GUI

Contents

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)

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.

2.1 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.

2.2 Log tab

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

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.

3.1 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`

3.2 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`

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.

4.1 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.

4.2 Selection Tab

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

4.3 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.

4.4 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.

4.5 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.

4.6 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*.

4.7 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.

5 Physical functionalities module

5.1 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)
```

5.2 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*.

5.3 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.

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, ...)

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.