Device Error Handling in ChimeraTK.

Martin Killenberg

5th December 2019

8th MicroTCA Workshop for Industry and Research
DESY, Hamburg
Device errors in control applications

Start a server without the hardware

- No error handling (bad)
  - just crashes, don’t know why

- Typical error handling (OK)
  - Tells user it can’t reach hardware
  - Quits

- Ideal error handling (Good)
  - Server starts
  - Reports device error to the control system
  - Normal operation once device is available
Device errors in control applications

Start a server without the hardware

- No error handling (bad)
  - just crashes, don’t know why

- Typical error handling (OK)
  - Tells user it can’t reach hardware
  - Quits

- Ideal error handling (Good)
  - Server starts
  - Reports device error to the control system
  - Normal operation once device is available

Device error while server is running

- No error handling (bad)
  - just crashes, don't know why

- Typical error handling (tedious)
  - Catch errors wherever you access the hardware
  - Take appropriate action

- Ideal error handling
  - ???

We have noticed that a large fraction of code in control applications is error handling. The error handling strategy is usually the same: report error to control system, wait until error has gone, resume operation.

⇒ Lots of concepts and code copied (even inside one application)!
Device errors in control applications

Start a server without the hardware

- No error handling (bad)
  - just crashes, don’t know why

- Typical error handling (OK)
  - Tells user it can’t reach hardware
  - Quits

- Ideal error handling (Good)
  - Server starts
  - Reports device error to the control system
  - Normal operation once device is available

Device error while server is running

- No error handling (bad)
  - just crashes, don’t know why

- Typical error handling (tedious)
  - Catch errors wherever you access the hardware
  - Take appropriate action

- Ideal error handling
  - ???

We have noticed that...

- a large fraction of code in control applications is error handling
- error handling strategy is usually the same
  - Report error to control system
  - Wait until error has gone
  - Resume operation

⇒ Lots of concepts and code copied (even inside one application)!

M. Killenberg (DESY)
**Modules**

- Input/output variables
- Application Modules
  - One thread per module
ChimeraTK ApplicationCore

Modules

- Input/output variables
- Application Modules
  - One thread per module
- Special modules
  - Device module
  - Control system module

Connections

Mostly auto-generated

High locality

Algorithms don’t need to know how variables are connected

Perfect modularity, as modules are self-contained

M. Killenberg (DESY)
ChimeraTK ApplicationCore

Modules
- Input/output variables
- Application Modules
  - One thread per module
  - Special modules
    - Device module
    - Control system module

Connections
- Mostly auto-generated

High locality
- Algorithms don’t need to know how variables are connected
- Perfect modularity, as modules are self-contained
void Controller::mainLoop() {
    while(true) {
        temperatureReadback.read(); // waits until temperatureReadback has been updated
        temperatureSetpoint.read(); // update the temperature setpoint

        heatingCurrent = gain * (temperatureSetpoint - temperatureReadback);

        heatingCurrent.write();
    }
};

- Process variables are represented by inputs and outputs
- They behave like normal numbers with additional read() and write()
void Controller::mainLoop() {
    while(true) {
        temperatureReadback.read(); // waits until temperatureReadback has been updated
        temperatureSetpoint.read(); // update the temperature setpoint
        heatingCurrent = gain * (temperatureSetpoint - temperatureReadback);
        heatingCurrent.write();
    }
}

- Process variables are represented by inputs and outputs
- They behave like normal numbers with additional `read()` and `write()`

How does this help with device error handling?
Each process variable has an error handling loop

- Reading from the device backend can cause a runtime error
- An error message is send
- Process variable waits for recovery message

```cpp
ProcessVariable::read(); // waits until temperatureReadback has been updated

ProcessVariable::read(); // update the temperature setpoint
```
Device module thread

- Open the device backend at application start
- Wait for error messages
Device module thread

- Open the device backend at application start
- Wait for error messages
- Send error message to control system
- Try to re-open the device backend (inner loop)
Device module thread

- Open the device backend at application start
- Wait for error messages
- Send error message to control system
- Try to re-open the device backend (inner loop)
- Send OK to control system when successful
- Send recovery message to all process variables
Device module thread

- Open the device backend at application start
- Run initialisation sequence
- Wait for error messages
- Send error message to control system
- Try to re-open the device backend (inner loop)
- Try to re-initialise the device
- Send OK to control system when successful
- Send recovery message to all process variables
Device error handling

The whole picture

ProcessVariable::read():
- read request
- backend.read()
- OK: send value, report error, wait for recovery
- runtime_error: report error, wait for recovery

Device module thread:
- initialise
- wait for error
- send status "error" to CS
- re-open backend
- OK: send status "OK" to CS, report successful recovery
- runtime_error: report error

M. Killenberg (DESY)
ChimeraTK
- design modular, multi-threaded applications
- talk to hardware
- interface with the control system infrastructure

Device error handling in ApplicationCore
- build into the framework
- available out of the box (no extra code required)
- option to initialise device after (re-)connection
Software Repositories

- ChimeraTK source code: https://github.com/ChimeraTK
- Ubuntu 16.04 packages are available in the DESY DOOCS repository.

Documentation and Tutorials

- API documentation https://chimeratk.github.io/
- Tuesday’s tutorials on the MicroTCA Workshop Indico page
- e-mail support: chimeratk-support@desy.de
Backup
What else is new in ApplicationCore?

**Status monitor**
- Check value for upper threshold, lower threshold or window
- Threshold for error and warning
- Pre-defined status results
  - OK
  - Error
  - Warning
  - Intentionally off
- Work in progress: **Automatic status aggregator**

**Hierarchy modifier**
- Model your variable content to fit the process view (not how you have to implement it in C++)
- Enables automatic connection of variables
  ⇒ Even easier connection code