

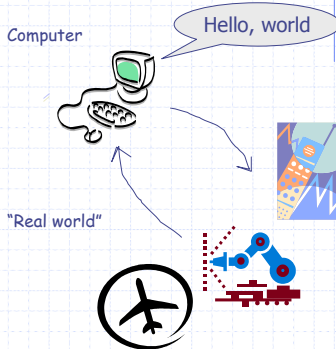
Realtime Systems

John Reekie
University of Technology, Sydney

Terms of Use: Creative Commons Attribution-ShareAlike 2.0
<http://creativecommons.org/licenses/by-sa/2.0/>

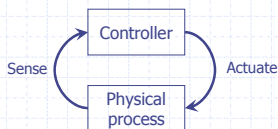
What is a realtime system?

◆ Realtime systems are systems that interact with the "real world."



In this lecture we will be focusing on things that are of architectural significance.

Two major types



A control system controls a physical process.



[Picture of CD]



A signal processing system processes and produces signals in time

Control systems

- ◆ Washing machine
- ◆ Automotive engine and brake control
- ◆ Robotics
- ◆ Oil refinery



Signal processing systems

- ◆ Digital recording studio
- ◆ Satellite communication
- ◆ Video compression
- ◆ Image recognition
- ◆ Film animation

Quality attributes

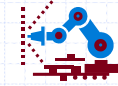
- ◆ Performance
 - *Predictable* time response is paramount
 - Additional CPU speed does not necessarily result in higher performance
- ◆ Reliability
 - Real-time systems are unforgiving
 - Highly-critical systems are often real-time
- ◆ Security
 - Relevant to networked systems



The real world runs in real time...



Discrete time
Discrete values



Continuous time
Continuous values

...but computers don't ☹

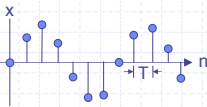
Melting clock from <http://www.designtoscano.com/>

Continuous- vs discrete-time



$$\forall t \in \text{Reals}, x(t) = \sin(2\pi \times 440t)$$

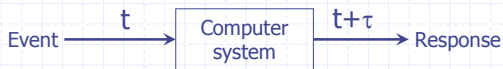
The domain of this signal is Reals. It has a value at every instant in time.



$$\forall n \in \text{Integers}, x(n) = \sin(2\pi n \times 440T)$$

The domain of this signal is Integers. It has a value only at multiples of T . It is an approximation of the continuous-time signal.

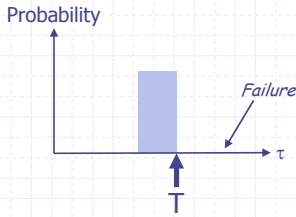
The real world responds instantaneously...



...but computers don't ☹

Unlike a physical system, the response of a computer system is always delayed by some non-zero amount of time (τ).

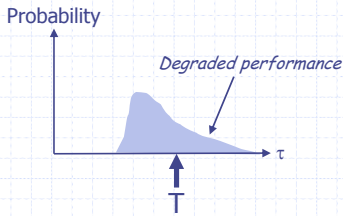
Time waits for no man[†]



A hard real-time system must respond within a specified time T . If it does not, the system has **failed**.

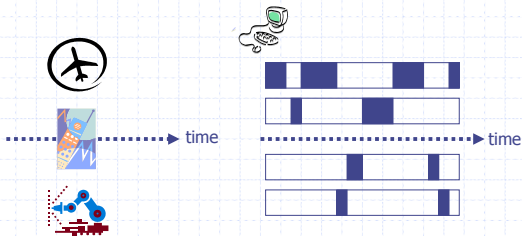
[†]Or woman

Well, not much anyway...



A soft real-time system also has a target response time T . If the response time is not met, it suffers degraded performance.

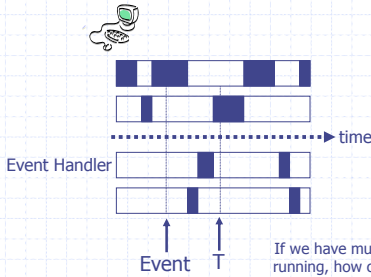
Why is this hard for computers?



Multiple activities are simultaneous

Multiple activities are time-sliced

Getting around to an answer...



If we have multiple tasks (processes) running, how can we be sure that the one responsible for handling an event executes soon enough?

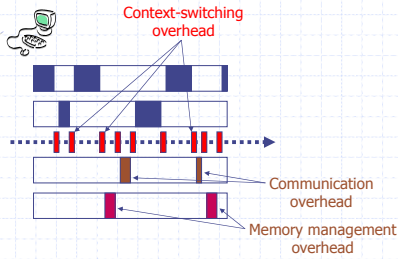
Priorities

- ◆ Higher-priority tasks can pre-empt lower-priority tasks
- ◆ Various algorithms for determining task priorities (rate-monotonic)
- ◆ Not always able to guarantee predictable execution
- ◆ Subject to well-known issues (priority inversion)
- ◆ Supported by any off-the-shelf real-time operating system (RTOS)

Time-triggered scheduling

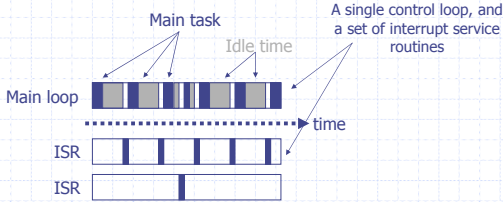
- ◆ Recent advancement in embedded system scheduling
- ◆ All tasks execute on fixed multiples of a clock
- ◆ All tasks execute in small increments, with a known worst-case execution time (WCET)
- ◆ Guarantees predictability
- ◆ Can be run on standard RTOSs, as well as on dedicated platforms

Operating systems are burdensome



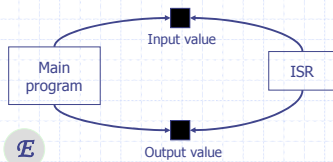
There are numerous overheads associated with an operating system. RTOS's attempt to minimize or control these overheads.

Down to bare metal



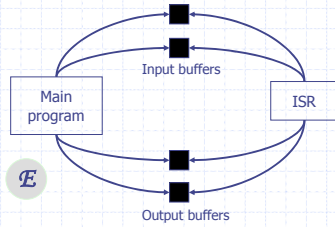
If the operating system gets in the way... get rid of it

A simple realtime program



The ISR and the main program communicate via variables. The main program must read and process each value before the next time the ISR produces a new value.

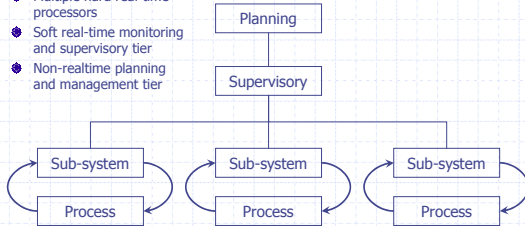
Double-buffering



The ISR and the main program communicate via pairs of buffers. While the ISR is filling one buffer, the main program is reading the other and filling one of the output buffers.

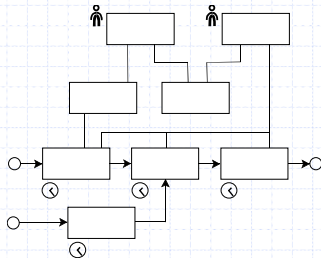
Modern embedded systems

- Multiple hard real-time processors
- Soft real-time monitoring and supervisory tier
- Non-realtime planning and management tier



The view of an embedded system as a single, isolated processor, is hopelessly outdated.

A common pattern



This structure often shows up in systems with real-time components.

Further readings

◆ Bass, Clements, Kazman, *Software Architecture in Practice* – case study of an Air Traffic Control system, example of a hard realtime system.

- Availability and Performance key quality attributes
- Safety critical
- Usability is also important – needs to support the complex distributed work of the air traffic controllers
