```
//*****************************************************************************
//  Test of SystemC Fixpoint Data Type
//
//  Licensing:
//    This code is distributed under GNU LGPL license.
//
//  Modified:
//    2012.06.30
//
//  Author:
//    Based on SCLive 3.0 and www.asic-world.com example codes
//
//    Modifications by Young W. Lim
//
//
//*****************************************************************************


#define SC_INCLUDE_FX

#include <systemc.h>

SC_MODULE (first_counter) {
  sc_in_clk     clock;
  sc_in<bool>   reset;
  sc_in<bool>   enable;
  sc_out<sc_fixed<5,4> >        counter_out;

  sc_fixed<5,4>         count;

  void incr_count() {
    while (true) {
      wait();
      if (reset.read() == 1) {
        count = 0;
        counter_out.write(count);
      } else if (enable.read() == 1) {
        count = count + 1;
        counter_out.write(count);
      }
    }
  }


  void print_count() {
    while (true) {
      wait();
      cout << "@" << sc_time_stamp() <<
        "::Counter Value " << counter_out.read() << endl;
    }
  }


  SC_CTOR(first_counter) {
    SC_THREAD(incr_count);
    sensitive << clock.pos();

    SC_THREAD(print_count);
    sensitive << counter_out;
  }


};


class testbench: public sc_module {
public:
  sc_out<bool> clk;
  sc_out<bool> reset;
  sc_out<bool> count;
```

```
  SC_HAS_PROCESS(testbench);
  testbench(sc_module_name nm): sc_module(nm) {
    SC_THREAD(clk_gen);
    SC_THREAD(stimuli);
  }
  void clk_gen() {
    while(true) {
      clk.write(true);
      wait(10, SC_NS);
      clk.write(false);
      wait(10, SC_NS);
    }
  }

  void stimuli() {
    while(true) {
      reset.write(true);
      count.write(false);
      wait(10, SC_NS);
      reset.write(false);
      wait(50, SC_NS);
      count.write(true);
      wait(200, SC_NS);
    }
  }

};



// Top level
class top: public sc_module {
public:
  sc_signal<bool> clk_sig, count_sig, reset_sig;
  sc_signal<sc_fixed<5,4> > q_sig;

  first_counter uut ;
  testbench tb ;


  top(sc_module_name nm): sc_module(nm), uut("uut"), tb("tb") {
    tb.clk(clk_sig);
    tb.reset(reset_sig);
    tb.count(count_sig);
    uut.clock(clk_sig);
    uut.reset(reset_sig);
    uut.enable(count_sig);
    uut.counter_out(q_sig);
  }
};


int sc_main(int argc, char * argv[]) {
  sc_set_time_resolution(1, SC_NS);
  top verif_env("verif_env");

  sc_trace_file *tf;
  tf = sc_create_vcd_trace_file("trace");
  sc_trace(tf, verif_env.clk_sig, "clk_sig");
  sc_trace(tf, verif_env.reset_sig, "reset_sig");
  sc_trace(tf, verif_env.count_sig, "count_sig");
  sc_trace(tf, verif_env.q_sig, "q_sig");

  sc_start(1000, SC_NS);
  sc_close_vcd_trace_file(tf);
  return (0);


}
```