```verilog
`timescale 1ns / 1ps
/****************************************************************************
 * File Name: UART.v
 * Project: UART
 * Designer: Marc Dominic Cabote
 * Email: marcdominic011@gmail.com
 * Rev. Date: 5 May, 2018
 *
 * Purpose:This project is intedend for introducing Universal Asynchronous
 *         Receiver Transmitter (UART)--with both the transmit and the receive
 *         part. At the heart of the Tx and Rx Engines are shift regusters
 *         that shifts data in and out one bit at a time. This project should
 *         be able to do the requirements asked.
 *         1) walk the LEDS
 *         2) display a banner
 *         3) display what key has been pressed
 *         4) process characters accordingly
 *         5) Limit the user to 40 characters
 *
 * Notes:   -  This module has an asynchronous reset input.
 *
 ****************************************************************************/


module UART(  input clk, rst, Rx, eight, pen, ohel,
              input [3:0] baud,
             output [15:0] reads, writes,
             output reg [15:0] leds,
             output Tx);

      wire load, clear; //load and clear wire
      wire ovf, ferr, perr, data_status_sel; //Rx Status
      wire rst_out; //aiso wire
      wire TxRdy, RxRdy;   //TxRx wires
      wire TxPulse, RxPulse; //ped out wires
      wire interrupt_ack, write_strobe, read_strobe, UART_int;//TB wires
      wire [7:0] data, Rx_status, status;
      wire [15:0] port_id, in_port;
      wire [15:0] out_port;
      wire [18:0] k;


   //====================================================================
   // AISO
   //====================================================================
      aiso           aiso (.clk(clk),
                           .rst(rst),
                           .rst_out(rst_out));

   //====================================================================
   // Baud Decoder
   //====================================================================
      baud_decoder  baudrt(.baud(baud),
                           .k(k));

   //====================================================================
   // Transmit Engine
```

```
    //==================================================================
       transmitEngine   tx (.clk(clk),
                            .rst(rst_out),
                            .load(load),
                            .eight(eight),
                            .pen(pen),
                            .ohel(ohel),
                            .out_port(out_port[7:0]),
                            .k(k),
                            .TxRdy(TxRdy),
                            .Tx(Tx));

    //==================================================================
    // Receive Engine
    //==================================================================
       receiveEngine    rx (.clk(clk),
                            .rst(rst_out),
                            .Rx(Rx),
                            .eight(eight),
                            .pen(pen),
                            .reads0(clear),
                            .even(~ohel),
                            .k(k),
                            .RxRdy(RxRdy),
                            .perr(perr),
                            .ferr(ferr),
                            .ovf(ovf),
                            .data(data));

    //==================================================================
    // Transmit PED
    //==================================================================
       ped            txPED(.clk(clk),
                            .rst(rst_out),
                            .signal(TxRdy),
                            .pulse(TxPulse));

    //==================================================================
    // Receive PED
    //==================================================================
       ped            rxPED(.clk(clk),
                            .rst(rst_out),
                            .signal(RxRdy),
                            .pulse(RxPulse));
    //==================================================================
    // Data/Status Mux
    //==================================================================
       Data_Status dat_stat(.select(data_status_sel),
                            .data(data),
                            .status(status),
                            .in_port(in_port));
    //==================================================================
    // RS flop to TB
    //==================================================================
       srflop       srflop (.clk(clk),
                            .rst(rst_out),
                            .s(UART_int),
                            .r(interrupt_ack),
                            .srOut(sr_out));
   //====================================================================
   // TramelBlaze
   //====================================================================
      tramelblaze_top   TB(.CLK(clk),
                            .RESET(rst_out),
                            .IN_PORT(in_port),
                            .INTERRUPT(sr_out),
                            .OUT_PORT(out_port),
                            .PORT_ID(port_id),
                            .READ_STROBE(read_strobe),
                            .WRITE_STROBE(write_strobe),
                            .INTERRUPT_ACK(interrupt_ack));

      // assign load
      assign load = (port_id == 16'h0000) & (write_strobe);
      // assign clear
      assign clear = (port_id == 16'h0000) & (read_strobe);
      // Data-Status Select
      assign data_status_sel = (port_id == 16'h0001) ? 1'b1 : 1'b0;
      // assign status
      assign status = {3'b0, ovf, ferr, perr, TxRdy, RxRdy};
      // generate uart interrupt from 2 PEDs
      assign UART_int = TxPulse | RxPulse;
      // leds for debugging
      always @(posedge clk, posedge rst_out) begin
         if (rst_out)
            leds <= 16'h0;
         else if (port_id == 16'h0002 && write_strobe)
            leds <= out_port;
      end

endmodule

```