Inhoud

1 Opgave 1 2
1.1 2
2 Opgave 2 2
3 Opgave 3 2
3.1 2
3.2 2
3.3 2
3.4 3
3.5 3
4 Opgave 4 3
4.1 3
4.2 3
4.3 3
4.4 3
5 Opgave 5 4
5.1 4
6 Opgave 6 5
6.1 5
6.2 5
6.3 5
7 Opgave 7 5
8 Opgave 8 5
8.1 5
9 Opgave 9 6
9.1 6
9.2 6
9.3 7
9.4 7
9.5 7

1 Opgave 1

Clock = 32MHz

Delay = 500ms = 2Hz

Nieuwe frequentie = Clock/1024 = 31250Hz

PER = Nieuwe frequentie/Delay = 31250/2 = 15625

Ik heb de strcmp bij scanf gezet ipv ISR en mijn van mijn Richting variabele 2 variabelen gemaakt, omdat dit anders problemen gaf doordat het een mogelijkheid was dat scanf juist de Richting variabele veranderde op het moment dat ISR zijn strcmp uitvoerde.

1.1

Als we dit doen voor meerdere taken gaat dit zeer veel interrupts eisen en gaat de code veel onduidelijker worden. OS is hier dus ideaal voor aangezien deze dit voor ons gaat doen en er ook een mogelijkheid is om priority niveaus aan te geven.

2 Opgave 2

3 Opgave 3

xTaskCreate parameters (pg.34)

pvTaskCode = Looplicht

pcName = “looplicht”

usStackDepth = configMINIMAL_STACK_SIZE

pvParameters = NULL

uxPriority = `tskIDLE_PRIORITY+1`

pxCreatedTask = NULL

3.1

Cursus: “De _delay_ms() functie maakt intern gebruik van een ‘delay loop’. Dit wil zeggen dat de processor een aantal keer een lus zal doorlopen waarvan het aantal processorcycli om deze te doorlopen gekend is.”

Wat dus wilt zeggen dat als de OS schakelt naar de Terminal task deze delay loop stopt en pas verder gaat wanneer er terug geschakeld wordt naar de Looplicht task, waardoor de delay extra groot wordt.

3.2

De Terminal task reageert niet meer door dat de we met CPU time starvation zitten, waardoor deze task nooit meer aan de beurt geraakt. Dit komt doordat Looplicht task een hogere prioriteit heeft gekregen en tasks met lagere prioriteit niet meer aan de beurt geraken.

3.3

Bij _delay_ms() blijft de processor vastzitten in de taak waarin het aangeroepen wordt. Bij vTaskDelayUntil() wordt de taak op blocked gezet waardoor het pas verder gaat na een bepaalde tellerwaarde en de processor tegelijkertijd nog andere tasks kan uitvoeren.

3.4

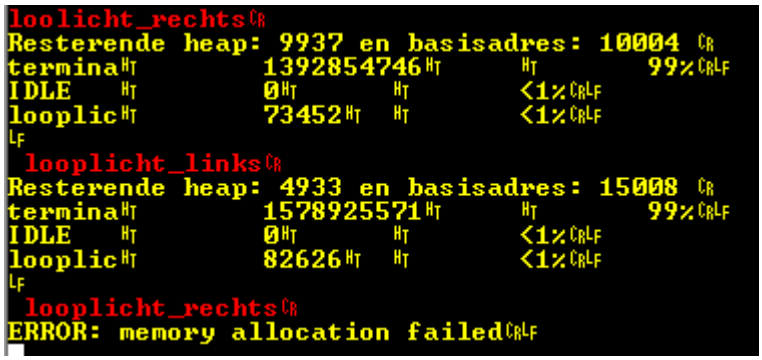
Bij vTaskDelay() is de tijd dat de task uit de blocked state gaat relatief met wanneer vTaskDelay() werd opgeroepen. Bij vTaskDelayUntil() is deze tijd absoluut en dus niet relatief met wanneer vTaskDelayUntil() werd opgeroepen.

3.5

Beide tasks gebruiken nu maar 1%, maar mijn Leds branden nu wel niet meer. Dit komt door dat bij co-operative scheduling een task gaat lopen tot het zijn controle over de cpu zelf loslaat, waardoor we blijven vastzitten in de Terminal task want deze wacht continu op input. Bij pre-emptive scheduling kan de OS een taak stopzetten ipv te moeten wachten tot de taak dit zelf doet.

4 Opgave 4

4.1



De adressen worden in stijgende richting op de heap toegekend. Als er meer geheugen wordt opgevraagd dan er vrij is faalt de memory allocation.

4.2

Bij A wordt het geheugen continu overschreven, maar bij B wordt er continu nieuw geheugen toegekend waardoor op den duur de heap vol geraakt. De oplossing is door achter de for loop nog vPortFree(num) te zetten. Dit maakt het geheugen dat num alloceert terug vrij.

4.3

“

* A sample implementation of pvPortMalloc() and vPortFree() that permits
 * allocated blocks to be freed, but does not combine adjacent free blocks
 * into a single larger block (and so will fragment memory). See heap_4.c for
 * an equivalent that does combine adjacent blocks into single larger blocks.
 *

”

Het probleem is dus duidelijk dat twee vrije aan elkaar liggende blokken niet terug tot 1 blok gevormd worden en er dus fragmentatie ontstaat. Dit zorgt ervoor dat het nieuwe blok dat even groot is als de twee vorige blokken opgeteld niet gealloceerd kan worden.

4.4

Doordat de 2 blokken die zijn vrijgemaakt niet langs elkaar liggen is het geheugen hierna in 2 delen opgesplitst wat ervoor zorgt dat een nieuw blok met geheugengrootte van deze 2 blokken samen niet aangemaakt kan worden.

“

* A sample implementation of pvPortMalloc() that allows the heap to be defined

* across multiple non-contigous blocks and combines (coalescences) adjacent

* memory blocks as they are freed.

*

“

Bij heap_5.c zou dit dus wel werken.

5 Opgave 5

5.1

De stack groeit naar onder en op niet gebruikte stack locaties staat er databyte waarde a5.

TCB = Rood en Stack = Groen.

```
data 0x275A  a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5
data 0x277C  a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5
data 0x279E  a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5
data 0x27C0  a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 02 ee 25 a5 a5 a5 a5 9a 61 28 0a 20 00 20 00 28 4c
data 0x27E2  28 4c 00 09 e0 0a a0 28 0a 20 00 00 00 20 00 00 00 31 0d 20 00 20 00 28 4c 28 4c 00 08 28 1a 00 04 21
data 0x2804  1b 00 c3 00 00 04 15 0d 00 20 00 0d 00 20 86 28 19 00 00 00 1b c8 32 38 34 30 31 16 15 14 13 12 11 10
data 0x2826  09 08 07 06 28 3e 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 00 20 a9 28 4b 00 19 d6 d9 20 fe 01
data 0x2848  4e 28 f2 28 f2 28 01 00 f2 28 0b 29 03 00 02 02 db 2b 60 00 ee 25 fe 01 00 00 28 66 00 19 df f2 28 01
data 0x286A  00 f2 28 0b 29 03 00 02 02 db 20 4d 00 ee 25 19 02 00 00 28 81 00 19 df f2 28 01 00 f2 28 0b 29 03 00
data 0x288C  02 02 db 98 39 00 ee 25 34 02 00 00 28 9c 00 19 df f2 28 01 00 f2 28 0b 29 03 00 02 02 dc 8d 26 00 ee
data 0x28AE  25 4f 02 00 00 28 b7 00 19 df f2 28 01 00 f2 28 0b 29 03 00 02 02 db 05 13 00 ee 25 6a 02 00 00 28 d2
data 0x28D0  00 19 df f2 28 01 00 f2 28 0b 29 03 00 02 02 00 00 00 00 ee 25 93 02 00 00 29 28 33 22 11 81 61 32 00
data 0x28F2  e6 27 00 00 9d 61 9d 61 f2 28 9a 61 02 00 00 00 00 00 f2 28 00 00 02 ee 25 72 65 63 75 72 73 69 00 03
data 0x2914  00 02 00 db b3 73 00 00 00 00 00 00 81 61 04 01 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5
data 0x2936  a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5
data 0x2958  a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5
data 0x297A  a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5
data 0x299C  a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5
data 0x29BE  a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5
data 0x275A  a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5
data 0x277C  a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 02 ee 25
data 0x279E  a5 a5 a5 a5 9a 61 27 d4 20 00 20 00 28 16 28 16 00 09 e0 0a a0 27 d4 20 00 0a a0 20 00 00 00 31 20 20
data 0x27C0  00 20 00 28 16 28 16 00 08 27 e4 00 04 21 1a 00 e3 00 00 04 10 0d 00 20 00 0d 00 20 86 27 e3 00 00 00
data 0x27E2  1b c8 32 38 34 30 31 16 15 14 13 12 11 10 09 08 07 06 28 08 17 16 15 14 13 12 11 10 09 08 07 06 05 04
data 0x2804  03 02 00 20 a9 28 15 00 19 d6 d9 20 c8 01 18 28 f2 28 f2 28 01 00 f2 28 0b 29 03 00 02 02 db be 86 00
data 0x2826  ee 25 c8 01 00 00 28 30 00 19 df f2 28 01 00 f2 28 0b 29 03 00 02 02 db b3 73 00 ee 25 e3 01 00 00 28
data 0x2848  4b 00 19 df f2 28 01 00 f2 28 0b 29 03 00 02 02 db 2b 60 00 ee 25 fe 01 00 00 28 66 00 19 df f2 28 01
data 0x286A  00 f2 28 0b 29 03 00 02 02 db 20 4d 00 ee 25 19 02 00 00 28 81 00 19 df f2 28 01 00 f2 28 0b 29 03 00
data 0x288C  02 02 db 98 39 00 ee 25 34 02 00 00 28 9c 00 19 df f2 28 01 00 f2 28 0b 29 03 00 02 02 dc 8d 26 00 ee
data 0x28AE  25 4f 02 00 00 28 b7 00 19 df f2 28 01 00 f2 28 0b 29 03 00 02 02 db 05 13 00 ee 25 6a 02 00 00 28 d2
data 0x28D0  00 19 df f2 28 01 00 f2 28 0b 29 03 00 02 02 00 00 00 00 ee 25 93 02 00 00 29 28 33 22 11 81 61 32 00
data 0x28F2  b0 27 00 00 9d 61 9d 61 f2 28 9a 61 02 00 00 00 00 00 f2 28 00 00 02 ee 25 72 65 63 75 72 73 69 00 03
data 0x2914  00 02 00 db c9 99 00 00 00 00 00 00 81 61 04 01 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5
data 0x2936  a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5
data 0x2958  a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5
data 0x297A  a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5
data 0x299C  a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5
data 0x29BE  a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5
data 0x275A  a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5
data 0x277C  a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5
data 0x279E  a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 02 ee 25 a5 a5 a5 a5 9a 61 27
data 0x27C0  ef 20 00 20 00 28 31 28 31 00 09 e0 0a a0 27 ef 20 00 0a 20 20 00 00 00 31 0d 20 00 20 00 28 31 28 31
data 0x27E2  00 08 27 ff 00 04 21 1b 00 c3 00 00 04 13 0d 00 20 00 0d 00 20 86 27 fe 00 00 00 1b c8 32 38 34 30 31
data 0x2804  1b 00 c3 00 00 04 15 0d 00 20 00 28 23 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 00 20 a9 28 30
data 0x2826  00 19 d6 d9 20 e3 01 33 28 f2 28 f2 28 01 00 f2 28 0b 29 03 00 02 02 db b3 73 00 ee 25 e3 01 00 00 28
data 0x2848  4b 00 19 df f2 28 01 00 f2 28 0b 29 03 00 02 02 db 2b 60 00 ee 25 fe 01 00 00 28 66 00 19 df f2 28 01
data 0x286A  00 f2 28 0b 29 03 00 02 02 db 20 4d 00 ee 25 19 02 00 00 28 81 00 19 df f2 28 01 00 f2 28 0b 29 03 00
data 0x288C  02 02 db 98 39 00 ee 25 34 02 00 00 28 9c 00 19 df f2 28 01 00 f2 28 0b 29 03 00 02 02 dc 8d 26 00 ee
data 0x28AE  25 4f 02 00 00 28 b7 00 19 df f2 28 01 00 f2 28 0b 29 03 00 02 02 db 05 13 00 ee 25 6a 02 00 00 28 d2
data 0x28D0  00 19 df f2 28 01 00 f2 28 0b 29 03 00 02 02 00 00 00 00 ee 25 93 02 00 00 29 28 33 22 11 81 61 32 00
data 0x28F2  cb 27 00 00 9d 61 9d 61 f2 28 9a 61 02 00 00 00 00 00 f2 28 00 00 02 ee 25 72 65 63 75 72 73 69 00 03
data 0x2914  00 02 00 db be 86 00 00 00 00 00 00 81 61 04 01 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5
data 0x2936  a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5
data 0x2958  a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5
data 0x297A  a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5
data 0x299C  a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5
data 0x29BE  a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5
```

6 Opgave 6

6.1

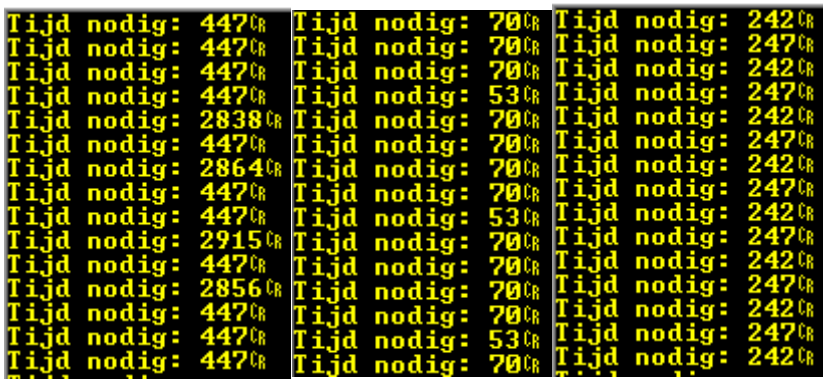
Volatile verteld de compiler dat de waarde van de variabele op elk moment kan veranderen zonder dat er actie wordt ondernomen door de code die de compiler dichtbij vindt. (Url)

6.2

Doordat de waarde tegelijk verandert wordt wanneer hij uitgelezen wordt, dit zorgt voor foute waarden bij het uitlezen.

6.3

Semaphoor: Atomair: Queue:



Op vlak van snelheid is Atomair het snelst dan Queue en als traagste Semaphoor. Het probleem met Atomair is dat er geen task switching mogelijk is tijdens de sectie dit zorgt ervoor dat het real time gedrag wordt verstoord.

7 Opgave 7

8 Opgave 8

Eerst beginnen met receive functie.

```
USART.CTRLA=0b00010000; // Pagina 259

static int stdio_getchar(FILE *stream)
{
	char Data;
	xQueueReceive(FifoReceive, &Data, portMAX_DELAY); // Data inlezen
	return Data;
}

ISR(USART_RXC_vect) { // Receive Interrupt Vector
	xQueueSendFromISR(FifoReceive, &USART.DATA, NULL); // Data doorsturen
}
```

8.1

De terminal task vraagt nu niet meer 99% processor tijd.

Transmit functie:

```
USART.CTRLA=0b00010100; // Pagina 259

static int stdio_putchar(char c, FILE * stream)
{
	xQueueSend(FifoTransmit, &c, portMAX_DELAY); // Databyte op FIFO zetten
	if (USART.STATUS & 0b00100000) { // Kijken of UART data aan het verzenden is
		xQueueReceive(FifoTransmit, &USART.DATA, portMAX_DELAY); // Databyte van
FIFO afhalen en rechtsreeks naar USART DATA register schrijven
	}
}

ISR(USART_TXC_vect) {
	xQueueReceiveFromISR(FifoTransmit, &USART.DATA, NULL); // Databyte van FIFO
afhalen en naar USART.DATA register schrijven
}
```

9 Opgave 9

Vind je in FreeRTOSconfig.h

tick rate = 1000Hz = 1ms

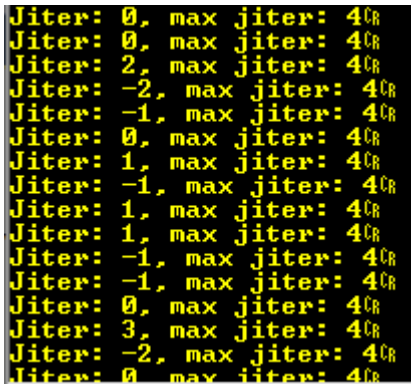
cpu clock = 32MHz

delay = 100ms

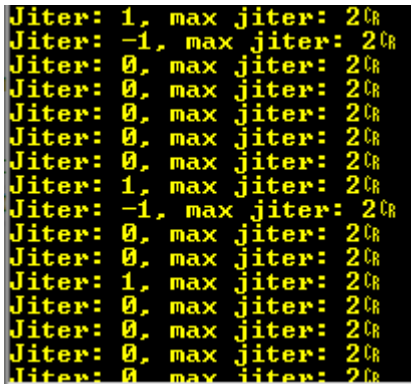
t_interval = cpu clock / tick rate * delay

Voor de max jiter kan je deze best voor een paar cycli verwerpen zodat deze kan balanceren.

9.1



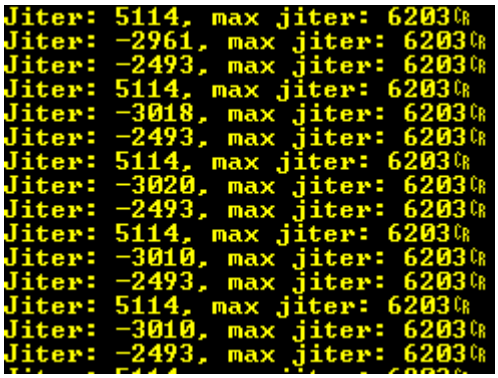
9.2



9.3

Dit komt doordat het looplicht een hogere prioriteit heeft en hier continu naar terug wordt geschakeld waardoor de data van de LEDs niet continu is en de timing constraints niet gerespecteerd worden.

9.4



9.5

Bij D zien we veel grotere waardes van jitter dit komt doordat we hier de interrupts uitzetten. Bij A en D zien we af en toe negatieve waarden wat betekend dat er te vroeg van task geswitched wordt. A en B zijn redelijk gelijklopend alleen is bij B de jitter iets kleiner.