

Rapporttitel

ONDERTITEL RAPPORT

Naam | Cursustitel | Datum

# Week 3

## 3.3

### B

Code:



### C

Omdat de flag verandert moet kunnen worden in de ISR, en de compiler moet deze variabele negeren tijdens het optimaliseren. Als dit niet gebeurt kan de variabele zodanig veranderd worden dat de ISR en niet meer bij kan.

### D

Code:



## 3.4



## 3.5



## 3.6

Different functions for task management:





Task structure with task functions

**enum** {*NOT\_AVAILABLE*, *WAITING*, *READY*};

**void** **Toggle\_Green**(**void**){

GPIOD->ODR ^= 0x1000;

}

**void** **Toggle\_Orange**(**void**){

GPIOD->ODR ^= 0x2000;

}

**void** **Toggle\_Red**(**void**){

GPIOD->ODR ^= 0x4000;

}

**void** **Toggle\_Blue**(**void**){

GPIOD->ODR ^= 0x8000;

}

**struct** tasks{

**void** (\*function)(**void**);

**int** period;

**int** counter;

**int** initial\_delay;

**int** state;

} task;

**struct** tasks task\_list[8];

Main function section for task adding and interrupt waking:

**int** ret = Add\_task(&Toggle\_Green, 200, 100);

**if** (ret == 0){

**while**(1);

}

ret = Add\_task(&Toggle\_Orange, 500, 200);

**if** (ret == 0){

**while**(1);

}

ret = Add\_task(&Toggle\_Red, 750, 300);

**if** (ret == 0){

**while**(1);

}

ret = Add\_task(&Toggle\_Blue, 300, 400);

**if** (ret == 0){

**while**(1);

}

// Do forever:

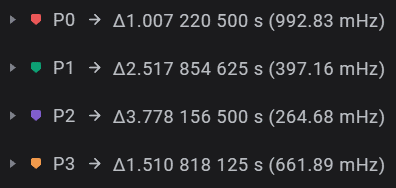
**while** (1)

{

\_\_WFI();

runReadyTasks();

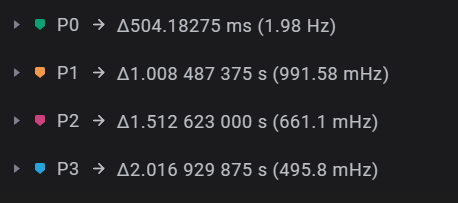
}



Results from logic analyzer

|  |  |
| --- | --- |
| Logic signal | Corresponding function |
| P0 | Toggle\_Green |
| P1 | Toggle\_Orange |
| P2 | Toggle\_Red |
| P3 | Toggle\_Blue |

## 3.7



Results from the logic analyzer including the initial delay ticks. Every tick is 0.005s, therefore 100 ticks is equal to 0.5S

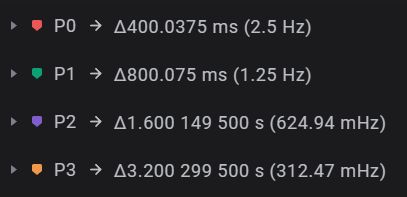
|  |  |  |  |
| --- | --- | --- | --- |
| Logic signal | LED | Set cycles (ticks) | Corresponding time (s) |
| P0 | Green | 100 | 0.5 |
| P1 | Orange | 200 | 1 |
| P2 | Red | 300 | 1.5 |
| P3 | Blue | 400 | 2 |

# Week 4

## 4.1

### D

The times are x4 because every 1ms the tasks switches to the next task. Therefore after 4ms task 1,2,3 and 4 are 1ms further in their own scope. This is an order execution of tasks, without priorities.



|  |  |  |  |
| --- | --- | --- | --- |
| Logic signal | LED | Set cycles (ticks) | Corresponding time (ms) |
| P0 | Green | 100 | 400 |
| P1 | Orange | 200 | 800 |
| P2 | Red | 300 | 1200 |
| P3 | Blue | 400 | 1600 |

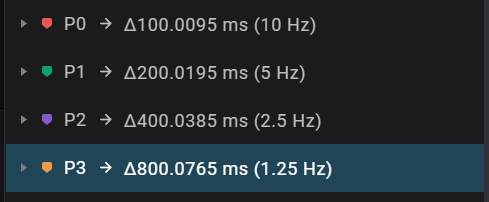
## 4.2

### A

Adjustments in code:



### B



|  |  |  |  |
| --- | --- | --- | --- |
| Logic signal | LED | Set cycles (ticks) | Corresponding time (ms) |
| P0 | Green | 100 | 100 |
| P1 | Orange | 200 | 200 |
| P2 | Red | 300 | 400 |
| P3 | Blue | 400 | 800 |

## 4.3

### A



Code for changing period to 2ms



Added a set Control function to set the nPRIV bit high.