Configuration

Contents

1 Introduction

What: loading configuration from different medias and store it into arrays.
This library is located at:

- *libLDA/include/conf/*
- *libLDA/conf/*

TODO: Implement DB access.
There is 3 way to load configuration. All will put configuration into the same arrays use next by the other modules.

- From configuration file.
- From static configuration that will be replace by BD access.
- From command line.

Here is the command line options for instance:

```
$ conf/utCommandLine poldhcp54.conf -h
Usage: conf/utCommandLine [options] CFG_FILE

Options:
  -h | --help                           this cruft
  -c CONF | --conf-source=CONF          specified the way to load configuration
                                        (default file): CONF among file, static, bd
  -p ETH_PORT | --eth-port=ETH_PORT     override the ethernet port on the host
                                        specified in the config file (eth0, ...)
  -m MAC_ADDR | --lda-address=MAC_ADDR override the ethernet mac address
                                        specified in the config file (in
                                        aa:bb:cc:dd:ee:ff format)
  -v LEVEL | --debug-print=LEVEL        set stdout logging level (default: INFO)
                                        among DEBUG, INFO, WARNING, ERROR, NONE
  -f --syslog                           using syslof instead of cerr
  -n | --no-reinit                      don't try to reinit DCC and DIF hardware
  -s | --single                         do not enter into infinite loop mode
  -i | --pcap-buffsize                  set PCAP input buffer size (unit is 1Ko packet)
  -t | --trig-buffsize                  set the read-out trigger buffer size (unit is 1 trigger)
  -d | --data-buffsize                  set the read-out data buffer size (unit is 2 bytes)
  -S | --freq-slc                       set slow control frequency (unit is Hertz)
  -T | --freq-slc                       set trigger frequency (unit is Hertz)
  -U | --freq-slc                       set user read out frequency (unit is Hertz)
```

2 API

Several configuration media classes derive from the `conf` abstract class. A derived class is build depending on the command line parameters.

2.1 DIF Id

The `DIFid` class allow to identify each DIF and is use to map them into arrays.

```
  class DIFid
  {
  public:
    static const unsigned INVALID   = 0;
    static const unsigned BROADCAST = ~0;
    explicit DIFid(unsigned lda_diflink_id /* base 1 */ = INVALID);
    explicit DIFid(unsigned lda_dcclink_id /* base 1 */,
   unsigned dcc_diflink_id /* base 1 */);
    DIFid(DIFid const& other);
    DIFid const& operator= (DIFid const& other);
    ~DIFid();

    int cmp(DIFid const& other) const;
    inline bool operator== (DIFid const& other) const;
    inline bool operator!= (DIFid const& other) const;
    inline bool operator< (DIFid const& other) const;
    inline bool operator> (DIFid const& other) const;
    inline bool operator<= (DIFid const& other) const;
    inline bool operator>= (DIFid const& other) const;

    inline void set(unsigned lda_diflink_id /* base 1 */ = INVALID);
    inline void set(unsigned lda_dcclink_id /* base 1 */,
    unsigned dcc_diflink_id /* base 1 */);

    inline unsigned get_lda_port_id() const { return this->_lda_difport_id; }
    inline unsigned get_dcc_port_id() const { return this->_dcc_difport_id; }

  protected:
    unsigned _lda_difport_id;
    unsigned _dcc_difport_id;
  };
};
std::ostream & operator<< (std::ostream & os,
   LLR_CALICE_CONF::DIFid const& dif_id);
```

2.2 Conf

The `conf` abstract class define methods to insert configuration into arrays.

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

  class Conf
  {
  public:
```

```
  Conf();

  typedef boost::variant<std::string, int, bool> config_parameter_t;
  typedef std::map<std::string, config_parameter_t> config_parameters_t;
  typedef std::map<DIFid, boost::shared_ptr<config_parameters_t>>
          dif_parameters_t;

  template <typename T>
  T const& get_config_parameter(std::string const& key,
 config_parameter_t const& dflt_val) const; /* throw */

  template <typename T>
  T const& get_config_parameter(DIFid const& for_dif,
 std::string const& key,
 config_parameter_t const& dflt_val) const;
  void serialize();

  bool _set_cfgparm(std::string const& key, config_parameter_t const& val);
  bool _set_cfgparm(DIFid const& for_dif,
    std::string const& key, config_parameter_t const& val);

  config_parameters_t               _global_config_parameters;
  dif_parameters_t                  _per_dif_config_parameters;

protected:
  config_parameter_t const&
  _get_config_parameter(std::string const& key,
config_parameter_t const& dflt_val) const;
  config_parameter_t const&
  _get_config_parameter(LLR_CALICE_TESTS::DIFid const& for_dif,
std::string const& key,
config_parameter_t const& dflt_val) const;
};
```

2.3 ConfHard

The `ConfHard` class provide a simple example explaining how to write into the arrays.

```
ConfHard::ConfHard()
{
  bool rc = true;
  //struct ether_addr ethaddr;
  DIFid difId;

  /* LDA */
  //this->_lda_ethernet_port = std::string("lo"); //"eth2");
  //parse_mac(ethaddr, "5e:70:0c:d2:79:68");
  //this->_lda_ethernet_address = ethaddr;
  rc=rc&& this->_set_cfgparm("lda_ethernet_port",
     (std::string)"lo");
  rc=rc&& this->_set_cfgparm("lda_ethernet_addr",
     (std::string)"00:00:00:00:00:00");

  /* Global */
  rc=rc&& this->_set_cfgparm("bypass", false);
  rc=rc&& this->_set_cfgparm("rng_packets_num", 511);
  rc=rc&& this->_set_cfgparm("rng_packets_words", 500);
  rc=rc&& this->_set_cfgparm("rng_packets_gap", 2700);
```

```
  /* DIF */
  difId = DIFid(1);
  rc=rc&& this->_set_cfgparm(difId, "bypass", true);
  rc=rc&& this->_set_cfgparm(difId, "rng_packets_num", 511);
  rc=rc&& this->_set_cfgparm(difId, "rng_packets_words", 500);
  rc=rc&& this->_set_cfgparm(difId, "rng_packets_gap", 2700);

  /* DIF */
  difId = DIFid(10, 1);
  rc=rc&& this->_set_cfgparm(difId, "bypass", false);
  rc=rc&& this->_set_cfgparm(difId, "rng_packets_num", 511);
  rc=rc&& this->_set_cfgparm(difId, "rng_packets_words", 500);
  rc=rc&& this->_set_cfgparm(difId, "rng_packets_gap", 2700);

  if (!rc) {
    throw KeyError("Bad parametee entrie: cannot build conf");
  }

  LLR_LOGGER_INFO("ConfHard configuration written into arrays");
}
```

3 Compilation

nothing special.