LCD 制御回路

厳密には LCD 制御回路制御回路（LCD 制御回路を制御する回路）

ソースコード

```
`define INIT    2'b00
`define SETPOS0 2'b01
`define SETPOS1 2'b10
`define WRITE   2'b11

module lcdctrl(clk,reset,lcd_e,lcd_rs,lcd_rw,sf_d,data0,data1,data2,data3,data4,data5);

 input clk,reset;
 input [15:0] data0,data1,data2,data3,data4,data5;
 output lcd_e,lcd_rs,lcd_rw;
 output [11:8] sf_d;
 reg lcd_e;

 reg [1:0] state;
 reg [2:0] index;
 reg [3:0] addr;
 reg [31:0] counter;
 reg [26:0] ctrl;
 wire set_enb;
 wire ret;
 wire [19:0] wait_cnt;
 reg [7:0] ascii;
 reg [3:0] hex;

 assign lcd_rw   = 0;
 assign ret      = ctrl[26];
 assign lcd_rs   = ctrl[25];
 assign set_enb  = ctrl[24];
 assign sf_d     = ctrl[23:20];
 assign wait_cnt = ctrl[19:0];

 always @(posedge clk or negedge reset)
   if(!reset) begin
     addr <= 0;
     counter <= 0;
     state <= `INIT;
     index <= 0;
   end else if(counter<wait_cnt)
     counter <= counter + 1;
   else begin
     counter <= 0;
     if(!ret)
       addr <= addr + 1;
     else begin
       addr <= 0;
       if(state==`INIT)
         state <= `SETPOS0;
       else if(state==`SETPOS0 || state==`SETPOS1)
         state <= `WRITE;
       else if(state == `WRITE) begin
         if(index== 5)
           index <= 0;
         else
           index <= index + 1;
         if(index == 2)
           state <= `SETPOS1;
         else if(index == 5)
           state <= `SETPOS0;
       end
     end
   end

 always @(posedge clk or negedge reset)
   if(!reset) lcd_e <= 0;
   else if(set_enb && counter >= 1 && counter <= 12) lcd_e <= 1;
   else lcd_e <= 0;

 always @(state or addr or ascii)
   case(state)
   `INIT:
     case(addr)
     4'h0: ctrl = {3'b000, 4'h0, 20'hB71B0};
```

```
    4'h1: ctrl = {3'b001, 4'h3, 20'h320D7};
    4'h2: ctrl = {3'b001, 4'h3, 20'h01397};
    4'h3: ctrl = {3'b001, 4'h3, 20'h00800};
    4'h4: ctrl = {3'b001, 4'h2, 20'h00800};
    4'h5: ctrl = {3'b001, 4'h2, 20'h00041};
    4'h6: ctrl = {3'b001, 4'h8, 20'h00800};
    4'h7: ctrl = {3'b001, 4'h0, 20'h00041};
    4'h8: ctrl = {3'b001, 4'h6, 20'h00800};
    4'h9: ctrl = {3'b001, 4'h0, 20'h00041};
    4'hA: ctrl = {3'b001, 4'hC, 20'h00800};
    4'hB: ctrl = {3'b001, 4'h0, 20'h00041};
    4'hC: ctrl = {3'b101, 4'h1, 20'h1482F};
    default: ctrl = {3'bxxx, 4'hx, 20'hxxxxx};
    endcase
  `SETPOS0:
    case(addr)
    4'h0: ctrl = {3'b001, 4'h8, 20'h00041};
    4'h1: ctrl = {3'b101, 4'h0, 20'h00800};
    default: ctrl = {3'bxxx, 4'hx, 20'hxxxxx};
    endcase
  `SETPOS1:
    case(addr)
    4'h0 : ctrl = {3'b001, 4'hC, 20'h00041};
    4'h1 : ctrl = {3'b101, 4'h0, 20'h00800};
    default: ctrl = {3'bxxx, 4'hx, 20'hxxxxx};
    endcase
  `WRITE:
    case(addr)
    4'h0 : ctrl = {3'b011, ascii[7:4], 20'h00041};
    4'h1 : ctrl = {3'b011, ascii[3:0], 20'h00800};
    4'h2 : ctrl = {3'b011, ascii[7:4], 20'h00041};
    4'h3 : ctrl = {3'b011, ascii[3:0], 20'h00800};
    4'h4 : ctrl = {3'b011, ascii[7:4], 20'h00041};
    4'h5 : ctrl = {3'b011, ascii[3:0], 20'h00800};
    4'h6 : ctrl = {3'b011, ascii[7:4], 20'h00041};
    4'h7 : ctrl = {3'b011, ascii[3:0], 20'h00800};
    4'h8 : ctrl = {3'b011, 4'hA, 20'h00041};
    4'h9 : ctrl = {3'b111, 4'h0, 20'h00800};
    default: ctrl = {3'bxxx, 4'hx, 20'hxxxxx};
    endcase
  default: ctrl = {3'bxxx, 4'hx, 20'hxxxxx};
  endcase

reg [15:0] data;
always @(posedge clk or negedge reset)
  if(!reset) data <= 0;
  else if(addr == 0)
    case(index)
    3'd0: data <= data0;
    3'd1: data <= data1;
    3'd2: data <= data2;
    3'd3: data <= data3;
    3'd4: data <= data4;
    3'd5: data <= data5;
    endcase

always @(addr or data)
  case(addr)
  4'h0,4'h1: hex = data[15:12];
  4'h2,4'h3: hex = data[11:8];
  4'h4,4'h5: hex = data[7:4];
  4'h6,4'h7: hex = data[3:0];
  default: hex = 4'hx;
  endcase

always @(hex)
  case(hex)
  4'h0 : ascii = 8'h30;
  4'h1 : ascii = 8'h31;
  4'h2 : ascii = 8'h32;
  4'h3 : ascii = 8'h33;
  4'h4 : ascii = 8'h34;
  4'h5 : ascii = 8'h35;
  4'h6 : ascii = 8'h36;
  4'h7 : ascii = 8'h37;
  4'h8 : ascii = 8'h38;
  4'h9 : ascii = 8'h39;
  4'hA : ascii = 8'h41;
  4'hB : ascii = 8'h42;
  4'hC : ascii = 8'h43;
  4'hD : ascii = 8'h44;
  4'hE : ascii = 8'h45;
```

```
    4'hF : ascii = 8'h46;
    default : ascii = 8'hxx;
    endcase

endmodule
```