1 sec.

• by a radio receiver for UTC or IAT (available in some countries)

1/86400 of a mean solar day … or … 1/31566925.9747 of the tropical year for 1900 (Ephemeris Time defined 1955) … or … ... radiation corresponding to the transition between two hyperfine levels of the ground state of the caesium 133 atom

• by employing time-stamps or sequence numbers of received sensor-readings (Universal Time Coordinated).

It is given and used as external reference as long as an ‘is given and used as external reference as long as an ‘

• by introducing occasional leap ticks) – difference between UTC and IAT is < 0.5s

Using an existing time-frame?

Corrected UT1, UT0, UT0

Generating a time frame?

What is time?

What is time?

What is time?

What is time?

What is time?

What is time?

What is time?

What is time?

What is time?

What is time?

What is time?

What is time?
Real-Time & Embedded Systems

Notions of time and space

Generating a time frame?
- by a timer generating a regular interrupt
- by employing a RTC-module

Using an existing time frame?
- by employing time-stamps or sequence numbers of received sensor-readings
- by a radio receiver for UTC or IAT (available in some countries)

Common programming languages guarantee a ‘resolution’ and ‘accuracy’ of time (in reference to a ‘second’), not its origin or meaning.

What time is it?

What is embodiment?

Working hypothesis:
- Embodied phenomena are those that by their very nature occur in real time and real space.

Phenomenology:
- The phenomena of experience as the central aspects and building blocks of understanding.
  (rather than: “finding the ‘truth’”)

applied to and trying to combine aspects of
- Ontology (about the nature of being and categories of existence)
- Epistemology (the study of knowledge)

Meaningfully embedded systems are part of an ‘ecological niche’ (Rolf Pfeifer):

- The operational environment supports and employs the system
- The embedded system is constructed as a part of the operational environment and according to the task
- The task is meaningful considering the morphology and cognitive ability of the system as well as the response from the environment

☞ i.e. being situated, embodied, and self-sufficient

☞ Real-time and embedded systems are the technical instantiations of embodiment

☞ (Paul Dourish)

☞ Embodied phenomena are those that by their very nature occur in real time and real space.

☞ (Rolf Pfeifer): ‘ecological niche’

☞ Edmund Husserl (1859-1938, Vienna, Halle, Göttingen, Freiburg):
  - founder of the phenomenological tradition
  - as a trial to establish modern science which is firmly grounded on the phenomena of experience (instead of being an abstract mathematical construct).
  - Phenomenology originally as a method to examine the nature of intentionality
  - Cooled the terms
    - Noema: the objects of consciousness
    - Noesis: the mental experiences of those objects
    - Lehenswelt (life-world): the inter-subjective world of everyday-experience
  - Husserl rejected pure abstract and formalized reasoning

☞ Martin Heidegger (1889-1976, Freiburg):
  - Moved phenomenology from a discussion about mental phenomena
    - separated from the physical world (Cartesian dualism)
    - to a discussion about connected physical and mental phenomena.
  - Moved the central questions from epistemology to ontology (‘Being and Time’, 1927).
  - The meaning is not ‘in the head’ but in the world!
  - Cooled terms:
    - Dasein (being-in-the-world): ‘being as inseparable from the world in which it occurs’
    - being as always purposeful and active
    - the world as an unconscious but accessible background
    - Zubehör (ready-to-hand): ‘equipment as a part of actual interaction with the world’
    - Vorhanden (present-at-hand): ‘equipment as a conscious model’

☞ Maurice Merleau-Ponty (1908-1961, Paris (Sorbonne)):
  - ‘The Phenomenology of Perception’ (1945)
  - Embodiment has these implications:
    - a body as a physical entity
    - a body as a set of physical skills and situated responses gained from the physical world
    - a body as a set of ‘cultural skills’ gained from the cultural world in which it is embedded
  - Embodied perception as a bi-directional sensation and a basis for empathy
    (Perception itself does not exist).
  - see also: Phenomenology of Jean-Paul Sartre

Recent works in robotics (and insights about biological sensors) blur the line between action and perception even further.

☞ back to the working hypothesis:
  - Embodied phenomena are those that by their very nature occur in real time and real space.

☞ Embodiment has three implications:

  ⇒ There is no such thing as ‘intelligence’, ‘autonomy’ or any other cognitive process, which is independent of a physical environment.
  ⇒ There is no such thing as a universal system or body (mechanical design, robot, device, …) which is operational in all physical environments.

☞ Implications:

☞ (present-at-hand): ‘equipment as a conscious model’

☞ (ready-to-hand): ‘equipment as a part of actual interaction with the world’

☞ (life-world): the inter-subjective world of everyday-experience

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

☞ (being-in-the-world): ‘being as inseparable from the world in which it occurs’

⇒ There is no such thing as ‘intelligence’, ‘autonomy’ or any other cognitive process, which is independent of a physical environment.

⇒ There is no such thing as a universal system or body (mechanical design, robot, device, …) which is operational in all physical environments.

☞ Implications:

⇒ There is no such thing as ‘intelligence’, ‘autonomy’ or any other cognitive process, which is independent of a physical environment.

⇒ There is no such thing as a universal system or body (mechanical design, robot, device, …) which is operational in all physical environments.
**Real-Time & Embedded Systems**

**Notions of time and space**

**What is embodiment?**

- Embodied skills depend on a tight coupling between perception and action...
- Tight coupling between perception and action means to operate under real-time constraints...

**Real-Time & Embedded Systems**

**Interfacing with time**

**Programming primitive ‘Delay’**

- Alternative for ‘busy-wait’ and interrupts: Suspend a task for a fixed time...
- but all these process or thread delays are precise only in their lower bound!

---

**Real-Time & Embedded Systems**

**Notions of time in Ada**

**Ada.Real-Time package**

```ada
package Ada.Real-Time is
  type Time is private;
  Time.First : constant Time;
  Time.Last : constant Time;
  Time.Unit : constant := 10#1.0#E-9; -- ns
  type Time_Span is private;
  Time_Span.First : constant Time_Span;
  Time_Span.Last : constant Time_Span;
  Time_Span.Unit : constant Time_Span;
  Tick : constant Time_Span; -- actual clock resolution
  function Clock return Time;
  (... operations on and conversions with time and time.span ...)
end Ada.Real-Time;
```

---

**Real-Time & Embedded Systems**

**RT-Java time classes**

Time root class:

```java
public abstract class Hg@ResolutionTime implements java.lang.Comparable
  direct known subclasses: AbsoluteTime, RelativeTime, RationalTime
  • similar to Ada.Real-Time, but no requested accuracy
  • adds the concept of frequency (rational time), but does not guarantee for equidistant instantiations.
  Clock Class:
  public abstract class Clock
  {
    public static Clock getRealtimeClock ()
    {
      local drift:
    }
  }
```

---

**Real-Time & Embedded Systems**

**Interfacing with time**

**Real-time clock interface in POSIX**

```c
#define CLOCK_REALTIME ...; // clockid_t type
struct timespec { time_t tv_sec; /* number of seconds */
long   tv_nsec; /* number of nanoseconds */
};
typedef ... clockid_t;
int clock_gettime (clockid_t clock_id, struct timespec *tp);
```
Interfacing with time

Programming primitive ‘Delay’

- Alternative for ‘busy-wait’ and interrupts: Suspend a task for a fixed time
  - if task is enabled for interrupts, it can also be activated earlier!

  Task body T is
  "Interval: constant Duration: = 5.0; -- sec.
  Start_Time: Time;"
  begin
  loop
  Action;
  delay until Next_Time;
  Next_Time:= Next_Time + Interval;
  end loop;
  action
  end T;

Absolute & relative delays are available in:
- Real-Time Java, Ada95, ... and many other process-oriented languages

Only relative delays are available in some ‘low-level’ environments:
- POSIX: nanoseep (absolute delays need to be constructed employing timers and signals)

Only absolute delays are available in some ‘harder’ real-time environments:
- Occam2, Ada95 (Ravenscar profile), ...

Delay time calculation is not atomic!

Real-Time & Embedded Systems

Relative delay

(task body T)

Relative as absolute delay?

(task body T)

Absolute delay

(task body T)

Zero delay?

(task body T)

Delays

As a third alternative to busy-waiting and infinite blocking, timeouts are implemented in:
- Shared variable communications
  - Semaphore
  - Conditional critical regions
  - Monitors
  - Protected objects
- Message passing between processes
  - Asynchronous and synchronous message transfers
  - Remote procedure calls
  - Remote objects
- Actions

Timeouts

POSIX:

```c
if (sem_timedwait(&call, &timeout) < 0) {
  if (errno == ETIMEDOUT) {
    /* timeout occurred, try something else */
  } else {
    /* some other error occurred, do something about it */
  }
} else {
  /* semaphore is locked successfully, go ahead */
}```

Suspend current process until the semaphore call is open or the time-span timeout has passed
The deadline is fulfilled and there is a result in any case

-- use result

Continue in time and with the best result possible

Get a first approximation and employ spare time for refinements:

1. Get a first approximation in fixed amount of time and well before the deadline
2. Inspect the deadline and if there is enough spare time:
3. Improve the result and keep a record of improvements (while keeping an eye on the deadline)
   a. If the most precise result is delivered before the deadline occurs: ✔
   b. Else use the closest approximation so far: ✔

   The deadline is fulfilled and there is a result in any case

Timeout on actions

Try delivering, else refine the result further

No-wait on incoming calls

(same for task-entry calls (message passing) and protected object calls (monitors))

task body Controller is
time limit
begin loop select accept Call (T: Temperature) do
  -- action if the temperature was delivered in time
  Current_Temp := T;
  end loop;
end select;

-- measure temperature T
select Controller.Call(T);
else
  -- action if temperature could not be delivered in time
  delay 1.0; -- sec.
end select;

-- normal processing
end loop;
end Controller;

Try calling for 500 ms

No-wait on entry calls

(same for task-entry calls (message passing) and protected object calls (monitors))

task body Sensor is
time limit
begin loop select
  accept Shutdown do
    -- termination actions
    exit;
  end Shutdown;
  or
  accept Call (T: Temperature) do
    -- action if temperature can not yet be delivered, e.g. further refine the measurement
    Current_Temp := T;
    end loop;
  or
  delay 0.5; -- sec.
end select;

-- normal operation
end loop;
end Sensor;

Timeouts on incoming calls

Accept calls for a limited time-span of 1s

Timeouts on entry calls

Accept any number of calls until a closing time 'Deadline' for this entry

Interfacing with time

Common RT-systems concept:

Timeliness is often more important than Precision

1. Get a first approximation in fixed amount of time and well before the deadline
2. Inspect the deadline and if there is enough spare time:
3. Improve the result and keep a record of improvements (while keeping an eye on the deadline)
   a. If the most precise result is delivered before the deadline occurs: ✔
   b. Else use the closest approximation so far: ✔

The deadline is fulfilled and there is a result in any case
### Temporal Scopes

**Common attributes:**
- Minimal & maximal delay after creation
- Minimal elapsed time
- Minimal execution time
- Absolute deadline

---

**Deadlines (absolute, elapse, or execution time) can be:**
- **Hard**
  - Single failure leads to severe malfunction
  - Results are meaningless after the deadline
- **Firm**
  - Results are meaningful after the deadline
  - Only multiple or permanent failures threaten the whole system
- **Soft**
  - Results may still be useful after the deadline

---

**Temporal Scopes can be:**
- **Periodic**
  - e.g. controllers, samplers, monitors
- **Aperiodic**
  - e.g. ‘periodic on average’ tasks, burst requests
- **Sporadic / Transient**
  - e.g. mode changes, occasional services
Direct recursions

- enabling a full pre-runtime analysis

- either verified at compile-time

- or checked at runtime

- checked for schedulability at compile-time

- verified at compile-time

- or by adding mandatory timeout

- assert lower and upper limits for the number of iterations per block

- but: no formal time-scope specification!

- no loop! (scheduler activates process)

- no formal time-scope specification!

- no schedulability analysis!

- Real-Time Euclid was suggested by Kligerman and Stoyenko in 1986

- or by adding mandatory timeout

- connect event to an interrupt-address

- Scheduled event to an interrupt-address

- Real-time Java

- Esterel

- Ada95

- POSIX

- DSP

- CRL

- Ada-Emulation of the above

- RT-Euclid program

- Esterel

- Real-time Java

- Ada95

- POSIX

- DSP

- CRL

- Ada-Emulation of the above

- RT-Euclid program

- Evaluation of the constraints and assertions:

  - Timing:
    - verified at compile-time

  - Activation / Deactivation:
    - checked for schedulability at compile-time and enforced by the scheduler at run-time.

  - Iterations and recursion:
    - either verified at compile-time or checked at run-time.
Real-time Java

Real-time Java comes with:

- multiple sets of predefined time-scope parameters
- a scheduler class (with a predefined priority scheduler)
- Schedulability (feasibility) analysis possible.
Specifying timing requirements

Real-time Java

The minimal required implementation supplies:

- Priority scheduling
- On-line schedulability analysis
- Deadline violation handlers
- (max. execution time deadline checks are suggested but not required)
- A sporadic scheduler is not required (although the sporadic release parameter set is).

Hard-real-time environments require the exclusive usage of ‘No heap real-time threads’

(synchronization with an ‘object-oriented’ thread invalidates the feasibility assurances)

- Ada95 has no explicit time-scope expressions at task-level.
  - Ada95 offers...
    - tasking
    - a priority scheduler (the only required scheduler)
    - synchronization and communication primitives
    - asynchronous transfer of control, timed calls, timeout on actions, etc.
  - ... but no hardware timers!

... to create the basis for most kinds of hard real-time-scope manually.

- ... but no automatic schedulability analysis!

Real-Time & Embedded Systems

Specifying timing requirements

Esterel

Since Esterel is a synchronous language, ...

... all actions and communications take zero time by definition.

- There is no expression for continuous, non-zero time-scoops.
- Time is interpreted in the reductionist way as a sequence of events
- Time-scoops translate to signal-relations and signal-counters

Continuous time scopes need to be taken into account while

1. analysing and reducing the problem to a zero-time atomic system
2. implementing the synchronous system on an actual system.

- Continuous time-scoops for the ...

... validation of the zero-time assumption!
Satisfying timing requirements

Two paths towards fulfilling rt-requirements:

☞ Real-time logic

formal, correct in its specifications, & offers calculus for asynchronous, real-time systems
- needs to ignore most real world effects, like jitters, drifts, failures, interferences, etc. pp.
- gives a correct solution according to the specification

☞ Complex systems oriented approach

deals with existing computer systems, sensors, & offers a set of approximating methods
- not complete or correct in any formal sense
- deals with real-world systems, gives 'robust' systems, passes rigorous experiments

Real-time logic approach

establish a reduced base by removing asynchrony:
- reduce all asynchronous, analogue, fractal, jitter-, drift, or failure-affected parts of the system
to a fully synchronous and discrete system: chapters 5 and 6
- formulates the specification on the basis of the reduced synchronous system

Complex systems oriented approach

• System identification and compile-time analysis:
  - Calculate or limit statement durations
  - Calculate or limit iterations and recursions
  - Analyze potential dead- or life-locks = chapter 8
  - Calculate schedulability = chapter 7

• Run-time analysis and checks:
  - Dynamic scheduling schemes: Re-validate schedulability = chapter 7
  - Check for all constraints and assertions at run-time: chapter 9

• Supply fault-tolerant behaviours:
  - Error recoveries, mode changes, ...

Fulfilling rt-requirements:

Real-time logic approach:

• Reduce the problem:
  - Reduce any asynchronous, analogue, dynamical, fractal, jitter-, drift, or failure-affected parts of the system to a fully synchronous and discrete system: chapters 5 and 6
  - Formulate the specification on the basis of the reduced synchronous system

• Compile the reduced system:
  - Verify the reduced synchronous against the specifications: not covered in this course
  - Compile the reduced system to an actual system:
    - The resulting actual system will be executable on a real machine and employ real devices

• Re-check the actual system (e.g. by means of a complex systems-approach): ...

Summary

Time & Space

• What is time? / What is embodiment?
  - Approaches by different faculties to understand the basis for this course

• Interfacing with time
  - Formulating local time-dependent constraints
  - Access time, delay processes, detect timeouts (in different languages)

• Specifying timing requirements
  - Formulating global timing constraints
  - Understanding time-scope parameters (and expressing them in different languages)

• Satisfying timing requirements
  - Real-time logic and complex systems approach