ELT-ESE-3 DSBL

Digitale Signaalbewerking practicum

HAN Elektrotechniek/Embedded Systems

ir drs E.J Boks

Opdracht 5 : implementeer het FIR filter uit opdracht 4 op het practicumboard

Doel:

Implementeer het FIR filter uit opdracht 4 in een Cortex-M4F (de STM32F412ZGT) microcontroller. Het practicumboard bevat deze microcontroller, alsmede een 24-bit ADC (TI ADS131A02) als een 16-bit DAC (MAX5136) converter, en is ideaal om een filter te implementeren en te testen .

Tijd:

2 weken.

Benodigde zaken:

- Werkstation in B1.29/B1.33.
- Het Lynn & Fürst theorieboek .
- De algemene instrukties voor het practicum moet je hebben doorgelezen.
- STM32 RGT+DSB Discovery board, af te halen bij Henk Schepers.
- Lees de beschrijving van het STM32F412G-Discovery board.
- Lees de datasheet voor de TI ADS131A02 24 bit ADC.
- Lees de datasheet voor de Maxim MAX5136 16 bit DAC.
- Geschreven en geteste klassen uit Opdracht 1.
- Een werkende applikatie uit Opdracht 4.

Instellingen

Stel het te implementeren filter in met de volgende specificaties :

- Bemonsteringfrequentie: 4 kHz
- doorlaatband: 200-300 Hz
- Fixed-point : 15 bits coefficienten
- Filterorde : 64
- Venster: Hamming

Beschrijving

Genereer een Q15 formaat header bestand met het desktop programma van opdracht 4. Gebruik deze filterwaarden vervolgens in een digitaal filter dat als in de microcontroller wordt toegepast. Het digitale filter is al door jullie ontworpen en getest in opdracht 4, en daarom moet toepassing van het filter in de embedded software zonder al te grote problemen kunnen worden uitgevoerd.
De volgende zaken zijn van belang voor het succesvol uitvoeren van de practicumopdracht:

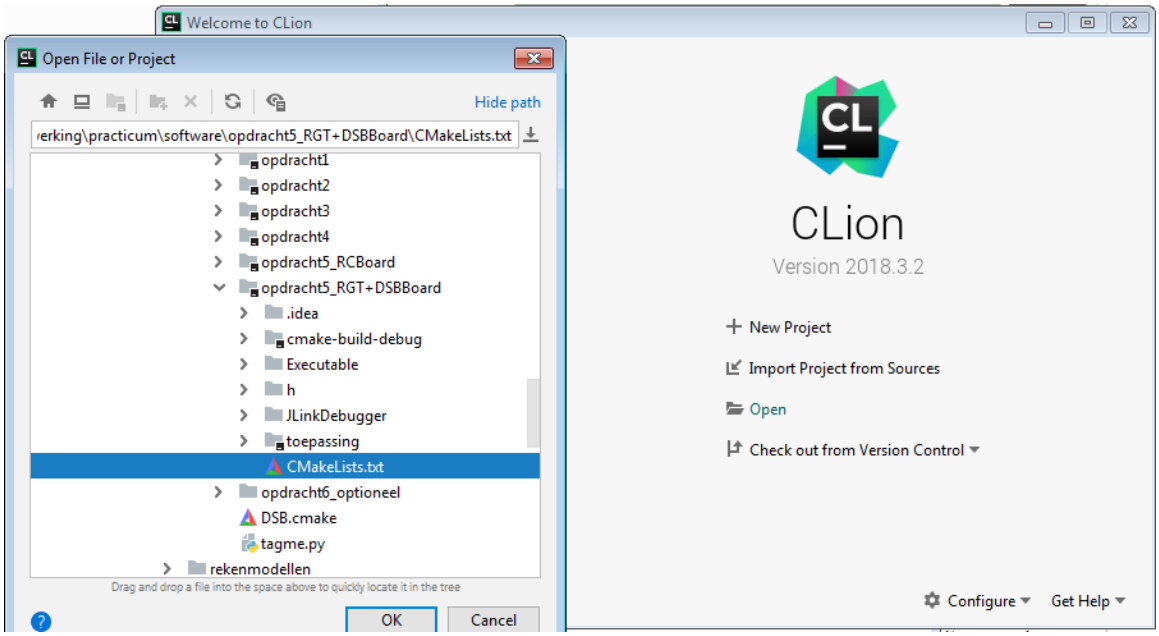
- De practicumcode wordt uitgevoerd vanuit een klasse die `STM32FilterApp` heet.

- De microprocessor moet met de juiste bemonsteringfrequentie worden ingesteld. De driver van de ADC bevat hier voor de juiste funktie. De ADC driver heet ADS131A02 en de implementatie in STM32FilterApp heet `ads131a02.` De bemonsteringfrequentie wordt gezet in met de `zetSampFreq()` funktie. Voor een beschrijving van de parameters naar deze funktie, kijk in de ADS131A02 pdf dokumentatie.
- De DAC moet worden opgestart. Zie de MAX5136 klasse voor een driver van dit device. Voor een beschrijving van de parameters van deze driver, kijk in de MAX5136 pdf dokumentatie.
- Let op de kanaalkeuzes voor zowel de ADC als de DAC. De kanalen moeten worden ingesteld (kijk in de DAC en ADC klassen). De goede kanalen staan als alias in student.h klaar:

```
static constexpr auto DSB_ADC_Channel=ADS131A02::Kanaal::K2;
static constexpr auto DSB_DAC_Channel=MAX5136::Kanaal::K2;
```

De software moet er voor zorgen dat met deze bemonsteringfrequentie een signaal wordt ingelezen in de ADC. Daarna moet het ingelezen signaal in het FIR filter worden bewerkt. De gefilterde waarde moet vervolgens met behulp van de ingebouwde DAC weer als analoge waarde worden aangeboden. Ontwerp op papier een stroomdiagram en implementeerd dit vervolgens in de `hoofdlus()` funktie.

Het projekt wordt met Jetbrains CLion uitgevoerd, ook op een Windows machine :



Uitvoering van software ontwikkeling met CLion

Clion is een moderne en veelzijdige C/C++ ontwikkelomgeving waar jullie al bij het vak ECS mee kennis gemaakt hebben.

Het programmeren en debuggen van de embedded software gebeurt met behulp van Segger Ozone. Ook deze tool zijn jullie al tijdens het ECS practicum tegengekomen.

Opdracht

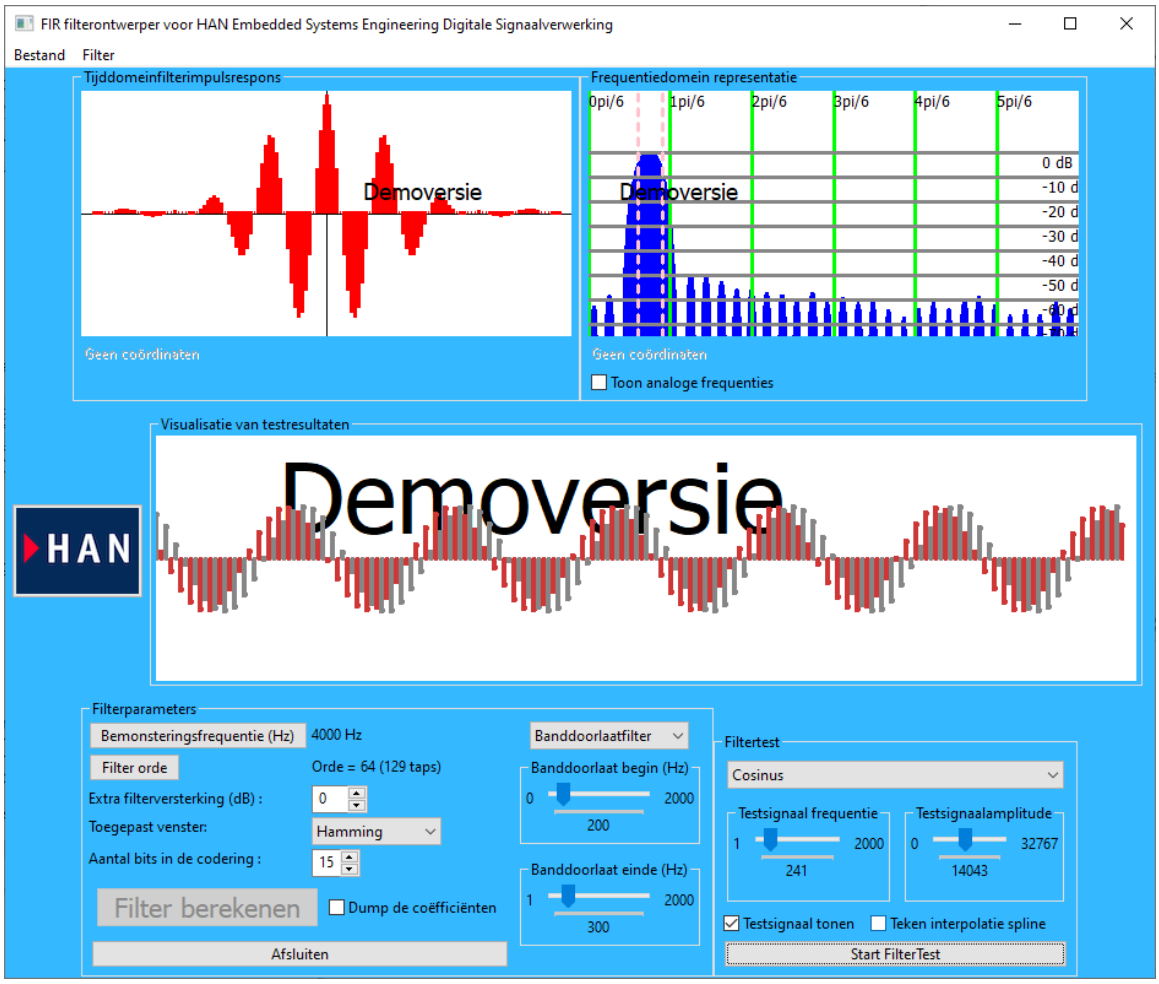
Van jullie wordt verwacht dat je de volgende zaken oplevert:

- Maak het filter in opdracht 4. Exporteer de filter coefficienten als header bestand. Noem het headerbestand met de coefficienten *firfilterexport.h* en sla dit op in het CLion projekt. Zoek zelf een goede plek daar voor en zorg dat de compiler het kan vinden op de opgegeven header paden (tip : de directory “h”).
- Bepaal een bemonsteringfrequentie instelling voor de ADS131A02 die de gevraagde instelling mogelijk maakt.
- Werk de C++ klasse STM32FilterApp verder uit. Deze klasse implementeert het fixed-point FIR filter en alle instellingen voor de ADC en de DAC. Verder wordt in de klasse het lezen van de ADC, het filteren en het aansturen van de DAC uitgevoerd.
    - De FIR Filter klasse moet dezelfde zijn als het door jullie geschreven filter in opdracht 4, met als input een set coefficienten gegenereerd met het desktop programma van

opdracht 4.

- Gebruik een oneindige lus voor het filteren.
- Implementeer alles in de funktie STM32FilterApp::hoofdlus() .

- Laad het programma in het RGT+DSB Board met Segger Ozone.
- Test het programma met een funktiegenerator en een oscilloscoop. De funktiegenerator wordt aangesloten op de ingang BNC connector, de skoop op de uitgang BNC connector.

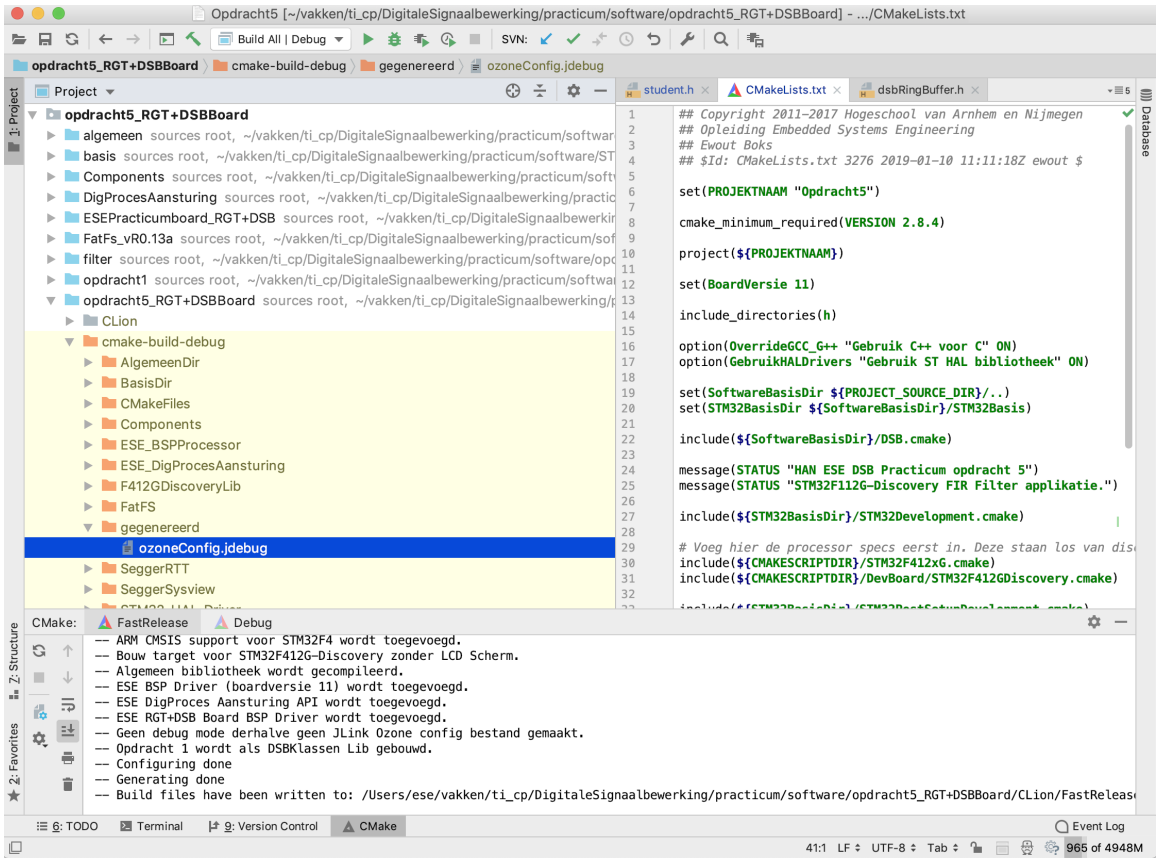


Uitvoering met Ozone

Debugging van de Cortex microcontroller wordt net als bij ECSL gedaan met Segger's Ozone debugger.

Voor meer informatie over Ozone, kijk voor een introduktie naar dit Ozone filmpje . Daarnaast wordt meer uitgelegd in deze en deze behandeling.

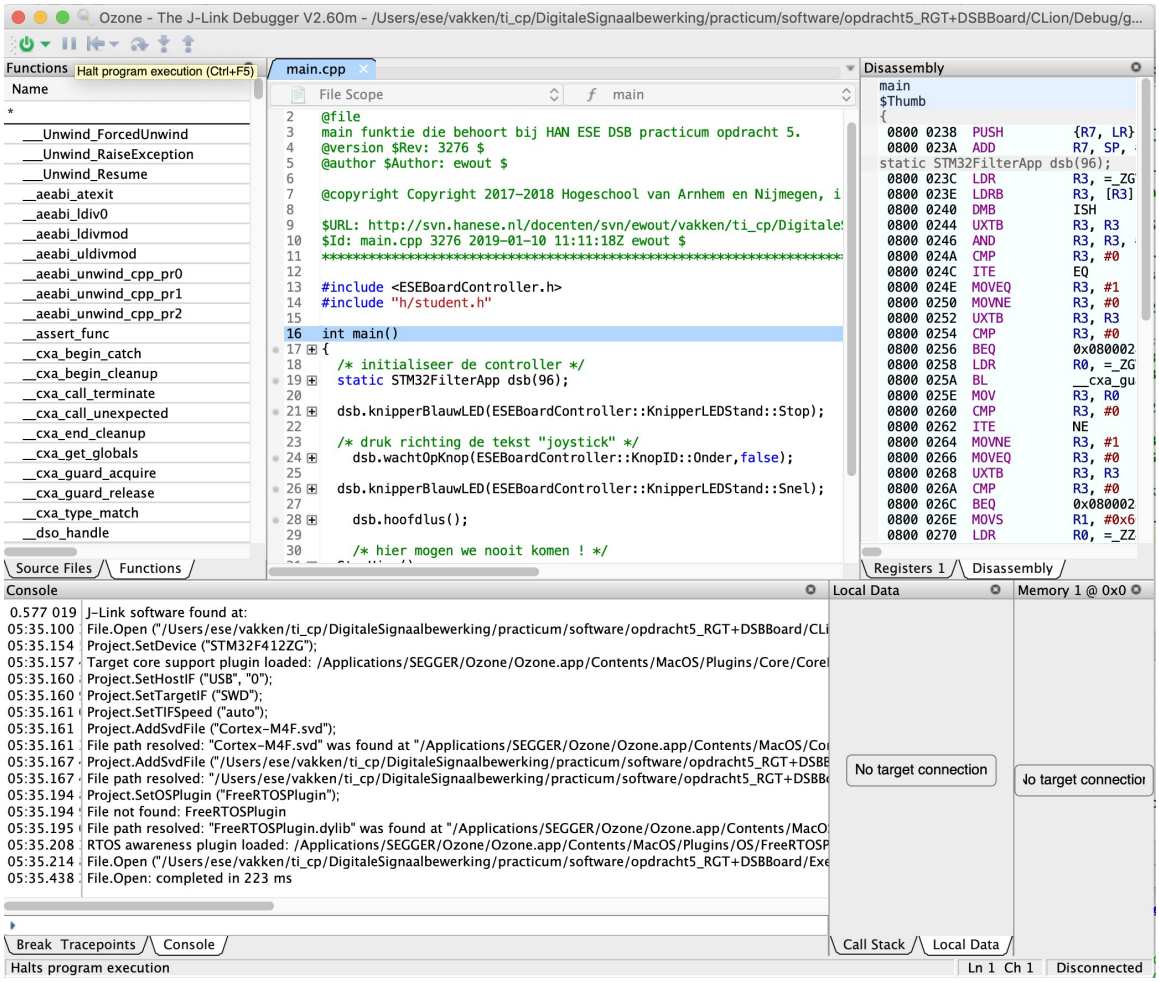
Ozone wordt net als bij ECSL gestart met een in CLion gegenereerd bestand “ozoneConfig.jdebug” :



➔ Open dit bestand “ozoneConfig.jdebug” in Ozone als een bestaand projekt.

➔ Maak verbinding met het doel met een JLink of de ingesloten STLink (in de JLink-modus).

Debugging met Ozone ziet er dan zo uit :



Oplevering:

- Verzorg een demonstratie van het embedded filtersysteem met behulp van funktiegenerator en oscilloscoop (een set aanwezig in lokaal bij docent). Zet de funktiegenerator op **1,2V PP** en de offset op **700 mV** om vervorming te voorkomen.
- Schrijf een klein verslag, met daarin een toelichting op de broncode van embedded applicatie.

Lever nu na goedkeuring het gehele practicumwerk in!