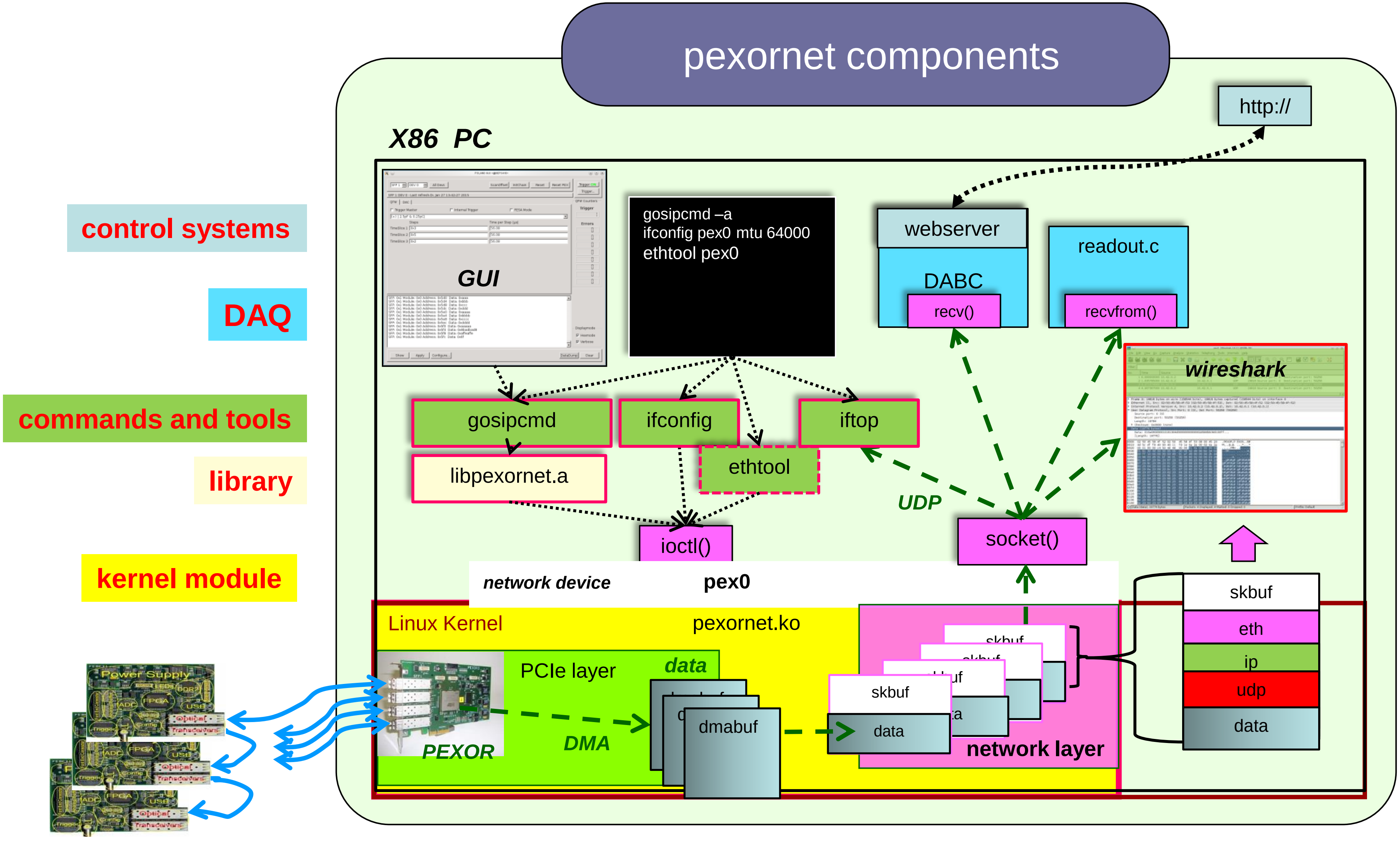
mbspex and pexornet - linux device drivers for PCIe optical receiver data acquisition and control

Jörn Adamczewski-Musch, Nikolaus Kurz, Sergei Linev, GSI, Darmstadt, Germany



control systems

DAQ

commands and tools

library

kernel module

pexornet components

```
gosipcmd –a
ifconfig pex0 mtu 64000
ethtool pex0
```

Poster Session 2
Poster 56

GSI

Jörn Adamczewski-Musch

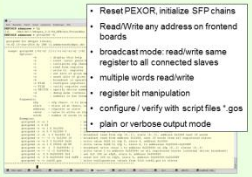
mbspex and pexornet - linux device drivers for PCIe optical receiver data acquisition and control

Jörn Adamczewski-Musch, Nikolaus Kurz, Sergei Linev, GSI, Darmstadt, Germany

Abstract

The GSI PEXOR family PCIe boards are used as interface for data acquisition from various detector front-ends, linked by up to 4 chains of optical fiber connections. Communication with the front-end hardware is handled by the proprietary *gosip* protocol. A trigger module TRIXOR extends the PEXOR by additional signal connections for triggered data acquisition.
For several years the PEXOR boards have been applied with the data acquisition framework MBS. On Linux x86 platform, the device driver software *mbspex* implements concurrent access to the PEXOR front-ends from MBS DAQ, and from separate control applications, like the command line tool *gosipcmd* or hardware specific configuration GUIs.
Besides the established character driver *mbspex*, a network driver *pexornet* has been developed to evaluate a lightweight DAQ system with readout from PEXOR via UDP socket. Therefore common network tools can be applied for driver configuration and data debugging. Moreover, the *gosipcmd* tool and its adjusted API library are fully applicable also for *pexornet*. A simple example DAQ application with *pexornet* UDP readout has been implemented with the software framework DABC, delivering the same data file format and online monitoring capabilities as MBS.
Readout performance of a test set-up has been measured both with MBS / mbspex, and with DABC / pexornet.

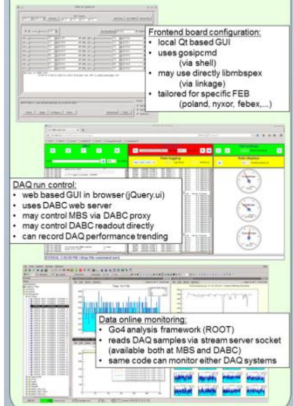
gosipcmd



- Reset PEXOR, initialize SFP chains
- Read/Write any address on frontend boards
- broadcast mode: read/write same register to all connected slaves
- multiple words read/write
- register bit manipulation
- configure / verify with script files *.gos
- plain or verbose output mode

commandline control interface *gosipcmd*:
available for both drivers with (almost) same syntax
(*pexornet* adds "start/stop acquistion" commands to change interrupt readout state)

control GUIs

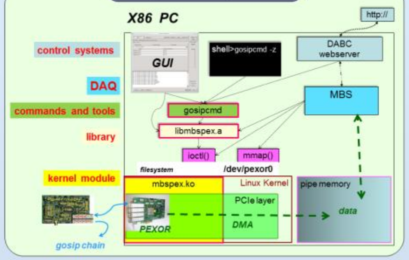


Frontend board configuration:
- local Qt based GUI
- uses gosipcmd
  (via shell)
- may use directly libmbspex
  (via linkage)
- tailored for specific FEB
  (poland, nyxor, febex,...)

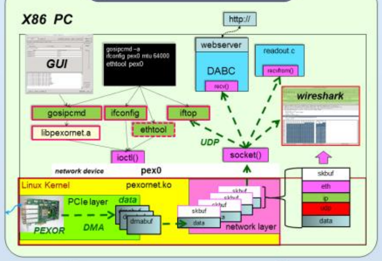
DAQ run control:
- web based GUI in browser (jQuery.ui)
- uses DABC web server
- may control MBS via DABC proxy
- may control DABC readout directly
- can record DAQ performance trending

Data online monitoring:
- Go4 analysis framework (ROOT)
- reads DAQ samples via stream server socket (available both at MBS and DABC)
- same code can monitor either DAQ systems

mbspex components



pexornet components



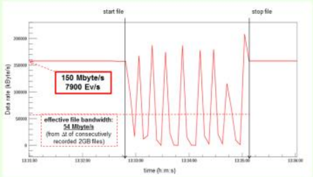
mbspex kernel module

- **character driver** accessible via */dev/pexor0*
- debugging and tuning *via /sys/class/mbspex/pexor0*
- all frontend and receiver control via custom file *ioctl()*
- concurrent access is protected by kernel mutex
- *mmap()* maps physical **DMA buffer memory outside kernel space** ("MBS pipe"), reserved at boot time,
- trigger interrupt handler changes wait semaphore to be evaluated in userland via *ioctl()*
- **explicit data request from MBS required for readout**
- **tailored for DAQ software framework MBS**

pexornet kernel module

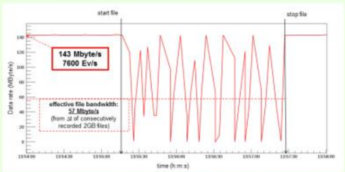
- **network driver** registered as interface *pex0*
- debugging and tuning *via /sys/class/net/pex0*
- all frontend and receiver control via socket *ioctl()*
- interface configuration with generic network tools
- concurrent control access is protected by kernel mutex
- **internal pool of DMA buffers** according the defined MTU
- **trigger interrupt bottom half does implicit data request, read out and preparation of socket buffers**
- readout is protected against control access by spinlock
- frontend data is delivered via generic *socket()* as **UDP packets from a virtual remote host**
- **various DAQ frameworks and other software may read and inspect data**

DAQ with MBS/mbspex



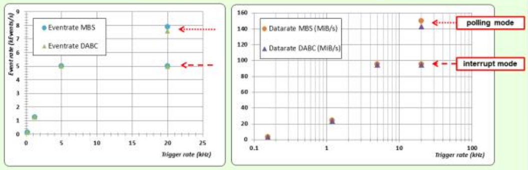
Data rate trending at MBS /mbspex readout of 2 FEBEX sampling ADCs at one PEXOR chain. Triggered by 20 kHz pulser. Host: 8 soft cores, 4GB RAM, kernel 3.2.0-amd64. Decrease of rate is due to file writing to NFS mounted disk.
CPU load: 3 x 100% (polling for frontend data mode, early trigger clear)

DAQ with DABC/pexornet



Data rate trending at DABC/pexornet readout of 2 FEBEX sampling ADCs at one PEXOR chain. Triggered by 20 kHz pulser. Host: 8 soft cores, 4GB RAM, kernel 3.2.0-amd64.Decrease of rate is due to file writing to NFS mounted disk.
CPU load: 95% ksoftirq, 60% dabc threads (polling for frontend data mode)
Lost UDP packets > 6840 x 20.3 kB = 136 MB (from local event counter check)

DAQ performance comparison



Comparison of data taking between MBS/mbspex and DABC/pexornet at different pulser trigger frequencies. Hardware setup with 2 FEBEX sampling ADC frontends at one SFP chain, as described above. Acquired data is not written to file, but checked for validity by online analysis at stream server socket with Go4 software. Both DAQ systems can fully handle event rates up to 5kHz (90MiB/s) for such setup. At 20kHz trigger rate, the "early trigger clear" readout mode can increase perfomance up to 8kHz (150MiB/s). In this mode, the TRIXOR trigger hardware is reset before data is read from the double buffered FEBEX front ends. Here in fact one DAQ process (MBS), or the kernel tasklet (pexornet), is polling for the "data ready" state of the GOSIP transmission, instead of waiting for the next trigger interrupt.

Conclusions

Control software and GUIs:
- both drivers allow concurrent control access during data taking
- pexornet can use all tools of mbspex
- pexornet can also use generic network tools

DAQ frameworks:
- mbspex is bound to MBS framework
- pexornet can be read out by any UDP receiver software
- pexornet can produce MBS data format with DABC receiver

DAQ performance:
- both drivers show similar performance
- pexornet may lose UDP packets depending on load (e.g. file i/o)!
- mbspex with MBS has no data loss - (instead: backpressure on hardware dead time due to explicit readout requests!)

pexornet TO DO:
- implement missing network hooks in kernel module (ethtool,...)
- performance tuning
- DMA into pre-allocated socket buffers?
- implement other readout protocol than UDP?

mailto: j.adamczewski@gsi.de

20-th IEEE Real-Time Conference, 5-10 June 2016, Padova, Italy