

## 課題18

---

### コード

#### Conter.v

```
input clk,
input stop,
input manual_up,
input manual_down,
output reg [3:0] q,
output reg c
);
reg [3:0] cnt = 4'd0;

always @(posedge clk) begin
    if (!manual_up && !manual_down) begin
        if(!stop) begin
            if (cnt == 4'd9) begin
                cnt <= 0;
                c <= 1;
            end else begin
                cnt <= cnt + 1;
                c <= 0;
            end
        end
    end else begin
        if (manual_up) begin
            if (cnt == 4'd9) begin
                cnt <= 0;
                c <= 1;
            end else begin
                cnt <= cnt + 1;
                c <= 0;
            end
        end
        if (manual_down) begin
            if (cnt == 0) begin
                cnt <= 9;
                c <= 0;
            end else begin
                cnt <= cnt - 1;
                c <= 0;
            end
        end
    end
    q <= cnt;
end
endmodule
```

```
module m_6_counter_manual(
    input clk,
    input stop,
    input manual_up,
    input manual_down,
    output reg [2:0] q,
    output reg c
);
    reg [2:0] cnt = 3'd0;

    always @(posedge clk) begin
        if (!manual_up && !manual_down) begin
            if(!stop) begin
                if (cnt == 3'd5) begin
                    cnt <= 0;
                    c <= 1;
                end else begin
                    cnt <= cnt + 1;
                    c <= 0;
                end
            end
        end else begin
            if (manual_up) begin
                if (cnt == 3'd5) begin
                    cnt <= 0;
                    c <= 1;
                end else begin
                    cnt <= cnt + 1;
                    c <= 0;
                end
            end
            if (manual_down) begin
                if (cnt == 0) begin
                    cnt <= 5;
                    c <= 0;
                end else begin
                    cnt <= cnt - 1;
                    c <= 0;
                end
            end
        end
        q <= cnt;
    end
endmodule

module m_24_counter_manual(
    input clk,
    input stop,
    input manual_up,
    input manual_down,
    output reg [3:0] q0, // units
    output reg [3:0] q1 // tens
);
    reg [4:0] cnt = 5'd0;
```

```

always @(posedge clk) begin
    if (!manual_up && !manual_down) begin
        if (!stop) begin
            if (cnt == 5'd23) cnt <= 0;
            else cnt <= cnt + 1;
        end
    end
    if (manual_up) begin
        if (cnt == 23) cnt <= 0;
        else cnt <= cnt + 1;
    end
    if (manual_down) begin
        if (cnt == 0) cnt <= 23;
        else cnt <= cnt - 1;
    end
    q1 <= cnt / 10; // tens
    q0 <= cnt % 10; // units
end
endmodule

module m_time_set(
    input clk,
    input mode,          // SW[9]
    input set_sec,        // SW[0]
    input set_min,        // SW[1]
    input set_hr,         // SW[2]
    input btn_up,         // BTN[0]
    input btn_down,       // BTN[1]
    input [3:0] milli_sec, // 100msカウント表示

    output reg manual_up_sec,
    output reg manual_down_sec,
    output reg manual_up_min,
    output reg manual_down_min,
    output reg manual_up_hr,
    output reg manual_down_hr,

    output reg [9:0] led_out
);

// -----
// 長押しカウンタ
// -----
reg [4:0] up_cnt;
reg [4:0] down_cnt;
parameter LONG_PRESS = 5'd5; // 適宜調整

// m_timer_decoder インスタンス
wire [9:0] timer_led;
m_timer_decoder u_decoder(milli_sec, timer_led);

always @(posedge clk) begin

```

```

if (mode) begin
    // BTN長押し判定
    if (btn_up) begin
        up_cnt <= 0;
    end else begin
        if (up_cnt < LONG_PRESS) up_cnt <= up_cnt + 1;
        else up_cnt <= LONG_PRESS;
    end

    if (btn_down) begin
        down_cnt <= 0;
    end else begin
        if (down_cnt < LONG_PRESS) down_cnt <= down_cnt + 1;
        else down_cnt <= LONG_PRESS;
    end

    manual_up_sec  <= set_sec  & ((up_cnt == LONG_PRESS) | (up_cnt
== 1));
    manual_down_sec<= set_sec  & ((down_cnt == LONG_PRESS) |
(down_cnt == 1));
    manual_up_min  <= set_min  & (up_cnt == LONG_PRESS);
    manual_down_min<= set_min  & (down_cnt == LONG_PRESS);
    manual_up_hr   <= set_hr   & (up_cnt == LONG_PRESS);
    manual_down_hr <= set_hr   & (down_cnt == LONG_PRESS);

    // 設定モード
    led_out[0] <= set_sec;
    led_out[1] <= set_min;
    led_out[2] <= set_hr;
    led_out[9] <= 1'b1;          // 光らせる

    // デバッグ用LED[3-8]
    led_out[3] <= btn_up;          // BTN[0] 押下
    led_out[4] <= btn_down;        // BTN[1] 押下
    led_out[5] <= (up_cnt == LONG_PRESS); // BTN[0] 長押し判定中
    led_out[6] <= (down_cnt == LONG_PRESS); // BTN[1] 長押し判定中
    led_out[7] <= manual_up_sec;    // 設定モード中
    led_out[8] <= manual_down_sec; // SW[0-2] 選択フラグ

end else begin
    // 通常モード: 0.1秒カウントをデコードして LED[0-9] に表示
    led_out <= timer_led;

    manual_up_sec  <= 0;
    manual_down_sec<= 0;
    manual_up_min  <= 0;
    manual_down_min<= 0;
    manual_up_hr   <= 0;
    manual_down_hr <= 0;

    up_cnt <= 0;
    down_cnt <= 0;
end

```

```
end  
endmodule
```

## Timer.v

```
module m_prescale50M(input clk, output c_out);  
    reg [25:0] cnt;  
    wire wcout;  
  
    assign wcout = (cnt == 26'd49999999) ? 1'b1 : 1'b0;  
    assign c_out = wcout;  
  
    always @(posedge clk) begin  
        if (wcout)  
            cnt <= 26'd0;  
        else  
            cnt <= cnt + 26'd1;  
        end  
    end  
endmodule  
  
// 1/50000000 PreScaler (for 50 MHz input -> 10 Hz)  
module m_prescale5M(input clk,output c_out);  
    reg [22:0] cnt;  
    wire wcout;  
  
    assign wcout=(cnt==23'd4999999) ? 1'b1 : 1'b0;  
    assign c_out=wcout;  
  
    always @(posedge clk) begin  
        if(wcout==1'b1)  
            cnt=0;  
        else  
            cnt=cnt+1;  
        end  
    end  
endmodule  
  
module m_prescale_50M_60Hz(input clk,output c_out);  
    reg [20:0] cnt;  
    wire wcout;  
  
    assign wcout=(cnt==21'd833333) ? 1'b1 : 1'b0;  
    assign c_out=wcout;  
  
    always @(posedge clk) begin  
        if(wcout==1'b1)  
            cnt=0;  
        else  
            cnt=cnt+1;  
        end  
    end  
endmodule
```

```

module m_prescale_50M_3600Hz(input clk, output c_out);
    reg [16:0] cnt;
    wire wcout;

    assign wcout=(cnt==17'd138888) ? 1'b1 : 1'b0;
    assign c_out=wcout;

    always @(posedge clk) begin
        if(wcout==1'b1)
            cnt=0;
        else
            cnt=cnt+1;
    end
endmodule

module m_timer_decoder(input [3:0] dcnt, output [9:0] wsec);
    assign wsec[0]=(dcnt==4'd0) ? 1'b1 : 1'b0;
    assign wsec[1]=(dcnt==4'd1) ? 1'b1 : 1'b0;
    assign wsec[2]=(dcnt==4'd2) ? 1'b1 : 1'b0;
    assign wsec[3]=(dcnt==4'd3) ? 1'b1 : 1'b0;
    assign wsec[4]=(dcnt==4'd4) ? 1'b1 : 1'b0;
    assign wsec[5]=(dcnt==4'd5) ? 1'b1 : 1'b0;
    assign wsec[6]=(dcnt==4'd6) ? 1'b1 : 1'b0;
    assign wsec[7]=(dcnt==4'd7) ? 1'b1 : 1'b0;
    assign wsec[8]=(dcnt==4'd8) ? 1'b1 : 1'b0;
    assign wsec[9]=(dcnt==4'd9) ? 1'b1 : 1'b0;
endmodule

```

## SevenSegment.v

```

module m_seven_segment(input [3:0] input_data,output [7:0] output_data);
    function [7:0] LedDec;
        input [3:0] num;
        begin
            case (num)
                4'h0:      LedDec = 8'b11000000; // 0
                4'h1:      LedDec = 8'b11111001; // 1
                4'h2:      LedDec = 8'b10100100; // 2
                4'h3:      LedDec = 8'b10110000; // 3
                4'h4:      LedDec = 8'b10011001; // 4
                4'h5:      LedDec = 8'b10010010; // 5
                4'h6:      LedDec = 8'b10000010; // 6
                4'h7:      LedDec = 8'b11111000; // 7
                4'h8:      LedDec = 8'b10000000; // 8
                4'h9:      LedDec = 8'b10011000; // 9
                4'ha:      LedDec = 8'b10001000; // A
                4'hb:      LedDec = 8'b10000011; // B
                4'hc:      LedDec = 8'b10100111; // C
                4'hd:      LedDec = 8'b10100001; // D
                4'he:      LedDec = 8'b10000110; // E
                4'hf:      LedDec = 8'b10001110; // F
            endcase
        end
    endfunction
endmodule

```

```

        default:      LedDec = 8'b11111111;  // LED OFF
    endcase
end
endfunction
assign output_data=LedDec(input_data);
endmodule

module m_seven_segment_add_dot(input [7:0] input_data, output [7:0]
output_data);
    assign output_data = input_data & 8'b01111111;
endmodule

```

## TopModule.v

```

module TopModule(
    //////////// CLOCK ////////////
    input                CLK1,
    input                CLK2,
    //////////// SEG7 ////////////
    output [7:0]         HEX0,
    output [7:0]         HEX1,
    output [7:0]         HEX2,
    output [7:0]         HEX3,
    output [7:0]         HEX4,
    output [7:0]         HEX5,
    //////////// Push Button ////////////
    input [1:0]          BTN,
    //////////// LED ////////////
    output [9:0]         LED,
    //////////// SW ////////////
    input [9:0]          SW

);

// -----
// 100ms クロック生成 (m_prescale5M を使用)
// -----
wire clk_100ms;
m_prescale5M u0(CLK1, clk_100ms);

// -----
// 1秒パルス用カウンタ (100ms×10 = 1秒)
// -----
wire clk_1s;
reg [3:0] cnt_100ms;
always @(posedge clk_100ms) begin
    if (SW[9]) begin
        cnt_100ms <= cnt_100ms; // 設定モードで停止
    end else begin

```

```

        if (cnt_100ms == 4'd9) cnt_100ms <= 0;
        else cnt_100ms <= cnt_100ms + 1;
    end
end
assign clk_1s = (cnt_100ms == 4'd9) && !SW[9];

// -----
// m_time_set 出力 (manual_up/down + LED)
// -----
wire manual_up_sec, manual_down_sec;
wire manual_up_min, manual_down_min;
wire manual_up_hr, manual_down_hr;

wire [3:0] milli_sec;
assign milli_sec = cnt_100ms;

m_time_set u_set(
    .clk(clk_100ms),
    .mode(SW[9]),
    .set_sec(SW[0]),
    .set_min(SW[1]),
    .set_hr(SW[2]),
    .btn_up(BTN[0]),
    .btn_down(BTN[1]),
    .manual_up_sec(manual_up_sec),
    .manual_down_sec(manual_down_sec),
    .manual_up_min(manual_up_min),
    .manual_down_min(manual_down_min),
    .manual_up_hr(manual_up_hr),
    .manual_down_hr(manual_down_hr),
    .milli_sec(milli_sec),
    .led_out(LED)
);

// -----
// 共通カウンタ (手動+自動)
// -----
wire [3:0] sec0, sec1, min0, min1, hr0, hr1;
wire clk_sec0, clk_sec1, clk_min0, clk_min1, clk_hr;

// 秒
m_10_counter_manual c_sec0(
    .clk(clk_1s),
    .stop(SW[8]),
    .manual_up(manual_up_sec),
    .manual_down(manual_down_sec),
    .q(sec0),
    .c(clk_sec0)
);

m_6_counter_manual c_sec1(
    .clk(clk_sec0),
    .stop(SW[8]),
    .manual_up(manual_up_sec),

```



```

        .manual_down(manual_down_sec),
        .q(sec1),
        .c(clk_sec1)
    );

    // 分
    m_10_counter_manual c_min0(
        .clk(clk_sec1),
        .stop(SW[8]),
        .manual_up(manual_up_min),
        .manual_down(manual_down_min),
        .q(min0),
        .c(clk_min0)
    );

    m_6_counter_manual c_min1(
        .clk(clk_min0),
        .stop(SW[8]),
        .manual_up(manual_up_min),
        .manual_down(manual_down_min),
        .q(min1),
        .c(clk_min1)
    );

    // 時
    m_24_counter_manual c_hr(
        .clk(clk_min1),
        .stop(SW[8]),
        .manual_up(manual_up_hr),
        .manual_down(manual_down_hr),
        .q0(hr0),
        .q1(hr1)
    );

    // -----
    // 7セグメント表示
    // -----
    m_seven_segment u2(sec0, HEX0);
    m_seven_segment u3(sec1, HEX1);
    m_seven_segment u4(min0, HEX2);
    m_seven_segment u5(min1, HEX3);
    m_seven_segment u6(hr0, HEX4);
    m_seven_segment u7(hr1, HEX5);

endmodule

```

## 動作確認

- 通常モードではLEDでミリ秒を示すことを確認した
- 通常モードでは7セグが秒、分、時間を示すことを確認した
- 設定モードにはSW[9]で入り、SW[0,1,2]で秒、分、時間の設定を選択できることを確認した

- 設定モードではBTN[0]で増加、BTN[1]で減少できることを確認した
- 設定モードではBTNの長押しで連続増減できることを確認した
- SW[8]でカウントを停止できることを確認した