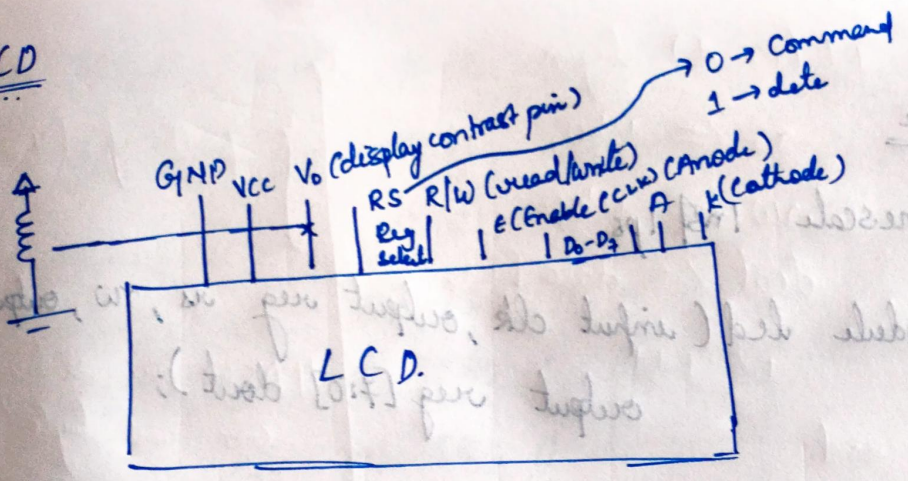
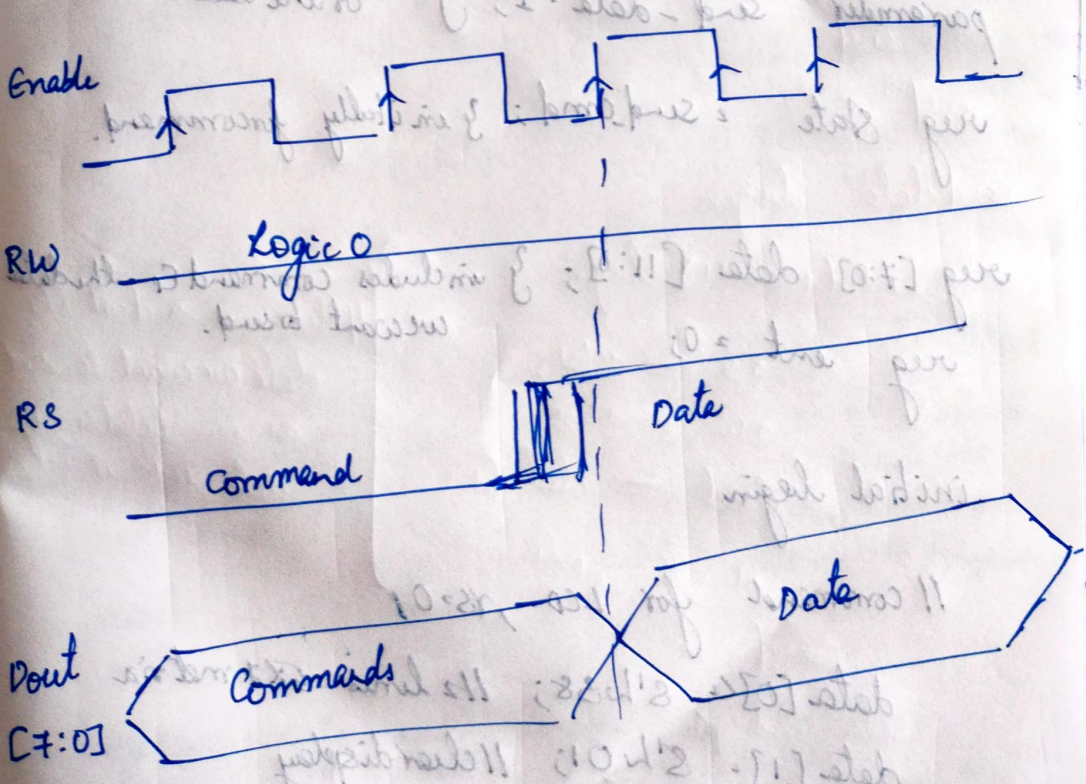
LCD



- On each +ve edge of an enable we send a new data to the LCD.



Code

```
timescale 1ns/1ps

module lcd(input clk, output reg rs, rw, output en,
           output reg [7:0] dout);

integer count = 0;  // to maintain the time on the physical board.
integer i = 0;  // maintain count of data & command.
parameter send_cmd = 0;
parameter send_data = 1;  // to set the state for command or the data.

reg state = send_cmd;  // initially for command.

reg [7:0] data [11:0];  // includes command & the data we want to send.
reg ent = 0;

initial begin
// command for LCD, rs=0
data[0] <= 8'h38;  // 2 lines 5x7 matrix
data[1] = 8'h01;  // clear display
data[2] = 8'h0E;  // display on cursor blinking
data[3] = 8'h06;  // increment cursor from left to right
data[4] = 8'h80;  // force cursor from beginning of first line.
```

```
/// Data for LCD , rs = 1        // I want to send
                                   - vardhan to display
data[5] <= 8'h76;   // v
data[6] <= 8'h61;   // a
data[7] <= 8'h72;   // r
data[8] <= 8'h64;   // d
data[9] <= 8'h68;   // h
data[10] <= 8'h61;  // a
data[11] <= 8'h6E;  // n
```

ASCII values

```
always @(posedge clk) begin
  if (count < 10)   // As it was simulation on tool count = 10
                    // if it was on Basys3 board
                    // where clk = 100MHz
                    // count = 1000000
                    //   ↑
                    //  10ms //.
    count = count + 1;
  else
    begin
      count <= 0;
      ent <= ~ent;
    end
end.
```

// Complete always block is to maintain a slower clk. based on our board clk frequency.

```
always @(posedge ent) begin
case (state)
send_cmd: begin
```

to maintain the state.

```
if (i<4) begin
    rs <= 1'b0;
    rw <= 1'b0;
    dout <= data[i];
    i<= i+1;
end
else begin
    state <= send_data;
    dout <= data[i]
    i<= i+1;
end
```

for the first 4 commands

// As we are sending 5 command bytes

this runs for last one & to change the state.

```
send_data : begin
    if (i<=11) begin
        rs <= 1'b1;
        rw <= 1'b0;
        dout <= data[i];
        i <= i+1;
    end
    else begin
        i <= 0
        state <= send_cmd;
        rs <= 1'b0;
        rw <= 1'b0;
        dout <= 8'h00;
    end
```

// Here we send all the data we want to send as our total size of data is 12 it's <=11

// Again we move on to send command state with i=0

```
    end
  endcase
end

assign en = ent;        // to maintain the enable given to the
                           led with our ent signal as a
                           clk.
endmodule.
```

---

x
