Real-Time Systems (2IMN20)
- Practical training 2015/2016 -

Leo Hatvani, Erik J. Luit and Reinder J. Bril
Technische Universiteit Eindhoven (TU/e),
Department of Mathematics and Computer Science,
Group System Architecture and Networking (SAN),
Den Dolech 2, 5612 AZ Eindhoven, The Netherlands
`L.Hatvani@TUe.NL`, `E.J.Luit@TUe.NL`, `R.J.Bril@TUe.NL`

Recommendations

As a reminder, we recapitulate the recommendations from the general introduction:

1. Keep in mind that this practical training is about real-time systems, so ask yourself if certain behavior that your solutions imply are desirable for this kind of systems.
2. Be critical and study the code carefully.
3. Be attentive to clues in the code provided.
4. Whenever a first question mentions something, this is in many cases a hint towards an implementation in a subsequent question.

Please be aware that the method `CountDelay()` is introduced to illustrate what happens when jobs violate computation bounds.

Week 1: Non-interruptable task execution (`SchedulerNPBasic.c`)

- The submission deadline for the exercises of week 1 is Thursday, February 18^{th}, 2016 (23:59h).
- You will be graded on a scale from 0-10; with each of the questions you can earn 1 point.
- You are expected to motivate your answers briefly.

1 Preparation

A package will be available containing the following source files:

- `Led.h` and `Led.c`;
- `Clock.h` and `Clock.c`;
- `ErrorCodes.h`;
- `Scheduler.h` and `SchedulerNPBasic.c`; and
- `SchedTest.c`.

Study the distributed code[1].

[1] The MSP430x4xx User's Guide can be found at http://www.win.tue.nl/san/education/2IN26/MSP430%20-%20general.pdf.

2 Basic questions

Answer the following questions:

1. Timer frequency and periods of tasks:
   (a) What is the timer frequency, or how many timer ticks are there per second?
   (b) What are the periods of the tasks in `TestSched.c` (in seconds)?
2. Which task has the highest priority?
3. Which leds are actually toggled during runtime?
4. Is the execution-time of the interrupt-handler fixed or fluctuating?

3 Hypotheses

In this part, we will work towards a formulation of hypotheses about the behavior of the system. For this purpose, you are asked to draw the *expected behavior* of an example system.

3.1 Questions about detailed start-up behavior

The aim is to show *the order* in which leds are changed. It is therefore sufficient to draw a timeline for 0.25 seconds after the system has been initialized.

Specific questions

5. Which led is changed first?
6. The period of `BlinkRed` is 0. What happens with the *BlinkRed* function?

3.2 Questions about coarse-grain behavior

The aim is to show ON/OFF outputs over a longer interval of time. Draw a timeline for 3.5 seconds, using a granularity of approximately 0.25 second. Make the following assumptions for the latter:

- the time to start-up the system can be neglected;
- apart from `CountDelay(60000)`, the duration of all functions may be neglected;
- `CountDelay(60000)` in the body of `BlinkGreen()` lasts 0.75 seconds.

Specific questions

7. After 1.5 seconds, `BlinkGreen` is activated. Does its execution fit within one timer period?
8. How many interrupts can be pending simultaneously? What happens when more interrupts arrive?
9. The example application suffers from *drift*. How can this drift be observed?

3.3 Additional question about detailed behavior

The aim is to show the execution of the timer-interrupt handler. Draw a timeline for the execution of the interrupt handler from the end of the 4^{th} execution t_{END} of `BlinkGreen` for 5 ms. Make the following assumptions for the latter:

- a clock period T_{Clk} of 1 ms;
- the Interrupt Handler takes 0.45 ms to execute when no tasks are activated;
- the Interrupt Handler finishes 0.2 ms after t_{END};
- the last clock interrupt came at $t_{\mathrm{END}} - 0.6ms$.

Take into account how many interrupts may be pending.

Specific questions

10. How many consecutive executions of the Interrupt Handler take place after it finishes 0.2 ms after t_{END}?