life.augmented

AN4629 Application note

ADC hardware oversampling for microcontrollers of the STM32 L0 and L4 series

Introduction

This application note provides an overview of the on-chip hardware Analog-to-Digital Converter (ADC) oversampling engine integrated in microcontrollers belonging to the STM32 L0 and L4 series.

The main benefit the user can get from the hardware oversampling is increased SNR (signal-to-noise ratio) with less CPU interaction, resulting in overall lower power consumption compared with the software-based implementation.

Table 1. Applicable products

Type	Product series
Microcontrollers	STM32L0
	STM32L4

Contents

1 Oversampling as a way to improve the quality of signal acquisition . 5

1.1 Quantization noise 5

1.1.1 SNR evaluation 6

1.2 Processing gain achievable with oversampling 6

1.3 Dithering 7

1.4 Overview of hardware oversampler in STM32 L0 and L4 microcontrollers 8

2 Hardware oversampling to reduce power consumption 10

2.1 Software implementation 10

2.2 Hardware implementation 10

2.3 Results 10

3 Conclusion 12

4 Revision history 13

List of tables

Table 1. Applicable products 1
Table 2. Comparison of SW and HW implementation of ADC oversampling technique. 10
Table 3. Document revision history 13

List of figures

Figure 1. Ideal N-bit ADC quantization 5
Figure 2. Quantization noise spectrum 6
Figure 3. Quantization noise gain 7
Figure 4. Dithering technique 7
Figure 5. Frequency response of Accumulate-and-Dump filter 8

1 Oversampling as a way to improve the quality of signal acquisition

1.1 Quantization noise

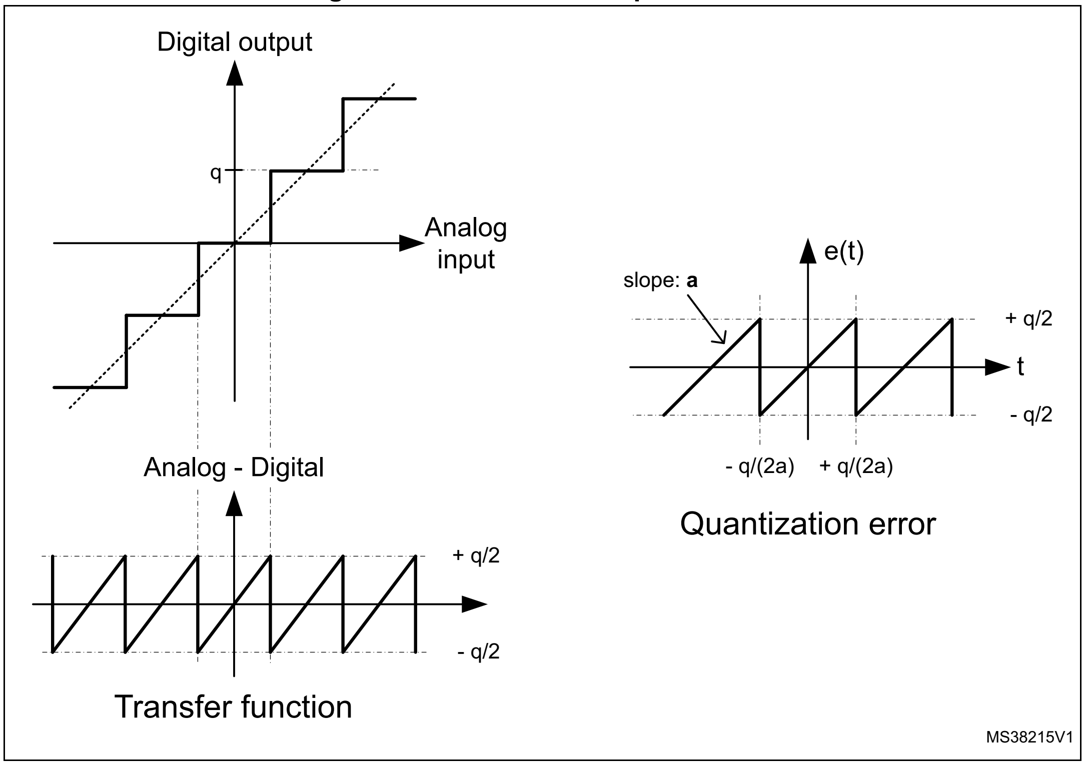
Analog-to-digital converters (ADCs) provide the transformation of analog signals into an array of digital codes. This operation performs an amplitude quantization of the analog input signal. This quantization is due to the fact that the binary output words have a finite length, normally in the range of 6 to 18 bits. The error between the input signal and the quantized signal is called the quantization error.

The maximum error an ideal converter makes when digitizing a signal, is ±½ LSB (Least Significant Bit) as shown in the transfer function (left side of *Figure 1*). The LSB is also often called a quantum (q).

The quantization error e(t) as a function of time is shown in the right side of *Figure 1*.

It is often approximated by a saw tooth signal, and we generally admit that an uncorrelated sawtooth waveform is a good representation of the quantization error for any AC signal, and that it behaves like a wideband noise.

Figure 1. Ideal N-bit ADC quantization



1.1.1 SNR evaluation

The quantization error e(t) is defined as

$$e(t) = a \times t$$

with $-q / 2a < t < +q / 2a$.

So the RMS value of e(t) is:

$$\sqrt{\overline{e(t)}^2} = \sqrt{\frac{a}{q} \cdot \int_{q/(2a)}^{(-q)/(2a)} (e(t)^2)dt} = \frac{q}{\sqrt{12}}$$

The theoretical signal-to-noise (SNR) ratio can now be calculated assuming a full-scale input sine wave is expressed as follows:

$$s(t) = q \times 2^{(N-1)} \times \sin(2 \pi f t)$$

Using the 2 previous equations, we can compute the SNR of an ideal N-bit converter:

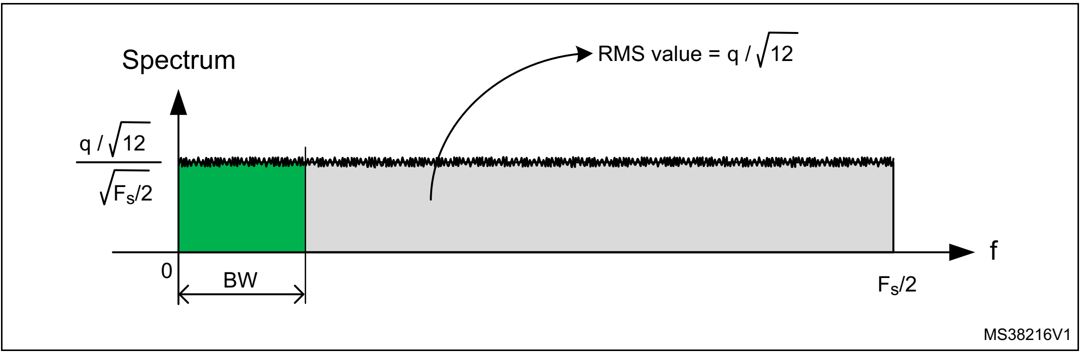
$$SNR = 6.02 \times N + 1.76 \text{ dB.}$$

It is important to notice that the RMS quantization noise is measured over the full Nyquist bandwidth (from DC up to $F_s / 2$).

1.2 Processing gain achievable with oversampling

In most of the cases, we can consider that the quantization noise is uncorrelated with respect to the input signal. In this condition, the quantization noise is approximately Gaussian and spread more or less uniformly over the Nyquist bandwidth (see *Figure 2*).

Figure 2. Quantization noise spectrum

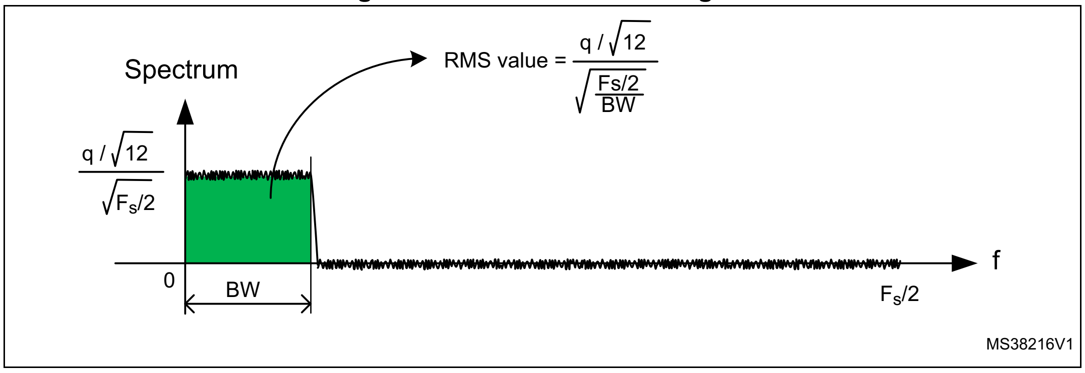

Spectrum
RMS value = $q / \sqrt{12}$
$\frac{q / \sqrt{12}}{\sqrt{F_s/2}}$
0
BW
$F_s/2$
f
MS38216V1

However, under certain conditions where the sampling clock and the signal are harmonically correlated, the quantization noise becomes correlated, and its energy is concentrated in the harmonics of the signal. In conditions where the quantization noise does not appear as random noise, dithering must be applied (see *Section 1.3*).

In several applications, the useful signal occupies a bandwidth (BW) smaller than $F_s / 2$.

If digital filters are used to remove the noise outside BW, the total RMS noise will be reduced (see *Figure 3*): the RMS value of the quantization noise will be divided by a ratio which depends on the useful bandwidth (BW) with respect to the sampling rate (F_s).

Figure 3. Quantization noise gain


Spectrum
RMS value = $\frac{q/\sqrt{12}}{\sqrt{\frac{Fs/2}{BW}}}$
$\frac{q/\sqrt{12}}{\sqrt{F_s/2}}$
0
BW
$F_s/2$
f
MS38216V1

Then we can reformulate the previous SNR expression by taking into account this processing gain, by filtering the out-off band noise:

$$SNR = 6.02 \times N + 1.76\ dB + 10 \times Log_{10}\ OSR$$

This expression is valid over a bandwidth of BW, with OSR = F_s / (2 × BW), where OSR is called the Over Sampling Ratio.

1.3 Dithering

The technique presented above works nicely if the quantization noise is a white one.

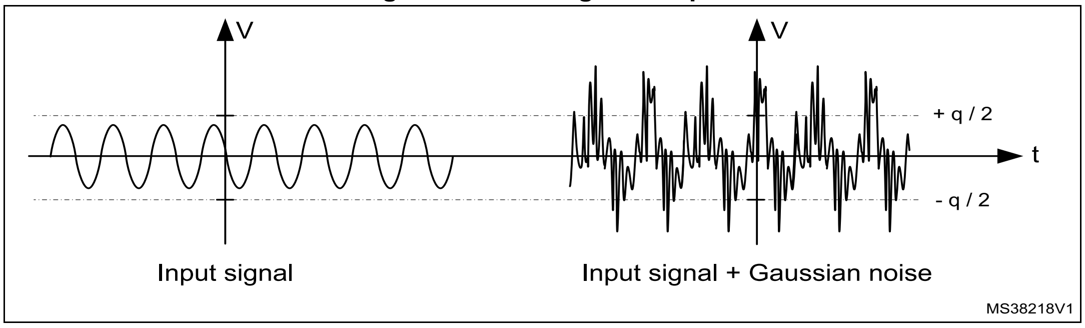
However, if the sampling clock and the signal are harmonically correlated (in this case the quantization noise becomes correlated as well), or when the input signal amplitude is smaller than q / 2, the processing gain will not work properly.

This is due to the fact that the quantization noise is no longer random for the first case, and because in theory there will be no code transitions when the signal is smaller than the quantization step for the second case.

A way to solve these issues is to use the dithering technique, that consists in adding a small Gaussian noise to the input signal (see the left part of *Figure 4*) in order to get a signal (see the right part of *Figure 4*) able to insure LSB toggling.

Dithering also insures that the quantization noise will be always a random one, independently from the input signal.

Figure 4. Dithering technique



For sure this noise must be as small as possible in order to avoid important degradation of the SNR.

The impact of SNR can be strongly reduced if the noise is shaped: for example, we can imagine that dithering noise is filtered in the wanted BW, and is only present out-off the wanted BW.

The embedded DAC can be used for generating the dithering signal.

If the application does not require the capture of signal smaller than the quantization step and if the quantization error can be considered as wideband noise, the dithering technique can be omitted. This is the case of the 12-bit SAR ADC embedded on STM32L0 and STM32L4 devices.

1.4 Overview of hardware oversampler in STM32 L0 and L4 microcontrollers

The HW oversampling engine in STM32 L0 and L4 microcontrollers accumulates the results of ADC conversions. The accumulated output data can be right-shifted (and rounded) to provide selected bit-depth in relation to OSR. The output value is not updated every sampling period, but once N samples are accumulated, therefore, the output data rate is decimated by factor of OSR.

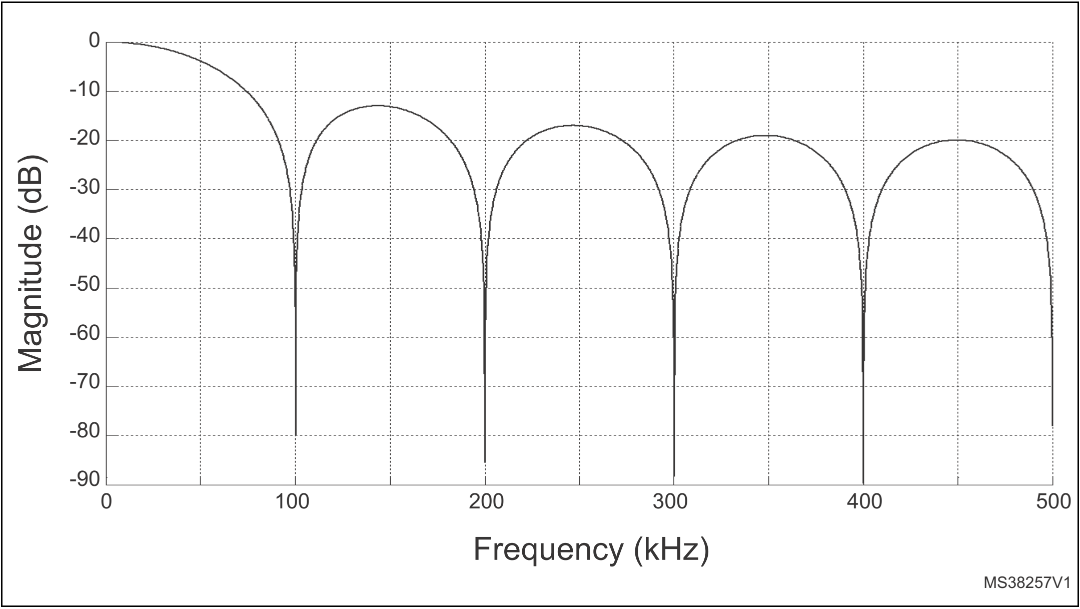
The result is average of accumulated samples as follows:

$$\text{Result} = \frac{1}{M} \times \sum_{0}^{N-1} \text{Conversion}(t_n)$$

where both N and M can be adjusted.

"Accumulate and average” stage can be thought of as a kind of digital filter (often called Accumulate-and-Dump). The frequency response of such filter is equivalent to a first order Cascaded-integrator-comb (CIC1) Hogenauer filter. The frequency response in case of sampling frequency 1 MHz and OSR=10 can be seen in *Figure 5*.

Figure 5. Frequency response of Accumulate-and-Dump filter



Even if not a perfect low-pass filter, it has an interesting property, i.e. the very high attenuation of the sampling frequency. Then, it’s effective in canceling the out of band noise resulting in increased signal to noise ratio.

2 Hardware oversampling to reduce power consumption

The ADC oversampling method can be implemented by hardware or by developing a dedicated software routine.

The advantage of hardware implementation is that the total energy budget needed for processing the ADC acquired samples is reduced in comparison to the software implementation where all the data processing needs to be done by the core.

Two test project emulating the common data acquisition tasks have been developed and executed on the same system to evaluate the energy difference and to demonstrate how much energy can be saved by using the hardware oversampling.

2.1 Software implementation

The project for demonstrating the software implementation of oversampling method is consisting in the following steps, repeated every 100 ms:

1. configuring the system/data acquisition
2. capturing of 64 samples by ADC and storing them in the memory by using DMA while the core is in SLEEP low-power mode
3. processing the acquired data by CPU to get oversampled value
4. putting the system in STOP mode for the rest of the 100 ms interval

2.2 Hardware implementation

The project showing the hardware implementation carries out the same task, except that the data processing is done by the ADC oversampling engine, hence the CPU can be inactive during the acquisition and oversampling:

1. configuring the system/ data acquisition
2. capturing of 64 samples and processing them by ADC oversampling engine while the core is in SLEEP low-power mode
3. putting the system in STOP mode for the rest of the 100 ms interval.

2.3 Results

The energy consumption for the data acquisition and processing task, and the average current consumption for the whole 100 ms period for both demonstration projects are detailed in *Table 2*.

Table 2. Comparison of SW and HW implementation of ADC oversampling technique

Implementation	Data acquisition and processing time	Acquisition task charge		Average current (during 100 ms)
Hardware	6.06 ms	896 pAh	3.23 µC	37 µA
Software	6.80 ms	1099 pAh	3.96 µC	44 µA

The HW oversampling implementation can save about 20% of the energy consumed to complete the acquisition and data processing task with lower coding effort and CPU time.

3 Conclusion

This application note has explained the basics of the oversampling technique used to improve the SNR performances (and thus the effective resolution) of ADCs integrated in microcontrollers of the STM32 L0 and L4 series.

The cornerstones of the oversampling technique are:

- the RMS quantization noise of an ADC is q / √(12), over the Nyquist bandwidth;
- if the wanted bandwidth is smaller than the Nyquist bandwidth, by using a filter to remove the out off band noise, the quantization noise is reduced in proportion;
- dithering can be used if the quantization noise does not behave like a wideband noise.

The hardware implementation of the ADC oversampling technique reduces the time and energy needed by the CPU for the data processing tasks associated with a software implementation, resulting in lowering the overall power consumption.

4 Revision history

Table 3. Document revision history

Date	Revision	Changes
25-Jun-2015	1	Initial release.