

Deploying an EMG RMS envelope measurement application to an STM32 Discovery kit using the ASN Filter Designer



Author: Dr. Sanjeev Sarpal

Application note (ASN-AN024)

July 2023 (Rev 4)

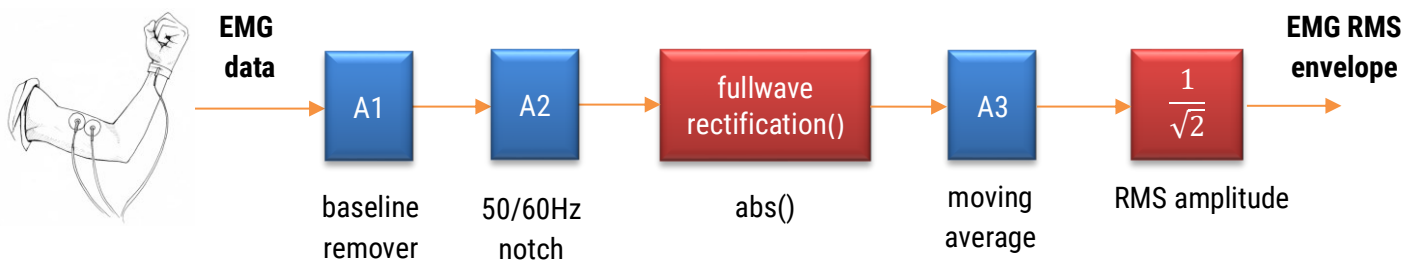
Synopsis

Electromyography (EMG) is an experimental biomedical engineering technique for evaluating and recording the electrical activity produced by skeletal muscles. The EMG is performed using an instrument called an electromyograph, where the electromyograph detects the electrical potential generated by muscle cells when these cells are electrically or neurologically activated. The captured signals can be analysed by signal processing techniques in order to detect medical abnormalities and to analyse the biomechanics of human movement.

This application note demonstrates the design of a suitable EMG filtering chain using the ASN Filter Designer's FilterScript language and its interactive customisation for an RMS (root-mean-square) envelope measurement signal processing application. Details are given on how the developed application may be deployed to an STM32 Discovery kit for evaluation in a real-time embedded application. Additional details are also given about deployment to Matlab and Python for further analysis in popular scientific computing environments.

Introduction

The standard building blocks required for an EMG RMS envelope measurement application are shown below:

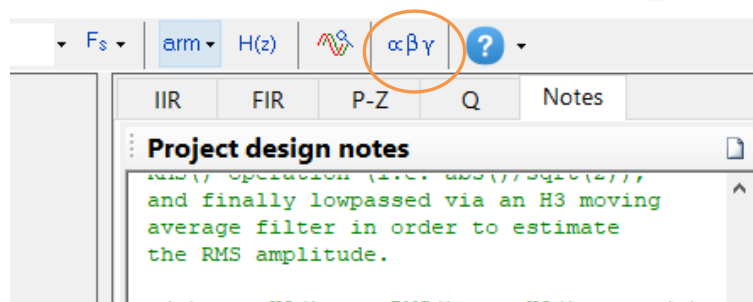


A description of each building block is given overleaf.

Building block	Equation	Description
A1	$\frac{1 - z^{-1}}{(w_c + 1) + (w_c - 1) z^{-1}}$	A baseline removal filter (DC removal) can be realised with a simple first order pole-zero filter. Where, $w_c = \tan(\frac{\pi f_c}{f_s})$ is used to set the highpass filter's bandwidth.
A2	$\frac{1 - 2 \cos w_c z^{-1} + z^{-2}}{1 - 2r \cos w_c z^{-1} + r^2 z^{-2}}$	A notch filter. Where, $w_c = \frac{2\pi f_c}{f_s}$ controls the centre frequency, f_c (50 or 60Hz) of the notch, and r controls the bandwidth of the notch.
Fullwave rectification	abs()	Abs () function.
A3	$1 + z^{-1} + z^{-2} + \dots z^{-M}$	An Mth order moving average filter.

1. Symbolic mathematical filter scripting language (ASN FilterScript)

As seen, there are three filters and two mathematical operations. Combining the A1 and A2 filters into one IIR filter (H2), we can implement the above operation in the ASN Filter Designer with the Symbolic Filter Scripting language.



The scripting language (ASN FilterScript) supports over 82 scientific commands and provides designers with a familiar and powerful programming language, while at the same time allowing them to implement complex symbolic mathematical expressions for their filtering applications.

ASN FilterScript offers the unique and powerful ability to modify parameters on the fly with the so-called *interface variables*, allowing for real-time updates of the resulting frequency response. This has the advantage of allowing the designer to see how the coefficients of the symbolic transfer function expression affect the frequency response and the filter's time domain dynamic performance.

Please refer to the ASN Filter Designer FilterScript [reference guide](#) for a detailed overview.

1.1. ASN FilterScript code

FilterScript's `dcremover()` and `notch()` functions can be used to implement the A1 and A2 filters respectively. Notice that the `dcremover()` function just requires a single cut-off frequency in Hz, and that the `notch()` function requires a centre frequency, `fc` and bandwidth, `BW` in Hz.

The `dcremover()` and `notch()` functions produce two digital filter objects, A1 and A2. As such, these two objects can be merged into a single filter¹ object (Hd) via the `augment()` function. The "symbolic" keyword sets the display format of the resulting digital filter object.

```
A1=dcremover(f1,"symbolic"); // A1 filter
A2=notch(fc,BW,"symbolic"); // A2 filter

Hd=augment(A1,A2,"symbolic"); // merge filters
```

The code snippet shown below demonstrates how to design and merge three filters implementing a highpass and a double notch at 50Hz and 60Hz respectively,

```
A1=dcremover(f1,"symbolic"); // A1 highpass filter
A2=notch(50,BW,"symbolic"); // A2 50Hz filter
A3=notch(60,BW,"symbolic"); // A3 60Hz filter

Hd=augment(A1,A2,"symbolic"); // merge A1 and A2 filters
Hd=augment(Hd,A3,"symbolic"); // merge A3 with Hd filters
```

1.1.1. Interface variables and the H1 filter

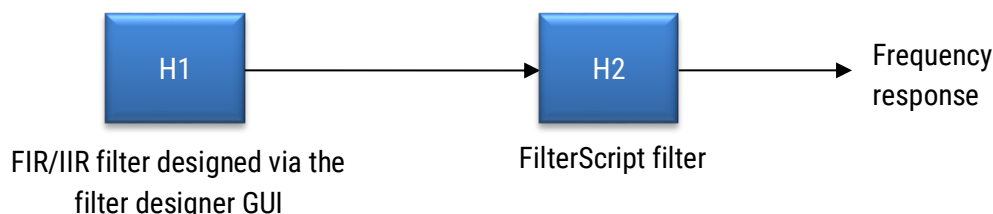
FilterScript allows developers to modify parameters on the fly with the so-called *interface variables*, allowing for real-time updates of the resulting frequency response without modifying and recompiling the script.

An interface variable is defined as a vector expression:

```
interface name = {minimum, maximum, step_size, default_value};
```

where all entries must be real scalar values. Vectors and complex values will not compile.

The `ClearH1` keyword allows you to remove the H1 filter (primary) from the tool's internal filter cascade and just use the H2 (secondary) filter. The relationship of the H1 and H2 filters is shown below:



As seen, the main FIR/IIR filter designed via the filter designer GUI is assigned to the *primary filter*, H1. All poles and zeros defined via the filter script are added to a *secondary filter block*, H2.

¹ `augment()` calls the `conv()` function that is the same as performing an algebraic multiplication of the numerator and denominator filter polynomials.

The complete FilterScript code becomes (including interface variables):

```
ClearH1; // clear primary filter from cascade
interface BW = {1,4,0.5,1}; // notch BW
interface fc = {0, fs/2,fs/100,fs/4}; // notch cut-off frequency
interface fl = {0, 10,0.5,0.5}; // hpf cut-off frequency range

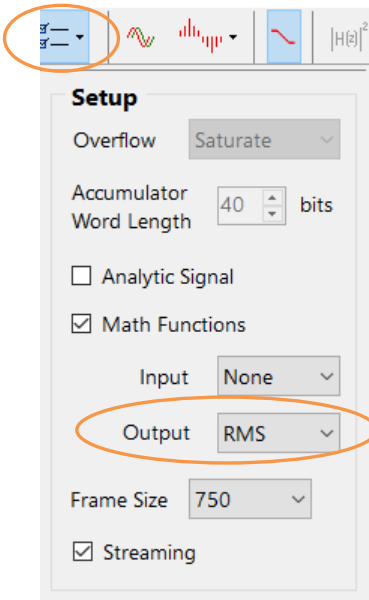
Main()

A1=dcremover(fl,"symbolic"); // A1 filter
A2=notch(fc,BW,"symbolic"); // A2 filter

Hd=augment(A1,A2,"symbolic"); // merge filters

Num=getnum(Hd);
Den=getden(Hd);
Gain=getgain(Hd);
```

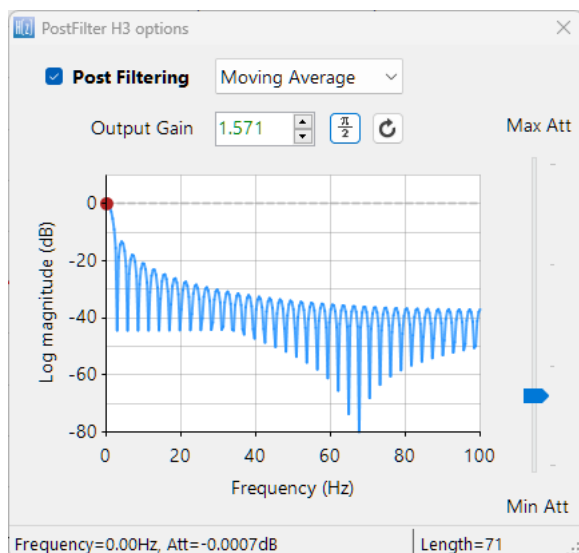
2. Mathematical functions and post filtering



The fullwave rectification and $\frac{1}{\sqrt{2}}$ mathematical operations can be summarised as one `RMS()` operation, i.e. $\frac{\text{abs}()}{\sqrt{2}}$ and implemented via the Output Math Function in the signal analyser. This scaling trick is made possible as H3 is a linear filter (see the section on post filtering), although this operation is equivalent to using `abs()` with an output Gain of 0.707.

The tool supports the following mathematical functions: `abs`, `angle`, `ln`, `RMS`, `sqr` and `sqrt`.

2.1. Post Filtering



The ASN Filter Designer's signal analyser implements an extra post filter, H3. Unlike the H1 and H2 filters, the H3 filter is *always lowpass* and is preceded by an optional mathematical function operation (see above).

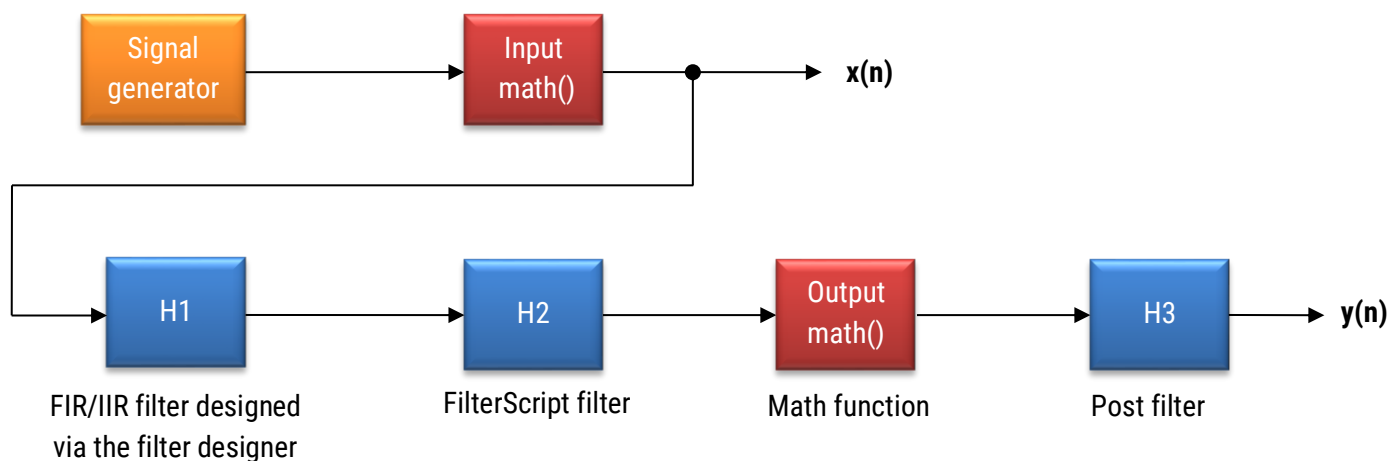
For the application considered herein, the H3 moving average filter can be implemented in the signal analyser's H3 post filter,



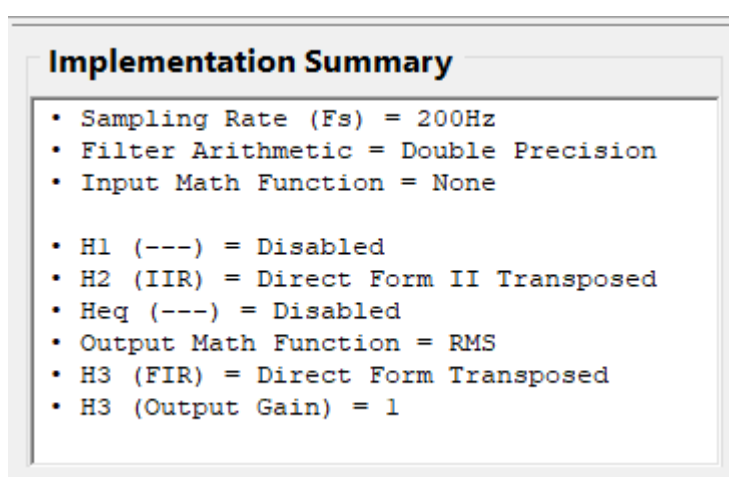
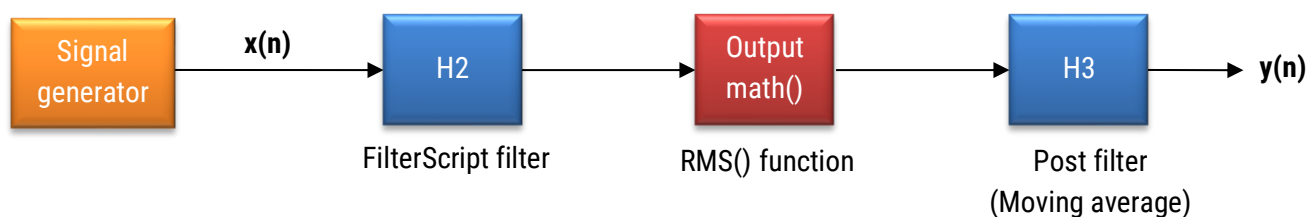
The `abs()` function is used for fullwave rectification. The Fourier series for $|\sin(wt)|$ results in a DC gain scaling factor of $\frac{2}{\pi}$ which should be corrected by setting the **Output Gain** to $\frac{\pi}{2}$ (1.571).

3. Signal cascade, building blocks and frequency response

The tool's complete signal cascade is shown below.

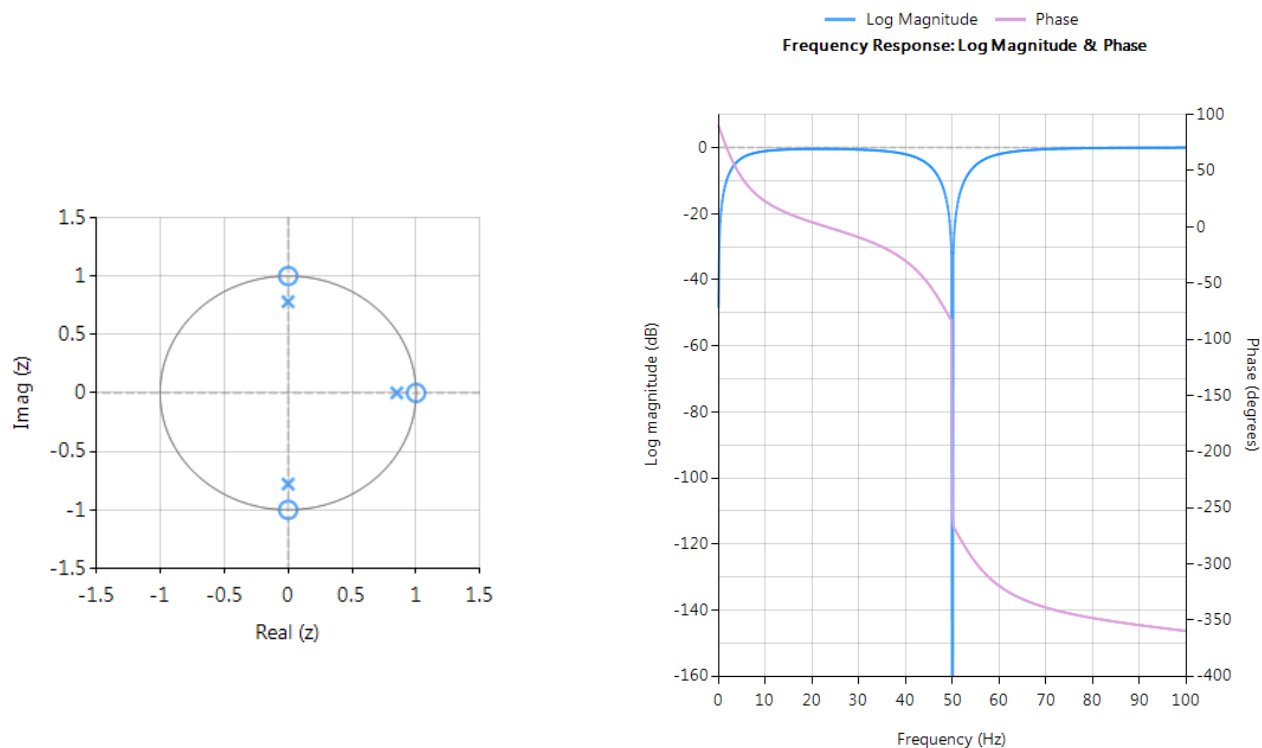



Notice that the Input math () and H1 filter blocks are **disabled** for the application considered herein. The actual signal cascade used for the application is therefore:



3.1. P-Z chart and frequency response

Upon running the FilterScript code, we obtain the following P-Z chart and frequency response.



As seen, the A1 and A2 filters have been combined correctly. You may alter the filter parameters interactively by using the signal analyser's Interface Variable Controller UI, 

The screenshot shows the 'Interface Variable Controller' window, which allows users to adjust filter parameters. The window contains a table with the following data:

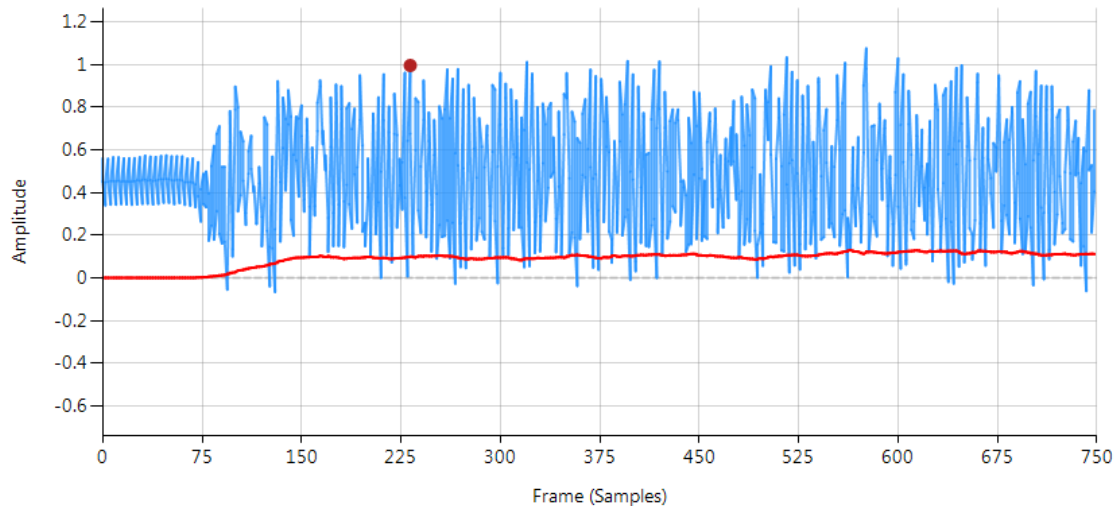
Name	Value
BW	1
f_c	50
f_l	0.5

Below the table, there is a slider control and an 'Apply' button.

4. Example project

An example project file (`\Projects\emg_ex.afd`) is provided with the tool. This project file implements the concepts described herein and uses an example EMG datafile (`\Datafiles\EMGSensorData.txt`, which may be loaded into the signal generator under the Data File option) with an additive 50Hz sinusoid. Depending on your installation directory structure, it may be necessary to find the `\Datafiles` directory manually and re-load the datafile.

The chart shown below demonstrates the complete signal processing operation. Where, it can be seen that the filtering operation (shown in red) has nicely eliminated the baseline offset and the effects of the 50Hz interference and produced an accurate estimate of the RMS amplitude of the EMG signal's envelope.

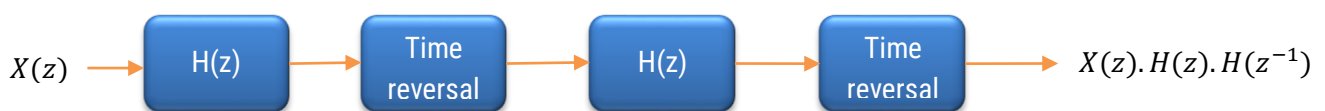


5. Zero-phase filtering

ASN Filter Designer includes the option for enabling zero-phase filtering

$$|H(z)|^2$$

This is very useful for eliminating the effects of an IIR filter's non-linear phase, as it is set to zero. The zero-phase filtering operation is anti-causal and therefore cannot be run in real-time, as seen in the following block diagram:



After enabling Block-based mode (see below), and clicking on enable, the tool will automatically perform the zero-phase filtering operation on all filters in the cascade, i.e. H1, H2, Heq and H3. Although it should be noted that enabling this option will not modify the original filter transfer function displayed in the main design canvas.

The filtering effects on an input waveform are:

1. Zero phase distortion.
2. The net filter transfer function is equal to the squared magnitude of the original filter transfer function.
3. The net filter order is double the original filter order.



Block-based mode should be selected by unchecking the **Streaming** checkbox in the Setup menu.



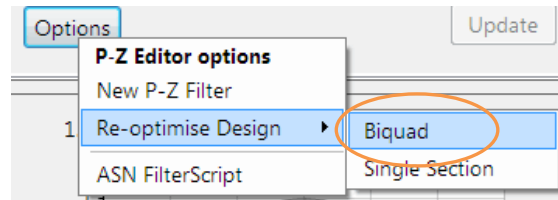
Frame Size

☐ Streaming

6. Deploying to an STM32 Arm Cortex-M microcontroller

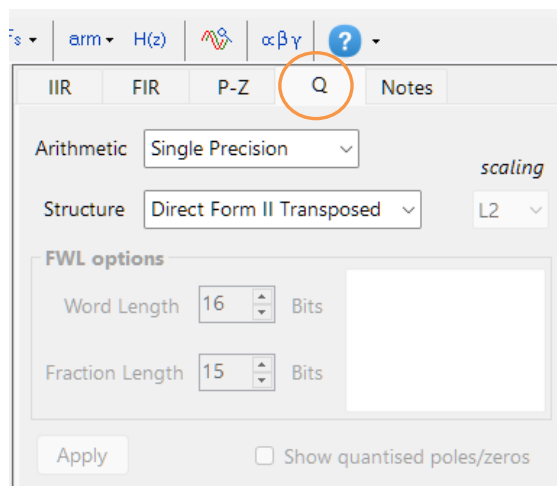
ASN's DSP ANSI C filtering framework extends the functionality of Arm's CMSIS-DSP library by virtue of supporting the complete signal cascade. The framework is actually hardware agnostic and can be run on any hardware, but has been optimised for Arm Cortex-M processors.

Before generating the code, the H2 filter (i.e. the filter designed in FilterScript) needs to be firstly re-optimised (transformed) to an H1 filter (main filter) structure for deployment. The **options** menu can be found under the **P-Z** tab in the main UI.



NB. After completing this step, the H2 filter will be removed from the cascade. The next step involves setting the desired Precision arithmetic and filter structure, as described below.

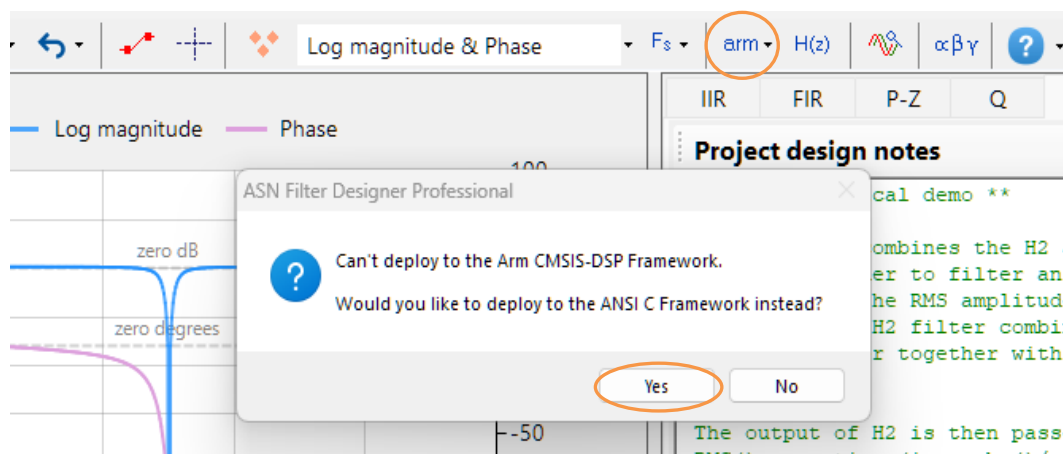
All floating-point filters designs must be based on **Single or Double Precision** arithmetic and must be either have a **Direct Form I** or **Direct Form II Transposed** filter structure. A **Biquad** cascade using the **Direct Form II Transposed** structure is advocated for floating point implementation by virtue of its higher numerical stability and accuracy.



Quantisation and filter structure settings can be found under the **Q** tab (as shown on the left). Setting **Arithmetic** to **Single Precision** and **Structure** to **Direct Form II Transposed** and clicking on the **Apply** button configures the IIR considered herein for the ANSI C software framework.

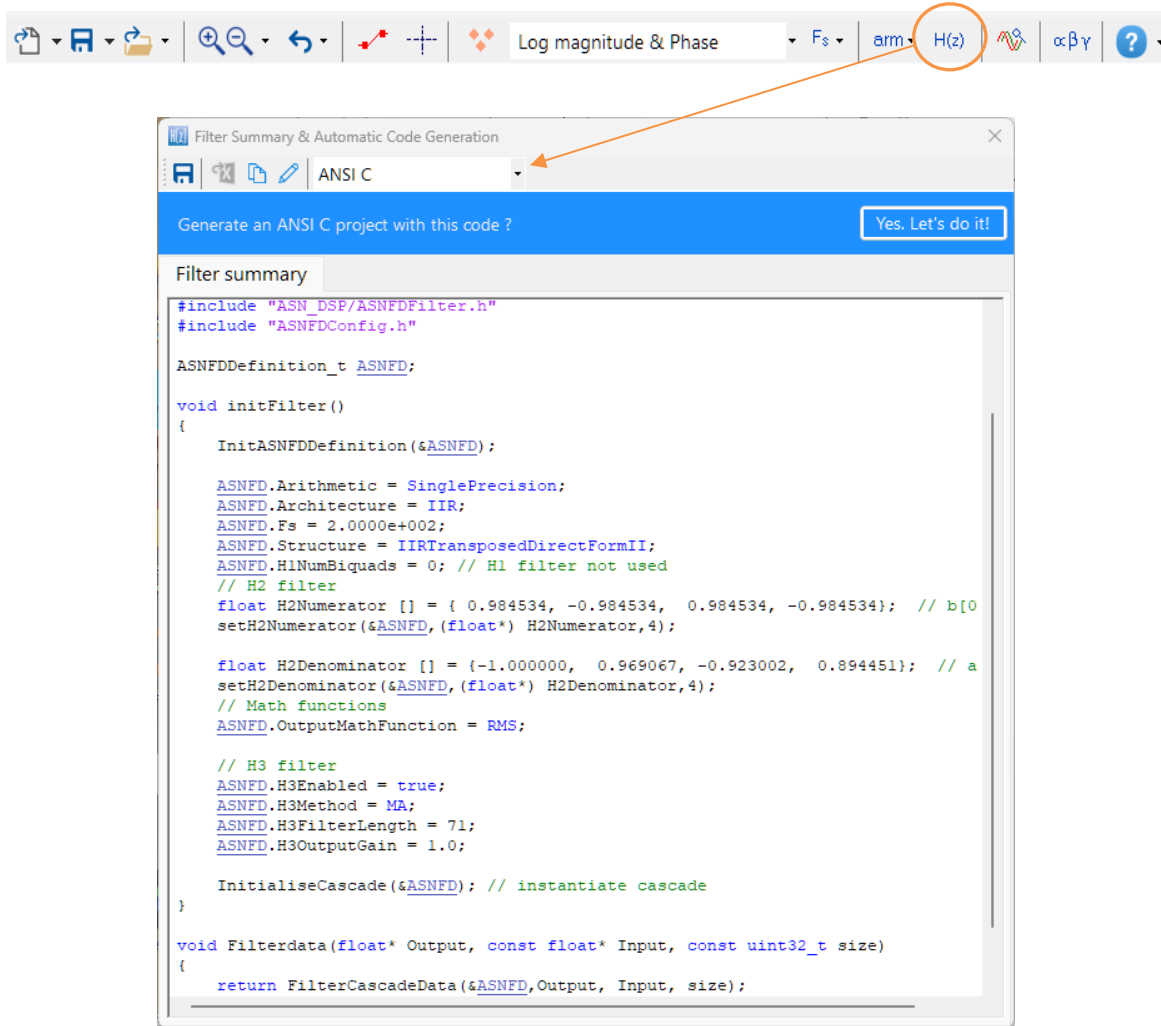


Professional licence users only: Clicking on the **arm** button will automatically convert the H2 filter into an H1 filter and generate the C code based on the current filter structure and quantisation settings.

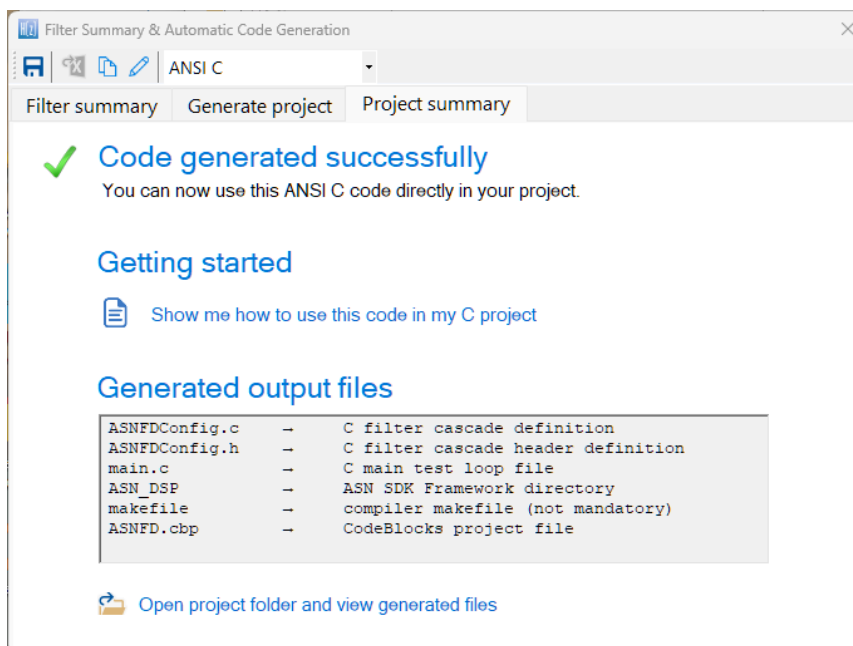


6.1. Generating application code for Arm microcontrollers

After completing the aforementioned steps, you may use the code generator UI to generate the application C code.



Clicking on **Yes. Let's do it!** generates a CodeBlocks project, which can be used in any modern IDE, such as Arm MDK and STM32Cube IDE.



6.2. Deploying to an STM32 Discovery kit and benchmarking

The [STM32F469 Discovery kit](#) is a very popular development platform for biomedical signal processing applications. The onboard Arm Cortex-M4 based microcontroller is a very capable processor, providing enough computational performance while maintaining low power and cost with floating-point operations.



The following steps should be undertaken for integrating the deployed C code library into an STM32CUBE-IDE project.

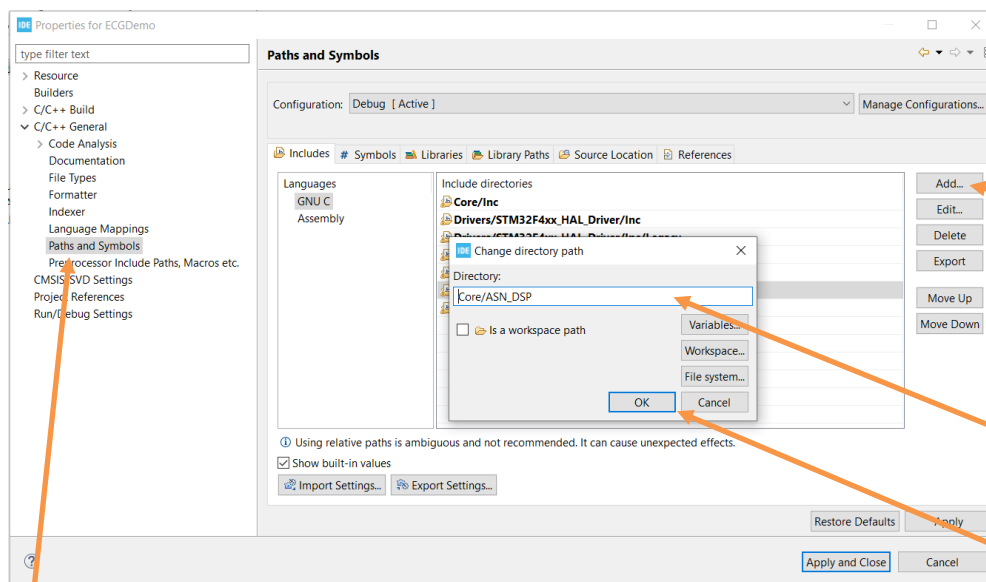


1. Generate the ANSI C filter project using ASN Filter Designer – see section 6.1.

After generating the code, the working directory of the project should look like:

Name	Date modified
ASN_DSP	01-Sep-21 6:31 PM
ASNFD.cbp	01-Sep-21 2:16 PM
ASNFDConfig.c	01-Sep-21 6:31 PM
ASNFDConfig.h	01-Sep-21 2:17 PM
main.c	01-Sep-21 6:33 PM
makefile	01-Sep-21 2:16 PM

2. Copy the `ASN_DSP` folder to your project folder.
3. Copy `ASNFDConfig.c` to the `Src` folder and `ASNFDConfig.h` file to the `Inc` folder.
4. Now open your project with STM32Cube-IDE, and go to **Properties** → **C/C++ General** → **Paths & Symbols**
5. Click on the “Add” button and enter the path of the `ASN_DSP` library folder.



Step 2: Click on “Add”

Step 3: Enter the path to the folder

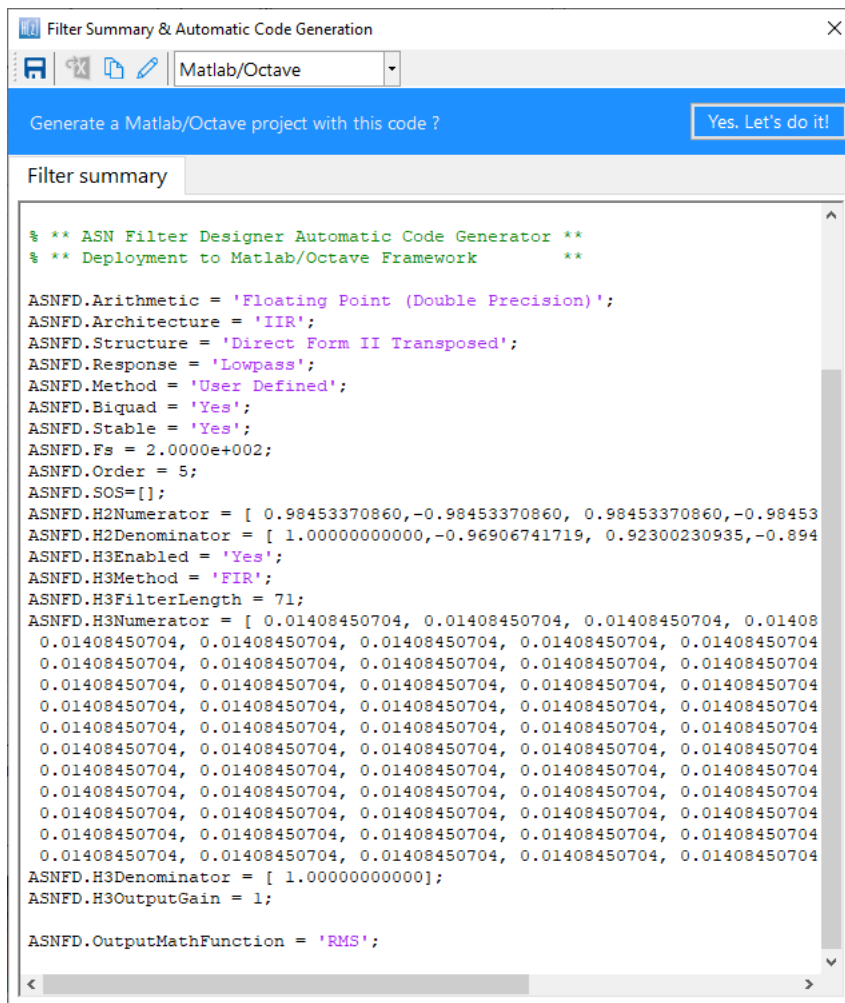
Step 4: Click on “OK”

Step 1: Click on “paths and Symbols”

Running the algorithm on the STM32 Discovery kit, we obtained a benchmark of 2.25ms for 1024 samples (5 seconds worth of data) with the onboard Arm Cortex-M4F² processor running at 180MHz with O2 optimisation.

² F suffix signifies that the device has an FPU (floating point unit).

7. Deploying to Matlab and Python



The complete design may also be exported to Matlab, Python or even Scilab for further evaluation with the 3rd party frameworks. A version for all three platforms of the example considered herein is available for evaluation:

\Matlab\EMGDataDemo.m

\Scilab\EMGDataDemo.sce

\Python\EMGDataDemo.py

As with the ANSI C code generator, a code project wizard is available allowing for the generation of a complete project, as described in these articles: [Matlab](#) and [Python](#).

Document Revision Status

Rev.	Description	Date
1	Document reviewed and released.	21/01/2016
2	Document FilterScript code upgraded	23/05/2021
3	Updated hyperlinks and zero-phase filtering section	16/11/2022
4	Added more explanations for FilterScript and C code generation	05/07/2023

Support and product details

- ▶ ASN Filter Designer product [home page](#) for a complete product overview.
- ▶ ASN Filter FilterScript [reference guide](#) for detailed examples.
- ▶ ASN Technical support: support@advsolned.com