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

HARPO PROJECT

Online software documentation

Nicolas Roche, Simon Chollet,

October 29, 2015

HARPO PROJECT

Online software documentation

Nicolas Roche, Simon Chollet,

Laboratoire Leprince-Ringuet

October 29, 2015

Abstract

Online software documentation for Harpo.

Keys words: Harpo, software, documentation

Tutors: Denis Bernar, Yannick Geerebaert, Patrick Poilleux

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

Contents

Introduction

- **Project Description**
  γ-ray polarimetry and high-angular-precision astronomy, above pair creation threshold

  You will find here HARPO's DAQ software documentation:

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

  Thanks reading me.

- **LLR team**

name	job	know-how	contact ((+33) 1.69.33.)
Denis Bernard*	Physics	Analyse	55.34 denis.bernard@llr.in2p3.fr
Yannick Geerebaert*	Electronic	DAQ	56.14 geerebaert@llr.in2p3.fr
Patrick Poilleux	Mecanic	Detector	56.62 poilleux@llr.in2p3.fr
Simon Chollet	Computer Science	DAQ	56.34 s.chollet@llr.in2p3.fr
Nicolas Roche	Computer science	libLDA	56.35 nicolas.roche@llr.in2p3.fr

* : officer in charge.

- **Contacts**

name	entity	know-how	contact
Denis Calvet	CEA	DAQ	01 69 08 69 09 denis.calvet@cea.fr

Part I

Harpo DAQ software

This section provides the HARPO's DAQ software specifications.
Nicolas's TODO List:

- Forbidden

Chapter 1

Voice of the customer

1.1 Introduction

This section provides the HARPO's DAQ customers needs.

1.2 udpClient

1.2.1 Buffers

1.2.2 threads

1.3 Log using syslog daemon

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

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

1.4 Non regression tests

1.5 Packets formats

1.5.1 Input

1.5.2 Output

Chapter 2

Functional analysis

This section provides explanation on the DAQ architecture.

Chapter 3

Technical specifications

3.1 Introduction

Chapter 4

Implementation

4.1 Introduction

4.2 Installing, compiling and testing

4.3 API

4.3.1 Configuration

4.4 Syslog

4.4.1 Redirection & modifying severity

4.4.2 Rotation

Chapter 5

Verification

Software status...

Chapter 6

Validation

6.1 Introduction

This section intent to validate the software production.

6.2 Bugs to solve

Part II

Miscellaneous

Chapter 1

Approach

1.1 Introduction

1.1.1 Introduction

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

1.1.2 Generic DAQ

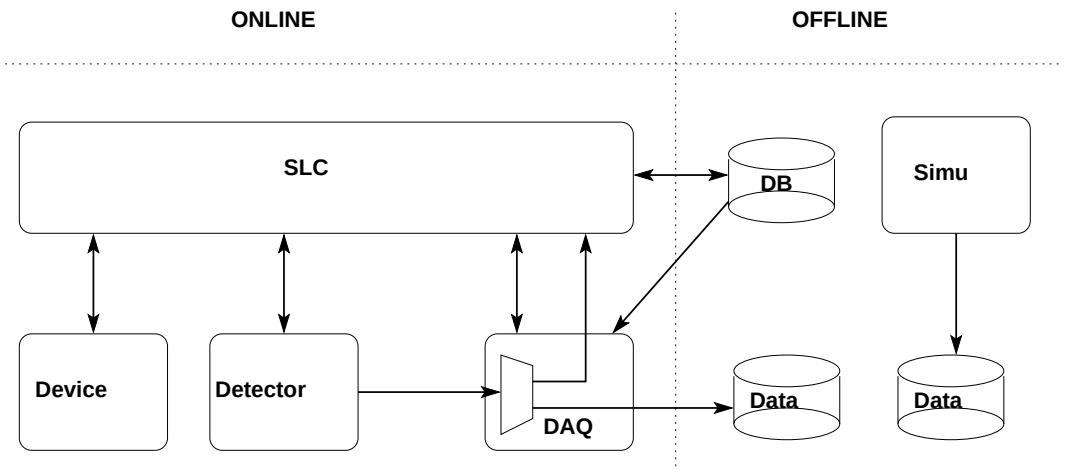

Figure 1.1: Generic architecture

Online architecture is sometime done by a single framework (HESS, XDAQ) and sometime split into 3 compounds:

- Owner graphical interface
- Slow control: «SLC» (TANGO, ENX) wich is mainly a state machine manager.
- Data aquisition: «DAQ» (NARVAL) wich is mainly a multiplexer also called event builder.

Detector usually own 2 kind of links:

- A synchronous link to the slow control
- An asynchronous fast link to data aquisition

1.1.3 Harpo prototype

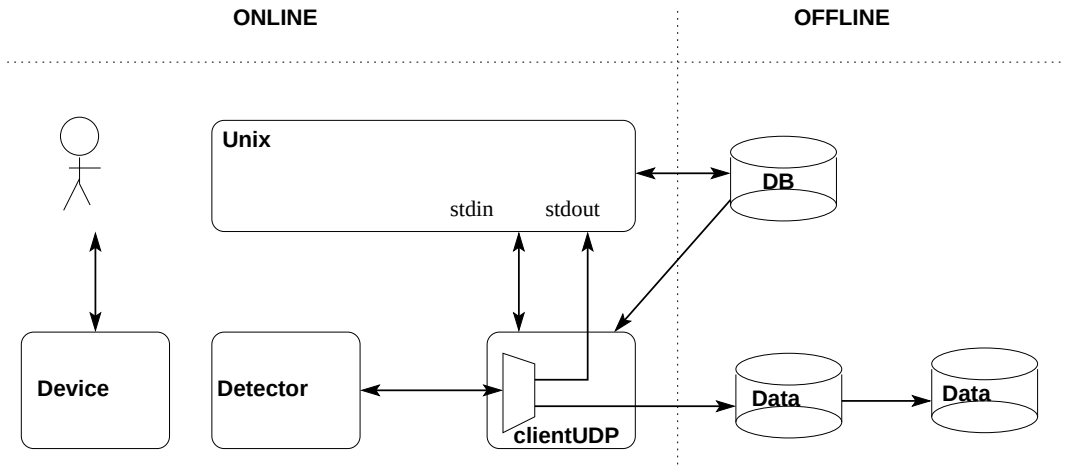

Figure 1.2: Harpo architecture

As the Calice's experiment, we drive the detector using the DAQ.

- We only have a synchronous link to the detector.
- Most of the embedded server functionalities have been exported to the client (DAQ).

Chapter 2

Procedure

2.0.4 Introduction

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

2.0.5 Data format

There is one output file by instance of clientUPD (by card).

```
outputFile  = fileHeader event{nb-events}
event       = eventHeader areqPaquet{nb-asics}
eventHeader = eventSize eventNumber                                [        8 bytes]
areqPacket  = dccPacket{nb-channels}
dccPacket   = payloadSize dccHeader fem data{nb-capas} trailer     [     1044 bytes]
fem         = femHeader args timeH timeL event wordCount           [       16 bytes]
trailer     = align{?} trailerH trailerL                           [        4 bytes]

fileHeader  = RYYYY.MM.DD-HH:MM                                    [       20 bytes]

eventSize   = 4 bytes (see computation above)
eventNumber = 4 bytes

payloadSize = 2 bytes (usually 0x0414 = 1044)
dccHeader   = 2 bytes (usually 0x4000, seen at 0x5000)

femHeader   = 2 bytes (from 0x0000 to 0x000f)
args        = 2 bytes (from 0x0012 to 0x01D4 by step of 6: 2,8,e,4,a,0,6,c)
timeH       = 2 bytes
timeL       = 2 bytes
event       = 2 bytes
wordCount   = 2 bytes (usually 0x0200 = 512: same as nb-capas I guess)

data        = 2 bytes

align       = 2 bytes (0x0000)
trailerH    = 2 bytes (0x0000)
trailerL    = 2 bytes (0x0000)

exemple:
00000000: 5232 3031 312e 3130 2e31 332d 3133 3a33  R2011.10.13-13:3
```

```
00000010: 363a 3437-0004 d7c8 0000 0001-0414 4000  6:47.........@.
00000020: 000d 0012 0003 f092 c95d 0200-0159 0147  .........]...Y.G
...
00000420: 014a 0148 014a 014b 0150 010a-0000 0000- .J.H.J.K.P......
00000430: 0414 4000 000e 0018 0003 f092 c95d 0200  ..@..........]..
00000440: 0139 011d 011a 010f 0116 0117 011d 011a  .9..............
```

- We can compute nb-capas looking at the upper exemple:
  0x42c - 0x2c = 0x400 => 512 words of data
- dccPacket size should be:
  sizeOf(dccPacket) = 16 + (nb-capas * 2) + (4 or 5)
  note: maybe sometime one «align» 0x0000 word is added. In fact even if we have 3 consecutives 0x0000 words, the dccPacket size do not looks to change so I guess the last capa value is in fact 0x0000 (but strange).
- areqPacket size should be:
  sizeOf(areqPacket) = sizeOf(dccPacket) * nb-channels
- event size should be:
  sizeOf(event) = 8 + (sizeOf(areqPacket) * nb-asics)
- Global file size should be:
  sizeOf(outputFile) = 20 + (sizeOf(event) * nb-events)

2.0.6 bgLink bug

These 2 files show the following error, resulting in a dccPacket lost:

```
-------------------------------------
dcc(00): Warning: cmd(27) areq 0 0 0 3 78
 failed: -4 Fem(0): Fem_ReceiveData failed

dcc(00).rep(27): -4 Fem(0): Fem_ReceiveData failed
dcc(00).rep(27.4): 1044 bytes of data (expecting 71 more packets)
-------------------------------------
```

- file YB_10C.DAT

  ```
  0004d7d0:                                [0004 d7c8  .......!........
  0004d7e0: 0000 0002 0414 4000 0006 0012 005c 0f61  ......@......\.a
  ...
  0004e000: 0118 011f 0111 0109 0000 0000 0414 4000  ..............@.
  0004e010: 0008(001e)005c 0f61 c95e 0200 0134 011f  .....\.a.^...4..
  ...
  0004e420: 0414 4000 0009(0024)005c 0f61 c95e 0200  ..@....$.\.a.^..
  ...
  0009afa0: 0000 0000]0004 d7c8 0000 0003 0414 4000  ..............@.
  0009afb0: 0001 0012 00b3 c8de c95f 0200 015b 0146  .........._...[.F
  ```

  – Size of loop on args from 0x0012 to 0x0012 on event 2 is:
  0x0009afa4 - 0x0004d7dc = 0x4D7C8 => 317384

  This seems correct (nb-capas=512, nb-channels=76 and nb-asics=4) but only if we do not have align word in the files (in fact we should have last values at 0x0000 but this is strange):

```
1044   <= sizeOf(dccPacket)   <= 1045
79344  <= sizeOf(areqPacket) <= 79420
317384 <= sizeOf(event)      <= 317688
```

- file YB_10B.DAT

```
0004d7d0:                                [0004 d3b4  .......#........
0004d7e0: 0000 0002 0414 4000 0006 0012 0062 d225  ......@......b.%
...
0004e000: 011f 0117 0123 010c 0000 0000 0414 4000  .....#........@.
0004e010: 0008(001e)0062 d225 c954 0200 0137 011e  .....b.%.T...7..
...
0004e420: 0414 4000 000a(002a)0062 d225 c954 0200  ..@....*.b.%.T..
...
0009ab90:]0004 d7c8 0000 0003 0414 4000 0000 0012  ..........@.....
```

  - At 0x0004e420: We jump from args 0x001e to 0x002a while we expected to jump to 0x0024.
  - Size of loop on args from 0x0012 to 0x0012 on event 2 is: 0x0009ab90 - 0x0004d7dc = 0x4D3B4 => 316340 (we miss exacly 1044 with above)

2.0.7 Client code

Code is here: *harpo@llrharpo: /Partage/harpo/trunk/design/informatique/ClientLLR*
The makefile is here: *ClientLLR/linux/makefile* :

- *ClientLLR/client/clientUDP.c*
- *ClientLLR/gblink/gblink.c* : maccro for data packet
- *ClientLLR/util/histo.c* : histogram object
- *ClientLLR/dcc/pedestal_packet.c* : pedestal packet
- *ClientLLR/dcc/ethpacket.c* : DCC -> pc
- *ClientLLR/dcc/endofevent_packet.c* : end of event packet

File included:

```
#include <stdio.h>
#include <time.h>
#include <netinet/in.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/ioctl.h>
#include <sys/uio.h>
#include <unistd.h>
#include <stropts.h> // stream interface
#include <string.h>
#include <errno.h>
#include <sched.h>
#include <math.h>

#include "platforms/linux/platform_spec.h" (nothing special)

#include "util/linux/sock_util.h"
#include "util/timerlib.h"
#include "util/histo.h"
```

```
#include "gblink/gblink.h"
#include "gblink/linux/gblink_ps.h"

#include "fem/after.h"
#include "fem/fem.h"
#include "fem/err_codes.h"
#include "fem/fem_reg.h"
#include "fem/fem_mapping.h"
#include "fem/fem_t2k_map.h"

#include "dcc/dcc.h"
#include "dcc/ethpacket.h"
#include "dcc/endofevent_packet.h"
#include "dcc/busymeter.h"
#include "dcc/pedestal_packet.h"
#include "dcc/bufpool_err.h"

#include "client/midasformat.h"
```

Chapter 3

Tools

file */etc/network/interfaces*:

```
auto lo
iface lo inet loopback

auto eth1
iface eth1 inet dhcp

auto eth0
iface eth0 inet static
address 192.168.10.1
netmask 255.255.255.255
up route add -host 192.168.10.12 dev eth0
down route del -host 192.168.10.12 dev eth0

auto eth2
iface eth2 inet static
address 192.168.10.2
netmask 255.255.255.255
up route add -host 192.168.10.13 dev eth2
down route del -host 192.168.10.13 dev eth2
```

Usuals commands:

```
# /etc/init.d/networing restart
# route -n
# tcpdump -i eth0 -xx -s1024 ether host 0:1:2:3:4:5
# tcpdump -i eth2 -xx -s1024 ether host 0:1:2:3:4:6
```

Part III

Annexes

- Milestones

...	working on

- Todo list

...

- Done

09-2011	...

- Troubles

...

Bibliography

[1] P. Baron. Asic after. Ref. EDMS: 008673, september 2006.

[2] Denis Calvet. T2k tpc read-out electronics. september 2008.

[3] Denis Calvet. T2k tpc read-out electronics. december 2009.