```
/* -*- c++ -*- */
/*
 * Copyright 2015 <F. Stockmayer>
 */

#ifdef HAVE_CONFIG_H
#include "config.h"
#endif

#include <gnuradio/io_signature.h>
#include "crc8_framer_impl.h"

#define POLYNOM 0x0700

namespace gr {
  namespace mymod {

    crc8_framer::sptr
    crc8_framer::make()
    {
      return gnuradio::get_initial_sptr
        (new crc8_framer_impl());
    }

    /*
     * The private constructor
     */
    crc8_framer_impl::crc8_framer_impl()
      : gr::sync_interpolator("crc8_framer",
              gr::io_signature::make(1, 1, sizeof(char)),
              gr::io_signature::make(1, 1, sizeof(char)), 2)
    {
        init_crc_tab();
    }

    /*
     * Our virtual destructor.
     */
    crc8_framer_impl::~crc8_framer_impl()
    {
    }

    int
    crc8_framer_impl::work(int noutput_items,
                           gr_vector_const_void_star &input_items,
                           gr_vector_void_star &output_items)
    {
        const char *in = (const char *) input_items[0];
        char *out = (char *) output_items[0];

        // Do <+signal processing+>
        for(int i=0; i < noutput_items/2; i++){
          d_data = in[i];

          out[2*i] = d_data;
          out[2*i+1] = crc_tab[d_data];

        }

        // Tell runtime system how many output items we produced.
        return 2*(noutput_items/2);
    }
```

```
    // Calculate CRC8 for all possible 8-Bit Data
    void
    crc8_framer_impl::init_crc_tab(void){
        int i, j, k;
        unsigned short crc;

        for (i=0; i<256; i++){
                crc = (unsigned short) i;
                // Process 8 Bit Data plus 8 zero filled crc-bits
                for (j=0; j<16; j++) {
                        if ( crc & 0x8000 ) crc = ( crc << 1 ) ^ POLYNOM;
                        else crc = crc << 1;
                }
                crc_tab[i] = (crc >> 8) & 0xff;
        }
    }

  } /* namespace mymod */
} /* namespace gr */
```