Lab #2
⋆
Periodic tasks and Alarms

March 19, 2013

Note: All the software and documents are stored at `http://www.irccyn.ec-nantes.fr/~bechenne/trampoline-isis`

1 Goal

Real-Time systems are reactive systems which have to do processing as a result of events. You have seen in Lab #1 how to start processing as a result of an internal event of the system: by activating a task (`ActivateTask` and `ChainTask` services) or by setting an event (`SetEvent` service). In this lab, you will see how to trigger processing as a result of time passing (expiration of an Alarm). This lab uses the following concepts: alarm, counter. On the TP-ECN board, TIMER0 is used as interrupt source for alarms. The interrupt is sent every 1ms.

Go into the `trampoline/labs_isis/lab2` directory.

2 First application

lab2 application starting point This application implements a periodic task that reads the push buttons of the board every 100ms.

Using the application of lab2 as a starting point, program an application which does a computation when a button is pushed. You will use a task named `t_process`, priority 3 that on odd execution displays "processing triggered" and on even executions clears the LCD.

3 Second application

The second application will use 2 periodic tasks: `t1` (priority 2, period 1s) and `t2` (priority 1, period 1.5s). t1 toggles LED S0 each time it executes and t2 toggles LED S1.

Question 1 *Do not program directly. Give by hand the 20 first states of the LED given by the execution of the application with the display date of each state (0 being the application startup date). Is the whole system periodic ? If yes, what is the period and the behavior.*

Question 2 *The application needs a counter and 2 alarms. In Trampoline/ARM a counter is connected to a timer with a 1ms cycle time. What maximum* `TICKSPERBASE` *do you use to fulfill the application requirements ?*

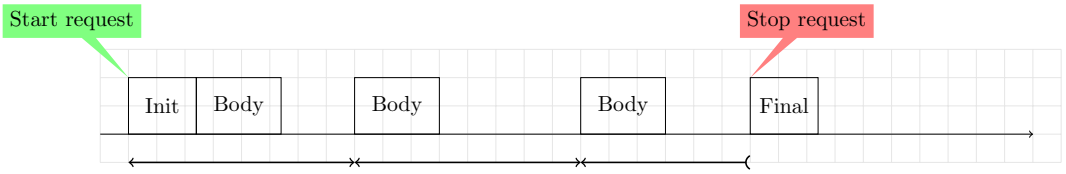
Question 3 *How are the alarms configured to fulfill the application requirements ? Declare the counter and both alarms and write the application. Verify it works.*

4 Third application

In the third application, alarms, counters and polling on the push buttons are mixed. This application is a system with 2 push buttons. After starting the system waits. When the button is pressed, the system start a function F that is implemented using a periodic task (period = 1s). To "see" F, uses a blinking LED as in the second application. When the button is pressed again, function F is stopped. When the switch is pressed, the system is shutdown as quickly as possible (ie ShutdownOS is called).

Question 4 *Design and program this application using Trampoline.*

Requirements change. Now function F implementation needs an Init code (runs once when the F is started) and a Final code (runs once when F is stopped). This corresponds to the following diagram:



Question 5 *Modify the application to take the new requirements into account. Use 3 basic tasks to implement function F. Init and Final print their names on the LCD.*

Question 6 *Same question but with only one extended task to implement function F.*

5 Fourth application

In this part, you will implement a watchdog. It is a mechanism that allows to stop a processing or the waiting for an event when a deadline occurs.

Question 7 *In your application, each time F0 is pressed, F1 must be pressed within 2 seconds. In such case, you print the time between the two occurrences. Otherwise, an error message is displayed. If 2 or more F0 are got within 3 seconds from the first one they are ignored.*

Question 8 *What is happening if the timeout occurs just after F1 has been pressed but before the waiting task got the event ? (draw a Gantt diagram of this scenario) If your application does not handle correctly this scenario, modify it.*

6 Fifth application

Program a chase[1] with a 0.5s period. To do it, use 4 periodic tasks. Each periodic task manages a LED. The chase effect is done by using alarms with a time shift between them.

When F0 is pressed, the chase stops. When F1 is pressed, the chase continues. When F2 is pressed, the chase direction changes (even if it is stopped).

[1] chenillard in French