

R X 3 2 G 4 1 0 应 用 笔 记

FMAC 滤波 ADC 应用指导

文档编号：AN00020

版本：V1.0

目录

1 简介	5
2 滤波器介绍	6
3 使用 Matlab 生成 FIR 滤波器	7
4 使用 Matlab 验证 FMAC 的 FIR 滤波结果	8
5 FMAC 专用定点数 q1.15 格式	10
6 ADC 的采样值转 q1.15 格式	10
7 版本历史	11

表目录

表 2.1 滤波器特性对比	6
表 7.1 版本历史	11

图目录

图 3.1 Matlab 滤波器设计工具	7
图 4.1 100KHz 信号输入 vs 滤波后输出	9
图 4.2 复合信号输入 vs 滤波后输出	9

1 简介

FMAC 滤波器数学加速器单元可以帮助 CPU 减轻工作负担，让处理器从频繁或冗长的滤波操作中解放。从而使处理器能够执行其他任务。

FMAC 以固定点有符号整数格式运行。输入和输出值为 q1.15 格式。且 FMAC 的滤波器前馈参数需要用户自行测试后提供，以获得最佳的滤波效果。

本文以 RX32G410（512KB Flash / 64KB SRAM）系列芯片为例，试用 Matlab 举例生成一个标准滤波器模型，以及介绍如何把 ADC 采样数据转换为标准 q1.15 格式用于 FMAC，以便用户参考，快速上手熟悉 FMAC。

2 滤波器介绍

滤波器在实现上主要分为模拟滤波器和数字滤波器。

- 模拟滤波器例如 RC/LC 电路，适用于高频或简单滤波需求。
- 数字滤波器例如 FIR/IIR，通过算法实现，适用于嵌入式系统。

RX32G410 的 FMAC 是一个滤波加速器，主要支持 FIR（有限脉冲响应）和 IIR（无限脉冲响应）两种数字滤波器。FMAC 可以提供专用指令加速 FIR 和 IIR 运算，减少 CPU 负载。

- FIR 滤波器无反馈回路，输出仅依赖当前和过去输入（前馈结构）。较为稳定，无振荡风险，但对比 IIR 相同性能下可能需要更高阶数，对 CPU 资源要求较高。
- IIR 滤波器有反馈回路，输出依赖当前输入及过去输出（递归结构）。需谨慎设计以避免振荡。对比 IIR 在相同阶数下，效率更高，更适合资源受限的场景。

表 2.1 滤波器特性对比

特性	FIR 滤波器	IIR 滤波器
脉冲响应	有限长	无限长
稳定性	无条件稳定（适用精密传感器信号）	需设计保证（适用信道滤波）
相位特性	线性相位（适用音频图像处理）	非线性相位（适用语音处理）
阶数	滤波器系数 - 1	反馈回路中极点数量
计算复杂度	复杂，高阶数（适用高精度滤波）	简单，递归结构（适用实时控制）

对比完 FIR 和 IIR，最重要的就是滤波器的频率，可以分为以下几种：

- 低通滤波器（LPF）：允许低于截止频率的信号通过，阻止高频信号。
- 高通滤波器（HPF）：允许高于截止频率的信号通过，阻止低频信号。
- 带通滤波器（BPF）：允许特定频带内的信号通过，阻止其他频率。
- 带阻滤波器（BEF）：阻止特定频带内的信号，允许其他频率通过。
- 全通滤波器（APF）：所有频率均通过，但调整相位（如延时均衡）。

本应用手册以为 ADC 滤波为例，FIR 滤波器更符合我们的需求。假设 RX32G410 的 ADC 时钟为 64MHz，采样周期选择 55.5T，那么，ADC 的采样率为： $64\text{MHz}/(55.5+12.5) = 941.176\text{KHz}$ 。

根据奈奎斯特定理，最大有效信号频率是采样率的一半，即约 470.588KHz。假设我们需要处理的信号最高频率是这个值的一半以下，比如 180KHz，那么滤波器的截止频率可设为 180KHz。

根据项目对阻带衰减（需抑制的噪声幅度）的实际需求，可增加或减少滤波器阶数。本应用手册以简单的 4 阶为基础进行测试。最终我们的滤波器参数为：4 阶 180K 截止频率的 FIR 滤波器。

3 使用 Matlab 生成 FIR 滤波器

使用 Matlab 可以帮助我们快速建立一个 FIR 数字滤波器。

打开 Matlab，在命令行窗口输入 `>> filterDesigner`

会弹出滤波器设计工具。

按照滤波器需求：

- 响应类型：低通滤波器
- 设计方法：FIR（以最小二乘法为例）
- 滤波器阶数：4 阶

相关频率设定：

- 采样率 F_s ：1MHz
- 通带截止频率（通带响应的截止频率） F_{pass} ：220KHz
- 阻带起始频率（阻带衰减开始的最小频率） F_{stop} ：260KHz

滤波器的过渡带宽： $F_{stop} - F_{pass} = 40\text{KHz}$ （过渡带宽越窄，需要的阶数就越高），所以在当前阶数固定为 4 阶时，可根据需求改变 F_{pass} 和 F_{stop} ，以改善滤波器。

完成参数设计后点击设计滤波器，可以得到幅值响应图，如下所示：

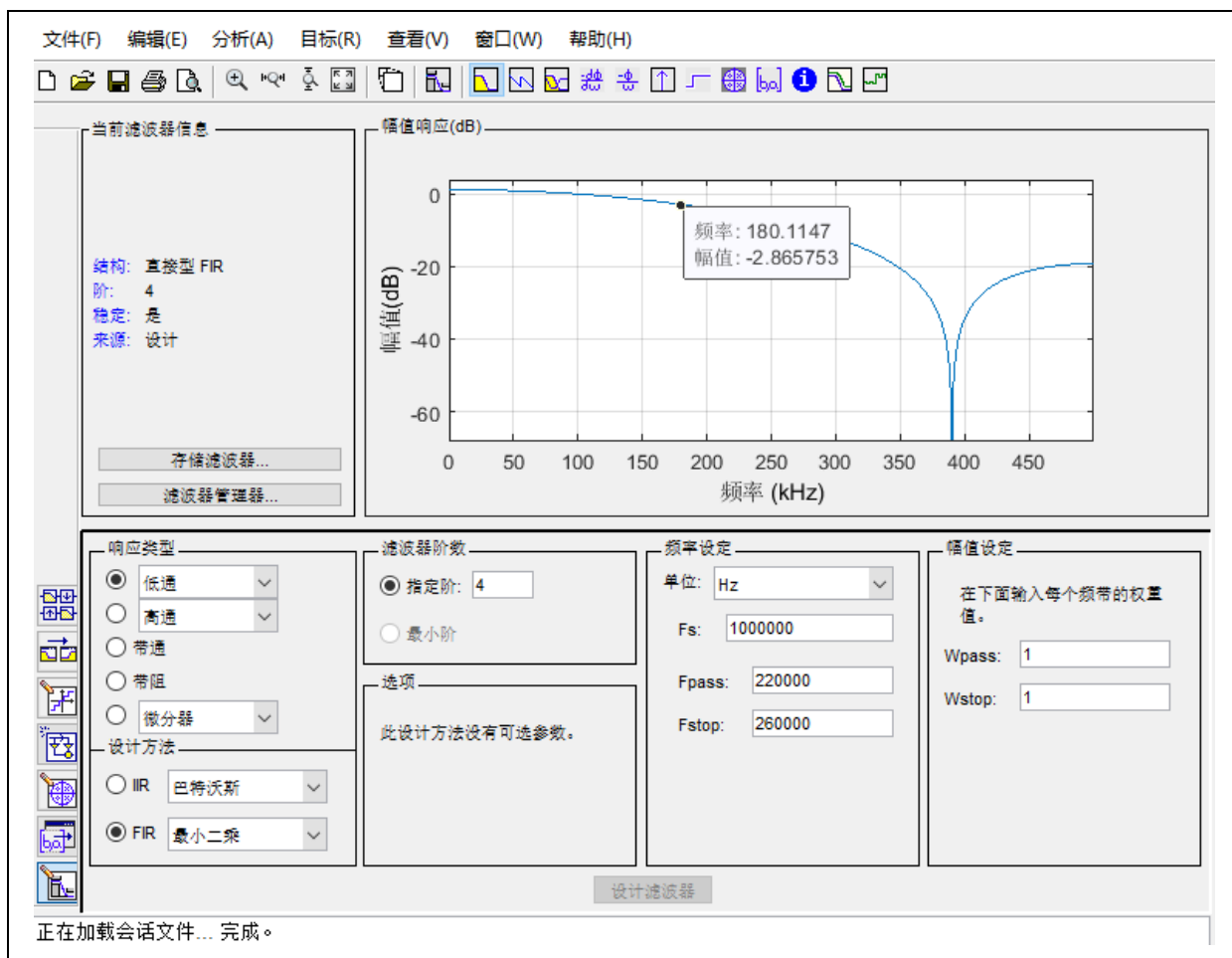


图 3.1 Matlab 滤波器设计工具

4 使用 Matlab 验证 FMAC 的 FIR 滤波结果

从上图中可以看到，调整 Fpass 为 220KHz 时，4 阶 FIR 滤波器在 180K 的衰减保持在 3dB 以内。

导出并保存滤波器参数， $b = [0.0213, 0.3153, 0.4781, 0.3153, 0.0213]$;

想要验证滤波器效果，可以使用 Matlab 输入一个 100KHz 和 300KHz 混叠的信号灌入 FIR 滤波器，看滤波后的数据是否有效抑制了高频噪声，便可完成 FIR 滤波器的性能验证。

Matlab 具体代码如下：

```
%% 1. 定义滤波器参数
b = [0.0213, 0.3153, 0.4781, 0.3153, 0.0213]; % 分子系数
a = 1; % FIR 滤波器分母系数为 1
% 创建滤波器对象
firFilter = dfilt.dffir(b);

%% 2. 生成测试信号
fs = 1e6; % 采样率 1MHz
t = 0:1/fs:1e-3; % 1ms 时长 (1000 个采样点)
% 生成 100kHz 和 300kHz 正弦波
f1 = 100e3;
f2 = 300e3;
signal_100k = sin(2*pi*f1*t);
signal_300k = sin(2*pi*f2*t);
% 生成复合测试信号 (同时包含两个频率)
signal_clean = signal_100k + signal_300k;

%% 3. 应用滤波器并绘图
% 滤波复合测试信号
filtered_signal = filter(b, a, signal_clean);
% 滤波 100kHz 信号
filtered_100k = filter(b, a, signal_100k);
% 滤波 300kHz 信号
filtered_300k = filter(b, a, signal_300k);

%% 4. 绘制时域波形
figure;
subplot(2,1,1);
plot(t, signal_clean, 'b', t, filtered_signal, 'r');
title('复合信号输入 vs 滤波后输出');
legend('输入', '输出');

figure;
subplot(2,1,1);
plot(t, signal_100k, 'b', t, filtered_100k, 'r');
title('100kHz 输入 vs 滤波后输出');
legend('输入', '输出');

figure;
subplot(2,1,1);
plot(t, signal_300k, 'b', t, filtered_300k, 'r');
title('300kHz 输入 vs 滤波后输出');
legend('输入', '输出');
```

注：本滤波器仅作为举例使用，实际滤波器参数和滤波器效果和滤波器模型以及滤波器阶数相关，需结合实际应用对滤波器模型参数进行调整和测试。

使用 Matlab 输出时域波形，以对比输入信号和滤波后输出信号。

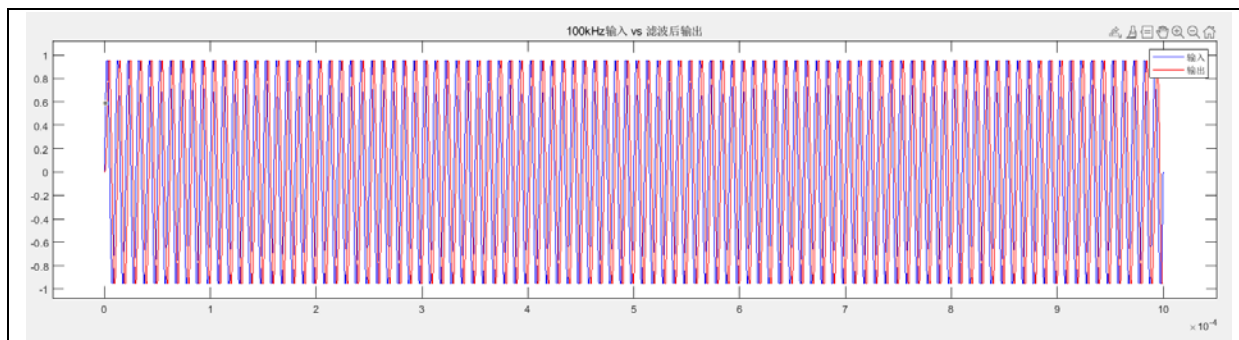


图 4.1 100KHz 信号输入 vs 滤波后输出

根据上述 100KHz 时域波形图可以看到，100KHz 的波形输入与滤波后的波形几乎一致，没有发生变化，说明该滤波器满足低通的条件。

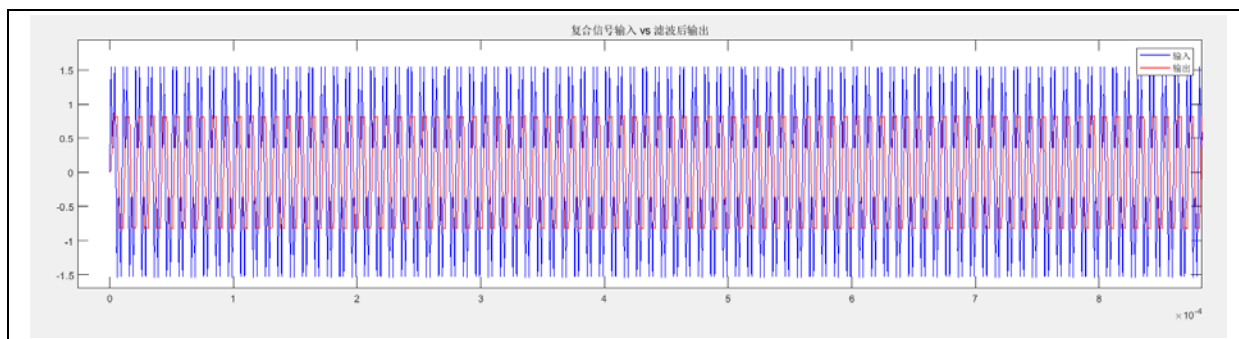


图 4.2 复合信号输入 vs 滤波后输出

根据上述混叠 100KHz 和 300KHz 的复合信号时域波形图可以看到，300KHz 的波形被有效滤波。用户可根据实际噪声频率和计算性能考量来选择滤波器的截止频率和阶数以获得较为平衡的滤波效果，或采用不同的滤波器设计方法以获得最佳的滤波模型。

5 FMAC 专用定点数 q1.15 格式

RX32G410 的 FMAC 以固定点有符号整数格式运行。输入和输出值为 q1.15 格式。

和通用 DSP 定点运算的 q1.15 格式不同，通用 DSP 中的 q1.15 代表有 1 位整数位，而 RX32G410 的 FMAC 模块的 q1.15 没有该整数位，是由符号位代替。

所以在 FMAC 模块 q1.15 格式中，数字由 1 个符号位和 15 个小数位（无整数位）表示。

因此，数值范围为 -1 (0x8000) 到 $1 - 2^{-15}$ (0x7FFF)。

即 FMAC 的 q1.15 实际数值范围： $-1 \leq x < 1$ (0x8000 = -1, 0x7FFF ≈ 0.9999695)。

6 ADC 的采样值转 q1.15 格式

RX32G410 的 ADC 是数据寄存器是 12 位。即 0~4095 (0x0000~0x0FFF)。

但 q1.15 是的范围为 -1 (0x8000) 到 $1 - 2^{-15}$ (0x7FFF)。FMAC 模块 q1.15 格式中，数字由 1 个符号位和 15 个小数位（无整数位）表示。所以 ADC 的采样值是不能直接用于 FMAC 进行运算的。需要对 ADC 的数据进行转换。

```
uint16_t adc_reg = 0;
int16_t adc_signed = 0;
int16_t q15_value = 0;
int16_t ADC_InjectToQ15(ADC_HandleTypeDef* hadc)
{
    while (!(hadc->Instance->SR & ADC_SR_JEOC));
    adc_reg = hadc->Instance->JDR1;
    adc_signed = (int16_t)(adc_reg << 4) >> 4;
    q15_value = (int16_t)((adc_signed * 32768.0f) / 2048.0f);
    if (q15_value > 32767) { q15_value = 32767; }
    if (q15_value < -32768) { q15_value = -32768; }
}
```

ADC 注入组选择右对齐，利用偏移量 ADC_JOFR1 设为 2048 (0x800)，这样读取 ADC 的数据寄存器，会减去偏置电压 2048 (0x800)。

例如：

1. 当采样值为 0，由于 ADC_JDR1 是无符号类型，原采样值 0 (0x00) 读取为 63488 (0xF800)。
2. 对 63488 (0xF800) 左移 4 位再强制为 int16_t 类型可保留符号位，即 $63488 (0xF800) \ll 4 = 1015808 (0xF8000)$ ，强制为 int16_t 类型后 $(\text{int16_t}) 1015808 (0xF8000) = -32768 (0x8000)$ 。
3. 对 -32768 (0x8000) 右移 4 位可恢复原值的同时保留其符号位，即 $-32768 (0x8000) \gg 4 = -2048 (0xF800)$ 。

再将 -2048 通过公式转换为 q1.15 格式即可送入 FMAC 进行滤波加速。

7 版本历史

表 7.1 版本历史

日期	版本	更改内容
2025 年 07 月 29 日	V1.0	初版