MÁSTER UNIVERSITARIO EN SISTEMAS ESPACIALES
POR LA UNIVERSIDAD POLITÉCNICA DE MADRID

POLITÉCNICA

2020.03.30

Data handling
Real-time systems
Real-time clocks and timers

Juan Antonio de la Puente juan.de.la.puente@upm.es

STRAST

Time references

- Real-time systems must execute their actions within specified time intervals
- Time can be measured using different reference systems, e.g.
  - Universal time (UT1): based on Earth rotation
  - Barycentric Dynamical Time (TDB): for Solar system
  - International Atomic Time (TAI): based on atomic clocks
  - Coordinated Universal Time (UTC): based on TAI
    - adjusted to remain within 1 s of UT1
  - Standard (zone) time: for legal uses, e.g. CET/CEST, GMT
    - CET = UTC +1

Real-time clocks and timers

- Real-time programs need to
  - access the value of real-time
    - not the same as time-of-day or universal time
    - real time clock values must be monotonic non-decreasing
      ✓ no big jumps either
  - delay execution of a task
    - delay task for some time interval
    - delay task until some absolute time is reached
  - execute some actions at specific times
  - monitor response times of tasks
  - monitor consumption of processor times by tasks
- Ada provides support for time-related functions

Basic time facilities in Ada

- Ada.Calendar package
  - a time-of-the-day clock
    - may change with time zone and daylight saving time
    - not appropriate for real-time tasks

```
package Ada.Calendar is
   type Time is private;
   ...
   function Clock return Time;
   ...
end Ada.Calendar;
```

- The primitive data type `Duration` represents time intervals measured in seconds
  - fixed-point real type, at least -86 400.0..+86 400.0

The Ada.Real_Time package

- Provides a fine-grained Time type with a monotonic clock
- And a Time_Span type for time intervals

```
package Ada.Real_Time is

   type Time is private;
   ...
   type Time_Span is private;
   ...
   Tick : constant Time_Span;

   function Clock return Time;
   ...
end Ada.Real_Time;
```

Delay statements in Ada

- Relative delay statement

  ```
  delay <duration>;
  ```

  suspends the task for the specified time duration

- Absolute delay statement

  ```
  delay until <time>;
  ```

  suspends the task until the real-time clock reaches the specified time

Periodic and sporadic tasks

- Tasks in real-time systems may have different execution patterns
  - Periodic tasks are requested to execute every T seconds
    - *T* is the task period
  - Aperiodic tasks are requested to execute whenever some event E occurs
    - *E* is the activation event of the task
  - Sporadic tasks are aperiodic tasks that have a minimum separation between two successive occurrences of the activation event
    - The minimum separation is denoted by *T*

Periodic task pattern

```
task type Periodic (Period : Time_Span);

task body Periodic is
   Next_Time : Time := Clock;
begin
   loop
      delay until Next_Time;
      ... -- periodic job
      Next_Time := Next_Time + Period;
   end loop;
end Periodic;
```

Aperiodic task pattern (1)

```
task Aperiodic;

protected Event is
   procedure Signal;
   entry Wait;
private
   Occurred : Boolean := False;
end Event;
```

Aperiodic task pattern (2)

```
protected body Event is

   procedure Signal is
   begin
      Occurred := True;
   end Signal;

   entry Wait when Occurred is
   begin
      Occurred := False;
   end Wait;

end Event;
```

Aperiodic task pattern (3)

```
task body Aperiodic is
begin
   loop
      Event.Wait;
      ...                    -- aperiodic action
   end loop;
end Sporadic;
```

Sporadic task pattern

```
task type Sporadic (Separation : Time_Span);
```

```
task body Sporadic is
   Release_Time : Time;
   Next_Release : Time := Clock;
begin
   loop
      delay until Next_Release;
      Event.Wait;
      Release_Time := Clock;
      ...                  -- sporadic action
      Next_Release := Release_Time + Separation;
   end loop;
end Sporadic;
```