```
`timescale 1ns/1ps
// Verilog code to test UTC and DTC of your 16 bit counter



module testTC();
  reg clkin, btnR, btnC, btnU, btnD, btnL;
  reg [15:0] sw;
  wire [15:0] led;
  wire [6:0] seg;
  wire [3:0] an;
  wire dp, UTC, DTC;
  wire [7:0] D7Seg3,D7Seg2,D7Seg1,D7Seg0;

  Toplevel
   UUT (
      .clkin(clkin),
      .btnR(btnR),
      .btnU(btnU),
      .btnD(btnD),
      .btnL(btnL),
      .btnC(btnC),
      .sw(sw),
      .seg(seg),
      .dp(dp),
      .led(led),
      .an(an)
      );
  show_7segDisplay  showit (.seg(seg),.dp(dp),.an(an),
              .D7Seg0(D7Seg0),.D7Seg1(D7Seg1),.D7Seg2(D7Seg2),.D7Seg3(D7Seg3));

  integer TX_ERROR = 0;
  // Make UTC & DTC available to this testbench - 2 choices
  // 1) modify your design (temporarily) to connect UTC & DTC to leds
  // 2) use cross module references, e.g. assign UTC = UUT.utc;
  assign UTC = led[0];
  assign DTC = led[15];

// Run this simulation for 3ms. If correct TX_ERROR should be 0 at the end.
// Interesting things happen around 2640usec to 2654usec

  parameter PERIOD = 10;
  parameter real DUTY_CYCLE = 0.5;
  parameter OFFSET = 2;
```

```
initial    // Clock process for clkin
begin
  btnC = 1'bx;
  btnR = 1'b0;
  btnU = 1'bx;
  btnD = 1'bx;
  btnL = 1'bx;

#OFFSET
    clkin = 1'b1;
forever
begin
  #(PERIOD-(PERIOD*DUTY_CYCLE)) clkin = ~clkin;
end
end

initial
begin
#2000;
btnC=1'b0;
btnU=1'b0;
btnD=1'b0;
btnL=1'b0;
btnR=1'b0;
sw = 16'h9034;
#200;
btnR = 1'b1;
#500 btnR = 1'b0;
#300 btnC=1'b1;
btnR = 1'b0;
#2640000 btnC=1'b0;
#200; btnU=1'b1;
#600; btnU=1'b0;
#200; btnU=1'b1;
#700; btnU=1'b0;
#200; btnU=1'b1;
#800; btnU=1'b0;
CHECK_UTC(1'b1); CHECK_DTC(1'b0);
#400;
CHECK_UTC(1'b1); CHECK_DTC(1'b0);
#200;
btnU=1'b1;
#200;
btnU=1'b0;
#100;
CHECK_UTC(1'b0); CHECK_DTC(1'b1);
```

```
   #200; btnU=1'b0;
   #200; btnU=1'b1;
   #100;
   CHECK_UTC(1'b0);  CHECK_DTC(1'b0);
   #300 btnD=1'b1;
   #200;
   CHECK_UTC(1'b0);  CHECK_DTC(1'b1);
   #200; btnD=1'b0;
   #200; btnD=1'b1;
   #100;
     CHECK_UTC(1'b1); CHECK_DTC(1'b0);
     #200; btnD=1'b0;
   #200; btnD=1'b1;
   #100;
   CHECK_UTC(1'b0); CHECK_DTC(1'b0);
   #200; btnU=1'b0;
   #200; btnU=1'b1;
   #100;
   CHECK_UTC(1'b1); CHECK_DTC(1'b0);
   #200; btnU=1'b0;
   #200; btnU=1'b1;
   #100;
   CHECK_UTC(1'b0); CHECK_DTC(1'b1);
   #100;
   end

   task CHECK_UTC;
   input good_TC;

   #0 begin
        if (good_TC !== UTC) begin
          TX_ERROR = TX_ERROR + 1;
        end
      end
  endtask

   task CHECK_DTC;
   input good_TC;

   #0 begin
        if (good_TC !== DTC) begin
          TX_ERROR = TX_ERROR + 1;
        end
      end
  endtask
endmodule
```

```
module show_7segDisplay (
 input [6:0] seg,
 input dp,
 input [3:0] an,
 output reg [7:0] D7Seg0, D7Seg1, D7Seg2,D7Seg3);

 reg [7:0] val;

 wire AN0, AN1, AN2, AN3;
 assign AN0=an[0];
 assign AN1=an[1];
 assign AN2=an[2];
 assign AN3=an[3];

  always @(AN0 or val)
  begin
    if (AN0 == 0) D7Seg0 <= val;
    else if (AN0 == 1) D7Seg0 <= " ";
    else D7Seg0 <= 8'bX;   //  non-blocking assignment
  end

  always @(AN1 or val)
  begin
    if (AN1 == 0) D7Seg1 <= val;
    else if (AN1 == 1) D7Seg1 <= " ";
    else D7Seg1 <= 8'bX;   //  non-blocking assignment
  end

  always @(AN2 or val)
  begin
    if (AN2 == 0) D7Seg2 <= val;
    else if (AN2 == 1) D7Seg2 <= " ";
    else D7Seg2 <= 8'bX;   //  non-blocking assignment
  end

  always @(AN3 or val)
  begin
    if (AN3 == 0) D7Seg3 <= val;
    else if (AN3 == 1) D7Seg3 <= " ";
    else D7Seg3 <= 8'bX;   //  non-blocking assignment
  end

  always @(seg)
  case (seg)
```

```
  7'b0111111:
       val = "-";
  7'b1111111:
       val = " ";
  7'b1000000:
       val = "0";
  7'b1111001:
       val = "1";
  7'b0100100:
       val = "2";
  7'b0110000:
       val = "3";
  7'b0011001:
       val = "4";
  7'b0010010:
       val = "5";
  7'b0000010:
       val = "6";
  7'b1111000:
       val = "7";
  7'b0000000:
       val = "8";
  7'b0011000:
       val = "9";
  7'b0001000:
       val = "A";
  7'b0000011:
       val = "B";
  7'b1000110:
       val = "C";
  7'b0100001:
       val = "D";
  7'b0000110:
       val = "E";
  7'b0001110:
       val = "F";
  default:
       val = 8'bX;
  endcase
endmodule
```