New DCS Libraries for the EMC

Tobias Triffterer

Ruhr-Universität Bochum – Institut für Experimentalphysik I 31.05.2022

CAN and SocketCAN

- Many devices of the EMC use CAN Bus
- CAN: Controller Area Network
- Simple Field Bus
- Two Wires (CAN High + CAN Low), GND optional

- SocketCAN: API of the Linux Kernel to treat CAN bus as a network protocol
- As all Linux Kernel APIs based on C, not C++

⇒ Create reusable library to use SocketCAN comfortably in C++

Class CanBusConnection

- Frontend for the library user
- Move-constructible through “pimpl pattern”
- Sending a CAN frame:
  `void sendCanFrame(const CanFrame& canframe)`
- Setting filters for receiving CAN frames
- Filters evaluated by the Kernel
- Reception via Callback, no read method

⇒ `std::function<void(const CanFrame&)> Callback`

Class CanFrame

- Encapsulates a single CAN frame the C++ way
- Convertible to and from `struct can_frame` of the Kernel
- Construction Example (DLC determined automatically):

```
const CanFrame request
{
  _address,
  CanFrame::CanFrameData
  {
    OpCodes::ReadWiperPositionAllChannels,
    boardid
  }
};
```

Error Handling

- Errors handled via C++ Exceptions
- `BadCanFrameMetadata`
  - Address out of range
  - Too many bytes in `CanFrameData`
- `SystemCallException`
  - Any call to the kernel fails
  - Wrapper for C’s `errno`
  - Provides name of failed system call, error code and error message for said error code

New LED Pulser Library

- LED Pulser: Custom device for the EMC to create light pulses that are similar to the scintillation of $PbWO_4$
- Check the readout chain and monitor radiation damage
- Controlled via CAN bus

- `libledpulser2.so` incorporates CAN bus library via static linking (can be changed via compile option)
- `libledpulser2.so` depends only on C++ standard libraries and the Linux kernel
- Only selected interface classes exported in symbol table and thus accessible to the library user, internal implementation protected

Library for FEMC HV Control Board

- EMC uses Avalance Photo Diodes (APDs) as photodetectors except in forward region close to beam pipe
- Gain of APDs varies heavily with operating voltage (approx. 7 %/V)
- One supply voltage for eight APDs due to space constraints
- HV Control Board on backplane for fine-tuning voltage of each APD
- 16 read operations, 5 write operations
- Read: Synchronous and asynchronous requests
- Write: Always wait for confirmation from board
- Callbacks based on `std::function` for every read operation available

(A)synchronous Details

- “Request Log” to communicate between threads
- Each operation waiting for a reply creates entry in request log and submits request on CAN bus
- When data from the board is received, the first matching open entry in the request log is closed and the received data is stored
- Requesting thread retrieves data from request log and returns it to user
- If no reply is received within `WaitDuration`, a `TimeoutException` is thrown

Code Example

```
struct RequestLogEntry final
  {
      OpCode opcode {};
      Modifier modifier {};
      OutputChannel outchannel {};
      InputChannel inchannel {};
      std::atomic_bool replyReceived = false;
      std::atomic_bool replyDataStored = false;
      Reply reply {};
  };
```

Full code:

- `https://gitlab.ep1.rub.de/dcs/libemchvboard/-/blob/main/src/emchvboardimplementation.h`
- `https://gitlab.ep1.rub.de/dcs/libemchvboard/-/blob/main/src/emchvboardimplementation.cpp`

EPICS Integration

- Libraries intentionally independent of EPICS
- Plan for these libraries:
    - Independent library
    - Stand-alone test application
    - Device support based on library to connect to EPICS
- Advantages:
    - Easier to debug
    - Less complexity
    - EPICS sometimes overkill for lab tests
    - Proper isolation of components

Outlook

- New CAN Bus Library used by several projects of my own

⇒ It works!

- In the future extension to CAN-FD
- Interface can be simplified using C++20 and variadic templates

- Code available at EP1 Gitlab: `https://gitlab.ep1.rub.de/dcs/CanBusToolbox/`
- If no account, ask me.