High level interface

Contents

1 Introduction **1**

2 API **1**
2.1 Run 1
2.2 What do it do 2

3 Compilation **3**
3.1 Flags 3

4 Unitary tests **4**

1 Introduction

What: Emplementing tests and the final driver.
This library is located at:

- *libLDA/include/run/*
- *libLDA/run/*

2 API

2.1 Run

This class allow to manage 2 thread:

- one thread deal with pcap receiving, transforming and storing packets
- the main thread permit to XDAQ application to read-out the stored data.

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

class Run
{
public:
  Run();
  Run(LLR_CALICE_CONF::Conf* conf, int prio1, int prio2); /* Throw */
  virtual ~Run();

  /* API */
  virtual bool configure();
  virtual void display();
  virtual void acquire();     /* embedded into thread */
  void start();
  void stop();
```

```
    LLR_CALICE_CONF::Conf* conf;
    std::string me;
    Lda lda;

  protected:
    int mainThreadFifoPriority;
    int recvThreadFifoPriority;
    Thread* recvThread;
    int recv_timeout_ms;
  };
```

2.2 What do it do

```
int main (int argc, char * argv[])
{
  LLR_CALICE_CONF::Conf *conf =
    LLR_CALICE_CONF::parseCommandLine(argc, argv);

  Run run(conf, 2, 1);
  int i=0;

  run.conf->serialize();
  //run.configure();

  if (!run.conf->get_config_parameter<bool>("infloop", true)) {
    // Single mode
    run.acquire();
    std::cout << std::endl;
  }
  else {
    // Infinite loop mode
    while (running) {
      std::cout << i++ << ": Start ";
      fflush(stdout);
      run.start();
      sleep(1);

      run.stop();
      std::cout << " Stop." << std::endl;
      //run.display();
    }
    std::cout << "Exiting..." << std::endl;
  }
}
```

Run the unit test:

```
# run/utRun poldhcp54.conf -f
Actual configuration is:
...

0: Start ..... Stop.
1: Start ..... Stop.
2: Start ..... Stop.
3: Start ..... Stop.
4: Start ..... Stop.
5: Start ..... Stop.
6: Start ..... Stop.
7: Start ..... Stop.
```

```
8: Start ........ Stop.        <-- ^C pressed
Exiting...

*** Total : miss 0 / 0 Pkts = nan%
 DIF [ 1,5] : 0 received + 0 missing 0 pkts and 0 errors.
 Dropped pkts:    0
 Unexpected pkts: 0
```

3 Compilation

3.1 Flags

Pre-compilation flags:

- Debug massage's flags: make this module sending more debugging messages. **Note:** This will slow the DAQ acquisition and may imply data lost.
- Fifo priorities: all thread's fifo priorities.
- Default frequencies in Hertz: defaults thread's frequencies.
- Default buffers size: defaults size for DIF's readout buffers.
- DIF_READOUT_TIMEOUT: time before sending incomplet trigger events.
- RUN_LOGGER_DEBUG: allow/desallow debuging messages.

/include/run/all.hh:

```
/* Debug messages enabled */
#define NOTHREAD_RUN   0
#define DEBUG_LDA      0
#define DEBUG_DIF      0

#define DEBUG_RUN      0
#define DEBUG_FLUSH    0
#define DEBUG_CONFIG   0
#define DISPLAY_INPUT  0
#define DEBUG_DRIVER   0
#define DEBUG_BUFFER   0

/* Fifo priorities */
#define FIFO_PRIO_TEST_DAQ   1
#define FIFO_PRIO_DUMP_DAQ   2
#define FIFO_PRIO_TEST_SLC   3
#define FIFO_PRIO_DUMP_SLC   4
#define FIFO_PRIO_CALICE_USR 5
#define FIFO_PRIO_CALICE_DAQ 6
#define FIFO_PRIO_CALICE_SLC 7
#define FIFO_PRIO_CALICE_TRG 8
#define FIFO_PRIO_MAX       99

/* Default frequencies in Hertz */
#define FREQ_TRG 50.   // Hz
#define FREQ_USR 2.    // Hz
#define FREQ_SLC 1.    // Hz

/* Default buffers size */
#define DEFAULT_PCAP_BUFFSIZE 500    // packets
```

```
#define DEFAULT_TRIG_BUFFSIZE 100    // triggers
#define DEFAULT_DATA_BUFFSIZE 300000 // 15000-16000? words

/* DIF read-out timeout */
#define DIF_READOUT_TIMEOUT 100000 // micro seconds

#if 1
#define RUN_LOGGER_DEBUG(expr)   LLR_LOGGER(LOG_DEBUG, this->me << " DEBUG: " << expr)
#else // optimisation: no debug message function call
#define RUN_LOGGER_DEBUG(expr)
#endif
```

4 Unitary tests

Here we can test the readout consistancy of binary output of the libLDA (see the /tools/udDevice and *tools/utDaq* unit tests).

```
$ tail -fn 0 data/dif-4-9.txt | src/run/calDump
0000:   b0
0001:   8d 00000001 00000001 00000001 000000003257 003257
0017:   b4
0018:   03 00031f 0000 0000 0000 0000 0000 0000 0001 0000
002c:   03 000119 0000 0000 0000 0000 0000 0000 0001 0000
0040:   03 000124 0000 0000 0000 0000 0000 0000 0001 0000
...
```