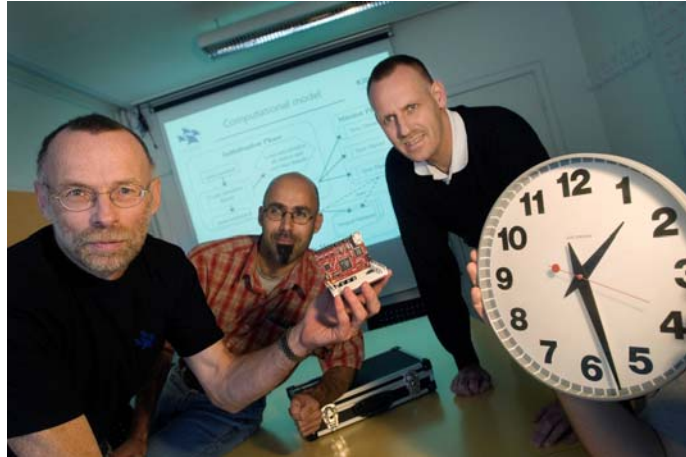


Ravenscar-Java Development



Summit'06
27 March 2006

Hans Søndergaard, Martin Astradsson, Bent Thomsen
Vitus Bering, Horsens FOSS, Hillerød CISS, Aalborg

1

Ravenscar-Java Development



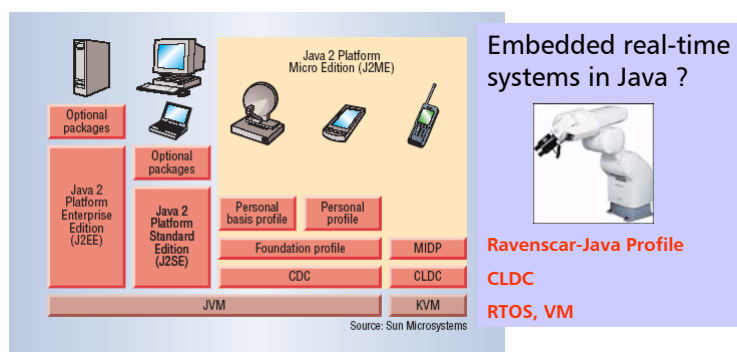
- The purpose of the project
 - implement the *Ravenscar-Java Profile* on the aJ-100 processor
 - find out to what extent *Real-Time UML* with advantage can be applied when designing embedded real-time systems
 - investigate through development of an *industrial case* how useful the Ravenscar-Java Profile is for design and development of industrial embedded systems with real-time requirements
 - compare the *Ravenscar-Java* solution with a *C++* solution.

2

Outline

- Java in embedded systems
- The Ravenscar-Java Profile
- Controller: aJ-100 Java processor
- Implementation of the Ravenscar-Java Profile
- Conclusion
- What next

Java overview



MIDP = Mobile Information Device Profile

- Mobile phone subscribers: $> 2 \cdot 10^9$ subscribers, 18 Sept. 2005
- 700 millions Java-enabled mobile phones, 2004
- At least 80 percent of mobile phones will support Java by 2006

From Java (1995) to ...

- Java
 - + object-oriented
 - + strongly typed
 - + multithreaded
 - + automatic garbage collection
 - + JVM: "write once, run anywhere"
 - + shorter development time in Java
- Java for real-time systems
 - threads and their scheduling too weak
 - automatic garbage collection
 - normal JVM: not usable for real-time systems

5

... to Real-Time Specification for Java, RTSJ, (2000) ...

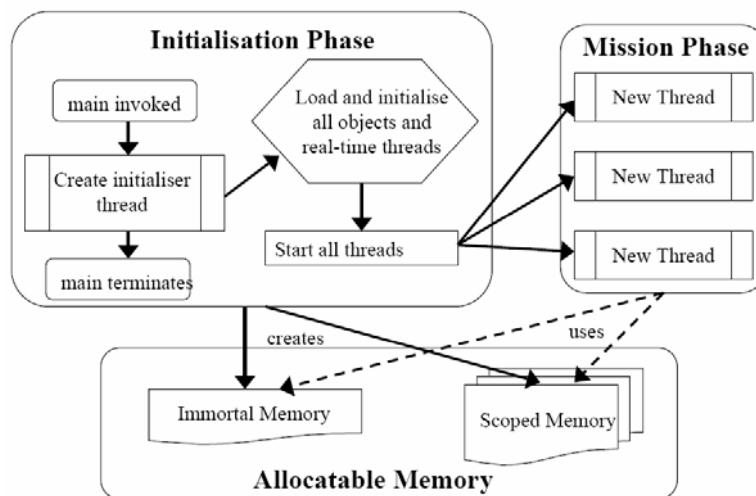
- Main enhanced areas
 - + Thread scheduling and dispatching
 - + Memory management
 - + Asynchronous event handling
 - + Physical memory access
- However
 - complex to implement and to use
 - difficult to make static analysis
 - probably best targeted at J2SE

6

... to Ravenscar-Java Profile (2002)

- subset of RTSJ (and a few new classes)
 - simple
 - reliable
 - predictable as to
 - + memory utilization
 - + timing
 - analysable
 - suitable for J2ME

Ravenscar-Java: two execution phases



Ravenscar-Java: characteristics



- No garbage collection
- Scheduling
 - + Fixed priority pre-emptive
- To avoid priority inversion
 - + priority ceiling protocol

9

Ravenscar-Java: programming constructs



- Real-time threads
 - Initializer thread
 - creates and initializes all threads, objects, memory allocation, etc.
 - Periodic threads
- Sporadic events and event handlers
 - only sporadic event with minimum arrival time
 - no aperiodic event handlers
- Memory
 - Immortal memory (can use the heap because of no GC)
 - Scoped memory (don't know how useful it is)
 - Raw memory

10

Controller: aJ-100 Java processor

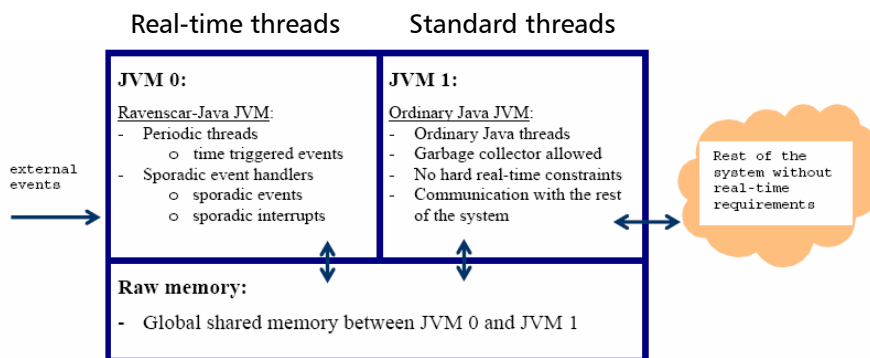


- from aJile Systems
- based on the 32-bit JEM2 Java chip developed by Rockwell-Collins
- uses Java bytecode as its native instruction set
- 100 MHz
- has an embedded real-time multi-threading kernel
 - no extra RTOS software layer
- supports two concurrent JVM units
- has all the common embedded peripherals
 - I/O Ports, Serial Interface, Timers, etc.
- board: JStik from Systronix

11



aJ-100 architecture



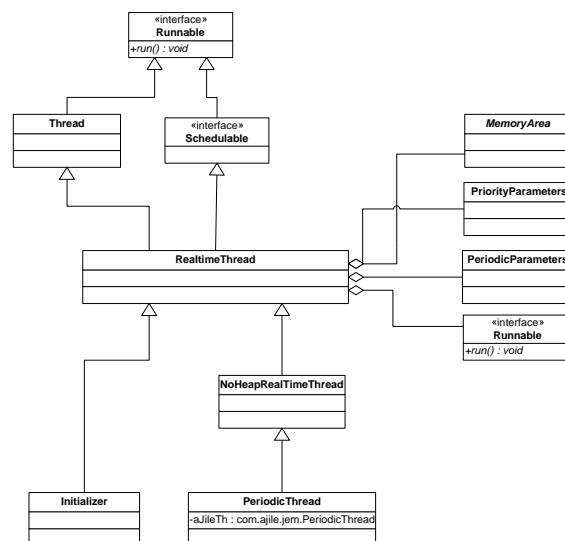
12

The aJile environment

- The runtime system based upon
 - J2ME (Java 2 Platform Micro Edition)
 - CLDC (Connected Limited Device Configuration 1.0)
- An aJile Java API to serve the processor
 - 85 interfaces and classes, e.g.
 - PianoRoll, PeriodicThread
 - rawJEM (low level access to physical memory)
 - GpioPin (controls general purpose IO pins)
- JEM Builder and Charade:
 - tools for static linking and loading, etc.

13

Implementation of the Ravenscar-Java Profile: Real-time threads: class diagram

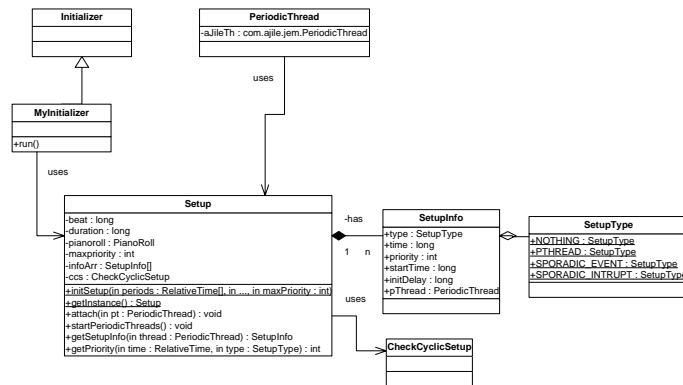


14

Implementation of the Ravenscar-Java Profile: Real-time threads: Rate monotonic setup



$$\text{period}_i < \text{period}_j \Rightarrow \text{priority}_i > \text{priority}_j$$



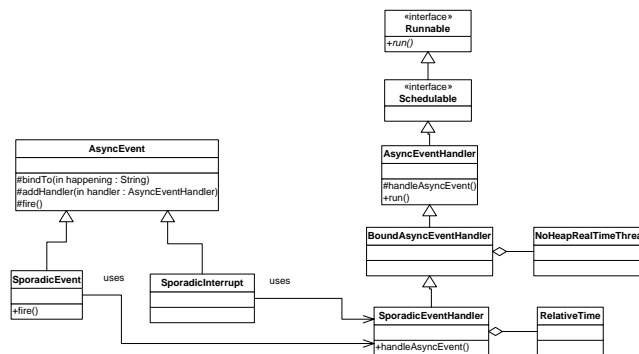
- aJile implements a fixed priority pre-emptive scheduler

15

Implementation of the Ravenscar-Java Profile: Asynchronous events



- only sporadic events
 - software-generated events: class `SporadicEvent`
 - hardware-generated events: class `SporadicInterrupt`



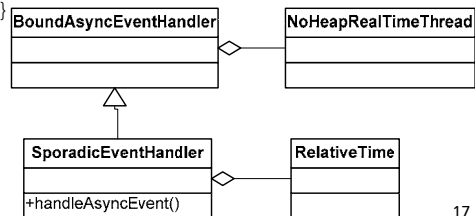
16

Implementation of the Ravenscar-Java Profile: Asynchronous events: NoHeapRealtimeThread



```
private class NoHeapRtThread extends NoHeapRealtimeThread
{
    public NoHeapRtThread (PriorityParameters priority, Runnable logic)
    {
        super (priority, null, null, logic);
    }

    public void run ()
    {
        while (true)
        {
            synchronized (handlerThread)
            {
                try { wait(); }
                catch (InterruptedException e) { }
            }
            logic.run();
        }
    }
}
```



17

Implementation of the Ravenscar-Java Profile: Event handlers



- **BoundAsyncEventHandler:**

```
void handleAsyncEvent ()
{
    if (handlerThread == null)
        super.handleAsyncEvent ();
    else {
        synchronized (handlerThread) {
            handlerThread.notify ();
        }
    }
}
```

- **SporadicEventHandler extends BoundAsyncEventHandler:**

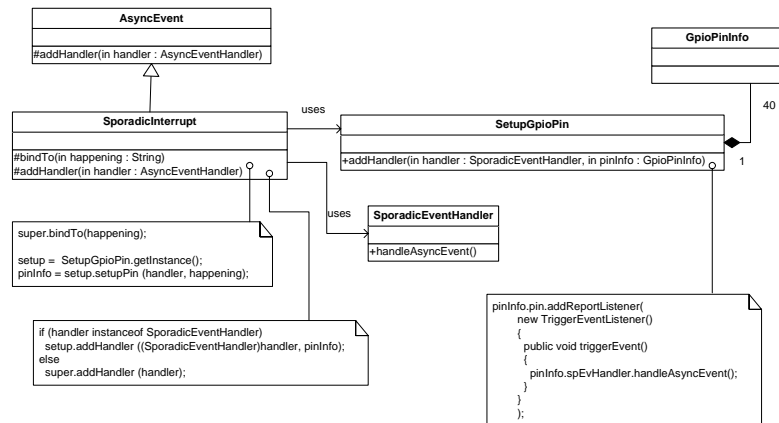
```
private long oldTime, newTime; // using nanoSec
private long minArrivalTime, deltaT;
..
public final void handleAsyncEvent ()
{
    newTime = rawJEM.getTime () * 1000; // micros * 1000 = nanos
    deltaT = newTime - oldTime;
    oldTime = newTime;
    if (deltaT < minArrivalTime)
        ; // ignore
    else
        super.handleAsyncEvent ();
}
```

18

Implementation of the Ravenscar-Java Profile: SporadicInterrupt



- aJ-100 has 5 IO ports. Each having 8 bits.
- Each pin can send a `triggerEvent` when the pin rises, falls, or both.
- Can set the pin to invert the logic: true if the pin is inverting it's logic level
- Can set the pin direction, true = output, false = input



19

Conclusion



- Implemented a clean version of the Ravenscar-Java Profile

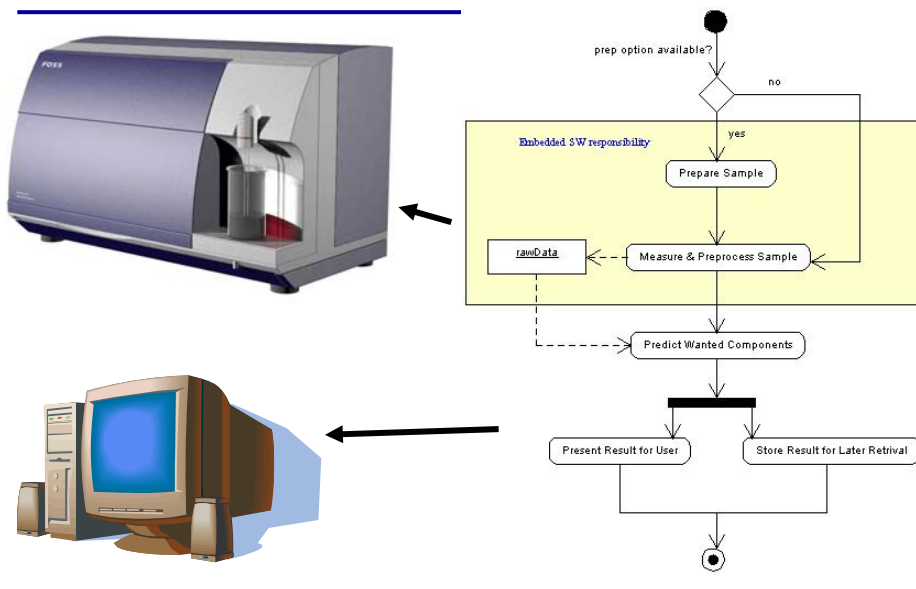
	# classes	# code lines	avg. code lines/class
Ravenscar	35	650	19
Utility	15	900	60
Total	50	1550	31

- Initial experiments suggests that it is an efficient platform for embedded real-time systems
 - changing from thread-to-thread: $< 1 \mu\text{sec}$
 - call of a method: $1.2 \mu\text{sec}$
 - 500 periodic threads, each with a stack size of 1000 bytes
 - execution time nearly the same as JOP (Java Optimized Processor)
 - execution time comparable with C, according to aJile Systems

20

Next step: - a real application from FOSS

RJD 



References

RJD 

- Ravenscar-Java Development, <http://www.cs.aau.dk/ravenscar/>
- Peter Puschner, and Andy Wellings: A Profile for High-Integrity Real-Time Java Programs. 4th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC), 2001.
- Jagun Kwon, Andy Wellings, and Steve King: Ravenscar-Java: A High Integrity Profile for Real-Time Java. Department of Computer Science, University of York, UK. York Technical Report YCS 342. May 2002.
- Andy Wellings: Concurrent and Real-Time Programming in Java. Wiley. 2004. ISBN: 0-470-84437-X.
- The Real-Time Specification for Java (RTSJ), <https://rtsj.dev.java.net/>, draft from July 31, 2002.

Prices

-
- JStik board, - including software: 500 \$
 - JStik only: 1 350\$
1000 200 \$
 - aJ-100 1 30 Euro

Questions

