TUPV012

Automated device error handling in control applications

M. Killenberg*, J. Georg, M. Hierholzer, C. Kampmeyer, T. Kozak, D. Rothe,
N. Shehzad, J. H. K. Timm, G. Varghese, C. Willner,
Deutsches Elektronen-Synchrotron DESY, Hamburg, Germany

Goal: Application code should not have to care about device connections and errors

Requirement 1: Framework provides common API for all devices → DeviceAccess
Requirement 2: User code can always read and write all its variables → ApplicationCore

Slide 2: ChimeraTK

C++ framework for control applications

- DeviceAccess
- ApplicationCore
- ControlSystemAdapter

Slide 3: ApplicationCore

- Multi-threaded applications
- Self-contained application modules

Slide 4: Data Validity Propagation

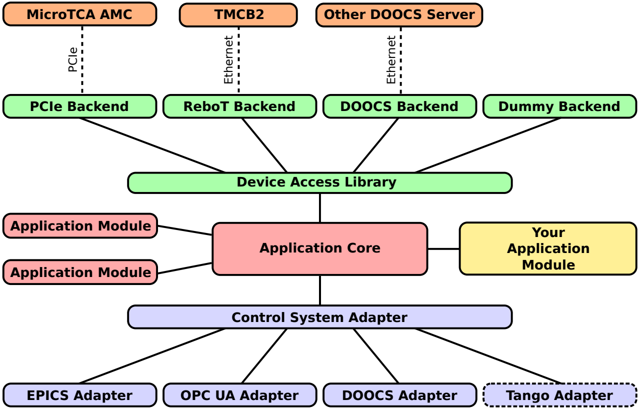
- Framework handles and reports device errors
- Data validity ‘faulty‘ is automatically propagated

Slide 5: Initial Value Propagation

- Clean application start
- Devices are initialised
- Modules are started in the correct order

ChimeraTK

Goal: Application code should not have to care about device connections and errors
Requirement 1: Framework provides common API for all devices → DeviceAccess



DeviceAccess

- Abstract access to various hardware
- Extensible backend interface

ApplicationCore

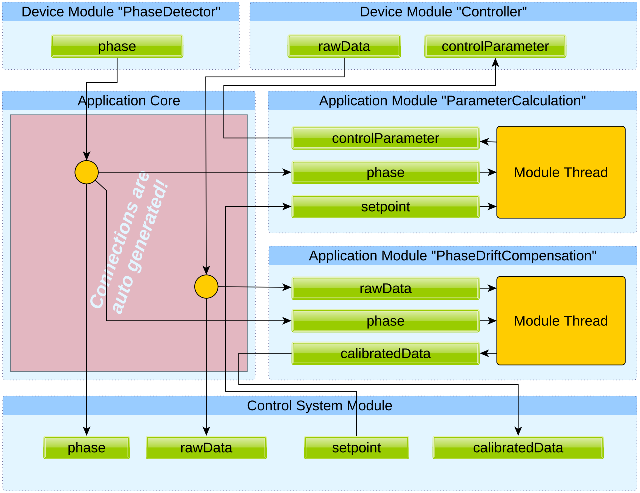
- Application modules implement application logic
- Multi-threaded

ControlSystemAdapter

- Abstract interface to various control system middleware
- Integrate via configuration

ApplicationCore

Goal: Application code should not care about device connections and errors
Requirement 2: User code can always read and write all its variables → ApplicationCore



Modules

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

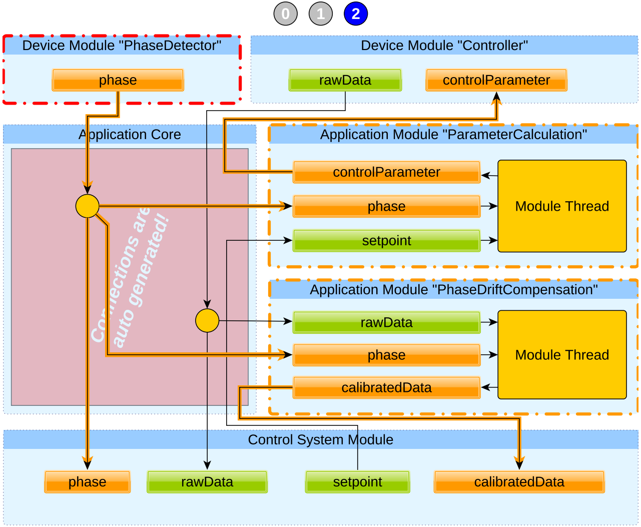
Connections

- Auto-generated from variable names

High locality

- Algorithms don't need to know how variables are connected
- Modules are self-contained
- Framework takes care about device connections and error handling

Data Validity Propagation



0 **Starting point**

- Application is running normally

1 **Error detected in "PhaseDetector"**

- Device module reports error to control system
- All variables are flagged as faulty and send with previous value

2 **Automatic data validity propagation**

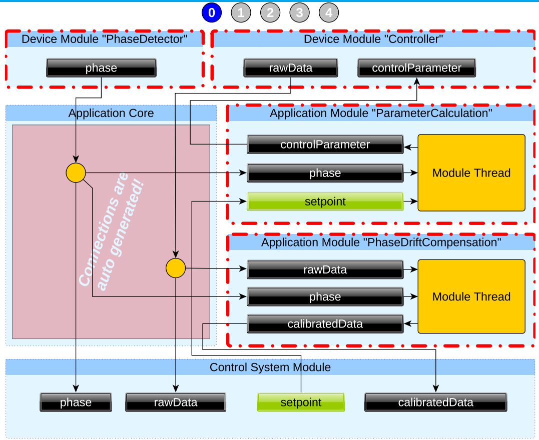
- Application modules are small and self-contained[1]
- All outputs depend on all inputs
- If one inputs is 'faulty' automatically all outputs are flagged as 'faulty'

In the background

- Framework tries to re-establish the connection to the hardware

[1] Device modules and the control system module are not small and self contained ⟶ Outputs don't depend on inputs. No automatic data validity propagation.

Initial Value Propagation — Start devices and modules in correct order



0 Application start

- Both devices still closed
- Modules waiting for initial value
- Control system module is getting initial value for `setpoint` from persistency layer

1 Devices are opening

- Initialisation sequence for both devices
- "Controller" still waiting for initial values
- "PhaseDetector" sends `phase`

2 "ParameterCalculation" is starting

- Sends `controlParameter`

3 Device "Controller" is starting

- Sends `rawData`

4 "PhaseDriftCompensation" is starting

- Sends `calibratedData`
- The application is up and running!

ChimeraTK is published under GNU GPL or GNU LGPL.

- Source code: `https://github.com/ChimeraTK`
- Ubuntu 20.04 packages: DESY DOOCS repository.
- e-mail support: chimeratk-support@desy.de