Introduction to the ChimeraTK Control System Adapter.

Martin Killenberg

M. Hierholzer, C. Schmidt, *DESY, Hamburg, Germany*
S. Marsching, *aquenos GmbH, Baden-Baden, Germany*
C. P. Iatrou, *Technische Universität Dresden, Dresden, Germany*
R. Steinbrück, M. Kuntzsch, *HZDR, Dresden-Rossendorf, Germany*

13th July 2016

4th ARD ST3 Workshop - Conctrol System Adapter Session, Berlin

HELMHOLTZ ASSOCIATION

µTCA®

Typical Scenario: Integrating a small device

Integrate the device into your EPICS environment

- Just a few Process Variables

⇒ Write a new EPICS IOC, not too much work...

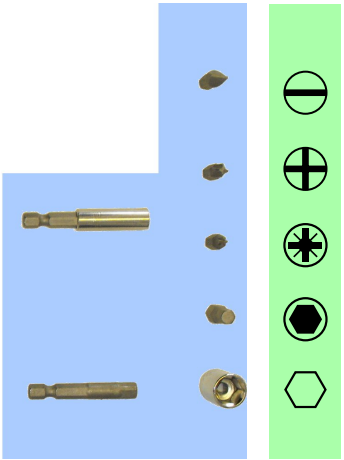
Integrate the same device into a DOOCS environment

- Just a few Process Variables
- DOOCS and EPICS are very different, not much code to reuse: Better start from scratch

⇒ Write a new DOOCS device server, not too much work...

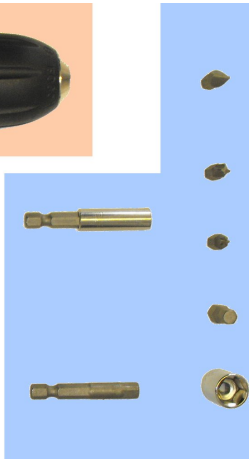
Sometimes You Need An Adapter

Sometimes You Need An Adapter



Sometimes You Need An Adapter

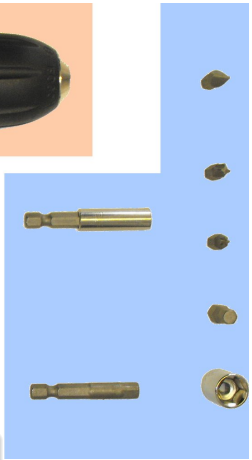
Device



Adapter

Control System

Sometimes You Need An Adapter



Adapter

Control System

EXAMPLE: LLRF Server

- $\mathcal{O}(400)$ process variables
- iterative learning algorithm
- feed forward table calculation

EXAMPLE: Target Control Systems

- DOOCS at FLASH,XFEL/DESY
- EPICS 3 at FLUTE/KIT
- WinCC/OPC-UA at ELBE/HZDR

Control System Adapter

Task

Complex control algorithms should be used with different control systems.

Control System Adapter

Task

Complex control algorithms should be used with different control systems.

Requirements For Abstraction

- Keep application code control system independent
- The algorithm must interact with the control system
- Use functionality provided by the control system
- No device-dependent code on the control system side

Additional Requirements:

- Thread-safety
- Real-time capability
- Must not copy large data objects (arrays)

Control System Adapter

Task

Complex control algorithms should be used with different control systems.

Requirements For Abstraction

- Keep application code control system independent
- The algorithm must interact with the control system
- Use functionality provided by the control system
- No device-dependent code on the control system side

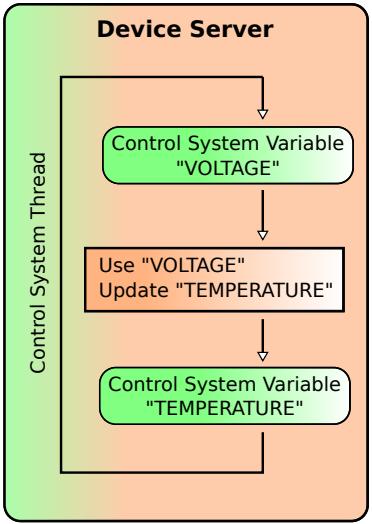
Additional Requirements:

- Thread-safety
- Real-time capability
- Must not copy large data objects (arrays)

First Implementation

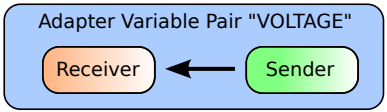
- Process variables to transfer data to/from the control system

A Typical Device Server (Without Adapter)

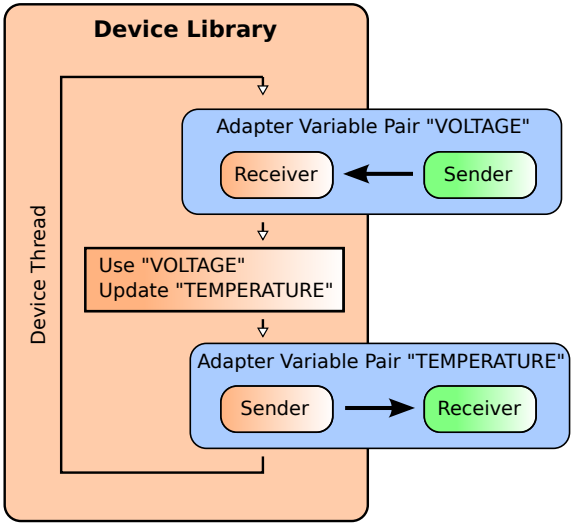


- Control system data types used inside the algorithm
- Control system variables can be locking/blocking
- Control system variables might not be thread safe
- Threading often handled by control system

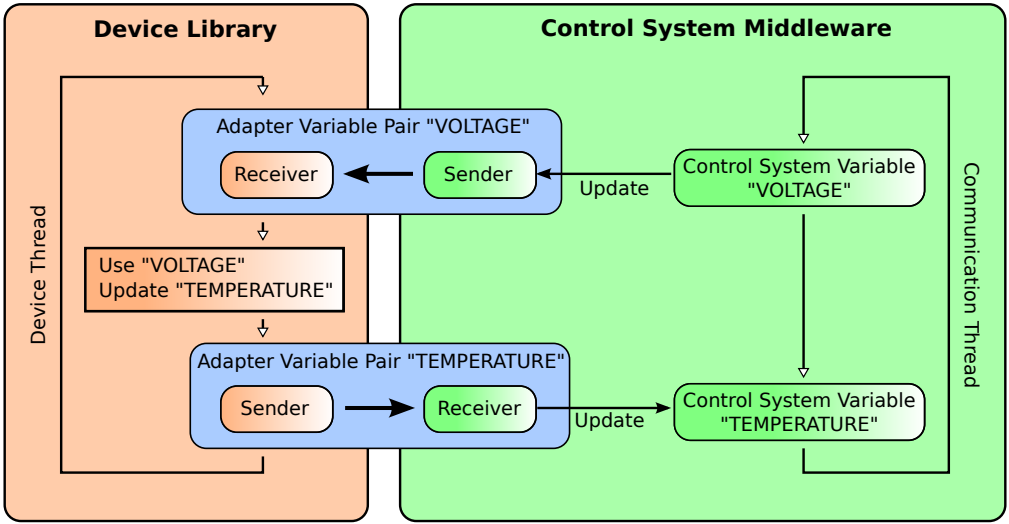
A Device Using The Control System Adapter



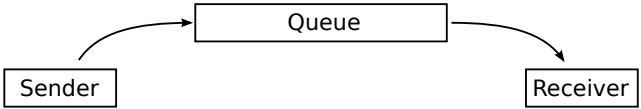
A Device Using The Control System Adapter



A Device Using The Control System Adapter

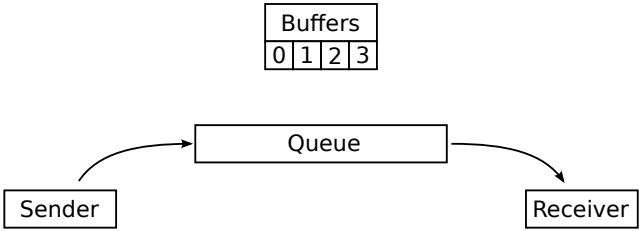


Implementation Of The Sender/Receiver Pair



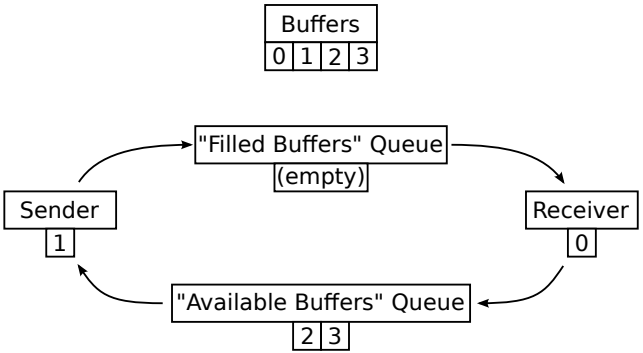
- Lock-free queue

Implementation Of The Sender/Receiver Pair



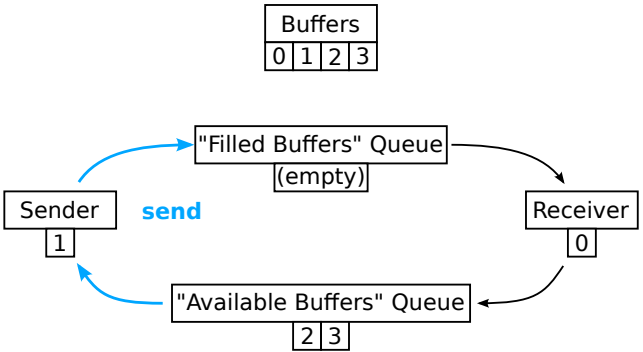
- Lock-free queue
- Pre-allocated buffers for arrays

Implementation Of The Sender/Receiver Pair



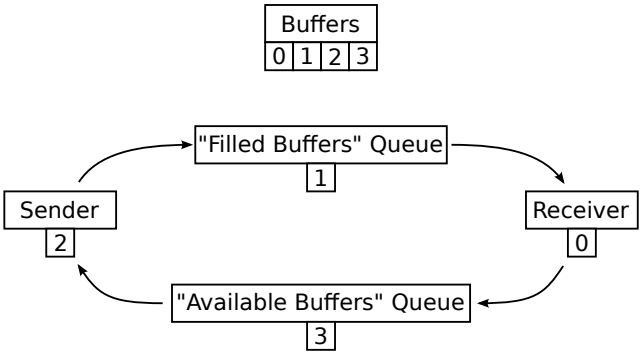
- Lock-free queues
- Pre-allocated buffers for arrays
- Copy references, not buffers

Implementation Of The Sender/Receiver Pair



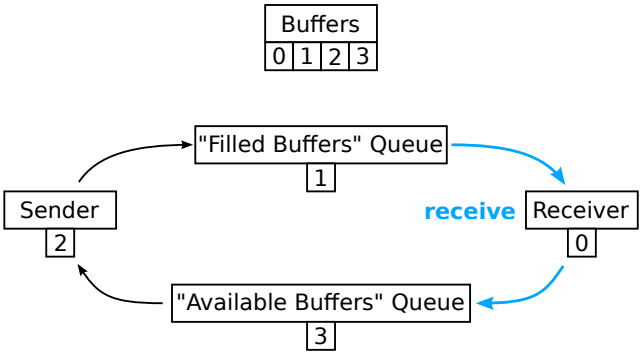
- Lock-free queues
- Pre-allocated buffers for arrays
- Copy references, not buffers

Implementation Of The Sender/Receiver Pair



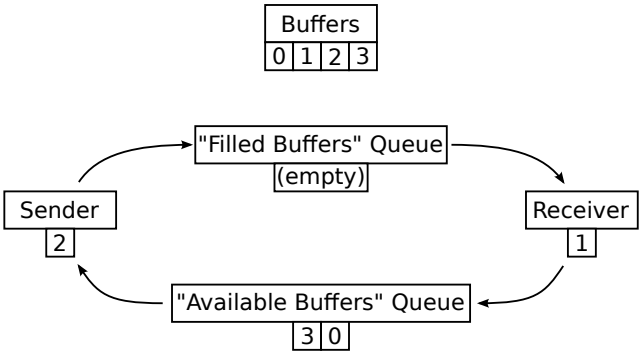
- Lock-free queues
- Pre-allocated buffers for arrays
- Copy references, not buffers

Implementation Of The Sender/Receiver Pair



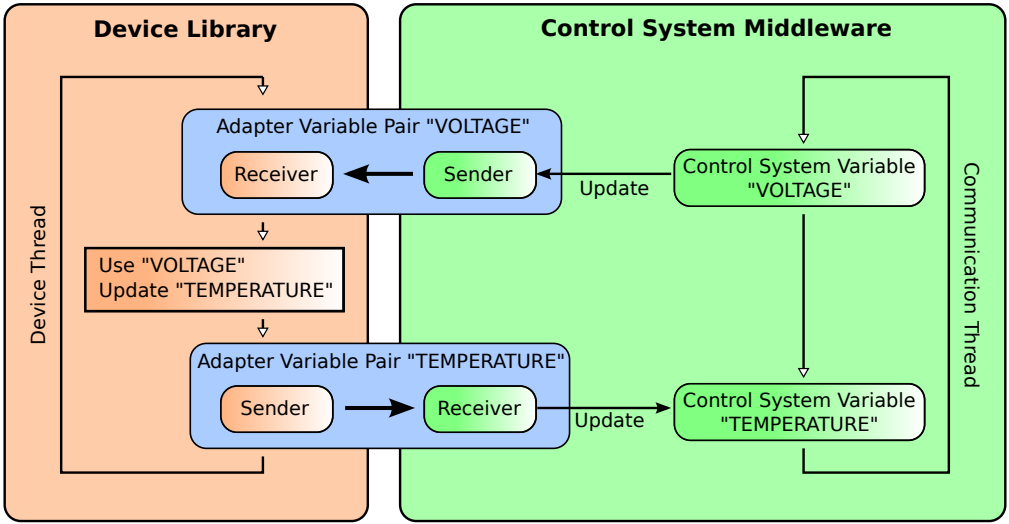
- Lock-free queues
- Pre-allocated buffers for arrays
- Copy references, not buffers

Implementation Of The Sender/Receiver Pair

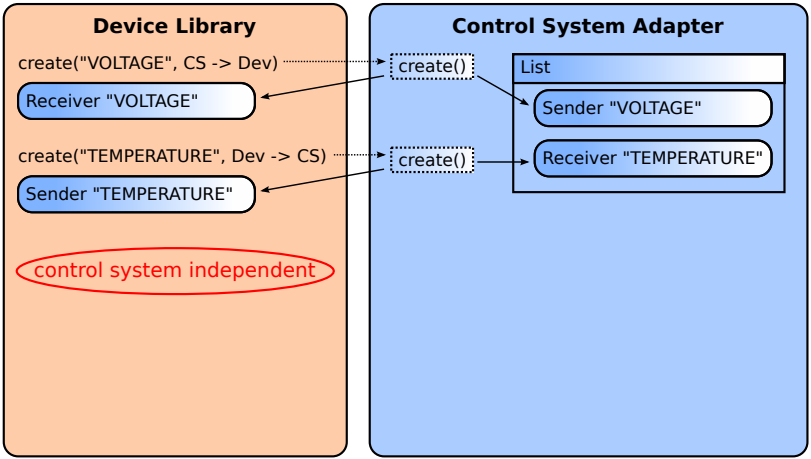


- Lock-free queues
- Pre-allocated buffers for arrays
- Copy references, not buffers

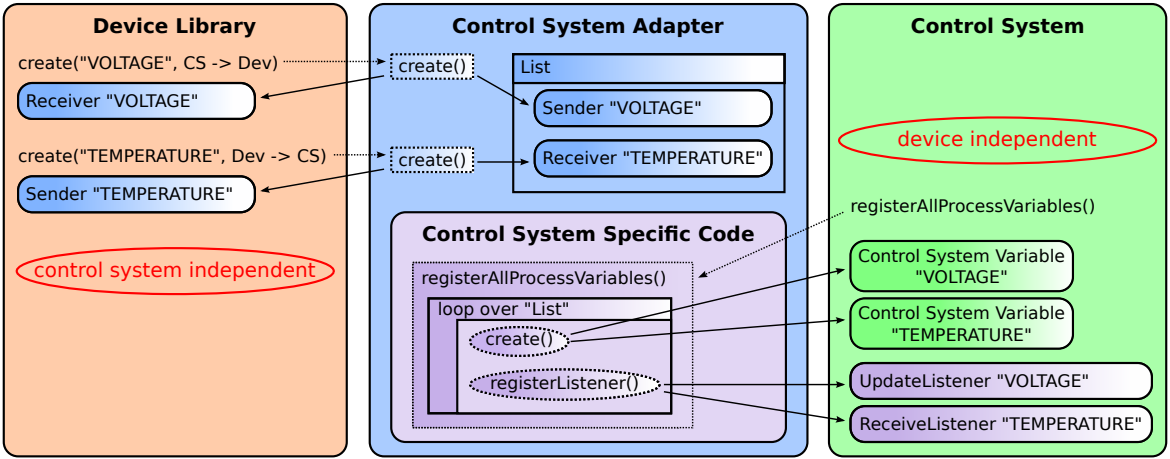
A Device Using The Control System Adapter



Registering Process Variables



Registering Process Variables



Status of the Control System Adapter

Adapter for Process Variables

Decouple application logic and control system

- Generic part
- Control system specific part
  - Implementations for DOOCS and EPICS 3
  - OPC-UA adapter currently being implemented

Red: Topics for today's meeting

Design Goals

- Control system independent process variables ✓
- Thread safety ✓
- Real time capability ✓
- Minimise copying ✓
- No device-dependent code on control system side (✓)

Next Steps: Tools for System Integration

- Name mapping for Process Variables
- Access to control system features
  - Limits
  - History
  - Engineering units
  - Handle alarms
- More sophisticated data objects
  - Currently only scalars and arrays

Name Mapping

- Each facility's control system has it's own naming scheme
- Each device logic provides Process Variables with an (internal) name

⇒ We need a name mapping during system integration

Planned tools of the adapter:

- XML file provided by the application part with list of Process Variables
- Middleware implementation extends/uses this file to
  - add middleware specific features
  - implement the binding to the middleware Process Variable instances

→ See next presentation

Accessing Control System Features

- To make the device code really CS independent, it must not rely on certain features to be present in the CS
- All features added to the CSA must be either:
  - implementable for any CS, or
  - non-essential and thus optional.
- Ideally: default implementation for each feature based on the current CSA interface (thus CS independent)
- For non-essential features, the default implementation does nothing
- CS-specific implementations can be added for CS which support the feature directly

Slide by Martin Hierholzer

Plugin mechanism for extra features

- Developers should be able to add new features fitting this scheme without modifying the CSA package
- Suggestion: Put all new features into plugins, to have a coherent interface
- Plugins can be loaded for each ProcessVariable
- Each plugin can provide its own configuration interface exposed to the device code
- Default implementation should always just use existing CSA features

Slide by Martin Hierholzer

List of features (for discussion)

- Guaranteed limits on controlSystemToDevice variables
- Device-provided validators for controlSystemToDevice variables
- History (non-essential optional feature)
- Engineering units
- Alarms
- Save and restore values for groups of variables
- Conversion to selectable engineering units (e.g. optical delay stage: mm or ps)

Potentially problematic features:

- DAQ
- Combined data types

Original slide by Martin Hierholzer

List of features (for discussion)

- Guaranteed limits on controlSystemToDevice variables
- Device-provided validators for controlSystemToDevice variables
- History (non-essential optional feature) → system integration, solved per middleware plugin
- Engineering units → Not a plugin, part of PV API
- Alarms
- Save and restore values for groups of variables
- Conversion to selectable engineering units (e.g. optical delay stage: mm or ps)→ Not a plugin, implement later, system integration?

Potentially problematic features:

- DAQ
- Combined data types

Original slide by Martin Hierholzer

Further points for discussion

- Do we need optional features? (Or are they all system integration?)
- How to implement combined data types?
- Do we want/need display limits on the device side?

Summary

ChimeraTK was formerly known as **MTCA4U**

Control System Adapter

- Basic implementation to read/write Process Variables ready
- Next steps: System integration and accessing control system features

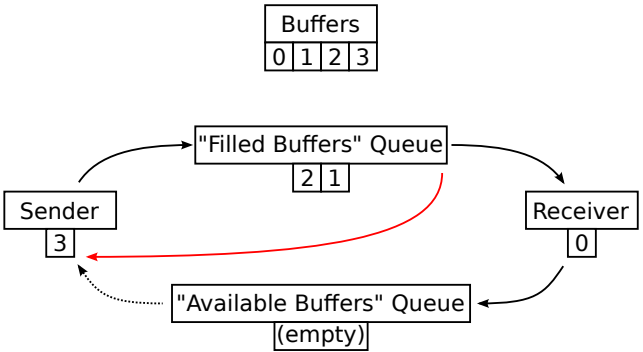
Software Repositories All software is published under the GNU GPL or the GNU LGPL.

- ChimeraTK: https://github.com/ChimeraTK
- EPICS 3 Adapter: http://oss.aquenos.com/svnroot/epics-mtca4u/

Backup

A Slow Receiver

Update the queue if the receiver is slow/down



- No free buffers for the sender
- Overwrite the oldest buffer
- Pop the head of the "filled buffers" queue (buffer 1)
- Send the buffer which has just been filled (buffer 3)