

Experiences with the Event-Driven REFLEX Operating System

Karsten Walther, Reinhard Hemmerling, Jörg Nolte
Chair Distributed Systems / Operating Systems
BTU Cottbus

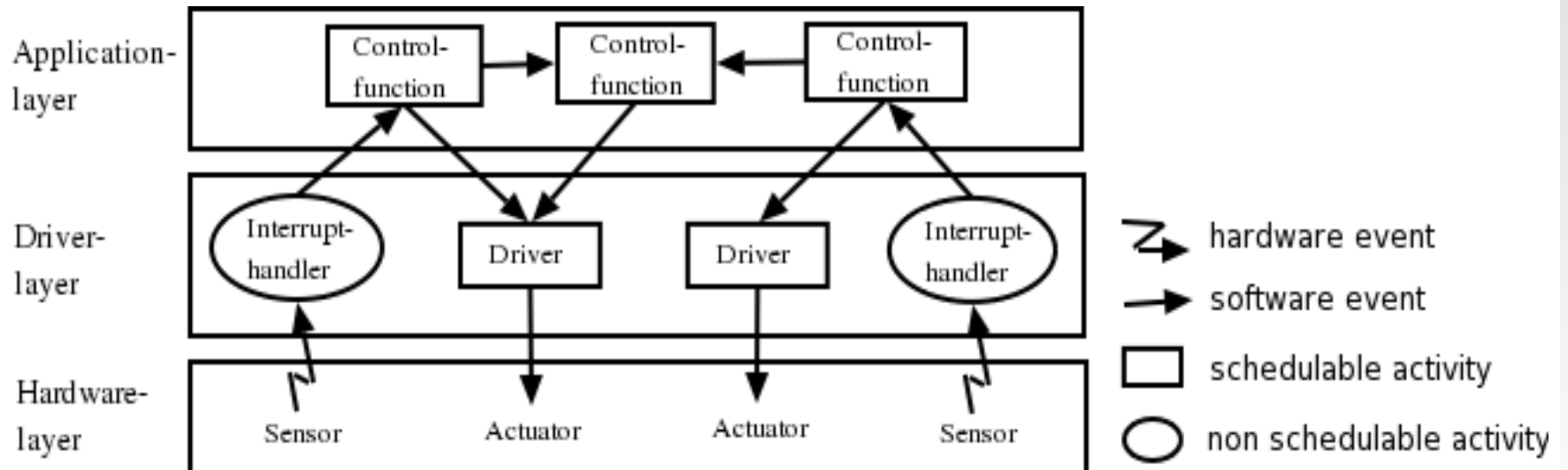
Overview

- Introduction to Reflex
- The Example Application
- Writing an Application
- Conclusion

REFLEX

- **Real-time Event FLOW EXecutive**
- Event flow based programming model
- Schedulable passive objects
 - according to an Earliest-Deadline-First strategy
- Implemented in C++
- Intended use in deeply embedded environments
 - (Real-Time) Controlling applications
 - Sensor networks

REFLEX-Scheme



REFLEX

- all flows of execution are started by an interrupt
- when data is written on an input, the corresponding activity is reported to the scheduler
- Scheduler activates the Activity then according ist EDF strategy
- Activities can be connected if datatype matches, so a system build out of highly reusable software fragments are possible

Activities

```
class Activity:public ListElem {
public:
    enum Status
    {
        RUNNING ,    //at most 1 Activity can be in this state
        INTERRUPTED , //not running, because it is interrupted by an interrupt
        SCHEDULED , //ready to run
        WAITING ,    //not used until now or in stand by
        DEACTIVATED //totally inactive, recognizes no events
    };

    Activity();

    virtual void run()=0;

    ...
};
```

Activities

- are the place for user code
- run-method is called by the scheduler

Sinks

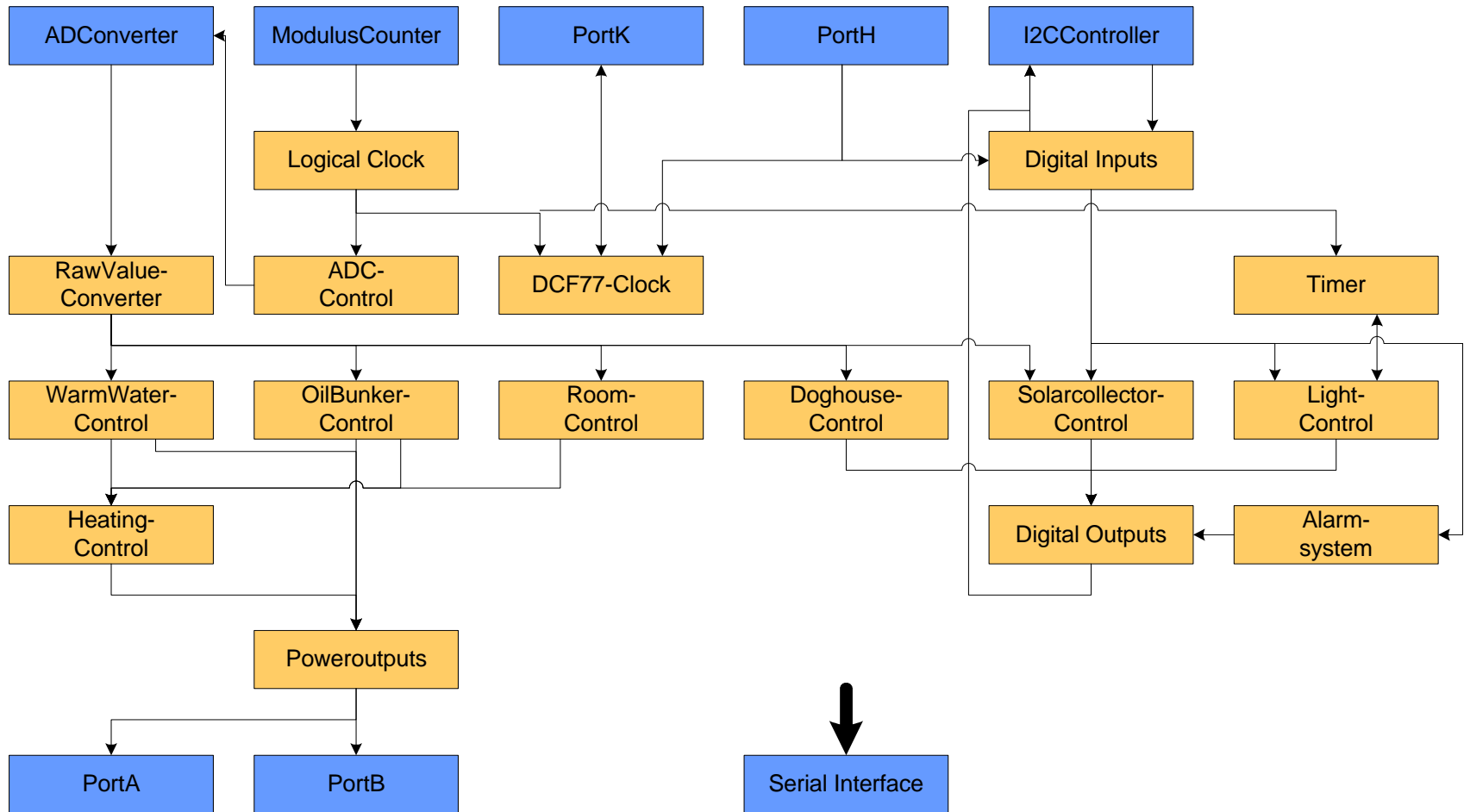
```
template<typename T>
class Sink {
public:
    virtual void assign(T value) = 0;
};
```

```
template<typename T>
class ExtendedSink : public Sink<T>{
public:
    virtual void set(T value) = 0;
    virtual void unset(T value) = 0;
};
```

Sinks

- is the abstraction to which flowdata is written
- key component for decoupling of activities
- has value copy semantics

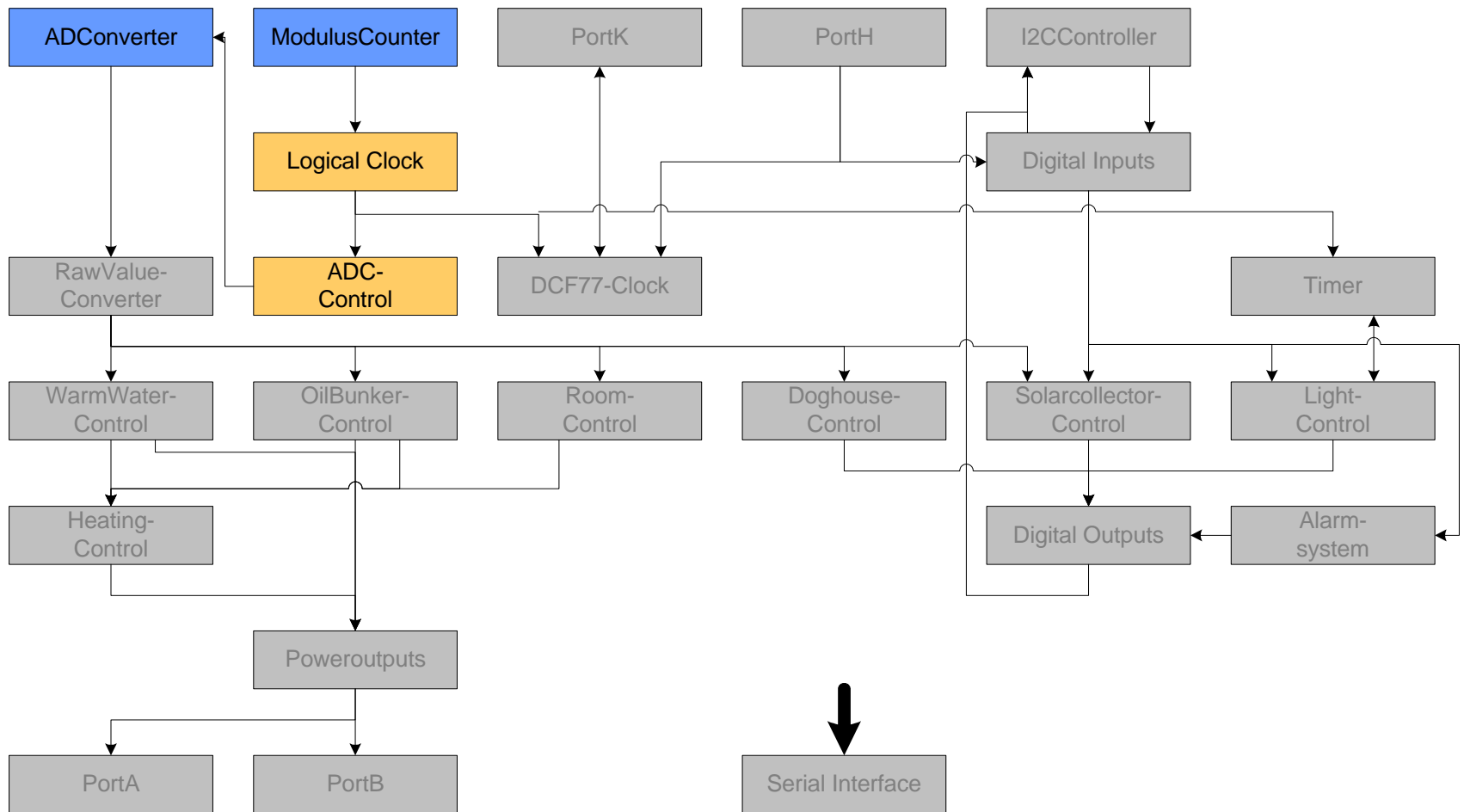
House Control System



House Control System

- Complex system, which has 2 logical parts, heating control and alarm system
- every rectangle is an Activity
- blue rectangle are drivers
- Serial interface is stdout

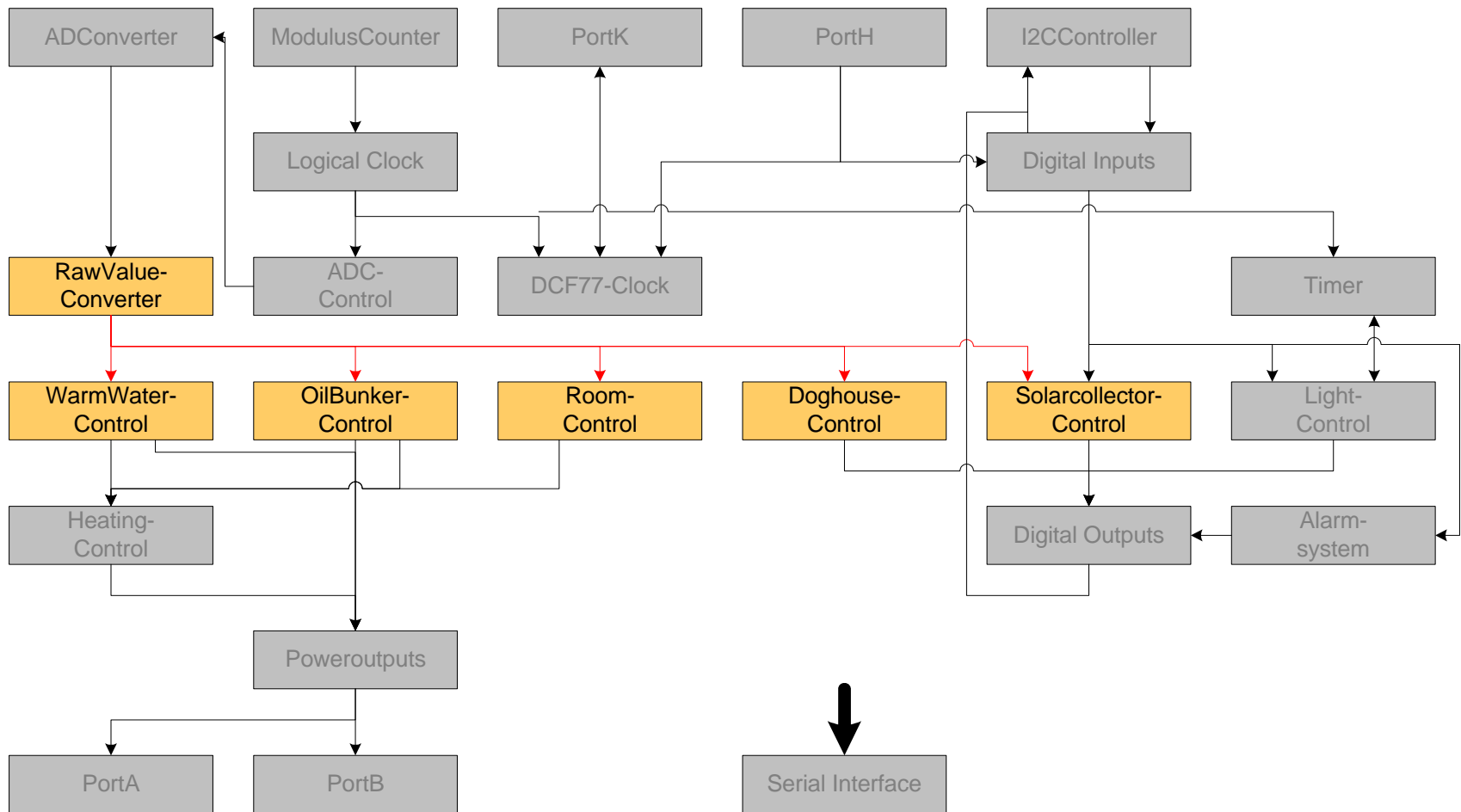
Sampling



Sampling

- most basic function for target applications
- periodically sampling can be implemented by inherently existing logical clock
- logical clock is also need by the scheduler

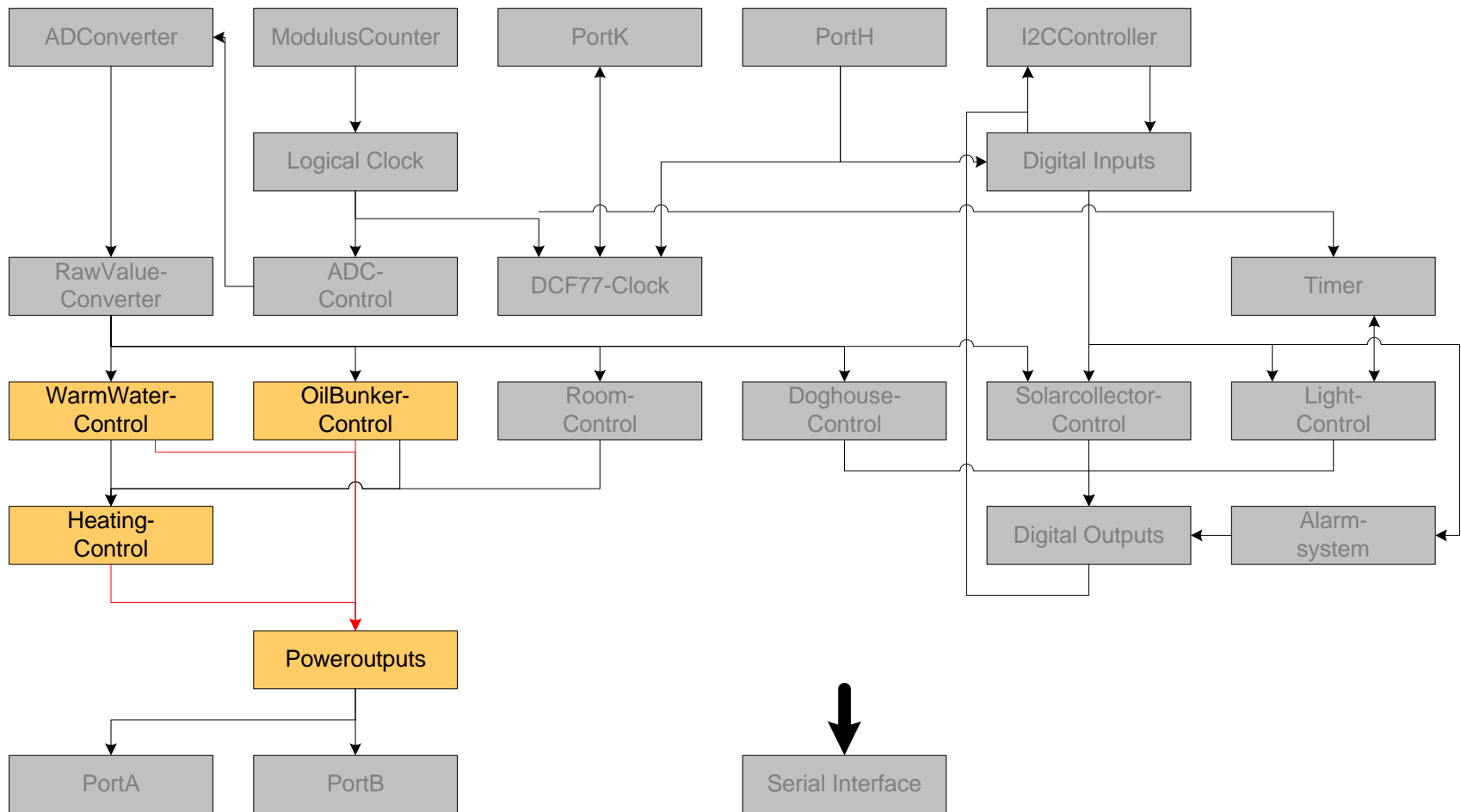
Distribution of Events



Distribution of Events

- Often more than one activity is interested in some data
- Nevertheless mostly the origin should not care about
- so a wire abstraction is needed
- the wire itself is a sink

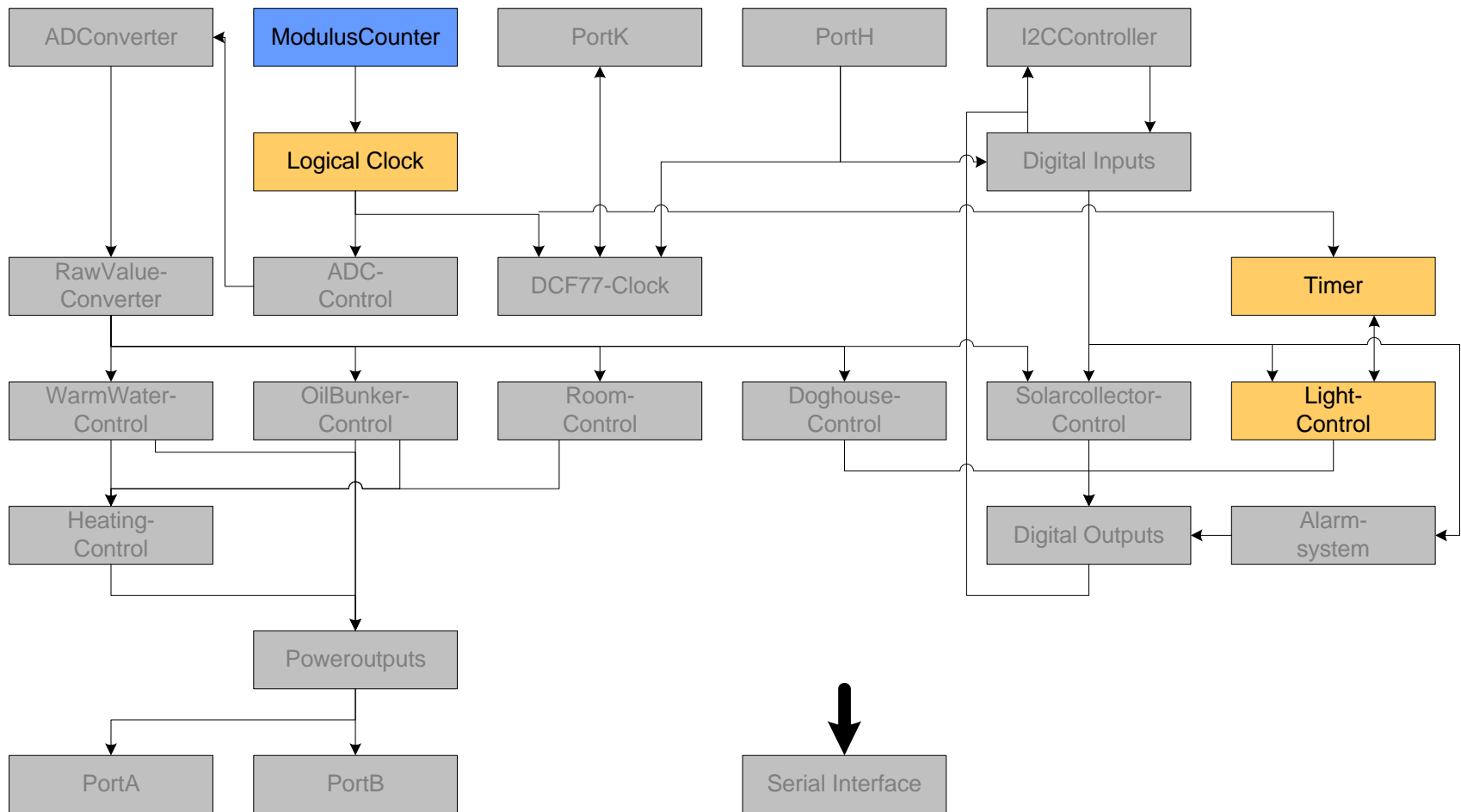
Combined Inputs



Combined Inputs

- needed for example for port sharing
- distinct origins writing to distinct bits
- for convenience extended sinks are used

Using Timers



Using Timers

- often used for timeout
- timer can be started and stopped
- timer fires by writing data to a sink

Multiple Inputs

- when there are no distinguishable inputs, run method has to check which input is written
- solution is declaring pseudo-activities for distinct inputs, which calling specific function
- deadlines are now input-specific

Conclusion

- REFLEX can be used in practice
 - Usability
 - Memory Consumption
- Concept of decoupled activities leads to high reusability

Outlook

- Analysis of real time capabilities
- Implementation of distributed systems with Reflex as base
- SDL toolchain for REFLEX (IHP)

Questions ?

