Unité Mixte de Recherche 7638 LLR - CNRS - X
LLR - École polytechnique
91128 PALAISEAU Cedex
France
Tél (33) 1 69 33 55 00
Fax (33) 1 69 33 55 08
http://llr.in2p3.fr

CALICE PROJECT

Online software documentation

Nicolas Roche

October 29, 2015

CALICE PROJECT

Online software documentation

Nicolas Roche

Laboratoire Leprince-Ringuet

October 29, 2015

Abstract

Online software documentation for Calice.
This documentation mostly describe tools used to connect the LLR electonics cards built for Calice experiment project. The goal reached is a unique library called “libLDA” wich handle evry kind of connected hardware and database. A lot of works remains, but I hope this library open a first door making us more aware of the weakness of the present configuration.

Keys words: Calice, software, documentation

Tutors: Vincent Boudry, Simon Chollet

Notes: Last version located at http://polype.in2p3.fr/ roche/

Contents

I CALICE's DAQ V2 software **5**

1 Voice of the customer **9**
1.1 Introduction . . . 9
1.2 Pcap Library . . . 9
1.2.1 Buffers . . . 9
1.2.2 threads . . . 9
1.3 Loading configuration from database . . . 11
1.4 Log using syslog daemon . . . 11
1.5 Non regression tests . . . 11
1.6 Packets formats . . . 11
1.6.1 Input . . . 11
1.6.2 Output . . . 12

2 Functional analysis **15**
2.1 Packets objects . . . 15
2.1.1 Introduction . . . 15
2.1.2 API . . . 16
2.1.3 Compilation . . . 18
2.1.4 Unitary tests . . . 19
2.2 Conf objects . . . 19
2.2.1 Introduction . . . 19
2.2.2 API . . . 20
2.2.3 Compilation . . . 22
2.3 Basic objects . . . 22
2.3.1 Introduction . . . 22
2.3.2 API . . . 23
2.3.3 Compilation . . . 24
2.4 Run objects . . . 25
2.4.1 Introduction . . . 25
2.4.2 API . . . 26
2.4.3 Compilation . . . 27
2.4.4 Unitary tests . . . 28
2.5 Misc modules . . . 29
2.5.1 Introduction . . . 29
2.5.2 API . . . 29
2.5.3 Compilation . . . 29
2.5.4 Unitary tests . . . 29

3 Technical specifications **31**
3.1 Introduction . . . 31
3.1.1 Architecture . . . 31
3.1.2 Compilation . . . 32
3.2 LDA . . . 33
3.2.1 LDA_version . . . 33
3.2.2 LDA_get_DIF_status . . . 33
3.2.3 LDA_en_restart_links . . . 34
3.2.4 LDA_send_fast_command . . . 34
3.2.5 LDA_dcm_relock . . . 35
3.3 DCC . . . 35
3.3.1 send_FC_DCC_reset . . . 35
3.3.2 send_FC_DCC_get_status . . . 35
3.3.3 send_FC_DCC_init_links . . . 36
3.3.4 send_BT_DCC_get_status . . . 36
3.3.5 send_BT_DCC_config_tx_rx . . . 36
3.3.6 send_BT_DCC_restart . . . 37
3.3.7 send_BT_DCC_start_RTT . . . 37
3.3.8 send_BT_DCC_stop_RTT . . . 37
3.3.9 send_BT_DCC_relock_DCM . . . 37
3.3.10 send_BT_DCC_register_blob . . . 37
3.3.11 send_MULTI_BT_DCC_get_status . . . 38
3.4 DIF . . . 38
3.4.1 send_FC_DIF_reset . . . 38
3.4.2 send_BT_DIF_register_state . . . 38
3.4.3 send_BT_DIF_command . . . 38
3.4.4 send_BT_DIF_write_debug_fifo . . . 39
3.4.5 send_FC_DIF_read_debug_fifo . . . 39
3.4.6 send_BT_DIF_read_debug_fifo . . . 39
3.4.7 send_BT_DIF_write_RAM . . . 39
3.4.8 send_BT_DIF_config_rndgen . . . 39
3.4.9 send_BT_DIF_read_bank . . . 40
3.4.10 send_BT_DIF_ignored . . . 40
3.4.11 send_FC_DIF_Guillaume . . . 40
3.4.12 send_BT_DIF_cmd_register . . . 40
3.5 Hight Level tests . . . 40
3.5.1 Configure links . . . 40
3.5.2 FIFO test . . . 41
3.5.3 RNG test . . . 41
3.5.4 Sending ASICs configuration . . . 42
3.5.5 Read out from ASICs . . . 42

4 Implementation **43**
4.1 Introduction . . . 43
4.2 Installing, compiling and testing libLDA . . . 43
4.3 API . . . 44
4.3.1 Configuration . . . 44
4.3.2 Read-out callback function . . . 45
4.4 Configuring Syslog . . . 47
4.4.1 Redirection & modifying severity . . . 47
4.4.2 Rotation . . . 47

5 Verification **49**
5.1 Dump . . . 49
5.1.1 Introduction . . . 49
5.1.2 API . . . 49
5.1.3 Exemple . . . 50
5.1.4 Test . . . 50
5.2 Ping . . . 50
5.2.1 Introduction . . . 50
5.2.2 Links configuration . . . 51
5.2.3 Non regression test . . . 51
5.3 Ping fifo . . . 51
5.3.1 Introduction . . . 51
5.3.2 Single test . . . 51
5.3.3 Non regression test . . . 51
5.4 Ping rng . . . 52
5.4.1 Introduction . . . 52
5.4.2 Single test . . . 52
5.4.3 Non regression test . . . 52
5.5 Config . . . 52
5.5.1 Introduction . . . 52
5.5.2 Single test . . . 53
5.5.3 Non regression test . . . 53
5.6 Driver . . . 53
5.6.1 Introduction . . . 53
5.6.2 Single test . . . 53
5.6.3 Non regression test . . . 54

6 Validation **57**
6.1 Introduction . . . 57
6.2 Bugs to solve . . . 57
6.3 ASU configuration . . . 57
6.4 Read-out chip's data . . . 57

II Miscellaneous **59**

1 Approach **61**
1.1 Introduction . . . 61
1.1.1 Firwares versions . . . 61
1.1.2 Jumpers . . . 61
1.1.3 Leds . . . 62
1.1.4 Inventory . . . 62
1.2 Subversion . . . 63
1.2.1 Introduction . . . 63
1.2.2 Using svn client . . . 65
1.2.3 Configuration . . . 66
1.3 Xilinx . . . 67
1.3.1 Introduction . . . 67
1.3.2 Xilinx installations . . . 68
1.3.3 Setup ISE 11.5 on Ubuntu 10.04 (lucid) . . . 69
1.3.4 Setup ISE DS 12.2 on Ubuntu 10.04 (lucid) . . . 69
1.3.5 Install JTAG support . . . 72
1.3.6 FPGA Editort . . . 73
1.3.7 Desktop shortcut icons . . . 74

1.3.8 Scan chain and load firmware . . . 75
1.4 DataBase . . . 76
1.4.1 Introduction . . . 76
1.4.2 Install on other OS . . . 79
1.4.3 Ecals parameters . . . 79
1.5 GUI . . . 80
1.5.1 Introduction . . . 80
1.5.2 Framework module . . . 80
1.5.3 Database module . . . 80
1.5.4 Electronical functionalities . . . 81
1.5.5 Physical functionalities module . . . 82
1.5.6 Conclusion . . . 83
1.5.7 Custumer's needs . . . 84
1.6 Online v2 . . . 85
1.6.1 Introduction . . . 85
1.6.2 Electronic DAQ . . . 85
1.6.3 Online server . . . 86
1.6.4 Slow control software . . . 87
1.6.5 DAQ software . . . 87
1.6.6 Data base . . . 87

2 Procedure **89**
2.1 poltst0 . . . 90
2.1.1 Introduction . . . 90
2.1.2 Partitions . . . 90
2.1.3 Shared partitions . . . 91
2.1.4 Dual boot . . . 91
2.1.5 Windows XP . . . 92
2.2 poldhcp54 . . . 93
2.2.1 Introduction . . . 93
2.2.2 Scientific Linux Cern 5.5 . . . 94
2.3 polntnr . . . 94
2.3.1 Introduction . . . 94
2.3.2 Dual boot . . . 94
2.3.3 Ubuntu LTS 10.04 . . . 96
2.3.4 Scientific Linux Cern 5.5 . . . 97
2.3.5 Windows XP . . . 97
2.4 pollinbtc . . . 98
2.4.1 Introduction . . . 98
2.4.2 Scientific Linux Cern 5.5 . . . 99
2.4.3 Virtual machine . . . 99
2.4.4 SVN stuffs . . . 99
2.4.5 Xilinx tools . . . 100
2.5 Ethernet . . . 100
2.5.1 Introduction . . . 100
2.5.2 Fast command . . . 100
2.5.3 Gigabit . . . 100
2.6 COM/ETH . . . 100
2.6.1 Introduction . . . 100
2.6.2 Configuration . . . 100
2.7 Purchase Orders . . . 101
2.7.1 Introduction . . . 101
2.7.2 Exemple Mail . . . 101

3 Tools **103**
3.1 lib pcap . . . 104
3.1.1 Introduction . . . 104
3.1.2 Python . . . 104
3.1.3 C . . . 104
3.2 Schroot . . . 113
3.2.1 Introduction . . . 113
3.3 Stow . . . 114
3.3.1 Introduction . . . 114
3.3.2 Exemple . . . 114
3.3.3 Using it on SLC . . . 114
3.4 Python . . . 115
3.4.1 Introduction . . . 115
3.4.2 QT4 designer . . . 116
3.4.3 Entry points . . . 116
3.5 Virtual Box . . . 116
3.5.1 Introduction . . . 116
3.5.2 Install . . . 116
3.5.3 Configuration . . . 117

III Annexes **119**
.0.4 Meeting 2011-10-24 . . . 131

Introduction

- **Project Description**
  A high granularity calorimeter system optimized for the particle flow measurement of multi-jet final states at the ILC running, with center-of-mass energy between 90 GeV and 1 TeV.

  You will find here CALICE's DAQ V2 software documentation about SDHCAL (Semi-Digital Hadronic Calorimeter):

  - Tools we use.
  - Documentation about the code we produce.
  - External documentations.

  Thanks reading me.

  I specialy want to thanks my collegues who help me to build the libLDA and this documentation.

  - Thanks to David Decotigny from who most code of the libLDA library comes and wich was interviewed for 2 mounth in order to produce the first version of the present documentation.
  - Thanks to Simon Chollet who help me scheduling and debating the project millestonnes, providing me good intellectual working conditions.
  - Thanks to Muriel Cerutti using the libLDA and reading this documentation.
  - Thanks to Emilia Bevecha and David Chamont to follow this project.
  - Thanks to Vincent Boudry to employ me at LLR.

- **Evo**: `01 44 27 81 50`

- **LLR team**

name	job	know-how	contact ((+33) 1.69.33.)
Vincent Boudry*	DHCAL Physics	management	55.37 vincent.boudry@llr.in2p3.fr
Daniel Jeans*	ECAL Physics	management	55.96 daniel.jeans@llr.in2p3.fr
Rémi Cornat*	Electronics	management + DIF ECAL	56.20 remi.cornat@llr.in2p3.fr
David Chamont*	Computer science	management	56.33 chamont@llr.in2p3.fr
Emilia Becheva*	Computer Science	management	56.34 becheva@llr.in2p3.fr
Simon Chollet*	Computer Science	management	56.34 s.chollet@llr.in2p3.fr
Franck Gastaldi	Electronics	DCC+LDA	56.13 gastaldi@llr.in2p3.fr
Muriel Cerutti	Computer Science	GUI ECAL	56.32 muriel.cerutti@llr.in2p3.fr
Nicolas Roche	Computer science	libLDA	56.35 nicolas.roche@llr.in2p3.fr

* : officer in charge.

- **Contacts**

name	entity	know-how	contact
Imad Laktineh*	IPNL	Management	04 72 43 11 15 i.laktineh@ipnl.in2p3.fr
Laurent Mirabito	IPNL	DAQ Tracker	04 72 43 19 85 l.mirabito@ipnl.in2p3.fr
Christophe Combaret	IPNL	DAQ	04 72 44 81 46 c.combaret@ipnl.in2p3.fr
Muriel Vander Donckt	IPNL	DAQ	04 72 44 84 83 muriel@ipnl.in2p3.fr
Guillaume Baulieu	IPNL	DB	g.baulieu@ipnl.in2p3.fr
Test room	IPNL		04 72 44 58 61
Guillaume Vouters	LAPP	Electronics DIF DHCAL	04 50 09 16 11 guillaume.vouters@lapp.in2p3.fr

Part I

CALICE's DAQ V2 software

This section provides the CALICE's DAQ V2 software specifications.

What software have to provide:

- **Electronic test bench**:
  We use Python to provide USB, serial and Ethernet links. Standard object introspection python module is used to develop tests.
- **Transport layer between DIF Hardware and XDAQ**:
  We provide a library that allow to load configuration from database and to access DIF read-out data from buffers, one buffer by DIF.
- **Only one channel**:
  We only have one channel to access the hardware, where we usually have 2, one for slow control and one for acquisition.

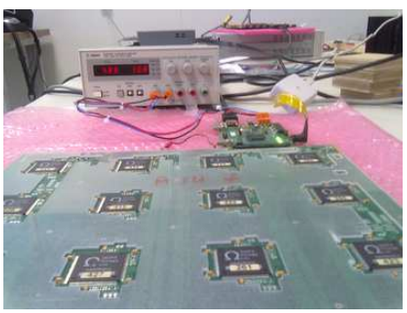

Figure 1: ASU, DIF and LDA in back

Figure 2: ASU and DIF

Figure 3: working

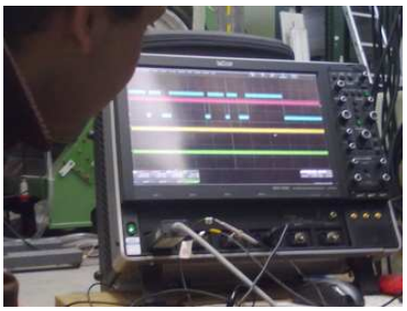

Figure 4: busy signal

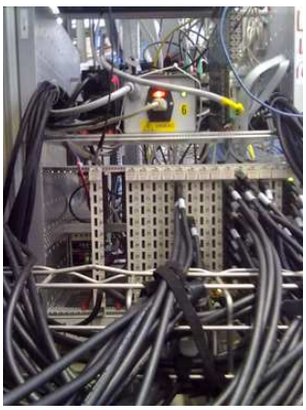

Figure 5: DCC

Chapter 1

Voice of the customer

1.1 Introduction

This section provides the CALICE's DAQ V2 customers needs.

1.2 Pcap Library

1.2.1 Buffers

As we use Ethernet protocol, we do not use the linux buffers provide for TCP. We use the Pcap library in order to provide input buffer.
DIF readout data and triggers buffer dispatch and reassemble the packets from the Pcap buffer.

1.2.2 threads

Each instance of the driver object create a receiveng thread that copy packets form Pcap to DIF buffers.
The XDAQ application main thread will read the DIF buffers using a call back function that only deliver the data for full trigger events (see bellow).
The XDAQ application should run a monitoring thread managing the buffers occupancy and avoiding dropped packed by the kernel.

Library architecture:

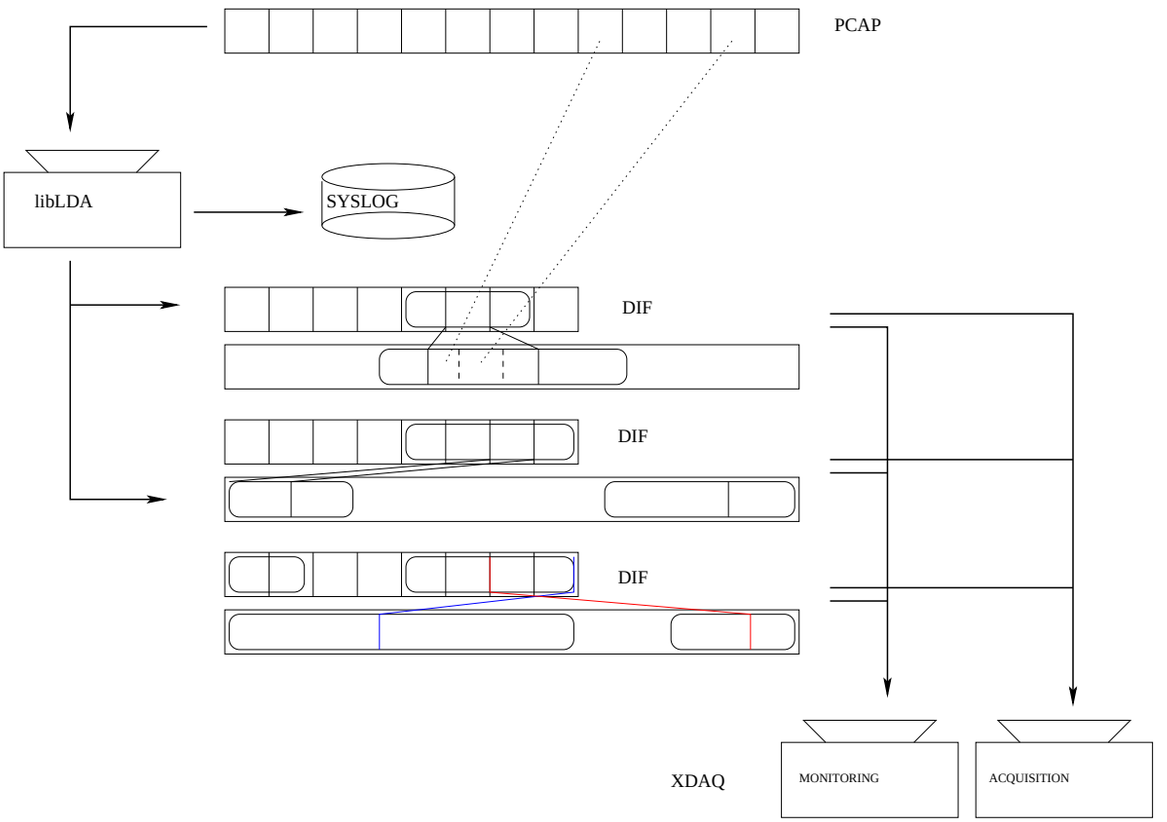

Figure 1.1: Functional specifications

User have to provide a call-back function to read-out the data. This permit to ensure that all chips have sent their data for the last trigger available.

```
void
readCallback(LLR_CALICE_CONF::DIFid difId,
     uint16_t *data, uint trigger, uint index, uint size);

void main(){
   ...
   run.read(readCallback);
}
```

In the upper schema, the callback function will be called 7 time to get all the available read-out data:

- 1 time for the first DIF, with a buffer content of 3 DIF packets.
- 2 times for the second DIF, with a buffer content of 2 DIF packets each time.
- 4 times for the third DIF, wich is the maximum.

Note: The exemple given above illustrate an ending run with all timeout expired. If not, the callback function buffer will only deliver the 3 first triggers (wich are already fully received and stored into all the DIF buffers).

(**More** on the implementation section).

1.3 Loading configuration from database

We provide an abstract class for configuration that will have to be implemented specially for database access.

```
class ConfDB : public Conf
  {
  public:
    ConfDB();
  };
```

(**More** on the functional analysis section).

1.4 Log using syslog daemon

```
$ tail -f /var/log/messages
...
Apr 28 03:09:08 poldhcp54 libLDA[29722]: INFO: read-out buffers size: 100 triggers for up to
...
```

(**More** on the implementation section).

1.5 Non regression tests

We provide no regression tests for both software modules and hardware functionality as connection, in/out data content check, read-out speed test.

```
# dump
# ping
# fifo
# rng
# config
# driver
```

(**More** on the verification section).

1.6 Packets formats

The packet format tend towards the one describe by specifications available in the documentation section.

1.6.1 Input

The input packets are dumped from the LDA's ethernet link.

```
# tcpdump -xx -i eth2 -s 1024 ether host 5e:70:0c:d2:77:e8

          0 1  2 3  4 5  6 7   8 9  A B  C D  E F
0x0000:  0022 1900 3d8b 5e70  0cd2 77e8 0810|0004
0x0010:  0001 dead 0001|1000  adde 0008 0100|...
```

note: DIF byte order have to be reverted: `3412` => `1234`

addr	words	field	comment
		[ODR...	
0x00	3	ETH Dst MAC	0022 1900 3d8b
0x06	3	ETH Src MAC	5e70 0cd2 77e8
0x0c	1	ETH Type	0809|0810|0811
		[LDA...	
0x0e	1	LDA Type	Sub System + Operation
0x10	1	LDA Modifier	LDA port &= --0F
0x12	1	LDA PktID	dead
0x14	1	LDA DataLength	
		[DIF...	
0x16	1	DIF packettype	DCC port &= --F0
0x18	1	DIF pktID	dead
0x1a	1	DIF type modifier	
0x1c	1	DIF data length	
		[BT...	
0x1e	?	Block Transfert	
		...BT]	
0x1e	?	Block Transfert	
?	1	DIF CRC	
		...DIF]	
?	?	LDA PAD	
?	2	LDA CRC32	
		...LDA]	

note: For now, slow-control data and readout data from asics are simply concatened into DIF block transfert.

1.6.2 Output

The ouptut packets are sent by the DIFs and relayed by the libLDA.

```
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
0054:   03 000173 0000 0000 0000 0000 0000 0000 0001 0000
0068:   03 000150 0000 0000 0000 0000 0000 0000 0001 0000
007c:   03 0001c4 0000 0000 0000 0000 0000 0000 0001 0000
0090:   03 0001f9 0000 0000 0000 0000 0000 0000 0001 0000
00a4:   03 0001bb 0000 0000 0000 0000 0000 0000 0001 0000
00b8:   03 0001bf 0000 0000 0000 0000 0000 0000 0001 0000
00cc:   03 000182 0000 0000 0000 0000 0000 0000 0001 0000
00e0:   03 0000b2 0000 0000 0000 0000 0000 0000 0001 0000
00f4:   03 0000ef 0000 0000 0000 0000 0000 0000 0001 0000
0108:   03 0000c9 0000 0000 0000 0000 0000 0000 0001 0000
011c:   03 000054 0000 0000 0000 0000 0000 0000 0001 0000
0130:   03 000060 0000 0000 0000 0000 0000 0000 0001 0000
0144:   03 000015 0000 0000 0000 0000 0000 0000 0001 0000
0158:   a3
0159:   b4
015a:   06 000112 0000 0000 0000 0000 0000 0000 0005 0000
016e:   06 000113 0000 0000 0000 0000 0000 0000 0004 0000
0182:   a3
0183:   b4
0184:   11 0001ba 0000 4000 0000 0000 0000 0000 0000 0000
0198:   a3
```

```
0199:   b4
019a:   12 0000b1 0000 0500 0000 0000 0000 0000 0000 0000
01ae:   12 0000b3 0000 0510 0000 0000 0000 0000 0000 0000
01c2:   a3
01c3:   b4
01c4:   14 0000bc 0000 0040 0000 0000 0000 0000 0000 0000
01d8:   a3
01d9:   b4
01da:   17 0000b1 0000 0000 0000 0000 0000 0000 01a6 0000
01ee:   17 0000b3 0000 0000 0000 0000 0000 0000 0162 0000
0202:   17 0000b2 0000 0000 0000 0000 0000 0000 0110 0000
0216:   17 0000d9 0000 0000 2000 0000 0000 0000 0000 0000
022a:   a3
022b:   a0
```

addr	bytes	field	comment
		[Global Header... (once by global trigger)	
0x1e	1	b0	
0x1f	4	DIF ID	defined by the ASU slow control
0x20	4	DIF Trigger counter	
0x24	4	Trigger USB busy	
0x28	4	Global Trigger counter	
0x3a	6	Absolute BCID	
0x40	3	BCID DIF	
		[Frame Header... (once by Chip)	
0x43	1	b4	
		[Data... (once by chip trigger <=128)	
0x44	1	Hardroc Header	
0x45	3	BCID	
0x48	16	Data	
		...Data]	
		...Global Header]	
		...Global Header]	

```
Send by Guillame Vouters:

Global Header (0xB0)
               DIF ID (8 bits)
               DIF Trigger counter 31-0 (32 bits)
               Trigger USB busy 31-0 (32 bits)
               Global Trigger counter 31-0 (32 bits)
               Absolute BCID 47-0 (48 bits)
               BCID DIF 23-0 (24 bits)
               { (envoyé j fois selon le nombre de hardrocs de la chaine)
               Frame_Header (0xB4)
                             { (envoyé i fois selon le remplissage de la mémoire du
                                hardroc concerné (max = 128))
                             Hardroc Header  (8 bits)
                             BCID  (24 bits)
                             Data (128 bits)
                             }
               Frame_Trailer (0xA3) (when all data OK or C3 when a problem occured
                                     during transfert)
               }
Global trailer (0xA0)
CRC MSB (8bits)
CRC LSB (8bits)
```

Chapter 2

Functional analysis

This section provides explanation on the libLDA architecture. The library provide:

- Loading configuration and sending it to hardware.
- Slow control, trigger, and acquisition threads manage by the `running` global variable.
- A read-out call ensuring the completeness of data regard to triggers that provide this data to a callback user function.

```
int main (int argc, char * argv[])
{
  /* Configure */
  LLR_CALICE_CONF::Conf* conf = LLR_CALICE_CONF::parseCommandLine(argc, argv);
  Driver run(conf);
  run.configure();

  /* Start */
  run.startSlowControl();
  run.start();
  run.startSoftTrigger();

  /* Read out */
  while(running) {
    usleep(run.delay_usr);
    run.dump(readCallback);
  }
  while (run.dump(readCallback) >0);
 }

 return 0;
}
```

2.1 Packets objects

2.1.1 Introduction

What: Managing packets content.

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

- *libLDA/include/packets/*
- *libLDA/packets/*

From *svn/calice/online-sw/trunk/daq/calice_packets/calice_packets.hpp* we can read:

```
 * Base class. Defines the notions of SDU and PDU common to all the
 * accessor objects.
 *
 * Typical topology of a SDU (aka. packet):
 *
 *  +-----------+--...-------...-----+--------+
 *  | Header    |       PDU          | Footer |
 *  +-----------+--...-------...-----+--------+
 *
 * The PktAccessor_Base class defines the access to the SDU and PDU
 * and allows to process the packet before it is actually sent. The
 * children classes would alow to access the various fields inside
 * the Header and Footer, and manipulate the packet attributes such
 * as the PDU size.
```

All the accessor objects inherit from the PktAccessor_Base ABC, which defines 2 notions: the SDU (Service Data Unit) and the PDU (Protocol Data Unit). The idea is that the whole packet is called the "SDU". It encapsulates a unique PDU, this PDU corresponding to the arbitrary data being transmitted by the packet. So packet=SDU, and data_carried_by_the_packet=PDU. The SDU is supposed to contain protocol data which have to be updated once the packet is ready to be sent, such as a CRC: the PktAccessor_Base interface declares the sync_crc() method to do just that.

2.1.2 API

PktAccessor_Base

```
class PktAccessor_Base
 {
 public:
   virtual ~PktAccessor_Base() { }
   virtual void * get_sdu() const =0;
   virtual unsigned int get_sdu_nbytes() const =0;
   virtual void * get_pdu() const =0;
   virtual unsigned int get_pdu_nbytes() const =0;
   virtual void sync_crc() =0;

 protected:
   PktAccessor_Base() { }
 };
```

PktAccessor_LDA_pkt

```
class PktAccessor_LDA_pkt
   : public PktAccessor_ODR_pkt
 {
 public:
   PktAccessor_LDA_pkt();
   PktAccessor_LDA_pkt(void const* sdu,
unsigned int sdu_nbytes,
bool sdu_is_garbage = false);
   PktAccessor_LDA_pkt(unsigned int min_pdu_nbytes);
   virtual void attach(void const* sdu,
unsigned int sdu_nbytes,
bool sdu_is_garbage = false); /* throw() */
```

```
    virtual void detach();
    virtual void * get_pdu() const;
    virtual unsigned int get_pdu_nbytes() const;
    virtual void set_pdu_nbytes(unsigned int min_pdu_nbytes);
    inline bool is_fast_command_request() const;

    /* Normal packets: */
    inline unsigned char get_LDA_subsystem() const;
    inline void set_LDA_subsystem(unsigned char);
    inline unsigned char get_LDA_optype() const;
    inline void set_LDA_optype(unsigned char);

    /* Same fields */
    inline unsigned short get_LDA_Modifier() const;
    inline void set_LDA_Modifier(unsigned short);
    inline unsigned short get_LDA_from_diflink_id() const; /* base 1 */
    inline void set_LDA_from_diflink_id(unsigned short /* base 1 */);
    inline unsigned short get_LDA_to_diflink_id() const; /* base 0 */
    inline void set_LDA_to_diflink_id(unsigned short /* base 0 */);
    inline unsigned short get_LDA_PktID() const;
    inline void set_LDA_PktID(unsigned short);
    inline unsigned short get_LDA_DataLength() const;
    inline void set_LDA_DataLength(unsigned short);

    /* Fast-command packets: */
    void init_LDA_FastCommand(unsigned short dif_mask,
      unsigned char comma,
      unsigned char data);
    inline bool unpack_LDA_FastCommand(unsigned short /*out*/& cmdw /*0xfa57*/,
       unsigned short /*out*/& dif_mask,
       unsigned char /*out*/& comma,
       unsigned char /*out*/& data) const;

  private:
    struct LDA_pkt *   _pkt;
    unsigned int       _pdu_nbytes; // Related to ODR::_pdu_nbytes
    uint16_t _compute_FC_parity() const; // Host byte order
    PktAccessor_LDA_pkt(PktAccessor_LDA_pkt const&); // Not implemented
  };
```

PktAccessor_DIF_pkt_block_transfer

```
  class PktAccessor_DIF_pkt_block_transfer
    : public virtual PktAccessor_Base
  {
  public:
    PktAccessor_DIF_pkt_block_transfer(void const* sdu,
       unsigned int buff_nbytes,
       bool buff_is_garbage = false,
       bool check_crc_ignore_dcc_nibble=false);
    explicit PktAccessor_DIF_pkt_block_transfer(unsigned int pdu_nbytes);
    virtual void * get_sdu() const;
    virtual unsigned int get_sdu_nbytes() const;
    virtual void * get_pdu() const;
    virtual unsigned int get_pdu_nbytes() const;
    virtual void sync_crc();
    inline unsigned short get_packettype() const;
    inline void set_packettype(unsigned short);
```

```
    inline unsigned short get_dcc_nibble() const;
    inline void set_dcc_nibble(unsigned short dcc_nibble);
    inline unsigned short get_pktID() const;
    inline void set_pktID(unsigned short);
    inline unsigned short get_type_modifier() const;
    inline void set_type_modifier(unsigned short);
    inline unsigned short get_data_length() const; // Number of 16b-words
    virtual void set_data_length(unsigned short num_16b_words);

  private:
    struct DIF_pkt_block_transfer * _pkt;
    const unsigned int _buff_nbytes; // Allocated mem size (FIXED)
    unsigned int       _sdu_nbytes;  // Related to _pdu_nbytes
    unsigned int       _pdu_nbytes;  // Related to _sdu_nbytes
  };
```

2.1.3 Compilation

Flags

There is two pre-compilation flags:

- CHECK_DIF_CRC: make the libLDA checking the CRC.
  Note: This will slow the DAQ acquisition and may imply data lost.

- DIF_FIRMWARE: make the libLDA compatible with both ECAL and DHCAL.

/include/packets/all.hh:

```
/* Check DIF CRC or not (faster) */
#define CHECK_DIF_CRC 0

/* Firmware version we use */
#define DIF_FIRMWARE DHCAL_v20
```

Ubuntu

```
# aptitude install g++ flex bison libboost-dev
$ make
$ make tests
```

SLC 5.5

```
# yum install gcc-c++ boost-devel
  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/packets
$ make
$ make tests
```

Gcc flag used

It seems to me that these flags are not needed, as I didn't mention diference with or without.

- `-fno-strict-aliasing`:
  Allow the compiler to assume the strictest aliasing rules applicable to the language being compiled. For C (and C++), this activates optimizations based on the type of expressions.

- `-pipe`:
  Use pipes rather than temporary files for communication between the various stages of compilation. This fails to work on some systems where the assembler is unable to read from a pipe; but the GNU assembler has no trouble.

- `-fomit-frame-pointer`:
  Don't keep the frame pointer in a register for functions that don't need one. This avoids the instructions to save, set up and restore frame pointers; it also makes an extra register available in many functions. It also makes debugging impossible on some machines.

- `-fno-delete-null-pointer-check`
  Use global dataflow analysis to identify and eliminate useless checks for null pointers. The compiler assumes that dereferencing a null pointer would have halted the program. If a pointer is checked after it has already been dereferenced, it cannot be null. In some environments, this assumption is not true, and programs can safely dereference null pointers. Use -fno-delete-null-pointer-checks to disable this optimization for programs which depend on that behavior.

2.1.4 Unitary tests

Here we can test the CRC computation of packet. This unit test is optimized for copy/past from the `tcpdump` output (where DIF bytes are inversed).

```
$ libLDA/src/packets/computeCrc
1234
crc= 0xd219
5678
crc= 0xe70b
^C
```

2.2 Conf objects

2.2.1 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)
```

2.2.2 API

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

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);
```

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;
  };
```

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");
}
```

2.2.3 Compilation

nothing special.

2.3 Basic objects

2.3.1 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.

2.3.2 API

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;
```

```
};
```

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;
};
```

2.3.3 Compilation

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
```

Ubuntu

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

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
```

2.4 Run objects

2.4.1 Introduction

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

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

2.4.2 API

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;
};
```

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
```

2.4.3 Compilation

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
```

2.4.4 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
...
```

2.5 Misc modules

2.5.1 Introduction

What: Miscellaneous stuffs.
This library is located at:

- *libLDA/include/misc/*
- *libLDA/misc/*

2.5.2 API

Log

"TestLog" class was used to dump log into memory. It is now change in order to use `syslog` daemon.

Thread

We provide a way to manage a secondary thread with fifo priorities.

Ring buffer

We provide ring buffer to share continus memory space between the 2 threads.

2.5.3 Compilation

Flags

Pre-compilation flags:

- DEBUG_THREAD: debuging messages about threads.
- DEBUG_TUNING: help to optimize basic's pcap parameters and run's buffer sizes.

/include/misd/all.h:

```
/* Constants values */
#define DEBUG_THREAD 0
#define DEBUG_TUNING 0
```

2.5.4 Unitary tests

```
# libLDA/misc/utLlrLog
[log.cc:45] INFO: Log info message
[log.cc:46] WARNING: Log warning Message
[log.cc:47] ERROR: Log error message

# libLDA/misc/utThread
1: start
1: thread :)
1: thread :)
1: thread :)
1: stop
2: start
2: thread :)  <-- ^C pressed
2: thread :)
2: stop
Will exit soon
```

```
# libLDA/misc/utAcbsm
[*                      ]
try to add 3 items ...
add 3 items [x..+                   ]
try to del 2 items ...
del 2 items [  x+                   ]
...
```

Chapter 3

Technical specifications

3.1 Introduction

"Pyserverdial" is a python framework that allow to send and analyse basic queries. Theses tests are in the *svn/calice/online-sw/trunk/pyserdiag* directory.

Clik on "Pyserverdial", "connect" to access all links one by one.
according to */calice@polcaldaq:.config/Calice/pyUSBSerialTests.conf*

- `echo` for testing send and reply
- `serial://COM1:115200` for RS232 on window
- `serial://dev:ttyS1` for RS232 on linux
- `serial://FTDI/DIF2` for USB link to DIF
- `serial:///dev/ttyUSB0:115200` for CCC via USB/RS232 apdapter
- `xmlrpc://` using RPC, the point to point remote procedure call (not used)
- `tcp://192.168.1.20:2300//` for CCC via comEth adapter
- `eth://1:2:3:4:5:6@nic4` for LDA on Windows
- `eth://1:2:3:4:5:6@eth6` for LDA on Linux

This script is the base of GUI interfaces for all the modules. For instance, "(send) load file" » "examples" » "llrtests" » "GUI_DIF.py" will open the DIF tests popup.

3.1.1 Architecture

- Controler is defined in *main.py*.
- DUT means Device Under Test object
- SEQ is a text input parsed block
- BLOCK is an octets block
- DevAddr is a string to identify devices
- action_loader use a sandbox "calicediag" to run object code
- inspect: standard object introspection python module
- According to "help" menu entry doc parsing is using parameters names

3.1.2 Compilation

Ubuntu LTS Lucid 10.04

```
# aptitude install qt4-designer pyqt4-dev-tools python-dpkt python-pcapy
$ ./setup.py build

$install -D /usr/bin/python /opt/ubuntu/usr/bin/netraw_python
# aptitude install libcap2-bin
# setcap 'CAP_NET_RAW+eip CAP_NET_ADMIN+eip' /opt/ubuntu/usr/bin/netraw_python

# getcap /opt/ubuntu/usr/bin/netraw_python
netraw_python = cap_net_admin,cap_net_raw+eip

$ /opt/ubuntu/usr/bin/netraw_python gui.py
```

Windows XP

According to *svn/calice/online-sw/trunk/pyserdiag/README.txt*, the gui also compile on windows with:

- Python 2.6.6
- PyQT 4.8.2
- py2exe 0.6.9
- msvcp90.dll from VC++ 2008 Express Fr to `c:\python26\DLLs\`
- click on `setup`
- click on `gui`

For serial link (no need to recompile):

- Create COM1 from VirtualBox and add it ("ajout de matériel") into Windows
- PySerial 2.5
- Install Visual C++ 2008 Express FR (just to have a C compiler)
- Install WpdPcap 4.1.2 sources (winpcap dev kit) into `c:\devel\oss\`
- pcapy 0.10.8: commenter tous les `#ifdef HAVE_PCAP_SENDPACKET`
- `c:\python26\python.exe setup.py install`
- install dpkt 1.7
- remove *Build* and *Dist* directories. `win> setup.py` will all build into *dist/*. *dist/gui.exe* is a 12MB stand-alone executable.
- it only required *wpcap.dll* istalled by WinPcap 4.1.2 driver

Test on Windows XP

- Upgrade Windows to SP3
- Install WinPcap 4.1.2 driver
- Run the pyserdiag GUI

Note: libpcap doesn't run using some ethernet drivers.

3.2 LDA

3.2.1 LDA_version

Send a packet to get the LDA version: Ask for registers 300f, 300e, 3000, 3015, 3010, 1002, 1000, 100a, 100e, 1022 and 1024.

- Sub System Encoding: 0x00 / LDA Registers
- Operation Encoding: 0x02 / Read
- LDA_Modifier: 0000 / packet should not be sent down
- LDA_PktID: 0000
- LDA_DataLength: 000b / For Register operations on the LDA it is the total number of LDA Register Packets that follow.

```
0x0000:  5e70 0cd2 510a 0022 1900 3d8b 0810 0002
0x0010:  0000 0000 000b 300f 0000 0000 300e 0000
0x0020:  0000 3000 0000 0000 3015 0000 0000 3010
0x0030:  0000 0000 1002 0000 0000 1000 0000 0000
0x0040:  100a 0000 0000 100e 0000 0000 1022 0000
0x0050:  0000 1024 0000 0000

0x0000:  0022 1900 3d8b 5e70 0cd2 510a 0810 0004
0x0010:  0000 0000 000b 300f 0000 2302 300e 0000
0x0020:  2011 3000 0000 0000 3015 0000 0000 3010
0x0030:  0000 0000 1002 0000 0010 1000 0000 0010
0x0040:  100a 0000 5249 100e 0000 0000 1022 0000
0x0050:  0010 1024 0000 0001
```

3.2.2 LDA_get_DIF_status

Read the 19 registers from bank 1 (DIF bank) of the LDA: Ask for registers 1000, 1002, 1004, 1006, 1008, 100a, 100b, 100e, 1010, 1011, 1012, 1013, 1018, 1019, 101a, 101b, 1020, 1022 and 1024.

- Sub System Encoding: 0x00 / LDA Registers
- Operation Encoding: 0x02 / Read
- LDA_Modifier: 0000 / packet should not be sent down
- LDA_PktID: 0000
- LDA_DataLength: 0013 / For Register operations on the LDA it is the total number of LDA Register Packets that follow.

```
0x0000:  5e70 0cd2 510a 0022 1900 3d8b 0810 0002
0x0010:  0000 0000 0013 1000 ffff ffff 1002 ffff
0x0020:  ffff 1004 ffff ffff 1006 ffff ffff 1008
0x0030:  ffff ffff 100a ffff ffff 100b ffff ffff
0x0040:  100e ffff ffff 1010 ffff ffff 1011 ffff
0x0050:  ffff 1012 ffff ffff 1013 ffff ffff 1018
0x0060:  ffff ffff 1019 ffff ffff 101a ffff ffff
0x0070:  101b ffff ffff 1020 ffff ffff 1022 ffff
0x0080:  ffff 1024 ffff ffff

0x0000:  0022 1900 3d8b 5e70 0cd2 510a 0810 0004
0x0010:  0000 0000 0013 1000 0000 0010 1002 0000
```

```
0x0020:  0010 1004 0000 0000 1006 0000 0000 1008
0x0030:  0000 0000 100a 0000 5000 100b 0000 0000
0x0040:  100e 0000 0000 1010 0000 0000 1011 0000
0x0050:  0000 1012 0000 0000 1013 0000 0000 1018
0x0060:  0000 0000 1019 0000 0000 101a 0000 0000
0x0070:  101b 0000 0000 1020 0000 0000 1022 0000
0x0080:  0010 1024 0000 0001
```

3.2.3 LDA_en_restart_links

Reset the LDA DIF links according to the mask: Write registers 1002, 1000, 1008, 4006, 4007 and 4008.

- Sub System Encoding: 0x00 / LDA Registers
- Operation Encoding: 0x01 / Write
- LDA_Modifier: 0000 / packet should not be sent down
- LDA_PktID: 0000
- LDA_DataLength: 0006 / For Register operations on the LDA it is the total number of LDA Register Packets that follow.

```
lda_out_mask  = 0x10
host_mac_addr = 00:22:19:00:3d:8b

0x0000:  5e70 0cd2 510a 0022 1900 3d8b 0810 0001
0x0010:  0000 0000 0006 1002 0000 0010 1000 0000
0x0020:  0010 1008 0000 0010 4006 0000 3d8b 4007
0x0030:  0000 1900 4008 0000 0022

0x0000:  0022 1900 3d8b 5e70 0cd2 510a 0810 0003
0x0010:  0000 0000 0006 1002 0000 0010 1000 0000
0x0020:  0010 1008 0000 0010 4006 0000 3d8b 4007
0x0030:  0000 1900 4008 0000 0022 0000
```

note: light the DCC green “link” led or the second upper DHCAL red led in case of direct LDA to DIF.

3.2.4 LDA_send_fast_command

"Send a fast command to the DIF(s)/DCC:

- Command Word: is set to a constant. Currently defined as 0xFA57.
- DIF_Link: 0010 / A mask which defines which port the command is for. A value of 0xFFFF would be used as a broadcast to all currently active DIF links.
- Comma: defines which comma character to use
- Data: defines which byte to send as the data byte.
- Parity: is a simple check (computed data).

```
lda_out_mask = 0x10
comma        = 0x7c
data         = 0x42

0x0000:  5e70 0cd2 510a 0022 1900 3d8b 0809 fa57
0x0010:  0010 7c42 0015
```

3.2.5 LDA_dcm_relock

Reset the LDA DIF links according to the mask: Write lda out mask into register 1024 and then clean this register.

- Sub System Encoding: 0x00 / LDA Registers
- Operation Encoding: 0x01 / Write
- LDA_Modifier: 0000 / packet should not be sent down
- LDA_PktID: 0000
- LDA_DataLength: 0001 / For Register operations on the LDA it is the total number of LDA Register Packets that follow.

```
lda_out_mask = 0x10

0x0000:  5e70 0cd2 510a 0022 1900 3d8b 0810 0001
0x0010:  0000 0000 0001 1024 0000 0010

0x0000:  0022 1900 3d8b 5e70 0cd2 510a 0810 0003 // Write ACK
0x0010:  0000 0000 0001 1024 0000 0010

0x0000:  5e70 0cd2 510a 0022 1900 3d8b 0810 0001
0x0010:  0000 0000 0001 1024 0000 0000

0x0000:  0022 1900 3d8b 5e70 0cd2 510a 0810 0003
0x0010:  0000 0000 0001 1024 0000 0000
```

3.3 DCC

3.3.1 send_FC_DCC_reset

Send a fast command to reset the DCC (7c/10):

```
lda_out_mask = 0x10

0x0000:  5e70 0cd2 510a 0022 1900 3d8b 0809 fa57
0x0010:  0010 7c10 0035
```

3.3.2 send_FC_DCC_get_status

Send FCMD K28.3/D15.0 (aka. 7C/D15.0, DCC get status) and print out the whole DCC register page:

- Sub System Encoding: 0x01 / DIF Transport
- Operation Encoding: 0x09 / Not documented ?!
- LDA_Modifier: 0005 / packet receive from LDA port 5
- LDA_PktID: d00d ?!
- LDA_DataLength: 0001 / For DIF operations it is the Number of DIF Packets that follow.
- DIF_packettype: e000 / DCC identifier
- DIF_packetId: 7c2f
- DIF_type_modifier: 0f40

- DIF_data_length: 0035

```
lda_out_mask = 0x10

0x0000:  5e70 0cd2 510a 0022 1900 3d8b 0809 fa57
0x0010:  0010 7c0f 0015

0x0000:  0022 1900 3d8b 5e70 0cd2 510a 0810 0109
0x0010:  0005 d00d 0001 00e0 2f7c 040f 3500 1807
0x0020:  0400 ff01 0000 0000 ff01 0000 0200 0000
0x0030:  0000 0400 0000 0000 0600 0000 0000 0800
0x0040:  4912 0000 0a00 490a 0000 0b00 0000 0000
0x0050:  0e00 0000 0000 1000 0000 0000 1100 0000
0x0060:  0000 1200 0000 0000 1800 0000 0000 1900
0x0070:  0000 0000 1a00 0000 0000 2000 0001 0000
0x0080:  2200 0100 0000 2400
```

note: the DIF and DCC inverts byte order: 0xabcd is sent as 0xcdba.

3.3.3 send_FC_DCC_init_links

Note: please remind that all the fast command are broadcasted to all DIFs connected to the DCC.
Send FCMD K28.3/D15.1 (aka. 7C/D15.1, DCC init links):

```
lda_out_mask = 0x10

0x0000:  5e70 0cd2 510a 0022 1900 3d8b 0809 fa57
0x0010:  0010 7c2f 0035
```

3.3.4 send_BT_DCC_get_status

Send BT "DCC get status" (dcc nibble 0x, type mod 2) and print out the whole DCC register page:

```
lda_out_link = 5

0x0000:  5e70 0cd2 510a 0022 1900 3d8b 0810 0102
0x0010:  0005 0000 0008 00e0 5555 0200 0000

0x0000:  0022 1900 3d8b 5e70 0cd2 510a 0810 0109
0x0010:  0005 d00d 0001 00e0 2f7c 040f 3500 1807
0x0020:  0400 ff01 0000 0000 ff01 0000 0200 0000
0x0030:  0000 0400 0000 0000 0600 0000 0000 0800
0x0040:  4912 0000 0a00 490a 0000 0b00 0000 0000
0x0050:  0e00 0000 0000 1000 0000
```

3.3.5 send_BT_DCC_config_tx_rx

Configure TX/RX enable regs of DCC (channel bitmaps):

```
lda_out_link = 5
txen         = 0x1ff // dcc out mask enabling transmission
rxen         = 0x1ff // dcc out mask enabling reception

0x0000:  5e70 0cd2 510a 0022 1900 3d8b 0810 0101
0x0010:  0005 0001 0014 00e0 5555 0100 0600 ff01
0x0020:  0000 0000 ff01 0000 0200
```

note: light the second upper DHCAL red leds.

3.3.6 send_BT_DCC_restart

Send restart:

```
lda_out_link = 5
restart_mask = 0x1ff

0x0000:  5e70 0cd2 510a 0022 1900 3d8b 0810 0101
0x0010:  0005 0001 000e 00e0 5555 0100 0300 0000
0x0020:  0000 0800
```

3.3.7 send_BT_DCC_start_RTT

Start a new RTT measure session:

```
lda_out_link = 5
channels     = 0x113

0x0000:  5e70 0cd2 510a 0022 1900 3d8b 0810 0101
0x0010:  0005 0001 0014 00e0 5555 0100 0600 1301
0x0020:  0000 0400 1301 0000 0800
```

3.3.8 send_BT_DCC_stop_RTT

Stop a RTT measure session:

```
lda_out_link = 5

0x0000:  5e70 0cd2 510a 0022 1900 3d8b 0810 0101
0x0010:  0005 0001 000e 00e0 5555 0100 0300 0000
0x0020:  0000 0400
```

3.3.9 send_BT_DCC_relock_DCM

Write 1 to bit 1 of DCC's register 0x24:

```
lda_out_link = 5

0x0000:  5e70 0cd2 510a 0022 1900 3d8b 0810 0101
0x0010:  0005 0001 000e 00e0 5555 0100 0300 0200
0x0020:  0000 2400
```

3.3.10 send_BT_DCC_register_blob

Send the given sequence of bytes to the DCC with the given type modifier (1 = status page write):

```
lda_out_link      = 5
dif_type_modifier = 0x1
ascii_bytes       = ff ff 00 00

0x0000:  5e70 0cd2 510a 0022 1900 3d8b 0810 0101
0x0010:  0005 0001 000e 00e0 5555 0100 0300 ffff
0x0020:  ffff 0000
```

3.3.11 send_MULTI_BT_DCC_get_status

```
lda_out_link = 5
times        = 2

0x0000:  5e70 0cd2 510a 0022 1900 3d8b 0810 0102
0x0010:  0005 0000 0008 00e0 5555 0200 0000

0x0000:  0022 1900 3d8b 5e70 0cd2 510a 0810 0109
0x0010:  0005 d00d 0001 00e0 2f7c 040f 3500 1807
0x0020:  0400 0100 0000 0000 0100 0000 0200 0100
0x0030:  0000 0400 0100 0000 0600 0100 0000 0800
0x0040:  0100 0000 0a00 0100 0000 0b00 0100 0000
0x0050:  0e00 0100 0000 1000 0100

0x0000:  5e70 0cd2 510a 0022 1900 3d8b 0810 0102
0x0010:  0005 0000 0008 00e0 5555 0200 0000

0x0000:  0022 1900 3d8b 5e70 0cd2 510a 0810 0109
0x0010:  0005 d00d 0001 00e0 2f7c 040f 3500 1807
0x0020:  0400 0100 0000 0000 0100 0000 0200 0100
0x0030:  0000 0400 0100 0000 0600 0100 0000 0800
0x0040:  0100 0000 0a00 0100 0000 0b00 0100 0000
0x0050:  0e00 0100 0000 1000 0100
```

3.4 DIF

3.4.1 send_FC_DIF_reset

Send FCMD K28.3/D1.1 (aka. 7C/21, reset DIF):

```
lda_out_mask = 0x10

0x0000:  5e70 0cd2 510a 0022 1900 3d8b 0809 fa57
0x0010:  0010 7c21 0015
```

note: This fast command is used to stop the rng test.

3.4.2 send_BT_DIF_register_state

Send a block transfer command to check if DIF are ready or not.
DIF reply with all registers at 0 if ready.

```
0x0000:  5e70 0cd2 777f 001b 217a bc85 0810 0102
0x0010:  0005 cba6 000a 0290 adde 1200 0100 0200

(DIF reply)
0x0000:  001b 217a bc85 5e70 0cd2 777f 0810 0109
0x0010:  0005 d00d 0001 0290 dada 1200 1000 0000
0x0020:  0000 0000 0000 0000 0000 0000 0000 0000
0x0030:  0000 0000 0000 0000 0000 0000 0000 cb91
```

3.4.3 send_BT_DIF_command

Send a block transfer command (DIF type modifier) + 1 data word:

```
lda_out_mask =
```

3.4.4 send_BT_DIF_write_debug_fifo

Write some data (sequence, starting at 1) to the DIF internal fifo:

```
lda_out_mask       = 0x10
dcc_pkttype_nibble = 9
num_bytes          = 24

0x0000:  5e70 0cd2 510a 0022 1900 3d8b 0810 0101
0x0010:  0005 0000 0020 0190 5555 0a00 1800 0102
0x0020:  0304 0506 0708 090a 0b0c 0d0e 0f10 1112
0x0030:  1314 1516 1718
```

3.4.5 send_FC_DIF_read_debug_fifo

Send FCMD K28.3/D2.1 (aka. 7C/22, read debug fifo):

note: this fast command differ between firmware versions 1 and 2:

- Firmware v1

  ```
  lda_out_link = 0x10
  ```

- Firmware v2

  ```
  lda_out_link = 0x10
  ```

3.4.6 send_BT_DIF_read_debug_fifo

Send BT 0xe to read the contents of the DIF internal FIFO:

```
lda_out_mask       = 0x10
dcc_pkttype_nibble = 9

0x0000:  5e70 0cd2 510a 0022 1900 3d8b 0810 0102
0x0010:  0005 0000 000a 1090 5555 0100 0100 0100

No reply... TODO
```

note: this doesn't works on firmware v2.

3.4.7 send_BT_DIF_write_RAM

Send BT 0xe to read the contents of the DIF internal FIFO:

```
lda_out_link =
```

3.4.8 send_BT_DIF_config_rndgen

Load Register bank 1 (DIF pseudo-random packet generator):

```
lda_out_link =
```

note:

- Configuration via `send_BT_DIF_config_rndgen` with "en" disabled, then click

- If num rndpkts set to 511 then generate an infinit stream of RNG packets
- **IMPORTANT**: inter rndpkt gap soit **must be** > 16
- Verification by `send_BT_DIF_read_bank` select_back set to 1 (cf map registers at Firmwares_DIF_de_test)
- Start with `send_BT_DIF_config_rndgen` with avec "en" enabled
- RAZ configuration: `send_BT_DIF_config_rndgen` with "en" disabled
- Infinit RNG packets stream should be stopped using `send_FC_DIF_reset`

3.4.9 send_BT_DIF_read_bank

Read DIF REG bank (BTCMD X0012):

```
lda_out_link =
```

3.4.10 send_BT_DIF_ignored

Read DIF REG bank (BTCMD Xff):

```
lda_out_link =
```

3.4.11 send_FC_DIF_Guillaume

Custom fast command:

```
lda_out_link =
```

3.4.12 send_BT_DIF_cmd_register

Send a block transfer command (DIF type modifier) + 1 data word:

```
lda_out_link =
```

3.5 Hight Level tests

3.5.1 Configure links

- `LDA_en_restart_links` light the DCC green "link" led or the second upper DHCAL red led in case of direct LDA to DIF.

```
0x0000:  5e70 0cd2 7968 0022 1900 3d8b 0810 0001
0x0010:  0000 0000 0006 1002 0000 0001 1000 0000
0x0020:  0001 1008 0000 0001 4006 0000 3d8b 4007
0x0030:  0000 1900 4008 0000 0022

0x0000:  0022 1900 3d8b 5e70 0cd2 7968 0810 0003
0x0010:  0000 0000 0006 1002 0000 0001 1000 0000
0x0020:  0001 1008 0000 0001 4006 0000 3d8b 4007
0x0030:  0000 1900 4008 0000 0022 0000
```

- `send_BT_DCC_config_tx_rx` light the second upper DHCAL red led.

```
0x0000:  5e70 0cd2 7968 0022 1900 3d8b 0810 0101
0x0010:  0001 0001 0014 00e0 5555 0100 0600 ff01
0x0020:  0000 0000 ff01 0000 0200
```

Note: provide a host mac address with "LDA_en_restart_links" if 2 setups share the same switch.

3.5.2 FIFO test

Status:

	ECALv1	ECALv2	DHCALv2
BT write	ok	ok	ok
FC read	data + echo	need flush at power on ; echo + data	data + echo
BT read	ok	KO	KO

note: We may prefer to use block transfer instead of fast command because we don't want the read orders arrive before the write orders.

- `LDA_en_restart_links`
- `send_BT_DCC_config_tx-rx`
- `send_BT_DIF_write_debug_fifo`

```
0x0000:  5e70 0cd2 7968 0022 1900 3d8b 0810 0101
0x0010:  0001 0000 0020 0110 5555 0a00 1800 0102
0x0020:  0304 0506 0708 090a 0b0c 0d0e 0f10 1112
0x0030:  1314 1516 1718
```

- `send_BT_DIF_read_debug_fifo`

```
0x0000:  5e70 0cd2 7968 0022 1900 3d8b 0810 0102
0x0010:  0001 0000 000a 0210 5555 0e00 0200 0100

0x0000:  0022 1900 3d8b 5e70 0cd2 7968 0810 0109
0x0010:  0001 d00d 0001 0112 0304 0506 0708 090a
0x0020:  0b0c 0d0e 0f10 1112 1314 1516 1718 f1da
0x0030:  33ee 0000 0000 0000 0000 0000
```

3.5.3 RNG test

Status:

	ECALv1	ECALv2	DHCALv2
hline config rng	ok	ok	ok
BT reset	ok	ok	KO

- `LDA_en_restart_links`
- `send_BT_DCC_config_tx-rx`
- `send_BT_DIF_config_rndgen`

```
0x0010:  0001 0000 0010 0150 ae60 1000 0400 4000
0x0020:  0100 0100 8000

replies:
0x0000:  ffff ffff ffff 5e70 0cd2 5cd6 0810 0109
0x0010:  0001 d00d 0001 0250 0400 0800 1000 2000
0x0020:  4000 8000 0001 0002 0004 0008 0010 0020
0x0030:  0040 0080 11a0 33e0 7760 eec0 cd21 9a43
```

```
0x0040:  3487 79ae e3fc d759 aeb3 4dc7 8b2e 165d
0x0050:  2cba 49d4 8308 0611 0c22 1844 3088 71b0
0x0060:  f3c0 f721 ee43 dc87 a9af 43ff 975e 2ebd
0x0070:  4dda 8b14 1629 2c52 58a4 a1e8 5371 a6e2
0x0080:  5d65 baca 6535 ca6a 94d5 390b 7216 e42c
0x0090:  c859 90b3 31c7 732e 4285

0x0000:  ffff ffff ffff 5e70 0cd2 5cd6 0810 0109
0x0010:  0001 d00d 0001 e65c ccb9 89d3 0307 060e
0x0020:  0c1c 1838 3070 60e0 d160 a2c1 5523 aa46
0x0030:  548d b9ba 63d5 d70a ae15 5c2b b856 70ad
0x0040:  f1fa f355 e6ab ddf7 ab4f 569f bd9e 6b9d
0x0050:  c79a 9f95 2f8b 4fb6 8fcc 0f39 1e72 3ce4
0x0060:  6968 d2d0 b501 6a03 d406 a80d 501b a036
0x0070:  406d 80da 1115 222a 4454 88a8 01f1 1342
0x0080:  2684 5da8 abf0 4741 8e82 0da5 0bea 0774
0x0090:  0ee8 0d70 1ae0 2560 b9441
```

3.5.4 Sending ASICs configuration

- LDA_en_restart_links
- send_BT_DCC_config_tx-rx
- send_BT_DIF_cmd_register
- test_lapp2
- test_lapp1

3.5.5 Read out from ASICs

- LDA_en_restart_links
- send_BT_DCC_config_tx-rx
- send_FC_start_acq
- send_trig_ext_cmd_register
- send_BT_start_RO_command

Chapter 4

Implementation

4.1 Introduction

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

4.2 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
```

4.3 API

4.3.1 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();
...
}
```

4.3.2 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;
}
```

4.4 Configuring Syslog

The logged messages are store using the SYSLOG daemon:

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

4.4.1 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
```

4.4.2 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

Chapter 5

Verification

Software status should be follow looking at this 4 non regression tests:

	Python	c++
fifo (up-down links links)	**BT didn't works**	**using FC insteed of BT**
rng (speed download link)	**register order change**	**didn't works after a soft reset**
config (configure)	**do not use headers**	**sometime loose the last word**
driver (read-out)	ok	ok

note: fifo and rng tests only access the DIF, where config and driver tests go up to the ASICS.

5.1 Dump

5.1.1 Introduction

What: Emplementing the "flush" function that provide a simple but complete example explaining how we capture packets.

Object located at:

- *libLDA/run/flush.o*
- *libLDA/run/dump*

5.1.2 API

In order to minimize the number of packets copies, we provide a callback function to the libpcap.

```
void Flush::callback(u_char *user, const struct pcap_pkthdr *h, const u_char *bytes)
{
  int pktLen = 0;
  static int npkts = 0;
  static CALDIF::PktAccessor_LDA_pkt ldaPkt;

  /* full access to API with run object */
  Flush *run = (Flush*)user;
  run->nbFlush++;
}

void Flush::flush(int msTimeOut)
{
  while (lda->dispatch(msTimeOut, Flush::callback, (u_char*)this) > 0);
}
```

5.1.3 Exemple

```
Flush::Flush(int argc, char * argv[])
  : Run(argc, argv) {}

void Flush::acquire()
{
  this->flush(1000);
  ::usleep(100000);
}

int main (int argc, char * argv[])
{
  Flush run(argc, argv);
  run.configure();

  /* run acquire as thread */
  run.nbFlush = 0;
  run.start();

  /* slow control */
  while(true) {
    printf("... %i packets flushed.\n", run.nbFlush);
    run.nbFlush = 0;
    sleep(1);
  }

  run.stop();
  return 0;
}
```

5.1.4 Test

```
# run/dump poldhcp54.conf -f
...
Flush(    5):      0%      1      1      2      1
Flush: [buff] [ 1,5] [ 1,8] [ lda] [ dcc]
Flush(    0):      0%      0      0      0      0

*** Total : miss 0 / 5 Pkts = 0%
 DIF [ 1,5] : 1 received + 0 missing 1 pkts and 0 errors.
 DIF [ 1,8] : 1 received + 0 missing 1 pkts and 0 errors.
 DIF [ lda] : 2 received + 0 missing 2 pkts and 0 errors.
 DIF [ dcc] : 1 received + 0 missing 1 pkts and 0 errors.
 Dropped pkts:    0
 Unexpected pkts: 0
```

5.2 Ping

5.2.1 Introduction

What: Emplementing the “connect” function and the “ping” utility.
Located at:

- *libLDA/run/connect.o*
- *libLDA/run/ping*

5.2.2 Links configuration

```
# run/fifo poldhcp54.conf -snf

Ping (    2):      0%  [ 1,5] [ 1,8]

*** Total : miss 0 / 2 Pkts = 0%
 DIF [ 1,5] : 1 received + 0 missing 1 pkts and 0 errors.
 DIF [ 1,8] : 1 received + 0 missing 1 pkts and 0 errors.
 Dropped pkts:    0
 Unexpected pkts: 0
```

5.2.3 Non regression test

```
# run/fifo poldhcp54.conf -f

lush: [buff] [ 1,5] [ 1,8]
Flush(    3):      0%      0      0
Ping (  602):      0%  [ 1,5] [ 1,8]
Ping (  602):      0%  [ 1,5] [ 1,8]
Ping (  186):      0%  [ 1,5] [ 1,8]

*** Total : miss 0 / 1393 Pkts = 0%
 DIF [ 1,5] : 695 received + 0 missing 695 pkts and 0 errors.
 DIF [ 1,8] : 695 received + 0 missing 695 pkts and 0 errors.
 Dropped pkts:    0
 Unexpected pkts: 0
```

5.3 Ping fifo

5.3.1 Introduction

What: Emplementing the “fifo” ping utility.
Binary located at:

- *libLDA/run/fifo*

This binary is used to test the uploading connection.

5.3.2 Single test

```
# run/fifo poldhcp54.conf -snf

Fifo (    6):      0%  [ 1,5] [ 1,8]

*** Total : miss 0 / 6 Pkts = 0%
 DIF [ 1,5] : 3 received + 0 missing 3 pkts and 0 errors.
 DIF [ 1,8] : 3 received + 0 missing 3 pkts and 0 errors.
 Dropped pkts:    0
 Unexpected pkts: 0
```

5.3.3 Non regression test

```
# run/fifo poldhcp54.conf -f

Flush: [buff] [ 1,5] [ 1,8]
Flush(    2):      0%      0      0
Fifo (   12):      0%  [ 1,5] [ 1,8]
```

```
Fifo (   12):      0%  [ 1,5] [ 1,8]
Fifo (   12):      0%  [ 1,5] [ 1,8]

*** Total : miss 0 / 38 Pkts = 0%
 DIF [ 1,5] : 18 received + 0 missing 18 pkts and 0 errors.
 DIF [ 1,8] : 18 received + 0 missing 18 pkts and 0 errors.
 Dropped pkts:    0
 Unexpected pkts: 0
```

5.4 Ping rng

5.4.1 Introduction

What: Emplementing the "rng" ping utility.

Binary located at:

- *libLDA/run/rng*

This binary is used to test the downloading connection.

Note: this test shows DIF sometime doesn't recept orders as they do not start the random packets generator..

5.4.2 Single test

```
# run/rng poldhcp54.conf -snf

Rng: [buff] [ 1,5] [ 1,8]
Rng  (    1):      0%      1      0

*** Total : miss 0 / 6 Pkts = 0%
 DIF [ 1,5] : 4 received + 0 missing 4 pkts and 0 errors.
 DIF [ 1,8] : 2 received + 0 missing 2 pkts and 0 errors.
 Dropped pkts:    0
 Unexpected pkts: 0
```

5.4.3 Non regression test

```
# run/rng poldhcp54.conf -f

Flush: [buff] [ 1,5] [ 1,8]
Flush(    3):      0%      0      0
Rng  ( 4941):      0%      0   4941
Rng  ( 4940):      0%      0   4940
Rng  ( 4941):      0%      0   4941
Rng  (  575):      0%      0    575

*** Total : miss 0 / 15400 Pkts = 0%
 DIF [ 1,5] : 0 received + 0 missing 0 pkts and 0 errors.
 DIF [ 1,8] : 15397 received + 0 missing 15397 pkts and 0 errors.
 Dropped pkts:    0
 Unexpected pkts: 0
```

5.5 Config

5.5.1 Introduction

What: Emplementing the "config" utility that is required to test ASICs configuration and behaviour.

Binary located at:

- *libLDA/run/config*

This binary is used to simulate a XDAQ configuration.
Note: this test shows that configuration less sometime one words.

5.5.2 Single test

```
# run/config poldhcp54.conf -snf

Conf (   14):      0%  [ 1,5] [ 1,8]

*** Total : miss 0 / 14 Pkts = 0%
 DIF [ 1,5] : 7 received + 0 missing 7 pkts and 0 errors.
 DIF [ 1,8] : 7 received + 0 missing 7 pkts and 0 errors.
 Dropped pkts:    0
 Unexpected pkts: 0
```

5.5.3 Non regression test

```
# run/config poldhcp54.conf -f

Flush: [buff] [ 1,5] [ 1,8]
Flush(    3):      0%      0      0
Conf (  622):      0%  [ 1,5] [ 1,8]
Conf (  548):      0%  [ 1,5] [ 1,8]
Conf (  150):      0%  [ 1,5] [ 1,8]

*** Total : miss 0 / 1323 Pkts = 0%
 DIF [ 1,5] : 660 received + 0 missing 660 pkts and 1 errors.
 DIF [ 1,8] : 660 received + 0 missing 660 pkts and 1 errors.
 Dropped pkts:    0
 Unexpected pkts: 0
```

5.6 Driver

5.6.1 Introduction

What: Emplementing the “driver” XDAQ will use.
Binary located at:

- *libLDA/run/driver*

This binary is used to simulate a XDAQ run.

5.6.2 Single test

Here we have set the `DISPLAY_PACKET` constant to 1 in the *include/run/all.hh* before compiling the libLDA.

```
# run/config poldhcp54.conf -s
...
INFO: Good, ASU are configured
# run/driver poldhcp54.conf -snf
[ 1,8] trigger 14 :
read-out 276 words
0000: b0 8000 0000 0e00 0000 0000 0000 0e00 0017 3673 a803 d567
```

```
000b: b4 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 a3
0016: b4 0080 0000 0000 0000 0000 0000 0000 0000 0000 0000 a3
0021: b4 0080 0000 0000 0000 0000 0000 0000 0000 0000 0000 a3
002c: b4 0080 0000 0000 0000 0000 0000 0000 0000 0000 0000 a3
0037: b4 0080 0000 0000 0000 0000 0000 0000 0000 0000 0000 a3
0042: b4 0080 0000 0000 0000 0000 0000 0000 0000 0000 0000 a3
004d: b4 0080 0000 0000 0000 0000 0000 0000 0000 0000 a3
0058: b4 0080 0000 0000 0000 0000 0000 0000 0000 0000 0000 a3
0063: b4 0080 0000 0000 0000 0000 0000 0000 0000 0000 0000 a3
006e: b4 0080 0000 0000 0000 0000 0000 0000 0000 0000 0000 a3
0079: b4 0080 0000 0000 0000 0000 0000 0000 0000 0000 0000 a3
0084: b4 0080 0000 0000 0000 0000 0000 0000 0000 0000 0000 a3
008f: b4 0080 0000 0000 0000 0000 0000 0000 0000 0000 0000 a3
009a: b4 0080 0000 0000 0000 0000 0000 0000 0000 0080 0000 a3
00a5: b4 0080 0000 0000 0000 0000 0000 0000 0000 0000 0000 a3
00b0: b4 0080 0000 0000 0000 0000 0000 0000 0000 0000 0000 a3
00bb: b4 0080 0000 0000 0000 0000 0000 0000 0000 0000 0000 a3
00c6: b4 0080 0000 0000 0000 0000 0000 0000 0000 0000 0000 a3
00d1: b4 0080 0000 0000 0000 0000 0000 0000 0000 0000 0000 a3
00dc: b4 0080 0000 0000 0000 0000 0000 0000 0000 0000 0000 a3
00e7: b4 0080 0000 0000 0000 0000 0000 0000 0000 0000 0000 a3
00f2: b4 0080 0000 0000 0000 0000 0000 0000 0000 0000 0000 a3
00fd: b4 0080 0000 0000 0000 0000 0000 0000 0000 0000 0000 a3
0108: b4 0080 0000 0000 0000 0000 0000 0000 0000 0000 0000 a3
0113: a0

*** Total : miss 0 / 27 Pkts = 0%
 DIF [ 1,8] : 26 received + 0 missing 26 pkts and 0 errors.
 DIF [ 1,5] : 1 received + 0 missing 1 pkts and 0 errors.
 Dropped pkts:    0
 Unexpected pkts: 0
```

Bug: this test shows that read-out less sometime one words (as configuration also do). This append more frequently because this is related to the number of the packets sent.

5.6.3 Non regression test

```
# run/driver poldhcp54.conf -nf
Drv: [pcap] [ 1,8]
Drv  (  540):      0%    46%
Drv  (  542):      0%    45%
Drv  (  540):      0%    45%
Drv  (  513):      0%    36%

*** Total : miss 0 / 2243 Pkts = 0%
 DIF [ 1,8] : 2160 received + 0 missing 2160 pkts and 0 errors.
 Dropped pkts:    0
 Unexpected pkts: 83
```

Warning: please use at less double buffer size than an expected event size. If the first and last word is corrupted as above, the 2 events are registered as a single event and overflow the buffer.

```
700: We receive DAQ:
0000: 0080 0000 0000 0000  0000 0000 0000 0000   0000 0000 a3b4

701: We receive DAQ:
0000: 0080 0000 0000 0000  0000 0000 0000 0000   0000 0000

704: We receive DAQ:
```

```
0000: a380

705: We receive DAQ:
0000: 0080 04c7 0000 0000  0000 04c7 0022 c238   052b 0000 18b4 ...
```

Chapter 6

Validation

6.1 Introduction

This section intent to validate the software production.

6.2 Bugs to solve

- DIF does'nt run RNG test a second time (after a soft reset).
- Sending configuration sometime loose the last word and push it into the next configuration.
- Reading data show empty chips lines.
- Reading data show strange BCids.

6.3 ASU configuration

We have to validate a configuration file extracted from the */data/DB/ch_28_26_37_32_36.sqlite* database available on all machine in Lyon. This database is browsable using the **Sqlite manager** Firerfox's plugin. Here we have such a file: *acqilc@lyoac26:/data/online/config/sc_lyon_1.txt*. Gains are about:

- B0: 190
- B1: 300
- b2: 600

6.4 Read-out chip's data

Tested on Scientific Linux 5.5:

- With one LDA and manual CCC

```
# ./exemple polcaldaq.conf -i1000 -t 100 -d 500000 -m4 -S2 -U10
```

- With one LDA and automatic CCC

```
# tools/utDevice polcaldaq.conf -i1000 -t 100 -d 500000 -m4 -S2 -U10
```

- With two LDA and automatic CCC

```
# tools/utDaq -i1000 -t 100 -d 500000 -m4 -S2 -U10
```

Note that we can capture the ethernet flow and re-inject it slower:

- Capture raw packets

```
# run/config polcaldaq.conf -i1000 -t 100 -d 500000 -m4 -S2 -U10
# tcpdump -xx -s 1024 -i eth0 ether host 5e:70:0c:d2:5c:d6 -w dump.raw
... ^C
```

- Run the libLDA without configuration:

```
# ./exemple polcaldaq.conf -p eth0 -n
```

- Re-inject the packets: **Note** that eth0 cable must be plugged to a switch, but the LDA must be unplugged to that switch. The `-Z` parameter is the mac address of the host which run the libLDA.

```
# ./pcat -I eth0 -M 5e:70:0c:d2:5c:d6 -Z 5c:26:0a:28:73:0a -m replay -i dump.raw
```

Part II

Miscellaneous

Chapter 1

Approach

1.1 Introduction

This section remind the actual status and provide the methods we use to update them.

1.1.1 Firwares versions

LDA

There is 3 FPGA but only **main** is checkable via slow control.

- startup: ?
- main: **v23.02.2011** => limited to 112Mb/s
- hdmi: ?

Using the libLDA, you should read the LDA version from log:

```
INFO: LDA version: 03-23-2011
```

DCC

Using the libLDA, you should read the DCC versions from log:

```
INFO: DCC@1 version: 18-07, id=16, status=4
```

DIFv

We are now working with v2.2l version. This may be change at compilation.

	v1	v1.2	v2	v2.2l
Fifo fast cod missing		x	?	?
Provides fake headers				x
Inverse rng data orders				x

1.1.2 Jumpers

Jumpers:

- LDA:
  - j5: none

  - j22: 3v3
  - j31: 2v5
  - j38: 0001 (marked side)
  - switchs:

```
           1 2 3 4 5 6 7 8
original: 1 0 1 1 0 1 1 1
modified: 1 0 1 0 1 1 1 1
```

- DIFF ECAL:
  - j1: none
  - j2: none
  - j3: 011
  - j4: 011
- DIFF DHCAL:
  - j6: none
  - j8: none

1.1.3 Leds

- LDA:
  - 6 red leds: ok
- DCC:
  - link: green=ok
  - clock: orangex2=ok
- DIFF ECAL:
  - green led = not programmed at power on
  - additionnal red led: blinking and next lited: ok

1.1.4 Inventory

- CCC:
  - **3**: LLR
  - **5**: LPNL
  - **6**: LLR
- LDA:
  - **9**: `5e:70:0c:d2:51:0a`
  - **10**: `5e:70:0c:d2:79:68` Low HDMI mezzanine
  - **11**: `5e:70:0c:d2:5e:95` Low HDMI mezzanine
  - **12**: `5e:70:0c:d2:77:e8`
  - **20**: `5e:70:0c:d2:78:34`
  - **21**: `5e:70:0c:d2:54:fe`

  - **25**: `5e:70:0c:d2:5c:d6`

- DCC:
  - **1**:

- DIFF ECAL:
  - **3**: LLR
  - **4**: LLR
  - **5**: LLR Warning
  - **6**: LLR
  - **7**: LLR
  - **13**: LLR
  - **14**: LLR
  - **15**: LLR
  - **16**: LLR
  - **17**: LLR

- DIFF DHCAL:
  - **5**: LPNL
  - **8**: LLR
  - **25**: LPNL
  - **27**: LLR
  - **30**: LLR
  - **35**: LLR
  - **36**: LLR
  - **40**: LLR
  - **154**: LPNL

- ASU DHCAL:
  - **7**: LLR

1.2 Subversion

1.2.1 Introduction

SVN is use to share sources by:

- boudry
- gastaldi
- ggrenier
- gvouters
- nroche
- rcornat

- combaret
- muriel
- Check out sources:

```
All:    $ svn co svn+ssh://svn-calice/calice (enter password twice)
Tagged: $ svn co svn+ssh://svn-calice/calice/online-sw/tags/libLDA-0.0
Last:   $ svn co svn+ssh://svn-calice/calice/online-sw/trunk/libLDA
Branche:$ svn co svn+ssh://svn-calice/calice/online-sw/branches/libLDA-SDHCAL-Cern-2011
```

- Update to last version:

```
$ svn update
$ svn info
Last Changed Rev: 1828
```

- Check local modifications:

```
$ svn status
$ svn diff {PATH}
Last Changed Rev: 1828
```

- Revert files:

```
$ svn revert {PATH}
```

- Commit changes:

```
$ svn add {PATH}
$ svn commit -m "{TEXT}"
```

- Update to an older revision:

```
$ svn info
Last Changed Rev: 1828
$ svn update -r 1800
```

- Tag a version:

```
$ svn cp trunk/libLDA tags/libLDA-0.2
A         tags/libLDA-0.2
$ svn commit
```

- Make a branch:

```
svn cp libLDA/ ~/svn/calice/online-sw/branches/libLDA-SDHCAL-Cern-2011
A         /home/nroche/svn/calice/online-sw/branches/libLDA-SDHCAL-Cern-2011
$ cd  ~/svn/calice/online-sw/branches/libLDA-SDHCAL-Cern-2011
$ svn commit -m "Création d'une branche à partir du code \
  se trouvant sur lyosdhcal4 vendredi 23/09/2011 à 10h"
```

- Erase a bad commit in trunk:

```
$ cd trunk/...
$ svn merge -r 2118:2116 .
$ svn commit
$ svn update
```

- Move a bad commit in trunk to an existing branch:

```
$ cd branches/...
$ svn merge -r 2109:2118 ../../trunk/libLDA
tf...
$ diff + cp
$ svn commit
$ svn update
```

1.2.2 Using svn client

Using command Lines

`$ svn ...`

- `$ svn info`
- `$ svn status`
- `$ svn diff --diff-cmd /usr/bin/diff -x -q`
- `$ svn diff`
- `$ svn update`
- `$ svn revert {PATH}`
- `$ svn add {PATH}`
- `$ svn del {PATH}`
- `$ svn commit -m "{TEXT}"`
- `$ svn mv {PATH} {PATH}`
- `$ svn mkdir {DIR}`
- `$ svn rmdir {DIR}`
- `$ svn cp {PATH}`
- `$ svn proplist`
- `$ svn propget`
- `$ svn propset`
- SVN use tags/branches that simply are copies of the main directory.
- SVN use FSFS BD (recommanded) or Berkley DB.

Using "svn-status mode" under emacs

`M-x svn-status`

C-h m	Display the keys binding
g	status update
U	update
a	add file
A	add file recursively
c	commit (C-x cc)
m	mark file
u	unmark file

Browsing the forge

diff on the forge: https://forge.in2p3.fr/projects/calice/repository/revisions/1609/diff

```
$ svn diff -r BASE:1606 | diffstat
$ svn diff -r BASE:1606 > myPatch
$ patch -p0 < myPatch   (see man patch)
```

1.2.3 Configuration

There is help here

For read-write access people must create there own key and send the public one. Next we must provide it to Loïc Tortay with a login and precise it is for Calice project.

Next, we can test the key doing:

```
$ ssh svn-calice
```

Here is a link to add building keys (there are english explanations too on the same web site).

Read Only access

For read-only access send the private key with instructions. Remind that consulation is granted via viewvc.

We need to build the */.ssh/config* file:

```
Host svn-calice-group svn-calice
        Hostname svn.in2p3.fr
        User calice
        IdentityFile ~/.ssh/id_rsa_group_ro_cvs-in2p3-fr
```

Read/Write from personnal account

We need to build the */.ssh/config* file:

```
Host svn-calice
    Hostname svn.in2p3.fr
    User nroche
    PasswordAuthentication no
    IdentityFile ~/.ssh/svnNRoche_dsa
    IdentitiesOnly yes
    RSAAuthentication yes
    PubkeyAuthentication yes
    ForwardX11 no
    ForwardAgent no
```

Finaly, we can use SVN:

```
<< ubuntu
# aptitude install subversion
==
# yum install subversion
>> slc
$ svn co svn+ssh://svn-calice/calice (enter password twice)
```

Read/Write from shared calice account

Provide theses files:

```
$ find /home/calice/Applications/nroche/ -type f
/home/calice/Applications/nroche/etc/ssh_config
/home/calice/Applications/nroche/etc/.ssh/id_rsa_nroche_cvs-in2p3-fr
/home/calice/Applications/nroche/bin/ssh
```

- */Applications/nroche/etc/ssh_config* file:

```
Host svn-calice
Hostname svn.in2p3.fr
User nroche
IdentityFile ~/Applications/nroche/etc/.ssh/id_rsa_nroche_cvs-in2p3-fr
        IdentitiesOnly yes
```

- */Applications/nroche/bin/ssh* file:

```
#! /bin/sh
exec /usr/bin/ssh -F $HOME/Applications/nroche/etc/ssh_config "$@"
```

Provide also theses files for confort:

```
$ find /home/calice/Desktop/outils\ svn/ -type f
/home/calice/Desktop/outils svn/svn_gastaldi.desktop
/home/calice/Desktop/outils svn/svn_nroche.desktop
/home/calice/Desktop/outils svn/README.txt
```

- */Desktop/outils svn/svn_nroche.desktop* file:

```
[Desktop Entry]
Version=1.0
Type=Application
Terminal=false
Icon[en_US]=gnome-network-properties
Name[en_US]=svn terminal nroche
Exec=sh -c 'cd "$HOME/Sources/calice" && \
env "PATH=$HOME/Applications/nroche/bin:$PATH" \
EDITOR=/usr/bin/gedit \
/usr/bin/gnome-terminal --profile=nroche'
Name=nroche svn
Comment=Terminal to issue svn commits for nroche
Icon=gnome-network-properties
```

Now we can test:

```
$ PATH=$HOME/Applications/nroche/bin:$PATH
$ ssh -v svn-calice

debug1: Reading configuration data /home/calice/Applications/nroche/etc/ssh_config
debug1: identity file /home/calice/Applications/nroche/etc/.ssh/id_rsa_nroche_cvs-in2p3-fr type -1
debug1: Trying private key: /home/calice/Applications/nroche/etc/.ssh/id_rsa_nroche_cvs-in2p3-fr
Enter passphrase for key '/home/calice/Applications/nroche/etc/.ssh/id_rsa_nroche_cvs-in2p3-fr':
```

1.3 Xilinx

1.3.1 Introduction

XILINX FPGA are embeded into :

- DIF (1)
- LDA (3: startup, main and hdmi), using version 11 for IP gigabit ethernet (but **not DS**)
- DCC, using version 12

Tools access the boards using JTAG (Joint Test Action Group stand for IEEE 1149.1 also called « Standard Test Access Port and Boundary-Scan Architecture »).

- IMPACT and ISE for XILINX's JTAG in and out streams
- OPENOCD for JTAG output streams: Linux Free Open Source Software (not used)
- QUARTUS for ALTERA (not XILINX) embed into DHCAL.
- FPGA EDITOR for add by hand pinoches in bitstreams. (XILINX: available in ISE)

File *svn/calice/online-sw/trunk/doc/altera-linux.txt* describe QUARTUS installation on `polcaldaq`. It wasn't tested. It was intented de be used by Guillaume for ALTERA FPGA.

1.3.2 Xilinx installations

Create a compte on Xilinx web site to download "SFD" software, wich is the fisrt minor relase (other are only updates and do not provide installer). File *svn/calice/online-sw/trunk/doc/xilinx-linux.txt* also describes the ISE installations. Installers binaries packages are saved at *calice@polcaldaq: /Download/pkg/*.

- Make sure you have at least 30 GB available where you want to install ISE (in the end, only 10 GB will be used but the installation process is... "under-optimized")
-
- Download *Xilinx_11.1_ISE_SFD.tar* from Xilinx,
- Untar it in */tmp*: `$ tar -C /tmp -xf Xilinx_11.1_ISE_DS_SFD.tar`,
- run `$ /tmp/Xilinx_11.1_ISE_SFD/xsetup` ...
  - leave the "cable drivers" checkbox unchecked.
  - I choose */mnt/data1/ubuntu/usr/xilinx11* as the base.
  - When asiking for licence, close the dialog box.
  - When asiking for Update, close the dialog box too (If not you have to jump the following point).
- You should remove installer now: `$ rm -fr /tmp/Xilinx_11.1_ISE_DS_SFD`.
- Download *Xilinx_11.5_ISE_DS_lin[64].tar* from Xilinx,
- Untar it in */tmp*,
- run `$ /tmp/Xilinx_11.5_ISE_DS_lin[64]/xsetup` and choose the same base.
- You should remove installer now.
- Installation is finished, you can test:

```
$ /mnt/data1/ubuntu/usr/xilinx11/ISE/bin/lin[64]/ise
$ /mnt/data1/ubuntu/usr/xilinx11/ISE/bin/lin[64]/impact
```

Note that ISE complain for a licence.

1.3.3 Setup ISE 11.5 on Ubuntu 10.04 (lucid)

- Load variables for licence */mnt/data1/ubuntu/etc/env-xilinx.sh*

```
XLX_TARGET=/mnt/data1/ubuntu/usr/xilinx11/
#XLX_LD_PRELOAD=/mnt/data1/ubuntu/usr/lib/libusb-xilinx-driver.so

# Licences
XILINXD_LICENSE_FILE='5296@ccflex04.in2p3.fr,5296@ccflex05.in2p3.fr,5296@ccflex06.in2p3.fr'
## XILINXD_LICENSE_FILE='1700@127.0.0.1,1800@127.0.0.1,1900@127.0.0.1'
export XILINXD_LICENSE_FILE

# X11 issues with fpga_editor
#LD_LIBRARY_PATH="$LD_LIBRARY_PATH:$XLX_TARGET/ISE/X11R6/lib/lin64"
#export LD_LIBRARY_PATH
```

- Now ISE doesn't complain for the licence:

```
$ . /mnt/data1/ubuntu/etc/env-xilinx.sh
$ /mnt/data1/ubuntu/usr/xilinx11/ISE/bin/lin[64]/ise
```

- In the main *settings[64|32].sh* file, replace the `pushd/popd` by a `_here=`/bin/pwd', cd $d, cd "$_here"`, and source becomes '.':

```
<<<
foreach i ( $newLst )
...
                        pushd "$d" > /dev/null
                        source "$sfn"
                        if ( $rc  == 0 ) then
                                popd > /dev/null
===
_here=`/bin/pwd`
for i in $newLst
...
                        cd "$d"
                        . "$sfn"
                        if [ $rc  = 0 ]; then
                                cd "$_here"
>>>
```

- Create a script wrapper */mnt/data1/ubuntu/usr/bin/xlx*:

```
#! /bin/sh

. /mnt/data1/ubuntu/etc/env-xilinx.sh
. /mnt/data1/ubuntu/usr/xilinx11/settings64.sh
exec env LD_PRELOAD="$XLX_LD_PRELOAD" "$@"
```

- Finished:

```
$ /mnt/data1/ubuntu/usr/bin/xlx impcat
$ /mnt/data1/ubuntu/usr/bin/xlx ise
```

1.3.4 Setup ISE DS 12.2 on Ubuntu 10.04 (lucid)

For version 12, you will find working configuration in *svn/calice/online-sw/trunk/doc/xilinx-ds12-lucid64.tgz*. This page describe the installation:

Validated on 3 different 64b machines.

- Download *Xilinx_ISE_DS_12.2_M.63c.1.1.tar* from Xilinx.
- Untar it in */tmp*
- Run `./xsetup` until completion. Let's call *DSBASE* the location on disk where you installed ISE : You should have a single *ISE_DS* directory under *DSBASE*
- In all subdirectories under *DSBASE/IDE_DS/* (ie. *ISE*, *EDK*, *PlanAhead*, *common*, ...), edit the *settings64.sh* file:
- Comment-out the line: *XIL_SCRIPT_LOC="$1"*
- Comment-out the line: *unset XIL_SCRIPT_LOC*
- Edit @DSBASE@/ISE_DS/settings64.sh:
- Remove the whole block from `if [ $# != 0 ]; then` until `unset XIL_SCRIPT_LOC_TMP_UNI ; fi`
- Right before the line '. *$d/settings64.sh "$d"*', add the line: *XIL_SCRIPT_LOC="$d"*
- Add the following script as *DSBASE/xilinx12.sh*:

```
#! /bin/sh

unset LANG
unset LANGUAGE

_here=`dirname "$0"`
here=`cd "$_here" && /bin/pwd`

. "$here"/ISE_DS/settings64.sh "$here"/ISE_DS
unset here

if [ -n "$XLX_LD_PRELOAD" ] ; then
  LD_PRELOAD="$LD_PRELOAD $XLX_LD_PRELOAD"
  export LD_PRELOAD
fi

exec "$@"
```

- *chmod 755* this *@DSBASE@/xilinx12.sh* script
- Create a new directory: @DSBASE@/ISE_DS/ZZZ_IN2P3
- Add the following *settings64.sh* file in that newly created DSBASE/ISE_DS/ZZZ_IN2P3/ directory:

```
XILINXD_LICENSE_FILE='5296@ccflex04.in2p3.fr,5296@ccflex05.in2p3.fr,5296@ccflex06.in2p3.fr'

LM_LICENSE_FILE="$XILINXD_LICENSE_FILE"
export XILINXD_LICENSE_FILE LM_LICENSE_FILE

XILINX_IN2P3=${XIL_SCRIPT_LOC}
export XILINX_IN2P3

if [ -n "$PATH" ]
then
   PATH=${XILINX_IN2P3}/bin:${PATH}
else
   PATH=${XILINX_IN2P3}/bin
fi
export PATH
```

```
# USB Cable
XIL_IMPACT_USE_LIBUSB=1
export XIL_IMPACT_USE_LIBUSB

XLX_LD_PRELOAD=${XILINX_IN2P3}/../../usb-driver/libusb-driver.so
export XLX_LD_PRELOAD
```

- Now create the following *bin* directory: *DSBASE/ISE_DS/ZZZ_IN2P3/bin*
- Add the following *xterm* script in that *DSBASE/ISE_DS/ZZZ_IN2P3/bin* directory:

```
#! /bin/sh

if [ x"$*" = x"-e sh --noprofile --norc" ] ; then
  exec gnome-terminal --sm-client-disable --disable-factory
--profile=xilinx-12 --title="XPS Terminal"
elif [ x"$1" = x-e ] ; then
  shift
  exec gnome-terminal --sm-client-disable --disable-factory
--profile=xilinx-12 --title="XPS run $1" -- "$@"
fi

exec /usr/bin/xterm "$@"
```

- Add the following *make* script in that *DSBASE/ISE_DS/ZZZ_IN2P3/bin* directory:

```
#! /bin/sh

_here=`dirname "$0"`
_here=`cd "$_here" && /bin/pwd`

# Make sure make/impact/etc. from here are used first, if any
PATH="$_here:$PATH"
export PATH

# Some xilinx tools refuse to work with the usb driver wrapper lib (64b)
unset LD_PRELOAD
exec /usr/bin/make "$@"
```

- symlink as gmake:

```
ln -s make @DSBASE@/ISE_DS/ZZZ_IN2P3/bin/gmake
```

- Add the following *impact* script in that *DSBASE/ISE_DS/ZZZ_IN2P3/bin* directory:

```
#! /bin/sh

# Make sure the LD_PRELOAD is set correctly
_here=`dirname "$0"`
_here=`cd "$_here" && /bin/pwd`

LD_PRELOAD="$XLX_LD_PRELOAD"
export LD\_PRELOAD

set -x

exec "$_here"/../../ISE/bin/lin64/impact "$@"
```

- Fix permissions:

```
chmod a+x DSBASE/ISE\_DS/ZZZ\_IN2P3/bin/*
```

From here, you should be able to run ISE, XPS, etc, but the JTAG won't work + you may see warning related to LD_PRELOAD not correctly defined and/or *libusb-driver.so* unavailable (-> see next section).

Note: due to limitations in FPGA editor, when launching ISE on the local screen of the machine, make sure the `DISPLAY` environment variable is set to `localhost:0`.

1.3.5 Install JTAG support

Trying to initialize the bundary chain using impact says: `Module windrvr6 is not loaded. Please reinstall the cabl`
This topic help on resolv that.

Configure udev

- Plug the xilinx cable, `$ dmesg` (or `# udevadm monitor`) talk little and the led is unlighted:

```
usb 2-2: new full speed USB device using ohci_hcd and address 3
usb 2-2: not running at top speed; connect to a high speed hub
usb 2-2: configuration #1 chosen from 1 choice
```

- Running `$ lsusb` should show the device:

```
Bus 002 Device 017: ID 03fd:0007 Xilinx, Inc.
```

- Unplug the cable
- Create */etc/udev/rules.d/xusbdfwu.rules*

```
$ cd DSBASE/common/bin/lin64/install_script/install_drivers/linux_drivers/pcusb

# Replace '=' by '-ge' when testing "ps -e | grep -c udevd"
$ sed -e 's:\(if \[ $TP_UDEV_ENABLED\) = \("1" \]\):\1 -ge \2:' -i setup_pcusb
$ sudo ./setup_pcusb
```

- restart udev: `# service udev restart`

```
$ tail /var/log/syslog
udevd[5826]: SYSFS{}= will be removed in a future udev version... in /etc/udev/rules.d/xusbdfwu.rul
udevd[5826]: BUS= will be removed in a future udev version... in /etc/udev/rules.d/xusbdfwu.rules:3
```

- Fix udev configuration:

```
# sed -e 's:SYSFS:ATTR:g;s:BUS:SUBSYSTEM:g;s:TEMPNODE:tempnode:g' -i /etc/udev/rules.d/xusbdfwu.rul
```

- Install fxload: `# aptitude install fxload`
- Plug the xilinx cable, `$ dmesg` says more and the led blind and now is lighted orange:

```
usb 2-2: new full speed USB device using ohci_hcd and address 4
usb 2-2: not running at top speed; connect to a high speed hub
usb 2-2: configuration #1 chosen from 1 choice
usb 2-2: USB disconnect, address 4
usb 2-2: new full speed USB device using ohci_hcd and address 5
usb 2-2: not running at top speed; connect to a high speed hub
usb 2-2: configuration #3 chosen from 1 choice
usb 2-2: USB disconnect, address 5
```

- Finally `$ lsusb` outpout make sure the firmware has been uploaded to the JTAG pod: should print the `:0008` as in the following output

```
Bus 001 Device 022: ID 03fd:0008 Xilinx, Inc.
```

- Note: that you may want to also fix the *udev* installation layout:

```
# mv /etc/udev/rules.d/xusbdfwu.rules /lib/udev/rules.d/99-xusbdfwu.rules
```

Install usb driver

To install the USB userland driver for the Xilinx tools:

- Download http://git.zerfleddert.de/cgi-bin/gitweb.cgi/usb-driver?a=snapshot;h=HEAD;sf=tgz
- `# aptitude install libusb-dev` needed for compilation
- Untar, cd to the directory created and then: `make`
- Copy the resulting *libusb-driver.so* file to its destination:

```
$ cp libusb-driver.so /mnt/data1/ubuntu/usr/lib/libusb-xilinx-driver.so
```

- Uncomment the corresponding line in */mnt/data1/ubuntu/etc/env-xilinx.sh*
- Now Error message is : `WARNING:iMPACT:923 - Can not find cable, check cable setup`
- Allow the current user to access the JTAG cable:

```
sudo adduser `id -nu` lp
```

- Sorry but now you should reboot.

Note: Never use 2 programs that access the jtag at the same time ! (you will have to reboot the machine)

1.3.6 FPGA Editort

- Add needed libraries

```
$ /mnt/data1/ubuntu/usr/bin/xlx fpga_edito
libXm.so.3: cannot open shared object file: No such file or directory

# aptitude install libmotif3
$ /mnt/data1/ubuntu/usr/bin/xlx fpga_editor
libstdc++.so.5: cannot open shared object file: No such file or directory

$ cd tmp
$ wget 'http://fr.archive.ubuntu.com/ubuntu/pool/universe/g/gcc-3.3/libstdc++5_3.3.6-17ubuntu1_amd6
$ dpkg -x libstdc++5_3.3.6-17ubuntu1_amd64.deb .
# cp /tmp/usr/lib/libstdc++.so.5 /usr/local/lib
# ldconfig
$ rm -fr /tmp/usr/
$ /mnt/data1/ubuntu/usr/bin/xlx fpga_editor
Wind/U X-toolkit Error: wuDisplay: Can't open display
```

- Export display: create the */etc/gdm/custom.conf* file and restart gdm:

```
[security]
# Needed for Xilinx FPGA editor ???
DisallowTCP=false
```

- Allow connection

```
$ env DISPLAY=localhost:0.0 /mnt/data1/ubuntu/usr/bin/xlx fpga_editor
Cannot register service: RPC: Unable to receive; errno = Connection refused

# aptitude install portmap
$ env DISPLAY=localhost:0.0 /mnt/data1/ubuntu/usr/bin/xlx fpga_editor
Symbol `_XtperDisplayList' causes overflow in R_X86_64_PC32 relocation
Symbol `_XtGetPerDisplayInput' causes overflow in R_X86_64_PC32 relocation
Warning!!: XKEYSYMDB environment variable is set to a wrong location
```

1.3.7 Desktop shortcut icons

Xilinx 11

- */Desktop/impact 11.desktop*:

```
[Desktop Entry]
Version=1.0
Type=Application
Terminal=false
Exec=/mnt/data1/ubuntu/usr/bin/xlx impact
Name=impact 11
Icon=/usr/share/pixmaps/apple-green.png
```

- */Desktop/ISE 11.desktop*:

```
[Desktop Entry]
Version=1.0
Type=Application
Terminal=false
Icon[en_US]=gnome-network-properties
Exec=/mnt/data1/ubuntu/usr/bin/xlx ise
Name[en_US]=ISE 11
Name=ISE 11
Icon=/usr/share/pixmaps/apple-red.png
```

Xilinx 12

- */Desktop/Xilinx-XTerm-12.desktop*:

```
#!/usr/bin/env xdg-open

[Desktop Entry]
Version=1.0
Type=Application
Terminal=false
Icon[en_US]=gnome-network-properties
Name[en_US]=Terminal Xilinx 12
Exec=@DSBASE@/xilinx12.sh gnome-terminal --profile=xilinx-12
Name=XTerm-xilinx-12
Icon=gnome-network-properties
```

- */Desktop/Impact-12.desktop*:

```
#!/usr/bin/env xdg-open

[Desktop Entry]
```

```
Version=1.0
Type=Application
Terminal=false
Icon[en_US]=gnome-network-properties
Name[en_US]=Xilinx Impact 12
Exec=@DSBASE@/xilinx12.sh impact
Name=Impact-12
Icon=gnome-network-properties
```

- */Desktop/XPS-12.desktop*:

```
#!/usr/bin/env xdg-open

[Desktop Entry]
Version=1.0
Type=Application
Terminal=false
Icon[en_US]=gnome-network-properties
Exec=@DSBASE@/xilinx12.sh xps
Name=XPS-12
Icon=gnome-network-properties
Name[en_US]=Xilinx XPS 12
```

- */Desktop/ISE-12.desktop*:

```
#!/usr/bin/env xdg-open

[Desktop Entry]
Version=1.0
Type=Application
Terminal=false
Icon[en_US]=gnome-network-properties
Name[en_US]=Xilinx ISE 12
Exec=env DISPLAY=localhost:0 @DSBASE@/xilinx12.sh ise
Name=ISE-12
Icon=gnome-network-properties
```

- */Desktop/SDK-12.desktop*:

```
#!/usr/bin/env xdg-open

[Desktop Entry]
Version=1.0
Type=Application
Terminal=false
Icon[en_US]=gnome-network-properties
Exec=@DSBASE@/xilinx12.sh xsdk
Name=SDK-12
Icon=gnome-network-properties
Name[en_US]=Xilinx SDK 12
```

1.3.8 Scan chain and load firmware

There is 2 ways to programs the FPGA:

- bitstream (*.bit*) for test
- flash memory (*.mcs*) that preserve firmware at next reboot

Please follow these steps:

- use LD_PRELOAD environement variable to shortcut the library calls which is done by using */mnt/data1/ubuntu/usr/bin/xlx*)
- double click on “boundary scan” and right clik in the window and select “initialize chain” that will scan the FPGA chain plug to JTAG, for instance on the LDA:
  - first flash memory: click “bypass”
  - second flash memory: click “bypass”
  - bitstream RAM memory: choose “bitstream file”
- click “assygn new configuration file” and “program”

Bitstreams are:

- DCC: *svn/calice/hardware/trunk/DCC/production/Projet_dcc_prod_hdl/ise/dcc_prod/top_dcc.bit*
- LDA main: *svn/calice/hardware/hardware/trunk/UK_firmware/LDA/firmware/ise/main/md2.bit*
- LDA hdmi: *svn/calice/hardware/trunk/UK_firmware/LDA/firmware/ise/hdmi/hdmi_module_top_level.bit*
- DIF ECAL: */mnt/data1/karmic64/home/calice/DIF2/DEV1/DIF2_DEV1_0.bit*

1.4 DataBase

1.4.1 Introduction

The database module is developped and maintained by Guillaume Baulieu at IPNL wich provide documentation available at https://lyosvn.in2p3.fr/ilc/wiki/ILCConfDB.

- A setup is a set of states. While each state contains the description of a subdetector configuration, the setup sumup the whole detector configuration.
- A state is a coherent set of 4 Configurations :
  1 LdaConfiguration, 1 DccConfiguration, 1 DifConfiguration and 1 AsicConfiguration.
  It contains all the configuration parameters of a sub-detector.
- Basically, a configuration is a list of objects :
  a LdaConfiguration is a list of LDA, a DccConfiguration is a list of Dcc and so on...
  When you want to store the LDA configuration of your sub-detector, you create a LdaConfiguration object and populate it with all your Ldas objects. The same can be done for DCC, DIF and ASICS.

The Daq, RunInfo and Job classes have been add by Laurent fo acquisition. Jobs are related to process on aquisition computer. Daq contain the XDaq configuration description.

Installation

```
$ ssh calice@poldhcp45
$ mkdir db && cd !$
$ wget ftp://lyoftp.in2p3.fr/baulieu/ILCConfDB-dev-0.8-1.i686.rpm
$ wget ftp://lyoftp.in2p3.fr/baulieu/ILCConfDB-lib-0.8-1.i686.rpm
$ wget ftp://lyoftp.in2p3.fr/baulieu/ILCOracle-dev-11.1-1.i686.rpm
$ wget ftp://lyoftp.in2p3.fr/baulieu/ILCOracle-lib-11.1-1.i686.rpm
$ su
# rpm -i ILCOracle-dev-11.1-1.i686.rpm ILCOracle-lib-11.1-1.i686.rpm \
         ILCConfDB-lib-0.8-1.i686.rpm ILCConfDB-dev-0.8-1.i686.rpm

/* python */
```

```
$ wget ftp://lyoftp.in2p3.fr/baulieu/ILCConfDB-pythonlib-0.9-1.i686.rpm
# yum install libxerces-c-3_1
# rpm -i ILCConfDB-pythonlib-0.9-1.i686.rpm
```

This will install the */usr/lib/libilcconfdb.so* shared library but also files into:

- */usr/lib/*
- */usr/include/*
- */usr/include/ILCConfDB*

Usage

Test file: *db.cc* (bellow is shorter)

```
#include <ILCConfDB/ILCConfDB.h>

int main() {
  int rc = 0;

  try{
    DBInit::init();//connection
    State* s = State::download("toto");
    DBInit::terminate();//disconnection
    delete s;
  }
  catch(Exception e){
    cout<<e.getMessage()<<endl;
  }

  return rc;
}
```

Compilation and execution:

```
$ g++ db.cc -l ilcconfdb
$ a.out

Composition of state TB_12-17_31_13_38_56_5_40_v01 :
1 lda
name= mac=5e:70:0c:d2:51:0a fw= hw=
2 dcc
name= addr=5e:70:0c:d2:51:0a:7 fw= hw=
name= addr=5e:70:0c:d2:51:0a:6 fw= hw=
15 dif
id=43 name=FT1010043 addr=5e:70:0c:d2:51:0a:6:5 enabled=1 type=DHCALDIF
sdhcal:
 AVDD_SHDN=0 DIF_IMON_GAIN=0 DVDD_SHDN=0 ENABLE_MONITORING=0 MASK=0
 MEMORY_DISPLAY_LIMIT_MAX=0 MEMORY_DISPLAY_LIMIT_MIN=0
 MEMORY_EFFICIENCY_LIMIT_MAX=0 MEMORY_EFFICIENCY_LIMIT_MIN=0
 MONITORED_CHANNEL=0 MONITOR_SEQUENCER=0 NUMERICAL_READOUT_MODE=0
 NUMERICAL_READOUT_START_MODE=0 POWER_ADC=0 POWER_ANALOG=0 POWER_DAC=0
 POWER_DIGITAL=0 POWER_SS=0 SLAB_IMON_GAIN=0 TIMER_HOLD_REGISTER=0

id=44 name=FT1010044 addr=5e:70:0c:d2:51:0a:6:4 enabled=1 type=DHCALDIF
sdhcal:
AVDD_SHDN=0 DIF_IMON_GAIN=0 DVDD_SHDN=0 ENABLE_MONITORING=0 MASK=0
MEMORY_DISPLAY_LIMIT_MAX=0 MEMORY_DISPLAY_LIMIT_MIN=0
```

```
MEMORY_EFFICIENCY_LIMIT_MAX=0 MEMORY_EFFICIENCY_LIMIT_MIN=0
MONITORED_CHANNEL=0 MONITOR_SEQUENCER=0 NUMERICAL_READOUT_MODE=0
NUMERICAL_READOUT_START_MODE=0 POWER_ADC=0 POWER_ANALOG=0 POWER_DAC=0
POWER_DIGITAL=0 POWER_SS=0 SLAB_IMON_GAIN=0 TIMER_HOLD_REGISTER=0
...
720 asic
difId=78 pos=1 type=HR2
hr2:
B0=250 B1=250 B2=250 CLKMUX=1 CMDB0FSB1=1 CMDB0FSB2=1 CMDB0SS=0
CMDB2FSB1=0 CMDB2FSB2=0 CMDB2SS=0 CMDB3FSB1=1 CMDB3FSB2=1 CMDB3SS=0
CTEST= DACSW=1 DISCRI0=1 DISCRI1=1 DISCRI2=1 DISCROROR=1 ENABLED=0
ENOCCHIPSATB=1 ENOCDOUT1B=1 ENOCDOUT2B=0 ENOCTRANSMITON1B=1 ENOCTRANSMITON2B=0
ENTRIGOUT=1 EN_OTAQ=1 HEADER=1 MASK0=0XFFFFFFFFFFFFFFFF
MASK1=0XFFFFFFFFFFFFFFFF MASK2=0XFFFFFFFFFFFFFFFF OTABGSW=1 OTAQ_PWRADC=1
PWRONBUFF=1 PWRONFSB0=1 PWRONFSB1=1 PWRONFSB2=1 PWRONPA=1 PWRONSS=1 PWRONW=1
QSCSROUTSC=1 RAZCHNEXTVAL=0 RAZCHNINTVAL=1 RS_OR_DISCRI=1 SCON=1 SEL0=1 SEL1=0
SELENDREADOUT=1 SELSTARTREADOUT=1 SMALLDAC=0 SW100F0=1 SW100F1=1 SW100F21 SW100K0=1
SW100K1=1 SW100K2=1 SW50F0=1 SW50F1=1 SW50F2=1 SW50K0=1 SW50K1=1 SW50K2=1 SWSSC=7
TRIG0B=1 TRIG1B=0 TRIG2B=0 TRIGEXTVAL=0
PAGAIN_VALUES= 115 140 136 136 120 111 119 143 125 121 128 135 117 128 118 133 116 130 134
135 146 149 117 132 135 134 117 116 131 130 139 112 110 126 147 119 120 135 118 123 112 129
135 125 128 129 120 124 132 155 119 122 131 132 129 130 135 130 147 128 135 130 136 127

difId=78 pos=25 type=HR2
hr2:
B0=250 B1=250 B2=250 CLKMUX=1 CMDB0FSB1=1 CMDB0FSB2=1 CMDB0SS=0
CMDB2FSB1=0 CMDB2FSB2=0 CMDB2SS=0 CMDB3FSB1=1 CMDB3FSB2=1 CMDB3SS=0
CTEST= DACSW=1 DISCRI0=1 DISCRI1=1 DISCRI2=1 DISCROROR=1 ENABLED=0
ENOCCHIPSATB=1 ENOCDOUT1B=1 ENOCDOUT2B=0 ENOCTRANSMITON1B=1 ENOCTRANSMITON2B=0
ENTRIGOUT=1 EN_OTAQ=1 HEADER=25 MASK0=0XFFFFFFFFFFFFFFFF
MASK1=0XFFFFFFFFFFFFFFFF MASK2=0XFFFFFFFFFFFFFFFF OTABGSW=1 OTAQ_PWRADC=1
PWRONBUFF=1 PWRONFSB0=1 PWRONFSB1=1 PWRONFSB2=1 PWRONPA=1 PWRONSS=1 PWRONW=1
QSCSROUTSC=1 RAZCHNEXTVAL=0 RAZCHNINTVAL=1 RS_OR_DISCRI=1 SCON=1 SEL0=1 SEL1=0
SELENDREADOUT=1 SELSTARTREADOUT=1 SMALLDAC=0 SW100F0=1 SW100F1=1 SW100F21 SW100K0=1
SW100K1=1 SW100K2=1 SW50F0=1 SW50F1=1 SW50F2=1 SW50K0=1 SW50K1=1 SW50K2=1 SWSSC=7
TRIG0B=1 TRIG1B=0 TRIG2B=0 TRIGEXTVAL=0
PAGAIN_VALUES= 127 122 144 122 118 128 122 126 133 145 113 126 130 134 111 122 137 136 129
140 139 118 119 126 115 130 133 124 124 124 138 128 128 125 128 113 120 147 123 118 133 125
125 123 127 120 128 133 127 130 139 126 131 129 136 144 126 125 134 135 120 145 121 136
...
```

Bugs or troubles

- setups still not used (only one state by setup)
- firmware and hardware fields not used
- ID Dif's key parameter is not documented.
- SDHCAL DIF parameters no used (all set to 0) ask Laurent and Christophe about it.

Error messages

There is no particular logging system. See explanations of the errors messages encountered bellow:

- `ORA-12514: TNS:listener does not currently know of service requested in connect descripto`
  This message came directely from the Oracle server wich is not operational.

1.4.2 Install on other OS

We consider that the above installation was done on the `calice@poldhcp45` account of a SLC55 machine.

```
$ mkdir ~/caliceDB
$ mkdir ~/caliceDB/lib
$ mkdir ~/caliceDB/include

$ cd ~caliceDB
$ scp calice@poldhcp45:/usr/lib/libilcconfdb.so lib
$ scp -r calice@poldhcp45:/usr/include/ILCConfDB include

$ files="/usr/lib/libclntsh.so /usr/lib/libnnz11.so /usr/lib/libocci.so /usr/lib/libociicus.so /usr/lib/
$ for f in $files ; do scp calice@poldhcp45:$f lib; done

$ files="/usr/include/ldap.h /usr/include/nzerror.h /usr/include/nzt.h /usr/include/occi.h /usr/include/
/usr/include/occiCommon.h /usr/include/occiControl.h /usr/include/occiData.h /usr/include/occiObjects.h
/usr/include/oci1.h /usr/include/oci8dp.h /usr/include/ociap.h /usr/include/ociapr.h /usr/include/ocide
/usr/include/ocidem.h /usr/include/ocidfn.h /usr/include/ociextp.h /usr/include/ocikpr.h /usr/include/oc
/usr/include/ocixstream.h /usr/include/odci.h /usr/include/oratypes.h /usr/include/ori.h /usr/include/or
/usr/include/orl.h /usr/include/oro.h /usr/include/ort.h /usr/include/xa.h"
$ for f in $files ; do scp calice@poldhcp45:$f include; done
```

The target machine have a 64 bits architecture :

```
# apt-get install libc6-dev-i386

$ g++ -c db.cc -o db.o -Iinclude

$ $ g++ db.o -o db -Iinclude -Llib -l ilcconfdb -fPIC
/usr/bin/ld: skipping incompatible lib/libilcconfdb.so when searching for -lilcconfdb
/usr/bin/ld: cannot find -lilcconfdb
collect2: ld returned 1 exit status
```

This will install the */usr/lib/libilcconfdb.so* shared library but also files into:

- */usr/lib/*
- */usr/include/*
- */usr/include/ILCConfDB*

1.4.3 Ecals parameters

Ecals parameters

Mail from Stéphane Callier:

```
Explications :
- en colonne G, les bits ecrits en rouge sont fixés et ne sont pas disponibles pour l'utilisateur
(les 3 cases jaunes etaient fixées dans la 1ere version et n'apparaissent pas encore dans le GUI de Muri
- en colonne H, le nom du parametre
- en colonne C, le nombre de bits
- en colonne J, explications pour le decodage lorsqu'il y a plusieurs bits et que l'affichage est du tex

Ce fichier est valable pour les ASUs ECAL avec les puces SPIROC2 et SPIROC2A.
La seule difference entre ces 2 puces, concerne la ligne 12 (en rose) qui n'est disponible que dans SPIR
```

```
Si tu as des questions...
A+
Stephane
```

1.5 GUI

1.5.1 Introduction

ECAL-GUI-PROTO-Nov2011.pdf ; ECAL-GUI-PROTO-Nov2011.pptx

This GUI is both a hight and low (debug) level tool using a unique library to access hardware. This GUI is a whole DAQ human interface in developpement at LLR only. This GUI is a debugging tools not designed for hight rate acquisition, not an hight rate acquisition system. There is an equivalent official project call XDaq leads by LPNL using the XDaq technology. Both of theses GUI will be based on the *libLDA.a* library to access hardware.

The GUI must be split into 4 logicals modules, able to work alone. This is a must be in order to decrease the whole project complexity.

- Framework
- Database (edit `.asu` and `.lda` files)
- Electronical functionalities (manage board one by one)
- Physical functionalities (manage a set of boards)

1.5.2 Framework module

Framework should replace the current windows manager we use to open X terminals and to type unitary tests command lines. It must provide a way to manages several threads. This may be done using the *libLDA/include/misc/thread.h* specifications.

Tab manager (todo first)

The tab manager have to provide a way to assemble different tools allowing them to be embedded by the XDaq next generation interface. It may be done using the Root technology which is supported by XDaq.

Log tab

This should display the */var/log/libLDA.log* file. It may be done by a thread executing the `tail -f` system command.

1.5.3 Database module

We do not use the Mysqli/Oracle IPNO's Database. It is a must be that we provide a first version using only the 2 first historical configuration file formats. We do that in order to provide compatibility with all the tools we used during the DAQ built.

ASU configuration Tab (todo first)

The purpose of the ASU configuration Tab is to parse, modify and serialise the ASU configuration files. The ASU configuration format is based on the ASIC configuration file format used by the LAL wich provide the ASICs. The format we used to load ASU configuration with python tool is a simple concatenation of the ASIC files. The ASU file's mime type may be `.asu`

LDA configuration Tab

The purpose of the ASU configuration Tab is to parse, modify and serialise the LDA configuration files. The LDA configuration format is defined by the C++ parser used in the firsts released of FIFO ang RNG test tools. All the C++ code developped as LLR, up to the *libLDA.a* back end, provide compatibility with this format. The ASU file's mime type may be `.lda`

1.5.4 Electronical functionalities

The purpose of this module is to replace the existing python tool. This module must enhance this python software providing a single interface to select an electronic board. This selection must silently provide the LDA and DCC port parameters for all the basic queries.

tcpdump Tab

We do not interpret the basic query's replies. This is a must be in the first release because the libLDA is not designed for. However all the uni-direction queries should have to be developped in the libLDA (this not the case now). As for the log tab, It may be done by a thread executing the `tcpdump -xx -s 1034 ether host 1:2:3:4:5:6` system command.

Selection Tab

The purpose of this tab is to allow electronic board selection using or not a `.lda` read only file.

LDA Tab

The purpose of this tab is to replace the existing *GUI_LDA.py* python windows. In the first release, it will only send queries and ignore the LDA replies.

DCC Tab

The purpose of this tab is to replace the existing *GUI_DCC*.py python windows. In the first release, it will only send queries and ignore the DCC replies.

DIF Tab

The purpose of this tab is to replace the existing *GUI_DIF.py* python windows. In the first release, it will only send queries and ignore the DCC replies.

CCC Tab

The purpose of this tab is to replace the existing tools we use to drive the DCC serial port. In the first release, it will only send queries and ignore the DIF replies. In this first version, the way to address the CCC will be compiled into the *libLDA.a*.

7 Other Tabs (todo first)

The purpose of this tab is to replace the other existing python windows:

- *GUI_DIF2.py*
- *GUI_DIF-borb.py* (todo first)
- *GUI_DIF-flash.py*
- *GUI_DIF-rndgen.py*
- *GUI_DIF-debugfifo.py*

- *GUI_DIF-ram1.py*
- *GUI_DIF-slab.py*

In the first release, it will only send queries and ignore the DIF replies.

1.5.5 Physical functionalities module

Selection Tab (todo first)

The purpose of this tab is:

- to select `.lda` files we want to drive.
- to provide the command line arguments used by *libLDA.a*
  note: there is 3 operation mode (soft, ilc and test-bench). today only the icl mode is working.
- to provide buttons to launch hight queries (dump, ping, fifo, rng, config, drive)
- to drive the CCC for ilc mode.
  It may be done in a similar way as the *libLDA/tools/daq.c* unit test do.
  In this first version, the way to address the CCC will be compiled into the *libLDA.a*.
- to provide a stop button.
- to provide a record/replay button (advanced version).

The command line argument are given by the *libLDA/src/run* executables:

```
libLDA# ./exemple conf/lyonCC.lda -m4 -n -i 1000 -d 50000 -S2 -fv DEBUG -h
Usage: ./exemple [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, db
  -p ETH_PORT | --eth-port=ETH_PORT    override the ethernet port on the host
                                       specified in the config file (eth0, ...)
  -a 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
  -m | --mode                          set the mode (default: 8)
                                       among 8 (ilc-manual), 0 (ilc-auto) or 4 (beam-test)
  -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-trg                      set trigger frequency (unit is Hertz)
  -U | --freq-usr                      set user read out frequency (unit is Hertz)
```

Slow control monitoring Tab

The purpose of this tab is to display the *libLDA.a* buffer occupancy and if packet have been dropped by the kernel. It may be done by a thread like the existing one in the *libLDA/tools/daq.c* unit test. The thread must adapt is work to for any hight level query (dump, ping, fifo, rng, config, drive). Such a functionalities already exists in the *libLDA.a*.

DIF readout monitoring Tab

The purpose of this tab is to display the *libLDA.a* output DIF buffers.
It should provide a way to select:

- the readout data file to parse
- a binary executable to parse the file

It may be done by a thread executing the `tail -fn0 libLDA/data/readoutfile | libLDA/src/run/parser` system command.

1.5.6 Conclusion

This project is:

- complex and need to be defined precisely.
- not defined to deal with the IPNL database and XDaq frameworks.
- a debugging tools not designed for hight rate acquisition.

Cost estimation:

Task	Man power cost	advancement
Framework module		
Tab manager	>2 weeks	to define
Log Tab	1 week	todo
Database module		
ASU configuration tab	2 week	done for ASIC
LDA configuration tab	2 week	todo
Electronic module		
tcpdump tab	1 week	todo
Selection tab	1 week	todo
LDA tab	1 week	todo
DCC tab	1 week	todo
DIF tab	1 week	todo
CCC tab	1 week	todo
Other tabs	4 weeks	todo
Physical module		
Selection Tab	2 weeks	todo
Slow control monitoring tab	2 week	todo
DIF readout monitoring tab	1 week	todo
Sumary		
All tasks	>22 weeks	todo

Note: There is only one developper notified to work halt time for this project (the project finalisation is estimated to 11 months by following this functional specifications, 2 years else)

Note: There is only two setup in LLR and no simulator. This lake of hardware to develop and test software will have an unknown but sensible cost to the developpement time. (We should have several ASU in august).

one developper notified to work halt time for this project (the project finalisation is estimated to 11 months by following this functionnal specifications, 2 years else)
The *libLDA.a* developper claims for a project manager to:

- drive this project according to such a specification document, not the custumer's needs
- ensure the GUI developper will be given all support needed to link the *libLDA.a*
- ensure the project developpers use a software development method (SVN sources sharing, and tests, ...)

1.5.7 Custumer's needs

The custumers are identified as:

- Vincent Boudry: simple whole acquisition, link leds (not possible with the actual libLDA)
- Rémy Cornat: globale interface with only one driver object (not possible for complexity reason), replace python
- Romane Poeschl (LAL): ?

```
Nous avons 5-6 besoins par ordre de priorité: (on peut discuter sur la
pertinance de l'ordre 2-3).

   1. Une libLDA (en C/C++) qui assure une interface aisé à la DAQ
          * Connection à 1 LDA
          * Chargement de configs
          * Envoi de toutes les commandes de bas niveau ou de haut niveau
          * Gestion des tampons de réception des données Ethernet
          * Vérif des CRC, déencapsulage & mise en tampon individuels
            DIF par DIF & évévnement par événement dans une mémoire partagée
                o fonction de timeout (actuellement par DIF -> par evt ?)
          * ...
   2. Une GUI+libLDA qui permette de modifier la configuration d'UNE DIF
      ECAL, de lancer une acquisition simple et de relire les données
      (format binaire).
          * Décodage d'un fichier de config SPIROC -> format humain
            modifiable (GUI): (MC)
          * Recodage de la config SPIROC -> fichier (MC)
          * Interfacage avec libLDA: quasiment fait (NR & MC)
          * séquencage de l'acquisition (~ copie du code «exemple» de
            NR): ~ fait ?
          * Commentaires:
                o ce travail est urgent et devrait être terminé très
                  prochainement ?
                o Quel nom lui donner ?
   3. Interfaçage de SPIROC dans la base de données de configuration de
      Lyon.
      La BD de Lyon gère les configs dans un format humain.  Le code de
      Decodage/recodage SPIROC écrit par Muriel doit être mis en place
      dans les outils de gestion de Guillaume Baulieu (en vacances
      actuellement).
      C'est quasiment un préalable à l'utilisation de XDAQ, sauf à gérer
      les configurations à partir de fichiers.
   4. Ajout à l'outil 1) du décodage des données du SPIROC de Thibault
      Frisson pour une visualisation d'une certain nombre d'éléments de
      débug.
   5. Développement d'un outils de débugage pour un large set-up
          * Reprise de l'outils 1) et extension à plusieurs DIFs
                o Gestion de multiples configs
                o Gestion «dynamique» des connections de LDA/DCC/DIF
                  avec indicateur de l'état des éléments
          * Ajout de fonctionnalité pour effectuer un séquencage manuel.
   6. Eventuellement: Développement d'un afficheur d'histogramme mieux
      fait que celui actuellement dans XDAQ.
      Il existe du code par ailleurs (DHCAL US), et un stagiaire à Lyon
      était censé travailler dessus et ce n'est pas urgent.
```

1.6 Online v2

1.6.1 Introduction

Spécifications :

- GASOLine.odp
- Spécifications GASOLine.pdf
- IEEE_NARVAL.pdf

Divide and conquer: use the modular conception historically introduce by electronics.

- Physical detector
- Electronic DAQ
- Online server
- Slow control software
- DAQ software
- Database (included or not into the SLC module)

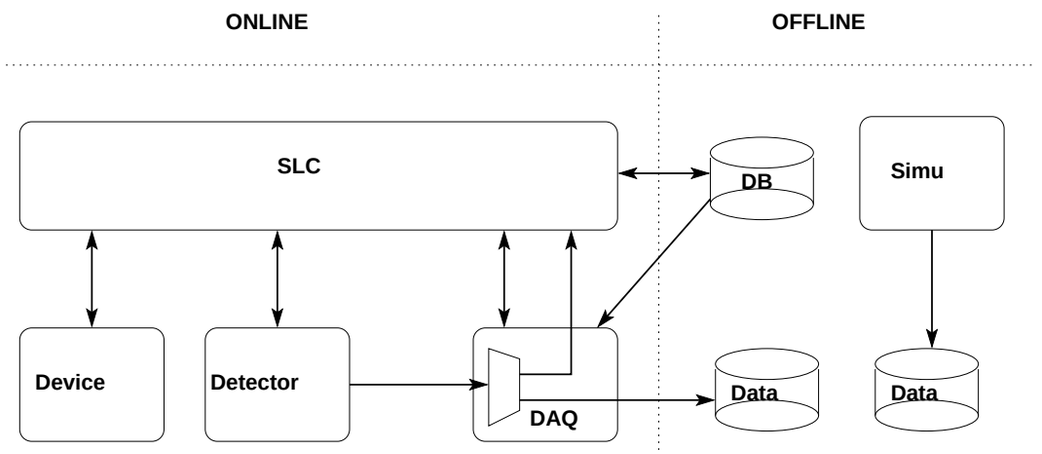

Figure 1.1: Generic architecture

1.6.2 Electronic DAQ

- The electronic DAQ is the layer to interact with the detector composed by ASU cards.
- The electronic DAQ is composed by DIF, potentials DCC, LDA and CCC cards.
- The electronic DAQ is driven by a unique CCC card managing the global state machine.
- The electronic DAQ is accessible via an undefined number of LDA cards (about 3).
- The electronic LDA cards cannot be access by two different machine at a time.
- The electronic LDA cards offer only one ethernet physical port to software.
- The electronic LDA's software ports cannot be slitted into data and control IP ports by a switch.

1.6.3 Online server

The goal of this new version is to allows 2 programs (data acquisition and slow control) to run in the same time.

- The online server act as an IP switch providing both slow control and data acquisition output cables. **Note:** the reason to upgrade from a library to a server is to divide by 2 the memory and cpu resources of the PC that drives the LDA.
- The server will embed at less 2 thread, one for data acquisition and one for slow control.
- The server will duplicate internally all the electronic DAQ cards registers.
- The slow control API will evolve up to SOAP API.
- The data acquisition API will allows TCP pushing (one more thread) or shared memory buffer Pooling.
- The Online server will not anymore connect to the database to read the elctronic DAQ configuration, this information will be given by the slow control software or else by a local XML file.

Todo:

- Logger
- Pcap injector (to test using tcpdump)
- Callback-function that listeen (to test with the injector)
- SLC Thread drive by socket (to test with telnet)
- SLC queries for internal Tree configuration (to test with telnet)
- SLC query for detector configuration
- DAQ output

Syslog and Syslogd

Syslogd server

```
# apt-get install sysklogd
```

file */etc/sysklogd.conf* or */etc/default/syslogd*

```
SYSLOGD="-r -m0"
```

file */etc/syslog.conf* or */etc/rsyslog.conf.d/50-default.conf*

```
local4.debug         /var/log/GASOline.log
```

```
# /etc/init.d/sysklogd restart
```

Syslog client file */etc/syslog.conf* or */etc/rsyslog.conf.d/50-default.conf*

```
local4.debug         /var/log/gasoline.log
local4.debug         @polntnr
```

```
# /etc/init.d/rsyslog restart
or
# /etc/init.d/syslog restart
```

```
$ cd gazoline/log
$  ./utlog -f local4 -s debug
```

1.6.4 Slow control software

- By using dedicated ethernet cables for the slow control, we can imagine a unique slow control GUI running on another host and managing all the Elecronic DAQ.
- The main advantage of a dedicated SLC soft among XDAQ is a friendly approach to the database (a GUI).
- Indeed, the slow control program will evolve around a python tree-view imported from and exportable to the DB.
- The slow control program should also drive the DAQ software, we guess using SOAP too.

1.6.5 DAQ software

- The DAQ software is mainly an event builder that serialise all the events from electronic DAQ into a physical output format.
- The DAQ's output must be both store on hard-disc and send to a real-time monitoring software.
- We are wandering if or if not the DAQ software should have access or not to the database.
- We must define a policy for lost packets that will not penalise the electronic DAQ neither the software DAQ.

1.6.6 Data base

- Need to enter the ECAL detector parameters into the oracle DB.
- Is the DB for configuration and for run acquisition storage the same ?

Chapter 2

Procedure

This section provides know-how procedures the CALICE team share.

- **Ideal topology**:
  PC (or ODR made by UK)
  - LDA: Link Data Agregator (made by UK)
  - (x2)
    - DCC (made by LLR)
    - (x10)
      - DIFF (made by LLR or LAPP)
      - (x9)
- **Power supply:**
  - ASU: ?
  - CCC: home made Alim (UK)
  - DCC: 5v, from 3mA 18W max Alim
  - LDA: 12V-0.5A and 5.5V-1A (ok from PC Alim)
  - DIF ECAL (Xilinx): 3.3v-0.5A (ok from PC Alim)
  - DIF DHCAL (Altera): 5v-0.5A (ok from PC Alim)

  Note: PC Alim is switched on by linking the green and black input of main motherboad cable.

- 4 kinds of links are used (identified by the Ethernet Type 809, 810, 811):
  - PC to CCC: RS232 (working but not used) reach throught an ACKSYS COM/ETH convertor.
  - PC (or ODR) to LDA: Ethernet Gigabit on rj45 (Ethernet but not IP)
  - CCC to LDA: HDMI plugged from dedicated LDA Mezzanine
  - LDA to DCC: Home made Ethernet over HDMI (no MAC address, see the documentation section)
  - DCC to DIF: Home made Ethernet over HDMI (no MAC address, see the documentation section)
  - DIF to ASU: 100+ pins connector

  Notes:
  - HDMI runs at 50 MHz that implies 40 Mb/s due to 8b/10b Encoding.

  - Question is: may LDA be replaced by standards Eternet Switch ?
  - With the actual firmware, LDA needs the CCC to work (6 red leds).
  - `uint16_t ODR_Data_and_PAD[0];` is a meta attribute for the following data in memory.
  - ODR (not used) state of the art (odr_server) is describe in *svn/calice/online-sw/trunk/doc/odr-calice.txt*.

- **Hosts**
  - `poltestcalice`: old DHCAL test bed (svn, xilinx 11, pyserdiag)
  - `polcaldaq`: DHCAL+ECAL test bed (svn, xilinx 11+12, pyserdiag, optical eth)
  - `poldhcp45`: SC55 for soft devel (svn, libLDA)
  - `poldhcp54`: new DHCAL test bed (svn, xilink 11+12, pyserdiag) ?
  - `pollinmc2`: SC55 for soft devel (svn, libLDA, XDaq)
  - `polcalslc`: SC55 pizza box for XDAQ devel (svn, libLDA, Xdaq)
  - `pollinbtc`: ECAL cosmic test bed (svn, libLDA, pyserdiag via virtual machine)
  - `polntnr`: Nicoas Roche's laptop
  - `polsmtp`: smtp
  - `polynet`: intranet
  - `polzope`: intranet
  - `llrforge`: */pwhash/*
  - `llroffice`: *ELog*
  - `polui0[1-7]`: gateways
  - `lyoac[19,20,26]`: IPNL gateways reachable only via `polui0x`
  - `lxplus.cern.ch`: Cern gateway

2.1 poltst0

2.1.1 Introduction

`poltst0` is a new server for DHCAL test bench. It is a dual boot SLC5.5 and UBUNTU LTS 10.04. It also embed a virtual WINDOWS.

Note: take care at kernel update to check grub use sda7 as root in file */boot/grub/menu.lst*:

```
title Ubuntu 10.04.2 LTS, kernel 2.6.32-31-generic
uuid a15e9963-3edf-4780-a876-aab29e6c6578
kernel /vmlinuz-2.6.32-31-generic root=/dev/sda7 ro quiet splash
initrd /initrd.img-2.6.32-31-generic
quiet
```

2.1.2 Partitions

```
# fdisk -l

Disk /dev/sda: 160.0 GB, 160000000000 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1               1          18      144553+  de  Dell Utility
/dev/sda2              19          30       96390   83  Linux
/dev/sda3   *          31          42       96390   83  Linux
```

```
/dev/sda4               43       19452   155910825    5  Extended
/dev/sda5               43          91      393561   83  Linux
/dev/sda6               92        2523    19535008+  83  Linux
/dev/sda7             2524        4955    19535008+  83  Linux
/dev/sda8             4956        7387    19535008+  83  Linux
/dev/sda9             7388        7995     4883728+  82  Linux swap / Solaris
/dev/sda10            7996       19452    92028321   83  Linux

Disk /dev/sdb: 250.1 GB, 250059350016 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1               1       30401   244196001   83  Linux
```

sda2	SLC	ext2	/boot
sda3	Ubuntu	ext2	/boot
sda5	unused		/boot
sda6	SLC	ext3	/
sda7	Ubuntu	ext3	/
sda8	unused		/
sda10	all	ext3	/data1 (homes)
sdb1	all	ext3	/data2 (unused)

2.1.3 Shared partitions

From Ubuntu LTS 10.04

/etc/fstab:

```
/dev/sda10 /data1 ext3 defaults 0 2

# ls -l /home
/home -> /mnt/data1/ubuntu/home

$ ls -l /home/nroche/Applications
/home/nroche/Applications -> /mnt/data1/ubuntu/usr
```

2.1.4 Dual boot

Scientific Linux Cern 5.5

Note that XDAQ recommand 32bit OS and that SLC5.5 amd64 have troubles to move the mouse.

- Download the boot.iso.

  ```
  $ md5sum /dev/sr0
  eadcf980712389a687beb5df1230207c  /dev/sr0
  ```

- Unseselect ipv6 support
- Use HTTP repository:
  - *linuxsoft.cern.ch*
  - */cern/slc55/i386*
- Select "Server"
- At reboot: use "auth config" to desable "kerbeos" and "firewall" to disable "SELinux"

Ubuntu LTS 10.04

- Download the amd64's iso

```
# aptitude install nvidia-current
```

- Install Grub for dual boot (insted of grub-pc):

```
# aptitude purge grub-pc
# rm /boot/grub/*
# aptitude install grub
# update-grub    (generate /boot/grub/boot.lst)
# grub-install   (install /boot/grub/stage[12])
```

- Create the file */boot/boot.lst*:

```
default 1
timeout 3
color cyan/blue white/blue

title SLC5 64
configfile (hd0,1)/grub/menu.lst

title Lucid 64
configfile (hd0,2)/grub/menu.lst
```

- Tell grub to use it (you can also do it at boot):

```
# grub
> install (hd0,2)/grub/stage1 (hd0) (hd0,2)/grub/stage2 (hd0,2)/boot.lst
```

2.1.5 Windows XP

From Ubuntu LTS 10.04

- Install virtualBox:

```
$ cat /proc/cpu | grep svm (tell if cpu allow virtalization)
# aptitude install virtualbox-ose (Open Source Edition)
# aptitude install virtualbox-ose-dkms (for virtualization: Debian Kernel Management System)
# aptitude install virtualbox-guest-additions (for mouse caption between Linux and Windows)
$ VirtualBox
```

- use 892MB of RAM and 20GB for HD.
- Add serial port COM1 (unconnected).
- Iso are store here:

```
$ find /data2/VM
/data2/VM/xpkeys.txt
/data2/VM/shared
/data2/VM/iso/llrxpsp2.iso
/data2/VM/iso/office2003.iso
/data2/VM/snapshots
/data2/VM/vdisks/winxp.vdi
```

- *VBoxGuestAdditions.iso* should be automaticly provide into CD images.

- From VirtualBox, select the iso and run it into Windows.
- update Windows to SP3
- defined a shared folder using the GUI and copy into your SVN private key
- you should retrieve it from Windows into the "network favorite folders"

SVN into XP

- install PuTTy authentification agent (pageant)
- load the key and store it into the Windows format
- add a shortcut to pagent into the "Démarage" folder accessible from the "start/all applications" menu
- click once by session at the pagent icon on the right bottom and give your SVN password
- install TurtoiseSvn
- check out `svn+ssh://nroche@svn.in2p3.fr/calice/online-sw/trunk/pyserdiag`

Eclipse into XP

- install the SUN's java JRE.
- use yoxos to download Eclipse with following modules:
  - Eclipse C/C++ DevTools
  - PyDev Extension
  - PyDev for Eclipse
  - (Subclipse ?)
  - (Subversion Revision Graph ?)
  - (JavaHL 1.6.0 win32 binary ?)
- Load sources using "Import/General/Existing Project into Workspace"
- Open project using "Open perspective/PyDev"
- From "Configuration/Python Run" add *setup.py*:
  - Main:
    - Project: `pyserdiag`
    - Main Module: `{workspace_loc:pyserdiag/setup.py}`
  - Arguments
    - Program arguments: `build`
    - Working directory (Other): `${workspace_loc:pyserdiag}`

VisualC++ 2010 Express FR into XP

- Note that the English version may not match with a French Windows system.

2.2 poldhcp54

2.2.1 Introduction

`poldhcp54` is an old server reconfigured with SLC5.5. It is used for intensive tests using the C++ code.

2.2.2 Scientific Linux Cern 5.5

Note that XDAQ recommand 32bit OS.

- Download the boot.iso.

```
$ md5sum /dev/sr0
eadcf980712389a687beb5df1230207c  /dev/sr0
```

- Hit enter at welcom page
- Choose “English” language
- Choose “FR-latin9” keyboard
- Choose “HTTP” installation method
- Disable “IPv6 support”
- Enter
  - *linuxsoft.cern.ch*
  - */cern/slc55/i386*
- Partition using default layout
- Install grub on */dev/hda*
- Choose “Europe/Paris” using UTC
- Deselect “Workstation” and select “Server”
- Install “Updates” repository
- Choose “Customize later”
- At reboot:
  - Disable “Firerwall”
  - Disable “SELinux”

2.3 polntnr

2.3.1 Introduction

`polntnr` is a new server for DHCAL test bench. It is a dual boot SLC5.5 and UBUNTU LTS 10.04. It also embed a virtual WINDOWS.

2.3.2 Dual boot

Partitions

```
# fdisk -l

Disque /dev/sda: 500.1 Go, 500107862016 octets
255 têtes, 63 secteurs/piste, 60801 cylindres
Unités = cylindres de 16065 * 512 = 8225280 octets
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Identifiant de disque : 0x77e3ed41
```

```
Périphérique Amorce  Début        Fin      Blocs     Id  Système
/dev/sda1               1          5       40131   de  Dell Utility
/dev/sda2   *           6         30      194560   83  Linux
/dev/sda3              30         54      195584   83  Linux
/dev/sda4              54      35968   288474113    5  Etendue
/dev/sda5              54         79      194560   83  Linux
/dev/sda6              79        103      194560   83  Linux
/dev/sda7             103       2534    19529728   83  Linux
/dev/sda8            2534       4966    19529728   83  Linux
/dev/sda9            4966       7397    19529728   83  Linux
/dev/sda10           7397       9829    19529728   83  Linux
/dev/sda11           9829      10437     4881408   82  Linux swap / Solaris
/dev/sda12          10437      11652     9764864   83  Linux
/dev/sda13          11652      35968   195311616   83  Linux
```

sda2	Ubuntu	ext2	/boot
sda7	Ubuntu	ext3	/
sda8	unused		/
sda12	all	ext3	/tmp
sdb13	all	ext3	/opt

Grub

- Create the file */boot/boot.lst*:

```
default 0
timeout 5
color cyan/blue white/blue

title Lucid 64
configfile (hd0,2)/grub/menu.lst
```

- Tell grub to use it (you can also do it at boot):

```
# grub
> install (hd0,1)/grub/stage1 (hd1) (hd0,1)/grub/stage2 (hd0,1)/boot.lst
```

Shared partitions

/etc/fstab:

```
# /opt was on /dev/sda13 during installation
UUID=ecbd5107-be1f-464d-8056-f3792fbf14dd /opt            ext3    defaults        0       2
```

```
# cd
# tar -zcf /root/home.tgz /home
# mkdir -p /opt/ubuntu/home
# mkdir -p /opt/ubuntu/usr
# chmod 777 /opt/ubuntu/usr
# tar -zxf /root/home.tgz -C /opt/ubuntu/
# ln -s /opt/ubuntu/home /home
$ ln -s /opt/ubuntu/usr ~/usr
# mkdir /opt/pkg
# chmod 777 /opt/pkg
$ rsync -avv --progress calice@poldhcp45:/mnt/data2/pkg /opt/pkg
```

2.3.3 Ubuntu LTS 10.04

- Download the amd64's iso
- Install Grub for dual boot (insted of grub-pc):

```
# aptitude purge grub-pc
# rm /boot/grub/*
# aptitude install grub
# update-grub    (generate /boot/grub/boot.lst)
# grub-install   (install /boot/grub/stage[12])
```

- Get my documentation environment:

```
# aptitude install emacs cvs ssh
$ export CVSROOT=:ext:nroche@narval.hd.free.fr:/cvsroot
$ export CVS_RSH=/usr/bin/ssh
$ export export CVSEDITOR=vi
$ mkdir cvs && cd !$
$ cvs co calicei

# aptitude install texlive texlive-latex-extra netbpm latex2html auctex psutils
# aptitude install transfig bind9-host? (for narvali)
$ cd calicei
$ make

Unknown commands: pageTitle URIpath URIroot formatAvailability URIhost => look at
http://polntnr.in2p3.fr/narvali/outils/latex/page.html#SECTION00041000000000000000

# aptitude install apache2
# ln -s /var/www /htdocs
# chown .www-data /var/www
# chmod g+w /var/www
# adduser nroche www-data
$ ssh localhost
$ rm /var/www/index.html
$ cd cvs/calicei
$ make install
```

- VirtualBox:
- Install:

```
$ cat /proc/cpu | grep svm (tell if cpu allow virtalization)
# aptitude install virtualbox-ose (Open Source Edition)
# aptitude install virtualbox-ose-dkms (for virtualization: Debian Kernel Management System)
# aptitude install virtualbox-guest-additions (for mouse caption between Linux and Windows)
$ VirtualBox
```

- use 892MB of RAM and 10GB for HD.
- Iso are store here:

```
$ find /opt/iso
```

- *VBoxGuestAdditions.iso* should be automaticly provide into CD images.
- From VirtualBox, select the iso and run it into the virtual OS.

2.3.4 Scientific Linux Cern 5.5

Note that XDAQ recommand 32bit OS.

- Download the boot.iso.

```
$ md5sum /dev/sr0
eadcf980712389a687beb5df1230207c  /dev/sr0
```

- Hit enter at welcom page
- Choose “English” language
- Choose “FR-latin9” keyboard
- Choose “HTTP” installation method
- Disable “IPv6 support”
- Enter
  - *linuxsoft.cern.ch*
  - */cern/slc55/i386*
- Partition using default layout
- Install grub on */dev/hda*
- Choose “Europe/Paris” using UTC
- Deselect “Workstation” and select “Server”
- Install “Updates” repository
- Choose “Customize later”
- At reboot:
  - Disable “Firerwall”
  - Disable “SELinux”
- Enable kernel’s modules compilation (needed by virtualbox-guest-additions):

```
# yum install gcc kernel-devel.i686
# cd /usr/src/kernels/...
# make oldconfig && make prepare
```

2.3.5 Windows XP

- Note that the English version may not match with a French Windows system.

2.4 pollinbtc

2.4.1 Introduction

`pollinbtc` is the new cosmic host for EHCAL test bench. It is an SLC5.5 embedding UBUNTU LTS 10.04 and WINDOWS as virtual machines.

- eth0: BCM5761 (pci)
- eth1: BCM5764M (mother board)

```
$ lsb_release -a
LSB Version: :core-4.0-ia32:core-4.0-noarch:graphics-4.0-ia32:graphics-4.0-noarch:printing-4.
Distributor ID: ScientificCERNSLC
Description: Scientific Linux CERN SLC release 5.6 (Boron)
Release: 5.6
Codename: Boron
```

Graphic card port

Please use the DVI port on the graphic card. After installation X Desktop will be display to the first port. If you use the HDMI port using an adapter, monitor go to sleep when we start X.

Raid 0 mirroring

At boot, press `F10` and next `^C` to enter the SAS utility. Using this utility, create an IS volume with disk 2 and 3. This raid 0 mirror will be useful to quickly dump data from acquisition.

Partitions

```
# fdisk -l
...
```

sda1	ext2	/boot
sda2	ext3	/
sda3	swap	
sda5	ext3	/home
sda6	ext3	/var
sda7	ext3	/usr/local
sdb1	ext2	/opt?

Backups and other

- */usr/local/calice/BACKUP_2011*: old stuffs
- */usr/local/calice/pkg*: Software packages (mostly electronic tools)
- */usr/local/calice/vm*: Virtuals machines
- */usr/local/calice/nroche*: SVN configuration
- */usr/local/calice/gastaldi*: SVN configuration
- */usr/local/calice/xilinx11*: XILINX11 software
- */usr/local/calice/xilinx12*: XILINX12 software

2.4.2 Scientific Linux Cern 5.5

Note that XDAQ recommand 32bit OS.

- Download the boot.iso.

```
$ md5sum /dev/sr0
eadcf980712389a687beb5df1230207c  /dev/sr0
```

- type `linux text` at welcome page (graphic card doesn't seems to work)
- Choose "English" language
- Choose "FR-latin9" keyboard
- Choose "HTTP" installation method
- Unseselect ipv6 support
- Use this HTTP repository:
  - *linuxsoft.cern.ch*
  - */cern/slc55/i386*
- Choose "text mode"
- Install grub boot loader in MBR: /dev/sda/ (else boot fails)
- Choose "Europe/Paris" using UTC
- Select "Workstation"
- At reboot: use "auth config" to desable "kerbeos" and "firewall" to disable "SELinux"

2.4.3 Virtual machine

Have a look in the tools section: VIRTUALBOX.

- Default machine folder (in preferences) : */usr/local/calice/vm*

Ubuntu LTS 10.04

- The iso we used is here: */usr/local/calice/vm/iso/ubuntu-10.04.2-desktop-i386.iso*

Windows XP

We re-use the previous XP virtual machine.

2.4.4 SVN stuffs

Have a look in the approach section: SUBVERSION

Note: there is a security process that re-arrange PATH on SLC5.6. If we want to use the icons on Desktop for SVN, in */etc/profile.d/zzz_hepix.sh*:

```
#!/bin/sh
return # please add this line to use svn icons on desktop
...
```

2.4.5 Xilinx tools

2.5 Ethernet

2.5.1 Introduction

2.5.2 Fast command

Ethernet use 8b10b (like usb) using several patterns for the same value allowing differential coding. Meta patterns called “K characters” are use to indicate start and end of frame. The called “DAQ v2” use its own “k character” to embed fast command (a single 10b byte) in the trames (LDA<->DIF links only).

Note that this coding (no returns to 0) allow to embed a clock, but this is not used in Calice project (which use a CCC card).

Display input and output rate every 4 seconds:

```
$ ifstate -nb -i eth6 4
```

2.5.3 Gigabit

LDA are delivered with Optical Mezzanines without SGMI for rate négatiation. In order to change it for an Electrical Mezzanine, we must ensure that is is “SFP/1250” compliant, but not “SFP/10-100-1250” for instance.

```
# ethtool eth6
Port: FIBRE
Link detected: yes
```

2.6 COM/ETH

2.6.1 Introduction

CCC interface use a female/female null modem cable. So as to command CCC (as all the other cards) form a single Ethernet cable, we provide such a device describe in this documentation.

2.6.2 Configuration

Via Ethernet using Telnet

Either configure the COMETH device by following the documentation sheet coming with device (as describe below). Otherwise don’t change its switches (no "admin mode" required), connect the device directly to an ethernet port of the PC, set the IP of your eth port to 192.168.1.XXX, eg: sudo ifconfig ethX 192.168.1.42 up Then telnet to the device: `telnet 192.168.1.20`

Afterwards simply configure its parameters: `> login root` ... (see below).

Via serial link using Screen

This show how we push the configuration (needed to talk to the CCC) with an USB/Serial adaptater:

- Try `stty -F /dev/ttyUSB0 -a`
- Plug the USB serial adaptater and retry the above command. (this meed the `pl2303` kernel module to be load)
- Login as root using `screen` :

```
$ screen /dev/ttyUSB0 2400
(type return)
> login root
password: root
Ok
> set default
> set net ip 192.168.1.20
> set serial baudrate 115200
> set serial mode raw         (no futur distant configuration)
> show serial port
 2300
   ---> This is the TCP port you need to use to talk to the
        remote serial dev (eg. CCC)
 > save
 > reset
C-a+c
$
```

2.7 Purchase Orders

2.7.1 Introduction

Contact Mr Janssens (or Mme Boisgard).

They treat big size purchase orders else they wait for a batch from US and take into. In any case, we need to phone them and ask them for a devis then for delays.

```
François JANSSENS
Ingénieur Commercial
Département Réseaux & Télécoms
Téléphone   : +33 (0)1 41 22 10 06
Mobile      : +33 (0)6 08 75 57 18
Télécopie   : +33 (0)1 41 22 10 01
Courriel    : francois.janssens@elexo.fr

Assistante Commerciale : Catherine BOISGARD
Téléphone   : +33 (0) 1 41 22 10 09
Courriel    : catherine.boisgard@elexo.fr

Société Elexo
20 Rue de Billancourt, 92100 Boulogne
Site web       : www.elexo.fr
```

2.7.2 Exemple Mail

```
Objet : Modules SFP + Carte optique

Bonjour,

Je vous avais contactes voici quelques mois pour l'achat de 3 puis 2
modules SFP, puis pour un devis. Cette fois-ci, je vous contacte pour un
devis en vue d'une commande tres probable.

Nous serions interesses par 3 modules SFP identiques a notre precedente
commande, a savoir : IMC 808-39010 SFP cuivre 1 Gbps modele mono-vitesse
(1250 Mbps) avec interface SERDES, connecteur RJ45, distance 100m max, RoHS.
```

```
En plus, nous sommes interesses par 3 exemplaires similaire mais en
version optique. Pour information, le module optique de reference que
nous utilisons est un Finisar FTLF8519P2BNL-ES. Nous en cherchons donc
des modeles equivalents.

Enfin, si vous pouvez egalement nous procurer une carte PCI-express
ethernet 1Gb optique dual-port Intel Gigabit EF (ref E1G42EF) a l'unite,
nous serions interesses a ce qu'un exemplaire figure sur le devis. Si
vous n'etes pas en mesure de proposer ce produit au detail, merci de
nous rediriger vers un detaillant qui pourrait nous fournir, si vous en
connaissez.

Merci par avance !
Bonne journee,
```

Here is the cost estimation.

Chapter 3

Tools

The aim of this section is to help installing, configuring and using the CALICE software. Here are the hosts and addresses we use at the LLR:

- LCIO: see *svn/calice/online-sw/trunk/doc/build-lcio.txt*

```
$ cvs -d :pserver:anonymous@cvs.freehep.org:/cvs/lcio co -d v01-51 -r v01-51 lcio
$ mkdir build           # <-- create an out-of-source directory for the build
$ cd build
$ cmake ..              # check build options ( change options with: cmake -DOPTION=ON|OFF ..   or
$ make install
```

- BOOST:
  - on Ubuntu: `# apt-get install libboost-dev libboost-thread1.40-dev`
  - on SLC55: gcc no good

```
$ cd /tmp

$ wget http://ftp.gnu.org/gnu/glibc/glibc-2.6.tar.gz
$ mkdir tmp && cd !$
$ ../configure CFLAGS="-O3 -march=i686" --prefix=...
$ make

$ wget ftp://gcc.gnu.org/pub/gcc/infrastructure/gmp-4.3.2.tar.bz2
$ ./configure --prefix=...
$ make
$ make check
$ make install

$ wget ftp://gcc.gnu.org/pub/gcc/infrastructure/mpfr-2.4.2.tar.bz2
$ ./configure --prefix=...
$ make
$ make check
$ make install

$ wget ftp://mirrors.kernel.org/gnu/gcc/gcc-4.4.3/gcc-4.4.3.tar.gz
$ ./configure --prefix=... -with-gmp=... -with-mpfr=...
$ env LD_LIBRARY_PATH=.../lib make

$ wget http://sourceforge.net/projects/boost/files/boost/1.29.0/boost_1_29_0.tar.bz2/download
$ tar ...
$ cp boost_1_29_0/boost .../usr/include/.
```

- System C:

```
$ cd /tmp
$ scp calice@polcaldaq:Downloads/pkg/osci/systemc-2.2.0.tgz .
$ scp calice@polcaldaq:Downloads/pkg/osci/TLM-2.0.1.tgz .
$ scp calice@polcaldaq:Downloads/pkg/osci/sc-2.2-intrepid64.diff .
$ tar -zxf ...
$ less INSTALL
$ mkdir objdir && cd !$
$ mkdir -b ~/stow/osci   (todo before configure!)
$ ../configure --prefix=/usr/local/stow/osci
$ patch -p1 < ../sc-2.2-intrepid64.diff
$ make
# make install
# cd ..
# mv TLM-2009-07-15/ /usr/local/stow/osci/share
# cd /usr/local/stow/
# stow -v osci
```

- ftdidrv:

```
# aptitude install libftdi-dev
```

- test_usb_dif:

```
# aptitude install libpcre3-dev
```

3.1 lib pcap

3.1.1 Introduction

RAWSOCKETS have 2 disadvantages:

- they induce 1 system call by paquet
- change on the buffer size are global for the system

According to */linux/Documentation/networking/packet_mmap.txt*, PAQUET MMAP allocates a ring buffer on user space like MMAP do (version >=1.0 only). LIBPCAP uses natively the PAQUET MMAP allocation. However, according to `man 7 pcap-filter`, LIBPCAP also provides packet filtering handled by kernel. It is used by TCPDUMP for instance.

note: once packet read return, packet are no longer hosted in the ring buffer.

3.1.2 Python

try "pcap dpkt" on google.

DPKT allow to work offline on packet captured with TCPDUMP and saved as RAW:

```
# tcpdump -i eth6 -xx -vv -w out.bin
```

cf */svn/calice/online-sw/trunk/tests_eth_dif/rpcap.py*. **Note** that *rpcap.py* only works with messages length equal 16.

3.1.3 C

pcap_next_ex

Looking to the sources, it seems that packets are copied using `pcap_next_ex`: Yes, this means that, if the capture is using the ring buffer, using pcap_next() or pcap_next_ex() requires more copies than using pcap_loop() or pcap_dispatch(). If that bothers you, don't use pcap_next() or pcap_next_ex().

- file *libpcap-1.1.1/pcap.c*:

```
/*
 * Default one-shot callback; overridden for capture types where the
 * packet data cannot be guaranteed to be available after the callback
 * returns, so that a copy must be made.
 */
static void
pcap_oneshot(u_char *user, const struct pcap_pkthdr *h, const u_char *pkt)
{
struct oneshot_userdata *sp = (struct oneshot_userdata *)user;

*sp->hdr = *h;
*sp->pkt = pkt;
}

int
pcap_next_ex(pcap_t *p, struct pcap_pkthdr **pkt_header,
    const u_char **pkt_data)
{
struct oneshot_userdata s;

s.hdr = &p->pcap_header;
s.pkt = pkt_data;
s.pd = p;

/* Saves a pointer to the packet headers */
*pkt_header= &p->pcap_header;

if (p->sf.rfile != NULL) {
int status;

/* We are on an offline capture */
status = pcap_offline_read(p, 1, pcap_oneshot, (u_char *)&s);

/*
 * Return codes for pcap_offline_read() are:
 *   -  0: EOF
 *   - -1: error
 *   - >1: OK
 * The first one ('0') conflicts with the return code of
 * 0 from pcap_read() meaning "no packets arrived before
 * the timeout expired", so we map it to -2 so you can
 * distinguish between an EOF from a savefile and a
 * "no packets arrived before the timeout expired, try
 * again" from a live capture.
 */
if (status == 0)
return (-2);
else
return (status);
}

/*
 * Return codes for pcap_read() are:
 *   -  0: timeout
 *   - -1: error
 *   - -2: loop was broken out of with pcap_breakloop()
 *   - >1: OK
```

```
 * The first one ('0') conflicts with the return code of 0 from
 * pcap_offline_read() meaning "end of file".
*/
return (p->read_op(p, 1, pcap_oneshot, (u_char *)&s));
}
```

- file *libpcap-1.1.1/pcap-linux*:

```
/*
 * Special one-shot callback, used for pcap_next() and pcap_next_ex(),
 * for Linux mmapped capture.
 *
 * The problem is that pcap_next() and pcap_next_ex() expect the packet
 * data handed to the callback to be valid after the callback returns,
 * but pcap_read_linux_mmap() has to release that packet as soon as
 * the callback returns (otherwise, the kernel thinks there's still
 * at least one unprocessed packet available in the ring, so a select()
 * will immediately return indicating that there's data to process), so,
 * in the callback, we have to make a copy of the packet.
 *
!!! HERE (read this) !!!
 * Yes, this means that, if the capture is using the ring buffer, using
 * pcap_next() or pcap_next_ex() requires more copies than using
 * pcap_loop() or pcap_dispatch().  If that bothers you, don't use
 * pcap_next() or pcap_next_ex().
!!! HERE !!!
 */
static void
pcap_oneshot_mmap(u_char *user, const struct pcap_pkthdr *h,
    const u_char *bytes)
{
struct oneshot_userdata *sp = (struct oneshot_userdata *)user;

*sp->hdr = *h;
memcpy(sp->pd->md.oneshot_buffer, bytes, h->caplen);
*sp->pkt = sp->pd->md.oneshot_buffer;
}
```

pcap_loop

cf:

- `$man pcap_loop`.
- Coding a Simple Packet Sniffer
- French version of the above document

Exemple

ping

```
#include <stdio.h>
#include <stdlib.h>    // exit

#include <pcap/pcap.h>
#include <arpa/inet.h> // htons, uint16_t

pcap_t* ethConnect(char* ethernetCard, char* filter)
```

```
{
  static char errbuff[PCAP_ERRBUF_SIZE];
  pcap_t* rc = (pcap_t*)0;
  struct bpf_program          pcap_filter;

  memset(& pcap_filter, 0, sizeof(pcap_filter));

  // create a live capture handle
  if ((rc = pcap_create(ethernetCard, errbuff)) == (pcap_t *)0){
    printf("pcap_create: %s\n", errbuff);
    goto error;
  }

  // activate a capture handle
  if (pcap_activate(rc) != 0){
    printf("pcap_activate failed.\n");
    goto error;
  }

  return rc;
 error:
  // close a capture device or savefile
  pcap_close(rc);

  // free a BPF program
  pcap_freecode(&pcap_filter);

  return (pcap_t*)0;
}

/*
  >>
  08 10 00 02
  00 00 00 00 00 0b

  30 0f 00 00 00 00
  30 0e 00 00 00 00
  30 00 00 00 00 00
  30 15 00 00 00 00
  30 10 00 00 00 00
  10 02 00 00 00 00
  10 00 00 00 00 00
  10 0a 00 00 00 00
  10 0e 00 00 00 00
  10 22 00 00 00 00
  10 24 00 00 00 00

  <<
  08 10 00 04
  00 00 00 00 00 0b
  30 0f 00 00 09 09   LDA_VERSION
  30 0e 00 00 00 01   LDA_REVISION
  30 00 00 00 00 00   LDA_ENABLES
  30 15 00 00 00 00   LDA_PKTGEN_TXCOUNT
  30 10 00 00 00 00   LDA_PKTGEN_CONTROL
  10 02 00 00 03 ff   DIF_LINK_RX_EN
  10 00 00 00 03 ff   DIF_LINK_TX_EN
  10 0a 00 00 00 00   DIF_LINK_STATUS1
```

```
  10 0e 00 00 00 00   DIF_LINK_NO_SIGNAL
  10 22 00 00 00 00   DIF_LINK_LOCKED
  10 24 00 00 00 01   DIF_LINK_DCM
  */
int buildPing(uint16_t** rc)
{
  static uint16_t ldapkt[100];
  int offs = 0;
  int i;

  // lda mac
  ldapkt[offs++] = htons(0x5e70);
  ldapkt[offs++] = htons(0x0cd2);
  ldapkt[offs++] = htons(0x77e8);

  // api mac
  ldapkt[offs++] = htons(0x0800);
  ldapkt[offs++] = htons(0x27a8);
  ldapkt[offs++] = htons(0xf0e8);

  ldapkt[offs++] = htons(0x0810); // ether type
  ldapkt[offs++] = htons(0x0002); // subsystem + optype
  ldapkt[offs++] = htons(0x0000); // modifier
  ldapkt[offs++] = htons(0x0000); // packet id
  ldapkt[offs++] = htons(0x000b); // data length

  ldapkt[offs++] = htons(0x300f); ldapkt[offs++] = htons(0x0000); ldapkt[offs++] = htons(0x0000);
  ldapkt[offs++] = htons(0x300e); ldapkt[offs++] = htons(0x0000); ldapkt[offs++] = htons(0x0000);
  ldapkt[offs++] = htons(0x3000); ldapkt[offs++] = htons(0x0000); ldapkt[offs++] = htons(0x0000);
  ldapkt[offs++] = htons(0x3015); ldapkt[offs++] = htons(0x0000); ldapkt[offs++] = htons(0x0000);
  ldapkt[offs++] = htons(0x3010); ldapkt[offs++] = htons(0x0000); ldapkt[offs++] = htons(0x0000);
  ldapkt[offs++] = htons(0x1002); ldapkt[offs++] = htons(0x0000); ldapkt[offs++] = htons(0x0000);
  ldapkt[offs++] = htons(0x1000); ldapkt[offs++] = htons(0x0000); ldapkt[offs++] = htons(0x0000);
  ldapkt[offs++] = htons(0x100a); ldapkt[offs++] = htons(0x0000); ldapkt[offs++] = htons(0x0000);
  ldapkt[offs++] = htons(0x100e); ldapkt[offs++] = htons(0x0000); ldapkt[offs++] = htons(0x0000);
  ldapkt[offs++] = htons(0x1022); ldapkt[offs++] = htons(0x0000); ldapkt[offs++] = htons(0x0000);
  ldapkt[offs++] = htons(0x1024); ldapkt[offs++] = htons(0x0000); ldapkt[offs++] = htons(0x0000);

  //printf("Packet is %i bytes:\n", offs*sizeof(uint16_t));
  for (i=0; i< offs; ++i) {
    if (i%8==0) printf("\n");
    printf ("%02x %02x ", ldapkt[i]>>8, ldapkt[i]&0xff);
  }
  printf("\n");

  *rc = ldapkt;
  return offs;
}

/*
//CALDIF::PktAccessor_LDA_pkt ldapkt(66);
//ldapkt.set_pdu_nbytes(0);  int i;

//uint8_t const ldaAdd[ETHER_ADDR_LEN]={0x5e, 0x70, 0x0c, 0xd2, 0x77, 0xe8};
////uint8_t const apiAdd[ETHER_ADDR_LEN]={0x00, 0x22, 0x19, 0x00, 0x3d, 0x8b};
//uint8_t const apiAdd[ETHER_ADDR_LEN]={0x08, 0x00, 0x27, 0xA8, 0xF0, 0xE8};
//struct ether_addr lda_mac;
```

```
//LLR_CALICE_TESTS::LDA_interface lldapkt[i]da("eth2", lda_mac);
//lda.set_debug();

//lda.flush_recv_queue();
//lda.set_intersend_pause(10);
//lda.send_tcpdump_magic(0x22, 'b');

//ldapkt.set_ether_dhost(ldaAdd);
//ldapkt.set_ether_shost(apiAdd);

//ldapkt.set_ether_type(0x0810);
//ldapkt.set_LDA_subsystem(0x00);
//ldapkt.set_LDA_optype(0x02);
//ldapkt.set_LDA_Modifier(0x0000);
//ldapkt.set_LDA_PktID(0x0000);
//ldapkt.set_LDA_DataLength(0x000b);

//uint16_t * ldacontents = (uint16_t*) ldapkt.get_pdu();
//lda.send_verbatim(ldapkt);
*/
int main()
{
  pcap_t *pcap = NULL;
  int rc = 0;
  char* filter = NULL;
  uint16_t* ldapkt = NULL;
  int offs = 0;

  offs = buildPing(&ldapkt);
  pcap = ethConnect("eth2", filter);

  // transmit a packetpkt
  if ((rc = pcap_inject(pcap, ldapkt, offs*2)) < 0){
    pcap_perror(pcap, "pcap_inject: ");
    goto error;
  }

  //printf("%i bytes injected\n", rc);

  // close a capture device or savefile
  pcap_close(pcap);

  exit(EXIT_SUCCESS);
 error:
  exit(EXIT_FAILURE);
}
```

listen

```
#include <stdio.h>
#include <string.h> // memset
#include <stdlib.h> // exit

#include <pcap/pcap.h>
#include <sys/poll.h>

#ifndef FALSE
#define FALSE 0
#endif
```

```
#ifndef TRUE
#define TRUE 1
#endif


/*
//LLR_CALICE_TESTS::LDA_interface lda("eth2", lda_mac);
//lda.set_debug();
//lda.flush_recv_queue();
*/
pcap_t* ethConnect(char* ethernetCard, char* filter)
{
  static char errbuff[PCAP_ERRBUF_SIZE];
  pcap_t* rc = (pcap_t*)0;
  struct bpf_program          pcap_filter;

  memset(& pcap_filter, 0, sizeof(pcap_filter));

  // create a live capture handle
  if ((rc = pcap_create(ethernetCard, errbuff)) == (pcap_t *)0){
    printf("pcap_create: %s\n", errbuff);
    goto error;
  }

  // set the snapshot length for a not-yet-activated capture handle
  if (pcap_set_snaplen(rc, 16*1024) != 0){
    printf("pcap_set_snaple failed.\n");
    goto error;
  }

  // set the buffer size for a not-yet-activated capture handle
  if (pcap_set_buffer_size(rc, 8UL*1024*1024) != 0){
    printf("pcap_set_buffer_size failed.\n");
    goto error;
  }

  /* We need to set a timeout > 0 so that pcap_setnonblock() below
     takes effect. If we keep the default value (0), then
     pcap_setnonblock() will have no effect, and any read operation
     will be infinitly blocking... which is the OPPOSITE from what
     we want. With a timeout > 0, then pcap_setnonblock() will
     effectively and correctly configure the interface as
     non-blocking. */

  // set the read timeout for a not-yet-activated capture handle
  if (pcap_set_timeout(rc, 10) != 0){
    printf("pcap_set_timeout failed.\n");
    goto error;
  }

  // activate a capture handle
  if (pcap_activate(rc) != 0){
    printf("pcap_activate failed.\n");
    goto error;
  }

  // compile a filter expression
```

```
  if (pcap_compile(rc, &pcap_filter, filter, 1, 0) != 0){
    pcap_perror(rc, "pcap_compile: ");
    goto error;
  }

  // set the filter
  if (pcap_setfilter(rc, &pcap_filter) != 0){
    pcap_perror(rc, "pcap_setfilter: ");
    goto error;
  }

  // set or get the state of non-blocking mode on a capture device
  if (pcap_setnonblock(rc, 1, errbuff) != 0){
    printf("pcap_setnonblock: %s\n", errbuff);
    goto error;
  }

  return rc;
 error:
  // close a capture device or savefile
  pcap_close(rc);

  // free a BPF program
  pcap_freecode(&pcap_filter);

  return (pcap_t*)0;
}

/*
  CALDIF::PktAccessor_LDA_pkt const* ldapkt = NULL;
  try
  {
  ldapkt = & lda.recv(100000000);
  }
  catch (LLR_CALICE_TESTS::LDAError & ex)
  {
  LLR_LOGGER_ERROR("Got error while waiting for pkt: "
  << ex.what());
  break;
  }

  std::stringstream sstrm;
  CALDIF::PktPrinter_Human.print(sstrm << "Received packet ",
  *ldapkt) << std::endl;
  CALDIF::hexdump(sstrm, ldapkt->get_sdu(), ldapkt->get_sdu_nbytes()
  - sizeof(struct ETH_word_CRC));
  LLR_LOGGER_DEBUG(sstrm.str());
*/

int
waitForPacket(int sd, unsigned timeout_ms)
{
  struct pollfd pfd;
  pfd.fd      = sd;
  pfd.events  = POLLIN|POLLRDNORM|POLLERR;
  pfd.revents = 0;
  return (1 == ::poll(&pfd, 1, timeout_ms));
}
```

```
int
ethRecv (pcap_t* pcap,
 struct pcap_pkthdr **pkt_header, const unsigned char **pkt_data,
 int timeout_ms)
{
  static char errbuff[PCAP_ERRBUF_SIZE];
  int rc = 0;
  int sd = -1;

  // get the file descriptor for a live capture
  if ((sd = pcap_fileno(pcap)) < 0){
    printf("pcap_fileno: %s\n", errbuff);
    goto error;
  }

  /* Try first to read from user memory */

  // read the next packet from a pcap_t
  if ((rc = pcap_next_ex(pcap, pkt_header, pkt_data)) < 0){
    printf("pcap_next_ex: %s\n", errbuff);
    goto error;
  }

  if (timeout_ms > 0
      && rc == 0 /* No packet immediatly available... */)
    {
      /* No packet => wait for one to arrive (negative timeout =
 infinite) */
      waitForPacket(sd, timeout_ms); /* Ignore return value to
 prevent famine */

      /* Second chance after poll()... */

      // read the next packet from a pcap_t
      if ((rc = pcap_next_ex(pcap, pkt_header, pkt_data)) < 0){
printf("pcap_next_ex: %s\n", errbuff);
goto error;
      }
    }

  return rc;
 error:
  return -1;
}

int main()
{
  unsigned int i;
  pcap_t                *pcap;
  struct pcap_pkthdr    *pkt_header;
  const unsigned char   *pkt_data;
  char* filter = \
    "( " \
    "    ether proto 0x0809 " \
    " or ether proto 0x0810 " \
    " or ether proto 0x0811" \
    ") " \
```

```
    "and ether src 5e:70:0c:d2:77:e8";

  pcap = ethConnect("eth2", filter);

  while (ethRecv(pcap, &pkt_header, &pkt_data, 10000000) > 0) {

    for (i=0; i< pkt_header->caplen; ++i) {
      if (i%16==0) printf("\n");
      printf ("%02x ", pkt_data[i]);
    }
    printf("\n");
  }

  printf("Timeout waiting for packet\n");

  exit(EXIT_SUCCESS);
}
```

3.2 Schroot

3.2.1 Introduction

```
# aptitude install schroot
```

fichier */etc/schroot/schroot.conf* or */etc/schroot/chroot.d/slc5_64*:

```
[slc5_64]
description=SLC 5 64 bits
type=directory
# location=/mnt/slc5_64
directory=/mnt/slc5_64
users=nroche
aliases=slc5_64
# run-setup-scripts=true
# run-exec-scripts=true
personality=linux
```

fichier */etc/schroot/mount-defaults*

```
# mount.defaults: static file system information for chroots.
# Note that the mount point will be prefixed by the chroot path
# (CHROOT_PATH)
#
# <file system><mount point><type><options><dump><pass>
proc /proc proc defaults 0 0
#procbususb      /proc/bus/usb   usbfs   defaults        0       0
sys /sys sysfs defaults 0 0
/dev            /dev            none    rw,bind         0       0
/dev/pts /dev/pts none rw,bind 0 0
/dev/shm /dev/shm none rw,bind 0 0
# /home /home none rw,bind 0 0
/data1 /data1 none rw,bind 0 0
/tmp /tmp none rw,bind 0 0
```

- `$ schroot -l`
- `$ schroot -vc slc5_64`

Warning, the */etc/group*, */etc/passwd*, */etc/gshadow* and */etc/shadow* will be erased on the chrooted system partition. You need to backup them so as to restore them before normal boot.

3.3 Stow

3.3.1 Introduction

Stow is a tool for managing the installation of multiple software packages (from sources) in the same run-time directory tree. One historical difficulty of this task has been the need to administer, upgrade, install, and remove files in independent packages without confusing them with other files sharing the same filesystem space.

3.3.2 Exemple

```
# apt-get install stow
$ mkdir -p ~/soft/toto
$ echo 'echo "Hello"' > ~/soft/toto/toto

$ cd ~/soft
$ stow -v toto
Stowing package toto...
LINK /data1/ubuntu64/home/nroche/toto to soft/toto/toto

$ stow -D -v toto
UNLINK /data1/ubuntu64/home/nroche/toto
```

3.3.3 Using it on SLC

- Install stow:

```
$ cd /tmp
$ wget ftp://ftp.gnu.org/gnu/stow/stow-1.3.3.tar.gz
$ tar -zxf stow-1.3.3.tar.gz
$ cd stow-1.3.3
$ ./configure --prefix=~/libLDA/usr
$ make
# make install
# mkdir /usr/local/stow
```

- Build external libraries:

```
$ mkdir tgz
$ wget -P tgz http://prdownloads.sourceforge.net/flex/flex-2.5.35.tar.bz2?download
$ wget -P tgz http://ftp.gnu.org/gnu/bison/bison-2.4.tar.gz
$ wget -P tgz http://www.tcpdump.org/release/libpcap-1.1.1.tar.gz
$ tar -jxf tgz/flex-2.5.35.tar.bz2 -C /tmp
$ tar -zxf tgz/bison-2.4.tar.gz -C /tmp
$ tar -zxf tgz/libpcap-1.1.1.tar.gz -C /tmp

$ cd /tmp/flex-2.5.35
$ ./configure --prefix=/usr/local/stow/flex-2.5.35
$ make
# make install

$ cd /tmp/bison-2.4
$ ./configure --prefix=/usr/local/stow/bison-2.4
$ make
# make install

$ cd /tmp/libpcap-1.1.1
$ PATH=/usr/local/stow/flex-2.5.35/bin:/usr/local/stow/bison-2.4/bin:$PATH \
```

```
  ./configure --prefix=/usr/local/stow/libpcap-1.1.1
$ PATH=/usr/local/stow/flex-2.5.35/bin:/usr/local/stow/bison-2.4/bin:$PATH \
  make
# make install

# cd /usr/local
# tar -zcf externalLibs.tgz stow/
# mv externalLibs.tgz ~nroche/libLDA/usr/
# chown nroche. ~nroche/libLDA/usr/externalLibs.tgz
```

- Install external libraries:

```
$ 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
```

- avr-libc:

```
$ cd /tmp
$ wget http://download.savannah.gnu.org/releases/avr-libc/avr-libc-1.0.5.tar.bz2
$ tar -jxf avr-libc-1.0.5.tar.bz2
$ cd avr-libc-1.0.5
(arf, do not compile with gcc4)
```

- libcap:

```
$ cd /tmp
$ wget http://www.kernel.org/pub/linux/libs/security/linux-privs/libcap2/libcap-2.20.tar.bz2
$ tar -jxf libcap-2.20.tar.bz2
$ cd libcap-2.20
$ ./configure --prefix=/usr/local/stow/libcap-2.20
# yum install libattr-devel
$ make
$ env DESTDIR=/usr/local/stow/libcap-2.20 make install
$ cd /usr/local/stow/
# stow -v libcap-2.20
# cat >> /etc/ld.so.conf
/usr/local/lib
^D
# ldconfig
# ldconfig -p | grep libcap
# cd /tmp/libcap-2.20
# env DESTDIR=/usr/local/stow/libcap-2.20 make install
# cd /usr/local/stow/
# stow -v libcap-2.20
(arf Since 2.6.24-rc2 the file system support for POSIX Capabilities is part of mainline.
So there is no kernel patching needed.
Because the PCaps for files are stored in the extended attributes of that file,
a further prerequisite is a file system, that supports extended attributes like ext3.)
$ uname -a
2.6.18
```

3.4 Python

3.4.1 Introduction

see:

- https://llrforge.in2p3.fr/trac/pyLLR/
- http://docs.python.org/tutorial/
- http://docs.python.org/library/index.html
- “cheeseshop”
- “python recipes”

3.4.2 QT4 designer

QT4 designer is used to create the */pylib/qt4_gui.ui* XML file. This file is process by `pyuic4` to *qt4_main.py* that only display the gui (do not modify it).

```
# apt-get install qt4-designer pyqt4-dev-tools
$ designer
$ pyuic4 qt4.gui.ui ...
```

Warning: check for `python-sip python-qt4 python-dpkt python-pcapy` *main.py* call gui by passing an event call backs object.

```
setupUI(self)
```

3.4.3 Entry points

- In a file, `__init__` play `main()` function role.
- In a directory, *__init__* file is needed and provide a packet unitary test entry.

3.5 Virtual Box

3.5.1 Introduction

Virtual-box allows to share the ethernet card amongs several operating system. This is usefull as the python code use qt4 that is not available on Scientific Linux 5.5.

3.5.2 Install

- Ubuntu:

```
$ cat /proc/cpu | grep svm (tell if cpu allow virtalization)
# aptitude install virtualbox-ose (Open Source Edition)
# aptitude install virtualbox-ose-dkms (for virtualization: Debian Kernel Management System)
# aptitude install virtualbox-guest-additions (for mouse caption between Linux and Windows)
$ VirtualBox
```

- SLC55: From http://www.virtualbox.org/wiki/Linux_Downloads choose “Red Hat Enterprise Linux 5 ("RHEL5") / Oracle Linux 5 ("OL5") / CentOS 5 i386”.

```
$ wget http://download.virtualbox.org/virtualbox/4.0.10/VirtualBox-4.0-4.0.10_72479_rhel5-1.i386.rp
# rpm ...
```

3.5.3 Configuration

- use 892MB of RAM and 20GB for HD.
- Add serial port COM1 (unconnected).
- Iso are store here:

```
$ find /data2/VM
/data2/VM/xpkeys.txt
/data2/VM/shared
/data2/VM/iso/llrxpsp2.iso
/data2/VM/iso/office2003.iso
/data2/VM/snapshots
/data2/VM/vdisks/winxp.vdi
```

- *VBoxGuestAdditions.iso* should be automaticly provide into CD images.
- From VirtualBox, select the iso and run it into Windows.
- update Windows to SP3
- defined a shared folder using the GUI and copy into your SVN private key
- you should retrieve it from Windows into the “network favorite folders”

Part III

Annexes

```
$ groff -man -Tascii libLDA.3 | col -b > manual.txt


LIBLDA(3)  libLDA Manual      LIBLDA(3)



NAME
       LibLDA - slow control and acquisition layer for Calice's LDA

SYNOPSIS
       #include "libLDA.hh"

       src/run/dump [-h] [OPTIONS] CONFIG
       src/run/ping [-h] [OPTIONS] CONFIG
       src/run/fifo [-h] [OPTIONS] CONFIG
       src/run/rng [-h] [OPTIONS] CONFIG
       src/run/config [-h] [OPTIONS] CONFIG
       src/run/driver [-h] [OPTIONS] CONFIG
       exemple [-h] [OPTIONS] CONFIG
       tools/utDevice [-h] [OPTIONS] CONFIG
       tools/utDaq [-h] [OPTIONS]


DESCRIPTION
       libLDA  is  a  library that drive the LDA Calice's devices. The library
       provides non-regression tests as dump, ping, fifo,  config  and driver
       and  exemples  executables  as exemple but also utDevice and utDaq that
       extend the purpose of this library in order  to drive  a  full Calice
       bench test.


OPTIONS
       The following options are supported:

       -c , --conf-source
      Specify  the  way to load configuration (default is file), among
      file, static and db.

       -p , --eth-port
      Override the ethernet port on the host specified in  configura-
      tion (eth0, ...)

       -a , --lda-address
      Override the  ethernet mac address specified in the config file
      (in aa:bb:cc:dd:ee:ff format)

       -v , --debug-print
      Set logging level (default: INFO) among: DEBUG,  INFO,  WARNING,
      ERROR, NONE

       -f , --syslog
      Use syslog instead of the standard output for log messages

       -n , --no-reinit
      Do not reinitialize LDA, DCC and DIF hardware.

       -s , --single
```

```
      Do not enter into infinite loop but do the job only once.

       -m , --mode
      Set the mode (default: 8) among: 8 (ilc-manual), 0 (ilc-auto) or
      4 (beam-test) Note: today only the beam-test is functional

       -o , --time-out
      Set the read-out trigger timeout (unit is microsecond)

       -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)
      Note: beware that data buffer size is at less equal to the  size
      of maximum event size.

       -S , --freq-slc
      Set slow control frequency (unit is Hertz)
      Note: this thread is not included into the libLDA.

       -T , --freq-slc
      Set trigger frequency (unit is Hertz)
      Note: this thread, not included in the libLDA, only run in the 8
      (ilc-manual) mode.

       -U , --freq-usr
      Set user read out frequency (unit is Hertz)
      Note: this thread is not included into the libLDA.

       -h, --help
      Print a short help text describing  the  supported  command-line
      options, and then exit.

FILES
       data/dif-X-Y.txt
      use by exemple to write output data

       data/EE:FF:X.Y.txt
      used by utDevice and utDaqP to write the output data


REPORTING BUGS
       Please send a report to nicolas.roche@llr.in2p3.fr

AUTHORS
       Nicolas Roche, David Decotigny.

LICENCE
       This  software is released under the GNU General Public License. Please
       read the COPYING file for more information.

SEE ALSO
       The full  documentation for  libLDA  is  maintained  as  a  Latex2Html
       intranet.    If  a   browser  is  installed  at  your  site,  the  URL
```

```
       <http://polype/~roche/calicei/> should give you access to the  complete
       manual.
       libLDA(5), calDump(1), pcat(1)


Version 1.0 August 16, 2011       LIBLDA(3)
```

```
$ groff -man -Tascii libLDA.5 | col -b > manual.txt


LIBLDA(5)  libLDA Manual      LIBLDA(5)



NAME
       LibLDA - configuration file format for Calice's libLDA


DESCRIPTION
       libLDA  is  a  library that drive the LDA Calice's devices. The library
       provides non-regression tests  and  exemples  executables.  These  exe-
       cutable all  accept  a configuration file from command line parameter.
       Theses files has the following structure, using the python array  gram-
       mar in order to gives hardware parameters and routes.


GRAMMAR
       stanza: parameter "=" (value|array)

       value: [:alnum:]* | array

       array: "{" port ":" value "}"

       port:  [:digit:]*

       Empty  lines  are ignored. If a line contains a hash mark (#), the hash
       mark and the remaining part of the line are ignored.


SHORT EXEMPLE
       # This is a comment
       LDA(ethernet_port = "eth6",
 ethernet_address = MACAddress("5e:70:0c:d2:5e:8e"),
 ports = {
   1: DIF(
     bypass = False,
     nbAsics = 24,
     drv_inpath_asu = "conf/sc1.asu"
   ),
   2: DCC(ports = {
     8: DIF(bypass = False,
nbAsics = 24,
drv_inpath_asu = "conf/sc2.asu"
     )
   })
 })
       data_buffer_size = 70000
       freq_slc = 2.

FIELD DESCRIPTION
   LDA
       ethernet_port
      Ethernet port name on the host. See '/sbin/ifconfig'  and  'sudo
      ethtool'.   Can  be  overriden  with the '-i IFNAME' flag of the
      program. Exemple value: eth1
```

```
     ethernet_address
    MAC address of the LDA we intent to configure.   Exemple value:
    MACAddress("5e:70:0c:d2:5e:8e")

DCC
     no parameter for DCC

DIF
     bypass Instead  of  writing  'DIF()',  write 'DIF(bypass=True)' to tell
    that the DIF should be detected but will not  be used  for  the
    tests.

     ftdi_id
    Offer a way to store the ftdi Id (not related to hardware).

     nbAsics
    Tell  the  hardware  how many ASICs are manage by the DIF.  Used
    values: 4 or 8 depending of how many ASU are  connected  to  the
    DIF.

     drv_inpath_asu
    Path of the ASU configuration file.

     rng_packets_num
    Number  of  packets to ask the DIF during random generator test.
    Note: 511 for inifite.

     rng_packets_words
    Number of 16b words in each DIF random generator packet (< 500)

     rng_packets_gap
    Gap between each DIF random generator packet (number of words)

GLOBAL
     reinit Do reinitialize LDA, DCC and DIF hardware.

     infloop
    Enter into infinite loop, so do the job more than only once.

     mode   Set the mode (default: 8) among: 8 (ilc-manual), 0 (ilc-auto) or
    4 (beam-test) Note: today only the beam-test is functional

     time_out
    Read-out trigger timeout (unit is microsecond)

     pcap_buffer_size
    PCAP input buffer size (unit is 1Ko packet).

     trig_buffer_size
    Read-out trigger buffer size (unit is 1 trigger)

     data_buffer_size
    Read-out data buffer size (unit is 2 bytes)

     freq_trg
    Slow control frequency (unit is Hertz)
    Note: this thread is not included into the libLDA.
```

```
       freq_usr
      Trigger frequency (unit is Hertz)
      Note: this thread, not included in the libLDA, only run in the 8
      (ilc-manual) mode.

       freq_slc
      User read out frequency (unit is Hertz)
      Note: this thread is not included into the libLDA.

REPORTING BUGS
       Please send a report to nicolas.roche@llr.in2p3.fr

AUTHORS
       Nicolas Roche, David Decotigny.

LICENCE
       This software is released under the GNU General Public License. Please
       read the COPYING file for more information.

SEE ALSO
       The full documentation for libLDA is maintained as a Latex2Html
       intranet.  If a browser is installed at your site, the URL
       <http://polype/~roche/calicei/>should give you access to the complete
       manual.
       libLDA(3)



Version 1.0 August 16, 2011       LIBLDA(5)
```

```
$ groff -man -Tascii calDump.1 | col -b > manual.txt


calDump(1) calDump Manual     calDump(1)



NAME
       calDump - Calice DIF's output data analyser

SYNOPSIS
       cat data/dif-4-9.txt | src/run/calDump

DESCRIPTION
       calDump is  a simple data converter from binary to hexadecimal.  Using
       it ensure the output data format is correct as will notify  all errors
       it found using syslog.

OPTIONS
       No option for now.

EXEMPLE
       This  allow  to rebuild  the file without re-run calDump: $ tail -fn 0
       data/dif-4-9.txt | src/run/calDump

REPORTING BUGS
       Please send a report to nicolas.roche@llr.in2p3.fr

AUTHORS
       Nicolas Roche, David Decotigny.

LICENCE
       This software is released under the GNU General Public License. Please
       read the COPYING file for more information.

SEE ALSO
       The  full  documentation  for  libLDA  is  maintained  as  a Latex2Html
       intranet.   If  a  browser  is  installed  at  your   site,   the   URL
       <http://polype/~roche/calicei/>should give you access to the complete
       manual.
       libLDA(5), pcat(1)



Version 1.0 August 16, 2011      calDump(1)
```

```
$ groff -man -Tascii pcat.1 | col -b > manual.txt

PCAT(1)    pcat Manual        PCAT(1)



NAME
       pcat - raw socket necat like tool

SYNOPSIS
       tools/pcat [-h] [OPTIONS]

DESCRIPTION
       pcat  is  a  tool  using  the libpcap basic features.  For instance, it
       allow to reinject raw socket data captured using tcpdump.

OPTIONS
       The following options are supported:

       -m     Mode amongs ping, snif, record, replay (default).
      ping send a LDA Version packet (or bit stream with -i)
      snif hexadecimal dump (as tcpdump)
      record binary dump (as tcpdump -w)
      replay re-inject a previous binary dump done with record

       -I     Interface to use like eth0

       -M     LDA mac address to use like xx:xx:xx:xx:xx:xx

       -Z     Host mac address to use (using replay mode)

       -t     Sleeping time between 2 paquets (using replay mode)

       -i     Input file

       -o     Output file

       -f     Used facility for logging

       -s     Used severity for logging

       -l     File to log into

       -h     Print a short help text describing  the  supported  command-line
      options, and then exit.

EXEMPLE
   CAPTURE DATA
       # ./config polcaldaq.conf -m4 -s
       # tcpdump -xx -s 1024 -i eth0 ether host 5e:70:0c:d2:54:fe -w dump.pcap
       (now activate triggers via CCC)

   RE-INJECT DATA
       #  ./exemple  polcaldaq.conf -i1000 -t100 -d500000 -m4 -S2 -U10 -p eth0
       -n
       # tools/pcat -I eth0  -M  5e:70:0c:d2:54:fe  -Z  aa:aa:aa:aa:aa:aa  -m
       replay -i dump.raw -t 100
```

BUGS

Please note that "lo" interface didn't accept to replay packets and unlinked ethernet card too. So you should use an existing ethernet interface linked to wathever other host in order to reinject packets using the pcat tool.

REPORTING BUGS

Please send a report to nicolas.roche@llr.in2p3.fr

AUTHORS

Nicolas Roche, David Decotigny.

LICENCE

This software is released under the GNU General Public License. Please read the COPYING file for more information.

SEE ALSO

The full documentation for libLDA is maintained as a Latex2Html intranet. If a browser is installed at your site, the URL <http://polype/~roche/calicei/>should give you access to the complete manual.
libLDA(5), calDump(1)

Version 1.0 August 16, 2011 PCAT(1)

- Milestones

DIF firmware upgrade to version 2	working on
LDA library	working on
Non regression tests	working on

- Todo list

provide to Guillaume Baulieu the Spiroc fields for DB
write the Online server specification
sending Slow control to ASU (to merge)
receiving data on trigger from ASU (to test)
check CRC computation/global trigger counter (bypassed)
install and test the XDAQ user interface (to document)
provide registers to libLDA for assynchronous slow control (v2)

- Done

12-2010	Transition's documentation
1-2011	Test servers dupplication
2-2011	SLC55 port (no libcap)
3-2011	Use callback with libpcap to bypass packets copy
4-2001	Use threads and share data between them
5-2001	Send configuration and read out data
6-2001	Port library software on ECAL

- Troubles

Working on ASUv2 since Lyon in fact use ASUv3
`send_FC_DIF_reset` doesn't works on DHCAL
`send_FC_DIF_read_debug_fifo` doesn't works on DIFv2
LDA suspected to not propagate all read orders
libcap need a kernel patch on SLC55 (must use setuid root)

.0.4 Meeting 2011-10-24

État des lieux (Vincent)

- Qu'est-ce qu'on fait avec la version actuelle, comment réduire les divergences ?
- Qu'est-ce qu'on met dans la nouvelle version ?
- IPNL: demander à lyon de garder à jour la branche (bypass thradSlc, callback, buffer)
- LLR ECAL: faire une branche (fonctions read/write bas niveau)
- LLR DHCAL: faire une version prod (via TAG) et une version dev (masques, relecture, revue de code config)

Parallélisation de l'envoie des config reste à faire.
3 Bancs de tests:

- LLR SDHCAL DIF0 + DIF ECAL: banc de test dédié à l'acquisition.
- LLR ECAL FW dev3 (2 détecteurs interchangeables).
- CERN SDHCAL (déménagé à Lyon fin Novembre, test Avril) => revenir à l'USB + distribution de l'horloge via CCC/LDA/DCC/DIF.
- Prévu avec la GigaDCC.

Outils de contrôle de Muriel:

- Bouton lancer un RUN ? (réponse: Configuration + start CCC)

TODO:

- Prévoir des créneaux et un reset distant pour le banc CERN SDHCAL.
- Demander à Guillaume de passer à la Dev3 au plus vite.
- Discuter sur la politique d'affectation des ressources pour les Threads.
- Demande sur le banc de test SDHCAL: Laisser en état 15 DIF, 5 DCC, 2 LDA.
- Online: Dev LLR + Test LLR via standalone avant intégration XDAQ.

LibLDA v2.5

6 mois de dévellopement

- 2 canaux: SLC + DAQ (l'aiguillage en sortie de LDA n'existe pas).
- Proposition Service Online: Passer d'une librairie à un serveur.
- Proposition Elec: ODR (2 ans)
- Base de donnée (sortie de la libLDA et envoyé par SLC via XML)
- Documenter la nouvelle version.
- Trouver un nom.

Bibliography

[1] Format of the read-out data of the dif. Version 1.1.1 r227
https://forge.in2p3.fr/attachments/95/DIF_readout_format.pdf.

[2] Dif - operating manual. Version 1.16
http://adweb.desy.de/~reinecke/DIF_Firmware_vers1_16.pdf, 2010.

[3] Mark Kelly (Manchester) and Matt Warren (UCL). Lda user doc, 2010.
https://twiki.cern.ch/twiki/bin/view/CALICE/LinkDataAggregator.

[4] Barry Green. Odr user doc, 2009.
https://twiki.cern.ch/twiki/bin/view/CALICE/OffDetectorReceiverProgrammersGuide.

[5] David Dacotigny. Svn calice online. https://svn.in2p3.fr/calice/.

[6] David Dacotigny. Forge calice online. https://forge.in2p3.fr/account/login_without_cas.

[7] Lcio web site. http://lcio.desy.de/.

[8] Servercom firmware.
http://www.acksys.fr/docs_us/Documentationstechniques/SERVERCOMUserGuide(DTUS043).pdf.

[9] Roc chips readout. LAL, IN2P3-CNRS, Université Paris-Sud 11, Orsay, France, 2010.

[10] Dif firmware specs v2. LLR, IN2P3-CNRS, 2011.

[11] Configuration database.
https://lyosvn.in2p3.fr/ilc/wiki/ILCConfDB, 2011.