The DAQ readout library vmelib *

B. Löher[1,2], A. Charpy[3], H. T. Johansson[3], H. Simon[2], H. T. Törnqvist[1], the R[3]B collaboration , and FAIR@GSI division

[1]Institut für Kernphysik, Technische Universität Darmstadt, Darmstadt, Germany; [2]GSI, Darmstadt, Germany;
[3]Chalmers Univ. of Technology, Göteborg, Sweden

Introduction

A wide variety of physics experiments depend on the correct initialization and a fast and safe readout of modular electronics connected to data acquisition (DAQ) computers via standardized data buses (e.g. VME, CAMAC, PCIe, or other). A unified and standalone library framework providing the facilities needed to handle the initialization, monitoring and readout tasks in a safe and easy way does not yet exist. Existing libraries are usually either bound to work within a certain DAQ environment, require changes in the code leading to frequent recompilation, are unsafe to use or not meant to be changed by the users at all. In this report we present VMELIB, a library to fill this gap by addressing the points above.

To be a viable option for inclusion in a new project, or even for replacing the specialized readout code in existing applications there are a number of requirements such a library should fulfill: configuration via text files on top of sane defaults; unified API, no dependence on environment; consistency checks (config and data); support standard and fast transfer modes as well as multi-event readout; provide logging and debugging facilities; allow command line diagnostics; unit tests for code quality; emphasis on portability and documentation. The realisation of a few of them in vmelib is discussed below.

Implementation

The lightweight library is written in pure ANSI C, and consists of several components (see Fig. 1): The *config* component reads and parses the configuration. The *crate* component represents crates (i.e. a data bus) and acts as basis for DAQ operation. Each crate contains modules connected to a common data bus. The *module* component represents a generic module. It exposes the common functional elements, such as creation, initialisation, and readout of a physical module, while hiding its specific implementation details. The *util* module contains auxiliary methods for logging, memory allocation and provides the vmelib API that is exported to the environment. The *test* module is a collection of automated offline test cases to ensure correctness of each vmelib component and to prevent regressions.

Configuration is done using text files, which declare a list of crates and their contained modules in a tree-like fashion. A complete configuration file can be very simple, here creating a new Crate object with ID 0 that contains two

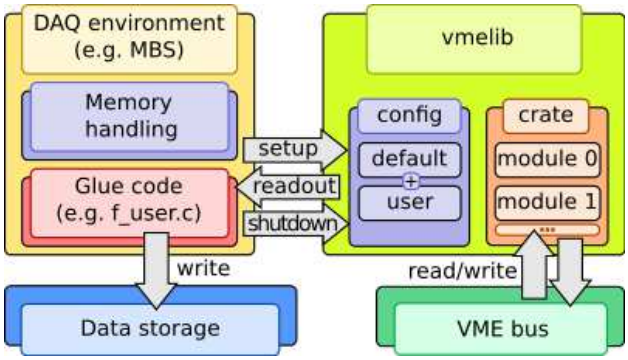

Figure 1: Schematic overview of vmelib (on the right), interacting with a DAQ environment and hardware.

modules with their respective bus addresses:

```
CRATE("VME0", 0) {    # Name and crate ID
 multi_event = true   # Use multi-event readout
 GSI_TRIDI(0x02000000) {}  # Type + address
 CAEN_V775(0x00010000) {
   common_start = true
   time_range = 1200 ns
}}
```

Default configurations are part of vmelib and the reason behind crate configurations being concise.

Vmelib provides the functionality to set the configuration and to read data from the modules, leaving data storage and other tasks to the environment. Therefore, the necessary interface to interact with vmelib using a single VME crate consists of only a few function calls:

```
Crate *vmelib_setup(LogCallback, char *cfg_file);
void crate_readout_prepare(Crate *crate);
uint32_t crate_readout(Crate *crate, void **mem);
void vmelib_shutdown(Crate *crate);
```

The complete API consists of a few more functions to allow fine-grained control, where needed.

Use cases

Vmelib has been used in a variety of experimental setups, both for testing and production. The main user is the R[3]B collaboration, which has used vmelib in experiments together with the MBS data acquisition starting from 2014 at GSI and Riken. The HIγS facility at TUNL has used vmelib successfully for their γ^3 experiments. The flexibility of vmelib allowing for quick changes of module parameters has been particularly appreciated in test setups.

* Work is supported by HIC for FAIR, GSI-TU Darmstadt cooperation PSP codes 1.2.2.4, 1.2.5.1.4, and the BMBF project 05P12RDFN8.

DOI:10.15120/GR-2015-1-MU-NUSTAR-NR-08