AN4990
Application note

Getting started with sigma-delta digital interface on applicable STM32 microcontrollers

Introduction

The DFSDM (digital filter for sigma-delta modulators) is an innovative embedded peripheral available in a selection of STM32 microcontrollers (see *Table 1: Applicable products*), and is of particular interest for applications that process external analog signals.

Although the DFSDM is a pure digital peripheral, it is designed to support a wide range of external analog front ends. By keeping the analog front-end part (sigma-delta modulator) outside of the microcontroller, the user has total flexibility to select the analog properties according to the application requirements (analog range, noise, sampling speed).

The raw converted digital data from the sigma-delta modulator is then processed by the DFSDM peripheral (digital filtering). The DFSDM configuration is flexible enough to support a wide range of converted data properties: output data width, output data rate, output frequency range.

From an application point of view, the DFSDM with its external analog front-end behaves like an ADC converter. Additional functions typical of an ADC are also available within the DFSDM such as analog watchdog, extremes detector and offset correction.

Reference:

[TUTORIAL]	In this document, *[TUTORIAL]* refers to a DFSDM simulator available in the form of a Microsoft® Excel® workbook, that can be downloaded from *www.st.com*, using home page search engine with keyword “DFSDM_tutorial”.

Table 1. Applicable products

Type	Applicable perimeter	Series, lines, references
Microcontrollers	Complete Series	STM32L4 Series, STM32L4+ Series, STM32H7 Series
	Complete Line	STM32F412 line, STM32F413/423 line
	STM32F76xxx	STM32F765BG, STM32F765BI, STM32F765IG, STM32F765II, STM32F765NG, STM32F765NI, STM32F765VG, STM32F765VI, STM32F765ZG, STM32F765ZI, STM32F767BG, STM32F767BI, STM32F767IG, STM32F767II, STM32F767NG, STM32F767NI, STM32F767VG, STM32F767VI, STM32F767ZG, STM32F767ZI, STM32F768AI, STM32F769AG, STM32F769AI, STM32F769BG, STM32F769BI, STM32F769IG, STM32F769II, STM32F769NG, STM32F769NI, STM32F769SL
	STM32F77xxx	STM32F777BI, STM32F777II, STM32F777NI, STM32F777VI, STM32F777ZI, STM32F778AI, STM32F779AI, STM32F779BI, STM32F779II, STM32F779NI

Contents

List of figures

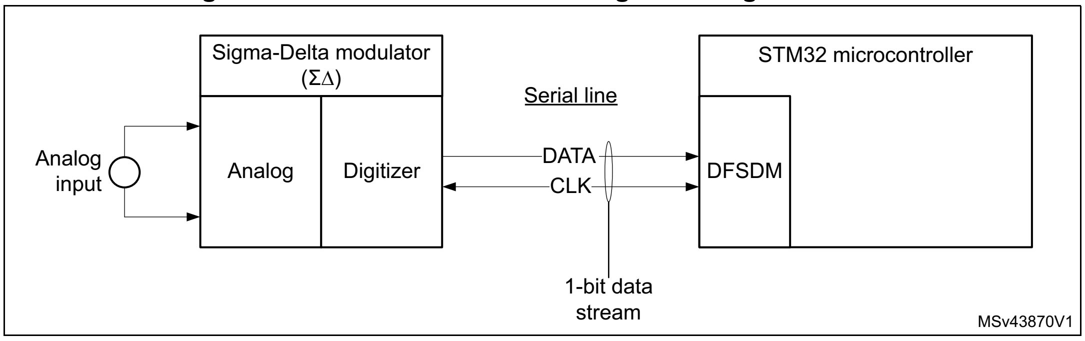
1 Overview of A/D conversion principle using DFSDM

This document supports Arm®[a]-based devices.

1.1 Fundamental concept of A/D conversion using DFSDM

The basic block diagram of analog-to-digital conversion using DFSDM is provided in *Figure 1*.

Figure 1. A/D conversion block diagram using the DFSDM



The external analog signal is processed by an external sigma-delta modulator which converts the analog signal into a digital 1-bit stream (DATA and CLK signals). The 1-bit stream is a fast serial line stream of logical ones and zeros: the DATA signal is sampled by CLK (clock signal). The average value of these logical ones and zeros, computed during a long enough time duration, represents the analog input value. The duration of the averaging period determine the precision of the analog input signal capture.

The averaging of the 1-bit stream is performed by the STM32 microcontroller DFSDM peripheral (DFSDM = digital filter for sigma-delta modulators). The DFSDM acquires and processes the 1-bit data stream (digital filtering, averaging). The DFSDM outputs data samples at a slower data rate than the input 1-bit stream but with a higher resolution. The DFSDM digital filter settings define the output resolution and data rate.

1.2 Sigma-delta modulator

The DFSDM peripheral requires an external analog front-end that performs the A/D conversion of the analog source. This external analog to digital conversion is performed in a sigma-delta modulator.

A sigma-delta modulator consists in a 1-bit[b] A/D converter which digitizes the input analog data into a serial digital data stream. The analog input is sampled and converted into a 1-bit digital data stream with alternating zeros and ones. The mean value of the digital stream

a. Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

b. In general the output of a sigma-delta modulator can be multi-bit however in this document the focus is on a 1-bit A/D converter (which is the most frequent case).

computed during a given time interval represents the average value of the input analog signal during the same time interval. The sigma-delta modulation principle could be presented as a special PWM modulation where both the period and the duty cycle would be modulated (whereas the period is fixed and only the duty cycle is modulated in a typical PWM modulation). See *Figure 2* and *Figure 3* for comparison between PWM and sigma-delta modulation.

The digital data stream outputting the sigma-delta modulator is then processed by the STM32 microcontroller DFSDM peripheral. The DFSDM performs a digital filtering using parameters that need to be configured according the application requirements.

Note: *For analysis, the digital stream is usually "converted" from binary 0 and binary 1 weights into +1 and -1 weights for comparison with input voltages cleared of any DC component. The zero input voltage generates duty cycle 50:50 (first order sigma-delta modulator is used).*

Figure 2. PWM modulation example

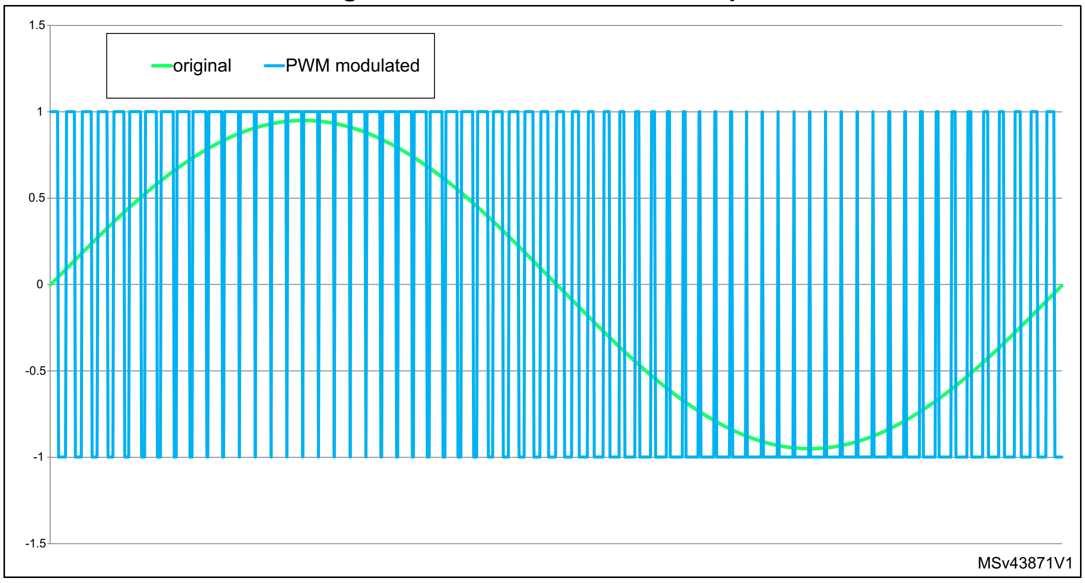
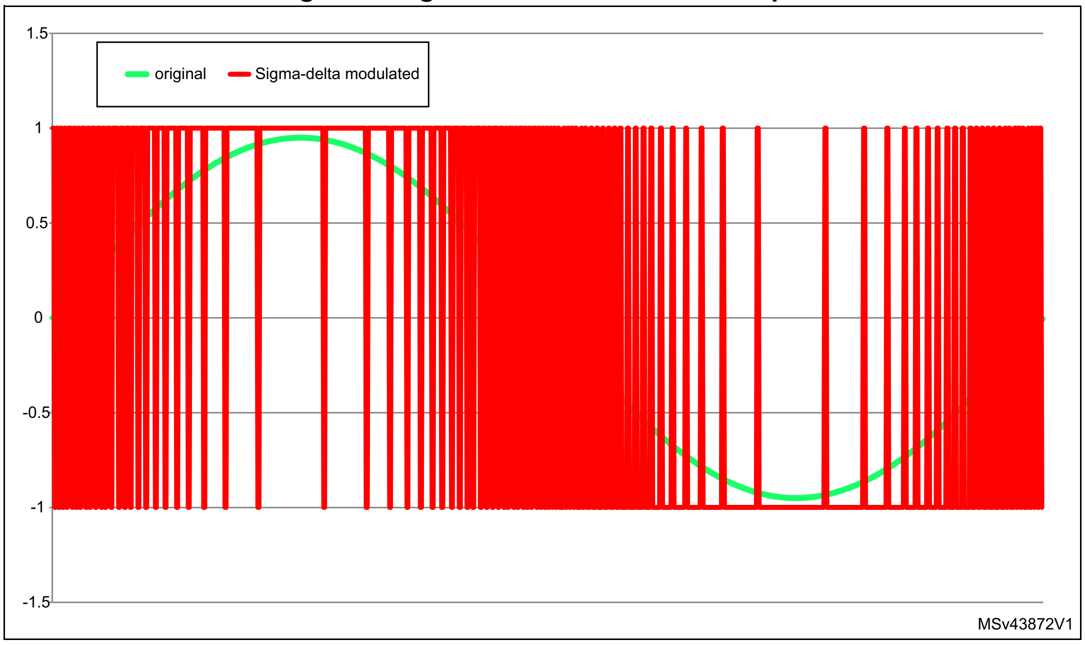


Figure 3. Sigma-delta modulation example



1.3 Digital filter

The DFSDM peripheral (digital filter for sigma-delta modulators) processes the digital part of the A/D conversion. The digital data stream is provided by an external sigma-delta modulator. The basic functionality of the DFSDM is to implement a digital filter. The DFSDM processing consists in averaging a fast rate input serial stream and producing a parallel, lower rate, data output with higher resolution. The DFSDM embedded filter features a set of configurable parameters that allow to tune the output resolution and data rate and meet the application requirements.

The DFSDM features additional ADC-related functionalities including:

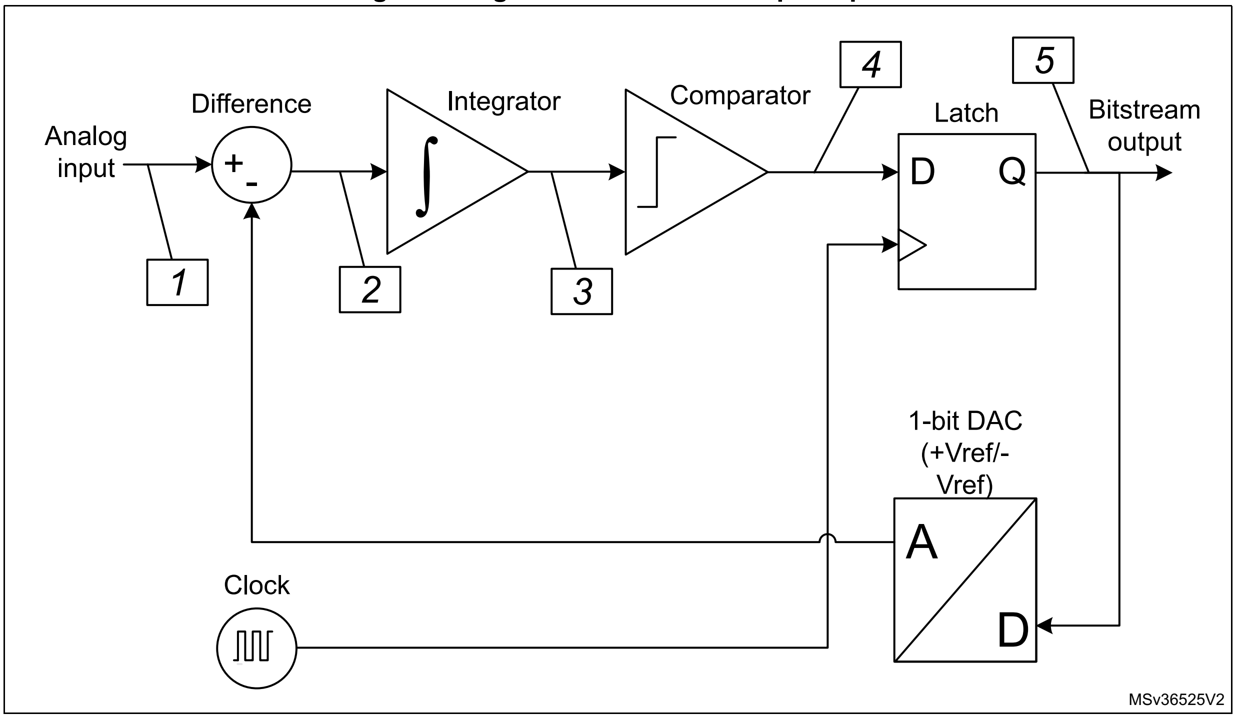
- Independent fast watchdog on each channel with programmable speed and resolution to detect input signals exceeding minimal or maximal allowed voltage levels.
- Break signal generation used to instantaneously report events like analog watchdog or short circuit detection to other peripherals (timers).
- Short circuit detector on each channel for very fast detection of signal clamping: when input voltage reaches one of the analog range limits and stays steady in excess of a given time duration (independent from the main conversion).
- Extreme detector to record minimal and maximal input voltage excursion.

2 Sigma-delta modulation principle (external analog front-end functioning, simulations)

2.1 Principle of sigma-delta modulation

The basic functional block diagram of a sigma-delta modulator is presented on *Figure 4*.

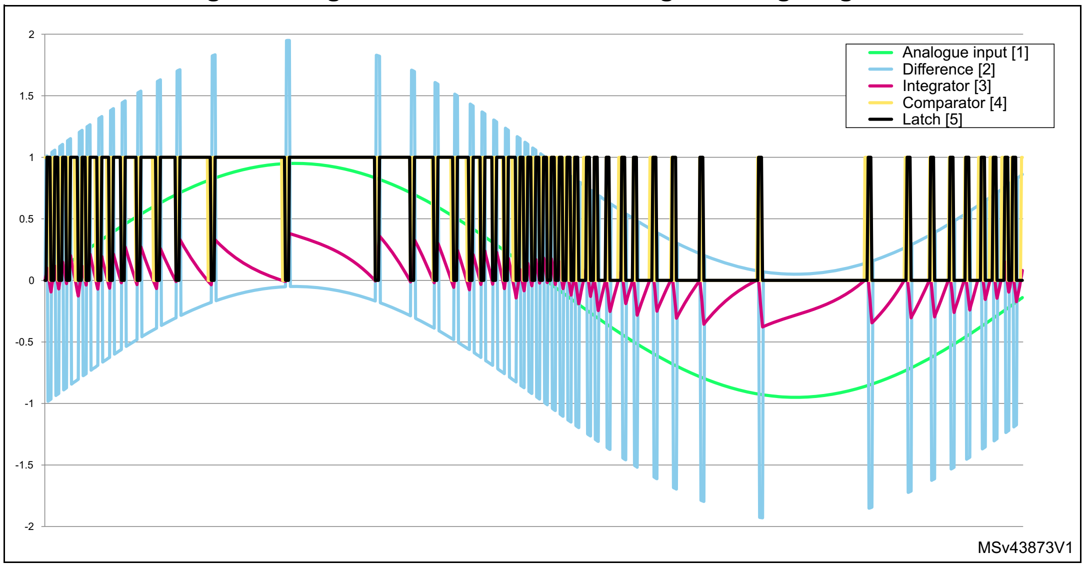
Figure 4. Sigma-delta modulation principle



1. The references [1] to [5] present in the above figure are used in the following paragraphs.

Figure 5 provides an example of the signals available at the different stages of the analog to digital conversion.

Figure 5. Sigma-delta modulator voltages timing diagram



1. The references [1] to [5] present in the above figure are used in the following paragraph.

The below description of sigma-delta modulation uses references present in *Figure 4* and *Figure 5*:

The analog input signal [1] is added to the 1-bit DAC output feedback from the comparator (+Vref or -Vref voltage) and the result [2] goes to the integrator. The integrator cumulates the difference between the analog input signal [1] and the 1-bit DAC output feedback (+Vref or -Vref voltage). The integrator output [3] is then compared with the zero voltage reference by the comparator. The comparator output [4] is latched periodically at the clock frequency by the D-latch to propagate the comparator result to the modulator output in quantized time steps (clock ticks). The D-latch output [5] is the digital 1-bit output from the sigma-delta modulator. The output is fed back to the 1-bit D/A converter which outputs only 2 possible analog voltages (usually implemented as a switch between +Vref and -Vref reference voltages). The data rate of the 1-bit output data stream is defined by the modulator clock frequency.

The output of the sigma-delta modulation is a digital data stream (black curve on *Figure 5*) that is clocked by the modulator clock. The average value of this output (computed in the digital domain) represents the input analog voltage. This digital average should be computed as the ratio between the number of ones versus the number of zeros observed during a given number of clock periods within the sigma delta output stream.

This digital data stream constitutes the input of the STM32 microcontroller DFSDM peripheral where it can be filtered. DFSDM configurable parameters need to be set according the application requirements.

2.2 Advantages of sigma-delta modulation

2.2.1 Noise shaping

The output signal from the sigma-delta modulation must be filtered to remove the high frequency content (the quantization noise) and retain only the useful frequency band. In order to properly design such filter, some understanding of the sigma-delta modulated signal spectrum is required. The sigma-delta modulated signal spectrum is different from the PWM modulation spectrum.

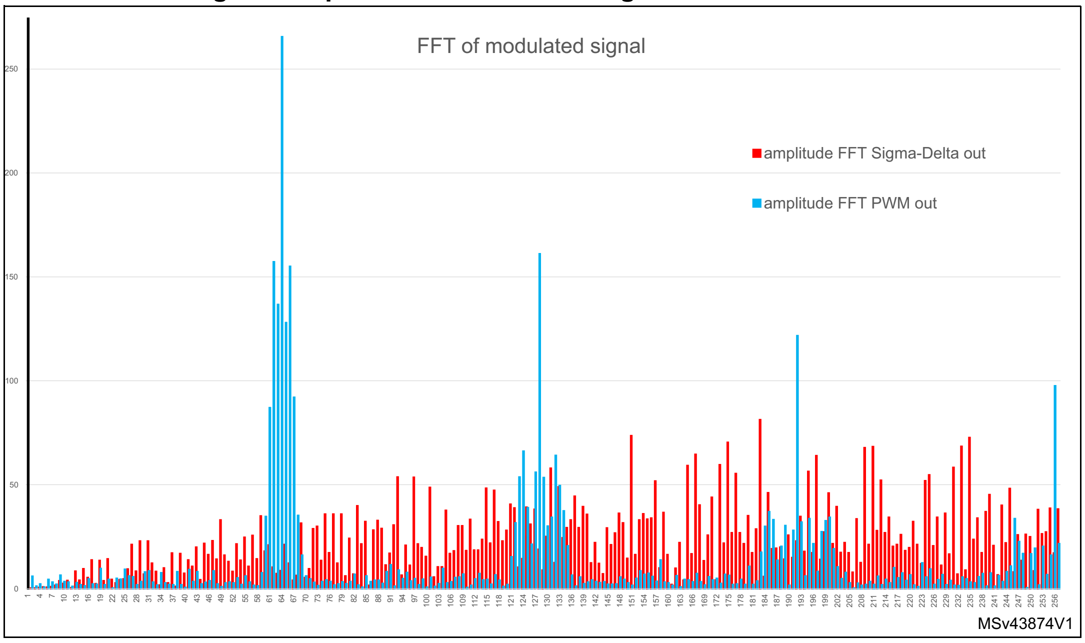
The PWM modulation signal is characterized by a fixed period and a variable duty cycle. Due to the fixed modulation period (or frequency), the PWM spectrum exhibits typical peaks of energy corresponding to the modulation base frequency and its harmonics. The removal of these harmonic peaks is more difficult by analog filtering (RC or LC filter).

The sigma-delta modulation uses both variable duty cycle and variable frequency. As a result, the energy on the sigma-delta spectrum is spread more evenly and is not concentrated on regularly spaced peaks like for the PWM (there is no fixed modulation frequency). Furthermore, there is usually more energy at higher frequencies for sigma-delta modulations than for PWM (due to higher modulation frequencies). The noise content of sigma-delta modulation can be easily removed by analog filtering (RC or LC filter).

A typical spectrum for a PWM and for a sigma-delta modulation are provided on *Figure 6* (corresponding respectively to signals presented in *Figure 2: PWM modulation example* and *Figure 3: Sigma-delta modulation example*). The PWM modulation produces higher peaks at lower frequencies, therefore the PWM processing requires a filter of higher order in order to properly reject the first peak of energy corresponding to the PWM modulation base frequency. In the sigma-delta modulation spectrum the energy is less present at lower frequencies so the filter design can be simpler. The filtering capabilities must be adapted to the order of the sigma-delta modulator. For example, digital microphones typically have a 4th order sigma-delta modulator, which has a low quantization noise in the useful band, but a very strong out-off band quantization noise. The filter order must be selected in order to suppress this strong out-off band quantization noise without affecting the useful band.

The effect of the sigma-delta modulation to spread the quantization noise more evenly and to higher frequencies is called "noise shaping". One can take advantage of such characteristic to design simpler filters and benefit from less noisy baseband signals.

Figure 6. Spectrum of PWM and sigma-delta modulation



2.2.2 Linearity of A/D conversion

The output from the sigma-delta modulator is 1-bit serial data stream. The resolution of this stream is only one bit which is usually not enough for the application. The method to increase the signal resolution consists in averaging the 1-bit stream during a given period of time. The averaged data stream has a wider resolution (typically 16-bit) but a slower sample rate. The averaging (filtering) operation is based on linear mathematical operation in the digital domain hence there is no added non-linear distortion due to filtering.

A/D converters with parallel data output usually use more analog elements than the sigma-delta modulators. For example the SAR ADC type with N-bits resolution uses internally a R-2R resistor network (or C-2C capacitor network) with N resistors (or capacitors). The resistors (capacitors) used in these type of ADCs must have a precise 1:1 or 1:2 resistance (capacitance) ratio. The linearity of the SAR ADC depends on the precision of the resistors (capacitors). In practice the ratios are never perfect and the imprecision is at the origin of non-linearities that impact the transfer curve.

For above mentioned reasons, the linearity of A/D conversion using sigma-delta modulation is usually better than with other type of A/D conversion. The non-linearity in the sigma-delta ADC conversion depends only on the sigma-delta modulator design (see *Figure 4: Sigma-delta modulation principle*). Furthermore, the non-linearity depends only on the influence of the input voltage on analog element intrinsic characteristics (capacitors/resistors in integrator and switches which capacitance/resistance changing with input voltage) and not on ratios between the different component nominal values. Linearity is more important for audio applications where the non-linearity causes signal distortion (static linearity INL is linked with dynamic linearity THD).

2.2.3 Scalable ADC resolution

The final output resolution for sigma-delta conversion is not fixed (as it is for example for a 12-bit SAR ADC). The output from a sigma-delta modulator is 1-bit resolution which can then be increased by following digital filtering (averaging) to the required resolution.

The drawback of this resolution increase is the reduction of the output data rate. This data rate decrease is predictable and must be computed for a given application requirement. For example a digital filter can be configured for a 24-bit output @ 1 kHz output data rate, or for a 16-bit output @ 50 kHz output data rate.

The resolution increase is theoretically unlimited but in practice one needs to take into consideration the noise and the errors caused by the components in the conversion path (sigma-delta modulator design, injected noise …). Another consideration is the stability of the signal during the sampling period that can affect the precision of the measurement, in particular for applications with very low data rate and high resolution.

2.3 Disadvantages of sigma-delta modulation

2.3.1 Offset and gain error

The average of the 1-bit data stream signal from sigma-delta modulator represents the mean value of the analog input signal. The precision of this digital output average (in the range [0..1]) can be affected by the following sigma-delta modulator components (see *Figure 4: Sigma-delta modulation principle*):

- resistors and capacitors in the integrator
- reference voltage in the 1-bit DAC (+Vref / -Vref)
- offset of the integrator

These components have tolerances: capacitor/resistor values, offset voltage, difference between +Vref and -Vref absolute voltage.

In the ideal case, a 1-bit digital output having in average the same amount of 0's and 1's corresponds exactly to a zero volt input signal. Due to the components tolerances (mentioned above) the input analog voltage corresponding in a same amount of 0's and 1's, is not exactly zero volt, and represents the offset error. The offset error can be compensated by software or by hardware (calibration process).

In the ideal case the output 1-bit digital signal with a 50:50 duty cycle (amount of 0's is equal to the amount of 1's) should correspond exactly to the zero input analog voltage. Due to the components tolerances (mentioned above) the input analog voltage corresponding to the 50:50 duty cycle is not exactly zero and represents the offset error. The offset error can be compensated by software or by hardware (calibration process).

The gain coefficient that is the ratio between the output data and the input voltage is also affected by the sigma-delta component tolerances. The difference between the theoretical gain and the actual measured conversion gain represents the gain error. It can be also compensated by calibration (usually in software).

Some characteristics of the sigma-delta components are also dependent on temperature, which affects in turn the offset and gain errors. The influence of the temperature on the integrator resistors and capacitors affects only the gain error. The offset error is less affected by the temperature changes because the errors affecting +Vref and -Vref, due to the symmetric nature of these references, are self compensated.

2.3.2 Lower data rate

With sigma-delta modulators, a way to increase the resolution is to increase the averaging of the 1-bit data stream (longer averaging time or higher filter order). This is why the sigma-delta converters are primarily used for lower data rate applications (usually for audio frequency range and lower frequencies). But in special cases where the excellent linearity is a must, the sigma-delta converters can be used for higher data rate applications.

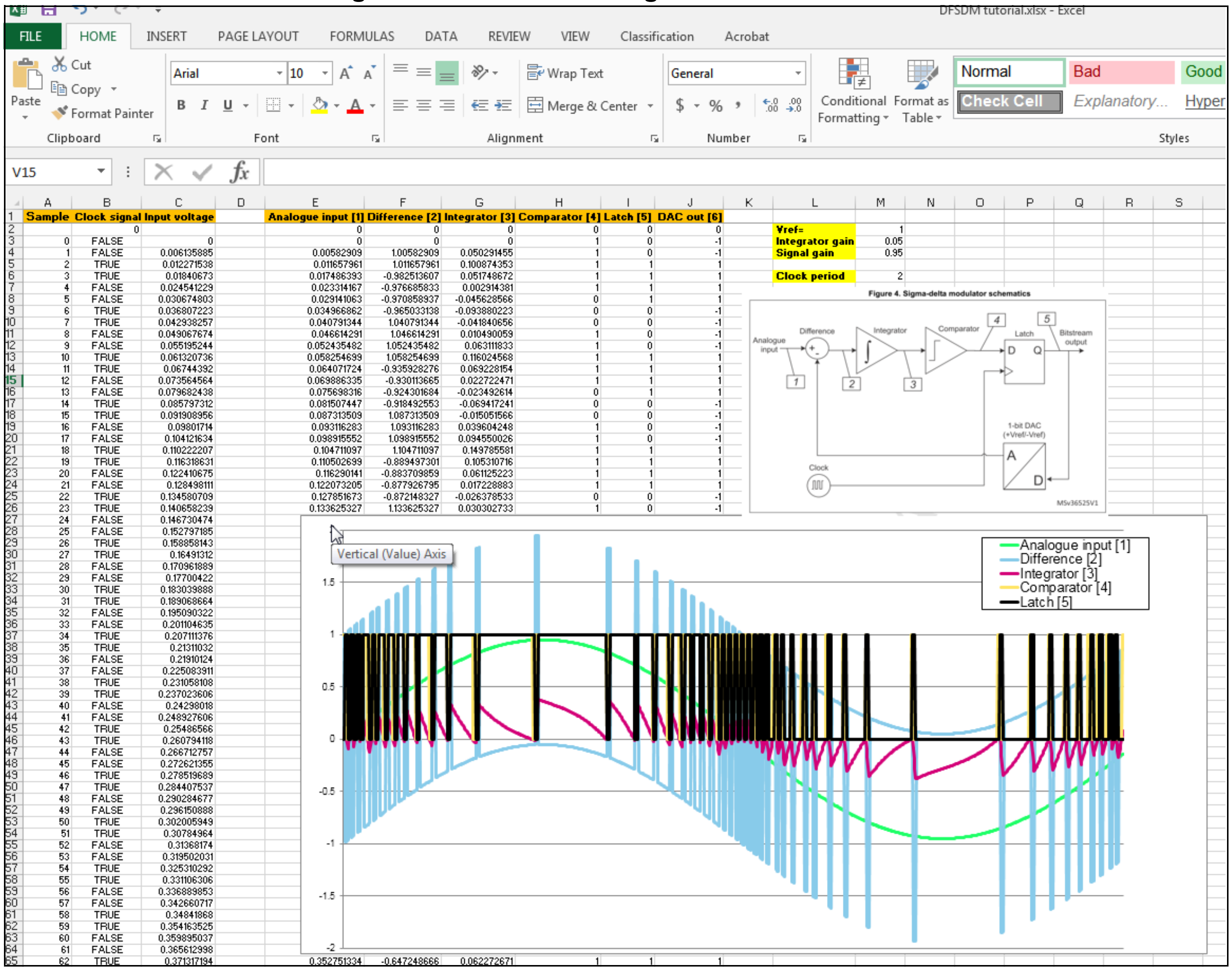
For quasi static applications (temperature sensors) the data rate is not a constraint and the sigma-delta modulator is often chosen for its scalable resolution capability (high resolution on low data rates).

2.4 Simulation of sigma-delta modulation

2.4.1 Simulation with *[TUTORIAL]*

To help understanding the sigma-delta modulation, the sigma-delta model (as shown in *Figure 4: Sigma-delta modulation principle*) has been implemented in *[TUTORIAL]*. The timing diagrams are provided for each voltage signal referenced in *Figure 4: Sigma-delta modulation principle*. The user can change some parameters and input voltages and see the impact at each stage of the sigma-delta modulator. A simulation example available in *[TUTORIAL]* and based on a sinewave input signal is shown on *Figure 7*.

Figure 7. Simulation of sigma-delta modulator



	A	B	C	E	F	G	H	I	J
1	Sample	Clock signal	Input voltage	Analogue input [1]	Difference [2]	Integrator [3]	Comparator [4]	Latch [5]	DAC out [6]
2		0		0	0	0	0	0	0
3	0	FALSE	0	0	0	0	1	0	-1
4	1	FALSE	0.006135885	0.00582909	1.00582909	0.050291455	1	0	-1
5	2	TRUE	0.012271538	0.011657961	1.011657961	0.100874353	1	1	1
6	3	TRUE	0.01840673	0.017486393	-0.982513607	0.051748672	1	1	1
7	4	FALSE	0.024541229	0.023314167	-0.976685833	0.002914381	1	1	1
8	5	FALSE	0.030674803	0.029141063	-0.970858937	-0.045628566	0	1	1
9	6	TRUE	0.036807223	0.034966862	-0.965033138	-0.093880223	0	0	-1
10	7	TRUE	0.042938257	0.040791344	1.040791344	-0.041840656	0	0	-1
11	8	FALSE	0.049067674	0.046614291	1.046614291	0.010490059	1	0	-1
12	9	FALSE	0.055195244	0.052435482	1.052435482	0.063111833	1	0	-1
13	10	TRUE	0.061320736	0.058254699	1.058254699	0.116024568	1	1	1
14	11	TRUE	0.06744392	0.064071724	-0.935928276	0.069228154	1	1	1
15	12	FALSE	0.073564564	0.069886335	-0.930113665	0.022722471	1	1	1
16	13	FALSE	0.079682438	0.075698316	-0.924301684	-0.023492614	0	1	1
17	14	TRUE	0.085797312	0.081507447	-0.918492553	-0.069417241	0	0	-1
18	15	TRUE	0.091908956	0.087313509	1.087313509	-0.015051566	0	0	-1
19	16	FALSE	0.09801714	0.093116283	1.093116283	0.039604248	1	0	-1
20	17	FALSE	0.104121634	0.098915552	1.098915552	0.094550026	1	0	-1
21	18	TRUE	0.110222207	0.104711097	1.104711097	0.149785581	1	1	1
22	19	TRUE	0.116318631	0.110502699	-0.889497301	0.105310716	1	1	1
23	20	FALSE	0.122410675	0.116290141	-0.883709859	0.061125223	1	1	1
24	21	FALSE	0.128498111	0.122073205	-0.877926795	0.017228883	1	1	1
25	22	TRUE	0.134580709	0.127851673	-0.872148327	-0.026378533	0	0	-1
26	23	TRUE	0.140658239	0.133625327	1.133625327	0.030302733	1	0	-1
27	24	FALSE	0.146730474						
28	25	FALSE	0.152797185						
29	26	TRUE	0.158858143						
30	27	TRUE	0.16491312						
31	28	FALSE	0.170961889						
32	29	FALSE	0.17700422						
33	30	TRUE	0.183039888						
34	31	TRUE	0.189068664						
35	32	FALSE	0.195090322						
36	33	FALSE	0.201104635						
37	34	TRUE	0.207111376						
38	35	TRUE	0.21311032						
39	36	FALSE	0.21910124						
40	37	FALSE	0.225083911						
41	38	TRUE	0.231058108						
42	39	TRUE	0.237023606						
43	40	FALSE	0.24298018						
44	41	FALSE	0.248927606						
45	42	TRUE	0.25486566						
46	43	TRUE	0.260794118						
47	44	FALSE	0.266712757						
48	45	FALSE	0.272621355						
49	46	TRUE	0.278519689						
50	47	TRUE	0.284407537						
51	48	FALSE	0.290284677						
52	49	FALSE	0.296150888						
53	50	TRUE	0.302005949						
54	51	TRUE	0.30784964						
55	52	FALSE	0.31368174						
56	53	FALSE	0.319502031						
57	54	TRUE	0.325310292						
58	55	TRUE	0.331106306						
59	56	FALSE	0.336889853						
60	57	FALSE	0.342660717						
61	58	TRUE	0.34841868						
62	59	TRUE	0.354163525						
63	60	FALSE	0.359895037						
64	61	FALSE	0.365612998						
65	62	TRUE	0.371317194	0.352751334	-0.647248666	0.062272671	1	1	1

	L	M
2	Vref=	1
3	Integrator gain	0.05
4	Signal gain	0.95
5		
6	Clock period	2

3 Digital filtering - principle and design

3.1 Function description

The digital filter performs the filtering (averaging) of the 1-bit data stream generated from the sigma-delta modulator. The filter output is a data word with higher resolution (usually 12 - 24 bits) but reduced data rate (decimation). The digital filter function consists in removing out-off band frequency components (quantization noise, unwanted signals...) and reducing the data rate according to the useful bandwidth (decimation).

The design of the filter has a strong influence on the A/D conversion characteristics and is the result of a compromise between the required parameters (sharpness of filter, filter tuning, final resolution, …) and the hardware implementation complexity (which leads to cost issue). The goal is to minimize the filter design complexity while meeting the required A/D characteristics.

Note: Signal filtering often requires more complex processing than simply averaging the 1-bit data stream.

Here are some elements to consider when designing a filter:

- Filter type:

Among various types of filter, the Sinc filter presents interesting characteristics, combining a cheap hardware implementation with acceptable level of performance. The Sinc filter has a frequency response that can be modelized by a sinc(x) function (hence its name). The Sinc filter is the most commonly used type in sigma-delta A/D converter implementations. It is cheap because no multipliers are required, and the filter coefficients are integers. The Sinc filter performs a simple “moving average” calculation over the 1-bit samples.

- Filter length (FOSR - filter oversampling ratio):

A longer filter (averaging of more samples) generates higher resolution but decreases the output data rate (decimation). Therefore the filter length is selected as a compromise between required conversion rate and final data resolution.

- Filter order (FORD):

The “moving average” calculation can be applied multiple times to already averaged samples. The filter order defines how many time the “moving average” calculation is applied to the same input samples. A higher order filter generates higher resolution (by adding more averaging loops) but increases latency.

3.2 Example of Sinc filter function - resolution increase

This section focuses on showing how multiple “moving average” calculations can increase the Sinc filter resolution.

Here is a detailed presentation of a 3rd order filter (FORD=3) with length FOSR=10 (with moving averaging at each filter stage):

- The observation is done on 3 periods of the input stream: 3 x FOSR = 30 samples (see *Figure 8* and *Figure 9*).
- The input 1-bit data stream is almost always ‘0’ except for '1' pulse per averaging period (FOSR=10) (see the "Input" curves on *Figure 8* and on *Figure 9*). Two input streams are tested: one stream with exactly equidistant pulses (at position 10, 20 and

30 on horizontal axis see *Figure 8*) and another stream where the second pulse is anticipated by one period to simulate a slightly higher pulse density (position 10, 19 and 29 on horizontal axis, see *Figure 9*).

- The first order filter stage is performing moving average (see the "1st order" curves) and the final result is sampled at each FOSR cycle and constitutes the 1st order filter result (at 10th, 20th, 30th cycle). It always produces '1' as the final result on both stream cases (*Figure 8* and *Figure 9*) because of simple moving average in 1st filter stage (only one '1' in each period).
- The second order filter stage is performing moving average on samples from first filter order stage (see the "2nd order /10" curves). The final result is sampled at each FOSR cycle and constitutes the 2nd order filter result (at 20th, 30th cycle). There is a visible difference in the final results between the two output streams (*Figure 8* and *Figure 9*): due to higher density of '1s' (with the higher density stream of *Figure 9* it produces higher value than with the lower density stream of *Figure 8*).
- The third order filter stage is performing moving average on samples from the second filter order (see the "3rd order /100" curves). The final result is sampled at each FOSR cycle and constitutes the 3rd order filter result (at 30th cycle). There is a visible difference in the final result between the lower density stream (*Figure 8*) and the higher density stream (*Figure 9*). Due to moving averaging the third output is smoother and more precise.
- The dynamic range of the signal at the output of the filter is $FOSR^{FORD}$ but requires FORD x FOSR samples before obtaining the first result because every filter stage must be filled with valid samples from previous filter stage (see *Figure 8* and *Figure 9*). In conclusion, higher order filters offer better resolution for a given stream duration or the same resolution for shorter stream duration. The drawback is a more complex hardware design and a longer initialization time for the filter.

Figure 8. Example of 3rd order filter outputs with one bit per filter length

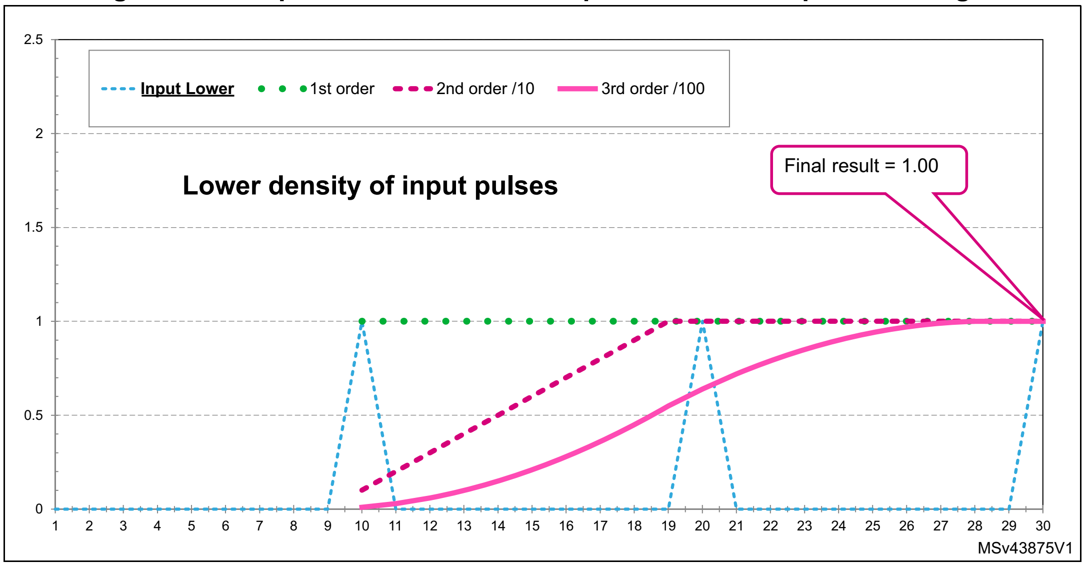
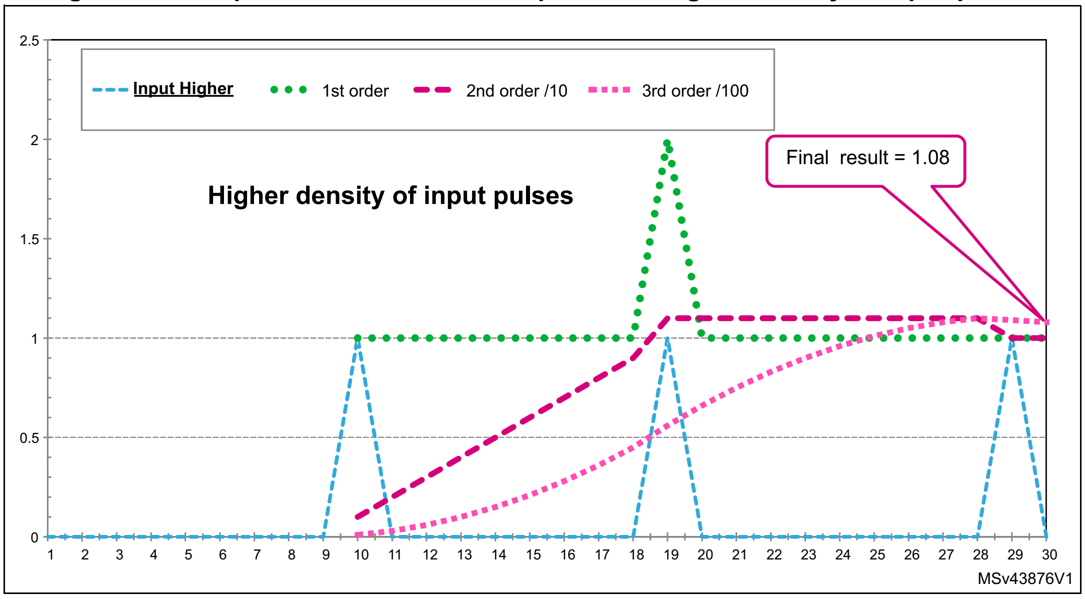


Figure 9. Example of 3rd order filter outputs with higher density of input pulses



3.3 Hardware design of Sinc filter

Sinc filters are characterized by a Sinc transfer function which equation in the digital domain is provided below. It can be interpreted as multiple moving average of input 1-bit data. The rest of this section focuses on finding simplifications in order to achieve an efficient and ultimately quite simple hardware implementation.

Equation 1:

$$y(n) = x(n) + x(n-1) + x(n-2) + \ldots + x(n-(FOSR-1))$$

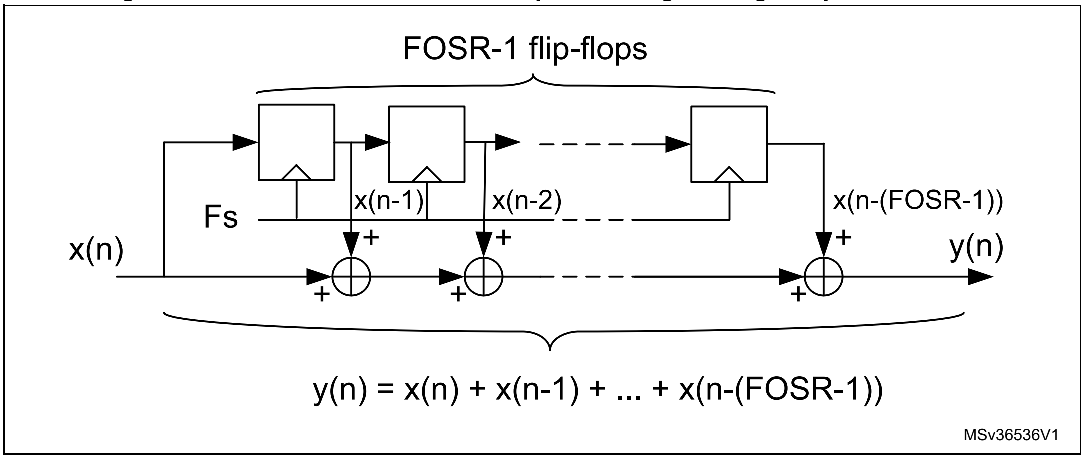
Where:

x(n) is the input of nth sample

y(n) is the output of nth sample

Figure 10 provides the straight forward translation of the above equation into a hardware implementation:

Figure 10. Basic schematic for simple moving average implementation



The above hardware implementation required (FOSR -1) adders and (FOSR-1) flip-flops. Some simplifications can be introduced that are developed hereafter.

The *Equation 1:* can be simplified by using previous output result y(n-1):

Equation 2:

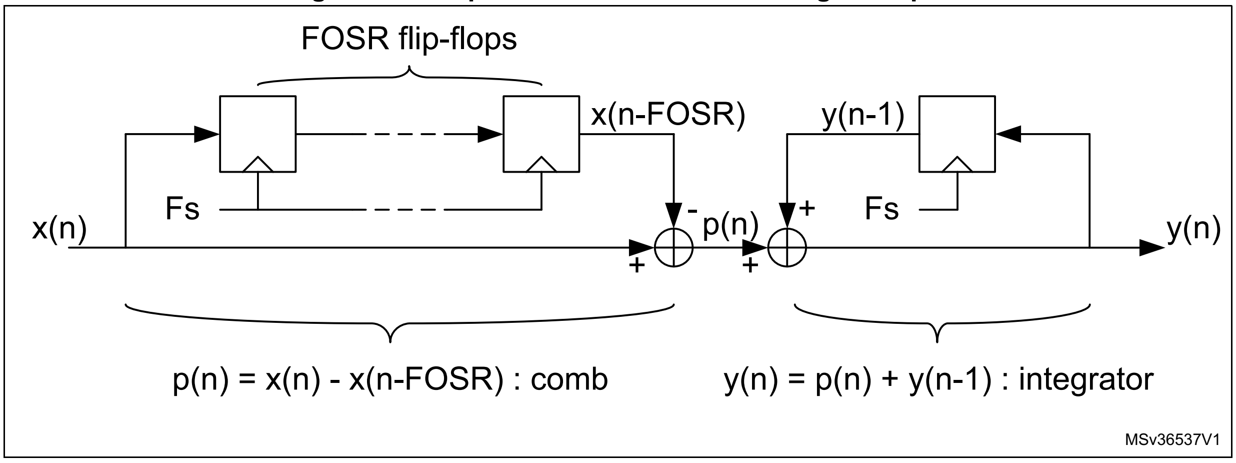
$$y(n) = x(n) + y(n-1) - x(n-FOSR)$$

Equation 3:

$$y(n) = x(n) - x(n-FOSR) + y(n-1)$$

Figure 11 provides a simplified schematic requiring only 2 adders and FOSR+1 flip-flops. The schematic can now be divided into two parts: comb and integrator.

Figure 11. Simplification of Sinc filter design - step 1


FOSR flip-flops
x(n-FOSR)
y(n-1)
Fs
Fs
x(n)
p(n)
y(n)
-
+
p(n) = x(n) - x(n-FOSR) : comb
y(n) = p(n) + y(n-1) : integrator
MSv36537V1

A further simplification can be obtained by converting the equation to Z-domain (frequency domain) according these correspondences:

- X(z) in the Z-domain corresponds to x(n) in the discrete time domain, where n is an integer number of sample clock periods (sample clock period = 1/Fs; Fs = sampling frequency)
- z^{-N} x X(z) corresponds to x(n-N) that is x(n) delayed by N sample periods.

The equation for a 1st order Sinc filter in Z-domain becomes:

$$Y(z) = X(z) - z^{-FOSR} \times X(z) + z^{-1} \times Y(z)$$

The corresponding transfer function H(z) is:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1 - z^{-FOSR}}{1 - z^{-1}}$$

In *Figure 11* we introduced a decomposition of the Sinc filter in two stages: "comb" and "integrator". The transfer function of such filter (called CIC: cascaded integrator-comb filter) is given by:

The comb transfer function in Z-domain:

$$H_C(z) = \frac{P(z)}{X(z)} = 1 - z^{-FOSR}$$

The integrator transfer function:

$$H_I(z) = \frac{Y(z)}{P(z)} = \frac{1}{1 - z^{-1}}$$

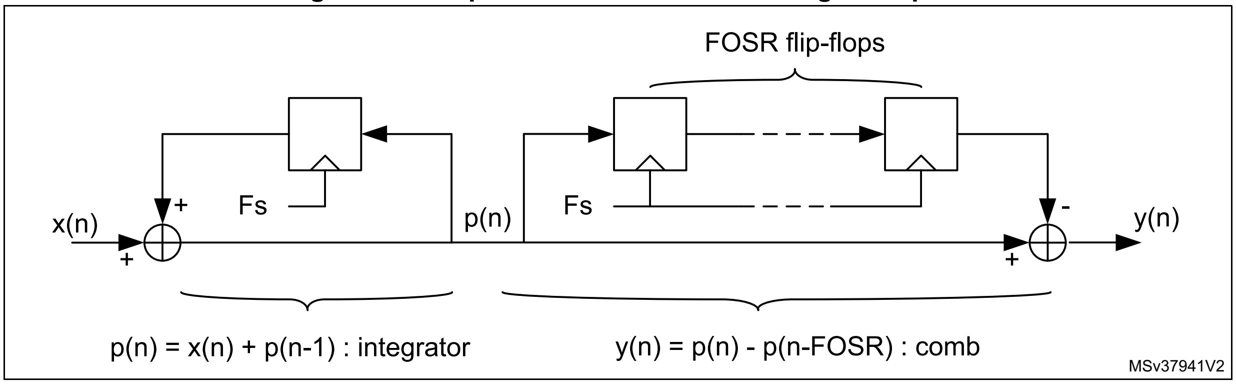
The overall transfer function is equivalent to the Sinc filter transfer function:

$$H_C(z) \cdot H_I(z) = \frac{P(z)}{X(z)} \cdot \frac{Y(z)}{P(z)} = (1 - z^{-FOSR}) \cdot \left(\frac{1}{1 - z^{-1}}\right) = \frac{1 - z^{-FOSR}}{1 - z^{-1}} = H(z)$$

One more filter simplification consists in permuting the comb and the integrator operation as shown in *Figure 12*:

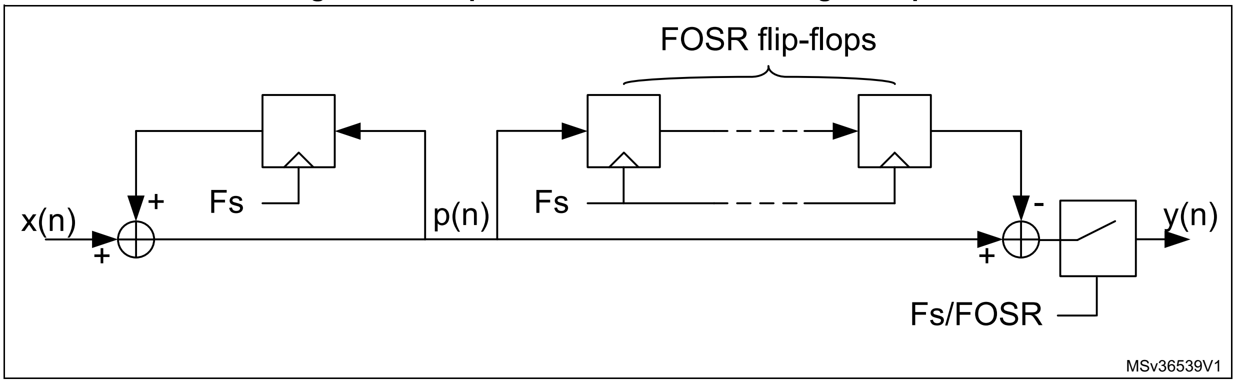
$$H(z) = H_I(z) \cdot H_C(z) = \left(\frac{1}{1 - z^{-1}}\right) \cdot (1 - z^{-FOSR})$$

Figure 12. Simplification of Sinc filter design - step 2


FOSR flip-flops
x(n)
+
+
Fs
p(n)
Fs
-
+
y(n)
p(n) = x(n) + p(n-1) : integrator
y(n) = p(n) - p(n-FOSR) : comb
MSv37941V2

Because the filtering limits the output signal bandwidth, it is possible to down-sample the output y(n) by a factor FOSR. This down-sampling is achieved by taking one sample of y(n) every FOSR clock cycles (see *Figure 13*).

Figure 13. Simplification of Sinc filter design - step 3



The down-sampled clock period is defined as:

$$T = \frac{FOSR}{F_s} = FOSR \cdot \tau \qquad \text{where: } : \tau = \frac{T}{FOSR}$$

The equation for the down-sampled comb section can be re-written as follows:

$$y(n \cdot \tau) = p(n \cdot \tau) - p(n \cdot \tau - FOSR \cdot \tau)$$

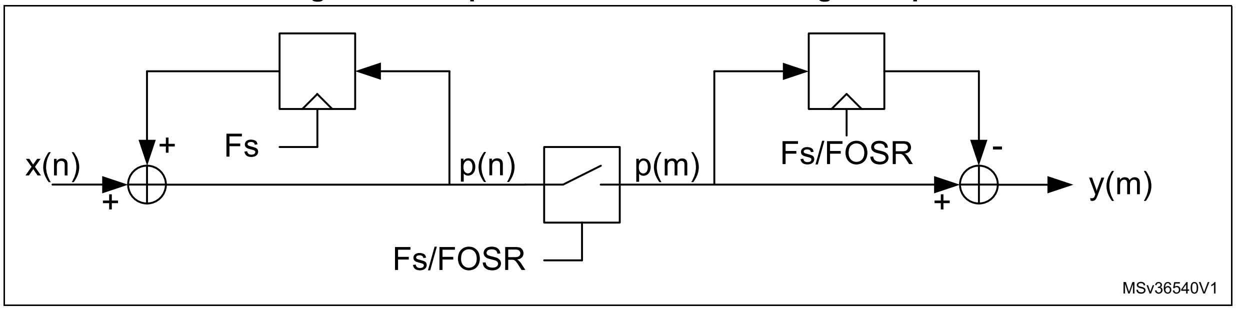
$$y\left(\frac{n \cdot T}{FOSR}\right) = p\left(\frac{n \cdot T}{FOSR}\right) - p\left(\frac{n \cdot T}{FOSR} - \frac{FOSR \cdot T}{FOSR}\right)$$

$$y(m) = p(m) - p(m-1) \qquad \text{where: } : m = \frac{n}{FOSR}$$

This operation corresponds to a down-sampling of p(n) by the factor FOSR, followed by a first order differentiator. The series of flip-flop sampled at high frequency can be replaced by one single flip-flop with down-sampled frequency. The final schematic is shown on *Figure 14*.

An effect of down sampling is to eliminate the moving average operation (because the output data rate is reduced).

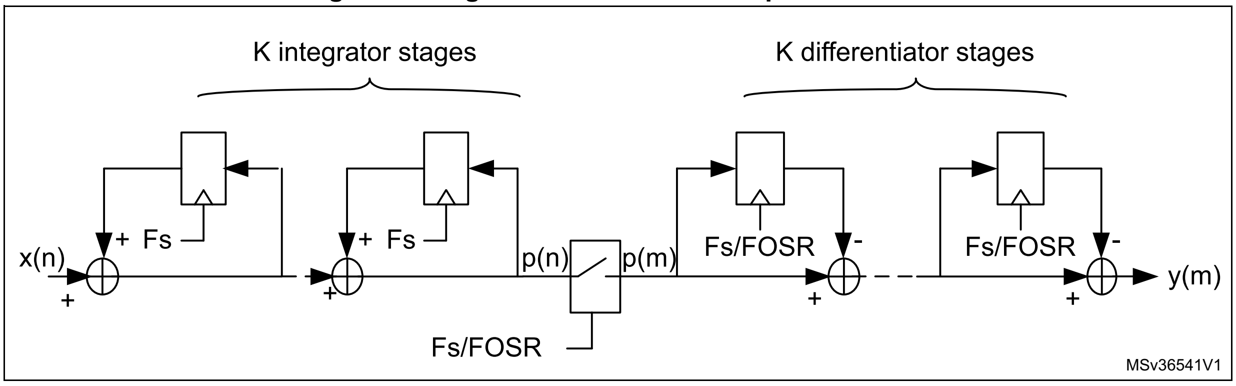
Figure 14. Simplification of Sinc filter design - step 4



The final schematic (*Figure 14*) is reduced to 2 flip-flops and 2 adders (all with FOSR-bit width).

Higher order Sinc filters are obtained as a serial cascade of first order filter stages where comb and integrator stages can be grouped together (*Figure 15*).

Figure 15. Higher order Sinc filter implementation



The transfer function of a higher order Sinc filter (with filter order = FORD) is provided by:

- Cascading FORD first order filters:

$$H(z) = \left[\left(\frac{1}{1-z^{-1}}\right)\cdot\ (1-z^{-FOSR})\right]\cdot\ \left[\left(\frac{1}{1-z^{-1}}\right)\cdot\ (1-z^{-FOSR})\right]\ldots$$

- Reordering:

$$H(z) = \left[\left(\frac{1}{1-z^{-1}}\right)\cdot\ \left(\frac{1}{1-z^{-1}}\right)\ldots\right]\cdot\ [(1-z^{-FOSR})\cdot\ (1-z^{-FOSR})\ldots]$$

- Final equation:

$$H(z) = \frac{Y(z)}{X(z)} = \left(\frac{1-z^{-FOSR}}{1-z^{-1}}\right)^{FORD}$$

The down-sampling is performed at the output of the final integrator stage, allowing to replace each comb stage with a unity delay differentiator at the down-sampled rate (see *Figure 15*).

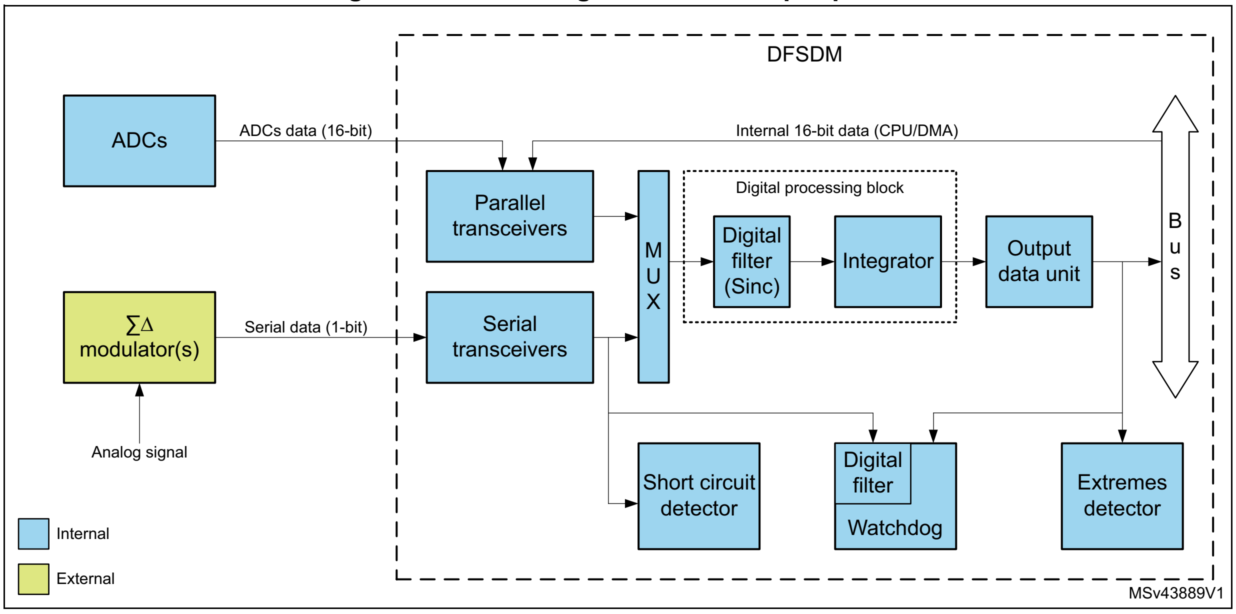
Note: The Sinc filter implementation can be extended to multi-bit width input signal (for example for parallel data input instead of serial data input). In this case the bit-width of flip flops and adders should be extended.

4 DFSDM peripheral operation

4.1 Block diagram

The block diagram of the DFSDM peripheral with all the internal functional blocks and their internal and external connections is provided on *Figure 16*. The DFSDM cannot be reduced to only a digital filter but consists in a complete digital peripheral that handles the overall A/D conversion (when associated with an external sigma-delta modulator).

Figure 16. Block diagram of DFSDM peripheral



4.2 DFSDM components

4.2.1 Serial transceivers

The “serial transceivers” block receives serial data from the external sigma-delta modulator. It features SPI and Manchester serial protocol formats with configurable rising/falling sampling clock edge in order to support most of the sigma-delta modulator types. It supports as well the PDM signal format used by digital microphones - see *Figure 28: MEMS microphone connection to DFSDM (stereo support)*.

Up to two digital microphones (configured as stereo microphone) can be connected to one DFSDM_DATINy pin. In this case, each microphone is configured to be sensitive to a different sampling clock edge (the signals from both microphones are present on a single DFSDM_DATIN data line). In order to distribute this composite signal to two different channels, both channels must be configured to take their serial input from the same pins (DFSDM_DATINy, DFSDM_CKINy) where the two microphones are physically connected (see *Figure 28: MEMS microphone connection to DFSDM (stereo support)*). Each channel is then configured to sample the data on a different sampling edge.

An external clock signal can be connected to DFSDM_CKINy pin (assuming sigma-delta modulator provides this clock signal) or the clock signal can be taken from the internal clock generator (assuming sigma-delta modulator needs external clock signal). The internal clock generator drives the sigma-delta modulator through DFSDM_CKOUT pin. The Manchester format protocol does not require external clock signal because this protocol is single-wire (on DFSDM_DATINy pin) and clock signal is reconstructed from Manchester coded stream. There is a clock signal presence detector that can be used in case of missing clock (external hardware failure) to trigger an interrupt.

The serial transceiver provides also a "Pulse skipper". The Pulse skipper allows to pause the output of the serial transceivers during a given number of sampling clock pulses (given count of 1-bit data samples are discarded). In practice, the "Pulse skipper" is used in beamforming applications in order to delay one channel data with respect to another channel data. Beamforming is a technique that consists in sensing signals (sounds) from a preferred direction by using an array of detectors (microphones). A delay is injected to each microphone signal relative to the previous microphone signal. The delay defines the preferred sensing angle.

4.2.2 Parallel transceivers

The parallel transceiver is an internal 16-bit parallel input register which can be accessed through the APB bus by the CPU, the DMA, or directly by the internal ADC. Its function is to allow post processing (filtering) of internal signals. Examples of usage:

- Filtering data captured by internal ADC
- Filtering data captured by SPI peripheral (through memory buffer)
- Filtering any 16-bit data stored in memory buffer (DMA transfer to DFSDM parallel transceiver).

4.2.3 Digital filter

The digital filter is a key component that processes the input data. The digital filter is configurable to support final application needs (speed, resolution). The following parameters can be configured:

- Filter order: Sinc1 ... Sinc5, FastSinc
- Oversampling ratio:
  - FOSR = 1 … 1024 (for Sinc1 … Sinc3, FastSinc)
  - FOSR = 1 … 215 (for Sinc4)
  - FOSR = 1 … 73 (for Sinc5)

The FOSR ranges provided above take into account the filter order and the filter internal resolution (32-bit width) in order to avoid overflow (case of a 1-bit input signal, for multi-bit parallel input signal the ranges are reduced).

4.2.4 Integrator

The optional integrator works as a simple adder. It sums up a given number of samples provided by the filter output. The number of filter output samples that are summed to provide the single integrator output is configurable in the range IOSR = 1 … 256. The integrator works like a Sinc1 filter (it is performing the average of a given number of samples).

4.2.5 Output data unit

The output data unit performs a final correction that consists in shifting the bits to the right and applying an offset correction on the data coming from the integrator.

Shifting bits to the right is used to:

- fit the 32-bit internal output into the final 24-bit register
- limit even more the final resolution (to 16-bit for instance in case of audio data)

The offset correction allows to calibrate the external sigma-delta modulator offset error. The user configures the offset register with a signed 24-bit correction, and this register is automatically added to the output result. The offset correction value is usually the result of a calibration routine embedded within the microcontroller software that performs the offset calibration calculation and stores the correction into the offset register.

All operations in the DFSDM peripheral are in signed format (filtering, integration, offset correction, right bit shift).

4.2.6 Analog watchdog

The analog watchdog function is the same as in the ADC peripheral. Its purpose is to control that the ADC data excursion stays within given limits by triggering the microcontroller CPU (interrupt) or other peripherals (break signal) when the signal exceeds predefined thresholds.

Not only the DFSDM output data but also the raw data from the sigma-delta modulator (serial transceiver output) can be monitored.

The analog watchdog can monitor serial data directly from the serial transceiver through a dedicated configurable digital filter (FOSR = 1…32, FORD = 1…3). This gives the user the ability to find the best compromise between monitoring speed and monitoring resolution, independently from the main data conversion speed. Some applications require that the reaction time to an input signal exceeding the thresholds, be faster than the main conversion speed. The monitoring of the thresholds for the main conversion itself is usually more precise but slower.

4.2.7 Short circuit detector

The short circuit detector is designed for very fast detection of sudden analog input signal saturation, when the signal is over or under the maximum allowed range. In normal situation the analog input signal gain is controlled (current or voltage sensing loop) in order to never reach such saturation levels. In some extreme situations (like a short circuit) the sensed signal measurements can exceed the operating range limits. In this case the reaction time must be as fast as possible (to switch off the supply for instance).

This very fast detection is implemented in the short circuit detector. The detection of signal saturation is based on the analysis of the 1-bit data stream coming from the serial transceiver. A typical sigma-delta modulator outputs a 1-bit signal with frequent transitions between ‘0’ and ‘1’ (to reproduce the analog signal fluctuations). When a saturation of the input signal occurs the output from the sigma-delta modulator gets stuck with either long series of logical ‘0’ (signal under negative threshold) or long series of logical ‘1’ (signal above positive threshold). If this situation exceeds a certain duration the short circuit detector triggers a "short circuit" event. The duration threshold to trigger a detection can be set in the DFSDM short circuit detector threshold setting (SCDT) in the range 1 to 256 input sample counts. For example if SCDT=100 the short circuit detector event is triggered as soon as at least 100 consecutive ‘0’ or ‘1’ are detected in the input 1-bit data stream. The short circuit detector can trigger an interrupt (software intervention) or activate a break signal (fast hardware intervention).

This type of short-circuit detection is much faster than the detection based on analog watchdog involving digital filtering over longer series of 1-bit input samples.

4.2.8 Extremes detector

The extremes detector simply analyzes the final output data samples and stores the minimum and maximum values into extremes registers. By reading those extremes registers it is possible to monitor the maximum and minimum level of the converted signal during a period of time. Those levels are useful inputs for software post processing, for example for signal normalization or for automatic gain control. Each time the extremes registers are read, they are re-initialized with their reset values and the extremes detection is restarted.

4.3 DFSDM simulation

The sigma-delta modulator and the DFSDM internal blocks behavior are modeled within *[TUTORIAL]*. The incidence of the input signal and the influence of the DFSDM parameters can be simulated and both the internal and output signals can be observed and analyzed.

The proposed simulations cover:

- Sigma-delta modulator (principle of operation demonstration)
- Sinc filter + integrator (digital filter functionality demonstration)
- Frequency characteristics of Sinc filters (LP filters shape demonstration)
- FFT of PWM and sigma-delta signals (noise shaping demonstration)
- High order filters operation (demonstration of resolution increase by multiple signal average)
- Delta-sigma modulator (DAC converter)

4.3.1 Sigma-delta modulator principle

This simulation is provided in the first worksheet of *[TUTORIAL]*.

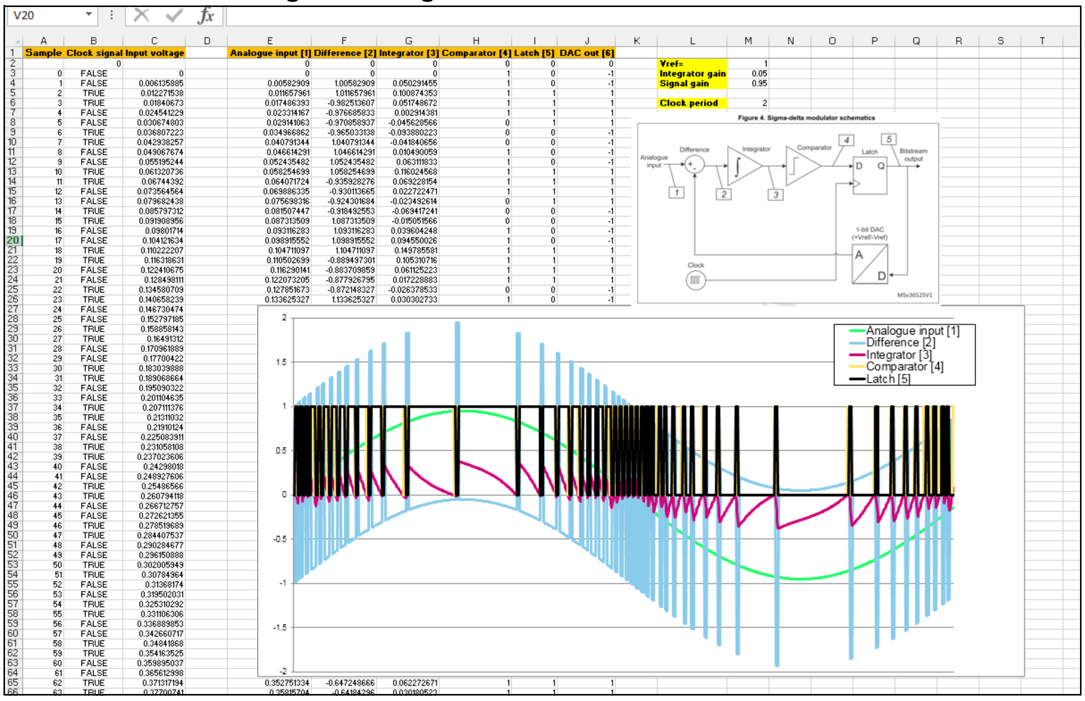
The simulation of the first order sigma-delta principle is based on the schematic provided in *Figure 4: Sigma-delta modulation principle* where each block has been modeled. The input signal to the modulator corresponds to one period of a sinewave signal. For each observation point (as referenced on *Figure 4*) corresponds a curve in the chart provided in *Figure 17*.

The user can modify the input signal shape or modify some parameters of the modulator (integrator gain, V_{ref} level) and visualize the impact on the digital output of the sigma-delta modulator.

This simulation illustrates the principle of the sigma-delta modulator and allows to visualize the internal signal at different stages of the processing.

Here the sigma-delta modulator is modeled as a first order, in order to explain the basic principle. In practice most sigma-delta modulators are using 2nd order with the exception of digital microphones that are typically using 4th order.

Figure 17. Sigma-delta modulator simulation



	A	B	C	D	E	F	G	H	I	J
1	Sample	Clock signal	Input voltage		Analogue input [1]	Difference [2]	Integrator [3]	Comparator [4]	Latch [5]	DAC out [6]
2		0			0	0	0	0	0	0
3	0	FALSE	0		0	0	0	1	0	-1
4	1	FALSE	0.006135885		0.00582909	1.00582909	0.050291455	1	0	-1
5	2	TRUE	0.012271538		0.011657961	1.011657961	0.100874353	1	1	1
6	3	TRUE	0.01840673		0.017486393	-0.982513607	0.051748672	1	1	1
7	4	FALSE	0.024541229		0.023314167	-0.976685833	0.002914381	1	1	1
8	5	FALSE	0.030674803		0.029141063	-0.970858937	-0.045628566	0	1	1
9	6	TRUE	0.036807223		0.034966862	-0.965033138	-0.093880223	0	0	-1
10	7	TRUE	0.042938257		0.040791344	1.040791344	-0.041840656	0	0	-1
11	8	FALSE	0.049067674		0.046614291	1.046614291	0.010490059	1	0	-1
12	9	FALSE	0.055195244		0.052435482	1.052435482	0.063111833	1	0	-1
13	10	TRUE	0.061320736		0.058254699	1.058254699	0.116024568	1	1	1
14	11	TRUE	0.06744392		0.064071724	-0.935928276	0.069228154	1	1	1
15	12	FALSE	0.073564564		0.069886335	-0.930113665	0.022722471	1	1	1
16	13	FALSE	0.079682438		0.075698316	-0.924301684	-0.023492614	0	1	1
17	14	TRUE	0.085797312		0.081507447	-0.918492553	-0.069417241	0	0	-1
18	15	TRUE	0.091908956		0.087313509	1.087313509	-0.015051566	0	0	-1
19	16	FALSE	0.09801714		0.093116283	1.093116283	0.039604248	1	0	-1
20	17	FALSE	0.104121634		0.098915552	1.098915552	0.094550026	1	0	-1
21	18	TRUE	0.110222207		0.104711097	1.104711097	0.149785581	1	1	1
22	19	TRUE	0.116318631		0.110502699	-0.889497301	0.105310716	1	1	1
23	20	FALSE	0.122410675		0.116290141	-0.883709859	0.061125223	1	1	1
24	21	FALSE	0.128498111		0.122073205	-0.877926795	0.017228883	1	1	1
25	22	TRUE	0.134580709		0.127851673	-0.872148327	-0.026378533	0	0	-1
26	23	TRUE	0.140658239		0.133625327	1.133625327	0.030302733	1	0	-1
27	24	FALSE	0.146730474							
28	25	FALSE	0.152797185							
29	26	TRUE	0.158858143							
30	27	TRUE	0.16491312							
31	28	FALSE	0.170961889							
32	29	FALSE	0.17700422							
33	30	TRUE	0.183039888							
34	31	TRUE	0.189068664							
35	32	FALSE	0.195090322							
36	33	FALSE	0.201104635							
37	34	TRUE	0.207111376							
38	35	TRUE	0.21311032							
39	36	FALSE	0.21910124							
40	37	FALSE	0.225083911							
41	38	TRUE	0.231058108							
42	39	TRUE	0.237023606							
43	40	FALSE	0.24298018							
44	41	FALSE	0.248927606							
45	42	TRUE	0.25486566							
46	43	TRUE	0.260794118							
47	44	FALSE	0.266712757							
48	45	FALSE	0.272621355							
49	46	TRUE	0.278519689							
50	47	TRUE	0.284407537							
51	48	FALSE	0.290284677							
52	49	FALSE	0.296150888							
53	50	TRUE	0.302005949							
54	51	TRUE	0.30784964							
55	52	FALSE	0.31368174							
56	53	FALSE	0.319502031							
57	54	TRUE	0.325310292							
58	55	TRUE	0.331106306							
59	56	FALSE	0.336889853							
60	57	FALSE	0.342660717							
61	58	TRUE	0.34841868							
62	59	TRUE	0.354163525							
63	60	FALSE	0.359895037							
64	61	FALSE	0.365612998							
65	62	TRUE	0.371317194		0.352751334	-0.647248666	0.062272671	1	1	1
66	63	TRUE	0.37700741		0.35815704	-0.64184296	0.030180523	1	1	1

	L	M
2	Vref=	1
3	Integrator gain	0.05
4	Signal gain	0.95
5		
6	Clock period	2

4.3.2 DFSDM filtering simulation (filter and integrator)

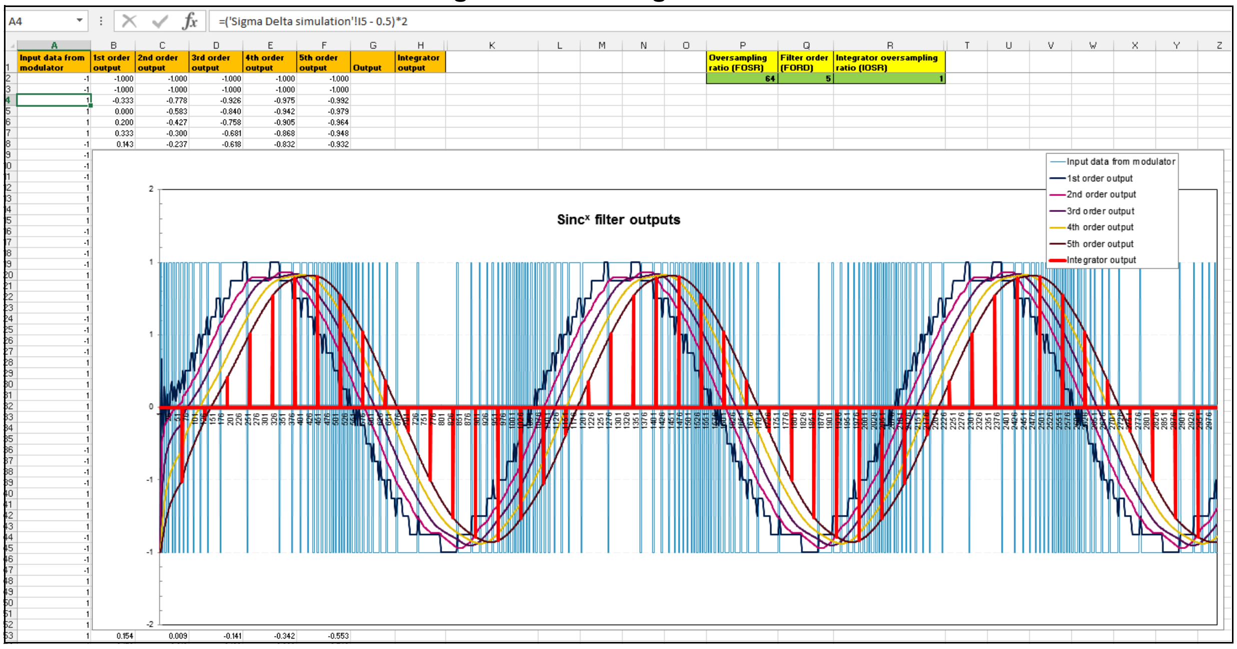
This simulation is provided in the second worksheet of *[TUTORIAL]*. This part of the DFSDM simulator corresponds to the Sinc filter and the integrator stages (see *Section 4.2.3: Digital filter* and *Section 4.2.4: Integrator* for details).

Both the Sinc filter and the integrator models are built within *[TUTORIAL]* and can be simulated. The user can change the FORD and FOSR parameters for the filter and the IOSR parameter for the integrator. The simulation results are identical to the results obtained with the actual DFSDM block. The user can tune in simulation the filter and the integrator parameters according to his application and observe the impact on the output signal shape without waiting for the actual prototypes.

This simulation requires a digital signal input (1-bit data stream) from the sigma-delta modulator (like in real application). In this particular simulation, the input is provided by the results of the first order sigma-delta modulator simulation also available in *[TUTORIAL]* (see *Section 4.3.1: Sigma-delta modulator principle*). By doing so, it is possible to compare the output digital signal (DFSDM filtering simulation curve) with the input analog signal applied to the sigma-delta modulator. The filter parameters should then be adjusted in order to reach the application requirements in term of acceptable error between the analog input signal into sigma-delta modulator versus the output digitized signal from the DFSDM (final output samples).

Figure 18 provides not only the output curve (FORD = 5) but also the internal data results at the output of each filter order. It shows the resolution gain that is obtained by performing more moving average loops.

Figure 18. Filtering simulation


A4 =('Sigma Delta simulation'!I5 - 0.5)*2

	A	B	C	D	E	F	G	H
1	Input data from modulator	1st order output	2nd order output	3rd order output	4th order output	5th order output	Output	Integrator output
2	-1	-1.000	-1.000	-1.000	-1.000	-1.000		
3	-1	-1.000	-1.000	-1.000	-1.000	-1.000		
4	1	-0.333	-0.778	-0.926	-0.975	-0.992		
5	1	0.000	-0.583	-0.840	-0.942	-0.979		
6	1	0.200	-0.427	-0.758	-0.905	-0.964		
7	1	0.333	-0.300	-0.681	-0.868	-0.948		
8	-1	0.143	-0.237	-0.618	-0.832	-0.932		
9	-1							
10	-1							
11	-1							
12	1							
13	1							
14	1							
15	1							
16	-1							
17	-1							
18	-1							
19	-1							
20	1							
21	1							
22	1							
23	1							
24	-1							
25	-1							
26	-1							
27	-1							
28	1							
29	1							
30	1							
31	1							
32	1							
33	1							
34	1							
35	1							
36	-1							
37	-1							
38	-1							
39	-1							
40	1							
41	1							
42	1							
43	1							
44	-1							
45	-1							
46	-1							
47	-1							
48	1							
49	1							
50	1							
51	1							
52	1							
53	1	0.154	0.009	-0.141	-0.342	-0.553		

Oversampling ratio (FOSR)	Filter order (FORD)	Integrator oversampling ratio (IOSR)
64	5	1

4.3.3 Frequency characteristics for Sinc filter

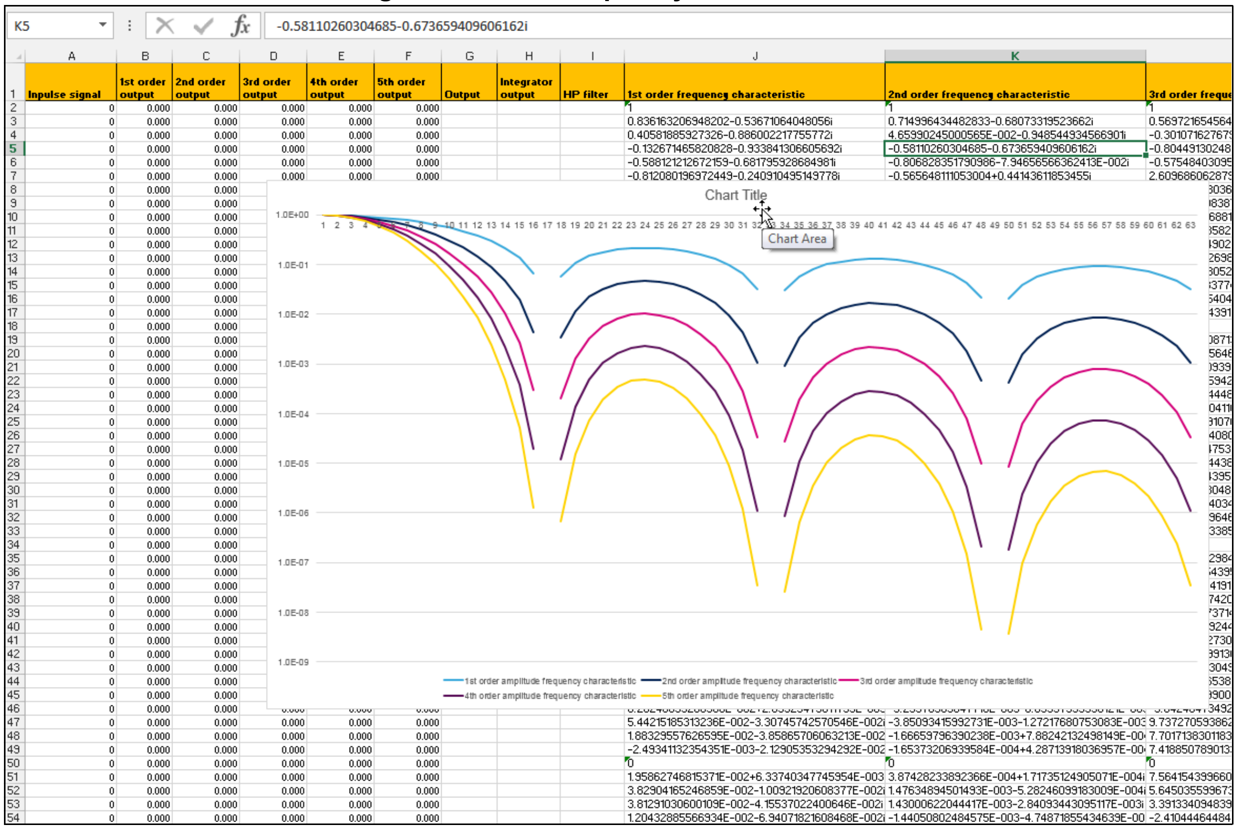
This simulation is provided in the third worksheet of *[TUTORIAL]*. It allows to visualize the frequency characteristics of the Sinc filter depending on the filter order.

The frequency characteristics of a filter can be determined by applying an impulse signal at the input of the filter and by computing the FFT transform of the impulse response at the output of the filter. This is how the filter characteristics have been computed in the simulation model. The model calculates the impulse response of the filter and submit the result to a 512-point FFT transform. The results for different filter orders are available in *[TUTORIAL]* and provided in *Figure 19*.

The frequency response has a comb shape with periodic attenuation points (notches - also called "zeroes") and a decreasing trend for higher frequencies (low pass filtering). The frequency of the first notch depends on the selected FOSR and is given by $f_{sampling}$/FOSR. In case of continuous sampling, $f_{sampling}$/FOSR is also the output data rate frequency. The frequency components of the input signal corresponding to the notches are completely rejected by the filter. This property can be used in some applications to remove external noise from the input signal at predetermined frequencies. As an example, the long wires that sometime are necessary to connect distant sensors to the microcontroller are prone to collect noise from the power network (50/60Hz) that can be filtered according the above property.

Another property is that the attenuation of the higher frequencies (other than notches which frequency depends only on FOSR) is proportional to the filter order (FORD). So the higher the filter order, the more rejection on higher frequencies. In applications which are sensing quasi static signals (like temperature or pressure sensors) a high order filter (FORD) is recommended as well as a high oversampling ratio (FORD) to suppress noise from AC perturbations. A further filtering of quasi static signals can be done by using the integrator located after the filter. Increasing the integrator oversampling ratio (IOSR) allows to performs additional averaging of the signal. In practice, the user should first set a (high) FORD and then set correctly the FOSR and the IOSR to suppress the noise from mains frequency (50/60Hz). The first notch of the overall block consisting of [Sinc filter + integrator] is at frequency: $f_{sampling}/(FOSR*IOSR)$.

Figure 19. Filter frequency characteristics

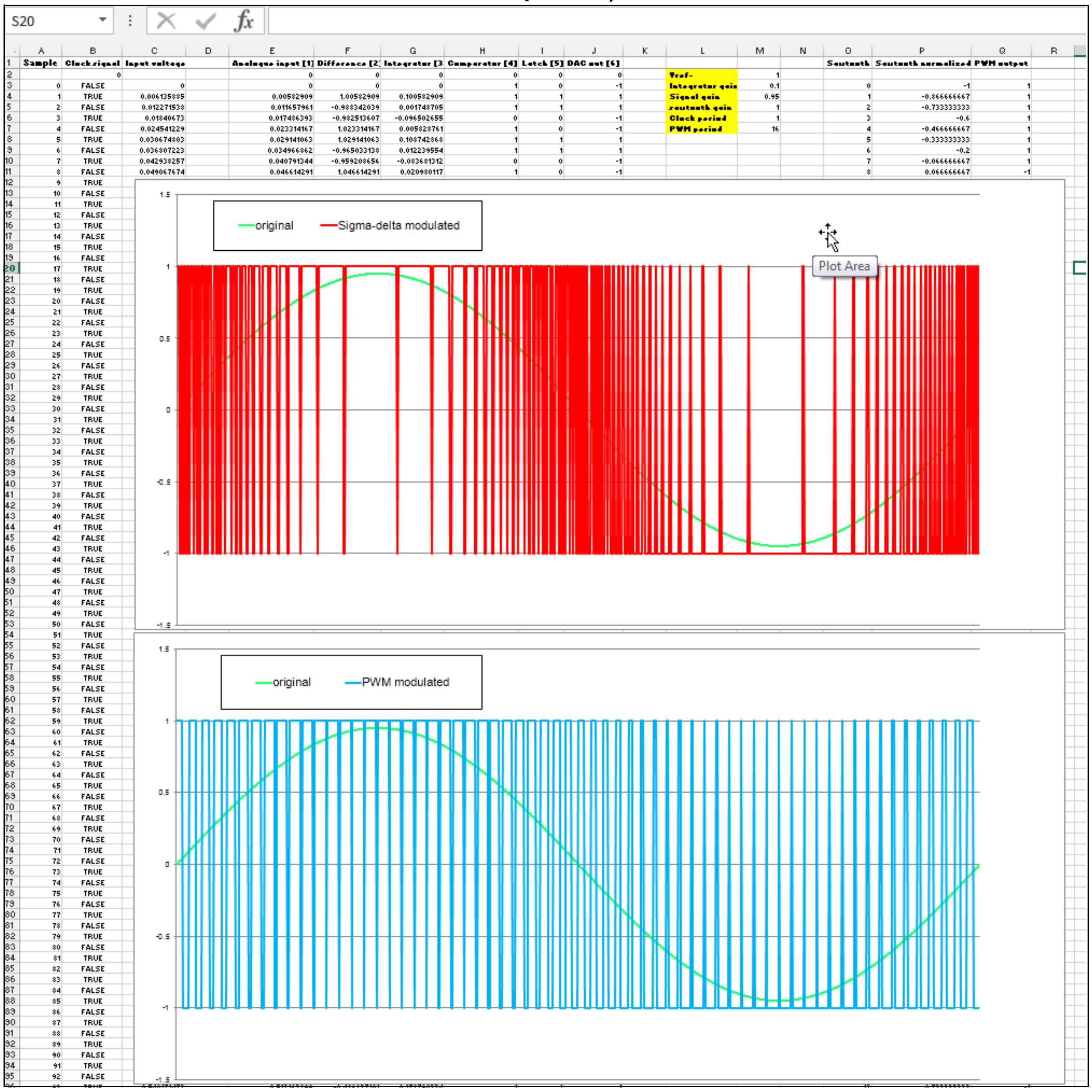


	A	B	C	D	E	F	G	H	I	J	K	
1	Inpulse signal	1st order output	2nd order output	3rd order output	4th order output	5th order output	Output	Integrator output	HP filter	1st order frequency characteristic	2nd order frequency characteristic	3rd order freque
2	0	0.000	0.000	0.000	0.000	0.000				1	1	1
3	0	0.000	0.000	0.000	0.000	0.000				0.836163206948202-0.536710640480561	0.714996434482833-0.68073319523662i	0.569721654564
4	0	0.000	0.000	0.000	0.000	0.000				0.40581885927326-0.886002217755772i	4.65990245000565E-002-0.948544934566901i	-0.30107162767
5	0	0.000	0.000	0.000	0.000	0.000				-0.132671465820828-0.933841306605692i	-0.58110260304685-0.673659409606162i	-0.80449130248
6	0	0.000	0.000	0.000	0.000	0.000				-0.588121212672159-0.681795928684981i	-0.806828351790986-7.94656566362413E-002i	-0.57548403095
7	0	0.000	0.000	0.000	0.000	0.000				-0.812080196972449-0.240910495149778i	-0.565648111053004+0.44143611853455i	2.60968606287
8	0	0.000	0.000									3036
9	0	0.000	0.000									838
10	0	0.000	0.000									6881
11	0	0.000	0.000									3582
12	0	0.000	0.000									4902
13	0	0.000	0.000									2698
14	0	0.000	0.000									3052
15	0	0.000	0.000									3774
16	0	0.000	0.000									5404
17	0	0.000	0.000									4391
18	0	0.000	0.000									
19	0	0.000	0.000									0871
20	0	0.000	0.000									5646
21	0	0.000	0.000									0939
22	0	0.000	0.000									5942
23	0	0.000	0.000									4448
24	0	0.000	0.000									0411
25	0	0.000	0.000									3107
26	0	0.000	0.000									4080
27	0	0.000	0.000									4753
28	0	0.000	0.000									4438
29	0	0.000	0.000									4395
30	0	0.000	0.000									3048
31	0	0.000	0.000									4034
32	0	0.000	0.000									9646
33	0	0.000	0.000									3385
34	0	0.000	0.000									
35	0	0.000	0.000									2984
36	0	0.000	0.000									4439
37	0	0.000	0.000									4191
38	0	0.000	0.000									7420
39	0	0.000	0.000									7371
40	0	0.000	0.000									9244
41	0	0.000	0.000									2730
42	0	0.000	0.000									3913
43	0	0.000	0.000									3049
44	0	0.000	0.000									6538
45	0	0.000	0.000									3900
46	0	0.000	0.000	0.000	0.000	0.000				[illegible]	[illegible]	[illegible]3492
47	0	0.000	0.000	0.000	0.000	0.000				5.44215185313236E-002-3.30745742570546E-002i	-3.85093415992731E-003-1.27217680753083E-003	9.737270593862
48	0	0.000	0.000	0.000	0.000	0.000				1.88329557626595E-002-3.85865706063213E-002	-1.66659796390238E-003+7.88242132498149E-00	7.7017138301183
49	0	0.000	0.000	0.000	0.000	0.000				-2.49341132354351E-003-2.12905353294292E-002	-1.65373206939584E-004+4.28713918036957E-00	7.418850789013
50	0	0.000	0.000	0.000	0.000	0.000				0	0	0
51	0	0.000	0.000	0.000	0.000	0.000				1.95862746815371E-002+6.33740347745954E-003	3.87428233892366E-004+1.71735124905071E-004i	7.564154399660
52	0	0.000	0.000	0.000	0.000	0.000				3.82904165246859E-002-1.00921920608377E-002i	1.47634894501493E-003-5.28246099183009E-004i	5.645035599673
53	0	0.000	0.000	0.000	0.000	0.000				3.81291030600109E-002-4.15537022400646E-002i	1.43000622044417E-003-2.84093443095117E-003i	3.391334094839
54	0	0.000	0.000	0.000	0.000	0.000				1.20432885566934E-002-6.94071821608468E-002i	-1.44050802484575E-003-4.74871855434639E-00	-2.41044464484

4.3.4 Noise shaping of sigma-delta modulation

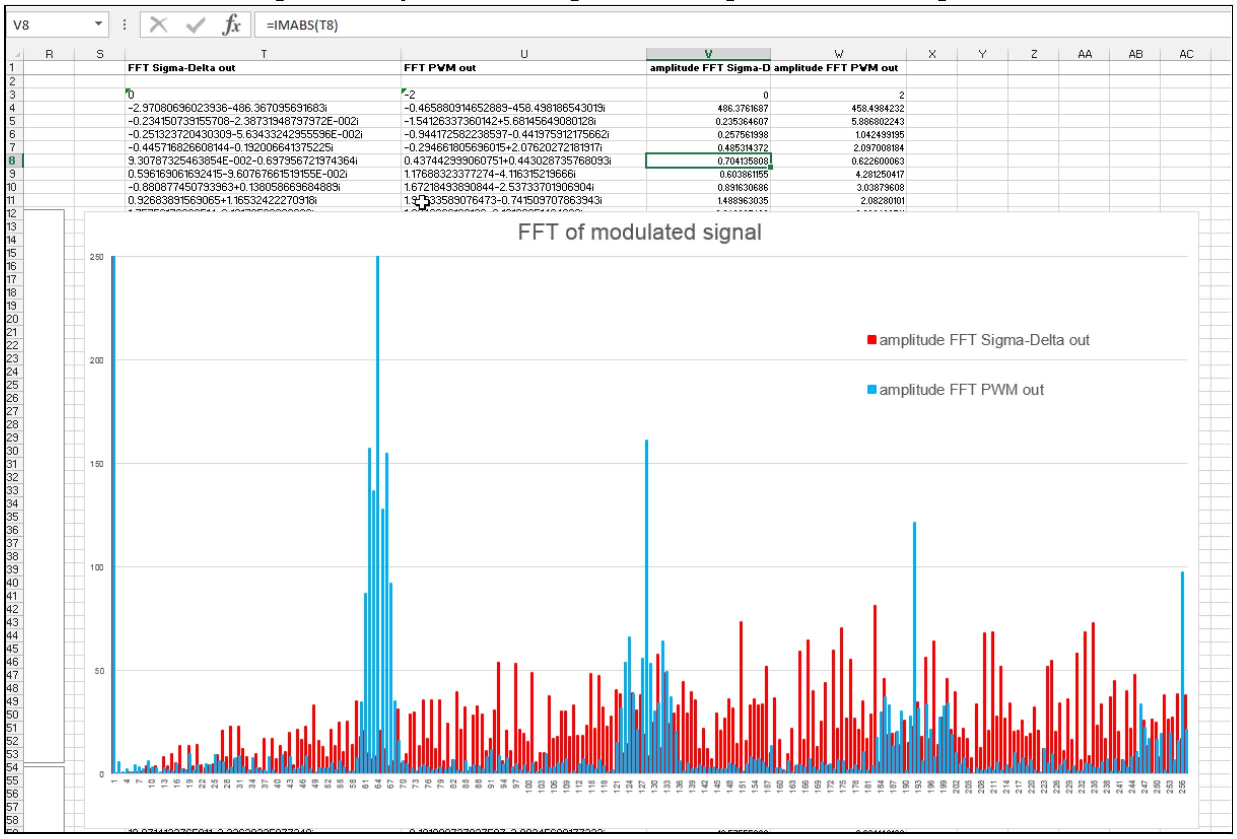
This simulation is provided in the fourth and the fifth worksheets of *[TUTORIAL]*. It compares the spectrum of a PWM modulated signal with the spectrum of a sigma-delta modulated signal and highlights the benefits of the later with respect to noise shaping. The same input signal (one period of a sinus wave, see *Figure 20*) is either sigma-delta modulated or PWM modulated before undergoing an FFT transform. The frequency spectrum of both the PWM modulated signal and the sigma-delta modulated signal are compared in *Figure 21*. The X-axis represents the frequency in multiple of the base signal frequency (signal base frequency and its harmonics), the Y-axis represent the spectrum density. Logically, the amplitude of the spectrum at index 1 (signal base frequency) is very large (~470). All the other peaks located at indexes > 1 would have to be filtered out in order to reconstruct the original signal.

Figure 20. Sigma-delta and PWM modulated signals (for frequency spectrum comparison)



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	Sample	Clock signal	Input voltage		Analogue input [1]	Difference [2]	Integrator [3	Comparator [4]	Latch [5]	DAC out [6]					Sawtooth	Sawtooth normalized	PWM output
2		0			0	0	0	0	0	0		Vref-	1				
3	0	FALSE	0		0	0	0	1	0	-1		Integrator gain	0.1		0	-1	1
4	1	TRUE	0.006135885		0.00582909	1.00582909	0.100582909	1	1	1		Signal gain	0.95		1	-0.866666667	1
5	2	FALSE	0.012271538		0.011657961	-0.988342039	0.001748705	1	1	1		sawtooth gain	1		2	-0.733333333	1
6	3	TRUE	0.01840673		0.017486393	-0.982513607	-0.096502655	0	0	-1		Clock period	1		3	-0.6	1
7	4	FALSE	0.024541229		0.023314167	1.023314167	0.005828761	1	0	-1		PWM period	16		4	-0.466666667	1
8	5	TRUE	0.030674803		0.029141063	1.029141063	0.108742868	1	1	1					5	-0.333333333	1
9	6	FALSE	0.036807223		0.034966862	-0.965033138	0.012239554	1	1	1					6	-0.2	1
10	7	TRUE	0.042938257		0.040791344	-0.959208656	-0.083681312	0	0	-1					7	-0.066666667	1
11	8	FALSE	0.049067674		0.046614291	1.046614291	0.020980117	1	0	-1					8	0.066666667	-1
12	9	TRUE															
13	10	FALSE															
14	11	TRUE															
15	12	FALSE															
16	13	TRUE															
17	14	FALSE															
18	15	TRUE															
19	16	FALSE															
20	17	TRUE															
21	18	FALSE															
22	19	TRUE															
23	20	FALSE															
24	21	TRUE															
25	22	FALSE															
26	23	TRUE															
27	24	FALSE															
28	25	TRUE															
29	26	FALSE															
30	27	TRUE															
31	28	FALSE															
32	29	TRUE															
33	30	FALSE															
34	31	TRUE															
35	32	FALSE															
36	33	TRUE															
37	34	FALSE															
38	35	TRUE															
39	36	FALSE															
40	37	TRUE															
41	38	FALSE															
42	39	TRUE															
43	40	FALSE															
44	41	TRUE															
45	42	FALSE															
46	43	TRUE															
47	44	FALSE															
48	45	TRUE															
49	46	FALSE															
50	47	TRUE															
51	48	FALSE															
52	49	TRUE															
53	50	FALSE															
54	51	TRUE															
55	52	FALSE															
56	53	TRUE															
57	54	FALSE															
58	55	TRUE															
59	56	FALSE															
60	57	TRUE															
61	58	FALSE															
62	59	TRUE															
63	60	FALSE															
64	61	TRUE															
65	62	FALSE															
66	63	TRUE															
67	64	FALSE															
68	65	TRUE															
69	66	FALSE															
70	67	TRUE															
71	68	FALSE															
72	69	TRUE															
73	70	FALSE															
74	71	TRUE															
75	72	FALSE															
76	73	TRUE															
77	74	FALSE															
78	75	TRUE															
79	76	FALSE															
80	77	TRUE															
81	78	FALSE															
82	79	TRUE															
83	80	FALSE															
84	81	TRUE															
85	82	FALSE															
86	83	TRUE															
87	84	FALSE															
88	85	TRUE															
89	86	FALSE															
90	87	TRUE															
91	88	FALSE															
92	89	TRUE															
93	90	FALSE															
94	91	TRUE															
95	92	FALSE															

Figure 21. Spectrum of sigma-delta signal and PWM signal



V8 =IMABS(T8)

	T	U	V	W
1	FFT Sigma-Delta out	FFT PWM out	amplitude FFT Sigma-D	amplitude FFT PWM out
2				
3	0	-2	0	2
4	-2.97080696023936-486.367095691683i	-0.465880914652889-458.498186543019i	486.3761687	458.4984232
5	-0.234150739155708-2.38731948797972E-002i	-1.54126337360142+5.68145649080128i	0.235364607	5.886802243
6	-0.251323720430309-5.63433242955596E-002i	-0.944172582238597-0.441975912175662i	0.257561998	1.042499195
7	-0.445716826608144-0.192006641375225i	-0.294661805696015+2.07620272181917i	0.485314372	2.097008184
8	9.30787325463854E-002-0.697956721974364i	0.437442999060751+0.443028735768093i	0.704135808	0.622600063
9	0.596169061692415-9.60767661519155E-002i	1.17688323377274-4.116315219666i	0.603861155	4.281250417
10	-0.880877450793963+0.138058669684889i	1.67218493890844-2.53733701906904i	0.891630686	3.03879608
11	0.92683891569065+1.16532422270918i	1.9[illegible]33589076473-0.741509707863943i	1.488963035	2.08280101

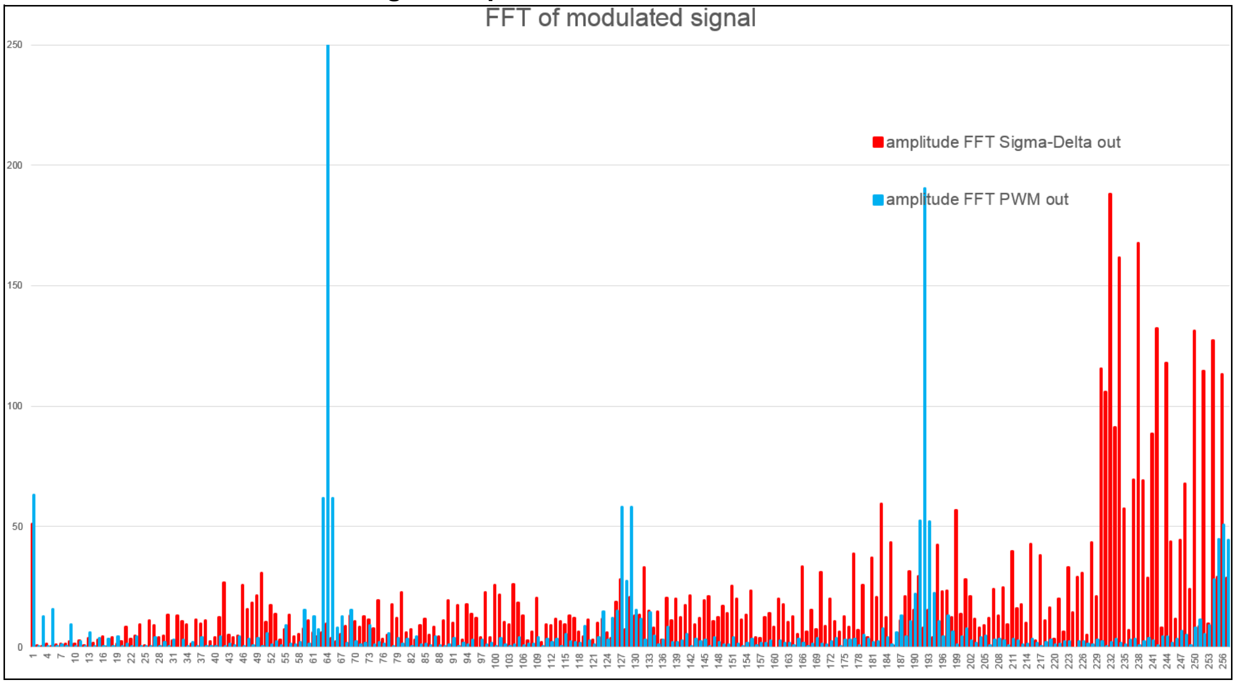
Figure 21 illustrates the fact that a PWM signal spectrum concentrates its energy on peaks located around multiple of the modulation frequency (modulation frequency = 16 x signal base frequency in this case). This is due to the characteristics of a PWM modulation that has a constant modulation frequency and a variable duty cycle. The amplitude of the peaks decreases with frequency increase.

On the opposite a 1st order sigma-delta modulation spectrum shows an amplitude that is low on the first harmonics, and then increases progressively with the frequency.

As a consequence of the previous observations, the requirements on the low-pass analog filter that is necessary to reconstruct the original signal (sinewave signal at index 1 on *Figure 21*) are lower for a sigma-delta modulated signal than for a PWM modulated signal. The low-pass filter (analog filter) must be designed with a cut-off frequency slightly higher than the useful analog bandwidth of the original signal to be observed, and must be sharp enough to sufficiently suppress all the higher frequency components resulting from the signal modulation. Because the PWM signal has more unwanted energy concentrated at lower frequency than the sigma-delta modulated signal, the low pass analog filter must be sharper (higher order analog filter).

Another particularity of the sigma-delta modulated signal is that the quantization noise on the lower part of the spectrum is almost constant and independent from the input signal resolution whereas for a PWM modulated signal, a degradation of the quantization noise affects the entire spectrum, and predominately the first harmonic. This higher sensitivity of the PWM modulated signal to a degradation of the quantization noise is put in evidence by decreasing the input signal magnitude from 95% of full scale (*Figure 21*) down to 10% of full scale (*Figure 22*).

Figure 22. Spectrum of sigma-delta signal and PWM signal with signal amplitude reduced to 10% of full scale



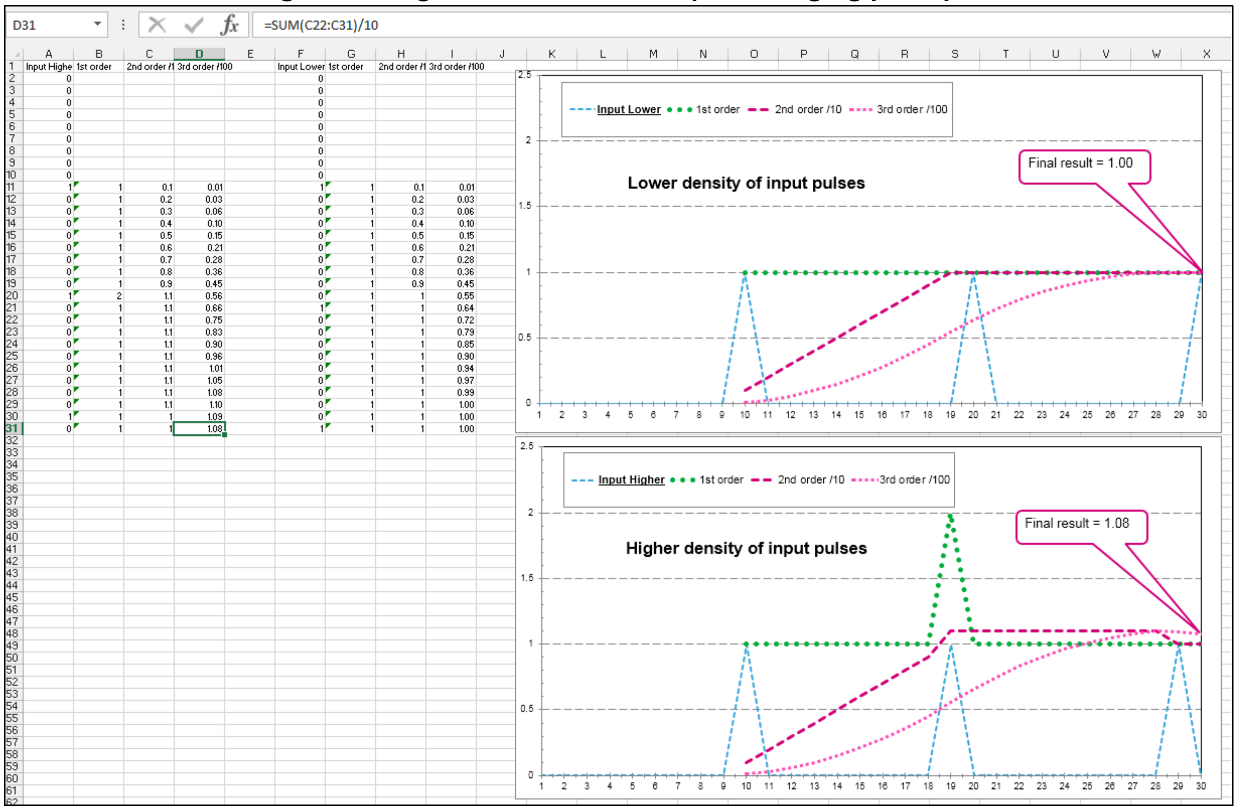
4.3.5 High order filters operation

This simulation is provided in the sixth worksheet of *[TUTORIAL]*. It shows how it is possible to increase the filter output resolution by the technique of multiple averaging of the same data (principle of the high order filters).

For this simulation the averaging period is 10 samples (FOSR=10). Over the averaging period the input signal consists of a single pulse (one sample at 1 and the nine other samples at 0). Multiple averaging is performed. Two different cases are observed (see *Figure 23*):

1. Equidistant pulses occurring at 10th, 20th and 30th sample. This situation correspond to a constant density signal.
2. Non-equidistant pulses occurring at 10th, 19th, 29th sample. This situation correspond to a signal with varying density.

Figure 23. High order filters - multiple averaging principle



D31 =SUM(C22:C31)/10

	A	B	C	D	F	G	H	I
1	Input Highe	1st order	2nd order /1	3rd order /100	Input Lower	1st order	2nd order /1	3rd order /100
2	0				0			
3	0				0			
4	0				0			
5	0				0			
6	0				0			
7	0				0			
8	0				0			
9	0				0			
10	0				0			
11	1	1	0.1	0.01	1	1	0.1	0.01
12	0	1	0.2	0.03	0	1	0.2	0.03
13	0	1	0.3	0.06	0	1	0.3	0.06
14	0	1	0.4	0.10	0	1	0.4	0.10
15	0	1	0.5	0.15	0	1	0.5	0.15
16	0	1	0.6	0.21	0	1	0.6	0.21
17	0	1	0.7	0.28	0	1	0.7	0.28
18	0	1	0.8	0.36	0	1	0.8	0.36
19	0	1	0.9	0.45	0	1	0.9	0.45
20	1	2	1.1	0.56	0	1	1	0.55
21	0	1	1.1	0.66	1	1	1	0.64
22	0	1	1.1	0.75	0	1	1	0.72
23	0	1	1.1	0.83	0	1	1	0.79
24	0	1	1.1	0.90	0	1	1	0.85
25	0	1	1.1	0.96	0	1	1	0.90
26	0	1	1.1	1.01	0	1	1	0.94
27	0	1	1.1	1.05	0	1	1	0.97
28	0	1	1.1	1.08	0	1	1	0.99
29	0	1	1.1	1.10	0	1	1	1.00
30	1	1	1	1.09	0	1	1	1.00
31	0	1	1	1.08	1	1	1	1.00

The results are presented for both input signal cases, and for 1, 2 or 3 averaging (similar to filter orders FORD=1, FORD=2, FORD=3). The final output is sampled at every 10th sample period (end of average period).

In the first case, since the input pulses are at equidistant intervals (every 10th sample), the output (sampled at 10th, 20th, 30th sample) is always 1.00, whatever the filter order.

In the second case, the input pulses are a little bit closer (pulses at 10th, 19th and 29th sample), so the pulse density is slightly higher. It has no impact on the simple averaging filter that provides the same result 1.00 (constant over the 1st, 2nd, 3rd averaging period) as in the first case. This is because it performs averaging over one period only (10 samples). That means the first order filter is not able to detect changes in pulse density (or a filter with higher FOSR must be designed, with slower data rate).

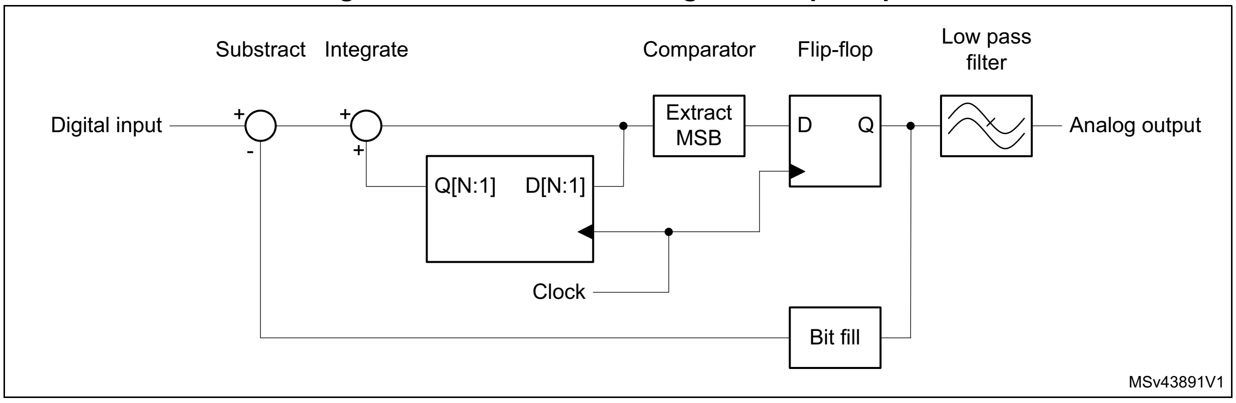
The double averaging is a bit different. The first valid result is present on the output after the 2nd averaging period because the double averaging uses 2 periods of signal to build one final sample. The results from this double averaging are: 1.10 (result after 2nd period) and 1.00 (result after 3rd period). That means the second order filter is able to detect the increase of pulse density during the 2nd averaging period of the signal and the decrease of the pulse density during the 3rd averaging period.

The triple averaging gives even more precision. The first valid result is present on the output after the 3rd averaging period because the triple averaging uses 3 periods of signal to build one final sample. The results from this triple averaging is: 1.08 (result after 3rd period). That means the third order filter gives even more precision on the pulse density.

4.3.6 Delta-sigma DAC simulation

This simulation is provided in the seventh worksheet of *[TUTORIAL]*. The sigma-delta modulation can be used not only for ADC conversion but also for DAC conversion. In this case, all analog blocks of the sigma-delta modulator diagram (see *Figure 4: Sigma-delta modulation principle*) are replaced by their digital equivalent (*Figure 24*).

Figure 24. First order delta-sigma DAC principle

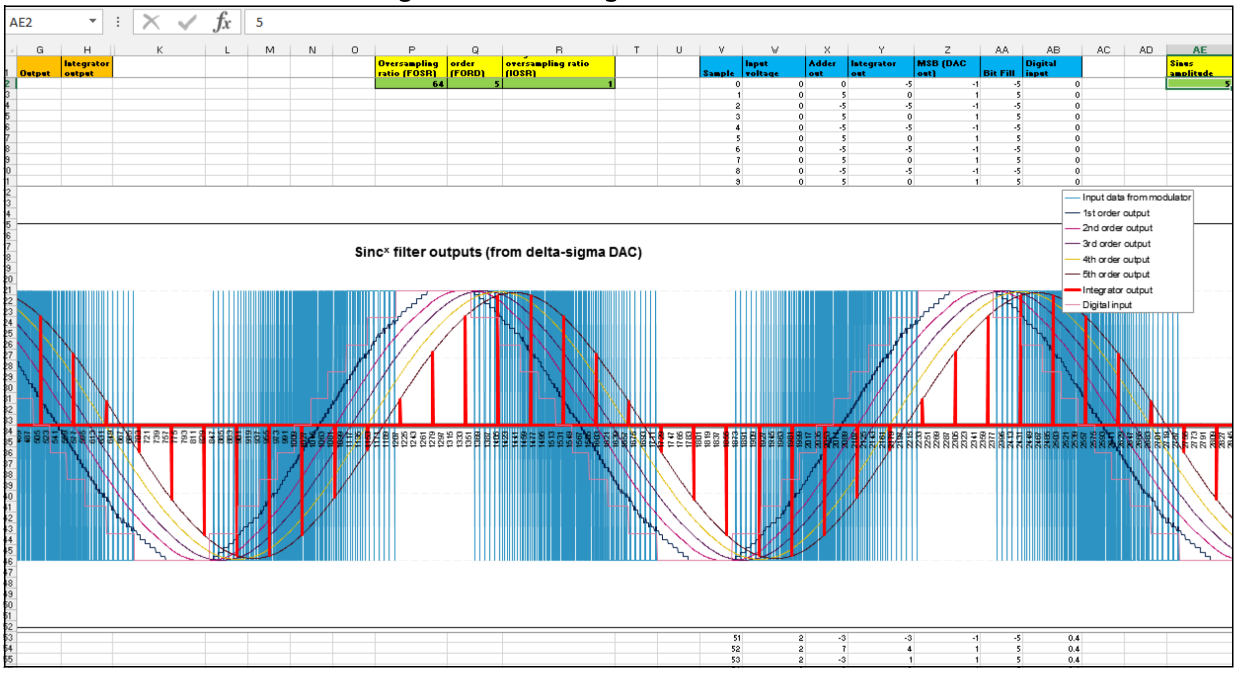


Here are the particularities of each block in the digital domain:

- The “Subtraction” block is a digital operation.
- The “Integrator” block is also a simple accumulate operation (sum of incoming bits).
- The “Comparator” block is a digital comparator. It determines the sign of the digital data word coming from the integrator (most significant bit = sign bit).
- The "Bit fill" block is the equivalent of the 1-bit DAC used in *Figure 4: Sigma-delta modulation principle*. It is replaced by an ADC which output is ±*digital_reference_word* according the sign provided by the comparator output. The magnitude of "*digital_reference_word*” should correspond to the input range of the digital signal into the comparator (equivalent to $+V_{ref}$, $-V_{ref}$ for sigma-delta converter).
- The "Low pass filter" block is an analog filter which performs analog filtering of the fast 1-bit digital stream. The fast 1-bit digital stream represents the sigma-delta signal and in this case is built from the parallel N-bit width digital input signal (delta-sigma modulator). The 1-bit digital stream has the same properties as a 1-bit digital stream in the sigma-delta modulator (noise shaping) and can be filtered with a simpler analog filter (with respect to PWM signal).

The simulation of this delta-sigma DAC gets a digitized sinewave signal with amplitude +/-5 as an input (see "Digital input" pink-color signal on *Figure 25*). In a real application, the fast 1-bit digital stream output (blue signal on *Figure 25*) should undergo an analog low-pass filtering. For the need of the simulation, the analog low pass filtering is replaced by a digital Sincx filter.

Figure 25. Delta-sigma DAC simulation



G: Output	H: Integrator output	P: Oversampling ratio (FOSR)	Q: order (FORD)	R: oversampling ratio (IOSR)	AE: Sinus amplitude
		64	5	1	5

V: Sample	W: Input voltage	X: Adder out	Y: Integrator out	Z: MSB (DAC out)	AA: Bit Fill	AB: Digital input
0	0	0	-5	-1	-5	0
1	0	5	0	1	5	0
2	0	-5	-5	-1	-5	0
3	0	5	0	1	5	0
4	0	-5	-5	-1	-5	0
5	0	5	0	1	5	0
6	0	-5	-5	-1	-5	0
7	0	5	0	1	5	0
8	0	-5	-5	-1	-5	0
9	0	5	0	1	5	0
51	2	-3	-3	-1	-5	0.4
52	2	7	4	1	5	0.4
53	2	-3	1	1	5	0.4

The final delta-sigma DAC output (see the various order outputs on *Figure 25*) show a quite well reconstructed "analog" sinewave despite the digital input sinewave quantization uses only 11 discrete levels [-5,...,+5]).

Once again the advantage of using a sigma-delta versus a PWM modulation is a simpler low-pass filtering (analog in the case of the DAC) of the output signal. This is for the same reason as the one developed in *Section 4.3.4: Noise shaping of sigma-delta modulation*.

4.3.7 High pass filter simulation

This simulation is provided in the eighth worksheet of *[TUTORIAL]*. High-pass filters are useful in many applications where the DC component and/or low frequency noise need to be removed from the input signal (audio applications including static pressure changes in microphone output, AC energy measurement, DC offset which vary with temperature, …).

The DFSDM does not implement any high pass filtering but it can be easily implemented in software as a post processing over the DFSDM sampled data (as an alternative to the use of an external hardware high-pass filter).

The high-pass filter used in the simulation is modeled by the following equation:

$$y(n) = (coeff / 256) \times (y(n-1) + x(n) - x(n-1))$$

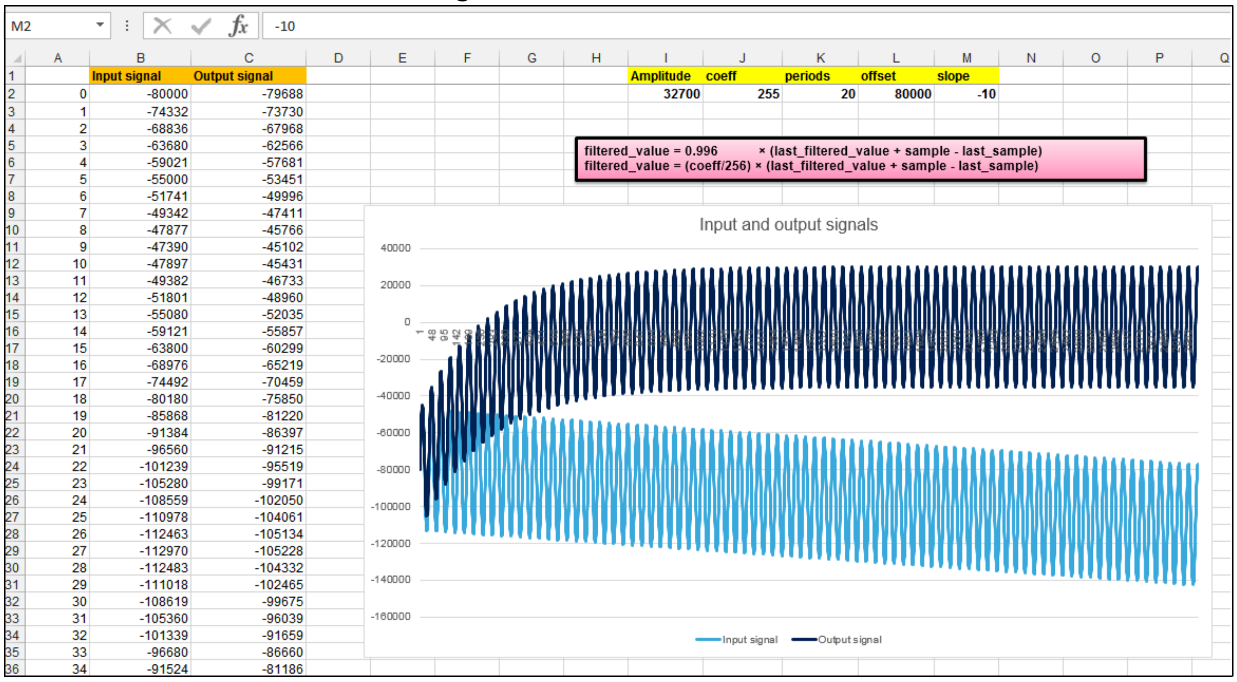
with

y(n):	= "filtered_value" in the simulation (filter output)
coeff:	= "coeff" in the simulation (filter cutoff frequency)
y(n-1)	= "last_filtered_value" in the simulation (previous filter output)
x(n)	= "sample" in the simulation (filter input sample)
x(n-1)	= "last_sample" in the simulation (filter previous input sample)

The complexity of this filter reduces to one multiplication, two additions and one shift operation (/256).

For the simulation a sinewave with a decreasing DC offset is applied to the high-pass filter (light blue curve in *Figure 26*).

Figure 26. HP filter simulation



M2 -10

	Input signal	Output signal
0	-80000	-79688
1	-74332	-73730
2	-68836	-67968
3	-63680	-62566
4	-59021	-57681
5	-55000	-53451
6	-51741	-49996
7	-49342	-47411
8	-47877	-45766
9	-47390	-45102
10	-47897	-45431
11	-49382	-46733
12	-51801	-48960
13	-55080	-52035
14	-59121	-55857
15	-63800	-60299
16	-68976	-65219
17	-74492	-70459
18	-80180	-75850
19	-85868	-81220
20	-91384	-86397
21	-96560	-91215
22	-101239	-95519
23	-105280	-99171
24	-108559	-102050
25	-110978	-104061
26	-112463	-105134
27	-112970	-105228
28	-112483	-104332
29	-111018	-102465
30	-108619	-99675
31	-105360	-96039
32	-101339	-91659
33	-96680	-86660
34	-91524	-81186

Amplitude	coeff	periods	offset	slope
32700	255	20	80000	-10

filtered_value = 0.996 × (last_filtered_value + sample - last_sample)
filtered_value = (coeff/256) × (last_filtered_value + sample - last_sample)

The high-pass filter output is represented by the dark blue line in *Figure 26*. After an adaptation period, the high-pass filter suppresses the DC and the low frequency content. The speed of adaptation depends on the "coeff" value (cut-off frequency of the filter).

4.4 Additional functions in DFSDM

4.4.1 Digital microphones (MEMS) support

Standardization in the domain of audio application has specified a PDM modulation (pulse density modulation) that is a common output format for digital microphones. A PDM signal is equivalent to a sigma-delta modulated signal and is therefore supported by the DFSDM.

Digital microphones are MEMS devices (micro electro mechanical systems) which are manufactured using semiconductor type of technology. The active actuator of such microphones consists in a membrane and a pair of micro electrodes. One of the electrode is fixed, the other one is incorporated within the membrane. As the air pressure (sound) is applied to the membrane, it moves the mobile electrode away from its default position and produces a change of capacity between both electrodes. The induced signal is processed by built-in electronic and output as a PDM modulated signal (pulse density modulation).

The digital microphones require an external clock signal (microphone CLK input signal) and data are sent over a DATA output line as a PDM modulated signal. The clock speed is usually in the range 1 to 3.2 MHz. The clock signal is provided by the DFSDM_CKOUT output signal and defines the microphone output data rate into the DFSDM. A schematic of a typical connection between a stereo digital microphone and the DFSDM is provided on *Figure 28*.

The DFSDM allows to connect 2 microphones in parallel through one single line (stereo configuration: left and right channel). The data and clock signals are common for both microphones. The clock signal is distributed from DFSDM_CKOUT pin to the left and right microphones. The output data signals from both microphones are multiplexed on the same wire: the left microphone provides data on the rising clock edge and the right microphone provides data on the falling clock edge (see *Figure 27*). The configuration of microphone for left or right channel is usually done configuring a pin on the microphone (L/R selection pin).

Figure 27. MEMS microphone outputs (L and R channel)

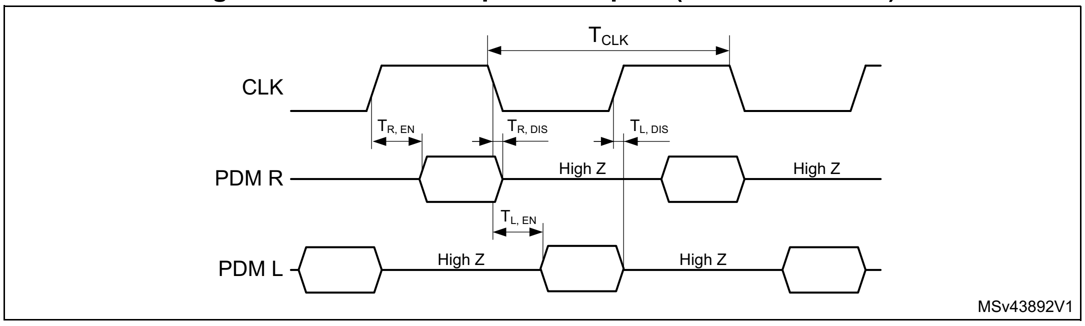
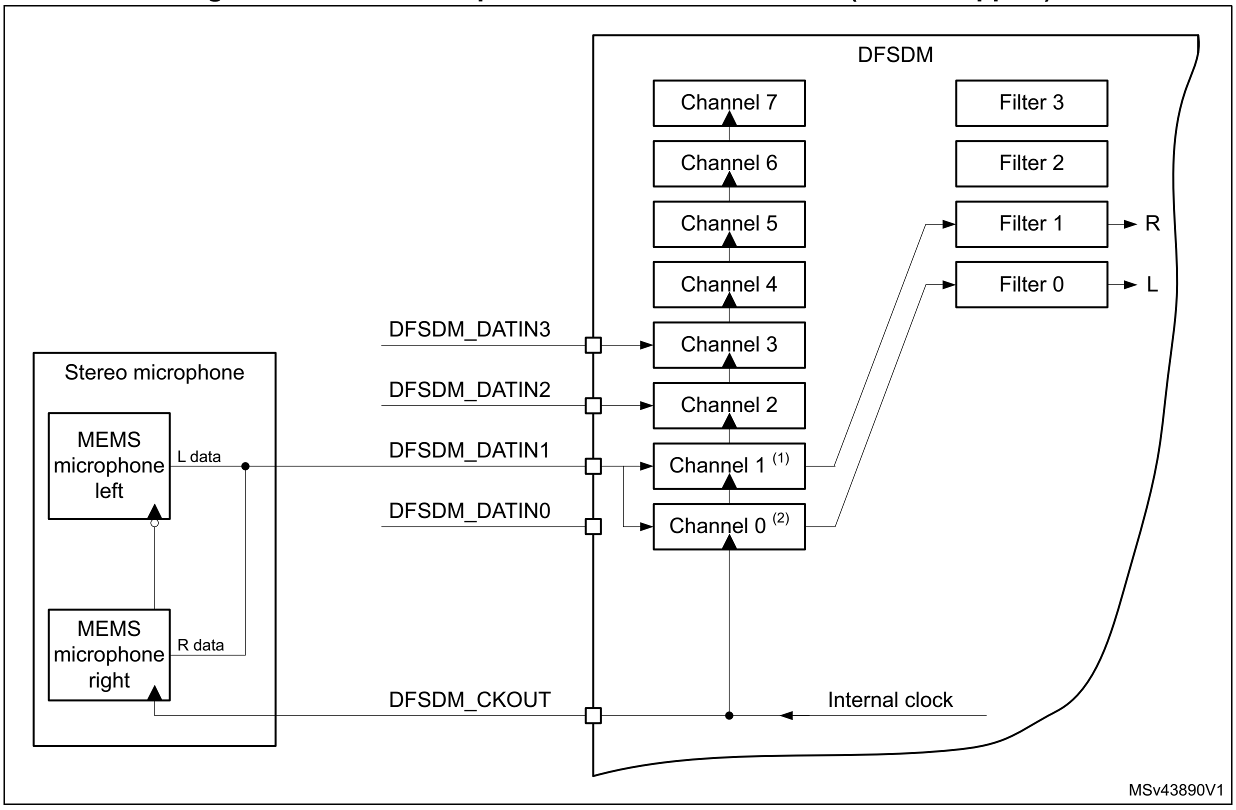


Figure 28. MEMS microphone connection to DFSDM (stereo support)



1. Direct input, falling edge sampling (R data).
2. Redirected from next channel, rising edge sampling (L data).

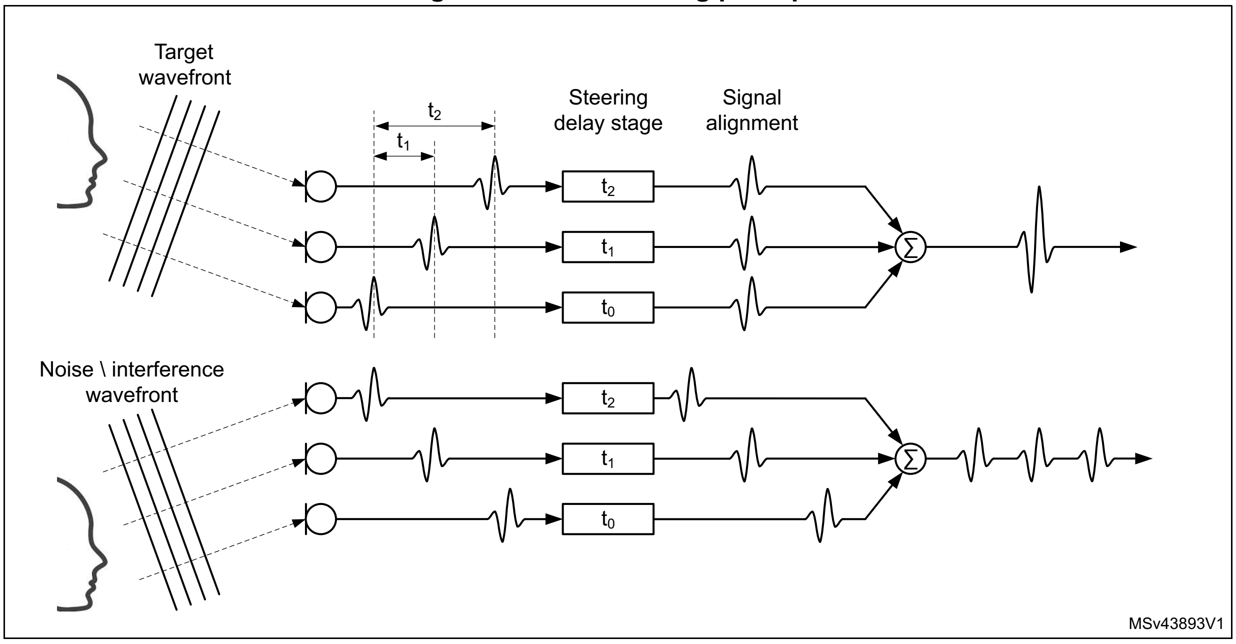
The separation of the two microphone signals on the DATA wire is performed by the DFSDM. The input to DFSDM channel x can be redirected in order to take the same input as channel (x+1). Then channel x is configured to sample data on rising edge and channel (x+1) is configured to sample data on falling edge. The clock signal for both channels is the same and internally connected to DFSDM_CKOUT signal. With such configuration, channel x is receiving data from left microphone and channel (x+1) is receiving data from right microphone. Both channels feed their own digital filters which eventually output two separate parallel data flows for left and right microphone channels. See *Figure 28* for a complete diagram of a stereo microphone application using the DFSDM.

4.4.2 Beamforming support

Beamforming is a technique giving to an array of fixed multi-directional sensors the ability to favor the signals coming from a particular direction (array of microphones in the example below). The preferred direction can be changed by firmware (without changing the position of the sensors).

In the example below, an array of microphones are positioned on a line with equidistant spacing (see *Figure 29*). An audio source is placed in front of the microphone array at an arbitrary angle, while at the same time unwanted noise and interferences are present all around. If signals from all microphones are simply summed together then only the sounds coming perpendicularly to the array are coherently summed (and therefore amplified). The sounds coming from non-perpendicular directions are not coherently summed.

Figure 29. Beamforming principle



In order to amplify the signals coming from another direction than perpendicular, one solution would be to mechanically rotate the microphone array, another solution consists in adding a specific delay to each microphone in the array, so that the various propagation delays of the useful signal arriving through each microphone are all compensated at the input of the adder. When the delay lines of all microphone are adjusted, only the sounds coming from the direction of the useful source are in phase and amplified (coherent combination), the sounds coming with a different angle are less amplified (incoherent combination) (see *Figure 29*).

The above description of beam forming algorithm is a introduction (simplified view). More elaborated algorithm can be implemented, providing even better performances.

Beamforming can be achieved with the DFSDM as follows. The DFSDM is able to sense signals coming from several microphones and process them separately to provide parallel outputs samples. The output samples from the different microphones are stored into separate data buffers in the microcontroller memory (for example 4 memory buffers from 4 microphones). The samples are then summed together by the microcontroller in order to provide the final output. The delay line is realized in 2 steps:

1. Coarse step: a DFSDM output buffer (for a given microphone) can be shifted with respect to another by a given number of samples (one sample corresponds to a delay of $1/f_{datarate}$, where $f_{datarate}$ is the output data rate (typical audio rates: 44.1kHz, 22.05kHz, 16kHz, …). A shift of 1 sample corresponds to a propagation distance of the sound in the air of $s = v / f_{datarate}$ (v is the sound speed in air ~343m/s).
   Example of coarse step:
   – fdatarate = 44.1 kHz: s = 343 / 44100 = ~7.8 mm
   – fdatarate = 16 kHz: s = 343 / 16000 = ~21.4 mm
2. Fine step: in order to support short distances between microphones in the array and to be able to fine tune the angle of the preferred sound source, a finer step is required. This is achieved by shifting samples on the 1-bit data stream between the microphone and the DFSDM rather than shifting DFSDM output samples in memory. The input 1-bit samples are sampled at PDM frequency. Here are examples of fine step distances:
   – f_{PDM} = 3 MHz: s = 343ms-1 / 3000000Hz = ~0.11mm
   – f_{PDM} = 200 kHz: s = 343ms-1 / 200000Hz = ~1.7mm

   This range of values is compatible with the fine tuning of the preferred angle for sound reception even for miniature microphone arrays.

The DFSDM fine step circuitry used to fine tune beamforming is not based on delay lines but consists in skipping some samples of the input 1-bit stream. Some of the 1-bit samples coming from the serial transceiver are masked (by clock signal gating) and are not sent to digital filter. The principle based on skipping clock pulses is shown on *Figure 30*.

Figure 30. Pulse skipping implementation for beamforming

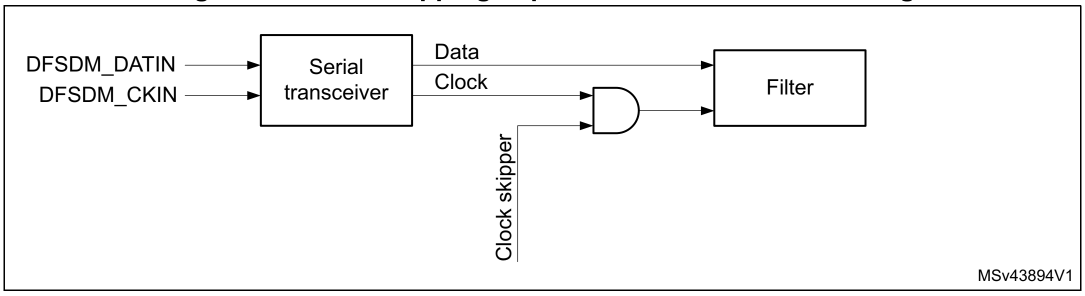
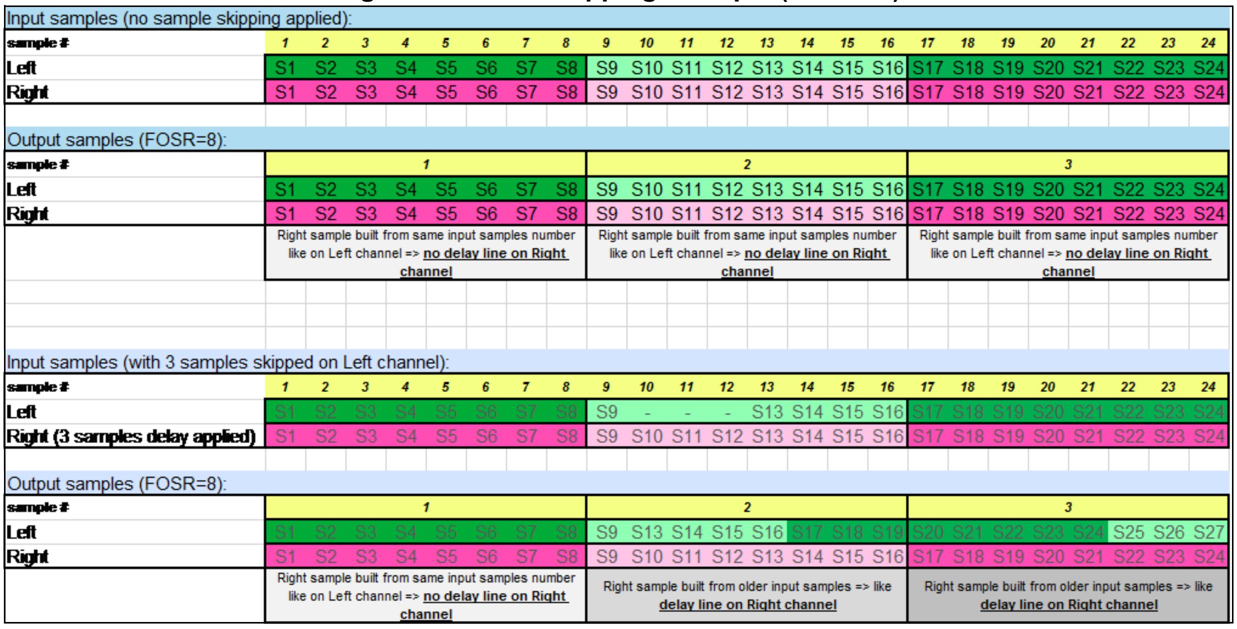


Figure 31 provides an example that illustrates the pulse skipping mechanism and its similarity with a delay line.

Figure 31. Pulse skipping example (FOSR=8)



Input samples (no sample skipping applied):

sample #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Left	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20	S21	S22	S23	S24
Right	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20	S21	S22	S23	S24

Output samples (FOSR=8):

sample #	1								2								3							
Left	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20	S21	S22	S23	S24
Right	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20	S21	S22	S23	S24
	Right sample built from same input samples number like on Left channel => no delay line on Right channel								Right sample built from same input samples number like on Left channel => no delay line on Right channel								Right sample built from same input samples number like on Left channel => no delay line on Right channel							

Input samples (with 3 samples skipped on Left channel):

sample #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Left	S1	S2	S3	S4	S5	S6	S7	S8	S9	-	-	-	S13	S14	S15	S16	S17	S18	S19	S20	S21	S22	S23	S24
Right (3 samples delay applied)	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20	S21	S22	S23	S24

Output samples (FOSR=8):

sample #	1								2								3							
Left	S1	S2	S3	S4	S5	S6	S7	S8	S9	S13	S14	S15	S16	S17	S18	S19	S20	S21	S22	S23	S24	S25	S26	S27
Right	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20	S21	S22	S23	S24
	Right sample built from same input samples number like on Left channel => no delay line on Right channel								Right sample built from older input samples => like delay line on Right channel								Right sample built from older input samples => like delay line on Right channel							

The top part of *Figure 31* represents the normal operation, without clock skipping applied. Each output sample is built from 8 input samples (FOSR=8). For beam forming several DFSDM filters need to be started simultaneously.

On the bottom part of *Figure 31* samples S10, S11, S12 have been skipped on the Left channel. The filter (with configuration: FOSR=8) waits for 3 more samples in order to get 8 complete 1-bit samples for each channel and be able to build one output sample. From the filter output prospective, it seems like the Right channel uses older 1-bit input samples than the Left channel for the same final output sample index. This is the same behavior as if the Right channel would have a 3-samples delay line (the delay line buffers holding the last 3 samples).

The above description of beam forming algorithm is a introduction (simplified view). More elaborated algorithm can be implemented, providing even better performances.

4.4.3 Audio clock support – independent clock operation

The DFSDM can clock an external sigma-delta modulator through a clock signal provided on the DFSDM_CKOUT pin. The clock frequency on this pin determines the input sampling frequency from which depends also the output data rate frequency. It is possible to select among the following sources to drive the DFSDM_CKOUT:

- DFSDM clock:
  - APB clock
  - System clock (independent from APB clock divider)
- PLL clock (audio PLL used for I2S)

The frequency from the selected clock source is divided by a factor in the range 2-256 (predivider ratio set according CKOUTDIV field in DFSDM_CHyCFGR1 register) in order to provide the DFSDM_CKOUT frequency.

In order to achieve a fine tuning of the DFSDM_CKOUT frequency, the PLL (providing both a multiplication and a division factor) must be selected as clock source. This is the appropriate selection for audio applications that require an accurate sampling frequency.

The DFSDM peripheral itself (transceivers, filters, additional functions) is using the APB clock or the system clock for digital processing. This “processing” frequency should be 4 times faster than the input sampling frequency (or 6 times if Manchester coding is used).

In order to save power consumption, both the sampling frequency and the processing frequency should be configured with the minimum values required by the application.

4.5 DFSDM power consumption optimization

The DFSDM consumption depends on:

- Enabled blocks:
  - Number of enabled channels
  - Number of enabled filters
  - Number of enabled features (analog watchdog, short circuit detector)
- Clock speed:
  - Transceivers
  - Digital filters (+ additional functions)

In order to optimize the power consumption, only the required blocks should be enabled when needed (transceivers, filters, additional functions) and the minimum clock speed to achieve the required input and output data rate should be used.

4.5.1 Power optimization in Sleep mode

The DFSDM may be used in Sleep mode. In this case, the DFSDM is typically used to monitor analog signals (analog watchdog enabled) and to wake up the CPU if an analog threshold is reached.

Examples of such applications are baby monitoring or broken glass detection where the CPU is almost always in sleep mode and the DFSDM is monitoring the sound level through the analog watchdog. If the sound threshold is reached, the CPU is woken up by the DFSDM analog watchdog interrupt, the DFSDM collects the captured data and the CPU performs analysis on the data collected by the DFSDM (to assess if it is really a baby sound or a broken glass sound). During the sleep mode it is not required that the DFSDM collects the data, only the analog watchdog functionality should be turned ON in order to optimize power consumption.

Here is a example of DFSDM and MCU configuration for low power operation (sound detection):

- Clocks:
  - Disable all peripherals except DFSDM
  - Use system clock at minimum frequency for DFSDM operation. (example: 4 MHz, use internal MSI oscillator to reduce even more consumption)
  - Reduce AHB and APB clocks to minimum because there is no need to access the peripherals during Sleep mode. Use high AHB and APB predividers (example: AHB predivider = 512, AHB clock = 4 MHz/512 = 7.8 kHz).
  - Use system clock (example: 4 MHz) for the DFSDM because the APB clock is too slow for digital filter processing.
  - Sampling clock for digital MEMS microphone should be less than DFSDM clock / 4 (example: 1 MHz is appropriate for microphone operation).
- DFSDM:
  - Configure the filter for appropriate output data rate (example: 16 kHz).
  - Set the analog watchdog for monitoring output data and set appropriate thresholds for sound detection.
  - Disable overrun errors of final data because data are not collected from DFSDM (only analog watchdog is working).
- After wakeup (interrupt from analog watchdog):
  - Set back AHB, APB clocks (predividers) for high speed operation to speed up DFSDM communication and data transfer (example: AHB predivider = 1, AHB clock = 4 MHz).
  - Optionally reconfigure DFSDM for higher output data rate for better sound recognition analysis (example: 44.1 kHz).
  - Start data collection from DFSDM into memory buffers (example: DMA data transfer).
  - Perform sound recognition over each data buffer (example: FFT analysis).
  - If a critical sound is detected, perform the necessary action (invoke alarm). If the sound was not critical (or no more sound is detected after a while) then go to the Sleep operation again and wait for next wakeup from analog watchdog.

With the above DFSDM scenario ongoing, the increase of the microcontroller power consumption (due to DFSDM activity) is limited to roughly a factor 2 (with respect to the consumption in a strict sleep mode without any DFSDM activity and same system clock frequency).

5 DFSDM peripheral configuration tutorial

The DFSDM functionality is similar to an A/D converter that would have an external analog front-end part. The DFSDM offers a lot of flexibility but requires a bit of methodology to navigate through all the settings. The following tutorial helps the user to configure the DFSDM according the application requirements for A/D conversion.

5.1 Configuration introduction

The basic application parameters for A/D conversion are:

- Input analog range (sigma-delta modulator property)
- Minimum output data rate
- Data rate precision setting
- Minimum output data resolution
- Number of collected channels (per ADC)
- Additional functionality:
  - Analog watchdog
  - Short circuit detector
- Type of data transfer: DMA, interrupt, polling.

Application related properties impacting the DFSDM configuration:

- System clock range
- Supply voltage
- Consumption limitation
- Used sigma-delta converter (or digital microphone) properties: clock speed range, input analog range, order of sigma-delta modulator.

In the next sections, the tutorial goes through the configuration of the various DFSDM parts in order to achieve the application requirements.

5.2 Clocks configuration

The DFSDM clock selection is guided by the data requirements:

- Minimum output data rate
- Data rate precision
- Minimum output data resolution

The goal is to set the filter parameters and the input sampling clock rate in order to reach the output data rate.

The input sampling clock selection is driven by the sigma-delta modulator (or digital microphone) characteristics:

Please refer to *Table 2* below providing examples of clocks configuration for two different applications.

5.3 Transceivers

5.3.1 Serial transceivers configuration

The serial transceivers convert the data present on the input pins (DFSDM_DATIN and DFSDM_CKIN) and provide them to the filters. The configuration of the transceivers depends on the characteristics of the external device(s) connected to the DFSDM (sigma-delta modulator or digital microphone).
The *Table 2* below provides examples of DFSDM configurations for two different applica-tions:

Application 1:

- Application: temperature measurement consisting in measuring voltage and current over a Pt100 sensor and determining the Pt100 sensor resistance/temperature. The temperature measurement is triggered by a timer every 1 second.
- External device: dual external sigma delta modulator STPMS2 for voltage and current channel measurement.
- Physical connection:
  - The STPMS2 uses only one SPI line (pins: CLK, DAT).
  - The STPMS2 CLK pin is driven by DFSDM_CKOUT pin
  - The STPMS2 DAT pin is connected to DFSDM_DATIN1 (serial data input on channel 1 pin, the channel 0 input is redirected to channel 1 pin, then signal on DATIN1 is connected to both channel 0 and channel 1).
  - DFSDM_CKIN1 pin (serial clock input on channel 1) is not used. Data are sampled using internally redirected clock from CKOUT pin (SPICKSEL [1:0] = 1).
- Post processing:
  - Data from channel 0 (U, voltage) are processed by filter 0.
  - Data from channel 1 (I, current) are processed by filter 1.
  - The resistance of Pt100 sensor is determined in firmware as R = U/I and is used to extrapolate the Pt100 sensor temperature.

Application 2:

- Application: audio recording at 48 kHz data rate.
- External device: MEMS digital microphone MP34DT01-M.
- Physical connection:
  - MP34DT01-M has one data line (pins: CLK, DOUT).
  - The MP34DT01-M CLK pin is driven by DFSDM_CKOUT.
  - The MP34DT01-M DOUT pin is connected to DFSDM_DATIN0 (serial data input on channel 0).
- Post processing:
  - Data from channel 0 (microphone) are processed by filter 0.
  - The output stream at a datarate of 48 kHz is stored in a RAM buffer by the DMA, in continuous mode.

Table 2. DFSDM application examples

Parameters		Application 1: temperature measurement		Application 2: audio sampling	
Description	Options	Setting[1]	Notes	Setting[1]	Notes
Selection of the serial protocol (SITP[1:0])	SPI type	x	STPMS2 uses SPI communication	x	MP34DT01-M uses SPI type protocol.
	Manchester type	-	-	-	-
Clock master	DFSDM[2]	x	STPMS2 has clock input pin	x	MP34DT01-M needs clock input.
	External device[3]	-	-	-	-
Active sampling edge	Rising edge	-	current stream active on rising edge	x	MP34DT01-M has pin “L/R” at GND level – rising edge is active.
	Falling edge	-	voltage stream active on falling edge	-	-
Clock source	APB clock System clock PLL clock	System clock (f_{HSI16})	f_{CKIN} requirement 2 MHz[4] f_{CKIN} configuration: $f_{CKIN} = f_{HSI16}$ / (CKOUTDIV + 1) with, f_{HSI16} = 16 MHz CKOUTDIV = 7	PLL clock	f_{CKIN} requirement 3.072 MHz[5] f_{CKIN} configuration: $f_{CKIN} = f_{HSE}$ * N/(Q*(CKOUTDIV+1)) with f_{HSE} = 16 MHz (crystal - precise clock) N =48 Q = 2 CKOUTDIV = 124
Input clock frequency	f_{CKIN}/f_{CKOUT}	2 MHz		3.072 MHz	
Number of data streams sent over one serial line	One stream[6]	-	Also supported by STPMS2 but requires one more wire/pin	x	Mono microphone is used.
	Two streams[7]	x	A common wire for current and voltage signal is used in this example. Only one pin/wire is necessary.	-	-
Clock accuracy	Low accuracy	x	No need of precise clock – temperature measurement is triggered each second by timer. Clock signal based on DFSDM clock (16MHz): CKOUTSRC = 0	-	-
	High accuracy	-	-	x	Required precise 48 kHz data rate (audio standard).

1. ‘x’ = valid setting, ‘-’ = invalid setting.

2. Device requires external clock (clock input into device).
3. Device uses internal clock (clock output by the device).
4. STPMS2 clock input range: 1 to 4.915 MHz.
5. f_{CKIN} = datarate * FOSR * IOSR with datarate = 48 kHz, FOSR = 64, continuous mode.
6. One stream sent (only one sampling edge active).
7. Two streams sent (one stream data active on rising clock edge, second stream data active on falling clock edge).

5.3.2 Parallel transceivers

The parallel transceivers offer an alternative consisting in providing 16-bit wide data into the digital filter directly from internal microcontroller data sources.

The parallel transceivers consist in a set of 32-bit (2x16-bit) parallel input registers feeding the digital filter. There is one 32-bit parallel input register (divided into 2x16-bit) for each serial input channel. The DATMPX[1:0] bits in DFSDM_CHyCFGR1 register allow to select the filter input from either the serial input or the 16-bit parallel input. If DATMPX[1:0] = 0 then data are taken from serial transceivers (as 1-bit serial stream) otherwise the data are taken from parallel transceivers (as 16-bit signed data).

Data can be written into parallel registers by the CPU, by the DMA (DATMPX[1:0]=2) or by the ADC (DATMPX[1:0]=1). Depending on the embedded ADC capability, some microcontrollers feature a dedicated fast internal 16-bit bus between ADC and parallel input registers.

Each channel has its own 32-bit parallel register which is divided into two 16-bit registers. It allows to use 32-bit accesses to those registers and write 2 input samples at each write access. The usage of upper and lower 16-bit samples in 32-bit registers depends on the selected "data packing operation mode" defined in field DATPACK[1:0]:

- Standard mode: only lower 16-bit sample is used
- Interleaved mode: both samples are used for the same channel (like a 2 samples FIFO buffer)
- Dual mode: the lower 16-bit sample is used as input for a given channel while the upper 16-bit sample is used as input for the next channel.

Writing one 16-bit sample into the parallel input register automatically generates one sampling clock signal for the digital filter which in turn automatically samples the parallel input register as next input to be processed. Therefore the final output data rate depends on the input data rate (CPU or DMA or ADC data transfer speed).

The parallel transceivers allow to perform fast low-pass filtering without CPU intervention. Here are 3 practical examples:

- Post-processing of ADC samples

  The ADC is used as data source.

  The ADC is configured to be used with DFSDM.

  The DFSDM configuration is as follows:

  – Parallel inputs selected from ADC: DATMPX[1:0]=1.
  – The ADC is sending its outputs data over 16-bit internal bus directly into parallel registers (only lower 16-bit part is used). The data packing has no meaning in case of ADC data inputs and it is recommended to set DATPACK[1:0]=0.
  – The DFSDM processes data according the setting of filter parameters (FOSR, FORD, IOSR) and then produces the final output data samples at a lower data rate. Example: averaging of data from ADC.
- Post-processing of data stored in RAM memory buffer (for instance low-pass filtering of data from ADC after software high-pass filtering)

  The parallel inputs are selected from the parallel input registers: DATMPX[1:0]=2.

  The CPU can write data directly from a buffer to the parallel registers. Alternatively the DMA can be used to load data into the parallel input registers.

  The data packing should be optimized according the application:

  – If data are stored in a buffer in 16-bit format, then DATPACK[1:0]=2 setting can be used to improve performance. In this case 2 consecutive samples can be written in one 32-bit transfer (FIFO buffer).
  – If data are stored in a buffer in 16-bit format and each second sample is from another data source (for example odd samples corresponding to audio L channel and even samples corresponding to audio R channel) then DATPACK[1:0]=3 setting can be used to improve performance. In this case one sample pair can be written in one 32-bit transfer (write 2 samples into 2 different channels parallel input registers).
- Post-processing of data from another communication peripheral: the analog data are transferred from the peripheral directly into the given parallel input register (DMA) and processed by the DFSDM to produce the final filtered samples. For instance an external ADC may be connected to the microcontroller SPI interface and the DFSDM used for low-pass filtering. In this case the setting of DFSDM is similar to case 2.

5.4 Filter

5.4.1 Sinc filter

The digital filter performs digital signal processing on data received by serial or parallel transceivers. Data can be in 1-bit format (serial transceivers) or in 16-bit signed integer format (parallel transceivers). Each sample received by the transceiver is automatically sampled by the digital filter for processing. According the digital filter configuration parameters, several input samples may be required until the first output data is available from the filter.

The filter implemented in the DFSDM is a low pass filter of the Sinc type (also known also as comb filter). The basic parameters for this filter are as follows:

- FOSR: filter oversampling ratio - defines how many samples are processed together in the moving average. FOSR range is 1..1024.
- FORD: filter order - defines the number of iterations of the moving average. FORD range is 1..5.

The configuration of the filter depends on the application requirements and are discussed in the next 4 sub-sections:

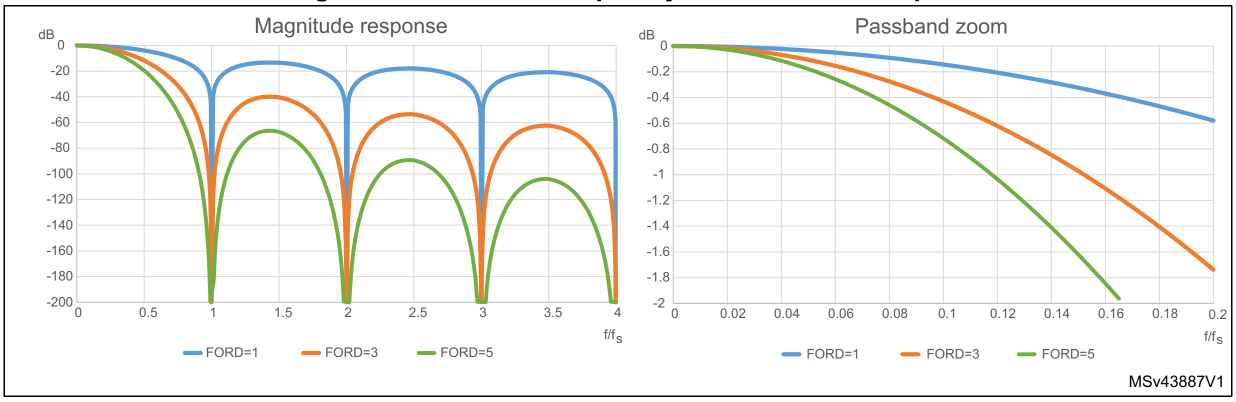
Low pass filtering characteristics

The filter transfer frequency characteristic is important for the application. The Sinc filter frequency characteristic for FORD = 1, 3 and 5 and for fixed FOSR = 8 is provided in *Figure 32: Sinc filter frequency characteristic shape* (a more general figure was introduced in *Figure 19*). The magnitude response curves in *Figure 32* indicate the amount of rejection applied to the high frequency components of the signal, outside of the filter passband (from around -20dB rejection for FORD = 1 down to around -100 dB for FORD = 5). The higher the filter order, the better the higher frequencies suppression (low pass filter). There is a side effect of using high filter order that may affect the signal within the useful filter passband. This is illustrated by the “passband zoom” curves of *Figure 32*. The filter is not flat within the passband but applies a slight attenuation that increases with the frequency and the filter order. Depending on the useful frequency bandwidth of the analog input signal, this may affect the quality of the filtered signal. If necessary, this attenuation can be compensated by application of a software compensation filter on the final output data.

The notch frequencies are those discrete frequencies where the attenuation of the signal is maximum (see *Figure 32: Sinc filter frequency characteristic shape*). The first notch frequency (and all its harmonics) of the Sinc filter is independent of FORD and depends only on the sampling clock and the selected FOSR: $f_{notch} = f_{sampling}$ / FOSR. Decreasing FOSR results in larger steps between notch frequencies.

In some applications, one can take advantage of the notch frequencies to better reject sources of noise with energy concentrated at a particular frequency. For instance if an application is sensitive to the 50 Hz mains frequency and its harmonics, it is recommended to tune the filters so that the first notch frequencies equals 50 Hz. With such configuration and filter tuning, it is possible to exploit very low signals in relatively high noise environment.

Figure 32. Sinc filter frequency characteristic shape



Output data resolution

A consequence of the Sinc filter operation (moving average) is to increase the resolution of the sampled signal (by a factor FOSR). Multiple averaging increases even more the resolution. The total resolution (in LSBs) of the output signal is then:

$\text{Resolution}_{out} = \text{Resolution}_{in} * \text{FOSR}^{\text{FORD}}$.

- Resolution_{in} correspond to the input data resolution (2 in case of serial data input or wider in case of parallel data input, for example 4096 for 12-bit parallel input).
- Caution must be taken to not increase Resolution_{out} over the 32-bit range because the filter is performing all internal operations in 32-bit resolution.

Output data rate

A consequence of the Sinc filter operation (moving average) is to reduce the output data rate with respect to the input sampling rate. The output data rate for a continuous signal conversion depends only on FOSR:

$\text{Datarate}_{out} = f_{sampling} / \text{FOSR}$ (equal to first f_{notch} frequency).

Since Datarate_{out} depends only on FOSR while Resolution_{out} depends on both FOSR and FORD, it is recommended to first adjust the FOSR to achieve the required output data rate, and then adjust the FORD to achieve the required output resolution.

Latency time

The latency represents the time that elapses between starting the filter (with valid input samples) and the first valid output sample. The latency has to be considered after starting the filter (first sample) and also after resuming the filter processing (in case input samples are not continuous and a trigger is used to restart conversion). The filter consists in a chain of clocked logic (adders and registers, principle of moving average) that needs to be initialized with valid data from previous input samples. If the input sample flow was stopped and therefore the history of input data was lost, it is necessary to wait that the filter logic is refilled with valid data before the output data becomes meaningful again. The filter latency is given by:

$T_{latency} = T_{sampling} * [(\text{FOSR}*\text{FORD}) + (\text{FORD}+1)]$

Note: *The theoretically minimum latency is $T_{latency} = T_{sampling} * (\text{FOSR}*\text{FORD})$ but due to filter implementation and optimization, a few more cycles are required represented by the term (FORD+1) in the formula.*

Simulation

The digital filter simulation available in *[TUTORIAL]* allows to:

- Define an input signal: either a signed 1-bit data stream or a signed 16-bit data stream
- Configure the filter parameters (FOSR, FORD and IOSR)
- Compute the final output signal and verify the influence of the filter parameters. An example of simulation result is provided in *Figure 18: Filtering simulation*.

5.4.2 Integrator

The integrator has one configuration parameter, the IOSR (integrator oversampling ratio). The integrator processing consists in accumulating (sum) IOSR consecutive samples from the filter output. This operation additionally decreases the output data rate (by a factor IOSR).

The requirements to consider in order to set the IOSR are:

- Required output data rate
- Required number of samples from the filter to produce one integrator output sample.

An example of simulation result extracted form *[TUTORIAL]* is provided in *Figure 18: Filtering simulation*.

5.5 Analog watchdog

The configuration parameters of the analog watchdog embedded inside the DFSDM peripheral are provided in *Table 3*.

Table 3. DFSDM analog watchdog parameters

Parameter	Description
AWLT	Analog watchdog low threshold. Signed 24-bit format.
AWHT	Analog watchdog high threshold. Signed 24-bit format.
AWFSEL	Analog watchdog fast mode select. 0: analog watchdog watches output final 24-bit data (after the digital filter) 1: analog watchdog watches data based on input 1-bit serial data samples(1)

1. In case AWFSEL=1 the serial input data are filtered by a dedicated analog watchdog filter that should be configured in the same way as the main digital filter:
- AWFORD: analog watchdog filter order
- AWFOSR = analog watchdog oversampling ratio
Due to reduced analog watchdog filter options (AWFOSR = 1…32, AWFORD = 1…3) the signal from this filter as a maximum resolution of 16 bits. In this case the resolution of the high (AWHT[23:0]) and low (AWLT[23:0]) threshold levels is limited to the higher 16 bits (AWHT[23:8], AWLT[23:8]).

Each input channel has its own comparator which compares the analog watchdog data (from analog watchdog filter) with analog watchdog threshold values (AWHT/AWLT). When several channels are selected (AWDCH[] in DFSDM_FLTxCR2 register), several comparison requests may be received simultaneously. In this case, the channel request with the lowest number is managed first and then continuing to higher selected channels. For each channel, the result can be recorded in a separate flag (AWHTF[], AWLTF[] in DFSDM_FLTxAWSR register). One comparison takes 1 DFSDM clock cycle. To be able to perform comparisons from all selected channels the user must properly configure the number of watched channels and analog watchdog filter parameters with respect to the input sampling clock speed and DFSDM frequency.

The DFSDM analog watchdog can generate interrupts and break signals. The interrupts are used to trigger the CPU and require a software intervention. The break signals (BKAWH[3:0]) are used for fast hardware interaction with other peripherals (for example stopping the timers to generate PWM in motor applications).

5.6 Short circuit detector

A short circuit detector is available on each channel and can be enabled with the SCDEN bit. The short circuit detector threshold is configured for each channel with SCDT[7:0] (range 0-255).

The short circuit detector features an upcounting counter that counts consecutive 0's or 1's on serial data receiver outputs. The counter is restarted each time a logical transition is detected in the data stream (from 1 to 0 or from 0 to 1). If the counter reaches the value stored in SCDT[7:0], then a short-circuit event is invoked.

There are to type of usage for the short circuit detector feature that are presented in the following 2 subsections.

True short circuit detection

In this case the goal is to detect situations where the DFSDM input signal is in the overflow state (continuously exceeding the external sigma-delta modulator full scale range specification) during a period of time exceeding the allowed duration of overflow state.

According application requirements, the maximum duration of the short circuit state is defined as $t_{shortmax}$. This is converted into a maximum number of consecutive 0's or 1's that is used to configure SCDT[7:0]:

$SCDT[7:0] = t_{shortmax} \times f_{sampling}$

Each time the short circuit detector detects that input signal is stable during SCDT[7:0] bits, an error is reported (the input signal is in saturation over the maximum allowed time).

Short circuit detection handling

There are 2 ways of handling the short circuit detection events:

- Software interrupt generation. The microcontroller application software (interrupt routine) should perform the required actions (feedback) to a given short circuit event (for example stop the PWM generation into motor control applications).
- Break signal. A break signal, configured through the BKSCD[3:0] field, can be transmitted to other peripherals without any software intervention (example the break signal can be used to control a timer peripheral and stop the PWM generation into motor control applications).

5.7 Pulse skipper

The DFSDM can be used in beamforming applications where the location of an input signal source can be selected according its angle of incidence versus an array of sensors. The principle is described in *Section 4.4.2: Beamforming support*. Beamforming is usually used in conjunction with an array of digital microphones within audio applications.

The configuration of the DFSDM for beamforming requires to calculate the respective delay to be applied to each microphone in the array (see *Figure 29: Beamforming principle*). The delay of each microphone is calculated according the preferred receiving angle and the input sampling frequency and converted into a number of clock pulses of the serial input signal sampling frequency. Before configuring the delays, the reception of each microphone input bitstream by the respective DFSDM channel must be started synchronously, this is handled by the DFSDM global enable signal (DFSDMEN=1). After the DFSDM conversions are started, the calculated delay for each microphone channel must then be programmed once in each respective PLSSKP[5:0] field for each channel. After configuring the PLSSKP[5:0] fields, the software should read back the PLSSKP[5:0] fields and check that the values have been reset to 0 before asserting that the DFSDM final output data have been processed with the required delays.

After being initially configured, the delays can be updated again without stopping the audio data reception, but only by adding-up to the already programmed delays. The software must remember the successive writes into each PLSSKP[5:0] field for each microphone channel and consider that the effectively applied delay is the cumulated value of all successive writes.

Example:

Let's assume 3 microphone channels have been synchronously started and their PLSSKP[5:0] fields programmed with respective delay values of 0, 5 and 10 corresponding to a preferred angle φ. If the preferred angle should be changed to -φ, a second write should be performed in PLSSKP[5:0] fields with the respective values 20, 10, 0, resulting in a cumulated total delay for the respective channels of 20, 15, 10 (this is the difference between the delays that determine the preferred angle and not the delay absolute values).

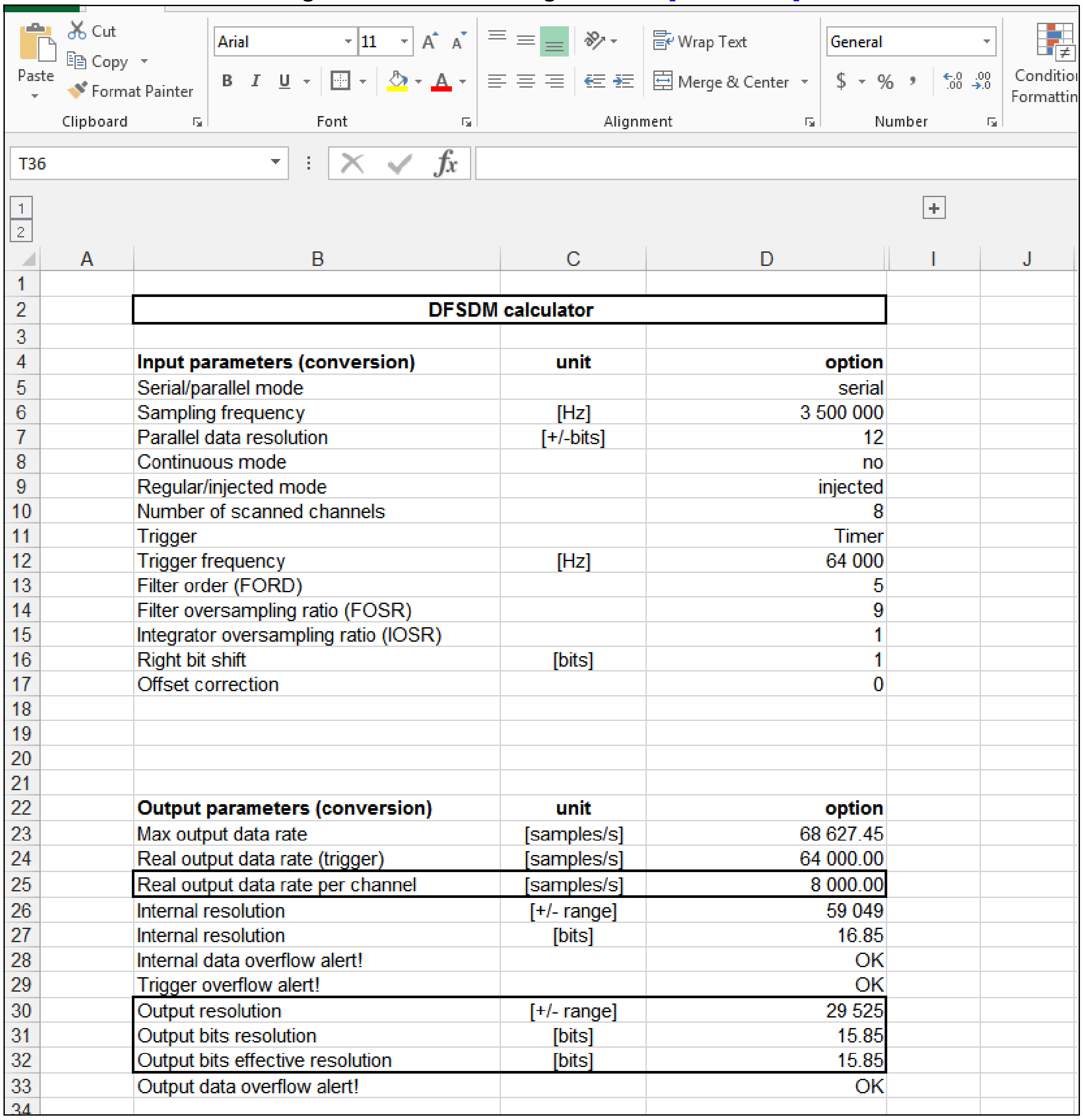
If the DFSDM is stopped (all continuous streams from microphones are stopped) the cumulated delays for all microphones are reset to zero at new DFSDM synchronous start (DFSDMEN=1).

5.8 Configuration using *[TUTORIAL]*

The *[TUTORIAL]* provides a configurable simulator based on the principles and formulas presented in this application note as well as a model of the DFSDM peripheral.

The interface of the DFSDM simulator is presented in a simple way, like if it would be a standard ADC. The user can configure the characteristics of the input signal and the DFSDM parameters according a specific analog-to-digital conversion application (sampling frequency, mode selection, filter parameters …). The user can play with the input parameters and converge to the expected final conversion properties (output data rate and resolution) or use the simulator as a verification mean of a predefined configuration. The second approach (more deterministic) involves a preliminary analysis consisting in translating the application requirements into a digital signal processing scenario ported onto the DFSDM architecture, with the help of this application note. The analysis is not always straightforward and the user may sometime consider some trade-offs (like resolution versus data rate).

Figure 33. DFSDM configuration in *[TUTORIAL]*



DFSDM calculator	

Input parameters (conversion)	**unit**	**option**
Serial/parallel mode		serial
Sampling frequency	[Hz]	3 500 000
Parallel data resolution	[+/-bits]	12
Continuous mode		no
Regular/injected mode		injected
Number of scanned channels		8
Trigger		Timer
Trigger frequency	[Hz]	64 000
Filter order (FORD)		5
Filter oversampling ratio (FOSR)		9
Integrator oversampling ratio (IOSR)		1
Right bit shift	[bits]	1
Offset correction		0

Output parameters (conversion)	**unit**	**option**
Max output data rate	[samples/s]	68 627.45
Real output data rate (trigger)	[samples/s]	64 000.00
Real output data rate per channel	[samples/s]	8 000.00
Internal resolution	[+/- range]	59 049
Internal resolution	[bits]	16.85
Internal data overflow alert!		OK
Trigger overflow alert!		OK
Output resolution	[+/- range]	29 525
Output bits resolution	[bits]	15.85
Output bits effective resolution	[bits]	15.85
Output data overflow alert!		OK

6 Conclusion

The DFSDM peripheral embeds advanced, and sometime complex, features that are useful and beneficial in many applications involving analog-to-digital conversion. This document introduces many different fields of application for the DFSDM, the advantages of using this peripheral and configurations examples as well as guidelines. The *[TUTORIAL]* allows to quickly experiment the influence of the various DFSDM parameters and validate that the software architecture choices and associated configurations allow to meet application requirements. The understanding of the principles underlying the DFSDM processing and how they can be implemented with appropriate tools and methods (the aim of this document) is a key factor for successful DFSDM-based applications.

7 Revision history

Table 4. Document revision history

Date	Revision	Changes
30-Mar-2018	1	Initial release.