Universiteit Antwerpen

Academiejaar 2020-2021

Faculteit Toegepaste Ingenieurswetenschappen

**6-Embedded Systems**

**Verslag Ontwikkelen met EFM32 MCU’s**

**Tijs Van Alphen**

**Bachelor of science in de   
industriële wetenschappen:**

**Elektronica ICT**



# Inhoudsopgave

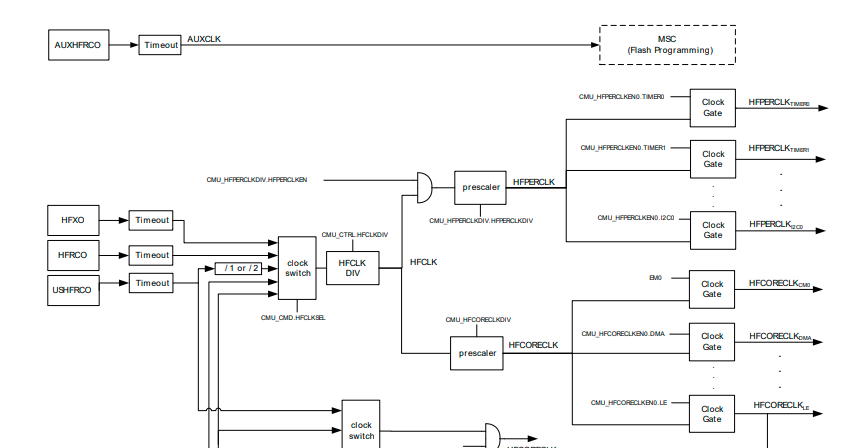
[0 Inhoudsopgave 2](#_Toc64280457)

[1 Oefening 1 Timer gebruiken 3](#_Toc64280458)

# Oefening 1 Timer gebruiken

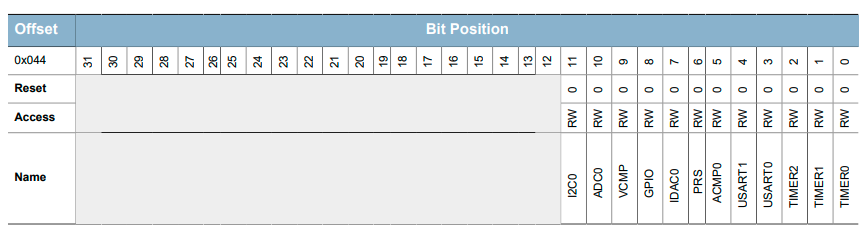
## Enable timer

Om te timer te gebruiken moeten we hem eerste enablen, want alle klokken staan standaard gedisabled. We doen dit om door op zoek te gaan in de CMU naar TIMER0.



We zien uit het schema dat we deze kunnen terugvinden in de HFPERCLKEN0 van de CMU.

De HFPERCLK is de High Frequency Peripheral Clock en alle peripherals zijn aangedreven door de deze klok. De peripherals staan clock gated als ze niet worden gebruikt. Om ze dus te gebruiken moeten we overeenkomende bit van de timer veranderen in de HPERCLKEN0 (HFPERCLK Enable0) register.



We zien dat de TIMER0 op bit 0 staat. We kunnen deze enablen met:



We zien hier dat we de -> operator gebruiken, hiermee kunnen we HFPERCLKEN0 bereiken in de CMU struct. Daarna kunnen we een short hand or operation doen met de laatste bit op 1 gezet.

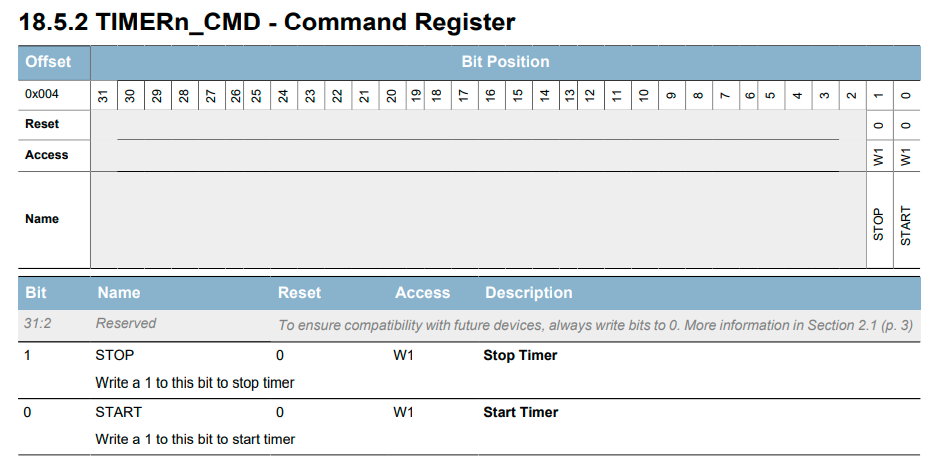
Een andere manier om dit te noteren is op de volgende manier:



De CMU\_HFPERCLKEN0\_TIMER0 is een variabele die hetzelfde is als het binair getal van hiervoor.

## Start timer

Voor het starten van de timer moeten we de startbit van het TIMER0\_CMD op 1 zetten.



Uit de datasheet is te halen dat bits 31:2 op 0 moeten zetten, stop zetten we uiteraard ook op 0 en de Start op 1.

We kunnen dit weer gewoon doen door ofwel een 1 op de laatste bit te zetten of er is ook hier weer een variabele die we kunnen gebruiken.



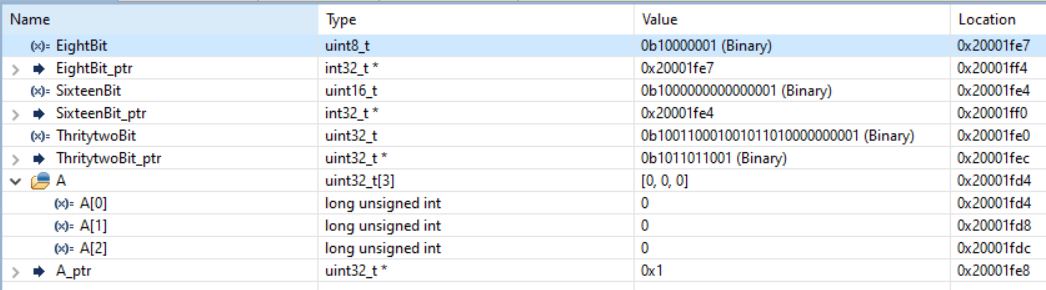
## Wait for threshold

We kunnen gewoon uit de timer de count halen op volgende manier.



## Variabelen

Na het creëren van de variabelen zien we ze ook opduiken in het Variabels Window in de Debugger.

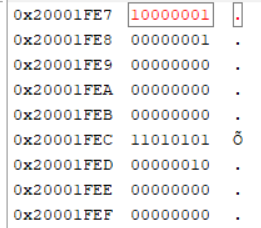


In het begin hebben de variabele nog random values, maar deze worden correct vanaf de debugger over de variabele declaratie gaat. Zo krijgen ook de pointers als value het adres van de variabelen.

### Adresgrootte

We kijken in het Memory venster hoeveel RAM er wordt gealloceerd per variabelen.

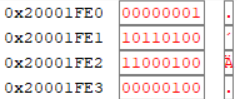
Wanneer we het programma debuggen wordt er getoond welke bit er worden geschreven in de RAM. Hier wordt de 8 bit variabele toegewezen, en we zien dat er 8 bit worden voorzien.



In deze screenshot wordt de 16 bit integer in het geheugen geplaatst. We zien duidelijk dat er hier 2 opeenvolgende bytes voor worden gebruikt.



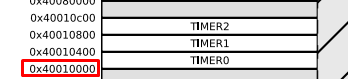
Ook hier bij de 32 bit variabele worden er 4 byte voor geschreven. Ik heb ok telkens gezorgd dat het getal in de variabele



Ik heb ook een keer getest door een groter getal in een variabel te steken dan dat er ruimte voor is. Bijvoorbeeld het getal groter dan 255 in een 8 bit integer te steken (). En dan zie je dat er toch enkel 8 bits in de RAM zullen worden aangepast.

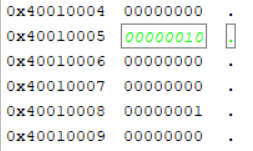
## Timer rechtstreeks starten/stoppen

Hiervoor moeten we eerst in de datasheet bepaalde adressen gaan opzoeken.

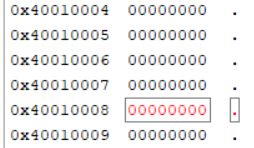


We starten met het nemen van de het adres van TIMER0 en tellen daar de offset van 0x004 bij voor het TIMER0\_CMD register te bekomen. Wanneer we dit aanpassen zien we de value op memoryaddress met offset 0x008 veranderen. Dit komt omdat op dit adres het status register staat van de timer. De value van deze register is 1 bij een running timer en 0 als timer is gestopt.

Op dit ogenblijk is de timer aan het lopen door de voorlaatste bit op 1 te zetten zal de timer stoppen (zie TIMERn\_CMD register).



Wanneer we dus nu op enter duwen zal de value op 0x40010008 op 0 springen.



## Frequentie bekijken

Voor het nagaan van de frequentie heb ik verschillende methoden geprobeerd. Het grootste probleem dat ik ondervond is dat je geen gelijkheid kunt doen in je if statement omdat de counter van de timer nooit perfect op het getal uitkomt.

Waar ik eerst mee ben begonnen is de limiet van de counter te veranderen. Deze is een 16 bit getal, maar is geen makkelijk getal om mee te werken. Om deze reden heb ik de Counter Top Value Register aangepast zodat deze maar tot 50.000 telt.



### Leds aansturen

Eerst en vooral moet de GPIO clock worden ge’enabled voor de leds te laten werken.



Daarna ben ik gaan opzoeken op welke pinnen de leds zitten en heb ik een kleine functie geschreven om ze makkelijk aan of uit te zetten.

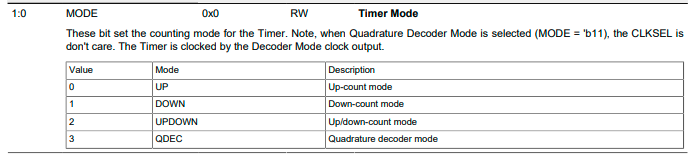
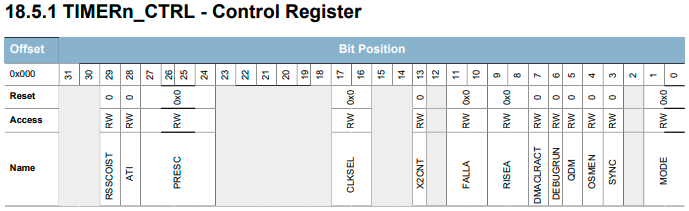


Het eerste keer proberen heb ik een boolean gemaakt die elke keer van status wissels als de klok terug op 0 komt. Het enige probleem is dat zoals ik hiervoor zei het niet mogelijk is om in een if statement te vergelijken met een enkel getal. Daarom heb ik genomen voor counter<10.



Het probleem dat ik hier ondervond is dat wanneer de timer meerdere keren onder 10 zullen de leds direct al wisselen van aan naar uit, en niet pas wanneer de counter rond is geteld.

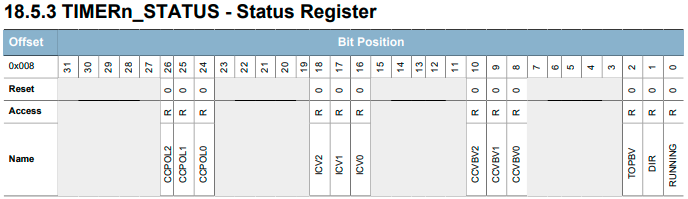
Een oplossing hiervoor heb ik gevonden door in de reference manuals te gaan zoeken. Zo ben ik er achter gekomen dat je de timer ook op en af kunt laten tellen. De Up/Down-count zal tellen tot aan het maximum dat is ingesteld en dan terug aftellen tot 0 enzovoort. We kunne dit aanpassen in de TIMERn\_CTRL register.

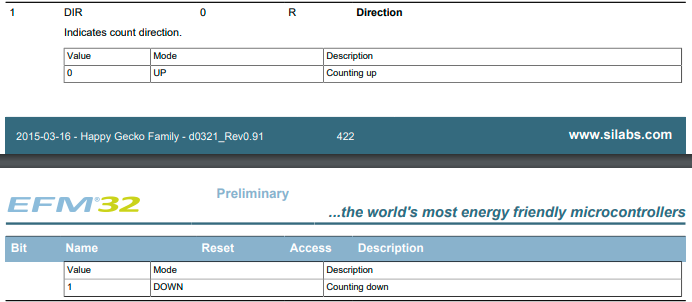
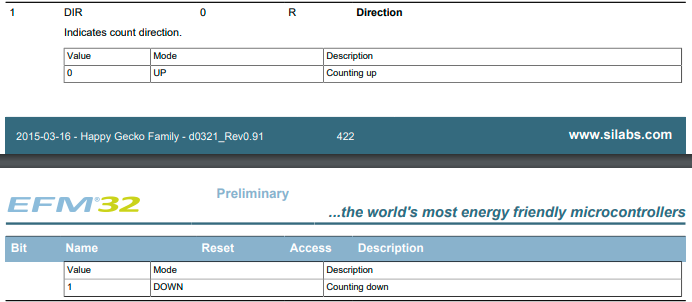


We stellen dus in dat de mode 2 is.



Verder kunnen we nu status register nagaan in welke richting de counter aan het tellen is.

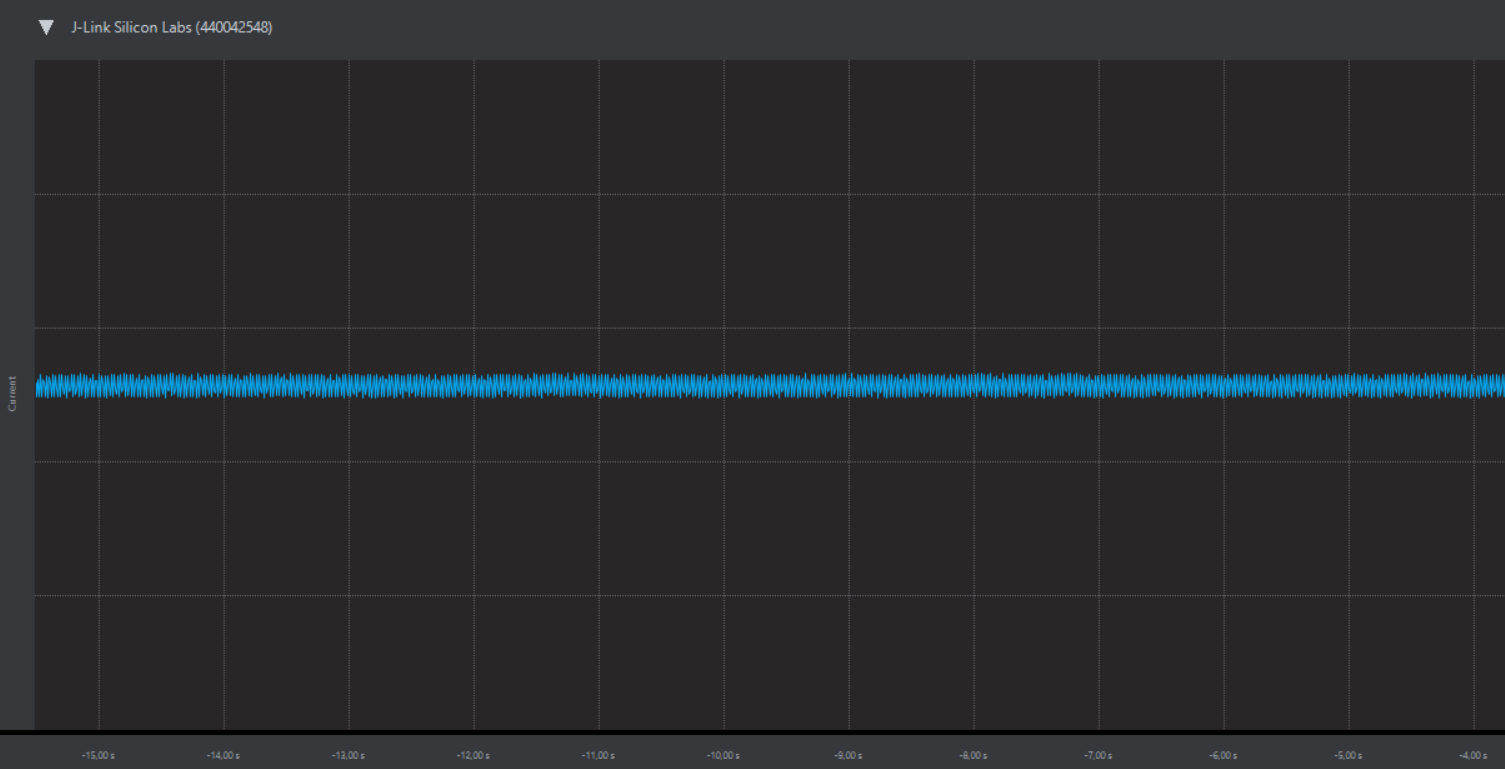




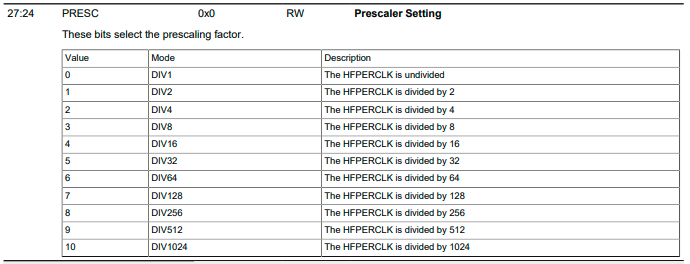
Hiermee kunnen we dus in een if statement uithalen in welke richting de counter aan het tellen is.



Als we dit met het Energy Profile controleren komen we de onderstaande afbeelding uit. We zien dat de klok nog veel te snel gaat en de leds daardoor niet echt uitgaan, we zullen de klok dus trager laten lopen door de prescaler aan te passen.

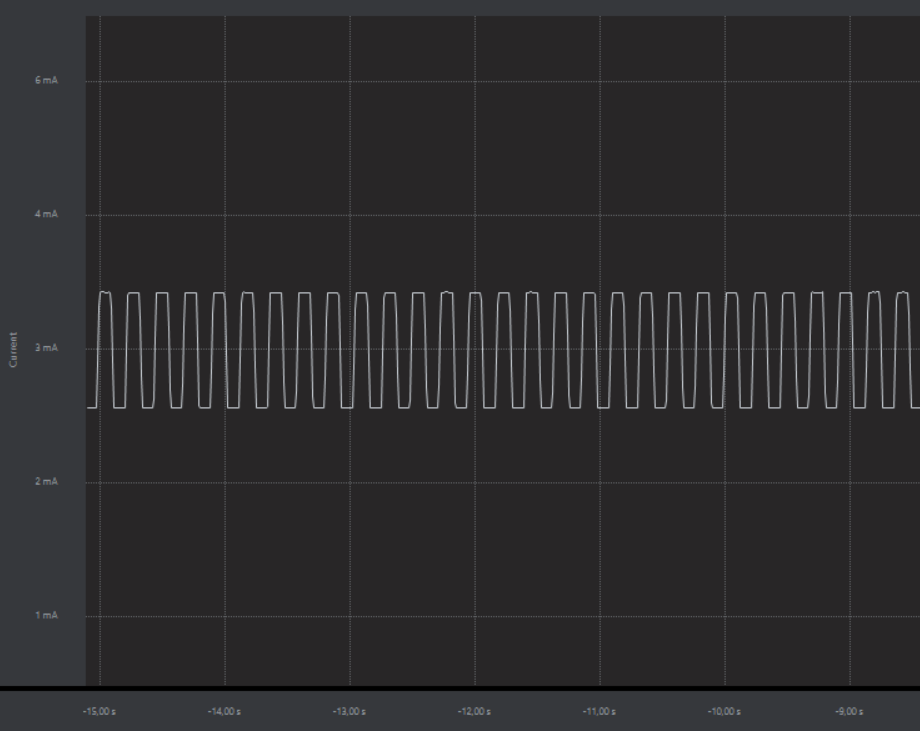


Een aantal afbeeldingen terug zien we dat de prescaler kan worden aangepast in het TIMERN\_CTRL register. De keuze is als volgt:

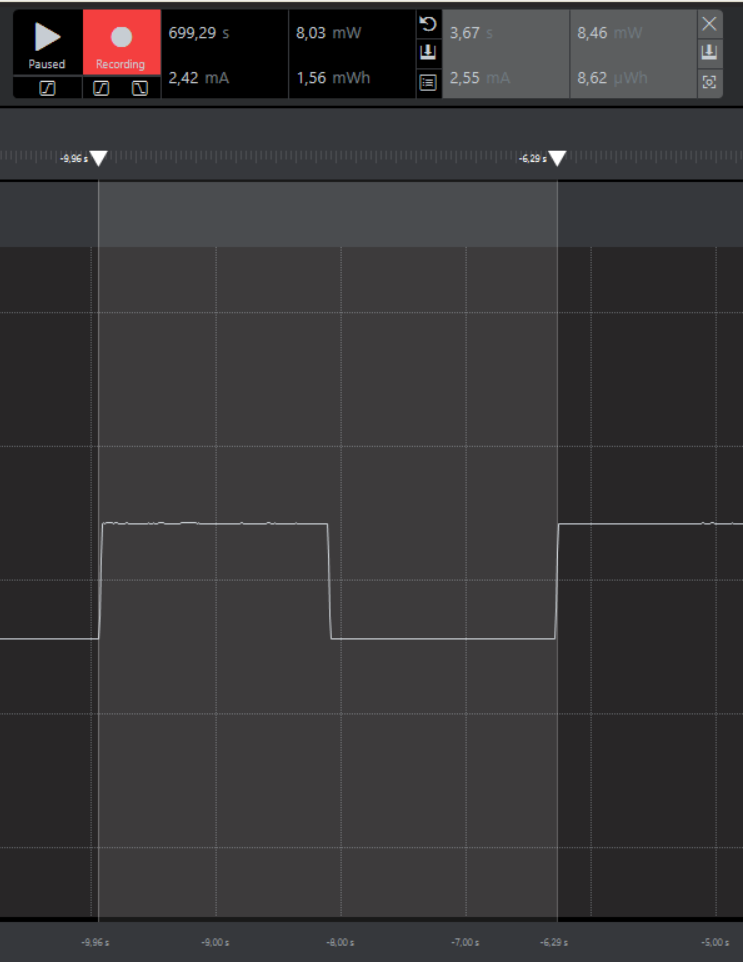


We zetten bijvoorbeeld de value op 5. We doen dit door het gehele prescale veld op 0 te plaatsen en dat hier 5 in te vullen.





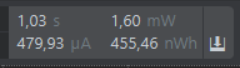
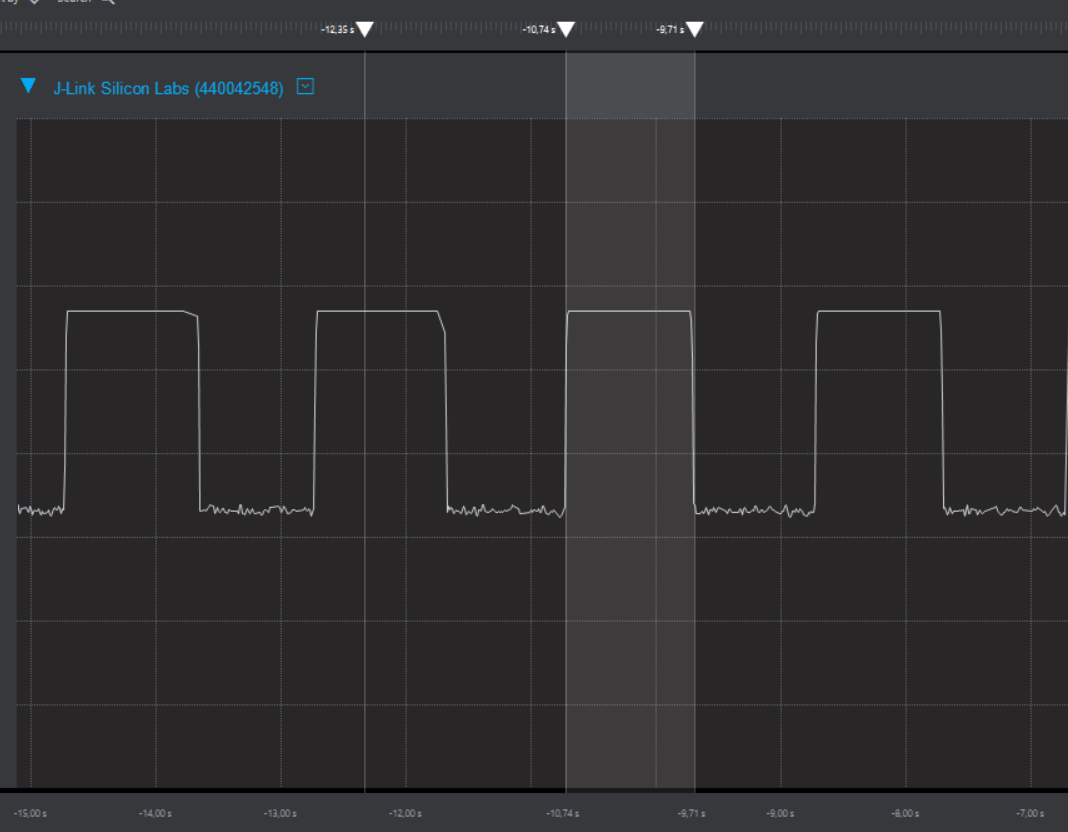
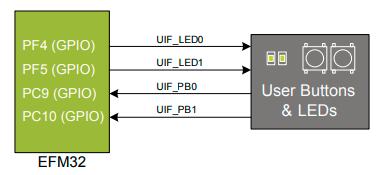
### Berekening

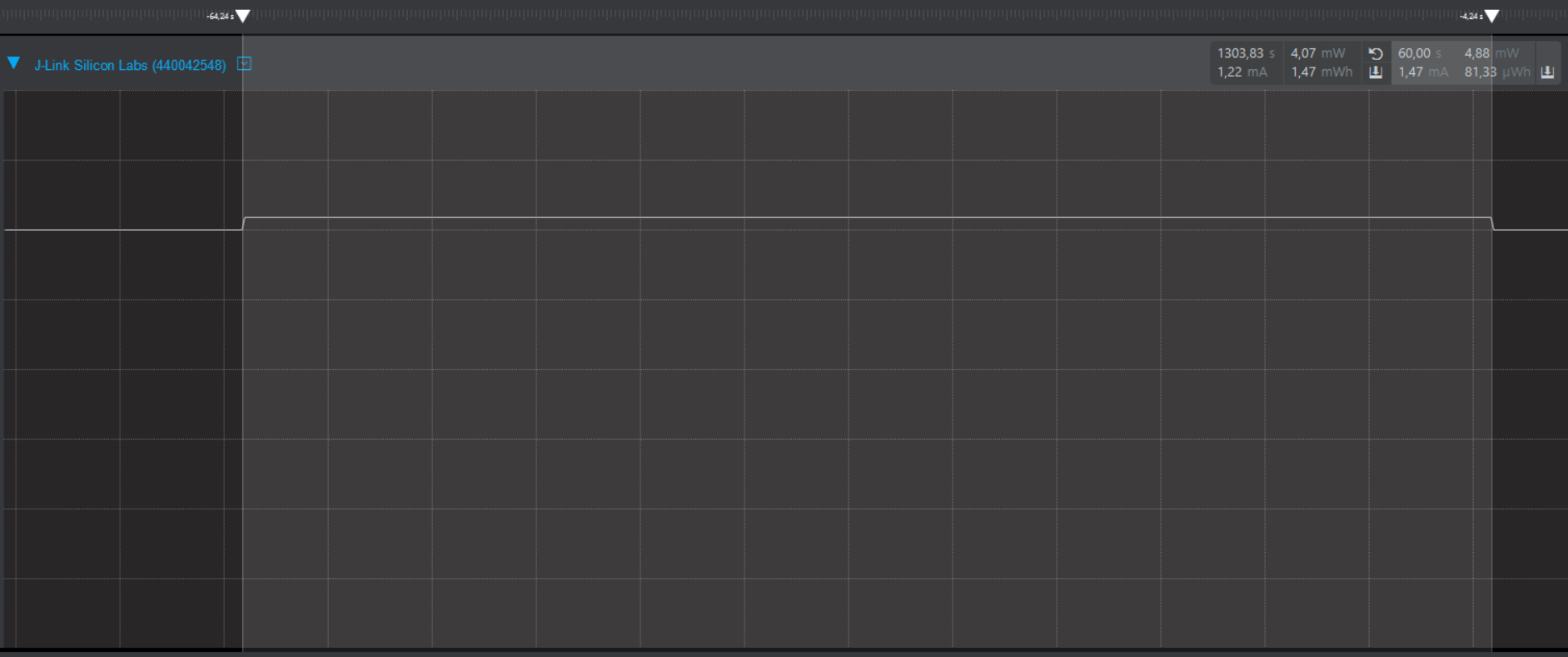
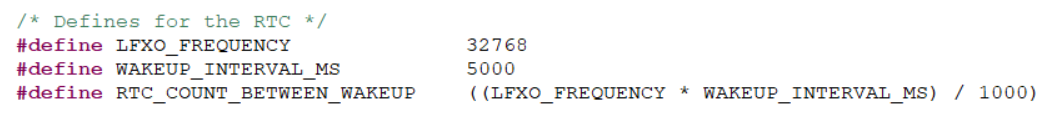
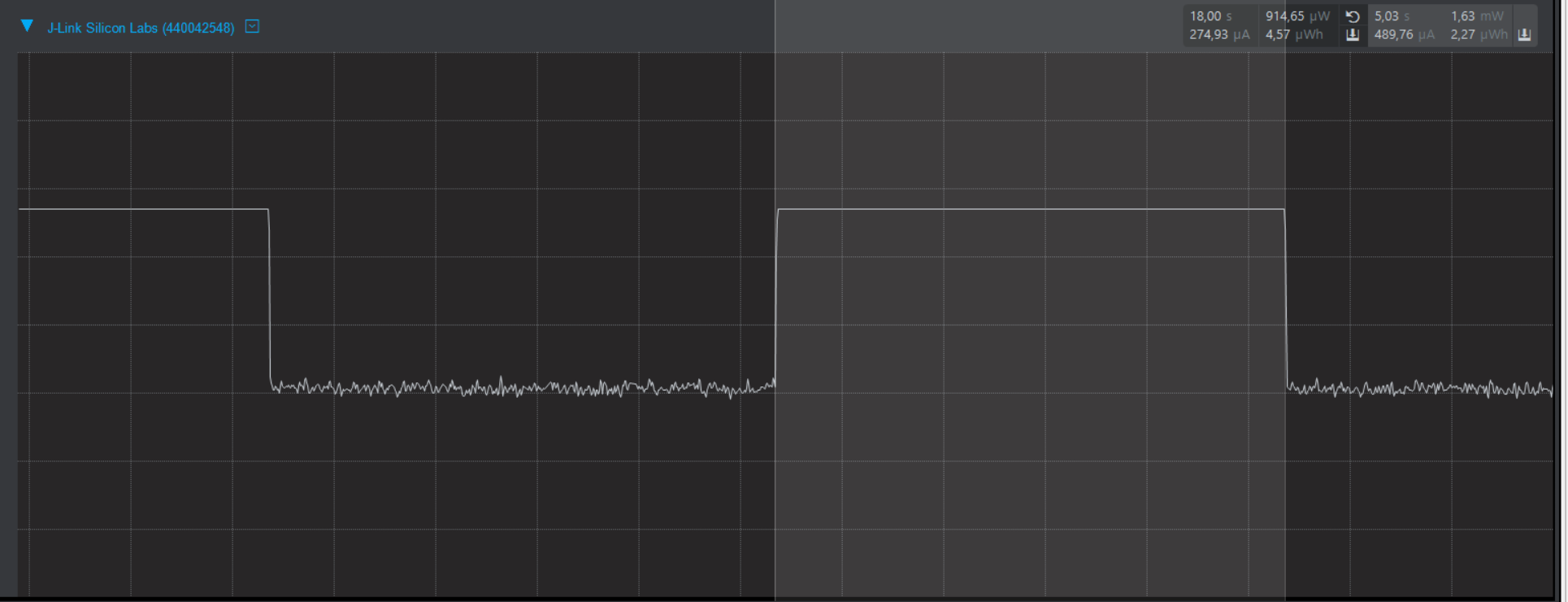


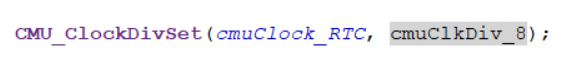
Op de afbeelding is de prescale gezet op DIV512. We zien dat de tijd voor een cycle gelijk is aan . We weten dat een cycle bestaat uit de klok die tot het maximum telt (50.000) en dan terug tot 0.

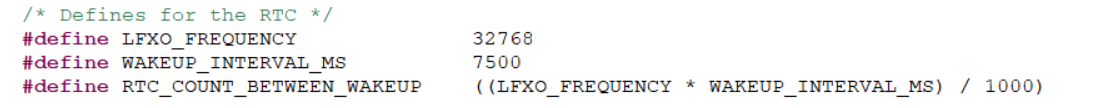
klokflanken

# Oefening 2

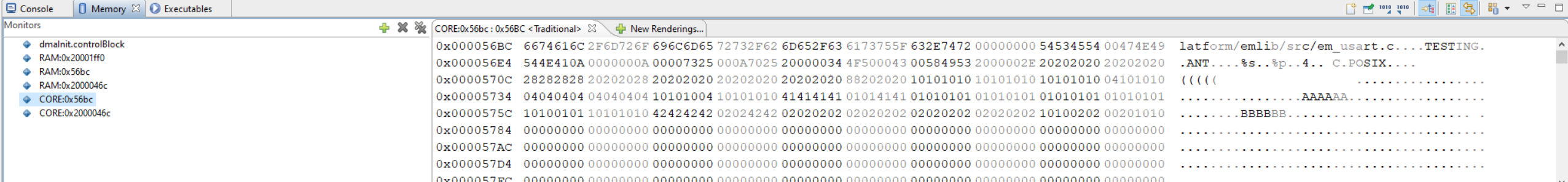




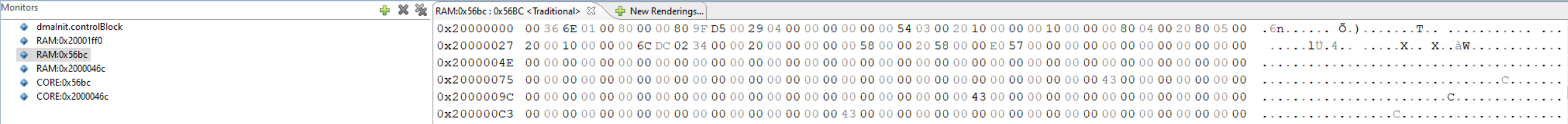




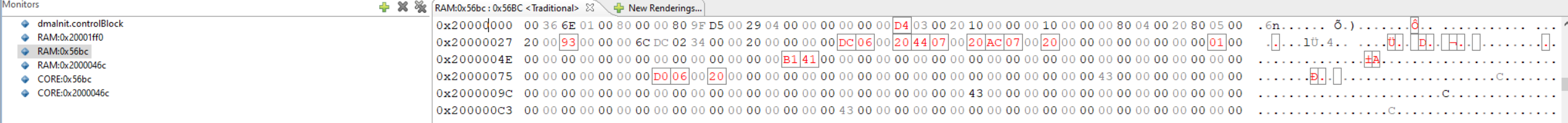
# 2.4



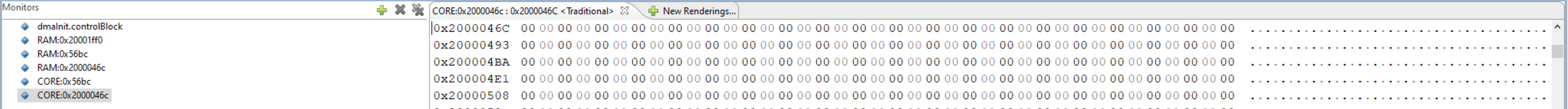
Flash memory: flashData voor en na DMA wijzigt niet



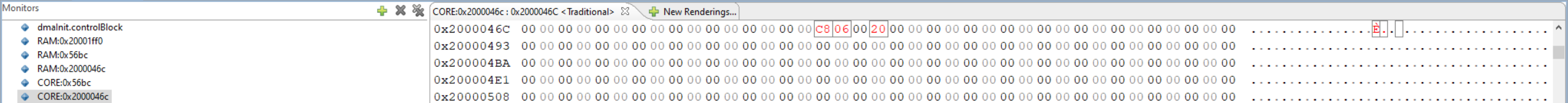
RAM memory: flashData voor



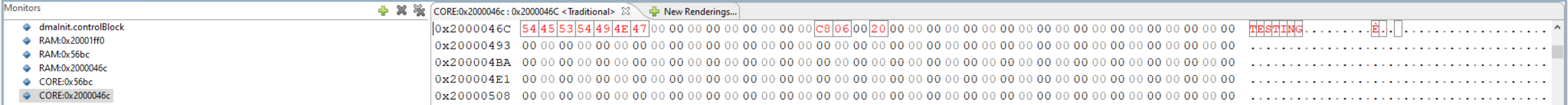
RAM: flashData na DMA



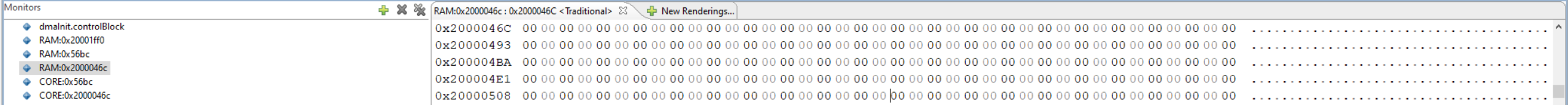
Flash mem: RAMbuffer voor DMA



Bij enablen DMA\_clock

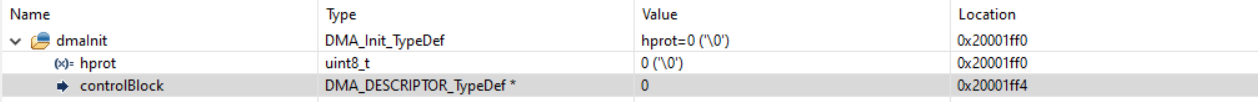


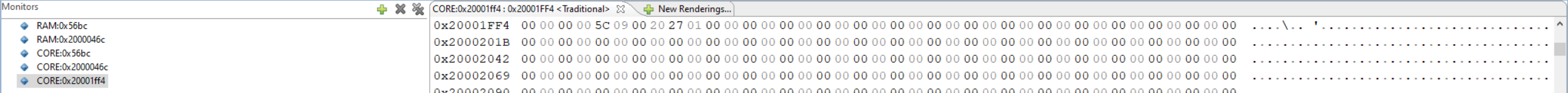
Na DMA



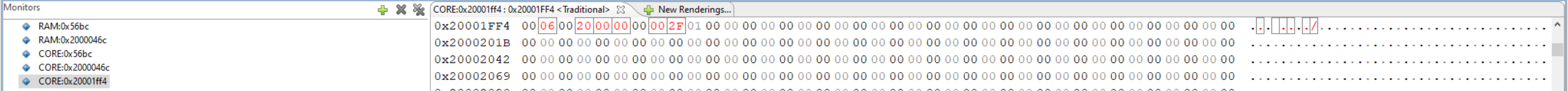
RAM: rambuffer voor DMA

RAM: ramBuffer na DMA gevraagd zich volledig hetzelfde als in het flashmemory

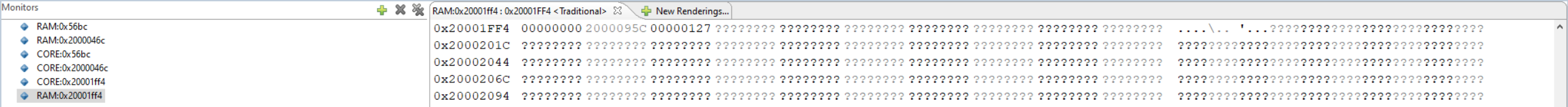




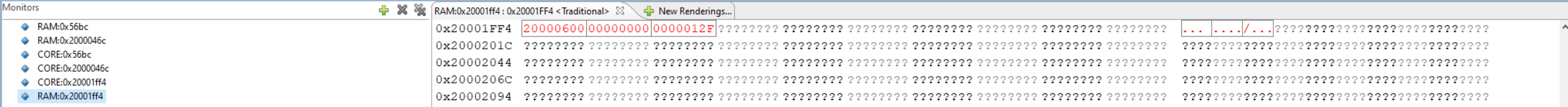
Flashmem: controlBlock voor DMA



Na DMA



RAM: controlBlock voor DMA



Na DMA: 00000012F wijzigde bij CHIP\_Init