

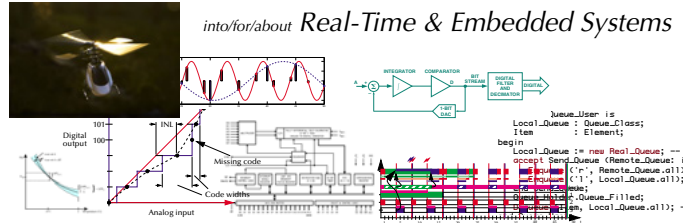
RES

Real-Time & Embedded Systems 2009

Uwe R. Zimmer – The Australian National University

what is offered here?

Overview, Perspectives, Paths, Methods, and some Theory



who could be interested in this?

anybody who ...

... would like to see immediate real-world involvement in his/her work

... would like to learn how to create predictability and fault-tolerant complex systems

... would like to know more about the usage of 95% of all μ processors

who are these people? – introduction



The course will be given by

Uwe R. Zimmer

and

Pat Bernardi



how will this all be done?

LECTURES:

- 2 lectures à 1.5h per week ... all the nice stuff and theory Monday & Tuesday, 9:00-10:30; in MCC T4, PSYC G8 resp.

LABORATORIES:

- 2 hours per week ... all the rough stuff and practice Monday 11:00-13:00 or Tuesday 13:00-15:00 – in CSIT N114, N113 resp. laboratory-enrolment: <https://cs.anu.edu.au/streams/>

RESOURCES:

- introduced in the lectures and collected on the course page: <http://cs.anu.edu.au/student/comp4330/> ... as well as schedules, slides, sources, etc. pp ... keep an eye on this page!

ASSESSMENT:

- exam at the end of the course (70%) plus laboratories performance (30%)

Topics in this course

- | | |
|---------------------------------------|----------------------------------|
| 1. Introduction & real-time languages | 6. Synchronisation |
| 2. Physical coupling | 7. Scheduling |
| 3. Interfaces | 8. Resource control |
| 4. Time & embodiment | 9. Reliability & fault-tolerance |
| 5. Asynchronism | |

Central textbook

Supporting literature

[Burns01]

Alan Burns and Andy Wellings
Real-Time Systems and Programming Languages
Addison Wesley, third edition, 2001

[Ari90]

M. Ben-Ari
Principles of Concurrent and Distributed Programming
Prentice Hall, 1990

[Cohen96]

Norman H. Cohen
Ada as a second language
McGraw-Hill series in computer science, 2nd edition

[Ada95RM] (on-line version available)

Ada Working Group
Ada 95 Reference Manual
– Language and Standard Libraries
ISO/IEC 8652:1995(E) with COR. 1:2000,
June 2001

many more references and links are available on the course page

Table of Contents

<p>1. Introduction & Real-Time Languages</p> <p>1.1. Features (and non-features) of a real-time system</p> <p>1.2. Components of a real-time system</p> <p>1.3. Real-time languages criteria</p> <p>1.4. Examples of actual real-time languages:</p> <ul style="list-style-type: none"> • Ada95, Esterel, Pearl, Real-time JAVA, ROSK <p>2. Physical coupling</p> <p>2.1. Physical phenomena</p> <p>2.2. Measuring temperature</p> <ul style="list-style-type: none"> • Thermocouples, thermocouples, thermoresistors, thermistors, noise temperature measurement and others <p>2.3. Measuring range and relative speed</p> <ul style="list-style-type: none"> • Triangulation, time of flight, intensity • Doppler methods, interferometry <p>2.4. Examples:</p> <ul style="list-style-type: none"> • Time of flight ultrasound, time-of-flight laser, Doppler current profiler <p>3. Converters & Interfaces</p> <p>3.1. Analogue signal chain in a digital system:</p> <ul style="list-style-type: none"> • Sampling data \Rightarrow aliasing \Rightarrow Nyquist's criterion \Rightarrow oversampling • Quantization \Rightarrow SNR, rms noise voltage, SNR, ENOB \Rightarrow Missing codes, INL, INE <p>3.2. A/D converters: flash, pipelined-flash, SAR, 2ⁿ-bit order Σ-A</p> <p>3.3. Examples:</p> <ul style="list-style-type: none"> • Fast and simple A/D converter example 	<ul style="list-style-type: none"> • Multi-channel A/D data logging interface example • Single 8-bit processor example • Complex 24-bit processor example: TPL-programming, atomic states, per-pixel scheduling, max. latency analysis, NEXUS debugging port <p>3.4. General device handling / sampling control / language requirements</p> <p>4. Time & Space</p> <p>4.1. What is time? / What is embodiment?</p> <ul style="list-style-type: none"> • Approaches by different facilities to understand the basis for this course <p>4.2. Interfacing with time</p> <ul style="list-style-type: none"> • Formulating local time-dependent constraints – Access time, delay processes, detect timeouts (in different languages) <p>4.3. Specifying timing requirements</p> <ul style="list-style-type: none"> • Formulating global timing constraints – Understanding time-scope parameters (and expressing them in different languages) <p>4.4. Satisfying timing requirements</p> <ul style="list-style-type: none"> • Real-time logic and complex systems approach <p>5. Asynchronism</p> <p>5.1. Interrupts / Signals</p> <ul style="list-style-type: none"> • Device / system / language / operating-system level interrupt control • Characteristics of interrupts and signals <p>5.2. Exceptions</p> <ul style="list-style-type: none"> • Exception classes / granularity / parametrization / propagation – Resumption and termination, specific language issues 	<p>5.3. Atomic Actions</p> <ul style="list-style-type: none"> • Definition / requirements / failure cases / implementation / error recovery <p>5.4. Asynchronous transfer of control / Interrupts in context</p> <ul style="list-style-type: none"> • Interrupts and ATC in real-time Java and Ada95 <p>6. Synchronization</p> <p>6.1. Shared memory based synchronization</p> <ul style="list-style-type: none"> • Flags, condition variables, semaphores, conditional critical regions, monitors, protected objects, • Guard evaluation times, nested monitor calls, deadlocks, simultaneous reading, queue management, • Synchronization and object orientation, blocking operations and re-queuing <p>6.2. Message based synchronization</p> <ul style="list-style-type: none"> • Synchronization models, addressing modes, message structures • Selective accepts, selective calls • Indeterminism in message based synchronization <p>7. Scheduling</p> <p>7.1. Basic real-time scheduling</p> <ul style="list-style-type: none"> • Fixed Priority Scheduling (FPS) with Rate Monotonic (RM/PC) Deadline Monotonic (DM/PC) • Earliest Deadline First (EDF) <p>7.2. Real-world extensions</p> <ul style="list-style-type: none"> • Aperiodic, sporadic, soft real-time tasks – Deadlines shorter than period - Co-scheduling and deferred pre-emption scheduling 	<ul style="list-style-type: none"> • Fault tolerance in terms of exception handling considerations – Synchronized tasks (priority inheritance, priority ceiling protocols) <p>7.3. Language support</p> <ul style="list-style-type: none"> • Ada95, ROSK \Rightarrow static, off-line analysis mostly – RT-Java \Rightarrow on-line, dynamic scheduling <p>8. Resource control</p> <p>8.1. Resource synchronization primitives</p> <ul style="list-style-type: none"> • Evaluation criteria for resource synchronization methods • Atomicity, liveliness, and double interaction <p>8.2. Resource reclaiming schemes</p> <ul style="list-style-type: none"> • Basic reclaiming, early start, and restriction vector algorithms • Resource reclaiming with task migration <p>8.3. Real-time resource control</p> <ul style="list-style-type: none"> • Policy and run-time issues to be considered <p>9. Reliability</p> <p>9.1. Terminology</p> <ul style="list-style-type: none"> • Faults, Errors, Failures – Reliability <p>9.2. Faults</p> <ul style="list-style-type: none"> • Fault avoidance, removal, prevention \Rightarrow Fault tolerance <p>9.3. Redundancy</p> <ul style="list-style-type: none"> • Static (EMR, NMR) and dynamic redundancy • Non-linear programming, and dynamic redundancy in software design <p>9.4. Reduce & Formalise</p> <ul style="list-style-type: none"> • Advers: Reverse profile • Real-time Logic
---	---	--	--