IN4343 – Lab 5

Casper van Wezel (4209192) & Erwin de Haan (4222814) – 2017-04-04

# Extend SchedulerPre.c with support for non-pre-emptive tasks

**1.a.**

1. **Does it make sense to have non-pre-emptive DIRECT tasks?**

Yes, time is a critical factor for a DIRECT task. With preemption you would lose all control over time again.

1. **Does it make sense to disallow non-pre-emptive DIRECT tasks?**

No, most DIRECT tasks cannot be preempted. Think about communication with time critical (external) devices, when a quick response is needed by a DIRECT task, you cannot stop mid-way responding to do another task first.

1. **Does it make sense to allow a DIRECT task to be executed when a non-pre-emptive task is interrupted by the interrupt handler?**

No, because the first task is non-pre-emptive, so it should never be interrupted. The DIRECT task has to wait (that's just the risk you get with non-pre-emptive tasks, starvation and deadline misses just around the corner!)

**1.b.** In HandleTasks() all the tasks with a higher priority are checked if they need to be invoked first. To be sure that this does not happen when the current task is an non-pre-emptive task, this can be checked by doing something like this at the start of the HandleTasks(). (Note: an extra flag is needed for this, which is fine sine Flags is an uint16\_t and we currently only have 4 bits in use.)

Taskp OldTask = CurrentTask();

if (!(OldTask->Flags & FPDS)) {

[...]

}

**2.**

**[Scheduler.h]:** Flag for non-preemtive tasks

#define FPDS 128

**[SchedulerPre.c:HandleTasks()]:** check FPDS flag before interrupting the old task.

Taskp OldTask = CurrentTask();

if (!(OldTask->Flags & FPDS)) {

[...]

}

Furthermore we check all the cases where Taskf() is executed for possible errors:

**[SchedulerPre.c:HandleTasks()]:**

Already fixed above for FPDS tasks, does not have to handle DIRECT tasks differently.

**[SchedulerPre.c:TimerIntrpt()]:**

if (t->Flags & DIRECT) { t->Invoked++; t->Taskf(); }

FPDS tasks will not be interrupted here because interrupts are still disabled (we are still in the Interrupt context).

**3.**

void HandleTasks(void)

{

int8\_t oldBP = BusyPrio; // Save BusyPrio = current task handling level

**Taskp OldTask = CurrentTask();**

**if (!(OldTask->Flags & FPDS)) { // First check if it is allowed to preempt**

**the current task.**

Pending = 0; // This instance will handle all new

// pending tasks.

BusyPrio = NUMTASKS-1; // Start at highest priority

while (BusyPrio != oldBP) {

Taskp CurTask = CurrentTask ();

while (CurTask->Activated != CurTask->Invoked) {

CurTask->Invoked++;

if (CurTask->Flags & TRIGGERED) {

\_EINT(); CurTask->Taskf(); \_DINT();

}

else CurTask->Activated = CurTask->Invoked;

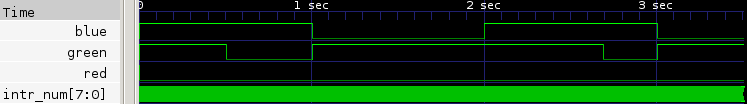
}

BusyPrio--;

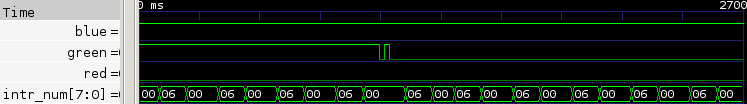
} }

**}**

**4.a.** In the two pictures below it can clearly be seen that the Blue task interrupts/pre-empts the counting Green task at t=2s.

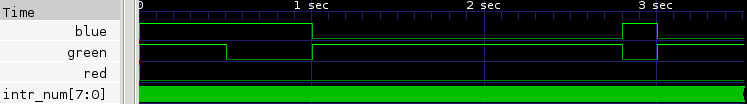


Figuur 1: Existing functionality (coarse-grain, first 3.5s)

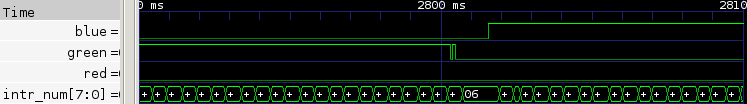


Figuur 2: Existing functionality (fine-grain fgreen,2, 2690-2700ms)

**4.b.** When registering the green task as non-preemptive (FPDS), the Blue task has to wait until the Green task is completely finished.



Figuur 3: New functionality (coarse-grain, 3500ms)

****

Figuur 4: New functionality (fine-grain fgreen,2, 2690-2700ms)

# Extend SchedulerPre.c with support for deferred pre-emptive tasks

**5.a. Which measures need to be taken when a non-FPDS task calls Yield()?**

HandleTasks() should be executed again, and just like the other calls it is nice to first check if there are any Pending tasks at all.

if (Pending) HandleTasks();

**5.b. Which measures need to be taken when a DIRECT tasks calls Yield()?**

Since DIRECT calls are time critical, they should just continue despite the Yield() call.

**6.**

// This operation allows all pending jobs of tasks with

// a priority higher than the currently executing task to preempt.

// Once all that pending (+new arrived) work has been completed,

// the pre-empted task is allowed to continue its execution non-pre-emptively.

void Yield(void){

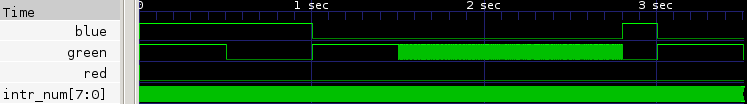
Taskp OldTask = CurrentTask();

// HandleTasks() if any Pending, and currently not doing a DIRECT or FPDS task

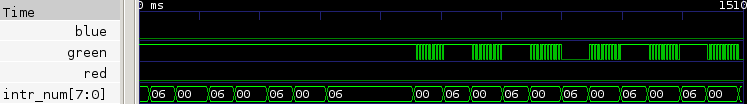
if (Pending && !(OldTask->Flags & (DIRECT | FPDS))) HandleTasks();

}

**7.** Registering BlinkGreen() as an FPDS does not give the most interesting results, but here we go. When removing the FPDS flag, one can observe that the Green Task will Yield(), in order for the higher Prio Blue task. When looking at the fine-grain picture, one can observe the longer spaces between the toggles, this is because of the Timer Interrupt of course (to confirm this, they are ~1ms apart, which corresponds to the 1024Hz).



Figuur 5: Extended with Yield() (coarse-grain, first 3.5s)



Figuur 6: Extended with Yield() (fine-grain, 1500-1510ms)