The Hardware Access Library

- A small FAQ
- What makes it user-friendly ?
- Performance
- Possible Extension
- Impossible Extension
- Status

For details on the library see talk on 7/11/ in Trigger Software Meeting

FAQ: Scope and Non-Scope of the library

- What for?
  - Userfriendly access to custom designed hardware modules.
  - Accelerate the writing of simple test programs.
  - Write fast code to configure hardware.
- Who can use it ?
  - Designer of test software for a new piece of hardware during debugging / test phase.
  - Designer of testbeam software.
  - Designer of DAQ-software in order to configure or monitor the status of the hardware.
- When can it be used ?
  - Always if high performance is not required.
  - Everything which is NOT DMA.

- When should it NOT be used?
  - If high performance is required (code for time critical data transfer in DAQ applications)
  - If you have strong reasons that you do not like that the Address-Map of your hardware is NOT hardcoded in a piece of code but resides in a configuration-file, database, ...
    (..in any way you could also write a C++ class which hard codes the AddressMap of your hardware...)
  - If you use labview I am not sure if you can profit from this library (...I do not know labview...)
- What technologies are supported ?
  - PCI hardware plugged into a Linux PC.
  - VME-access with one specific PC-to-VME interface (...which hopefully will be used by many groups in CMS...)
- Why nothing which already exists ?
  - I did not find anything similar elsewhere.
  - Nobody whome I asked knew about something similar.
  - It has been used in 2-3 projects and has been found useful (...well...by me and few other people...)

What makes it user-friendly

- You do not need to remeber addresses and masks:
  - The AddressMap of your hardware is written once in a configuration “file”.
  - If you change your hardware (e.g. “add a new register in your FPGA”) you just add a line in the configuration file.
  - Possibility to store sequences of write commands in “files” (examples: configure your PCI bridge, configure your VME interface, prepare your hardware for data taking (setting thresholds...). BUT: currently no fancy script-facility with variables, conditional statements etc)
  - You do not need to bother to shift bitfields around: This information is contained in the AddressMap. See example :

write (“thresholdX”, 0x00000005);

```
// AddressMap
// item          address       mask

thresholdX       0x00000c08    0x00000F00
thresholdY       0x00000c08    0x000000F0
```

read 0x00000c08	0x00000005 shift to mask
0xnnnnnnnn	0x00000500

0xnnnnnnnn & 0xFFFFF0FF | 0x00000500

result : 0xnnnnn5nn

- A sequencer excecutes series of write commands
  - use forany procedure which you might change (configuration paramenters)
- Easy write a class which performs more complicated tasks
  - Tasks which need “feedback” (i.e. which depend on the value of external parameters (for example values read back from your hardware, values given by software to your program)

Performance : what can you expect ?

- First measurement with PC / Linux / Pentium III 800 MHz
  - Time measured is the overhead due to the userfriendly features:
  - Write access like in example : 16 us
  - Write access without mask (only lookup of address) : 3us
  - Access to PCI device (memory access) in PC is directly memory mapped. No further delay due to software is expected.

Extensions

- Command to copy blocks of data around
  - Could use block transfer in VME
  - DMA in PCI will not be supported since there is no standard DMA engine in PC.

Impossible Extensions

- Locking of the device for exclusive access
  - should be done in higher hierachy (XDAQ framework : the XDAQ application should take care for this if necessary. There you have everything you need (which is “semaphores”)
- Access from many different processes, tasks, threads, …

  This is included in the XDAQ framework and therefore will not be implemented another time in this library

!!! It will stay simple, unfancy and slow !!!

BUT: **any new idea is welcome and will be seriously considered**

Status

- Everything (except the sequencer) is implemented.
- Testing will start NOW
  - the library will be used in the next DAQ-column prototypes (i.e. in the test-benches)
- The predecessor has been successfully used in the first column protoype (including a sequencer)