Implementation

Contents

1 Introduction

An exemple is provided to help using the libLDA: *libLDA/exemple.cc*.

2 Installing, compiling and testing libLDA

The installation and compilation process is describe in the *README* file:

```
****************
Install on SLC55
****************

# yum install subversion gcc-c++ boost-devel
$ cd /home/calice
$ svn co svn+ssh://svn-calice/calice/online-sw/trunk/libLDA
# tar -zxf libLDA/usr/externalLibs.tgz -C /usr/local
# cd /usr/local/stow
# /home/calice/libLDA/usr/bin/stow -v flex-2.5.35
# /home/calice/libLDA/usr/bin/stow -v bison-2.4
# /home/calice/libLDA/usr/bin/stow -v libpcap-1.1.1
# cat >> /etc/ld.so.conf
/usr/local/lib
^D
# /sbin/ldconfig

Please check PATH provides access to /usr/local/bin.

***********
Compilation
***********

$ cd libLDA
$ make

*******
Exemple
*******
```

```
$ emacs exemple.cc
$ g++ -c exemple.cc -I/home/calice/libLDA/include -o exemple.o
$ g++ exemple.o -lpcap -lpthread -Llib -lLDA -o exemple
# ./exemple
```

3 API

3.1 Configuration

The best way to improve the loading configuration is to add new class that implement a new inheritance of the *conf/conf.hh* virtual class.

However, you can first deal with this exemple code:

```
...
/*
 * When fixed, this should be done by
 * LLR_CALICE_CONF::parseCommandLine(argc, argv);
 */
LLR_CALICE_CONF::Conf* loadConfiguration()
{
  static LLR_CALICE_CONF::Conf conf;
  LLR_CALICE_CONF::DIFid difId;

  /* Add a direct DIF */
  //difId = LLR_CALICE_CONF::DIFid(2);
  //conf._set_cfgparm(difId, "bypass", false);

  /* Add a DIF via DCC */
  difId = LLR_CALICE_CONF::DIFid(1, 8);
  conf._set_cfgparm(difId, "bypass", false);
  conf._set_cfgparm(difId, "drv_inpath_asu",
    (std::string)"data/sc_HR2_24asic_3dac_a_1000.bak");

  /* Ethernet link */
  conf._set_cfgparm("lda_ethernet_port", (std::string)"eth0");
  conf._set_cfgparm("lda_ethernet_addr", (std::string)"5e:70:0c:d2:5c:d6");

  /* Global stuffs */
  conf._set_cfgparm("reinit", true);
  conf._set_cfgparm("infloop", true);
  conf._set_cfgparm("pcap_buffer_size", 1000);
  conf._set_cfgparm("trig_buffer_size", 20);
  conf._set_cfgparm("data_buffer_size", 6000);
  conf._set_cfgparm("freq_slc", (double) 1);
  conf._set_cfgparm("freq_usr", (double) 10);
  conf._set_cfgparm("freq_trg", (double) 100);
  conf.preConfigure();
  return &conf;
}
...
int main (int argc, char * argv[])
{
  LLR_CALICE_CONF::Conf* conf;

  conf = loadConfiguration();
  logSeverity = getLogSeverity("DEBUG"); // INFO
  logFacility = getLogFacility((char*)"local4"); // -1
  logInit();
```

```
  run.configure();
...
}
```

3.2 Read-out callback function

User must provide a callback function to get the all data available at the end of a trigger processing. The callback function will be call for each different DIF and may be called twice for each DIF if we encounter the end of the ring buffer.

```
/*
 * This function may be called 2 times with the same
 * trigger informations, but with 2 consecutive data
 * packets:
 *
 *                   size
 *               <----------->
 *  +-----------+-------------+
 *  |[t1    ][t2|    ][t3    ]|
 *  +-----------+-------------+
 *              ^    ^
 *             index i
 */

static void
readCallback(LLR_CALICE_CONF::DIFid difId,
     LLR_CALICE_RUN::TriggerInfo triggers, uint nbTriggers,
     uint index, uint16_t *data, uint size)
{
  static char buffer[409600] = "\0";   // printf -> stdout conversion
  uint8_t *readOut = (uint8_t*)data; // the libLDA only manage words
  files_it fileIt;
  std::ofstream *file;
  uint t, i, j, k, state;
  uint offsetT=0, offsetD=0;  // in bytes
  uint sizeT=0,   sizeD=size; // in words

  fileIt = files.find(difId);
  file = fileIt->second;

  *file << "<<<\n" << difId;
  *file <<"read callback "<<(index?"2nd time (":"(") <<size<<"):\n";

  /* find the current trigger (if index >0) */
  for (i=0, t=0;
       i<index && t<nbTriggers;
       i += triggers[t].readOutSize, ++t);

  /* compute trigger size and offset if needed */
  if (i>index) {
    /* current trigger was truncated at previous call */
    --t;
    sizeT = i-index;
    offsetT = (triggers[t].readOutSize - sizeT)*2;
  }
  else {
    sizeT = triggers[t].readOutSize;
    offsetT = 0; // byte offset into trigger data
```

```
  }

  /* while triggers remains in the data */
  while (offsetD < sizeD*2-1) {

    /* last trigger's data may be truncated */
    if (offsetD + sizeT*2 > sizeD*2)
      sizeT = sizeD - offsetD/2;

    /* print current trigger */
    sprintf(buffer, "read out%s trigger %x (%i bytes: [0x%x-0x%x])\n",
    (sizeT < triggers[t].readOutSize)?" part of":"",
    triggers[t].difTriggerCounter,
    sizeT*2, offsetT, offsetT+sizeT*2-1);
    *file << difId << buffer;

    /* print current trigger's data */
    sprintf(buffer, "%04x:\t", offsetT);
    for(i=0, j=0, k=0; i < sizeT*2; ++i, ++j)
      {
// print data byte
sprintf(buffer+strlen(buffer), "%02x", readOut[offsetD+i]);

// new word
if (j%2) {sprintf(buffer+strlen(buffer), " ");}

// end of chip record
if (readOut[offsetD+i] == 0xb4) {
  sprintf(buffer+strlen(buffer), "\n%04x:\t", offsetT+i);
}

// new line for each trigger
if (readOut[offsetD+i] == 0xb0)
  state = 0;
if (readOut[offsetD+i] == 0xb4) {
  state = 1;
  k=0;
}
else {
  if (state == 1){
    ++k;
    if (k == 20) {
      sprintf(buffer+strlen(buffer), "\n%04x:\t", offsetT+i);
      k=0;
    }
  }
}
      }

    sprintf(buffer+strlen(buffer),"\n\n");
    *file << buffer;

    // next trigger
    offsetD += sizeT*2;
    ++t;
    sizeT = triggers[t].readOutSize;
    offsetT=0; // reinit offset
  }
```

```
  *file << ">>>\n";
}

int main (int argc, char * argv[])
{
  LLR_CALICE_CONF::Conf* conf;
...
  LLR_CALICE_RUN::Driver run(conf, 4, 3);
  run.configure();

  run.start();            // acquire is done by a slave thread
  while(running){
    sleep(1);
    run.dump(readCallback);
  }
  run.stop();
  return 0;
}
```

4 Configuring Syslog

The logged messages are store using the SYSLOG daemon:

```
$ tail -f /var/log/messages
```

4.1 Redirection & modifying severity

Add a line in */etc/syslog* that get the `local4` facility use by C programs for logging.

```
local4.info    /var/log/libLDA.log
```

The `info` severity should be replaced by `debug` if you want the logs to be more talkative, however, each binaries need to provide the `-v DEBUG` command line parameter or to change source code in the *libLDA/misc/log.cc* file:

```
int logFacility = -1; //getLogFacility((char*)"local4");
int logSeverity = getLogSeverity((char*)"INFO");
```

Now you should monitor what happend with:

```
# /etc/init.d/syslog reload
                                          $ tail -f /var/log/libLDA.log
$ logger -p local4.info "hello"
                                          Apr 29 05:40:35 poldhcp54 calice: hello
```

4.2 Rotation

You may add a file rotation system by adding the */etc/logrotate.d/libLDA* file:

```
/var/log/libLDA.log {
  daily
  rotate 10
  size 32M
  compress
  copytruncate
  missingok
}
```

You can test the rotation by calling **logrotate**:

```
# /usr/sbin/logrotate -f /etc/logrotate.d/libLDA
$ ls /var/log/libLDA.log*
/var/log/libLDA.log
/var/log/libLDA.log.1.gz
```

logrotate is call by **cron** as defined in the */etc/cron.daily/logrotate* file