9. 数字钟

9.1 实验目的

设计一个具有计时功能和校时功能的数字时钟,使用右边 4 位数码管进行显示时间。

9.2 实验要求

右侧四位数码管显示小时和分钟，秒钟用 LED 闪烁标识。

三个按键用于时钟校准。

S0 用于切换正常计时，校准小时和分钟

S1 用于时钟的 “+”

S2 用于时钟的“-”

校准相应的刻度，该数码管闪烁。

9.3 实验原理

从上述的实验要求分析可得到此数字钟我们实现过程中要注意两个功能点：

1、计时显示功能：LED 闪烁显示秒钟读秒，数码管右侧两位显示分钟计时，数码管左侧两位显示时钟计时；

此功能的实现由两个细节功能实现：①1S 计时控制，与前面的实验中需要计时功能模块实现方式一致，注意此处计时的周期为 1S 即可；②计时过程中进位控制；进位控制有四处需要进位：

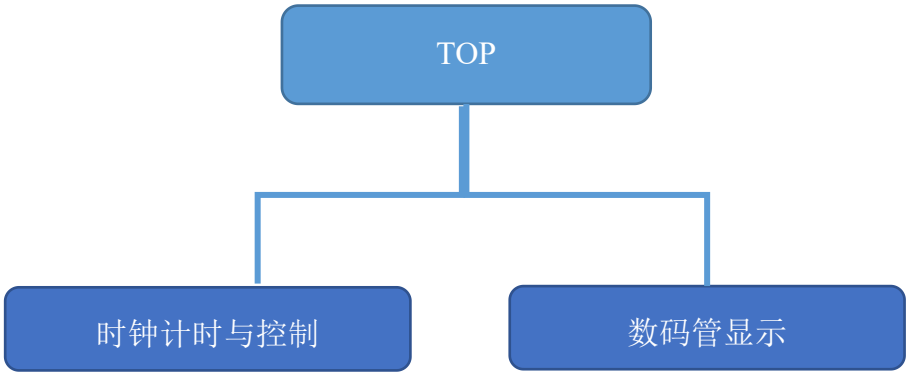
秒 进 分	LED 灯亮灭一次计数为 1，亮灭的一个周期为 1S,当 LED 亮灭 60 次时分钟计数个位加 1；
分$_{个位}$ 进 分$_{十位}$	当个位计数到 9 时，秒到分再次进位时，十位需要加一，个位置位为 0；
分进时	1 小时=60 分钟；当分钟计时为 59，并且秒到分再次进位时，时钟计数个位加 1；分钟计数置位为 0；
时$_{个位}$ 进 时$_{十位}$	显示为 24 小时制，当时钟计数十位小于 2 时，时钟个位个位计数到 9 时，分到时再次进位时，十位加 1；
24 小时制计满时归零	当时钟计数十位等于 2 时，时钟个位个位计数到 3 时，分到时再次进位时，时，分，秒三部分计数均归零；

2、计时校准功能：通过对应按键控制调整分钟计时与时钟计时，调整的过程中对应位需要闪烁；

此项功能中注意两点：①调整对应位，数码管该位进行闪烁；②调整时注意进位；

基于上述分析我们将项目分成两个部分：

1. 时钟计时与控制。
2. 数码管显示控制。



9.4 实验源码（完整源码查看 demo 源文件）

9.4.1 顶层设计

输入输出信号如下表：

信号	位宽	方向	描述
clk	1	输入	外部输入时钟，PGX-Lite 7K 板卡输入时钟为 50MHz
key	3	输入	时钟校准信号输入（轻触按键）
led	1	输出	时钟秒钟跳动显示（LED 灯闪烁一次，为 1S）
smg	8	输出	数码管段选输出
dig	4	输出	数码管位选输出

```
`timescale 1ns / 1ps
`define UD #1
module top_watch
(
    input clk,
    input [2:0]key,
```

```
    output led,
    output reg [3:0]dig,
    output reg [7:0]smg
);

  parameter CLK_FRE = 26'd50_000_000;
/*=============================================================
                        复位信号的产生
=============================================================*/
reg [4:0] rstn_cnt=0;
always @(posedge clk)
begin
    if(rstn_cnt==5'h1f)
        rstn_cnt <= `UD rstn_cnt;
    else
        rstn_cnt <= `UD rstn_cnt + 1'b1;
end

wire rstn;
assign rstn = rstn_cnt[4];
/*=============================================================
                            数字时钟的产生和控制
=============================================================*/
wire [3:0] hour_h_o,hour_l_o,minutes_h_o,minutes_l_o;
wire [2:0] state_flag;
watch_data_gen #(
    .CLK_FRE      (CLK_FRE      )
) u_watch_data_gen
(
    .clk            (clk            ), //input clk,
    .rstn           (rstn           ), //input rstn,
    .key            (key            ), //input [2:0]key,
    .hour_h_o       (hour_h_o       ), //output reg [3:0]hour_h_o,
    .hour_l_o       (hour_l_o       ), //output reg [3:0]hour_l_o,
    .minutes_h_o(minutes_h_o), //output reg [3:0]minutes_h_o,
    .minutes_l_o(minutes_l_o), //output reg [3:0]minutes_l_o,
    .second_led (led            ), //output reg second_led,
    .state_flag (state_flag )    //output reg [2:0]state_flag
);
/*=============================================================
                            时钟分频
=============================================================*/
wire clk_10khz;
div_clk #(
```

```
        .CLK_FRE       (CLK_FRE       )
)   div_clk
(
        .clk              (clk),
        .clk_10khz (clk_10khz)
);
/*=============================================================
                                数码管显示
=============================================================*/
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
(
        .clk(clk),
        .sec_en(led),
        .control_dig(state_flag),
        .sel(2'd0),
        .key(minutes_l_o),
        .dig(dig0),
        .smg(smg0)
);

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

seq_control seq_control_1
(
        .clk(clk),
        .sec_en(led),
        .control_dig(state_flag),
        .sel(2'd1),
        .key(minutes_h_o),
        .dig(dig1),
        .smg(smg1)
);

wire [3:0]dig2;
```

```
wire [7:0]smg2;

seq_control seq_control_2
(
        .clk(clk),
        .sec_en(led),
        .control_dig(state_flag),
        .sel(2'd2),
        .key(hour_l_o),
        .dig(dig2),
        .smg(smg2)
);

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

seq_control seq_control_3
(
        .clk(clk),
        .sec_en(led),
        .control_dig(state_flag),
        .sel(2'd3),
        .key(hour_h_o),
        .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
```

9.4.2 数字时钟产生与控制模块设计

在此模块中我们要实现前面描述的两个主要的功能点：计时与控制；

输入输出信号如下表：

信号	位宽	方向	描述
clk	1	输入	外部输入时钟，PGX-Lite 7K 板卡输入时钟为 50MHz
rstn	1	输入	外部输入复位信号，
key	3	输入	时钟校准信号输入（轻触按键）
hour_h_o	4	输出	时钟高位计数
hour_l_o	4	输出	时钟低位计数
minutes_h_o	4	输出	分钟高位计数
minutes_l_o	4	输出	分钟低位计数
second_led	1	输出	时钟秒钟跳动显示（LED 灯闪烁一次，为 1S）
state_flag	3	输出	数码管段选输出

Module 设计的关键点如下（完整 module 查看源文件）：

```
`timescale 1ns / 1ps
`define UD #1
module watch_data_gen #(
    parameter CLK_FRE = 26'd50_000_000
)
(
    input clk,
    input rstn,
    input [2:0]key,
    output reg [3:0]hour_h_o,
    output reg [3:0]hour_l_o,
```

```
        output reg [3:0]minutes_h_o,
        output reg [3:0]minutes_l_o,
        output reg second_led,
        output reg [2:0]state_flag
);

/*=====================================================
                        基准的产生
=====================================================*/
// 1s= 20ns * 50_000_000;
reg [25:0]second_cnt;
always @(posedge clk)
begin
        if(second_cnt==CLK_FRE-1'b1)
                second_cnt <= `UD 26'd0;
        else
                second_cnt <= `UD second_cnt + 1'b1;
end
//每个 1s 闪烁一次
always @(posedge clk)
begin
        if(!rstn)
                second_led <= `UD 1'b0;
        if(second_cnt==(CLK_FRE>>1)-1'b1)
                second_led <= `UD 1'b1;
        else if(second_cnt==CLK_FRE-1'b1)
                second_led <= `UD 1'b0;
end

reg [3:0]minutes_h;
reg [3:0]minutes_l;
reg [3:0]hour_h;
reg [3:0]hour_l;

reg [3:0]minutes_l_fix=0;
reg [3:0]minutes_h_fix=0;
reg [3:0]hour_l_fix=0;
reg [3:0]hour_h_fix=0;
/*=====================================================
                        校准逻辑产生
=====================================================*/
wire [2:0]key_out;

btn_deb #(
```

```
        .BT_WIDTH(4'd3),
        .CLK_FRE (CLK_FRE)
    )
    u_btn_deb
    (
        .clk       (clk),
        .btn_in (key),
        .btn_out(key_out)
    );
    /*=====================================================
        //key[0] -> k1 ;用于校准标记
        key_cnt = 3'd0 用于正常显示;
        key_cnt = 3'd1 用于分钟低位校准;
        key_cnt = 3'd2 用于分钟高位校准;
        key_cnt = 3'd3 用于时钟低位校准;
        key_cnt = 3'd4 用于时钟高位校准;
    =====================================================*/
    reg [2:0]key_out_reg=3'd0;
    always @(posedge clk)
    begin
        key_out_reg <= `UD key_out;
    end

    reg [2:0]key_cnt=3'd0;
    always @(posedge clk)
    begin
        if(key_cnt==3'd4 && (!key_out[0] && key_out_reg[0]))
            key_cnt <= `UD 3'd0;
        else if(!key_out[0] && key_out_reg[0])
            key_cnt <= `UD key_cnt + 1'b1;
    end
    /*=====================================================
                key[1] 用于"+"; key[2] 用"-"
    =====================================================*/
    always @(posedge clk)
    begin
        if(key_cnt==3'd0)//校准前将分钟低位和输出值保持一致
            minutes_l_fix <= `UD minutes_l;
        else  if(!key_out[1]  &&  key_out_reg[1]  &&  key_cnt==3'd1  &&
minutes_l_fix == 4'd9)//当处于分钟校准状态时，按下"+"按键,且此时校准值已经
为 9 时，再按下按键，则校准值变为 0
            minutes_l_fix <= `UD 4'd0;//"+"
        else  if(!key_out[2]  &&  key_out_reg[2]  &&  key_cnt==3'd1  &&
minutes_l_fix == 4'd0)//当处于分钟校准状态时，按下"-"按键,且此时校准值已经
```

```
为 0 时，再按下按键，则校准值变为 9
                minutes_l_fix <= `UD 4'd9;//"-"
            else if(!key_out[1] && key_out_reg[1] && key_cnt==3'd1)//当处于分钟
低位校准状态时，按下"+"按键,校准数值加 1；
                minutes_l_fix <= `UD minutes_l_fix + 1'b1;//"+"
            else if(!key_out[2] && key_out_reg[2] && key_cnt==3'd1)//当处于分钟
低位校准状态时，按下"-"按键,校准数值减 1；
                minutes_l_fix <= `UD minutes_l_fix - 1'b1;   //"-"
        end


        always @(posedge clk)
        begin
            if(key_cnt!=3'd2)//校准前数据和输出值保持一致
                minutes_h_fix <= `UD minutes_h;
            else   if(!key_out[1]   &&   key_out_reg[1]   &&   key_cnt==3'd2   &&
minutes_h_fix == 4'd5)//当处于校准状态时，按下"+"按键,且此时校准值已经为 5
时，再按下按键，则校准值变为 0
                minutes_h_fix <= `UD 4'd0;//"+"
            else   if(!key_out[2]   &&   key_out_reg[2]   &&   key_cnt==3'd2   &&
minutes_h_fix == 4'd0)//当处于分钟校准状态时，按下"-"按键,且此时校准值已经
为 0 时，再按下按键，则校准值变为 5
                minutes_h_fix <= `UD 4'd5;//"-"
            else if(!key_out[1] && key_out_reg[1] && key_cnt==3'd2)//当处于校准
状态时，按下"+"按键,按下"+"按键,校准数值加 1；
                minutes_h_fix <= `UD minutes_h_fix + 1'b1;//"+"
            else if(!key_out[2] && key_out_reg[2] && key_cnt==3'd2)//当处于校准
状态时，按下"-"按键,校准数值减 1；
                minutes_h_fix <= `UD minutes_h_fix - 1'b1;//"-"
        end


        always @(posedge clk)
        begin
            if(key_cnt!=3'd3 )//校准前数据赋值为 0
                hour_l_fix <= `UD 4'd0;
            else   if(!key_out[1]   &&   key_out_reg[1]   &&   key_cnt==3'd3   &&
hour_h_o != 4'd2 && hour_l_fix==4'd9)//当处于校准状态时，按下"+"按键,且此时
小时的高位不为 2 且校准值已经为 9 时，再按下按键，则校准值变为 0
                hour_l_fix <= `UD 4'd0;//"+"
            else if(!key_out[1] && key_out_reg[1] && key_cnt==3'd3 && hour_h_o
== 4'd2 && hour_l_fix==4'd3)//当处于校准状态时，按下"+"按键,且此时小时的高
位为 2 且校准值已经为 3 时，再按下按键，则校准值变为 0
                hour_l_fix <= `UD 4'd0;//"+"
```

```
        else if(!key_out[2] && key_out_reg[2] && key_cnt==3'd3 && hour_h_o != 4'd2 && hour_l_fix==4'd0)//当处于校准状态时，按下"-"按键,且此时小时的高位不为 2 时，校准位为 0，再按下按键，则校准值变为,9
            hour_l_fix <= `UD 4'd9;//"-"
        else if(!key_out[2] && key_out_reg[2] && key_cnt==3'd3 && hour_h_o == 4'd2 && hour_l_fix==4'd0)//当处于校准状态时，按下"-"按键,且此时小时的高位为 2 时，校准位为 0，再按下按键，则校准值变为 3
            hour_l_fix <= `UD 4'd3;//"-"
        else if(!key_out[1] && key_out_reg[1] && key_cnt==3'd3)//当处于校准状态时，按下"+"按键,按下"+"按键,校准数值加 1；
            hour_l_fix <= `UD hour_l_fix + 1'b1;//"+"
        else if(!key_out[2] && key_out_reg[2] && key_cnt==3'd3)//当处于校准状态时，按下"-"按键,校准数值减 1；
            hour_l_fix <= `UD hour_l_fix - 1'b1;
    end


    always @(posedge clk)
    begin
        if(key_cnt!=3'd4)//校准前数据和输出值保持一致
            hour_h_fix <= `UD hour_h;
        else if(!key_out[1] && key_out_reg[1] && key_cnt==3'd4 && hour_h_fix == 4'd2)//当处于校准状态时，按下"+"按键,且此时小时的高位为 2，再按下按键，则校准值变为 0
            hour_h_fix <= `UD 4'd0;//"+"
        else if(!key_out[2] && key_out_reg[2] && key_cnt==3'd4 && hour_h_fix == 4'd0)//当处于校准状态时，按下"-"按键,且此时小时的高位为 0，再按下按键，则校准值变为 2
            hour_h_fix <= `UD 4'd2;//"-"
        else if(!key_out[1] && key_out_reg[1] && key_cnt==3'd4 )//当处于校准状态时，按下"+"按键,按下"+"按键,校准数值加 1；
            hour_h_fix <= `UD hour_h_fix + 1'b1;//"+"
        else if(!key_out[2] && key_out_reg[2] && key_cnt==3'd4)//当处于校准状态时，按下"-"按键,校准数值减 1；
            hour_h_fix <= `UD hour_h_fix - 1'b1;//"-"
    end
    /*=====================================================
                        秒钟的产生:中间值
    =====================================================*/
    //秒钟计 60 次（0~59）为 1 分钟
    reg [5:0]second_flag=0;
    always @(posedge clk)
    begin
        if(second_cnt==CLK_FRE-1'b1 && second_flag==6'd59)
```

```
            second_flag <=`UD 6'd0;
        else if(second_cnt==CLK_FRE-1'b1)
            second_flag <=`UD second_flag + 1'b1;
    end
    /*==========================================================
                        分钟的产生:中间值
    ==========================================================*/
    //minutes_l gen
    always @(posedge clk)
    begin
        if(!rstn)//初始值为 0
            minutes_l <= `UD 4'd0;
        else if(key_cnt==3'd1)//校准时，分钟低位为校准值
            minutes_l <= `UD minutes_l_fix;
        else  if(second_cnt==CLK_FRE-1'b1  &&  second_flag==6'd59  &&
minutes_l==4'd9)//9 分 59 秒产生进位，低位赋值为 0
            minutes_l <= `UD 4'd0;
        else if(second_cnt==CLK_FRE-1'b1 && second_flag==6'd59)//60 秒产生
分钟的低位进位
            minutes_l <= `UD minutes_l +1'b1;
    end
    //minutes_h gen
    always @(posedge clk)
    begin
        if(!rstn)
            minutes_h <= `UD 4'd0;
        else if(key_cnt==3'd2)
            minutes_h <= `UD minutes_h_fix;
        else  if(second_cnt==CLK_FRE-1'b1  &&  second_flag==6'd59  &&
minutes_h==4'd5  && minutes_l==4'd9)//当为 59 分 59 秒的时候产生进位，分钟
的高位赋值为 0；
            minutes_h <= `UD 4'd0;
        else  if(second_cnt==CLK_FRE-1'b1  &&  second_flag==6'd59  &&
minutes_l==4'd9 )//9 分 59 秒的时候产生进位；
            minutes_h <= `UD minutes_h +1'b1;
    end
    /*==========================================================
                        小时的产生:中间值
    ==========================================================*/
    always @(posedge clk)
    begin
        if(!rstn)
            hour_l <= `UD 4'd0;
        else if(key_cnt==3'd3)
```

```
                hour_l <= `UD hour_l_fix;
            else if(hour_h!=4'd2 && hour_l==4'd9 && second_cnt==CLK_FRE-1'b1
&& second_flag==6'd59 && minutes_h==4'd5 && minutes_l==4'd9 )//9:59:59 或者
19:59:59，下一秒 hour_l 为 0；
                hour_l <= `UD 4'd0;
            else if(hour_h==4'd2 && hour_l==4'd3 && second_cnt==CLK_FRE-1'b1
&&          second_flag==6'd59              &&          minutes_h==4'd5          &&
minutes_l==4'd9 )//23:59:59;hour_l 为 0；
                hour_l <= `UD 4'd0;
            else     if(second_cnt==CLK_FRE-1'b1     &&     minutes_h==4'd5     &&
minutes_l==4'd9 && second_flag==6'd59)//XX:59:59;hour_l 加 1；
                hour_l <= `UD hour_l +1'b1;
        end

        always @(posedge clk)
        begin
            if(!rstn)
                hour_h <= `UD 4'd0;
            else if(key_cnt==3'd4)
                hour_h <= `UD hour_h_fix;
            else if(hour_h==4'd2 && hour_l==4'd3 && second_cnt==CLK_FRE-1'b1
&& minutes_h==4'd5 && minutes_l==4'd9 && second_flag==6'd59)//当时间刻度
为：23:59:59 ,hour_h 为 0
                hour_h <= `UD 4'd0;
            else      if(hour_l==4'd9      &&      second_cnt==CLK_FRE-1'b1      &&
minutes_h==4'd5 && minutes_l==4'd9 && second_flag==6'd59)//当时间刻度为：
09:59:59 or 19:59:59,hour_h 加 1
                hour_h <= `UD hour_h + 1'b1;
        end

        /*=====================================================
                            output
        =====================================================*/
        /*=====================================================
            //key[0] -> k1 ;用于校准标记
            key_cnt = 3'd0 用于正常显示；
            key_cnt = 3'd1 用于分钟低位校准；
            key_cnt = 3'd2 用于分钟高位校准；
            key_cnt = 3'd3 用于时钟低位校准；
            key_cnt = 3'd4 用于时钟高位校准；
        =====================================================*/
        always @(posedge clk)
        begin
            minutes_l_o <=`UD minutes_l;
```

```
end

always @(posedge clk)
begin
    minutes_h_o <=`UD minutes_h;
end

always @(posedge clk)
begin
    hour_l_o <=`UD hour_l;
end

always @(posedge clk)
begin
    hour_h_o <=`UD hour_h;
end

always @(posedge clk)
begin
    state_flag <= `UD key_cnt;
end

endmodule
```

9.4.3 时钟分频模块

```
`timescale 1ns / 1ps
`define UD #1
module div_clk #(
    parameter       CLK_FRE    = 26'd40_000_000
)
(
    input clk,
    output    clk_10khz
);
//times =20ns*5000=100us=10khz;
parameter CLK_DIV_100US = CLK_FRE/50_00;
reg [15:0]cnt;
always @(posedge clk)
begin
    if(cnt == CLK_DIV_100US - 1'b1)
        cnt<= `UD 16'd0;
    else
```

```
            cnt <= `UD cnt + 1'b1;
    end
    reg flag=1'b0;
    always @(posedge clk)
    begin
        if(cnt == (CLK_DIV_100US>>1) - 1'b1)
            flag <= `UD 1'b1;
        else if(cnt == CLK_DIV_100US - 1'b1)
            flag <= `UD 1'b0;
    end
    assign clk_10khz = flag;
    endmodule
```

9.4.4 数码管显示模块设计

数码管显示模块相比前一个实验需要增加一个功能：当进入校准模式时数码管的校准位需要进行闪烁，故而引入一个 1S 的周期信号，在 1S 时间内 0.5s 正常点亮，0.5s 不点亮使得数码管闪烁；闪烁对应位需要引入按键控制输出的 dig_ctl 信号（前面代码中有描述）；

闪烁控制的模块设计如下：

```
    always @(*)
    begin
        case(control_dig)
            4'd0: //正常显示
                case(sel)
                    2'd0:dig = 4'b0001;
                    2'd1:dig = 4'b0010;
                    2'd2:dig = 4'b0100;
                    2'd3:dig = 4'b1000;
                default:dig = 4'b0000;
                endcase
            4'd1://时钟高位校准
                case(sel)
                    2'd0:begin if(sec_en) dig = 4'b0001 ;else dig = 4'b0000; end
                    2'd1:dig = 4'b0010;
                    2'd2:dig = 4'b0100;
                    2'd3:dig = 4'b1000;
                default:dig = 4'b0000;
                endcase
            4'd2://时钟低位校准
                case(sel)
                    2'd0:dig = 4'b0001;
```

```
                    2'd1:begin if(sec_en) dig = 4'b0010 ;else dig = 4'b0000; end
                    2'd2:dig = 4'b0100;
                    2'd3:dig = 4'b1000;
                default:dig = 4'b0000;
                endcase
            4'd3: //分钟高位校准
                case(sel)
                    2'd0:dig = 4'b0001;
                    2'd1:dig = 4'b0010;
                    2'd2:begin if(sec_en) dig = 4'b0100 ;else dig = 4'b0000; end
                    2'd3:dig = 4'b1000;
                default:dig = 4'b0000;
                endcase
            4'd4: //分钟低位校准
                case(sel)
                    2'd0:dig = 4'b0001;
                    2'd1:dig = 4'b0010;
                    2'd2:dig = 4'b0100;
                    2'd3:begin if(sec_en) dig = 4'b1000 ;else dig = 4'b0000; end
                default:dig = 4'b0000;
                endcase
            default:dig = 4'b0000;
        endcase
    end
```

9.5 实验现象

加载后的显示结果为：数码管显示从 00：00 开始，LED1 闪烁（1 次/s）；

按轻触按键 KEY0，进入校准模式，第一次按下 S1，进入分钟低位计数校准调节，之后再次按下 S0，校准位将会往左移动 1 位，直到校准位为时钟计数高位时，按下 S0 将推出校准模式，进入正常计数模式；

在校准模式中按下轻触按键 S1 一次，对应校准位加 1，在可计数的最大值时会归 0；

在校准模式中按下轻触按键 S2 一次，对应校准位减 1，在减到 0 时会置位为可计数的最大值；