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
A. Piotrowski, FastLogic Sp. z o.o., Łódź, Poland
J. Wychowaniak, Łódź University of Technology, Łódź, Poland

24th November 2015

SRF Controls and CW Operation Mini-Workshop, Dresden-Rossendorf
Sometimes You Need An Adapter

Example:
- LLRF Server
- Target Control Systems
- DOOCS at FLASH, XFEL/DESY
- EPICS 3 at FLUTE/KIT
- WinCC/OPC-UA at ELBE/HZDR

Martin Killenberg (DESY)
Sometimes You Need An Adapter

Device Adapter

Control System

EXAMPLE:

LLRF Server

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
Sometimes You Need An Adapter

Device

Adapter

Control System

EXAMPLE:

LLRF Server

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

Martin Killenberg (DESY)

The MTCA4U Control System Adapter
Sometimes You Need An Adapter

**EXAMPLE:** LLRF Server
- $O(400)$ 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
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)
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
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

Adapter Variable Pair "VOLTAGE"

Receiver  ➔  Sender
A Device Using The Control System Adapter

Device Library

Adapter Variable Pair "VOLTAGE"
Receiver
Sender

Use "VOLTAGE"
Update "TEMPERATURE"

Adapter Variable Pair "TEMPERATURE"
Sender
Receiver
A Device Using The Control System Adapter

Device Library

Device Thread

Adapter Variable Pair "VOLTAGE"

Receiver

Sender

Use "VOLTAGE"
Update "TEMPERATURE"

Adapter Variable Pair "TEMPERATURE"

Sender

Receiver

Control System

Control System Thread

Control System Variable "VOLTAGE"

Update

Control System Variable "TEMPERATURE"

Update
Implementation Of The Sender/Receiver Pair

- Sender
- Receiver
- Queue
- Lock-free queue

Pre-allocated buffers for arrays
Copy references, not buffers
Implementation Of The Sender/Receiver Pair

| Buffers | 0 | 1 | 2 | 3 |

- 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

<table>
<thead>
<tr>
<th>Buffers</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
</table>

Sender: 1

Receiver: 0

"Filled Buffers" Queue
(empty)

"Available Buffers" Queue
2 3
Implementation Of The Sender/Receiver Pair

<table>
<thead>
<tr>
<th>Buffers</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
</table>

Sender

Receiver

Send

"Filled Buffers" Queue
(empty)

"Available Buffers" Queue

- Lock-free queues
- Pre-allocated buffers for arrays
- Copy references, not buffers
Implementation Of The Sender/Receiver Pair

Buffers
0 1 2 3

"Filled Buffers" Queue
1

"Available Buffers" Queue
3

Sender
2

Receiver
0

- Lock-free queues
- Pre-allocated buffers for arrays
- Copy references, not buffers
Implementation Of The Sender/Receiver Pair

| Buffers | 0 | 1 | 2 | 3 |

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

Sender Receiver
Buffers

"Filled Buffers" Queue

"Available Buffers" Queue

receive
Implementation Of The Sender/Receiver Pair

<table>
<thead>
<tr>
<th>Buffers</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sender</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Receiver</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- "Filled Buffers" Queue
- "Available Buffers" Queue
- (empty)
- Lock-free queues
- Pre-allocated buffers for arrays
- Copy references, not buffers
A Device Using The Control System Adapter

---

**Device Library**

- Adapter Variable Pair "VOLTAGE"
  - Receiver
  - Sender
  - Use "VOLTAGE"
  - Update "TEMPERATURE"

**Device Thread**

**Control System**

- Control System Variable "VOLTAGE"
  - Update

**Communication Thread**

- Adapter Variable Pair "TEMPERATURE"
  - Sender
  - Receiver
  - Update

- Control System Variable "TEMPERATURE"
Registering Process Variables

Device Library

create("VOLTAGE", CS -> Dev)

Receiver "VOLTAGE"

closest system independent

Control System Adapter

create()

List

Sender "VOLTAGE"
Registering Process Variables

**Device Library**

- `create("VOLTAGE", CS -> Dev)`
- `create("TEMPERATURE", Dev -> CS)`
- **Sender "TEMPERATURE"**
- **Receiver "VOLTAGE"**

**Control System Adapter**

- **List**
  - **Sender "VOLTAGE"**
  - **Receiver "TEMPERATURE"**

---
control system independent

Martin Killenberg (DESY)
Registering Process Variables

Device Library
- create("VOLTAGE", CS -> Dev)
  Receiver "VOLTAGE"
- create("TEMPERATURE", Dev -> CS)
  Sender "TEMPERATURE"

Control System Specific Code
- registerAllProcessVariables()
- loop over "List"
  - create()
  - registerListener()

Control System Adapter
- create()
  - List
    - Sender "VOLTAGE"
    - Receiver "TEMPERATURE"

Control System
- device independent
- registerAllProcessVariables()
- Control System Variable "VOLTAGE"
- Control System Variable "TEMPERATURE"
- UpdateListener "VOLTAGE"
- ReceiveListener "TEMPERATURE"
Status

Adapter for process variables

- Generic part
- Control system specific part
  - Implementations for DOOCS and EPICS 3

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
- Handle alarms
- Synchronise with timing (currently done by DOOCS)
- ...

**Missing: More sophisticated data objects**
- Currently only scalars and arrays

**Tasks for this workshop**
1. Identify which features are needed/wanted for the LLRF server!
2. Discussion: How to implement it in the adapter and for various control systems?
   *Implementation are very different in the various control systems!*
MTCA4U Control System Adapter

- Adapter to use device logic with different control systems
- Implementations for DOOCS and EPICS 3 exist
- Planned: support for OPC-UA
- How to use control system features through the adapter?

Software Repositories

- 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)