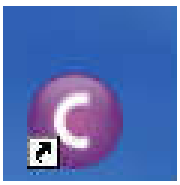
查看: 33505 | 回复: 427

spark51



219 积分 | 23 主题 | 173 帖子

中级会员

发消息

高级流水灯--水滴效果（渐变带拖尾效果）实现和讲解 [复制链接]

发表于 2011-12-6 09:03:44 | 只看该作者 1楼 电梯直达

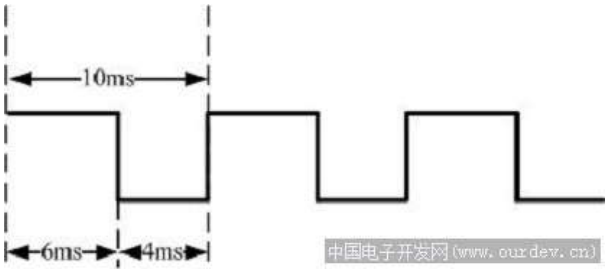
简介

学习嵌入式第一个例子通常都是控制一个LED亮灭，然后是花样繁多的流水灯，但不管灯的花样如何变化，单个LED的亮度没有变化，只有亮、灭两个状态，本章我们实现如何控制LED的亮度。

1 什么是PWM

脉冲宽度调制（Pulse Width Modulation，简称PWM），是利用微处理器的数字输出来对模拟电路进行控制的一种技术。

在本章的应用中可以认为PWM就是一种方波。比如图1：



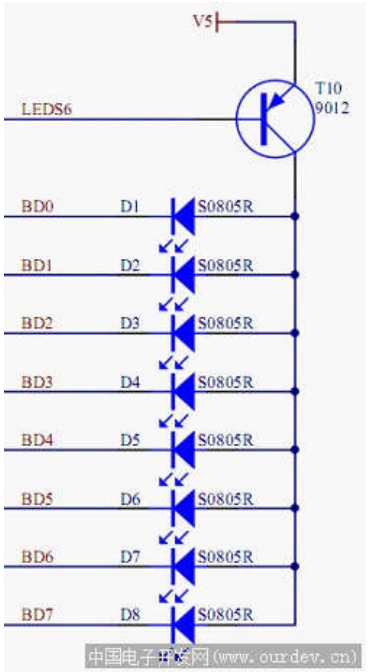
(原文件名:120611_0.png) 图1 方波

是周期为10ms，占空比为60%的PWM。

占空比：高电平在一个周期之内所占的时间比率。

2 硬件设计

在例说51单片机的第三章，我们讲过如何控制开发板上LED的亮灭。首先译码器输出端LEDS6为低，T10导通，给8个LED供电，然后通过缓冲器8个输出端BD0~BD7的控制LED的亮灭（低亮高灭）。



(原文件名:120611_1.png) 图2 LED硬件连接

如果BD口输出高低不断变化，则LED会闪烁；如果这种高低电平变化非常快，由于人的视觉暂留现象，LED就会出现不同的亮度。

3 软件设计

3.1 PWM能否控制亮度

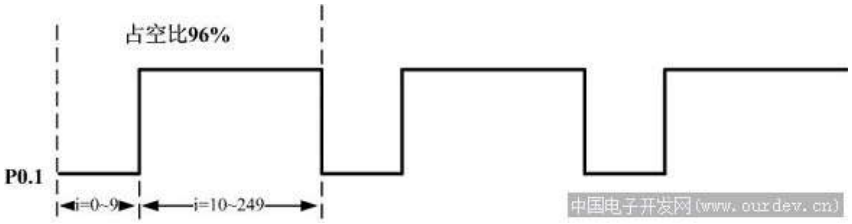
下面我们就用实践验证PWM是否能够控制LED的亮度，测试代码如下：

程序清单L1： 验证PWM能否控制LED的亮度

```
1       #include <reg52.h>
2       #include "my_type.h"
```

```
3     #include "hw_config.h"
4
5
6     void main(void)
7     {
8           u8 i = 0;
9
10           //使能独立LED的供电，即LEDS6输出低电平
11           LEDEN = 0;
12           ADDR0 = 0;
13           ADDR1 = 1;
14           ADDR2 = 1;
15           ADDR3 = 1;
16
17           //第一个LED亮
18           P0 = 0xFE;
19
20           while(1)
21           {
22                 for(i=0; i<250; i++)
23                 {
24                       if(i<10)
25                       {
26                             P0 &= 0xFD;  //第二个灯亮
27                       }
28                       else
29                       {
30                             P0 |= 0x02;  //第二个灯灭
31                       }
32                 }
33           }
34     }
```

L1(22-32)：这段代码实现P0.1输出占空比为96%的方波，而P0.0恒为低。
P0.1输出如图3所示（受纸张限制，图中高低电平长度比例和实际有偏差）。



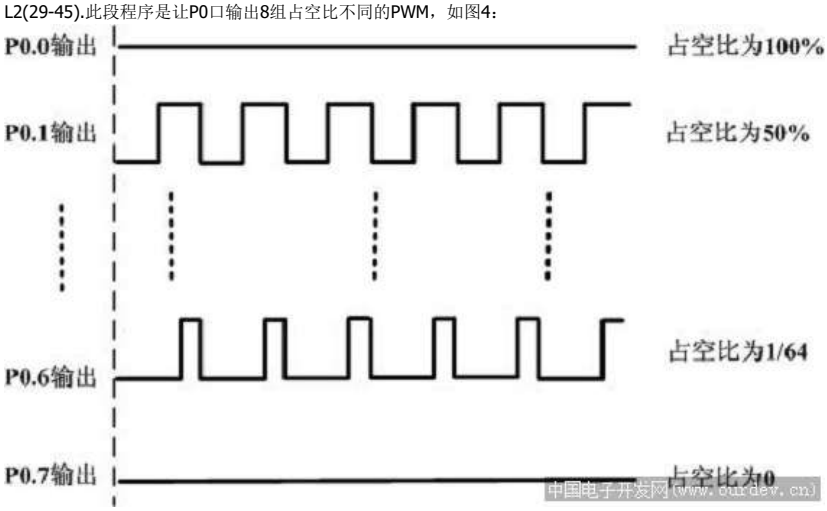
(原文件名:120611_2.png) 图3

下载验证：从开发板上可以看到运行效果，D1比D2亮。（这里说明一点：当P0输出低电平时，LED亮，所以，PWM的占空比越小越亮）。
3.2 产生8个亮度级别
3.1节的例子证实了我们的设想，PWM可控制LED的亮度，下面我们设计几组占空比不同的PWM，看看对LED亮度的控制效果。代码如下：
程序清单L2：不同占空比对LED亮度的控制

```
1     #include <reg52.h>
2     #include "hw_config.h"
3     #include "my_type.h"
4
5
6     //亮度级别表
7     code u8 LightLevel[8]={0,1,2,4,8,16,32,64};
8
9     void main(void)
10     {
11        u8 i = 0;
12           u8 j = 0;
13           u8 k = 0;
14           u8 temp = 0;
15
16        //使能独立LED的供电，即LEDS6输出低电平
17        LEDEN = 0;
18        ADDR0 = 0;
19        ADDR1 = 1;
20        ADDR2 = 1;
21        ADDR3 = 1;
22
23           //开始全灭
24           P0 = 0xFF;
25
26        while(1)
27        {
28                 //P0端口输出8组占空比不同的PWM
29                 for(i=0; i<64; i++)
30                 {
```

```
31                      for(j=0; j<8; j++)
32                      {
33                          if(LightLevel[j] <= i)
34                          {
35                              temp |= (1<<j);
36                          }
37                          else
38                          {
39                              temp &= ~(1<<j);
40                          }
41                      }
42
43                      P0 = temp;
44                  }
45              }
46      }
```

L2(29-45).此段程序是让P0口输出8组占空比不同的PWM，如图4：



(原文件名:120611_3.png) 图4

下载验证：从开发板上可以看到运行效果，从D1到D8的亮度逐渐增大。

3.3 水滴下落效果

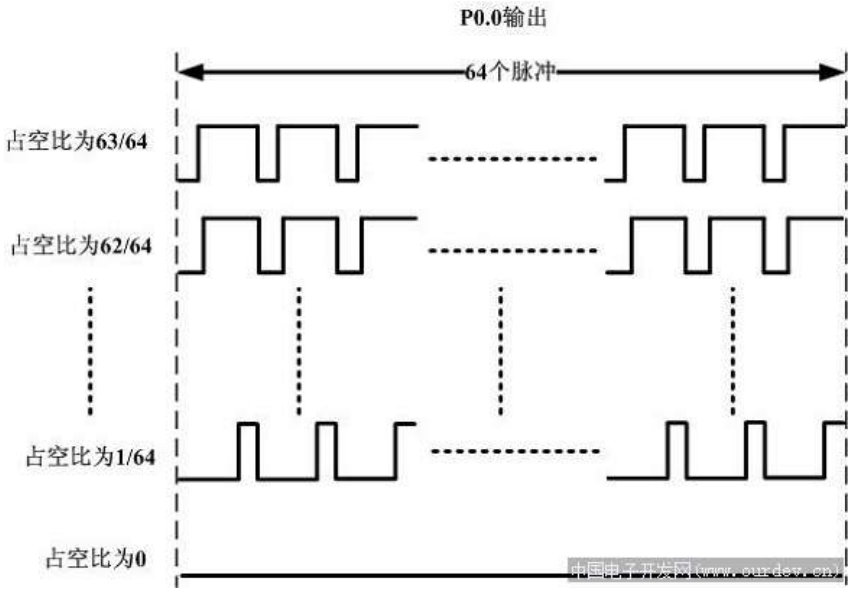
根据PWM可控制LED亮度的原理，我们用8个LED实现水滴下落的效果。第一步，水滴逐渐变大，用D1从暗变亮模拟；第二步，水滴下落，带有拖尾效果，LED逐个亮，移动速度加快，且越靠前的LED亮度越大。

程序清单L3 水滴流水灯

```
1       #include <reg52.h>
2       #include "hw_config.h"
3       #include "my_type.h"
4
5       //亮度级别表
6       code u8 LightLevel[8]={0,1,2,4,8,16,32,64};
7
8       //水滴时间，实现加速效果
9       code u8 LightTime[16]={16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1};
10
11
12       void main(void)
13       {
14          u8 i,j,k;
15          u8 temp,count;
16              u8 state;
17
18          //使能独立LED的供电，即LEDS6输出低电平
19          LEDEN = 0;
20          ADDR0 = 0;
21          ADDR1 = 1;
22          ADDR2 = 1;
23          ADDR3 = 1;
24
25          while(1)
26          {
27                  //开始全灭
28                  P0 = 0xFF;
29
30                  //---------------水滴逐渐变大（第一个LED亮度逐渐变大）---------
31                  for(i=0; i<64; i++)
32                  {
33                      //一个亮度级别发送64个脉冲
34                      for(j=0; j<64; j++)
35                      {
36                          P0 = 0xFE;
```

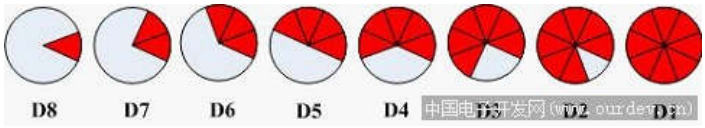
```
37                          //以i为亮度级别，随着i的增大，占空比增大
38                          for(k=0; k<64; k++)
39                          {
40                              if(k > i)
41                              {
42                                  P0 = 0xFF;
43                              }
44                          }
45                      }
46                  }
47
48                  //----------------------水滴降落过程----------------------
49                  for(state=0; state<16; state++)
50                  {
51                      //每一状态维持LightTime[state]个脉冲
52                      for(count=0; count<=LightTime[state]; count++)
53                      {
54                          //temp记录8个LED的状态，0代表亮，1代表灭
55                          temp = 0x00;
56
57                          //一个脉冲长度j从0到63
58                          for(j=0; j<64; j++)
59                          {
60                              //根据亮度表，依次确定8个LED当前状态，亮或灭
61                              for(k=0; k<8; k++)
62                              {
63                                  //以j为亮度级别，每个LED亮度不一样
64                                  if(LightLevel[k] == j)
65                                  {
66                                      temp |= (1 << k);
67                                  }
68                              }
69
70                              if(state <= 7)
71                              {
72                                  P0 = ~((~temp) >> (7-state));
73                              }
74                              else
75                              {
76                                  P0 = ~((~temp) << (state-7));
77                              }
78                          }
79                      }
80                  }
81              }
82      }
```

L2(31-46).实现水滴变大效果，这段代码的作用可用图形表达，如图5：



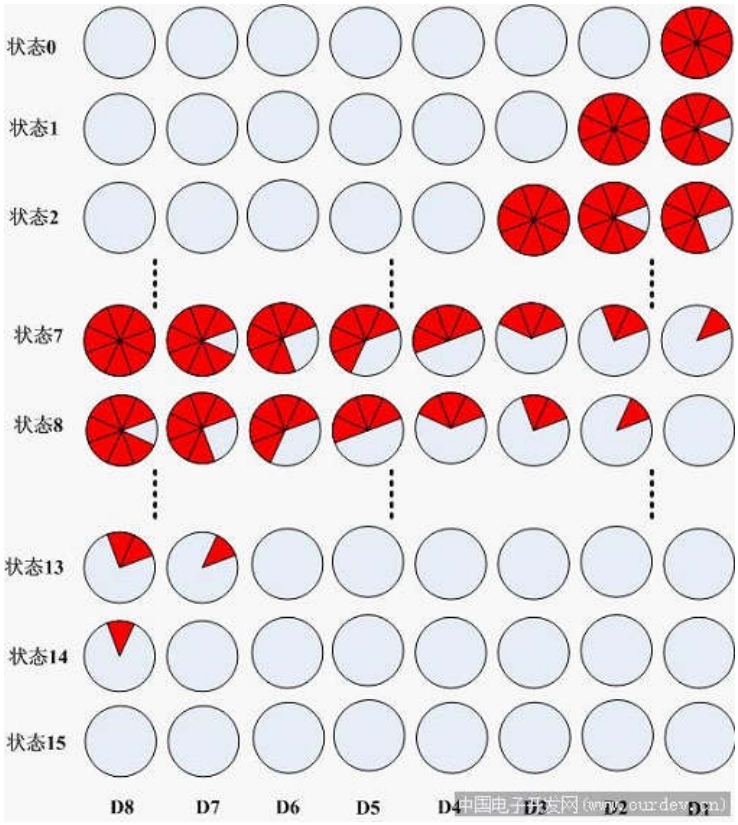
(原文件名:120611_4.png) 图5

控制D1由暗变亮，用了64个亮度级别，每个级别发送64个脉冲。
L2(49-81).实现水滴下落。代码就不逐行解释了，大家可根据注释自己分析，主要说一下实现的方法。
定义LED有8个亮度级别，若用开发板上的8个LED表示，如图6：



(原文件名:120611_5.png) 图6

图中的红色面积代表亮度程度。实现流水效果的方法就是：让所有的亮度依次经过在所有LED，如图7：



(原文件名:120611_6.png) 图7

状态的持续时间从0-15逐渐减小，以模拟水滴加速。
下载验证：下载到开发板上，可以看到水滴下落效果。
3.4 定时器产生PWM
前面3个例子中，我们用循环语句虽然能产生占空比不同的PWM，但PWM的周期不好控制，对此，我们学习如何用定时器产生特定周期PWM。
关于8051定时器的使用方法，大家可以参考例说51单片机的4章和5章。
我们用定时器0产生PWM，代码如下：
程序清单L4 定时器0产生PWM

```
1       #include <reg52.h>
2       #include "hw_config.h"
3       #include "my_type.h"
4
5
6       //亮度级别表
7       code u8 LightLevel[8]={1,2,4,8,16,28,50,64};
8
9       //函数声明
10       void timer0_init(void);
11
12       void main(void)
13       {
14            //使能独立LED的供电，即LEDS6输出低电平
15          LEDEN = 0;
16          ADDR0 = 0;
17          ADDR1 = 1;
18          ADDR2 = 1;
19          ADDR3 = 1;
20
21              timer0_init();
22
23              while(1)
24              {
25              }
26       }
27
28       /*************************************************************
29       函数名称：timer0_init
30       功    能：初始化定时器0
```

```
31      ************************************************************/
32      void timer0_init(void)
33      {
34            TMOD = 0x01;    //运行模式1
35        TH0  = 0xFF;    //10us中断
36        TL0  = 0xFA;
37        EA   = 1;       //开启中断
38        ET0  = 1;
39        TR0  = 1;       //启动定时器
40
41      }
42
43      /************************************************************
44      函数名称：timer0_overflow
45      功    能：定时器0溢出中断
46      ************************************************************/
47      void timer0_overflow(void) interrupt TIMER0_OVERFLOW
48      {
49            u8 i,temp = 0;
50            static u8 count = 0;
51
52            count++;
53            count %= 64;
54
55            for(i=0; i<8; i++)
56            {
57                  if(LightLevel <= count)
58                  {
59                        temp |= (1<<i);
60                  }
61                  else
62                  {
63                        temp &= ~(1<<i);
64                  }
65            }
66
67            P0 = temp;
68
69            TR0 = 0;
70            TH0 = 0xFF;    //重新赋值
71          TL0 = 0xF7;
72            TR0 = 1;
73      }
```

L4(32).初始化定时器0，没10us产生一次中断。

L4(55-65).控制输出8组不同占空比的PWM。这段代码功能和程序清单2中的功 能一致。

下载验证：下载到开发板上，可以看到D1到D8亮度逐渐增大。

3.5 亮度不同的点阵

学习了用定时器产生PWM，我们可以控制更多的LED，比如LED点阵的亮度。下面的例子实现LED点阵每行的亮度都不同。

程序清单5 亮度不同的点阵

```
1       #include <reg52.h>
2       #include "hw_config.h"
3       #include "my_type.h"
4
5
6       //亮度级别表
7       code u8 LightLevel[8]={1,2,4,8,16,32,50,64};
8
9       //函数声明
10       void timer0_init(void);
11
12       void main(void)
13       {
14            //使能控制点阵的译码器
15          LEDEN = 0;
16          ADDR3 = 0;
17
18            timer0_init();
19
20            while(1)
21            {}
22       }
23
24       /************************************************************
25       函数名称：timer0_init
26       功    能：初始化定时器0
27       ************************************************************/
28       void timer0_init(void)
29       {
```

```
30          TMOD = 0x01;    //运行模式1
31        TH0  = 0xFF;    //中断时间10us
32        TL0  = 0xF7
33        EA   = 1;       //开启中断
34        ET0  = 1;
35        TR0  = 1;       //启动定时器
36          }
37
38    /************************************************************************
39    函数名称：timer0_overflow
40    功    能：定时器0溢出中断
41    ************************************************************************/
42    void timer0_overflow(void) interrupt TIMER0_OVERFLOW
43    {
44          u8 i;
45          u8 p1_value = 0;
46          static u8 state = 0;   //点阵状态（扫描行数）
47          static u8 count = 0;
48
49          TR0 = 0;
50
51          count++;
52          if(count == 64)
53          {
54                state++;
55                state %= 8;
56                count = 0;
57          }
58
59          if(count < LightLevel[state])
60          {
61                P0 = 0x00;
62          }
63          else
64          {
65                P0 = 0xFF;
66          }
67
68          p1_value = P1 & 0xf8;
69          p1_value |= state;
70          P1 = p1_value;
71
72          TH0 = 0xFF;    //重新赋值
73        TL0 = 0xFA;
74          TR0 = 1;
75    }
```

L5(28).初始化定时器，每10us中断一次。
L5(51-57).每中断64次，点阵扫描移动到下一行，用state记录当前行数。
L5(59-66).扫描每一行输出的PWM都不一样，使用的方式和处理独立LED一致。
L5(68-70).输出点阵对应的位码。
下载验证：下载到开发板上，可以看到运行效果，点阵第一行最暗，越往下越亮。
3.6 点阵模拟音乐频谱分析效果
在很多音乐播放软件上，都有频谱分析的图形，如图8：

(原文件名:120611_7.png) 图8

我们用也可以模拟相似的图形，代码如下：
程序清单6：点阵模拟音乐频谱分析

```
1     #include <reg52.h>
2     #include "hw_config.h"
3     #include "my_type.h"
4
5     //频谱波形表
6     code u8 Wave[16][8]=
7     {
8           {0xFF,0xFF,0xFF,0xFF,0xFE,0xBB,0xFE,0xAA},
9           {0xFF,0xFF,0xFF,0xFE,0xFB,0xAE,0xFA,0xAA},
10            {0xFF,0xFF,0xFF,0xFE,0xEB,0xBE,0xEA,0xAA},
11            {0xFF,0xFF,0xFE,0xFB,0xAF,0xFE,0xAA,0xAA},
12            {0xFF,0xFE,0xFB,0xBE,0xEA,0xBA,0xAA,0xAA},
13            {0xFF,0xFE,0xBB,0xEE,0xBA,0xBA,0xAA,0xAA},
14            {0xFE,0xBB,0xEE,0xBA,0xAA,0xAA,0xAA,0xAA},
15            {0xBA,0xEF,0xBE,0xAA,0xAA,0xAA,0xAA,0xAA},
16            {0xEE,0xBB,0xFE,0xAA,0xAA,0xAA,0xAA,0xAA},
```

```
17          {0xEE,0xBB,0xFE,0xEA,0xAA,0xAA,0xAA,0xAA},
18          {0xFE,0xEB,0xBE,0xFE,0xAA,0xAA,0xAA,0xAA},
19          {0xFF,0xEE,0xBB,0xFF,0xAE,0xAA,0xAA,0xAA},
20          {0xFF,0xFE,0xAF,0xFB,0xEE,0xAA,0xAA,0xAA},
21          {0xFF,0xFF,0xFE,0xBB,0xEF,0xBA,0xAA,0xAA},
22          {0xFF,0xFF,0xFF,0xFE,0xAB,0xFF,0xEE,0xAA},
23          {0xFF,0xFF,0xFF,0xFF,0xFE,0xEB,0xBE,0xAA}
24      };
25
26      //亮度级别表
27      code u8 LightLevel[8]={1,2,4,8,16,32,50,64};
28
29      //函数声明
30      void timer0_init(void);
31
32      void main(void)
33      {
34          //使能控制点阵的译码器
35        LEDEN = 0;
36        ADDR3 = 0;
37
38          timer0_init();
39
40          while(1)
41          {
42          }
43      }
44
45      /*********************************************************************
46      函数名称：timer0_init
47      功　能：初始化定时器0
48      *********************************************************************/
49      void timer0_init(void)
50      {
51          TMOD = 0x01;    //运行模式1
52        TH0  = 0xFF;    //10us中断
53        TL0  = 0xFA;
54        EA   = 1;       //开启中断
55        ET0  = 1;
56        TR0  = 1;       //启动定时器
57
58      }
59
60      /*********************************************************************
61      函数名称：timer0_overflow
62      功　能：定时器0溢出中断
63      *********************************************************************/
64      void timer0_overflow(void) interrupt TIMER0_OVERFLOW
65      {
66          u8 i;
67          u8 p1_value = 0;
68          static u8 state = 0;        //点阵状态（扫描行数）
69          static u8 count = 0;
70
71          static u8 wave_state = 0;    //波形状态
72          static u16 wave_count = 0;
73
74          TR0 = 0;
75
76          //每中断1000次，改变波形状态
77          wave_count++;
78          if(wave_count == 1000)
79          {
80              wave_count = 0;
81              wave_state++;
82              wave_state %= 16;
83          }
84
85          //每中断64次，改变扫描的行
86          count++;
87          if(count == 64)
88          {
89              state++;
90              state %= 8;
91              count = 0;
92          }
93
94          if(count < LightLevel[state])
```

```
95          {
1               P0 = Wave[wave_state][state];
2           }
3           else
4           {
5               P0 = 0xFF;
6           }
7
8           //输出位码
9           p1_value = P1 & 0xf8;
10           p1_value |= state;
11           P1 = p1_value;
12
13           //定时器重新赋值
14           TH0 = 0xFF;
15         TL0 = 0xF7;
16           TR0 = 1;
17      }
```

L6(6).波形表，共16个状态，每个状态下有8个数据，即一个点阵界面。扫描点　　阵时循环发送，实现动态效果。
L6(77-83).每中断1000次，更换一个状态。
L6(86-101).和L5的功能一致，只是点阵段码输出的数据变成了波形表。

下载验证：下载程序到开发板，可以看到，点阵显示的频谱分析效果。

完整工程外链地址：http://ishare.iask.sina.com.cn/f/21824087.html

工程使用TKStudio，比较简单，很容易转KEILourdev_701989LXF2X2.zip(文件大小:11K) (原文件名:PWM_LED3.zip)

#在这里快速回复#　　快速回复

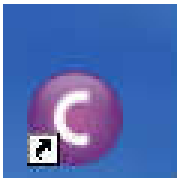
收藏 181

回复　　举报

spark51

楼主 | 发表于 2011-12-6 09:07:00 | 只看该作者　　2楼

代码发的时候有缩进，怎么发出来就没了呢。。。奇怪



219 积分 | 23 主题 | 173 帖子

中级会员

发消息

回复　　举报

yinglively

发表于 2011-12-6 09:11:14 | 只看该作者　　3楼

这东西比较好，LZ威武

929 积分 | 38 主题 | 853 帖子

高级会员

发消息

回复　　举报