basic queries

Contents

1 Introduction

What: Sending receiving packets.

This library is a part of the base software made by David Decotigny still available in the CVS's *svn/calice/online-sw/trunk/tests_eth_dif/* directory. It is now copied at:

- *libLDA/include/basic/*
- *libLDA/libcalice_basic.a/*
- *libLDA/basic/*

This library perform basic queries as "pyserdial" do.

TODO: Implement all the basic queries like for example the ping LDA so as to be able to display the firmware version.

2 API

2.1 LDA_interface

```
class LDAError : public std::runtime_error
{
public:
  LDAError(std::string const& msg) : std::runtime_error(msg) { }
  virtual ~LDAError() throw() { }
};

class LDA_interface
{
public:
  LDA_interface(std::string const& host_netdev,
struct ether_addr const& lda_mac_address) /* Throw */;
  virtual ~LDA_interface();

  inline void set_debug(bool dbg = true) {}
  inline bool get_debug() const          {}
  void configure_DIF_reply_destination(int timeout_ms = 3*1000);
  void prepare_send_headers(CALDIF::PktAccessor_LDA_pkt & pkt);
  void send_verbatim(CALDIF::PktAccessor_LDA_pkt const& pkt) /* Throw */;
  void send_tcpdump_magic(unsigned eth_type = 0x8888, char magic = 0x42);
```

```
  inline void set_intersend_pause(unsigned pause_ms = 1) {}
  inline unsigned get_intersend_pause() /* millisecond */ const {}
  void send(CALDIF::PktAccessor_LDA_pkt & pkt) /* Throw */;
  CALDIF::PktAccessor_LDA_pkt const& recv(int timeout_ms = -1) /* Throw */;
  inline unsigned hexdump_consume_recv_queue(std::ostream & os,
     unsigned timeout_ms = 0) {}
  inline unsigned flush_recv_queue(unsigned timeout_ms = 0) {}
  int dispatch(int timeout_ms, pcap_handler callbackFunction, u_char *data);

  bool                        _debug;
  static const unsigned       MAX_ETH_FRAME_SIZE;

protected:
  bool _wait_for_packet(unsigned timeout_ms) const;
  void _recv(int timeout_ms = -1, std::ostream * dbg_strm = NULL) /* Throw */;
  unsigned _consume_recv_queue(std::ostream * os, unsigned timeout_ms);

  pcap_t                    * _pcap;
  struct bpf_program          _pcap_filter;
  struct ether_addr           _sendto_dst;
  struct ether_addr           _sendto_src;
  unsigned                    _intersend_pause_ms;
  int                         _fd;
  CALDIF::PktAccessor_LDA_pkt _recv_access;
  unsigned                    _cur_pkt_id;
};
```

2.2 DIF_tools

```
class DIF_tools
{
public:
  DIF_tools(LDA_interface & lda,
    uint16_t lda_outlink_id = LDA_DIFLINK_ID_BROADCAST,
    uint8_t  dcc_nibble     = PKT_DCC_NIBBLE_BROADCAST);
  virtual ~DIF_tools();

  inline void set_debug(bool dbg = true) {}
  inline bool get_debug() const          {}
  inline void resync_lda_headers() {}
  inline LDA_interface & get_LDA() {}
  inline uint16_t get_lda_outlink_id() const {}
  inline uint8_t  get_dcc_nibble() const     {}
  inline void _SET_lda_outlink_id(uint16_t l) {}
  inline void _SET_dcc_nibble(uint8_t nb)     {}
  inline void send_FC_DCC_reset() /*throw*/ {}
  void send_FC_DCC_init_links(); /* throw */
  void send_DIF_fast_command(unsigned char comma,
     unsigned char data) /*throw*/;
  inline void send_FC_DIF_reset() /*throw*/ {}
  void send_BT_DIF_write_debug_fifo(uint16_t const* words,
    size_t nwords,
    unsigned pkt_id = 0x1234);/* throw */
  void send_FC_DIF_read_debug_fifo(); /* throw */
  void send_BT_DIF_read_debug_fifo(unsigned pkt_id = 0x60be); /* throw */
  void send_BT_DIF_config_rndgen(uint16_t rndpkt_size = 0x40,
 uint16_t num_rndpkts = 7,
 bool en = true,
```

```
   uint16_t inter_rndpkt_gap = 0x0); /* throw */
    void send_BT_DIF_cmd_register(unsigned pkt_id); /* throw */
    void send_BT_DIF_command(unsigned pkt_id,
     uint16_t command,
     uint16_t data); /* throw */
    void test_LAPP(unsigned pkt_id); /* throw */
    void test_LAPP2(unsigned pkt_id, char *path); /* throw */
    void send_BT_DIF_readback_memory(unsigned pkt_id); /* throw */
    void hexdump_BT_DIF_read_bank(std::ostream & os,
  uint16_t select_id = 0x01); /* throw */
  protected:
    void _reset_send_access();

    LDA_interface             & _lda;
    CALDIF::PktAccessor_LDA_pkt _send_access;
    uint16_t                    _lda_outlink_id;
    uint8_t                     _dcc_nibble;
  };
```

3 Compilation

3.1 Flags

There is 3 important pre-compilation flags:

- DEBUG_BASIC_DIF: make this module sending more debugging messages.
  Note: This will slow the DAQ acquisition and may imply data lost.
- CCC_LINK: make the libLDA compatible with both SERIAL and DHCAL.
- PCAP_DISPATCH_CNT: maximum number of packets to read from pcap without thread interruption. (reactivity vs task force).

/include/basic/all.hh:

```
/* Debug */
#define DEBUG_BASIC_DIF 0

/* CCC links */
#define CCC_LINK_TCP        0 // default version
#define CCC_LINK_SERIAL     1 // for usb/serial setup
#define CCC_LINK_TCP_ADDR   "192.168.1.20"
#define CCC_LINK_TCP_PORT   "2300"
#define CCC_LINK_SERIAL_TTY "/dev/ttyUSB0"

/* CCC link we use */
#define CCC_LINK CCC_LINK_TCP

/* lib PCAP
   - pcap_dispatch() processes packets until cnt packets are
   processed
   - pcap_set_buffer_size() sets the buffer size that will be
   used on a capture handle which is in units of bytes.
*/
#define PCAP_DISPATCH_CNT 1000
#define LLR_BASIC_LDA_PCAP_SNAP_LEN (1024+13) // do not change this
```

3.2 Ubuntu

```
$ sudo aptitude install flex bison libpcap-dev libboost-thread1.40-dev!
$ make
```

3.3 SLC 5.5

```
# yum install gcc-c++ boost-devel boost-doc
  Installing     : libstdc++-devel
  Installing     : kernel-headers
  Installing     : glibc-headers
  Installing     : glibc-devel
  Installing     : gcc
  Installing     : gcc-c++
  Installing     : libicu
  Installing     : boost
  Installing     : boost-devel

# cd libLDA/basic
# make
```