MSP430程序库<九>数码管显示

数码管也是单片机系统最常用的输出设备之一(还有液晶、发光二极管等)。七段(这里用的是8段，有小数点)数码管可以完成显示0-9数字和一部分的英文字符如：A、b。本文实现的程序完成显示数字和可显示的英文字符；同时完成数码管显示的 printf 函数的移植，以支持 printf 的格式化字符等好用的特点（我用的数码管8个排为一排，方便数字等的显示）。

1.硬件介绍：

这里所用到的硬件资源包括8个数码管、和 msp430单片机的两个8位 IO 口(这里用的是P3和 P5口，如有改变，可以通过宏定义更改)。

数码管是8个共阴的数码管，a-h 8段通过一个200Ω的电阻接到430单片机的 P5口。共阴端是由单片机的 P3口控制，单片机的一位 IO 通过一个三极管接到数码管的共阴端，以完成位选。

单片机的 P3口时数码管的位选口，某位为高则选中；P5口时段选口；要数码管显示时，通过 P3位选，选中某个数码管亮，P5段选选择8段（a-h）中的那些亮，从而控制某一位显示数字或字符。

要同时显示多个数码管，就要动态扫描；动态扫描时，本程序选用的是由看门狗的中断扫描显示：每1.9ms 显示其中的一位，动态扫描显示每一位，从而让数码管看起来是同时亮的。

2.程序实现：

数码管显示首先要有一个数码管显示的断码表（完成数字和字符到数码管段值的表），程序中采用了《MSP430系列单片机系统工程设计与实践》这本书推荐的方式实现的这个数码表：先用宏定义定义每段对应的单片机要输出的段值，然后再实现是个表，当硬件改变时，只需更改前面的每段的段值定义即可，改动的地方少了很多，代码如下：

```
/*宏定义，数码管 a-h 各段对应的比特，更换硬件只用改动以下8行*/
#definea        0x01                                    //   AAAA
#defineb        0x02                                    // F      B
#definec        0x04                                    // F      B
#defined        0x08                                    //   GGGG
#definee        0x10                                    // E      C
#definef        0x20                                    // E      C
#defineg        0x40                                    //   DDDD   HH
#defineh        0x80          //小数点

/*用宏定义自动生成段码表，很好的写法，值得学习*/
/*更换硬件无需重写段码表*/
    const charTab[] = {
     a + b + c + d + e + f,                  // Displays "0"
    b + c,                                  // Displays "1"
    a + b + d + e + g,                      // Displays "2"
    a + b + c + d + g,                      // Displays "3"
    b + c + f + g,                          // Displays "4"
    a + c + d + f +g,                       // Displays "5"
    a + c + d + e + f + g,                  // Displays "6"
```

```
    a + b + c,                          // Displays "7"
    a + b + c + d + e + f + g,          // Displays "8"
    a + b + c + d + f + g,              // Displays "9"
    a + b + c + e + f + g,              // Displays "A"
    c + d + e + f + g,                  // Displays "B"
    a + d + e + f,                      // Displays "C"
    b + c + d + e + g,                  // Displays "D"
    a + d + e + f + g,                  // Displays "E"
    a + e + f + g,                      // Displays "F"
    a + c + d + e + f,                  // Displays "G"
    b + c + e + f + g,                  // Displays "H"
    e + f,                              // Displays "I"
    b + c + d + e,                      // Displays "J"
    b + d + e + f + g,                  // Displays "K"
    d + e + f,                          // Displays "L"
    a + c + e + g,                      // Displays "M"
    a + b + c + e + f,                  // Displays "N"
    c + e + g,                          // Displays "n"
    c + d + e + g,                      // Displays "o"
    a + b + c + d + e + f,              // Displays "O"
    a + b + e + f + g,                  // Displays "P"
    a + b + c + f + g,                  // Displays "Q"
    e + g,                              // Displays "r"
    a + c + d + f +g,                   // Displays "S"
    d + e + f + g,                      // Displays "t"
    a + e + f ,                         // Displays "T"
    b + c + d + e + f,                  // Displays "U"
    c + d + e,                          // Displays "v"
    b + d + f + g,                      // Displays "W"
    b + c + d + f + g,                  // Displays "Y"
    a + b + d + e + g,                  // Displays "Z"
    g,                                  // Displays "-"
    h,                                  // Displays "."
    0                                   // Displays " "
};
#undefa
#undefb
#undefc
#undefd
#undefe
#undeff
#undefg
```

0-9的位置对应显示0-9，之后的是 A 开始往后显示，为了方便访问这个表格，定义了 AA 等一系列的常量，方便访问这个表。

```
#defineAA   10
#defineBB   AA+1
#defineCC   BB+1
#defineDD   CC+1
#defineEE   DD+1
#defineFF   EE+1
#defineGG   FF+1
#defineHH   GG+1
#defineII   HH+1
#defineJJ   II+1
#defineKK   JJ+1
#defineLL   KK+1
#definemm   LL+1
#defineNN   mm+1
#definenn   NN+1
#defineoo   nn+1
#defineOO   oo+1
#definePP   OO+1
#defineQQ   PP+1
#definerr   QQ+1
#defineSS   rr+1
#definett   SS+1
#defineTT   tt+1
#defineUU   TT+1
#defineVV   UU+1
#defineWW   VV+1
#defineYY   WW+1
#defineZZ   YY+1
#defineNEG ZZ+1     /*   -  */  //负号
#defineDOT NEG+1    /*   .  */  //小数点
#defineSP   DOT+1   /* 空白 */  //空格
```

A 从10开始访问这个表格，如果要显示 A 只需这样用 Tab[AA]，即可得到需要的段值，AA-空格的宏定义放在 H 文件里，方便其他文件访问（当要调用显示函数的时候需要 AA 等宏定义）。为什么是 AA 而不是 A 呢？主要原因是单字母的有几个已经在单片机430的头文件里定义了，为了访问的时候一致，就都用两个字母的了。

为了动态扫描，这里定义了一个全局数组(数码管的程序可以访问)Nixie[8]在这个里面的8个 char 对应8个数码管要显示的段值。初始值是8个数码管都不显示：

```
charNixie[8] = "\0\0\0\0\0\0\0\0";      //初始状态  不显示
```

动态扫描时，函数每1.9ms(设的看门狗定时中断)调用一次显示函数，每次显示一位(为了让中断占用更少的时间，这样中断里只需赋值即可)。函数如下：

```
voidDisplay()
```

```
{
 static chari = 0;        //记录扫描显示到哪位
   CTRL_OUT = 1<<i;
    DATA_OUT = Nixie[i];
    i++;
    if(i>7)
        i = 0;
}
```

这个函数供中断调用，i 用来保存要显示哪一位。CTRL_OUT 、DATA_OUT 是宏定义的位选和段选口。中断程序如下：

```
#pragmavector=WDT_VECTOR
__interrupt voidWDT_ISR()
{
    Display();
}
```

中断只调用了一个函数，这样很方便换其他中断来定时。

中断是必须初始设置的，还有 IO 口，要设为输出方向 ，初始化函数完成数码管用到的单片机资源的初始工作：

```
voidNixiettubeInit()
{
 WDTCTL = WDT_ADLY_1_9;  //看门狗内部定时器模式16ms
IE1 |= WDTIE;             //允许看门狗中断
   CTRL_DIR_OUT;
    DATA_DIR_OUT;
}
```

首先，设置中断并允许中断；然后设置位选和段选所用的端口为输出方向。CTRL_DIR_OUT; DATA_DIR_OUT; 和刚才用到的两个 OUT 的宏定义如下：

```
#defineDATA_DIR_OUT     P5DIR|=0XFF
#defineCTRL_DIR_OUT     P3DIR|=0XFF
#defineDATA_OUT         P5OUT
#defineCTRL_OUT         P3OUT
```

这样处理之后，要显示数字就很简单了:只需把要显示的数字或字符的段码值放入 Nixie[8] 数组对应的位置即可，如显示韩输入下：

```
voidNixiettubeDisplayChar(charch,charaddr)
{
 if(ch == DOT)        //小数点,不需单独占一位
   {
        Nixie[addr] |= Tab[ch];
    }
 else
   {
        Nixie[addr] = Tab[ch];
```

```
    }
}
```

如果是小数点，放入对应位置的 h 段即可，其他直接覆盖。

插入字符函数：在最右端插入数字或字符.

```
voidNixiettubeInsertChar(charch)
{
 if(ch == DOT)          ////小数点,不需单独占一位
   {
        Nixie[0] |= Tab[ch];
        return;
    }
    for(inti = 7;i > 0;i--)
      Nixie[i] = Nixie[i - 1];       //已显示字符左移一位
   Nixie[0] = Tab[ch];
}
```

这个也是先判断小数点，小数点直接放到 h 段，其他的，则要已显示的左移再覆盖最右一位，源程序的注释很详细，可具体才、可以下载附件的程序库。

数码管清除函数，这个函数把数码管全部显示去掉，即把缓存数组内每项都置为0：

```
voidNixiettubeClear()
{
    for(inti = 0;i < 8;i++)
      Nixie[i] = Tab[SP];       //显示空格
}
```

程序比较简单，这里就不多解释了。

数码管的程序就这么多了，所有函数都列出来了。下面开始介绍 printf 的移植，具体过程不再详细说了，详细过程参考：MSP430程序库<四>printf 和 scanf 函数移植。这里主要介绍所需程序。

单片机 printf 使用需要用户提供底层驱动-putchar 函数，printf 完成格式化等一系列活动后调用 putchar 输出字符流。只要实现 putchar，包含 stdio.h 文件，就可以使用 printf 函数。移植的数码管的 putchar 函数如下：

```
#include<stdio.h>
#include"ctype.h"        /*isdigit 函数需要该头文件*/
#include"Nixietube.h"

intputchar(intch)
{
 //'\f'表示走纸翻页，相当于清除显示
   if(ch=='\n'||ch=='\r')
        NixiettubeClear();

 //数字和对应 ASCII 字母之间差0x30    '1'=0x31 '2'=0x32...
 //isdigit 也是 C 语言标准函数
```

```
    if(isdigit(ch))
      NixiettubeInsertChar(ch-0x30);  //若字符是数字则显示数字
  else              //否则，不是数字，是字母
      {
      switch(ch)     //根据字母选择程序分支
        {
          case'A': case'a': NixiettubeInsertChar(AA);break; //字符 A
         case'B': case'b': NixiettubeInsertChar(BB);break; //字符 B
         case'C': case'c': NixiettubeInsertChar(CC);break; //...
            case'D': case'd': NixiettubeInsertChar(DD);break;
             case'E': case'e': NixiettubeInsertChar(EE);break;
             case'F': case'f': NixiettubeInsertChar(FF);break;
             case'G': case'g': NixiettubeInsertChar(GG);break;
             case'H': case'h': NixiettubeInsertChar(HH);break;
             case'I': case'i': NixiettubeInsertChar(II);break;
             case'J': case'j': NixiettubeInsertChar(JJ);break;
             case'K': case'k': NixiettubeInsertChar(KK);break;
             case'L': case'l': NixiettubeInsertChar(LL);break;
             case'M': case'm': NixiettubeInsertChar(mm);break;
             case'N':          NixiettubeInsertChar(NN);break;
             case'n':          NixiettubeInsertChar(nn);break;
             case'O':          NixiettubeInsertChar(OO);break;
             case'o':          NixiettubeInsertChar(oo);break;
             case'P': case'p': NixiettubeInsertChar(PP);break;
             case'Q': case'q': NixiettubeInsertChar(QQ);break;
             case'R': case'r': NixiettubeInsertChar(rr);break;
             case'S': case's': NixiettubeInsertChar(SS);break;
             case'T': case't': NixiettubeInsertChar(tt);break;
             case'U': case'v': NixiettubeInsertChar(UU);break;
             case'V': case'u': NixiettubeInsertChar(VV);break;
             case'W': case'w': NixiettubeInsertChar(WW);break;
          case'Y': case'y': NixiettubeInsertChar(YY);break; //...
         case'Z': case'z': NixiettubeInsertChar(ZZ);break; //字符 Z
         case'-':          NixiettubeInsertChar(NEG);break;//字符-
         case'.':          NixiettubeInsertChar(DOT);break;//小数点，直接显
示在右下角
         case' ':          NixiettubeInsertChar(SP);break; //空格
         default:           NixiettubeInsertChar(SP);break;//显示不出来的字
母用空格替代
        }
     }
   return(ch);  //返回显示的字符(putchar 函数标准格式要求返回显示字符)
  }
```

头文件必须包含 stdio.h，这样告诉编译器 printf 调用时，用这里的 putchar 函数。然后

判断字符，分类进行显示，不能显示的空一格。

数码管的程序就完成了，如果需要可以自己添加改写函数，如：当和键盘共同使用时，如果键盘移植了 scanf 函数，并且支持退格；可以改写函数-让数码管的 putchar 支持退格操作。或者用的是我的键盘程序，需要10多 ms 调用一次键盘处理函数，这样可以和这个数码管扫描公用一个中断：

```
voidDisplay()
{
 static chari = 0;        //记录扫描显示到哪位
   CTRL_OUT = 1<<i;
    DATA_OUT = Nixie[i];
    i++;
    if(i>7)
    {
        i = 0;
        KeyProcess();
    }
}
```

这样改写，然后把键盘的中断去掉(别忘了 key.h 包含和加入 KeyProcess(); 的声明；如果程序中有两个指向同一个中断时，会编译错误)；这样就可以键盘、和数码管共同使用了。

3.使用示例：

使用方法还是和之前一样，工程中加入 Nixietube.c 文件，然后在要调用的地方加入 Nixietube.h 的包含；如 puchr 函数，和示例工程的 main.c

main.c 调用的方式如下：

```
#include<msp430x16x.h>
#include<stdio.h>
#include"Nixietube.h"

voidClkInit()
{
    chari;
 BCSCTL1 &= ~XT2OFF;                 //打开 XT2振荡器
IFG1&=~OFIFG;                       //清除振荡错误标志
   while((IFG1&OFIFG)!=0)
    {
        for(i=0;i<0xff;i++);
      IFG1&=~OFIFG;                  //清除振荡错误标志
   }
 BCSCTL2 |= SELM_2+SELS+DIVS_3;   //MCLK 为8MHz，SMCLK 为1MHz
}

voidmain( void)
{
```

```
  // Stop watchdog timer to prevent time out reset
    WDTCTL = WDTPW + WDTHOLD;
     ClkInit();
     NixiettubeInit();
     _EINT();
  //while(1)
    {
          NixiettubeDisplayChar(AA,5);
          NixiettubeDisplayChar(DOT,5);
          NixiettubeInsertChar(2);
          NixiettubeInsertChar(DOT);
          NixiettubeInsertChar(2);
          printf("%1.2f",1.2);
     }
}
```

包含 msp430的头文件，以便使用430单片机的先关资源；加入 stdio.h 以使用 printf 函数；加入 Nixietube.h 使用数码管的相关程序。

还要注意，为了数码管正常显示，必须打开总中断，以使数码管动态扫描显示。另外，本程序单步调试看不到数码管正常显示，因为没有扫描。只有全速运行才可以看到数码管的显示情况。

数码管的程序就到这里。

相关文章及附件下载：http://www.ideyi.org/bbs/article_1077_372218.html