mbspex driver software for PEXOR/KINPEX readout boards

J. Adamczewski-Musch1, N. Kurz2, S. Linev1, and the FAIR@GSI project1
1 GSI, Darmstadt, Germany

Introduction
The GSI PEXOR/KINPEX (“PEX family”) PCIe boards were designed for data read out from various detector front-ends via optical SFP connections to an X86 PC host [1]. Communication between PEX and front-end hardware is handled via the gosip protocol [2]. For triggered data acquisition, the trigger module TRIXOR can be connected to PEX. The PEX boards have been applied for many years with the data acquisition framework MBS [3]. An improved Linux kernel module driver mbspex.ko has been implemented such that concurrent access from MBS and separate control processes is now possible. The new application library libmbspex provides higher level functionality to user space. Moreover, a command line tool gosipcmd allows inspection and configuration of any front-end register from an interactive shell, GUI, or remote web server. Fig. 1 shows these mbspex software components in a typical Linux host PC with MBS DAQ and several control applications.

X86 PC

![Diagram of mbspex software components]

Figure 1: Overview of mbspex software components on an X86 Linux node

Linux device driver
Access of any user program to PEX and the connected front-end hardware requires device driver software, usually consisting of a kernel module and optionally a corresponding user space library.

Kernel module
The mbspex kernel module merges the previous simple driver pexormbs for MBS data acquisition with the larger pexor driver for DABC and FESA frameworks [4]. It implements all basic gosip protocol functionalities, like initialization of chains, front-end field bus access, and “token” data request, as ioctl() functions. Additionally, there are DMA operations to send data to any destination pointer in physical memory (MBS "pipe", see Fig.1). All these ioctl() calls are protected by a kernel mutual exclusive semaphore. This allows concurrent access to the PEX device without crashing the system.

Since the kernel module keeps track of all initialized devices at the sfp chains, a “broadcast” i/o is possible: with one ioctl() the same value can be written to the same address on all devices of a chain, or of all chains. Furthermore, several registers of each frontend can be configured at once from a single ioctl() data bundle. This can be combined with broadcast mode and allows in principle to safely reconfigure all frontends at once while data acquisition read out is running.

On the other hand, all ioctl() calls of pexormbs driver are remained in mbspex driver with the same key values. So any legacy MBS code may ignore the “locked” ioctl() features and still work directly on the PEX board control registers. For this purpose file operation mmap() is still implemented to map the PEX board memory to virtual addresses of the MBS process. Alternatively, mmap() can map any physical PC memory to user space. MBS is using this to access the reserved “pipe” memory for subevent buffering.

Finally, mbspex.ko exports some PEX and TRIXOR registers via the kernel sysfs feature. The properties can be inspected by reading corresponding file handles under directory /sys/class/mbspex/.

User library
The libmbspex user space library is written in C language and uses the file system handle /dev/pexor0 with ioctl() calls as interface to the kernel module (see Fig.1). It provides high level functions for register i/o with the PEXOR board, with any single front-end, or with all configured front-end boards in a “broadcast” mode. Additionally, gosip “token mode” data transfer from the front-end buffers and DMA transfer to PC host memory can be initiated by simple function calls. All these functions are protected against concurrent access already in the kernel module. So different control applications like gosipcmd may link and use libmbspex simultaneously. Moreover, MBS user readout code can be based on libmbspex function calls only.

Application for MBS DAQ
The MBS DAQ framework does not operate the front-end hardware directly, but just ensures that user read-out functions are called whenever module TRIXOR receives a trigger signal. It does not require libmbspex functionali-
ty, but interacts with mbspex.ko by means of ioctl() and mmap() file operations. They are merely applied to wait for next trigger, and to map the pipe buffer physical memory (Fig. 1). These calls have been kept compatible with the previous kernel module pexormbs.ko, so no modifications to MBS framework have been needed. Also any legacy user readout code will work with mbspex.ko, since memory mapped access to PEX control registers is still supported.

However, to provide safe concurrent frontend access between MBS and external control tools, adjustments in MBS user readout code are necessary. Here any token data request must use primitive function calls of libmspex. An example of such readout code has been provided for POLAND/QFW front-ends of FAIR beam diagnostic projects [5].

**Command line tool gosipcmand**

The command line tool gosipcmand works as shell application on top of libmspex (Fig.1). It provides interactive command access to PEX board and the SFP-connected frontend registers via gosip protocol. The resulting values are printed to terminal. The main functionalities cover:

- reset PEX board, initialize SFP chains
- read/write any address on frontend slave
- incremental read/write from start address
- register bit manipulation
- broadcast mode: read/write same register at all connected frontends
- configure / verify with script files *.gos
- plain or verbose, hex or decimal output mode

A more complete list of available options can be printed using “gosipcmand -h”. At GSI gosipcmand is already provided at X86 Linux installations (hosts “X86L-nn”) for MBS v6.2.

**Frontend control GUI**

Since gosipcmand uses stdin/stdout as plain text data interface, it can serve as base for any special front-end configuration script, or graphical user interface (GUI) application.

**POLAND GUI**

An example of such frontend GUI has been developed for configuration of POLAND charge to frequency converters of beam diagnostics [5]. It is designed with Qt4 graphics library and shown as screenshot in Fig.2. Since it uses gosipcmand calls only, it is decoupled from the actual mbspex library version and may work both with mbspex and pexor driver installations, i.e. with MBS or FESA read out. The stdout of gosipcmand is redirected to an embedded text window which allows verbose register inspection, and dumping of event data buffers. PEX board and SFP chains may be initialized on click. Each POLAND frontend device can be selected and the meaningful registers displayed and manipulated. Moreover, it is possible to broadcast same register settings to all devices, as this is already supported at kernel module level. Also configuration scripts of gosipcmand (*.gos) may be selected and applied from the GUI.

The POLAND GUI is installed at GSI for MBS v6.2 on X86 Linux and available via alias “poland”.

**DABC webserver GUI**

Besides such local control GUI, a remote control of gosipcmand has been implemented as gosip plug-in for the webserver of software framework DABC [6]. This webserver runs as independent DABC process on the MBS Linux node (Fig.1) and provides a full interface to the local gosipcmand via HTTP request and response. A web browser version of the POLAND GUI has been implemented for this mechanism, using JavaScript with jQuery UI plug-ins (Fig.3).

Additionally, a generic gosipcmand browser command line GUI will be available as part of the DABC webserver controls for MBS v6.3 [7].
References


