5. 数码管动态显示例程说明

5.1 PGX-Nano 开发板简介

PGX-Nano 开发板上有两个四位八段式共阳数码管，数码管共有两类控制信号，为段选信号和位选信号，其中数码管的位选信号输入段连接到了驱动 2N5401 的输出段。

5.2 实验目的

动态控制左边 4 位八段数码管显示不同的数值；

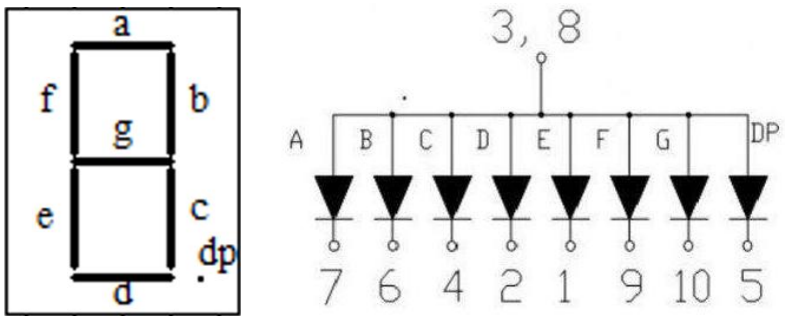
5.3 实验要求

4 个数码管显示不同的数字，按键 S0 控制左侧起第一个数码管，按一下数字加 1，从 0 到 9；按键 S1 控制左侧起第二个数码管，按一下数字加 1，从 0 到 9；按键 S2 控制左侧起第三个数码管，按一下数字加 1，从 0 到 9；按键 S3 控制左侧起第四个数码，按一下数字加 1，从 0 到 9。

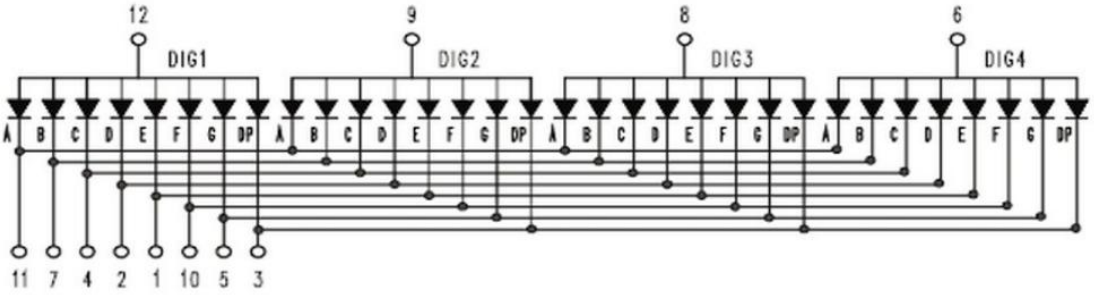
5.4 实验原理

5.4.1 数码管原理

数码管是一种半导体发光器件，其基本单元是发光二极管。能显示 4 个数码管叫四位数码管。数码管按段数分为七段数码管和八段数码管，八段数码管比七段数码管多一个发光二极管单元（多一个小数点显示）；按发光二极管单元连接方式分为共阳极数码管和共阴极数码管。共阳数码管是指将所有发光二极管的阳极接到一起形成公共阳极(COM)的数码管。共阳数码管在应用时应将公共极 COM 接到+5V，当某一字段发光二极管的阴极为低电平时，相应字段就点亮。当某一字段的阴极为高电平时，相应字段就不亮。共阴数码管是指将所有发光二极管的阴极接到一起形成公共阴极(COM)的数码管。共阴数码管在应用时应将公共极 COM 接到地线 GND 上，当某一字段发光二极管的阳极为高电平时，相应字段就点亮。当某一字段的阳极为低电平时，相应字段就不亮。

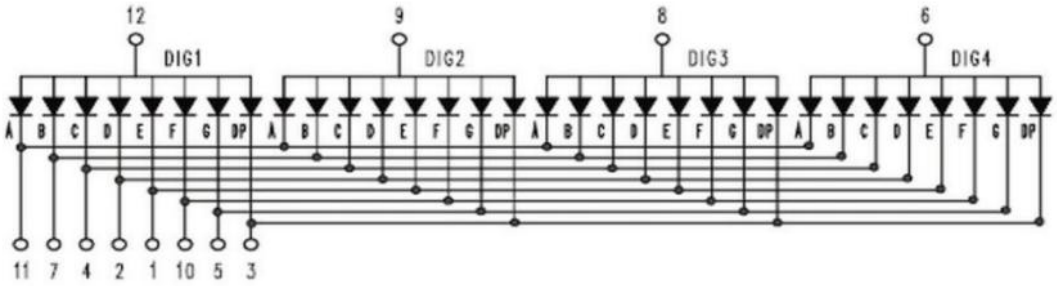


4 位共阳数码管内部管脚连接图如下：

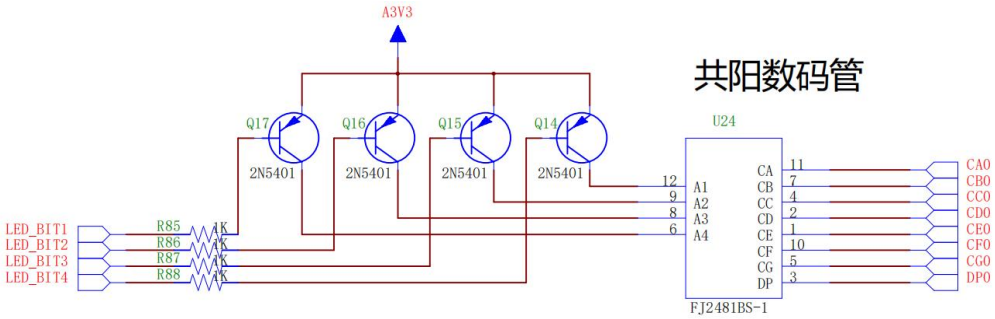


- 段选：段选由 8 根 led 灯组成，分别为 a，b，c，d，e，f，g，dp；
  由段选信号控制某段数码管点亮；
- 位选：位选由 4 组 8 个段选 LED 组成，分别为 DIG1、DIG2、DIG3、DIG4
  由选通信号控制第几块数码管点亮；

共阳极数码管上每组 8 段 LED 发光二极管阳极连接在一起，阳极由位选信号控制，阴极由段选信号控制，当提供位选信号高电平，段选信号低电平时，发光二极管被点亮。



PGX-Nano 开发板为数码管的位选信号配置了驱动 2N5401，其中当输入给 2N5401 低电平时，2N5401 会输出高电平，而 2N5401 输入端与 FPGA 相连，2N5401 的输出与数码管位选信号相连，因此 FPGA 输出低电平时，对应数码管位选信号有效。



5.4.2 数码管动态显示原理

四位数码管段选信号的对应段连接在一起，虽然极大程度上节约了引脚数量，但是也导致在同一时间，四位数码管只能显示相同的字段。如果希望看到四位数码管在同一时间显示不同字段，通常的解决办法为利用人眼视觉暂留特性，通过控制段选信号与位选信号的切换以达到动态扫描的目的。

人眼对于时间频率的响应近似一个滤波器，在一般室内强光下，对 15~20Hz 信号最敏感，有很强闪烁感(flick)，大于 75Hz 响应为 0，闪烁感消失。刚到达闪烁感消失的频率叫做临界融合频率(CFF)。在较暗的环境下，呈低通特性，且 CFF 会降低，这时对 5Hz 信号最敏感，大于 25Hz 闪烁基本消失。电影院环境很暗，放映机的刷新率为 24Hz 也不感到闪烁；这种特性也可以解析为视觉暂留特性，即当影像消失/变化时，大脑的影像不会立刻消失，而是保留一个短暂时间。

在设计数码管动态显示时，对于人眼观测来说，频率越高越好，但是数码管中的 LED 灯珠点亮对于高电平（关注发光响应时间）是有要求的，故而不是越高越好，取一个适当的刷新频率即可，实验中我们取刷新率为 10KHz。

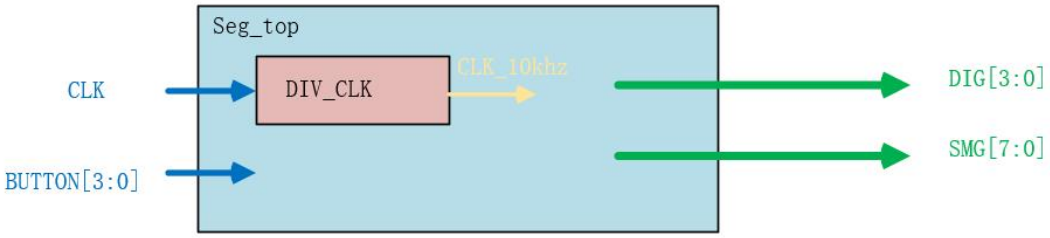
5.5 方案设计：

1、按键消抖：参考按键流水灯实验

2、按键计数：参考按键流水灯实验

3、数码管的分时显示；

实验框图如下：



5.6 实验源码（完整源码请参考 demo）

5.6.1 实验顶层

```
`timescale 1ns / 1ps

`define UD #1

module seg_dynamic(
   input            clk,//50MHZ 20ns
   input      [3:0]  button,
   output reg [3:0]  dig,
   output reg [7:0]  smg
   );
/*=====================================================
           key-vibration eliminate
=====================================================*/
wire [3:0]key;
btn_deb
#(
   .BTN_WIDTH(4'd4),
   .BTN_DELAY(20'hF423F)
)
u_btn_deb
(
    .clk(clk),
    .btn_in(button),
    .btn_deb(key)
);
/*=====================================================
           four key counts
=====================================================*/
wire [3:0] key0_cnt;
key_cnt key1
```

```
(
   .clk(clk),
   .key(key[0]),
   .key_times(key0_cnt)
);
wire [3:0] key1_cnt;
key_cnt key2
(
   .clk(clk),
   .key(key[1]),
   .key_times(key1_cnt)
);
wire [3:0] key2_cnt;
key_cnt key3
(
   .clk(clk),
   .key(key[2]),
   .key_times(key2_cnt)
);
wire [3:0] key3_cnt;
key_cnt key4
(
   .clk(clk),
   .key(key[3]),
   .key_times(key3_cnt)
);
/*=====================================================
              Divide the clock frequency
=====================================================*/
wire clk_10khz;
div_clk div_clk
(
   .clk     (clk),
   .clk_10khz (clk_10khz)  //frequency : 10KHz
);
/*=====================================================
               LED display
=====================================================*/
reg  [1:0]sel=0;
wire [3:0]dig0;
wire [7:0]smg0;

always @(posedge clk_10khz)
begin
```

```
        sel <= `UD sel+1'b1;
    end
    seq_control seq_control_0
    (
        .sel(2'd3),
        .key(key0_cnt),
        .dig(dig0),
        .smg(smg0)
    );

    wire [3:0]dig1;
    wire [7:0]smg1;

    seq_control seq_control_1
    (
        .sel(2'd2),
        .key(key1_cnt),
        .dig(dig1),
        .smg(smg1)
    );

    wire [3:0]dig2;
    wire [7:0]smg2;

    seq_control seq_control_2
    (
        .sel(2'd1),
        .key(key2_cnt),
        .dig(dig2),
        .smg(smg2)
    );

    wire [3:0]dig3;
    wire [7:0]smg3;

    seq_control seq_control_3
    (
        .sel(2'd0),
        .key(key3_cnt),
        .dig(dig3),
        .smg(smg3)
    );

    always @(posedge clk_10khz)
```

```
begin
   if(sel==2'b00)
      dig <= `UD dig0;
   else if(sel==2'b01)
      dig <= `UD dig1;
   else if(sel==2'b10)
      dig <= `UD dig2;
   else if(sel==2'b11)
      dig <= `UD dig3;
end

always @(posedge clk_10khz)
begin
   if(sel==2'b00)
      smg <= `UD smg0;
   else if(sel==2'b01)
      smg <= `UD smg1;
   else if(sel==2'b10)
      smg <= `UD smg2;
   else if(sel==2'b11)
      smg <= `UD smg3;
end

endmodule
```

5.6.2 按键消抖

```
`timescale 1ns / 1ps

`define UD #1

module btn_deb#(
    parameter                   BTN_WIDTH = 4'd8,
    parameter                   BTN_DELAY = 20'hF423F
)
(
    input                       clk,  //
    input      [BTN_WIDTH-1:0] btn_in,

    output reg [BTN_WIDTH-1:0] btn_deb
);
    reg [19:0]          cnt[BTN_WIDTH-1:0];
    reg [BTN_WIDTH-1:0] flag;
```

```
    reg [BTN_WIDTH-1:0] btn_in_reg;
    always @(posedge clk)
    begin
      btn_in_reg <= `UD btn_in;
    end

    genvar i;
    generate
    begin
        for(i=0;i<BTN_WIDTH;i=i+1)
        begin
            always @(posedge clk)
            begin
              if (btn_in_reg[i] ^ btn_in[i])
                flag[i] <= `UD 1'b1;
              else if (cnt[i]== BTN_DELAY)
                flag[i] <= `UD 1'b0;
              else
                flag[i] <= `UD flag[i];
            end

            always @(posedge clk)
            begin
              if(cnt[i]== BTN_DELAY)
                cnt[i] <= `UD 20'd0;
              else if(flag[i])
                cnt[i] <= `UD cnt[i] + 1'b1;
              else
                cnt[i] <= `UD 20'd0;
            end

            always @(posedge clk)
            begin
              if(flag[i])
                btn_deb[i] <= `UD btn_deb[i];
              else
                btn_deb[i] <= `UD btn_in[i];
            end
        end
    end
    endgenerate
endmodule
```

5.6.3 按键计数

```
`timescale 1ns / 1ps
`define UD #1
module key_cnt
(
    input clk,//50M,20ns
    input rstn_key,
    input key,
    output reg [3:0]key_times
);

reg key_reg;
always @(posedge clk)
begin
    key_reg <= `UD key;
end

always @(posedge clk )
begin
    if((key_reg)&&(~key)&&(key_times==4'd9))
        key_times <=`UD 4'd0;
    else if((key_reg)&&(~key))
        key_times <=`UD key_times + 1'b1;
end

endmodule
```

5.6.4 时钟分频

```
`timescale 1ns / 1ps
`define UD #1
module div_clk
(
    input clk,//50M
    output  clk_10khz//0.01M
);
reg [12:0]cnt;
always @(posedge clk)
begin
    if(cnt == 13'd4_999)
        cnt<= `UD 9'd0;
    else
```

```
        cnt <= `UD cnt + 1'b1;
end

reg flag=1'b0;
always @(posedge clk)
begin
    if(cnt == 13'd2_499)
        flag <= `UD 1'b1;
    else if(cnt == 12'd4_999)
        flag <= `UD 1'b0;
end
assign clk_10khz = flag;
endmodule
```

5.6.5 数码管显示

```
`timescale 1ns / 1ps
`define UD #1
module seq_control
(
    input [1:0]sel,
    input [3:0]key,
    output reg [3:0]dig,
    output reg [7:0]smg
);
/*=====================================================
=====================================================*/
always @(*)
begin
    case(sel)
        2'd0:dig = 4'b1110;
        2'd1:dig = 4'b1101;
        2'd2:dig = 4'b1011;
        2'd3:dig = 4'b0111;
        default:dig = 4'b1111;
    endcase
end
/*=====================================================
//0 1 2 3 4 5 6 7
//D F E D C B A DP
=====================================================*/
always @(*)
begin
    case(key)
```

```
            4'd0:smg = 8'b1000_0001;//"0"
            4'd1:smg = 8'b1100_1111;//"1"
            4'd2:smg = 8'b1001_0010;//"2"
            4'd3:smg = 8'b1000_0110;//"3"
            4'd4:smg = 8'b1100_1100;//"4"
            4'd5:smg = 8'b1010_0100;//"5"
            4'd6:smg = 8'b1010_0000;//"6"
            4'd7:smg = 8'b1000_1111;//"7"
            4'd8:smg = 8'b1000_0000;//"8"
            4'd9:smg = 8'b1000_0100;//"9"
            default:smg = 8'b1111_1111;
        endcase
    end



    endmodule
```

5.7 实验现象

4 个数码管显示不同的数字，按键 S0 控制左侧起第一个数码管，按一下数字加 1，从 0 到 9；按键 S1 控制左侧起第二个数码管，按一下数字加 1，从 0 到 9；按键 S2 控制左侧起第三个数码管，按一下数字加 1，从 0 到 9；按键 S3 控制左侧起第四个数码，按一下数字加 1，从 0 到 9。