Customer

Contents

1 Introduction

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

2 Pcap Library

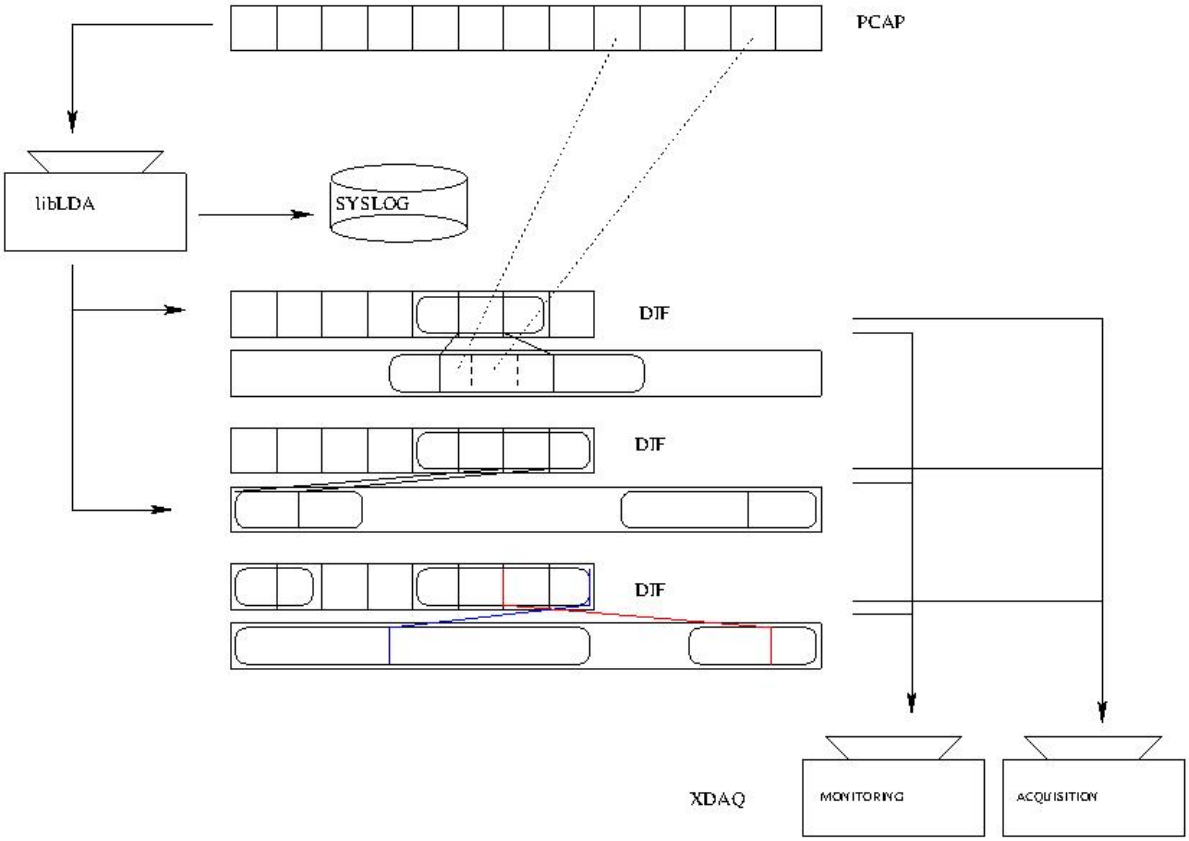
2.1 Buffers

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.

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:



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).

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).

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 6000 by
...
```

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

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).

6 Packets formats

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

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.

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)
```