

```
1  `timescale 1ns / 1ps
2  ///////////////////////////////////////////////////////////////////
3  // Company:
4  // Engineer:
5  //
6  // Create Date:    12:47:20 06/25/2021
7  // Design Name:
8  // Module Name:    I2C_Master
9  // Project Name:
10 // Target Devices:
11 // Tool versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 ///////////////////////////////////////////////////////////////////
21 module I2C_Master(
22     input wire clk,
23     input wire rst,
24     input wire [6:0] addr,
25     input wire [7:0] data_in,
26     input wire enable,
27     input wire rw,
28
29     //output reg [7:0] data_out,
30     output wire ready,
31
32     inout i2c_sda,
33     inout wire i2c_scl
34 );
35
36 localparam IDLE = 0;
37 localparam START = 1;
38 localparam ADDRESS = 2;
39 localparam READ_ACK = 3;
40 localparam WRITE_DATA = 4;
41 localparam WRITE_ACK = 5;
42 localparam READ_DATA = 6;
43 localparam READ_ACK2 = 7;
44 localparam STOP = 8;
45
46 localparam DIVIDE_BY = 4;
47
48 reg [7:0] state;
49 reg [7:0] saved_addr;
50 reg [7:0] saved_data;
51 reg [7:0] counter;
52 reg [7:0] counter2 = 0;
53 reg write_enable;
54 reg sda_out;
55 reg i2c_scl_enable = 0;
56 reg i2c_clk = 1;
57
```

```
58     assign ready = ((rst == 0) && (state == IDLE)) ? 1 : 0;
59     assign i2c_scl = (i2c_scl_enable == 0) ? 1 : i2c_clk;
60     assign i2c_sda = (write_enable == 1) ? sda_out : 'bz;
61
62     always @(posedge clk) begin
63         if (counter2 == (DIVIDE_BY/2) - 1) begin
64             i2c_clk <= ~i2c_clk;
65             counter2 <= 0;
66         end
67         else counter2 <= counter2 + 1;
68     end
69
70     always @(negedge i2c_clk, posedge rst) begin
71         if(rst == 1) begin
72             i2c_scl_enable <= 0;
73         end else begin
74             if ((state == IDLE) || (state == START) || (state == STOP)) begin
75                 i2c_scl_enable <= 0;
76             end else begin
77                 i2c_scl_enable <= 1;
78             end
79         end
80
81     end
82
83
84     always @(posedge i2c_clk, posedge rst) begin
85         if(rst == 1) begin
86             state <= IDLE;
87         end
88         else begin
89             case(state)
90
91                 IDLE: begin
92                     if (enable) begin
93                         state <= START;
94                         saved_addr <= {addr, rw};
95                         saved_data <= data_in;
96                     end
97                     else state <= IDLE;
98                 end
99
100                START: begin
101                    counter <= 7;
102                    state <= ADDRESS;
103                end
104
105                ADDRESS: begin
106                    if (counter == 0) begin
107                        state <= READ_ACK;
108                    end else counter <= counter - 1;
109                end
110
111                READ_ACK: begin
112                    if (i2c_sda == 0) begin
113                        counter <= 7;
114                        if(saved_addr[0] == 0) state <= WRITE_DATA;
```

```
115         else state <= READ_DATA;
116     end else state <= STOP;
117 end
118
119 WRITE_DATA: begin
120     if(counter == 0) begin
121         state <= READ_ACK2;
122     end else counter <= counter - 1;
123 end
124
125 READ_ACK2: begin
126     if ((i2c_sda == 0) && (enable == 1)) state <= IDLE;
127     else state <= STOP;
128 end
129
130 READ_DATA: begin
131     //data_out[counter] <= i2c_sda;
132     if (counter == 0) state <= WRITE_ACK;
133     else counter <= counter - 1;
134 end
135
136 WRITE_ACK: begin
137     state <= STOP;
138 end
139
140 STOP: begin
141     state <= IDLE;
142 end
143 endcase
144 end
145 end
146
147 always @(negedge i2c_clk, posedge rst) begin
148     if(rst == 1) begin
149         write_enable <= 1;
150         sda_out <= 1;
151     end else begin
152         case(state)
153
154             START: begin
155                 write_enable <= 1;
156                 sda_out <= 0;
157             end
158
159             ADDRESS: begin
160                 sda_out <= saved_addr[counter];
161             end
162
163             READ_ACK: begin
164                 write_enable <= 0;
165             end
166
167             WRITE_DATA: begin
168                 write_enable <= 1;
169                 sda_out <= saved_data[counter];
170             end
171
```

```
172         WRITE_ACK: begin
173             write_enable <= 1;
174             sda_out <= 0;
175         end
176
177         READ_DATA: begin
178             write_enable <= 0;
179         end
180
181         STOP: begin
182             write_enable <= 1;
183             sda_out <= 1;
184         end
185     endcase
186 end
187 end
188
189 endmodule
190
```