IN4343 – Lab 4

Casper van Wezel (4209192) & Erwin de Haan (4222814) – 2017-04-02

# Basic questions

**1.a.** BlinkRed() is registered with

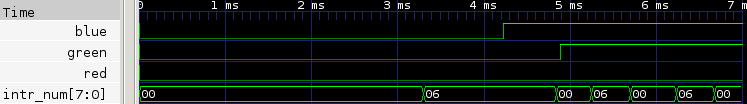
Phasing = 0;

Period = 0;

Flags = 0;

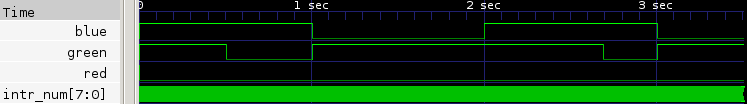
Because the Phasing & Period values are both equal to zero (and thus not larger than zero), the PERIODIC and TT (Timer Triggered) flags are not set. So the only task->Flags set will be the TRIGGERED flag.

So whenever the Red task is checked in the TimerIntrpt function the 'if (t->Flags & TT)' will always fail and thus the Taskf() will never be executed (and the LED will stay off, as seen in Figuur 1).

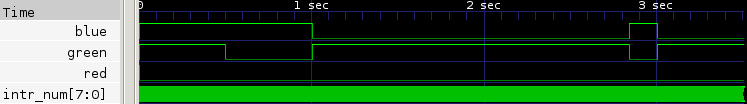


Figuur 1: Start-up behavior (fine-grain, first 7ms)

**1.b.** The major difference between the behavior of SchedulerPre.c and SchedulerNP.c is the moment where the on blue signal is toggled on.



Figuur 2: Start-up behavior (coarse-grain, first 3.5s)



Figuur 3: Start-up behavior NP (coarse-grain, first 3.5s)

# Implement Activate()

**2.** When the supplied parameter for the Prio is higher than the allowed number of tasks, E\_BOUNDS is returned (a strong precondition because this is just not possible). Otherwise the function will just continue executing and eventually return a E\_SUCCESS.

**3.**The new body of the RestoreSW() function looks like this:

if (Pending && (sw & INTRPT\_BIT)) HandleTasks ();

// r2 = sw

asm volatile ("mov.w %0, r2\n\t" :: "r"(sw));

The last line is still the same, it moves sw back into R2/Status Register. But before that a new condition is checked: If there are still tasks pending, and it is allowed to interrupt the current task, it will handle all tasks again (i.e. execute them with the correct priority).

**4.a.** DIRECT tasks should be executed as soon as they are ready, which is of course still within the TimerIntrpt function. Also for within Activation(), this is the correct way to go, so the Activated and Invoked counters are increased, and the Task is executed.

**4.b.** Ticks == 0

This means that this task has to be activated, this is done by setting the TRIGGERED flag and increasing the Activated count. In the HandleTasks() function, the inequality between the #activations and #invokes will be noticed and the task will be executed (still in order of Prio of course).

**4.c.** Ticks > 0

This means it is not yet time to activate the tasks, but to be able to time the number of time units to go, Ticks is stored into the Remaining parameter of the task (if it is not already set to a lower value)

**4.d.** When a direct task executes a function, it should activate this function instantaneously, to not delay the DIRECT parent task, however there is not enough information in this function to detect such cases.

**5.a.**

uint8\_t Activate(uint8\_t Prio, uint16\_t Ticks) {

if (Prio>=NUMTASKS) return (E\_BOUNDS);

Taskp CurTask = Prio2Taskp(Prio);

if (Ticks == 0) { //Activation is supposed to be now

if (CurTask->Flags & DIRECT){ // DIRECT task detected, so execute it right away

CurTask->Activated++;

CurTask->Invoked++;

CurTask->Taskf();

}

else { CurTask->Flags |= TRIGGERED; CurTask->Activated++; }

}

if (Ticks > 0) { //Activation is supposed to be somewhere in the future

//if Activation is not already scheduled, do so

if (Ticks < CurTask->Remaining || CurTask->Remaining == 0){

CurTask->Remaining = Ticks;

CurTask->Flags |= TT;

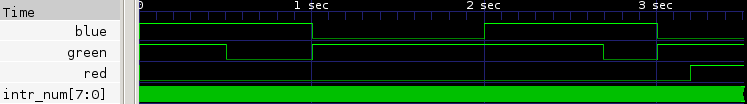
}

}

return E\_SUCCESS;

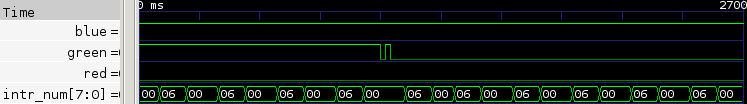
}

**6.** What can be seen in this picture is that the Red signal is indeed toggled 512 ticks after the fourth green toggle and the 60000 counts delay. (note: the second negative edge of the Green signal is actually a bit bouncy, so the fourth green toggle is at the second negative Green edge in this picture)

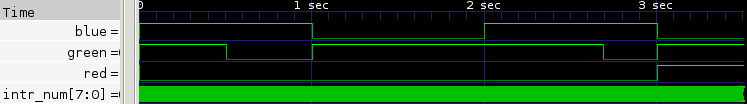


Figuur 4: ActivateRed(4,512)

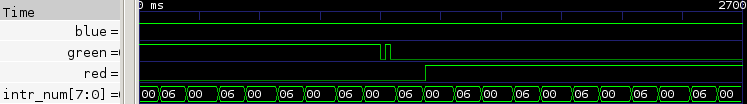
**7.a.** As expected, the 512 ticks delay between Green and Red as seen in 6) is gone now.



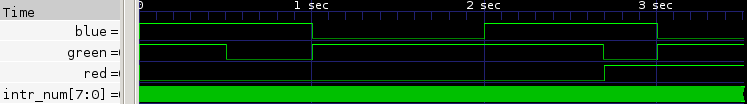
Figuur 5: ActivateRed(4,0) (fine-grain, 2nd falling edge (2690-2700ms))

Figuur 6: ActivateRed(4,0) (coarse-grain, first 3.5s)

**7.b.** Now the Red transition is in front of the Green transition (+60000 wait)

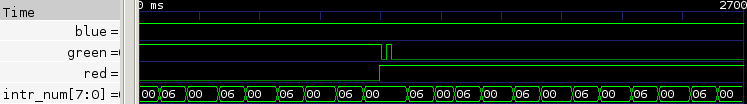


Figuur 7: ActivateRed(2,0) (fine-grain, 2nd falling edge (2690-2700ms))

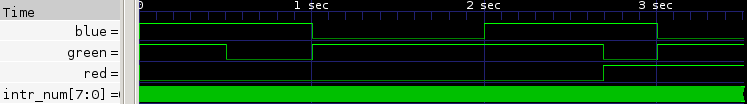


Figuur 8: ActivateRed(2,0) (coarse-grain, first 3.5s)

**8.a.** The Red signal is toggled now directly with the green one, just as expected.

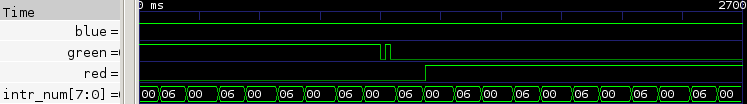


Figuur 9: ActivateRedDirect(4,0) (fine-grain, 2nd falling edge (2690-2700ms))



Figuur 10: ActivateRedDirect(4,0) (coarse-grain, first 3.5s)

8.b. And the priority does not make a difference.



Figuur 11: ActivateRedDirect(2,0) (fine-grain, 2nd falling edge (2690-2700ms))