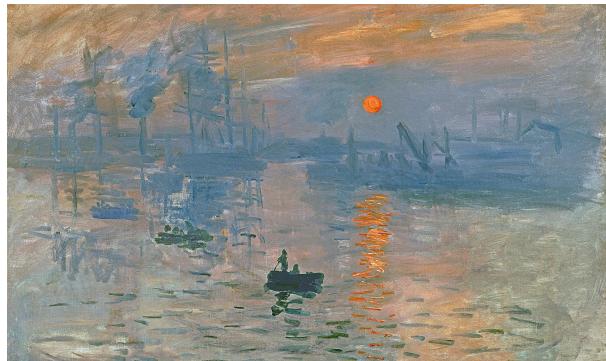


4 | 状态机实验

日出·印象（克劳德·莫奈）

逆境是人类获得知识的最高学府，难题是人们取得智慧之门。



~ · ~

4.1 | 实验目的

本次实验的目的是学习状态机的设计和状态机的 HDL 建模。

4.2 | 基本概念

状态机由状态寄存器和组合逻辑电路构成，能够根据控制信号按照预先设定的状态进行状态转移，是协调相关信号动作、完成特定操作的控制中心。状态机简写为 FSM (Finite State Machine)，主要分为两大类：

第一类，若输出只和状态有关而与输入无关，则称为 Moore 状态机；

第二类，输出不仅和状态有关而且和输入有关系，则称为 Mealy 状态机。

状态机的设计可通过状态转移图、状态转移表、HDL 建模等多种方法实现。

举一个 Mealy 波形图识别 1101 序列的例子：

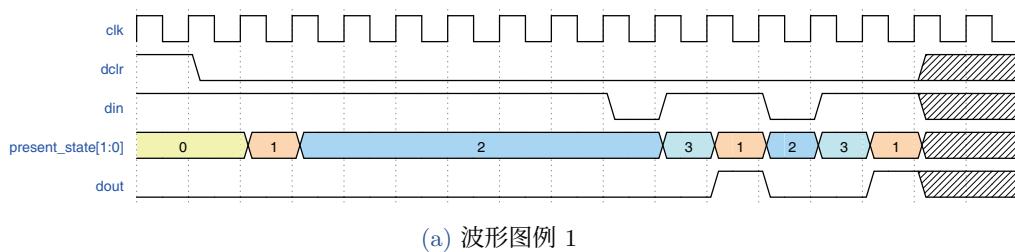


图 4.1: 波形图例

这两段 din 的序列都是 1101，因此都可以被识别，此时 dout 被置为 1。

补充：

例子中的这种叫做循环检测，也就是上一个序列结尾可以作为下一个序列的开端。（是本实验要求的）

而如果是非循环检测，每一个序列不能重复使用，上个 1101 出现后，这 4 位信号被“丢弃”，只有下一个完整的 1101 出现才再次出现高电位输出。（不是本实验要求的）

下图是 Mealy 状态机检测 1101 序列的状态转移图。

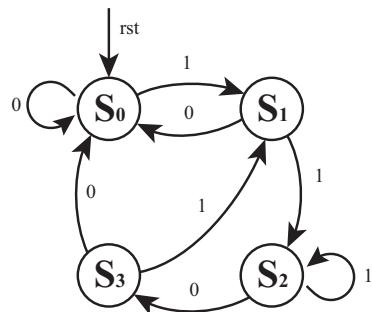


图 4.2: 状态转移图

4.3 | 实验内容

4.3.1 | 计时器实验

请参照附件 PPT 中“反应计时器”的示例，使用状态机实现并上版。具体说明如下：

- 测试人员通过某个按键点亮 LED 灯
- 被测人员看到 LED 灯被点亮后，尽可能快地按下按键
- 测量从 LED 被点亮时刻到开关被按下时刻的时间长度（单位：10 毫秒）

其中，需要两个按键输入、一个复位输入（请注意复位信号是高电平或低电平有效）；一个 LED 灯输出以及两个数码管输出。

注意：具体实现不需要与此说明完全一致。可以发挥自己的想象力，实现更为友好、更为科学的测试方式。例如：按下按键后设置五个 LED 灯间隔一秒依次点亮，再经过一个随机时间后五个 LED 灯同时熄灭。以此时间点为基准，测量到被测人员按下按键时刻的时间长度（单位：10 毫秒），且当此时间长度小于 100 毫秒时认为是无效成绩，计数器显示最大值（如 99），或当 LED 灯没有熄灭时按下，同样显示最大值（99）。

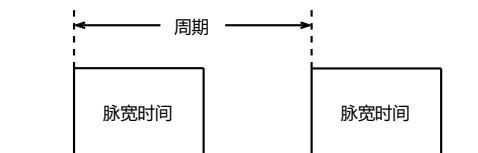
4.3.2 | 呼吸灯的实现

“智能手机的无趣，从失去呼吸灯开始。”

所谓的呼吸灯，就是在控制下，实现 LED 灯由暗渐亮，然后再由亮到暗，周而复始，就像人的呼吸一般。

想要实现呼吸灯的效果，首要的是如何实现 LED 呈现出不同的亮度，但我们知道 FPGA 的引脚电压只有 0 或者 1，显然用“正弦波/余弦波”信号输出是不现实的。我们可以通过改变引脚单位时间内高电平的输出时间来实现。

PWM (Pulse Width Modulation) 脉冲宽度调制，是我们实现呼吸灯的一种途径，接下来，介绍一些有关概念，方便你快速了解 PWM。

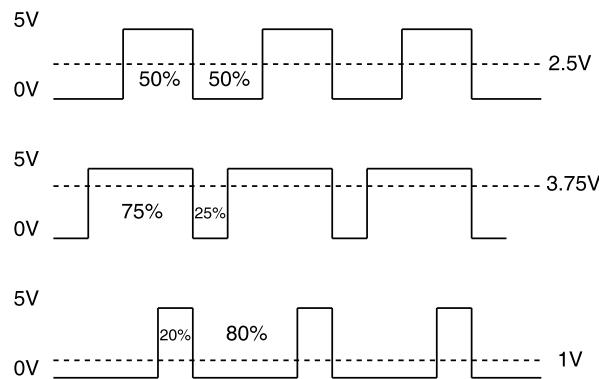


周期：如上图所示

占空比：是一个周期内，高电平时间和整个周期时间的比例。

例：周期若为 10ms，高电平时间为 8ms，占空比则为： $8 / (8+2) = 80\%$

所以，通过对占空比进行调节，即可得到不同大小的输出模拟电压，假设高电平为 5V，那么不同占空比情况下，实际输出的电压如下图所示：



请实现呼吸灯的效果。

4.3.3 | 加分项：拖影流水灯

在4.3.2的基础上，实现带有拖影效果的流水灯。（即当切换 LED 时，上一个被点亮的 LED 不完全熄灭，而是亮度随时间递减。）

4.3.4 | 在七段数码管上滚动显示学号

按照实验步骤E.1完成本实验。以学号 U202140000 为例：

1. 首先将学号中的 10 个数字存储在一个 40 位的寄存器 msgArray 中，每 4 位存放一个十进制数字；
2. 4 个数码管始终显示寄存器的高 16 位数据；
3. 用频率为 3Hz 的时钟控制 7 段数码循环显示：在时钟的上升沿进行向左循环移动 4 位，并显示。注意：记得要把 msgArray 中 msgArray[39:36] 的内容移到 msgArray[3:0] 中。
4. 复位时，寄存器恢复原始存储状态，7 段数码管显示第一组 4 位字符（寄存器中的高 16 位），即 U202；

下图显示了通过 4 个数码管滚动显示 401234567，初始时刻和经过 3 个周期以后的效果：

(a) $t(0)$ 时刻(b) $t(3)$ 时刻

图 4.3: 数码管滚动显示学号例

4.3.5 | 设计数码管控制器

要求能稳定控制板子上 8 个数码管（从左往右依次为 DK7-DK0）同时能够稳定实现数字。

1. DK7-DK6: 显示自己名字前两个汉字的首字母，并稳定显示。
2. DK5-DK4: 显示从 10 开始可以循环且暂停的倒计时，并稳定显示。
3. DK3-DK0: 通过按键输入自己学号的后四位，并稳定显示。

关于实验4.3.5的补充说明请见附录E

4.4 | 实验要求

1. 在“实验四报告模板”完成实验报告。
2. 打包提交实验报告和实验的源程序文件。
3. 实验需要找助教或老师验收。

E | 实验4.3.4补充说明

E.1 | 实验4.3.4需要的模块

各模块间关系设计如下：

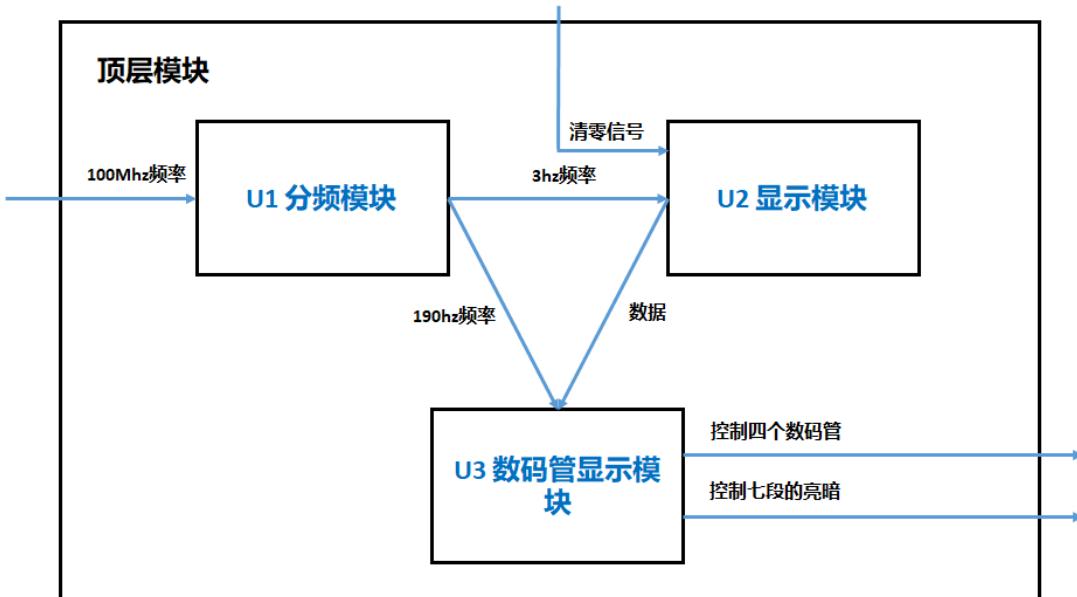


图 E.1: 模块图

各模块代码如下：

Listing 1: 顶层模块

```

1 module top(
2     input wire          clk100MHz,
3     input wire          rst      ,
4     output wire [ 3: 0] pos      ,
5     output wire [ 7: 0] seg
6 );
7
8 /*顶层连接线：
9  clk190Hz: 将分频模块的190Hz信号连接到数码管显示模块
10 clk3Hz: 将分频模块的3Hz信号连接到处理显示GPU模块
11 dataBus: 将处理模块处理后的数据连接到数码管显示模块
12 */
13
14     wire      clk190Hz, clk3Hz;
15     wire [15: 0] dataBus;
16
17 //例化三个子模块，并将他们连接
18 // TODO
19 endmodule

```

Listing 2: 分频模块 (clkDiv)

```

1 //这是一个分频模块
2 module clkDiv(
3     input wire clk100MHz, //100MHz的时钟频率是FPGA的系统时钟
4     input wire rst      , //复位信号
5     output wire clk190Hz , //分频得到的190hz
6     output wire clk3Hz   //分频得到的3hz
7 );
8
9 reg [25: 0] count; //定义一个计数器
10
11 assign clk190Hz = // TODO //根据计数器的第X位的变化产生新的时钟
12 assign clk3Hz   = // TODO //根据计数器的第X位的变化产生新的时钟
13
14 always @ (posedge clk100MHz)
15     if(rst)
16         count <= 26'd0;
17     else
18         count <= count + 26'd1;
19 endmodule

```

Listing 3: 显示处理模块 (GPU)

```

1 //这是数据处理后发送给数码管显示的模块
2 module GPU(
3     input wire      clk100MHz,
4     input wire      clk3Hz    ,
5     input wire      rst       ,
6     output wire [15: 0] dataBus
7 );
8     //移位寄存器
9     reg [39: 0] msgArray;
10
11    //将要显示的数据存为常数
12    parameter NUMBER = 40'hA202140000;
13
14    //把移位寄存器的高16位传递给数码管显示模块(四个数码管每个数码管显示四位数据)
15    assign dataBus = msgArray[39:24];
16
17    always @ (posedge clk100MHz)
18        if(rst)
19            msgArray<= NUMBER;
20        else if(clk3Hz)
21            // TODO
22 endmodule

```

Listing 4: 数码管显示模块 (segMsg)

```

1 //这是数码管显示模块
2 module segMsg(
3     input wire      clk190Hz,

```

```
4      input  wire          rst      ,
5      input  wire [15: 0] dataBus , //输入的数据总线
6      output reg   [ 3: 0] pos      ,
7      output reg   [ 7: 0] seg
8 );
9      reg   [ 1: 0] posC; //数码管计数器
10     reg   [ 3: 0] dataP; //数据的一部分
11
12    always @ (posedge clk190Hz)begin
13      if(rst)begin
14        pos   <= 4'hF;
15        dataP <= 4'hA;
16      end else begin
17        case(posC)
18          //将数据总线的3:0位显示在第一个数码管上
19          0: begin
20            pos   <= 4'b0001;
21            dataP <= dataBus[3:0];
22          end
23          //将数据总线的7:4位显示在第二个数码管上
24          1: begin
25            pos   <= 4'b0010;
26            dataP <= dataBus[7:4];
27          end
28          //将数据总线的11:8位显示在第三个数码管上
29          2: begin
30            pos   <= 4'b0100;
31            dataP<= dataBus[11:8];
32          end
33          //将数据总线的15:12位显示在第四个数码管上
34          3: begin
35            pos   <= 4'b1000;
36            dataP <= dataBus[15:12];
37          end
38        endcase
39      end
40    end
41
42    always @ (posedge clk190Hz ) begin
43      if(rst)
44        posC <= 2'd0;
45      else
46        posC <= posC + 2'd1;
47    end
48
49    //解码数据来控制数码管的每一段(这里取了0-9十个数字外加三个特殊显示)
50    always @ (dataP)
51      case(dataP)
52        0: seg = 8'b0011_1111;
53        1: seg = 8'b0000_0110;
```

```
54      2: seg = 8'b0101_1011;
55      3: seg = 8'b0100_1111;
56      4: seg = 8'b0110_0110;
57      5: seg = 8'b0110_1101;
58      6: seg = 8'b0111_1101;
59      7: seg = 8'b0000_0111;
60      8: seg = 8'b0111_1111;
61      9: seg = 8'b0110_1111;
62      10:seg = 8'b0100_0000;
63      11:seg = 8'b0000_0000;
64      default:seg = 8'b0000_1000;
65      endcase
66  endmodule
```

管脚约束见 Ego1 手册。

E.2 | 实验4.3.5对应的字母表

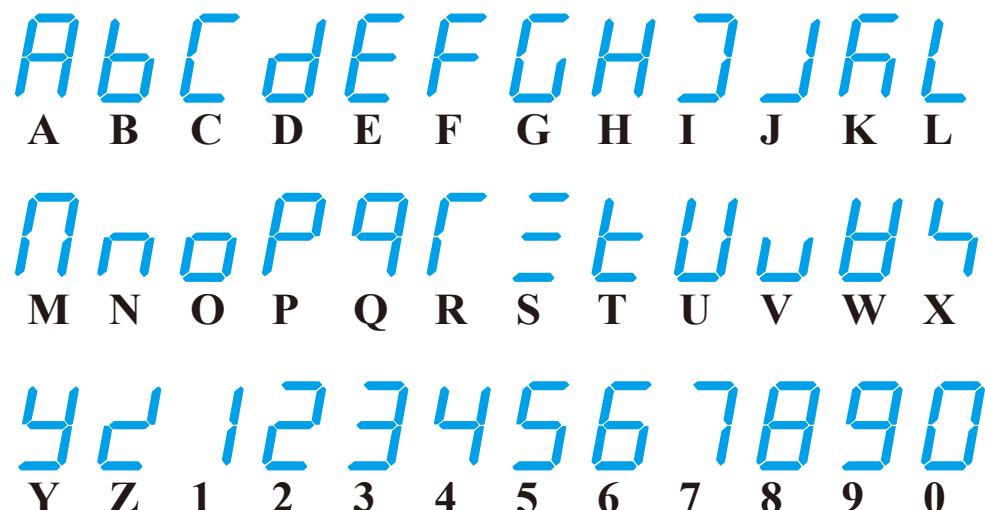
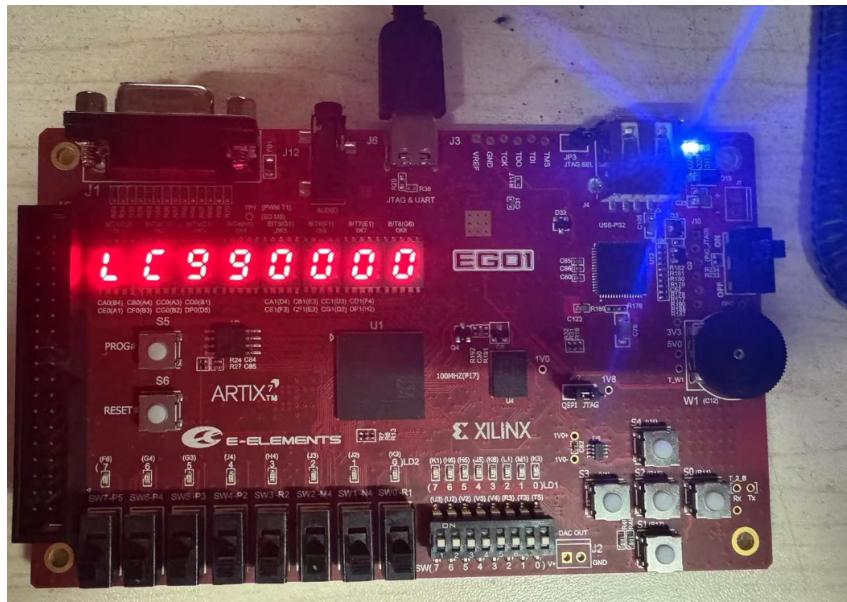


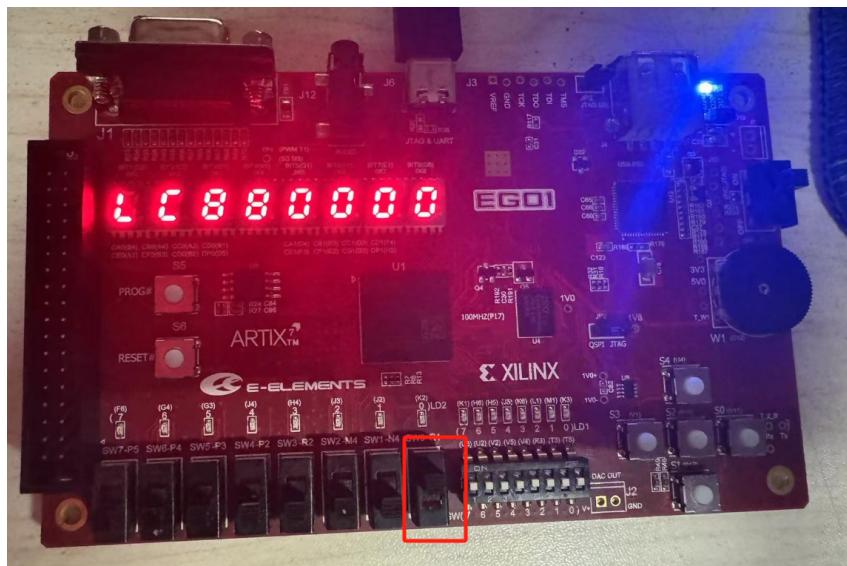
图 E.2: 字母表

E.3 | 实验4.3.5参考效果

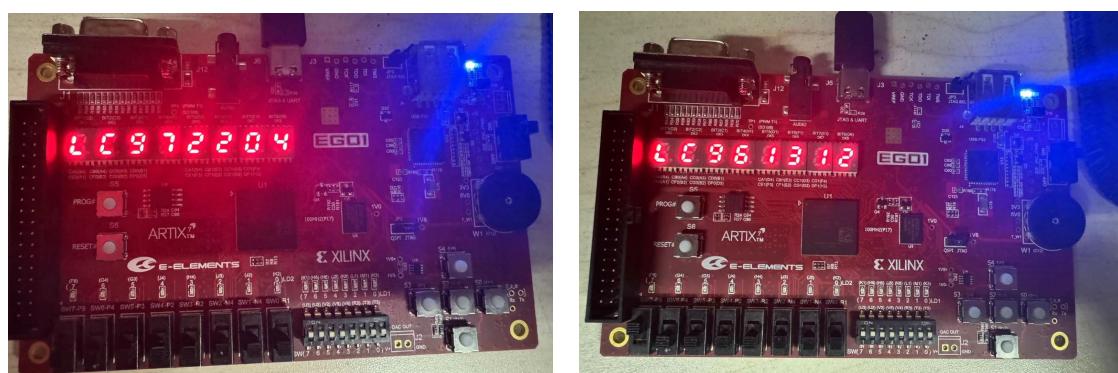
初始状态：前两位为字母，中间两位为可暂停倒计时，后四位显示可以键入的学号，初始为 0



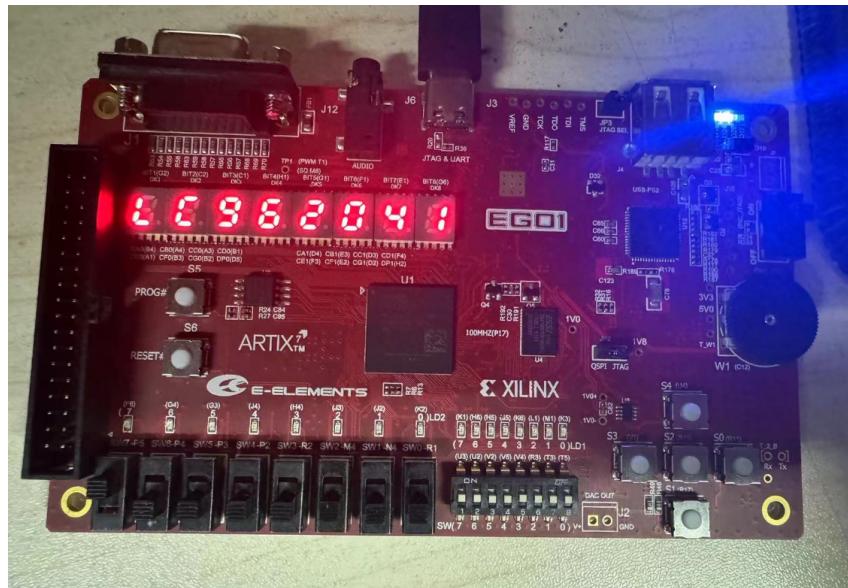
设置某个开关为倒计时启动开关，这里为红框所示



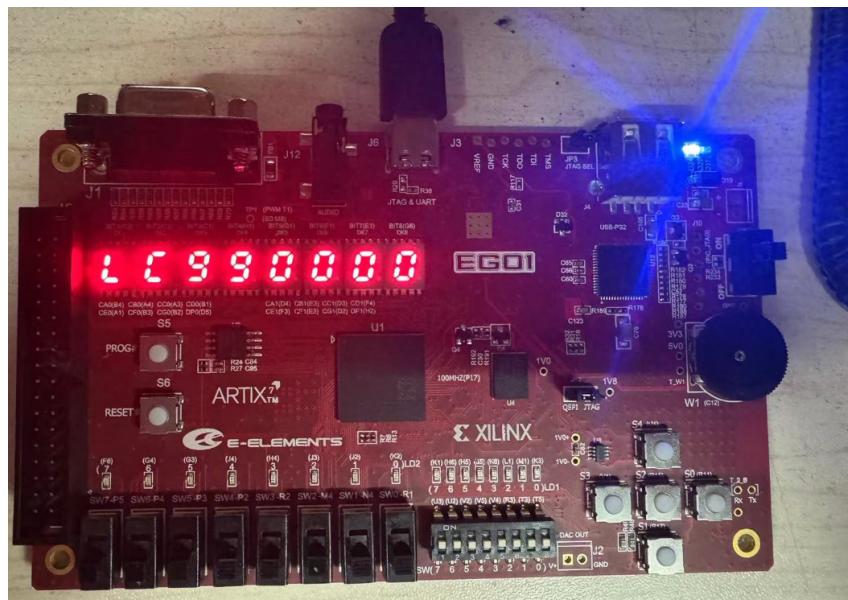
可实现八位数字的键入，实现方法不限，下图分别为前四位，后四位的键入



实现位移效果



实现复位/清零效果



强调：数码管的表现方式不固定，只要能实现要求效果即可，不一定需要按照本模板。