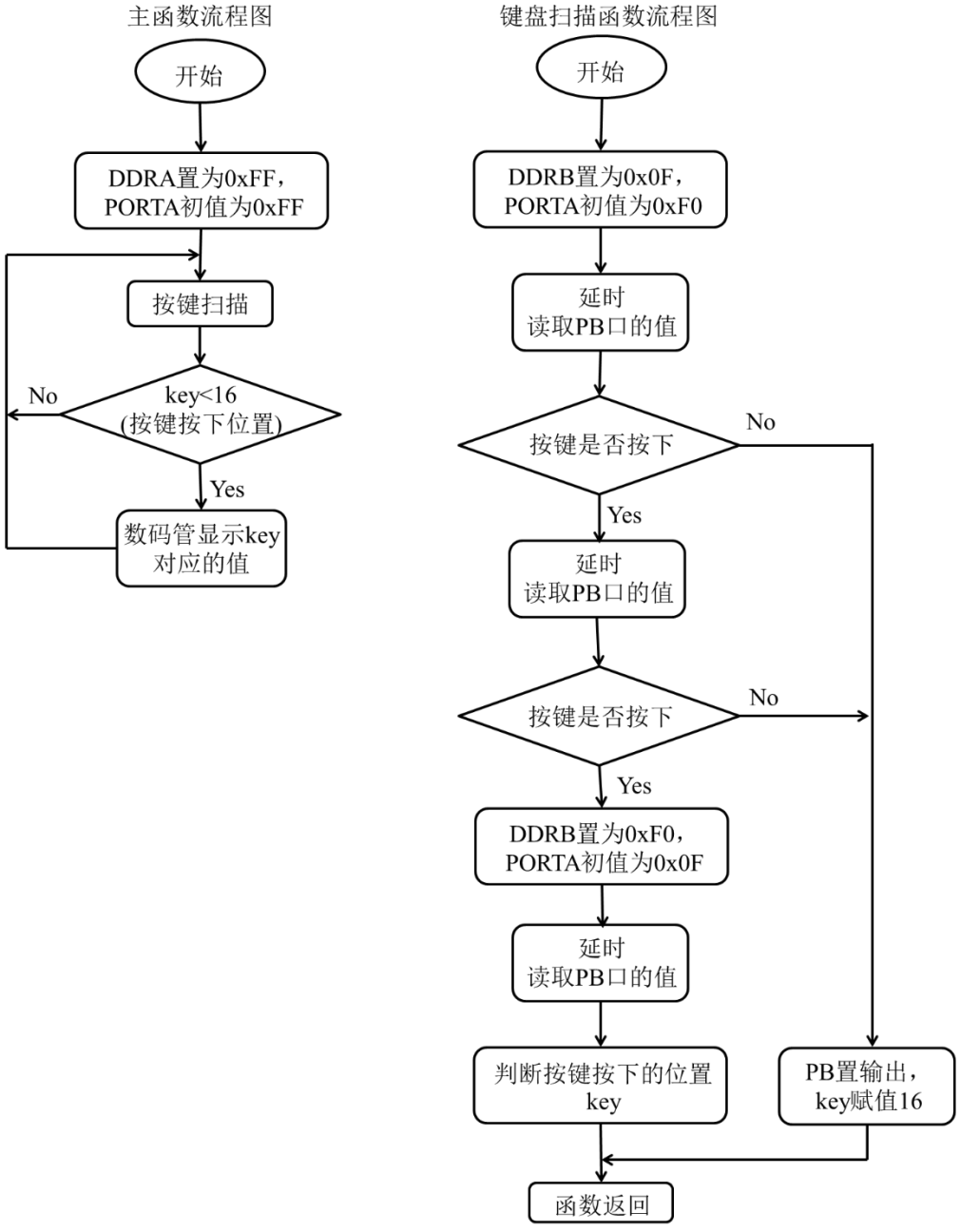
- **硬件电路**



➢ 元件清单

单片机 ATmega16	电阻 330 欧姆*8	带小数点 7 段 数码管
按键*16		

➢ 软件流程



3. 程序代码：

- **ICC 程序**

```
#include <iom16v.h>

#define uchar unsigned char
#define uint unsigned int

uchar const table[17] = {0xc0, 0xf9, 0xa4, 0xb0, 0x99, 0x92, 0x82, 0xf8
, 0x80, 0x90, 0x88, 0x83, 0xc6, 0xa1, 0x86, 0x8e, 0xff};
//数码管编码组
uchar const key_code[] = {0x77, 0xb7, 0xd7, 0xe7, 0x7b, 0xbb, 0xdb, 0xe
b, 0x7d, 0xbd, 0xdd, 0xeed, 0x7e, 0xbe, 0xde, 0xee, 0xff};
//键编码数组
uchar key; //键值

/*N*ms 延时函数*/
void delayms(uint n)
{
  uint i = 0, j;
  for (i = 0; i < n; i++)
    for (j = 0; j < 250; j++)
      ;
}

/*y*us 延时函数*/
void delayus(uint y)
{
  delayms(2 * y);
}

/*键盘扫描子函数*/
uchar keyscan(void)
{
  uchar scan1, scan2, keycode, j;
  DDRB = 0x0f; //高四位输入，低四位输出
  PORTB = 0xf0;
  delayus(2);
  scan1 = PINB; //读 PB 口
  if (scan1 != 0xf0)
  {
    delayms(10);
```

```
    scan1 = PINB;
    if (scan1 != 0xf0)
    {
      DDRB = 0xf0;
      PORTB = 0x0f;
      delayus(2);
      scan2 = PINB;
      keycode = scan1 | scan2;
      for (j = 0; j < 16; j++)
      {
        if (keycode == key_code[j])
        {
          key = j;
          return (key);
        }
      }
    }
  }
  else
    PORTB = 0xff;
  return (key = 16);

}

/*主函数*/
void main(void)
{
  DDRA = 0Xff;
  PORTA = 0xff;
  while (1)
  {
    keyscan();
    if (key < 16)
    {
      PORTA = table[key];
    }
  }
}
```

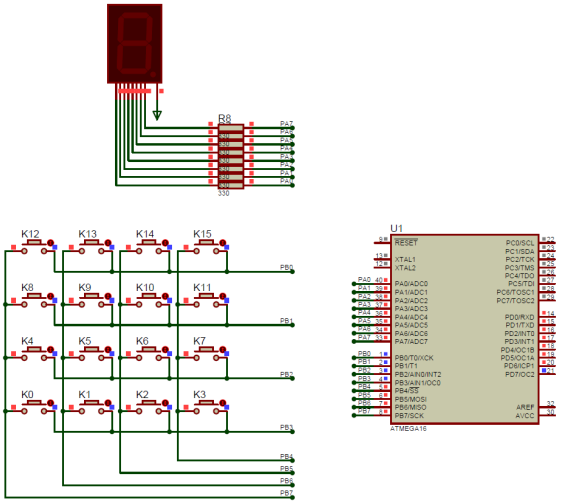
➢ CVAVR 程序

```
#include <mega16.h>

#define uchar unsigned char
#define uint unsigned int

uchar const table[17] = {0xc0, 0xf9, 0xa4, 0xb0, 0x99, 0x92, 0x82, 0xf8
, 0x80, 0x90, 0x88, 0x83, 0xc6, 0xa1, 0x86, 0x8e, 0xff};
//数码管编码组
uchar const key_code[] = {0x77, 0xb7, 0xd7, 0xe7, 0x7b, 0xbb, 0xdb, 0xe
b, 0x7d, 0xbd, 0xdd, 0xeed, 0x7e, 0xbe, 0xde, 0xee, 0xff};
//键编码数组
uchar key; //键值

/*N*ms 延时函数*/
void delayms(uint n)
{
  uint i = 0, j;
  for (i = 0; i < n; i++)
    for (j = 0; j < 250; j++)
      ;
}

/*y*us 延时函数*/
void delayus(uint y)
{
  delayms(2 * y);
}

/*键盘扫描子函数*/
uchar keyscan(void)
{
  uchar scan1, scan2, keycode, j;
  DDRB = 0x0f; //高四位输入，低四位输出
  PORTB = 0xf0;
  delayus(2);
  scan1 = PINB; //读 PB 口
  if (scan1 != 0xf0)
  {
    delayms(10);
    scan1 = PINB;
    if (scan1 != 0xf0)
    {
```

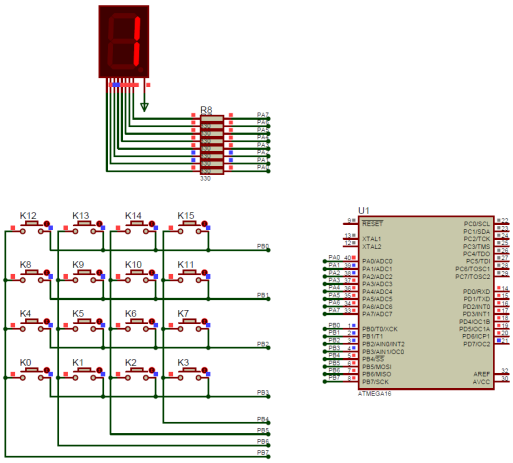
```
      DDRB = 0xf0;
      PORTB = 0x0f;
      delayus(2);
      scan2 = PINB;
      keycode = scan1 | scan2;
      for (j = 0; j < 16; j++)
      {
        if (keycode == key_code[j])
        {
          key = j;
          return (key);
        }
      }
    }
  }
  else
    PORTB = 0xff;
  return (key = 16);
}

/*主函数*/
void main(void)
{
  DDRA = 0Xff;
  PORTA = 0xff;
  while (1)
  {
    keyscan();
    if (key < 16)
    {
      PORTA = table[key];
    }
  }
}
```

4. 仿真结果：

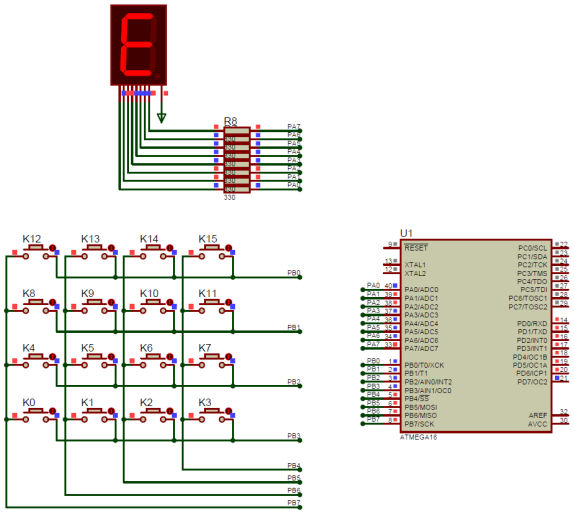
初始状态数码管不显示。



按下 K1 后显示数字“1”。



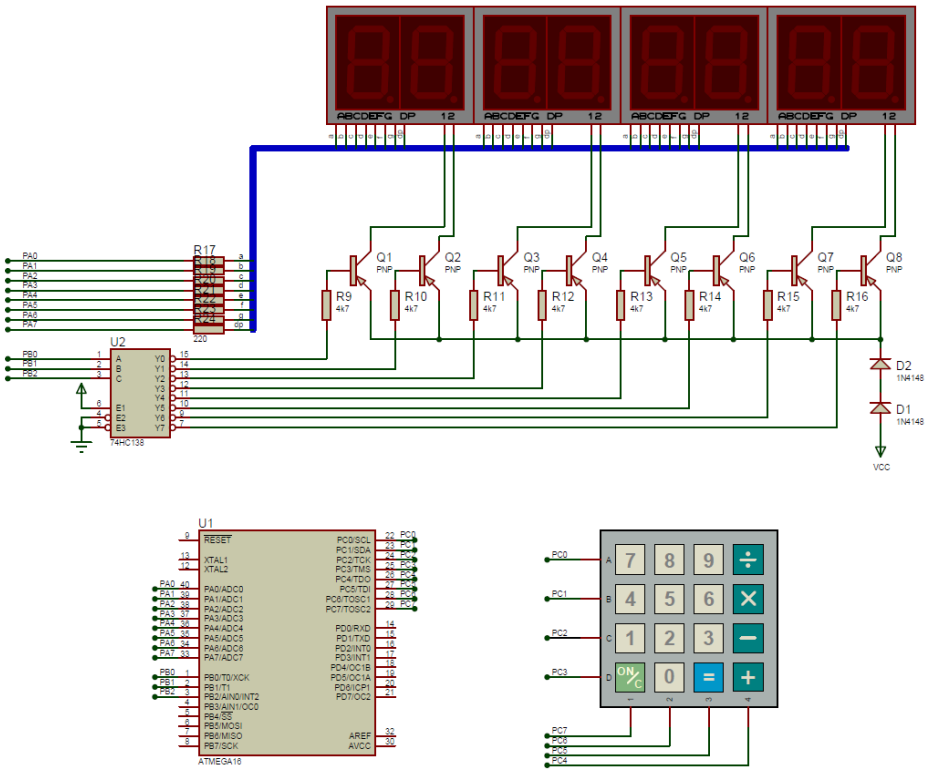
按下 K14 后显示“E”。



5. 改进：

Proteus 有 smallcalc 键盘，再结合前面实验的动态数码管实现一个简易计算器，可以进行结果在-9999999 到 9999999 内的整数加减乘除计算。

- **原理图：**



- **main.c：**

```
#include <mega16.h>
#include "calculator.h"

void main(void)
{
  CalcInit();

  CalcWork();

  while (1)
    ;
}
```

- **calculator.h：**

```
#ifndef __CALCULATOR_H
#define __CALCULATOR_H

#include <mega16.h>
#include "mydisplay.h"

/* 简易计算器 */

void CalcInit(void); //初始化

void CalcWork(void); //简易计算器开始运行

#endif
```

- **calculator.c：**

```
#include "calculator.h"

#define uchar unsigned char
#define uint unsigned int

void CalcInit(void)
{
  DisplayInit();
}

void DelayMs(uint n) //简单延时函数
{
  uint i = 0, j;
  for (i = 0; i < n; i++)
    for (j = 0; j < 250; j++)
      ;
}

uchar const keyCode[] = {0x77, 0xb7, 0xd7, 0xe7, 0x7b, 0xbb, 0xdb, 0xeb
, 0x7d, 0xbd, 0xdd, 0xeed, 0x7e, 0xbe, 0xde, 0xee, 0xff}; //按钮编码数组
uchar key;
                                                          //键值

uchar ButtonScan(void) //按钮扫描
{
  uchar scan1, scan2, code, j;
  DDRC = 0x0f; //高四位输入，低四位输出
```

```
  PORTC = 0xf0;
  DelayMs(2);
  scan1 = PINC; //读 PB 口
  if (scan1 != 0xf0)
  {
    DelayMs(10);
    scan1 = PINC;
    if (scan1 != 0xf0)
    {
      DDRC = 0x00;
      PORTC = 0x00;
      DDRC = 0xf0;
      PORTC = 0x0f;
      DelayMs(2);
      scan2 = PINC;
      while (PINC != 0x0f)
        ;
      code = scan1 | scan2;
      for (j = 0; j < 16; j++)
      {
        if (code == keyCode[j])
        {
          key = j;
          return (key);
        }
      }
    }
  }
  else
    //PORTC = 0xff;
    return (key = 16);
}

long preNumber = 0;
long curNumber = 0;
char operator= 0;

char CheckNumber(long number)
{
  if ((number / 10000000) != 0)
    return 0;
  else
    return 1;
}
```

```
void Calculate(void)
{
  long tempNumber = 0;
  switch (key)
  {
  case 0:
    preNumber = 0;
    curNumber = 0;
    break;
  case 1:
    tempNumber = curNumber * 10;
    if (CheckNumber(tempNumber))
      curNumber = tempNumber;
    break;
  case 2:
    switch (operator)
    {
    case 0:
      tempNumber = curNumber + preNumber;
      break;

    case 1:
      tempNumber = preNumber - curNumber;
      break;
    case 2:
      tempNumber = curNumber * preNumber;
      break;
    case 3:
      tempNumber = preNumber / curNumber;
      break;
    }
    if (CheckNumber(tempNumber))
      curNumber = tempNumber;
    break;
  case 3:
    operator= 0;
    preNumber = curNumber;
    curNumber = 0;
    break;

   case 4:
     tempNumber = curNumber * 10 + 1;
     if (CheckNumber(tempNumber))
       curNumber = tempNumber;
     break;
   case 5:
```

```
    tempNumber = curNumber * 10 + 2;
    if (CheckNumber(tempNumber))
      curNumber = tempNumber;
    break;
  case 6:
    tempNumber = curNumber * 10 + 3;
    if (CheckNumber(tempNumber))
      curNumber = tempNumber;
    break;
  case 7:
    operator= 1;
    preNumber = curNumber;
    curNumber = 0;
    break;
  case 8:
    tempNumber = curNumber * 10 + 4;
    if (CheckNumber(tempNumber))
      curNumber = tempNumber;
    break;

  case 9:
    tempNumber = curNumber * 10 + 5;
    if (CheckNumber(tempNumber))
      curNumber = tempNumber;
    break;
  case 10:
    tempNumber = curNumber * 10 + 6;
    if (CheckNumber(tempNumber))
      curNumber = tempNumber;
    break;
  case 11:
    operator= 2;
    preNumber = curNumber;
    curNumber = 0;
    break;
  case 12:
    tempNumber = curNumber * 10 + 7;
    if (CheckNumber(tempNumber))
      curNumber = tempNumber;
    break;

  case 13:
    tempNumber = curNumber * 10 + 8;
    if (CheckNumber(tempNumber))
      curNumber = tempNumber;
    break;
```

```
  case 14:
    tempNumber = curNumber * 10 + 9;
    if (CheckNumber(tempNumber))
      curNumber = tempNumber;
    break;
  case 15:
    operator= 3;
    preNumber = curNumber;
    curNumber = 0;
    break;
  }
}

int displayNumber = 0;

void DisplayRefresh(void)
{
  if (displayNumber != curNumber)
  {
    char i = 0;
    long tempNumber = curNumber;
    if (tempNumber > 0)
    {
      for (i = 0; i < 8; i++)
      {
        if ((tempNumber / 10 == 0) && (tempNumber % 10 == 0))
          DisplayChange(7 - i, 12);
        else
          DisplayChange(7 - i, tempNumber % 10);
        tempNumber /= 10;
      }
    }
    else if (tempNumber == 0)
    {
      DisplayChange(7, 0);
      for (i = 1; i < 8; i++)
      {
        DisplayChange(7 - i, 12);
      }
    }
    else
    {
      char flag = 0;
      tempNumber = -tempNumber;
```

```
      for (i = 0; i < 8; i++)
      {
        if (flag)
          DisplayChange(7 - i, 12);
        if ((tempNumber / 10 == 0) && (tempNumber % 10 == 0))
        {
          if ((flag != 1))
          {
            flag = 1;
            DisplayChange(7 - i, 11);
          }
        }
        else
          DisplayChange(7 - i, tempNumber % 10);
        tempNumber /= 10;
      }
    }
    displayNumber = curNumber;
  }
}

void CalcWork(void)
{
  while (1)
  {
    ButtonScan();
    if (key < 16)
    {
      Calculate();
      DisplayRefresh();
    }
  }
}
```

- **display.h：**

```
#ifndef __MYDISPLAY_H
#define __MYDISPLAY_H

#include <mega16.h>

/* 数码管显示（使用定时器定时扫描） */

void DisplayInit(void); //初始化
```

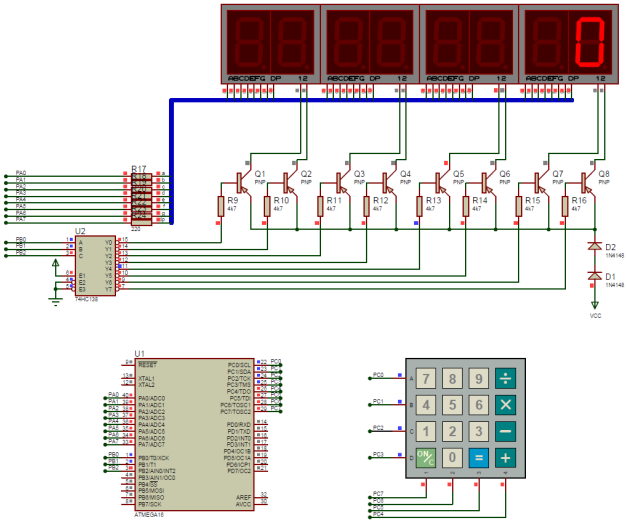
```
void DisplayChange(unsigned char segBit, unsigned char number); //更改第
bit 位数码管显示的数字

#endif
```

- **display.c：**

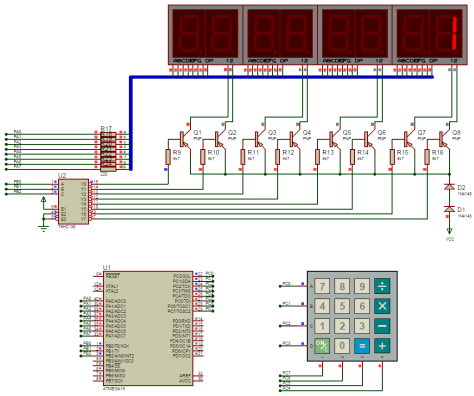
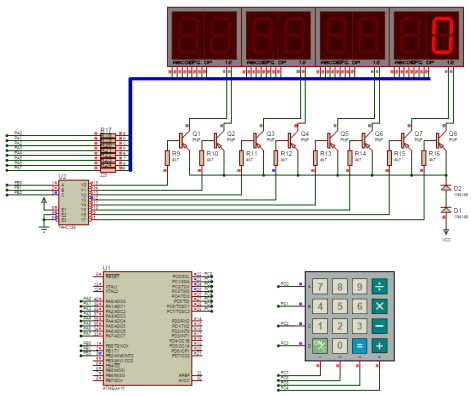
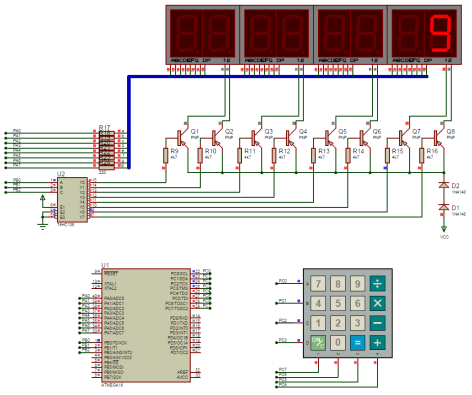
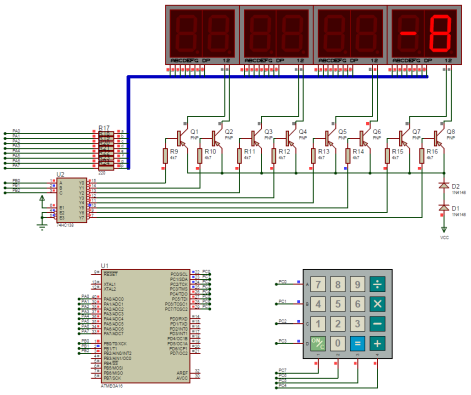
```
#include "mydisplay.h"

//数码管段码
flash unsigned char disCode[] = {0xc0, 0xf9, 0xa4, 0xb0, 0x99, 0x92, 0x
82, 0xf8, 0x80, 0x90, 0xf7, 0xbf, 0xff};

unsigned char segData[8] = {12, 12, 12, 12, 12, 12, 12, 12};

void DisplayInit(void)
{
  DDRA = 0xff;
  PORTA = 0xff;
  DDRB |= 0x07;
  PORTB &= 0xf8;

  TCCR2 = 0x42;  //定时计数器 2，CTC 模式，8 分频
  OCR2 = 124;    //1M/8/125 = 1k
#asm("sei");     //开启全局中断允许
  TCNT2 = 0x00;  //从 0 开始计数
  TIMSK |= 0x80; //允许比较中断
}

void DisplayChange(unsigned char segBit, unsigned char number)
{
  segData[segBit] = number;
}

unsigned char segIndex = 0;
interrupt[TIM2_COMP] void Timer2CompInt(void) //定时器 2 中断函数
{
  if (segIndex > 7)
    segIndex = 0;
  PORTA = 0xff;
  PORTB = (PORTB & 0xf8) | segIndex;
  PORTA = disCode[segData[segIndex]];
  segIndex++;
  TCNT2 = 0x00; //计数器清零
}
```

➢ 仿真结果

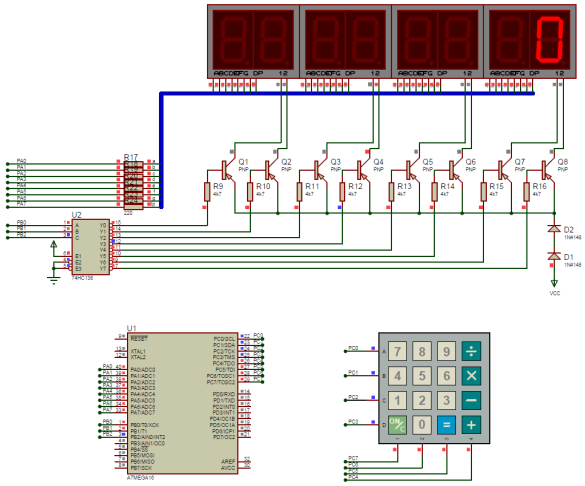
初始状态



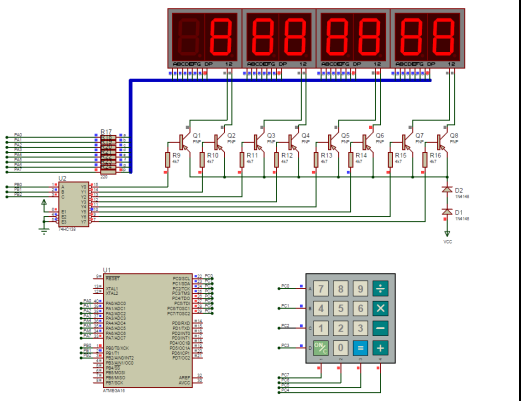
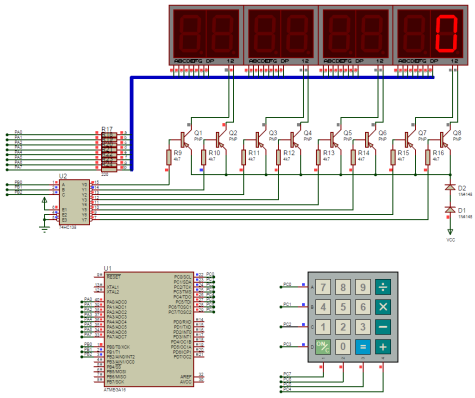
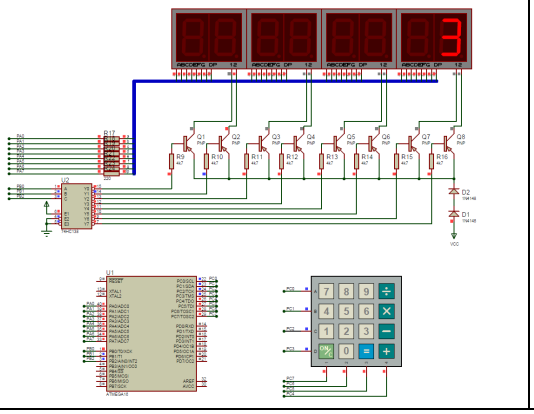
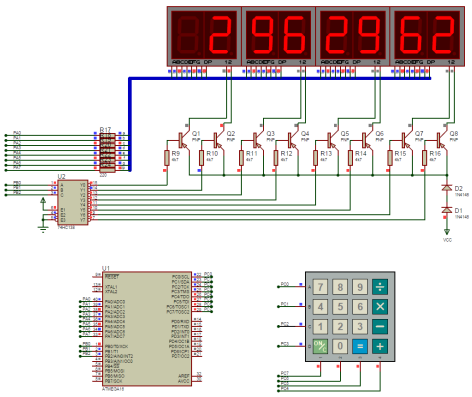
计算减法

输入 1	输入减号
	
输入 9	**输入等于号，得出结果**
	

归零



计算除法

<table>
<tr><td>输入 8888888</td><td>输入除号</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>输入 3</td><td>输入等于号，得出结果</td></tr>
<tr><td>  </td><td>  </td></tr>
</table>

附录：

main.c calculator.h calculator.c mydisplay.h mydisplay.c