# Inbetriebnahme FMAC

In diesem Kapitel wird die FMAC-Einheit (Filter Math Accelerator) konfiguriert und in Betrieb genommen. Um eine korrekte Funktion nachzuweisen werden verschiedene Testfälle definiert und ausgeführt.

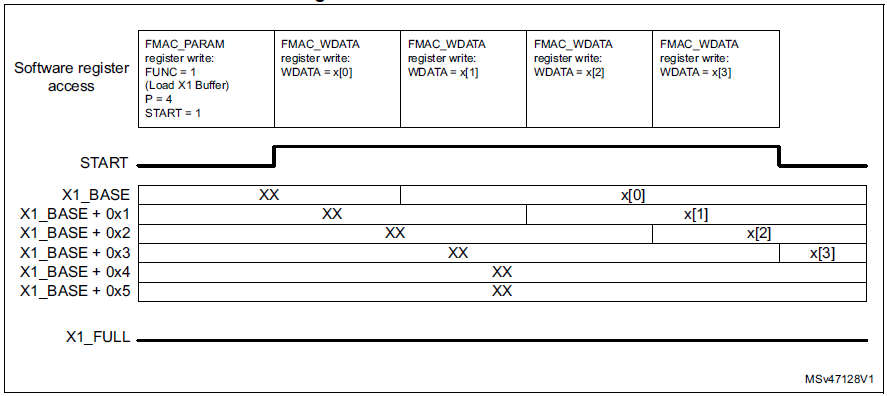
## FMAC Konfiguration in STM32CubeIDE

Ein neues Projekt wird erstellt und im Menü "Target Selection" unter dem Reiter "Board Selector" das Board NUCLEO-G474G4RE ausgewählt.

Unter dem Reiter "Categories" wird im Bereich "Computing" die FMAC-Einheit ausgewählt und aktiviert. Die Einstellungen "NVIC Settings" und "DMA Settings" werden zunächst nicht geändert, da die Inbetriebnahme ohne Interrupt und DMA erfolgen soll.

Da die aktuelle Version von STM32CubeIDE keine direkte Konfiguration von FMAC über die Benutzeroberfläche ermöglicht, muss die Konfiguration entweder manuell über Registers mithilfe des Referenzhandbuchs [] oder mithilfe der HAL (Hardware Abstraction Layer) erfolgen.

Um den Konfigurationsaufwand zu reduzieren, wird in diesem Fall die HAL-Bibliothek verwendet. Ein Beispiel für die Implementierung eines FIR-Filters mit HAL ist in der Application Note AN5305 [] zu finden.

Nach der Konfiguration des FMAC werden einige Statusregister ausgelesen, um sicherzustellen, dass sie den erwarteten Werten entsprechen. Dazu wird zunächst ein Beispiel für die Filterinitialisierung aus dem Referenzhandbuch (S. 495) verwendet.

Im Folgenden werden nur die einzelnen wichtigen Schritte beschrieben. Für die vollständige Programmliste siehe Anhang. Entsprechend den Application Notes muss zuerst die Clock für FMAC eingeschaltet werden:

\_\_HAL\_RCC\_FMAC\_CLK\_ENABLE();

Dieser Schritt wurde bereits von der Entwicklungsumgebung vorgenommen. Danach werden Float-Arrays mit Filterkoeffizienten und Eingangs-Impuls deklariert:

const float aFilterCoeffB[4] = {-0.01612203, 0.61864757, 0.61864757, -0.01612203};

Da die FMAC-Einheit nur mit dem Q1.15-Format arbeitet, müssen alle Werte in dieses Format konvertiert werden. Nun wird eine Struktur vom Typ FMAC\_FilterConfigTypeDef deklariert und alle Parameter definiert. Die FMAC-Einheit hat eine Speichergröße von 256. Dementsprechend kann ein FIR-Filter der maximalen Größe von 127 realisiert werden, wobei dann für den Output-Buffer nur eine Größe von 1 bleibt.

Dies ist allerdings unter der Annahme, dass die Eingangs- und Ausgangswerte synchron in den Input-Buffer geschrieben bzw. aus dem Output-Buffer gelesen werden. Falls dies nicht der Fall ist und die neuen Werte schneller ankommen als das gefilterte Signal gelesen wird, läuft der Output-Buffer über. Dementsprechend muss dies je nach Anwendung berücksichtigt werden und zusätzliche Puffer angelegt werden, was natürlich die maximal mögliche Filter-Ordnung reduzieren wird.

Zum Testen wird FMAC wie folgt parametriert:

* X1\_BUF\_SIZE = 6 (Eingangswerte-Buffer)
* X2\_BUF\_SIZE = 4 (B-Koeffizienten-Buffer)
* Y\_BUF\_SIZE = 2 (Ausgangs-Buffer)
* X1 Full Watermark
* Y Empty Watermark
* Filter Function Convolution
* CoeffASize = 0 (da es sich um einen FIR handelt)
* CoeffBSize = 4
* Filter = FMAC\_FUNC\_CONVO\_FIR

Der FMAC wird zuerst für den Polling-Mode konfiguriert. Dementsprechend sollen WIEN, RIEN, DMAWEN und DMAREN nicht gesetzt sein (RM S. 492, Tabelle 117). Nach der Konfiguration wird das FMAC\_CR-Register ausgelesen, um zu prüfen, dass die genannten Flags richtig gesetzt sind. Danach wird ein Buffer-Preload mit den ersten vier Samples durchgeführt:

HAL\_FMAC\_FilterPreload(&hfmac, aInputValuesImpulseResponse\_q15, 4, NULL, 0);

Nun wird der FIR gestartet. Als nächstes wird in einer For-Schleife in jeder Iteration ein Wert in den WDATA-Register (X-Buffer) geschrieben und ein Wert aus dem RDATA-Register (Y-Buffer) gelesen. Die gelesenen Werte im Q1.15-Format werden in einem Array gespeichert und anschließend in Float-Format konvertiert.

## Simple Moving Average Filter

Um eine korrekte Funktion des FMAC FIR-Filters nachzuweisen wird dieser zuerst ein einfaches gleitender Mittelwertbilder konfiguriert, welcher in englischsprachiger Literatur unter dem Begriff SMA (Simple Moving Average) bekannt ist.

Bei dieser Filterklasse entspricht die Ausgabe dem Durchschnitt der Eingangswerte über ein endliches Fenster. Mathematisch lässt sich ein diskreter Mittelwertbilder wie folgt beschreiben (Formel X.X):

– das Ausgangssignal am Zeitpunkt n

– das Eingangssignal am Zeitpunkt n

– Anzahl der Datenpunkte nach dem Mittelwert

– Anzahl der Datenpunkte vor dem Mittelwert

Frequenzgang:

Quelle: Oppenheim A. V., & Willsky A. S. with Hamid N. (1996). Signals and systems (2nd ed.). Prentice Hall Inc.. ISBN 0-13–814757–4

S. 476

Da bei einem FIR-Filter sich um ein kausales System handelt, hängt der Ausganswert y[n] nur von dem momentanen Wert x[n] und den vergangenen Werten x[n-1], x[n-2].. ab. Deschalb gilt für einen SMA FIR-Filter folgendes:

Die Ausgangssequenz eines LTD-Systems (linear, zeitinvariant und diskret) entsteht als Faltung von Eingangssignal und Impulsantwort.

Die Impulsantwort h[n] eines FIR-Systems entspricht exakt der Folge der Koeffizienten {b\_i} in der Differenzengleichung, die das System beschreibt.

Quelle : Meyer, Martin. Signalverarbeitung: .( 9. Aufl.), Springer Vieweg 2020. ISBN 978-3-658-32801-6

s. 234

Um eine korrekte Funktion des FMAC-FIR-Filters nachzuweisen wird zuerst ein FIR in Matlab mit gleichen Parametern modelliert.

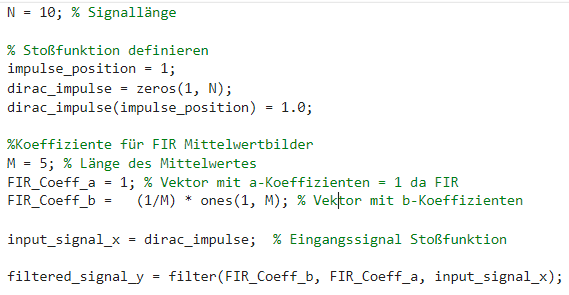
Aus der vorliegender Definition folgt für einen SMA-Filter mit N = 5:

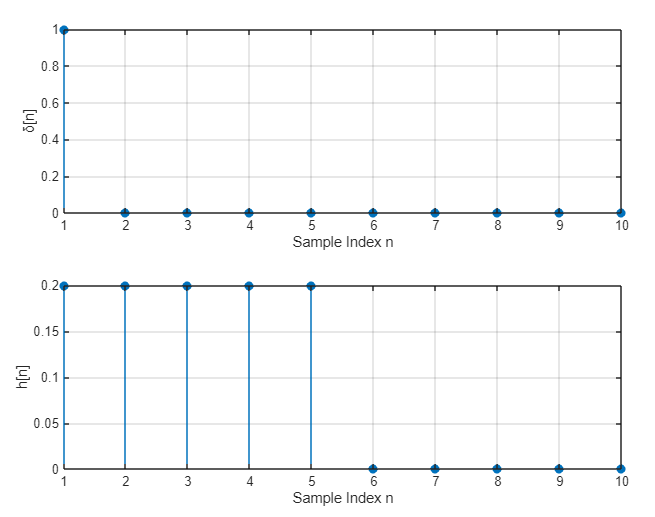
{ x[0] + x[1] + .. x[4]}

Für Dirac-Stoß gilt folgendes:

Daraus folgt dass Impulsantwort eines SMA-Filters ein Rechteckimpuls ist.

Mithilfe von o.g. theoretischen Grundlagen wird in Matlab folgendes Modell von SMA FIR-Filter aufgestellt.





## Datenformat Q15

In der digitalen Signalverarbeitung wird oft mit **Festkomma-Darstellung gearbeitet.**

Beispielweise wird in der FMAC Einheit Q15 Format verwendet.

**Festkomma-Darstellung** stellt Zahlen mit einem ganzzahligen und einem gebrochenen Teil in Zweierkomplement-Format dar.

Die Darstellung einer Q15-Zahl ist:

Wert = b14 \* 2^-1 + b13 \* 2^-2 + ... + b1 \* 2^-14 + b0 \* 2^-15

wobei bs das Vorzeichenbit (das 15. Bit) und bn die Ziffer für Bit n ist.

Der Zahlenbereich, der in einer Q15-Zahl unterstützt wird, liegt zwischen -1,0 und 1,0, entsprechend der kleinsten und größten darstellbaren Ganzzahl, -32768 bzw. 32767. Beispielsweise wird die Zahl 0,25 in Q15 als 0x2000 (8192) codiert.

Bei der Durchführung von Operationen mit Festkommazahlen lautet die Gleichung:

c = a <Operator> b

wobei a, b und c alle Festkommazahlen sind und <Operator> für Addition, Subtraktion, Multiplikation oder Division steht. Diese Gleichung gilt auch für Gleitkommazahlen.

(Quelle: an4841 Digital signal processing for STM32 microcontrollers using CMSIS)

Im Laufe der Inbetriebnahme wurde zuerst folgende empfohlene Konvertierung von float in Q15-Format angewendet.

aFilterCoeffB\_q15[i] = (int16\_t)(aFilterCoeffB[i]\*0x8000); // Convert from float to Q1.15 format

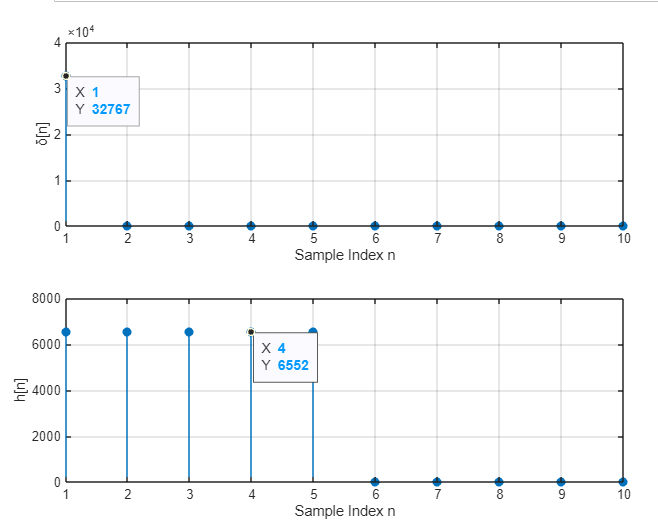
aOutputValues[i] = (**float**)aOutputValues\_q15[i]/(**float**)0x8000; // convert from Q1.15 to float

(Quelle: an5305 Digital filter implementation with the FMAC using STM32CubeG4 MCU Package)

Da die Berechnung in Matlab in Double-Format erfolgt und FMAC-Einheit mit Q15-Format rechnet, folgt eine Konvertierung zur Genauigkeitsverlust. Dies führte dazu, dass die Ergebnisse aus dem Matlab-Modell und dem FMAC nicht identisch waren.

Aus diesem Grund wird beschlossen ein Matlab-Modell in Q15-Format aufzustellen.

Das Ergebnis einer Simulation vom Impulsantwort vom SMA FIR-Filter mit ist in der Abbildung X.X zu sehen.



## Testcase 1

Der FIR-Filter soll das Eingangssignal mit dem SMA der Fensterlänge 5 falten. Da alle Koeffizienten gleich sind, entspricht der SMA an jeder Stelle einfach dem Durchschnitt der letzten 5 Eingangswerte. Im stationären Zustand (ab der fünften Stelle des Ausgangssignals) soll der Filterausgang konstant den Wert 0 liefern, da alle Eingangswerte null sind.

Testvektoren:

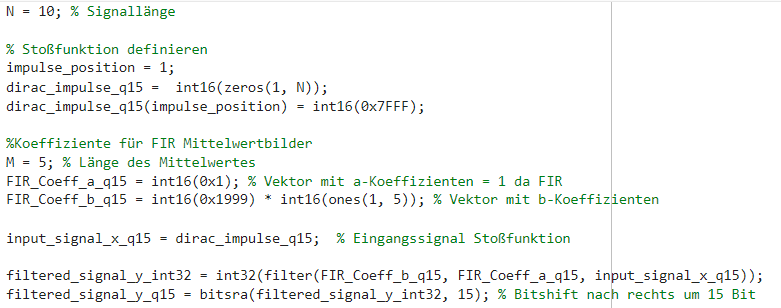
(int16) b = [6553, 6553, 6553, 6553, 6553];

(int16) x = [32767, 0, 0, 0, 0, 0];

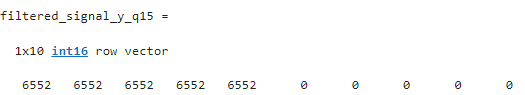
Erwarteter Ausgang:

(int16) y = [6552, 6552, 6552, 6552, 6552, 0, 0, 0, 0, 0];

### Matlab-Modell



Ausgabe in Matlab:



### Verifikation des Matlab-Modells

Der Ausgangswert 𝑦[𝑛] wird durch die Summe der Produkte der Koeffizienten und der entsprechenden Eingabewerte berechnet.

Da die Multiplikation zweier int16-Zahlen einen int32-Wert ergibt, muss dieser um 15 Bit nach rechts geschoben werden. Diese entspricht einer Division durch 32768:

Das Ergebnis aus dem Matlab-Modell stimmt mit der Berechnung überein. Das Modell ist nun verifiziert.

### Testausführung