CUARTO CURSO
SISTEMAS EN TIEMPO REAL

Escuela Politécnica Superior
Universidad de Huelva

Departamento de Ing. Electrónica,
Sistemas Informáticos y Automática

Práctica 3: Programación de Tareas en Ada

Manuel Sánchez Raya
Versión 0.1
24 de Noviembre de 2014

ÍNDICE

2.1. Introducción.

Con esta práctica se pretende que el alumno:

- Programe tareas periódicas.
- Programe los periodos, plazos de respuesta y detecte el incumplimiento de plazos de respuesta.

2.2.- Tareas a Realizar.

Se debe desarrollar una aplicación que ejecute cuatro tareas periódicas con las siguientes características, haciendo uso de tareas ADA:

- Las tareas escriben su nombre (p.ej A, B, C y D) y la hora.
- Los periodos de las tareas son, 1, 2, 3 y 4s.
- Los plazos de respuesta son iguales a los periodos.
- Se debe detectar y tratar el incuplimiento de los plazos de respuesta.

Para desarrollar la aplicación se deben tener a mano los apuntes de la asignatura, y tomar como base los siguientes procedimientos:

Procedimiento para escribir la hora:

```
with Ada.Real_Time;
with Ada.Dynamic_Priorities;
with Ada.Text_IO; with Ada.Integer_Text_IO; with
Ada.Float_Text_IO;
with Calendar;

procedure Write_Time (Actual_Time : Calendar.Time) is
begin
Ada.Integer_Text_IO.Put(Integer(Calendar.Year(Actual_Time)),4);
      Ada.Text_IO.Put(":");
Ada.Integer_Text_IO.Put(Integer(Calendar.Month(Actual_Time)),2);
      Ada.Text_IO.Put(":");
Ada.Integer_Text_IO.Put(Integer(Calendar.Day(Actual_Time)),2);
      Ada.Text_IO.Put(":");
Ada.Float_Text_IO.Put(Float(Calendar.Seconds(Actual_Time)));
      Ada.Text_IO.New_Line;
end Write_Time;
```

Paquete con declaraciones globales:

```
with System;
with Ada.Real_Time;

package Global is
      type     Execution_Type     is     (Correct,     Internal_Error,
External_Error);
```

```
type Periodic_Profile_Type is record
      Period : Ada.Real_Time.Time_Span;
      Phase : Ada.Real_Time.Time_Span;
      Priority : System.Priority;
      Deadline : Ada.Real_Time.Time_Span;
end record;

type Sporadic_Profile_Type is record
      Interarrival : Ada.Real_Time.Time_Span;
      Priority : System.Priority;
      Deadline : Ada.Real_Time.Time_Span;
end record;

Start_Time : Ada.Real_Time.Time := Ada.Real_Time.Clock;

procedure Safe_Stop;

end Global;
```

Paquete con definición de tarea periódica:

```
package P_Periodic_Task is
      task Periodic_Task;
end P_Periodic_Task;

with Global;
with Write_Time;
with Ada.Real_Time;
with Ada.Calendar;
with Ada.Text_IO;
package body P_Periodic_Task is

Task_Profile : Global.Periodic_Profile_Type :=
            (Period=> Ada.Real_Time.Milliseconds(. . . .),
             Phase=> Ada.Real_Time.Milliseconds(0),
             Priority=> . . . .,
             Deadline=> Ada.Real_Time.Milliseconds(. . .
.));

procedure Handle_Failure (Fail : Global.Execution_Type) is
begin
      . . . . .
end Handle_Failure;

procedure Periodic_Activity is
begin
      . . . . . .
end Periodic_Activity;

task body Periodic_Task is
use type Ada.Real_Time.Time;

Next : Ada.Real_Time.Time := Global.Start_Time +
Task_Profile.Phase;

begin -- Periodic_Task
loop
      . . . . . .
end loop;
```

```
exception
      . . . . . .
end Periodic_Task;
end P_Periodic_Task;
```

Procedimiento principal:

```
with Periodic_Task_X;
. . .
with Sporadic_Task_X;
. . .
procedure Main is
begin
      null;
end Main;
```

Fichero con los pragmas de configuración, según el formato de gnat:

```
pragma Task_Dispatching_Policy (FIFO_Within_Priorities);
pragma Queuing_Policy (Priority_Queuing);
pragma Locking_Policy (Ceiling_Locking);
```