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

20th October 2015
ICALEPCS 2015, Melbourne, Australia
Sometimes You Need An Adapter
Sometimes You Need An Adapter

Martin Killenberg (DESY)

The MTCA4U Control System Adapter
Sometimes You Need An Adapter

Device

Adapter

Control System

Martin Killenberg (DESY)

The MTCA4U Control System Adapter
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)
**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"

Sender
Receiver
A Device Using The Control System Adapter

Adapter Variable Pair "VOLTAGE"

Sender
Receiver

Use "VOLTAGE"
Update "TEMPERATURE"

Adapter Variable Pair "TEMPERATURE"

Sender
Receiver
A Device Using The Control System Adapter

Use "VOLTAGE"
Update "TEMPERATURE"

Device Thread
Communication Thread
Implementation Of The Sender/Receiver Pair

Sender

Queue

Receiver

- Lock-free queue

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

- Lock-free queue
- Pre-allocated buffers for arrays
Implementation Of The Sender/Receiver Pair

<table>
<thead>
<tr>
<th>Buffers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

- "Filled Buffers" Queue
  - (empty)

- "Available Buffers" Queue
  - 2 3

Sender: 1
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

"Filled Buffers" Queue
(empty)

"Available Buffers" Queue
2 3

Sender
1

Receiver
0

send

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

Martin Killenberg (DESY)
Implementation Of The Sender/Receiver Pair

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

Buffers

```
0 1 2 3
```

"Filled Buffers" Queue

Sender 1

"Available Buffers" Queue

Receiver 0

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

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

Sender

"Filled Buffers" Queue

1

Receiver

"Available Buffers" Queue

3

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

Buffers

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

"Filled Buffers" Queue
(empty)

Sender

Receiver

"Available Buffers" Queue

- 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

Control System Variable "VOLTAGE"

Update

Use "VOLTAGE"
Update "TEMPERATURE"

Communication Thread

Device Thread

Adapter Variable Pair "TEMPERATURE"

Sender

Receiver

Control System Variable "TEMPERATURE"

Update

Control System Device Library

Control System Variable
"VOLTAGE"

Update

Control System Variable
"TEMPERATURE"
Registering Process Variables

Device Library

create("VOLTAGE", CS -> Dev)

Receiver "VOLTAGE"

close 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**
- `create()`
  - List
    - Sender "VOLTAGE"
    - Receiver "TEMPERATURE"

control system independent
Registering Process Variables

**Device Library**
- `create("VOLTAGE", CS -> Dev)`
- `Receiver "VOLTAGE"

**Control System Specific Code**
- `create("TEMPERATURE", Dev -> CS)`
- `Sender "TEMPERATURE"

**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"

**Control System**
- `control system independent`
- `loop over "List"`
  - `create()`
  - `registerListener()`
Status

Adapter for process variables
- Generic part
- Control system specific part
  - Implementations for DOOCS and EPICS

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

- Limits
- History
- Engineering units

Implementations are very different in the various control systems!

- Discussions how to put this into the adapter
MTCA4U Control System Adapter

- Adapter to use device logic with different control systems
- Implementations for DOOCS and EPICS exist
- Planned: support for OPC-UA

Software Repositories

- EPICS extension: http://oss.aquenos.com/svnroot/epics-mtca4u/
- DOOCS extension: https://svnsrv.desy.de/desy/mtca4u_applications/D0OCS_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)