Integrating Control Applications Into Different Control Systems.

The MTCA4U Control System Adapter

Martin Killenberg

M. Hierholzer, C. Schmidt, *DESY, Hamburg, Germany*
S. Marsching, *aquenos GmbH, Baden-Baden, Germany*
J. Wychowaniak, *Łódź University of Technology, Łódź, Poland*
M. Kuntzsch, R. Steinbrück, *HDZR, Dresden, Germany*
C. Iatrou, L. Urbas, *TU Dresden, Dresden, Germany*

08th March 2016

Helmholtz M&T Annual Meeting — Satellite Session, KIT Karlsruhe

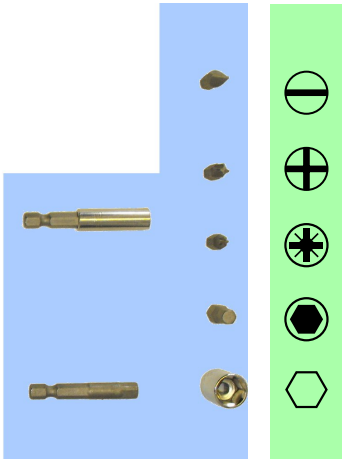
HELMHOLTZ ASSOCIATION

The MTCA4U Control System Adapter

- It is a adapter for **software applications**
- It is not related to any particular hardware, especially **not** to **MicroTCA**
- It is a **stand-alone** component
  (although developed as part of the MTCA4U tool kit)

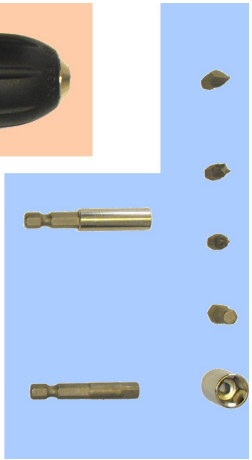
Sometimes You Need An Adapter

Sometimes You Need An Adapter



Sometimes You Need An Adapter

Device



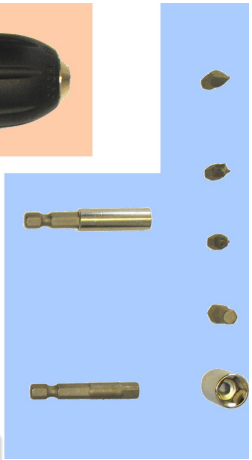
Adapter

Control System

Sometimes You Need An Adapter

EXAMPLE: LLRF Server

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



Adapter

Control System

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
- Minimise 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
- Minimise device-dependent code on the control system side

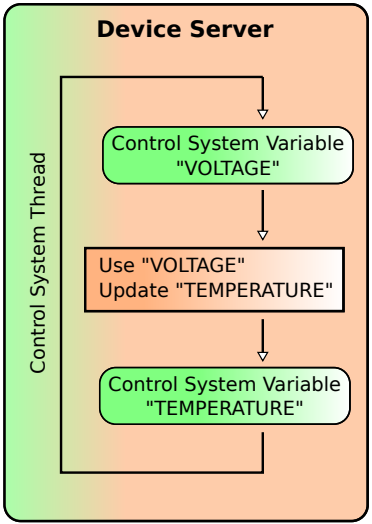
Additional Requirements:

- 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

Strong coupling of the device logic to control system details

Different Control Systems

Slide by Sebastian Marsching on last year's Control System satellite session

Comparison of Control Systems

Control System	Device Description	Device Model	Mutex
DOOCS	code based	object oriented	per group
EPICS	configuration based	channel based	per PV
TANGO	code based	object oriented	?
WinCC OA	configuration based	channel based	no (single threaded)

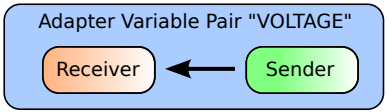
Plus different handling of

- limits
- alarms
- engineering units
- etc.

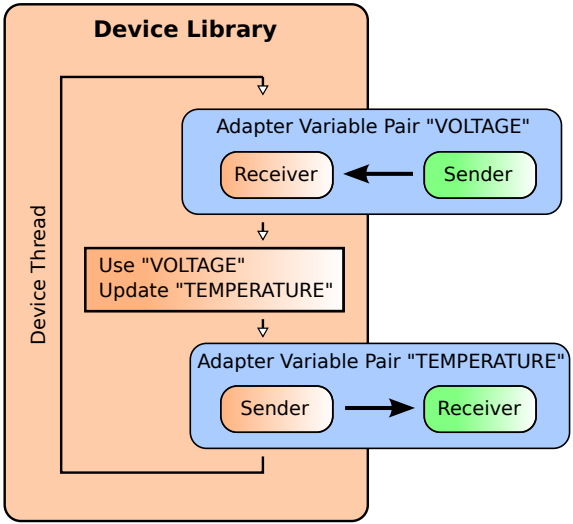
Completely different locking schemes

- Device code cannot access CS variables through a wrapper
- You need a separate copy on the device side

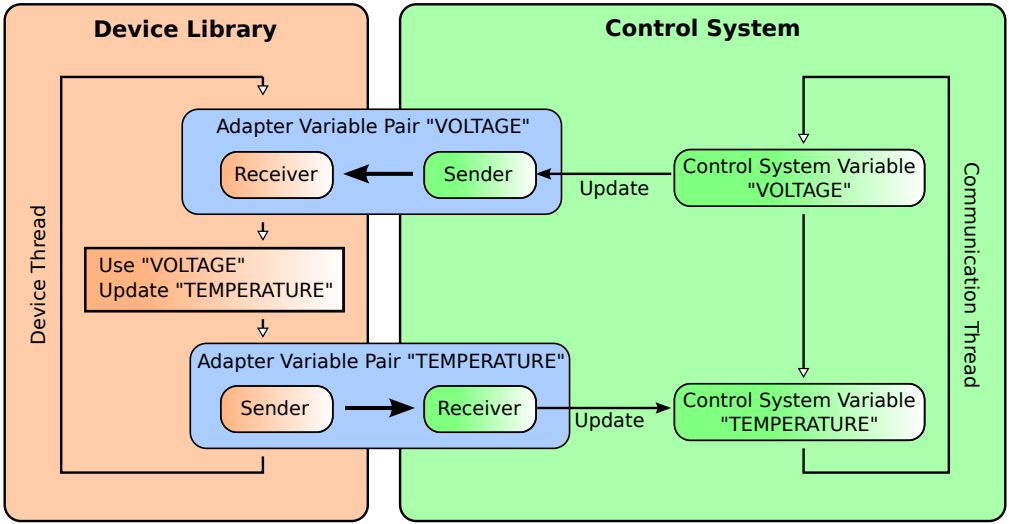
A Device Using The Control System Adapter



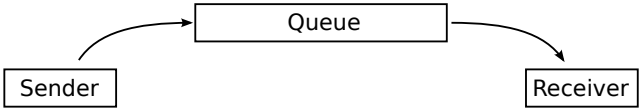
A Device Using The Control System Adapter



A Device Using The Control System Adapter

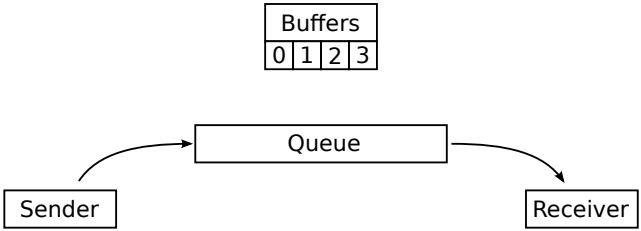


Implementation Of The Sender/Receiver Pair



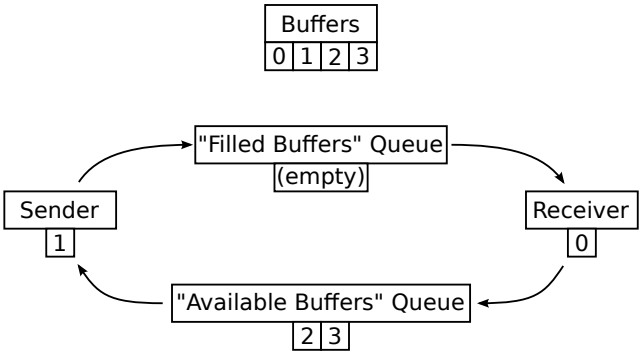
- Lock-free queue

Implementation Of The Sender/Receiver Pair



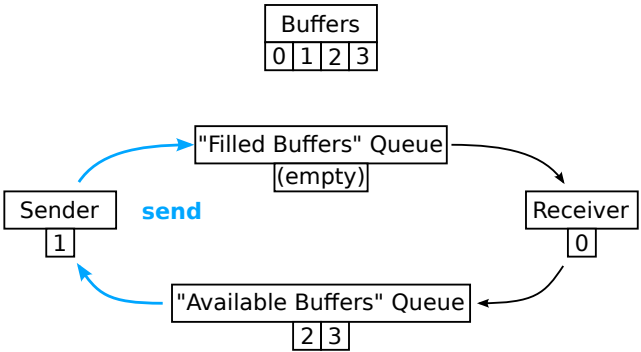
- Lock-free queue
- Pre-allocated buffers for arrays

Implementation Of The Sender/Receiver Pair



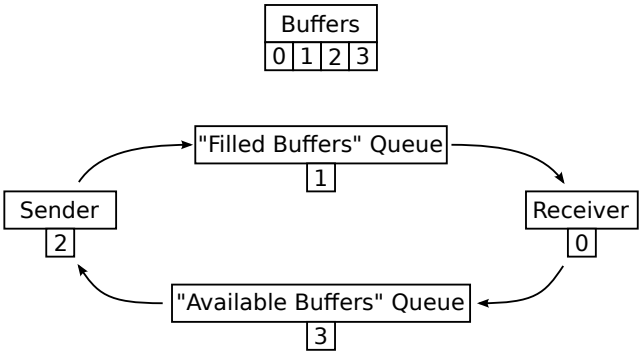
- Lock-free queues
- Pre-allocated buffers for arrays
- Copy references, not buffers

Implementation Of The Sender/Receiver Pair



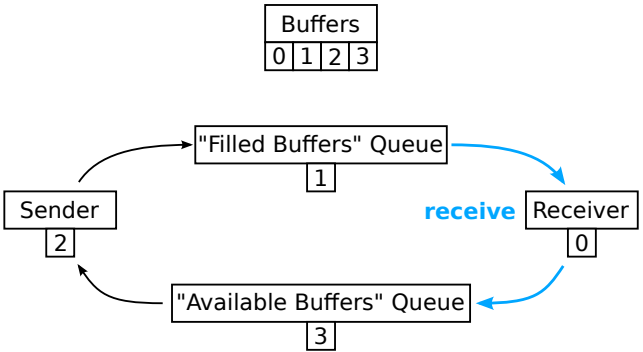
- Lock-free queues
- Pre-allocated buffers for arrays
- Copy references, not buffers

Implementation Of The Sender/Receiver Pair



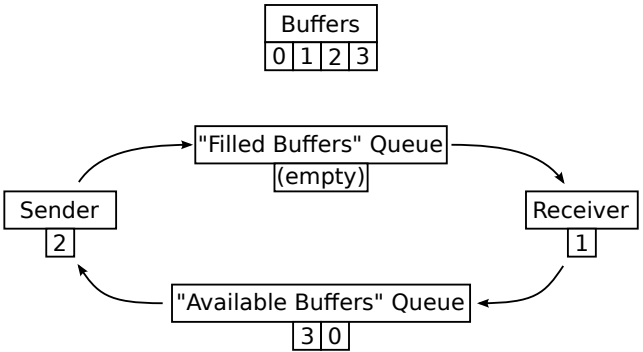
- Lock-free queues
- Pre-allocated buffers for arrays
- Copy references, not buffers

Implementation Of The Sender/Receiver Pair



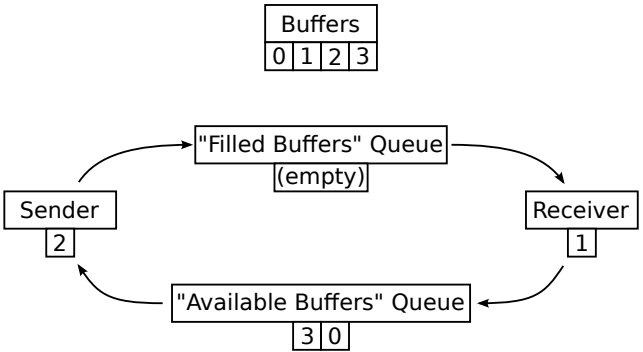
- Lock-free queues
- Pre-allocated buffers for arrays
- Copy references, not buffers

Implementation Of The Sender/Receiver Pair



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

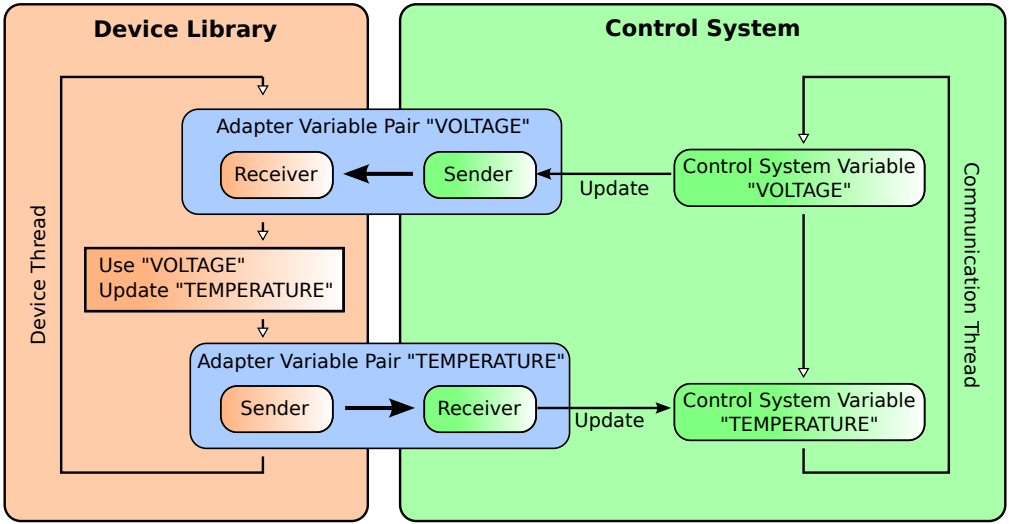
Implementation Of The Sender/Receiver Pair



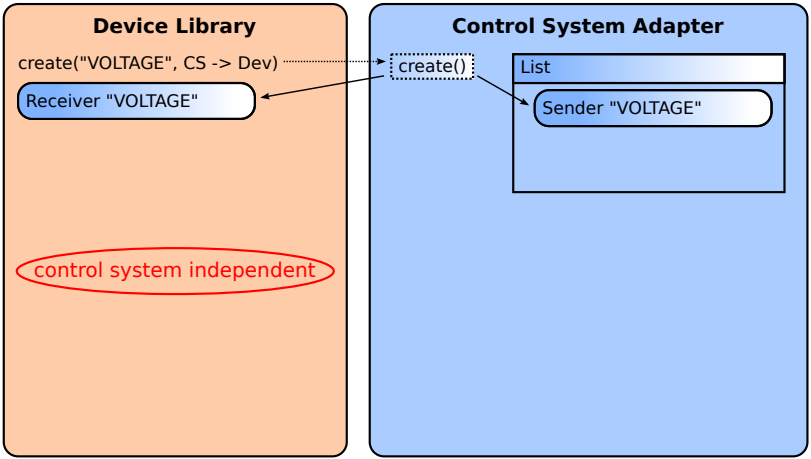
- Lock-free queues
- Pre-allocated buffers for arrays
- Copy references, not buffers

- Buffers are std::vector
- Swap content to avoid additional copies

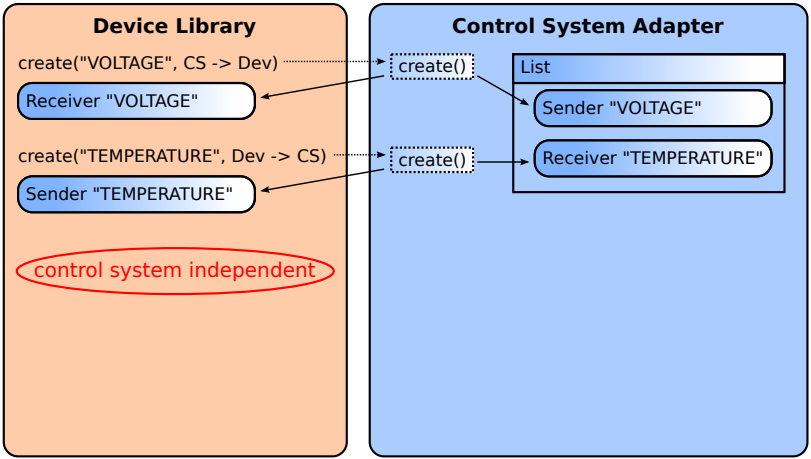
A Device Using The Control System Adapter



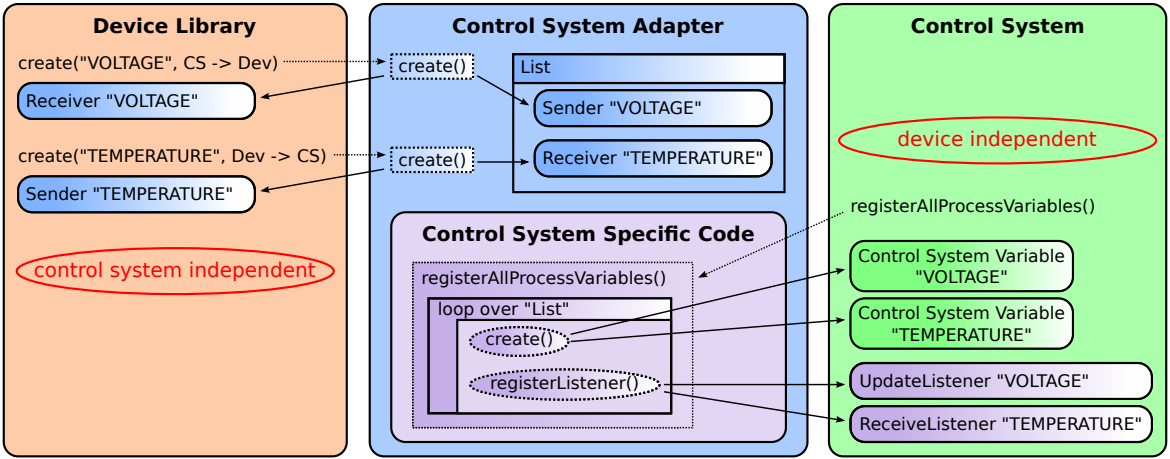
Registering Process Variables



Registering Process Variables



Registering Process Variables



Adapter for process variables

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

Design Goals

- Control system independent process variables ✓
- Thread safety ✓
- Real time capability ✓
- Minimise copying ✓
- Minimise device-dependent code on control system side (✓)

Next Steps: Extending The Adapter

Missing: Access to control system features

- Limits
- History
- Engineering units
- ...

Missing: More sophisticated data objects

- Currently only scalars and arrays

⇒ Talk by Martin Hierholzer

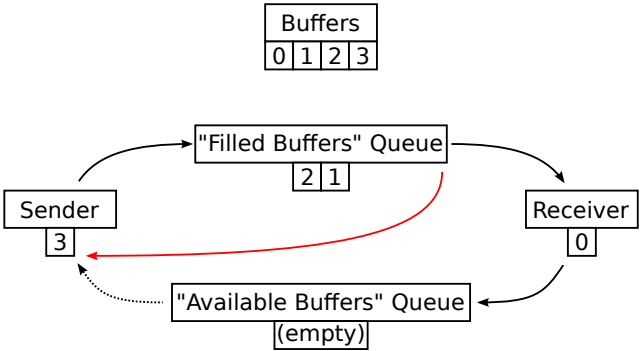
Software Repositories All software is published under the GNU General Public License.

- MTCA4U Control System Adapter: https://svnsrv.desy.de/public/mtca4u/ControlSystemTools/
- EPICS 3 extension: http://oss.aquenos.com/svnroot/epics-mtca4u/
- DOOCS extension: https://svnsrv.desy.de/desy/mtca4u_applications/DOOCS_Adapter/

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)