ELT-ESE-3 DSBL

Digital Signal processing practical work

HAN Electrical Engineering/Embedded Systems

E.J Boks MSEE MA

Assignment 5: implement the FIR filter from assignment 4 on the practical work board

Goal:

To implement the FIR filter from assignment 4 in a Cortex-M4F (the STM32F412ZGT) microcontroller. The practical work board contains this microcontroller, a 24-bit ADC (TI ADS131A02) and a 16-bit DAC (MAX5136) converter, and is ideal for implementing and testing a filter.

Time:

2 weeks.

Requirements:

- Workstation in Room B1.29 / B1.33.
- Lynn & Fürst theory book.
- You must have read the general instructions for the practical work assignment.
- STM32 RGT + DSB Discovery board, to be picked up at Henk Schepers.
- Read the description of the STM32F412G-Discovery board.
- Read the datasheet for the TI ADS131A02 24 bit ADC.
- Read the datasheet for the Maxim MAX5136 16 bit DAC.
- Written and tested classes from Assignment 1.
- A working application from Assignment 4.

Settings

Set the filter to be implemented with the following specifications:

- Sampling frequency: 4 kHz
- pass band: 200-250 Hz
- Filter order : 64

Description

Generate a Q15 format header file with the desktop program of command 4. Then use these filter values in a digital filter that must be written as an embedded code for the microprocessor.

The following things are important for the successful execution of the practical assignment:

- The lab code is executed from a class called `STM32FilterApp`.
- The microcontroller must be set with the correct sampling frequency. The driver of the ADC contains here for the correct function. The ADC driver is called ADS131A02 and the implementation variable in STM32FilterApp is called `ads131a02`. The sampling frequency is set with the setSamplingFreq() function. For a description of the parameters for this function, see the ADS131A02 pdf documentation.
- The DAC must be started. See the MAX5136 class for a driver of this device. For a description of the parameters of this driver, see the MAX5136 pdf documentation.
- Make certain that the channels for both ADC as DAC have been selected. The channels must be set (see the DAC and ADC classes for set functions). The proper channels have been prepared as aliases in student.h :
  ```
  static constexpr auto DSB_ADC_Channel=ADS131A02::Channel::K2;
  static constexpr auto DSB_DAC_Channel=MAX5136::Channel::K2;
  ```

The software must ensure that a signal is read into the ADC with this sampling frequency. Afterwards, the signal read in must be processed in the FIR filter. The filtered value must then be presented again as an analog value using the built-in DAC. Design a flow chart on paper and then implement it in the hoofdlus() (“main loop()”) function.

The project is carried out with Jetbrains CLion, also on a Windows machine:

Software development with CLion

Clion is a modern and versatile C / C ++ development environment that you have already met during the ECS course.

The programming and debugging of the embedded software is done with the help of Segger Ozone. You have come across this tool also during the ECS practicum.

Task:

You are expected to deliver the following items:

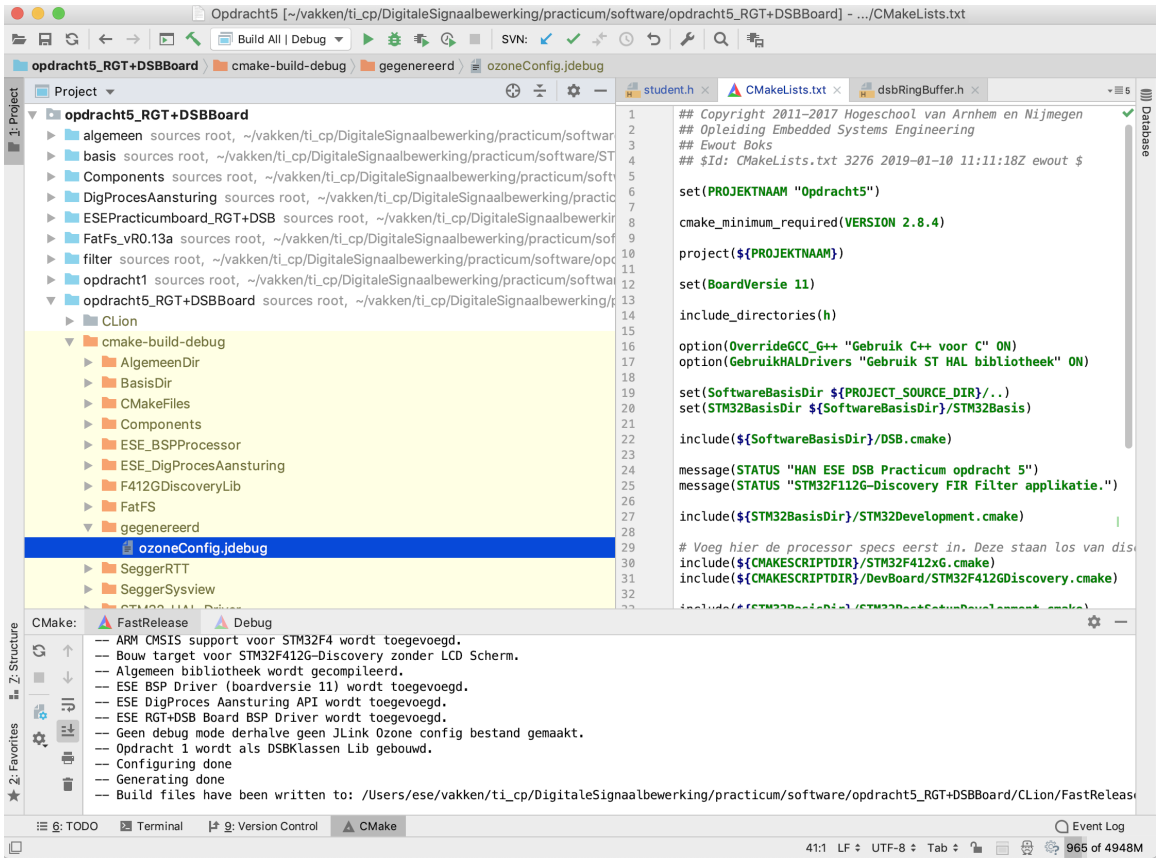
- Produce a filter with the programme from assignment 4. Export the filter coefficients as a header file. Name the header file with the coefficients firfilterexport.h and save it in the CLion project. Please determina a suitable location where the compiler can include the file (tip : the “h” directory).
- Determine a sampling frequency setting for the ADS131A02 that delivers the required sampling frequency.
- Complete the `STM32FilterApp` class. This class implements the fixed-point FIR filter and all settings for the ADC and DAC. Additionally, in the class reading the ADC, filtering the data and controlling the DAC are executed.
  - The FIR Filter class must be the same as the filter you wrote in assignment 4, with as input a set of coefficients generated with the desktop program of assignment 4.
  - Employ an endless loop for filtering.
  - Implement all functionality in `STM32FilterApp::mainloop()` .
- Load the program into the RGT + DSB Board with Ozone, as explained in the practical information.
- Test the code with a function generator and an oscilloscope. The function generator is to be connected to the BNC input jack, the scope to the output BNC jack.

Execution using Ozone:

Debugging the Cortex microcontroller is done just like with ECSL with Segger's Ozone debugger.

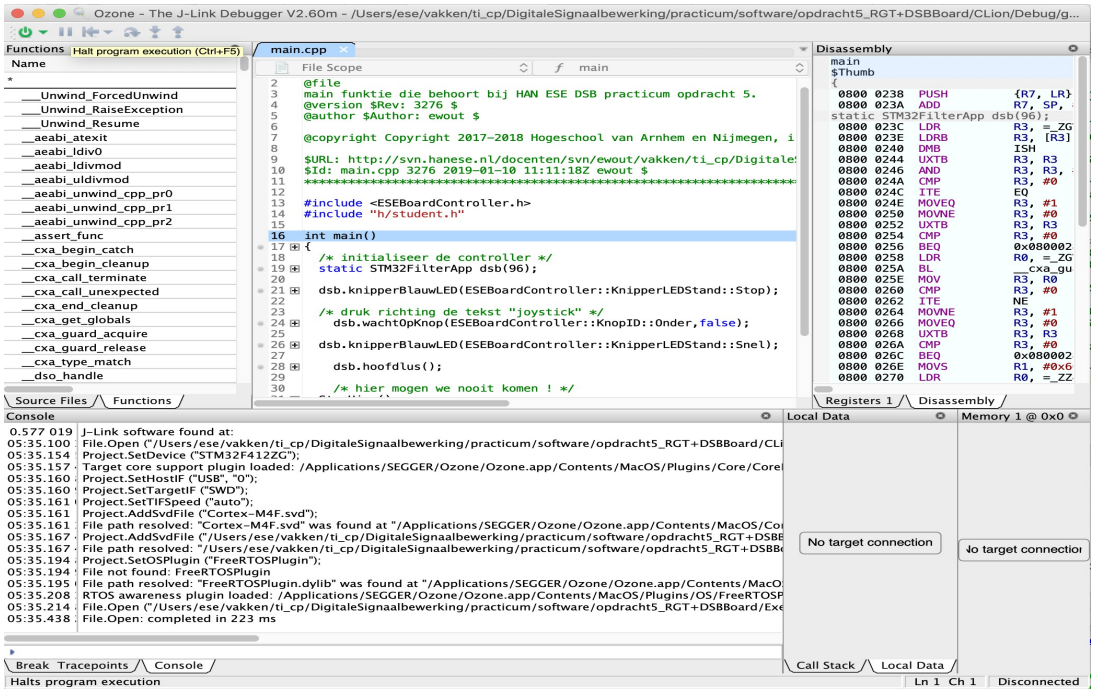
For more information about Ozone, look for an introduction at this Ozone video. In addition, more is explained in this and this video.

Ozone is started with a file "ozoneConfig.jdebug" generated in CLion:



➔ Open this file in Ozone as an existing Ozone project.
➔ Connect to the target with either a JLink or the embedded STLink (in JLink mode).

Debugging with Ozone looks like this ;



Delivery:

- Perform a demonstration of the embedded filter system with the help of a function generator and oscilloscope (a set present in the classroom with the lecturer). Set the function generator to **1.2V PP** and the offset to **700mV** to prevent distortion.
- Write a small report with an explanation of the source code of embedded application.

Now submit the entire practical work after approval!