AN2668
Application note

Improving STM32F1 Series, STM32F3 Series and STM32Lx Series ADC resolution by oversampling

Introduction

The STMicroelectronics STM32F1 Series, STM32F3 Series and STM32Lx Series Cortex®-M3 based microcontrollers embed a 12-bit enhanced ADC, sampling with a rate up to Msamples/s. For most applications, this resolution is sufficient, but in some cases where a higher accuracy is required, the concept of oversampling and decimating the input signal can be implemented to save the use of an external ADC solution and to reduce the application consumption.

This application note gives two methods to improve the ADC resolution. These techniques are based on the same principle: oversampling the input signal with the maximum 1 MHz ADC capability and decimating the input signal to enhance its resolution.

The method and the embedded software (STSW-STM32014) given within this application note apply to both Medium- and High-density STM32F1 Series products as well as all the STM32F3 Series and STM32Lx Series products. Some specific hints are given at the end of the application note to take advantage of the implementation of the DAC peripheral and the ADC dual mode into some STM32F1 Series, STM32F3 Series and STM32Lx Series devices.

This application note is split into two main parts: the first part describes how the oversampling increases the ADC-specified resolution while the second one describes the guidelines to implement the different methods available and gives the embedded software flowchart of their implementation on the STM32F1 Series, STM32F3 Series and STM32Lx Series devices.

Contents

List of tables

List of figures

1 General information

This application note applies to Arm®-based devices.

arm

2 Definition of ADC signal-to-noise ratio

The ADC gives a representation of an analog signal among a finite number of digital words. Since the digital domain is represented by a finite number of words which have to present a continuous signal, the conversion step introduces the quantization error function of the ADC input range and resolution.

For an ideal ADC, the quantization error is between ±0.5 LSB. In the case where the input signal is varying through many levels between samples, and the sampling rate is not synchronized with the input frequency, the quantization error can be considered as a white noise whose energy is uniformly spread from the DC domain to half of the sampling frequency. Refer to *Appendix A* for more details regarding the calculation of its density.

The SNR (signal-to-noise ratio) is the ratio of the ADC noise to the input signal power. For an ideal ADC, it is assumed that the SNR is equal to the quantization noise (no other noise source is considered) to the input signal. It is demonstrated that for a full-scale sinusoidal signal, the ADC SNR is maximum and given by the following formula:

$SNR_{dB} = 6.02N + 1.76$, where N is the ADC resolution.

It can be easily noticed that when the SNR increases, the ADC effective number of bits increases.

For a real ADC, different error sources must be considered: offset, gain, INL (integral nonlinear) and DNL (differential nonlinear). A brief description of these errors can be found in the STM32F1 Series, STM32F3 Series and STM32Lx Series datasheets. These errors degrade the ideal ADC resolution and determine the real effective number of bits of the ADC.

Improving the SNR enhances the ADC effective number of bits.

The following section demonstrates that sampling the input signal with higher rates than the Nyquist frequency improves the SNR. The Nyquist frequency is introduced in the next paragraph.

3 Nyquist theorem and oversampling

The Nyquist theorem states that in order to reconstruct the analog input signal, the signal must be sampled at a rate f_S (sampling frequency) that is greater than twice the maximum frequency component of the input signal.

The non-compliance with the Nyquist theorem causes aliasing effects and the analog signal cannot be fully reconstructed from the input samples. Therefore, for most applications, a low-pass filter is required at the ADC input to filter the frequencies lower than half of the sampling frequency. It is difficult to handle the filter constraints with low sampling frequencies.

The oversampling consists in sampling the input analog signal at rates higher than the Nyquist frequency limit, filtering the samples and reducing the sample rate by decimation. Using this method relaxes the anti-aliasing low-pass filter constraints.

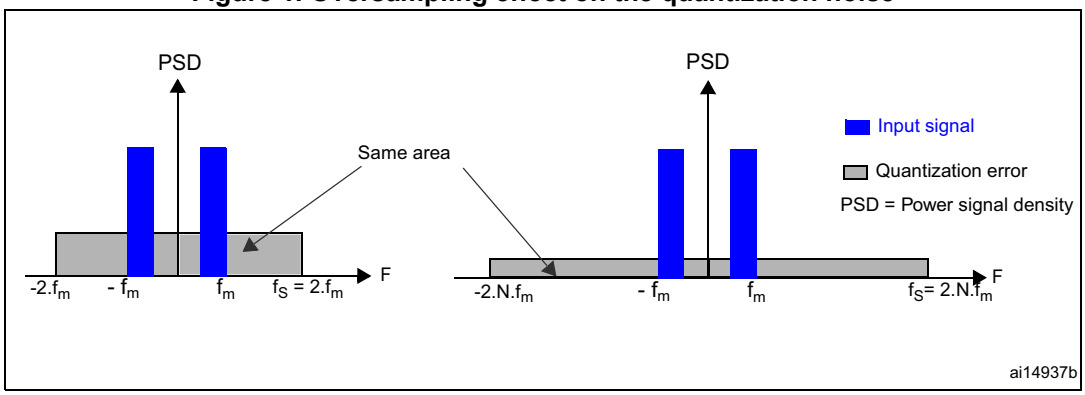
4 Oversampling using white noise

4.1 SNR of oversampled signal with white input noise

Assuming that the quantization noise is assimilated to a white noise, its power density is uniformly distributed between DC and half the Nyquist frequency. This power density is independent of the sampling frequency.

When sampling at higher rates, the quantization noise is spread over the bandwidth of the sampling frequency.

Figure 1. Oversampling effect on the quantization noise



According to *Figure 1*, when sampling the input signal at higher rates, the same noise power, represented by the area of the gray rectangle, is spread over a bandwidth equal to the sampling frequency which is much greater than the signal bandwidth fm. Only a relatively small fraction of the total noise power falls in the [–fm, fm] band, and the noise power outside the signal band can be greatly attenuated with a digital low-pass filter.

Reducing the quantization noise enhances the signal-to-noise ratio and, consequently, the ADC effective number of bits. Oversampling the input signal of OSR times faster than the Nyquist frequency gives the following SNR

$$SNR_{OVS} = 6.02 \cdot N + 1.76 + 10\log(OSR)$$

In conclusion each doubling of the sampling frequency reduces the in-band noise by 3 dB, and increases the measurement resolution by 1/2 bit. Therefore, 6dB SNR gain is required to add 1 resolution bit to the ADC.

In general, if p additional bits are required by the application then, the ADC sampling frequency should be at least:

$F_{OVS} = 4^p F_S$, where F_S is the current ADC sampling frequency used.

4.2 Decimation

The conventional meaning of averaging is adding m samples and dividing the result by m. Averaging several data from an ADC measurement is equivalent to a low-pass filter which attenuates the signal fluctuation and noise. The average method is often used to smooth and remove speaks from the input signal.

Note that a normal averaging does not increase the resolution of the conversion because the sum of m N-bit samples divided per m is an N-bit representation of the sample.

The decimation is an averaging method. When combined with the oversampling, the decimation improves the ADC resolution.

In fact, adding 4^p ADC N-bit samples, gives a representation of the signal on N+2p bits. In order to have p additional effective bits, the sum is shifted to the right by p bits.

This FIR filter with equal filter coefficients enables the user to filter the oversampling frequency by giving an output sample computed from the OSR input samples.

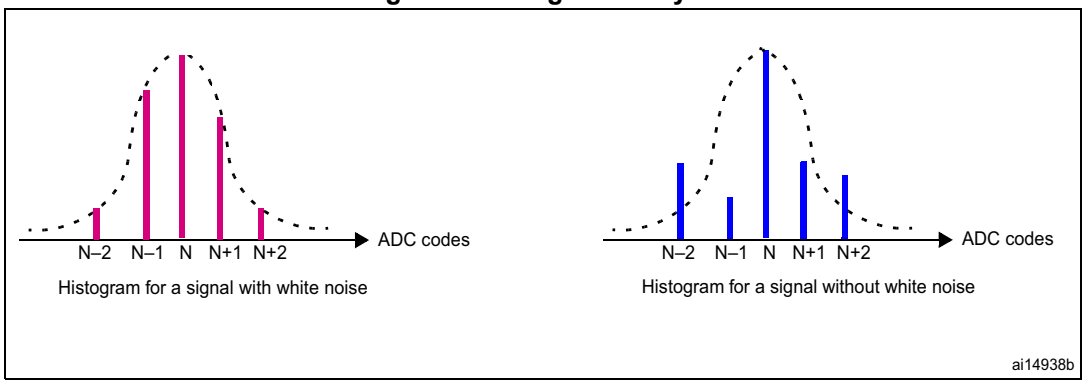
The oversampling method limits the maximum input frequency bandwidth. In fact, in the case of the STM32F1 Series, STM32F3 Series and STM32Lx Series (with maximum sampling rate around 1 Msps), signals having components up to 500 kHz can be processed by the ADC. If for example, two additional resolution bits are required, then the maximum input frequency that can be entered is 500 kHz/16 = 31.25 kHz when the oversampling is using a white noise.

4.3 When is this method efficient?

For the oversampling and decimating method to work properly, the following requirements must be satisfied:

- There should be some noise in the input signal. This noise must approximate white noise with a uniform power spectral density over the frequency band of interest.
- The noise amplitude must be sufficient to toggle the input signal randomly from sample to sample by an amount of at least 1 LSB. Otherwise, the input samples would have the same representation and the sum and average operations would not give any extra resolution. For most applications, the internal ADC thermal noise and the input signal noise are sufficient to use this method. In the case where the thermal noise does not have a high-enough amplitude to toggle the input signal randomly, then an artificial white noise should be injected into the input signal. This operation is referred to as “dithering”. Regarding this point, two questions can be raised. The first is “How to evaluate the ADC noise and test its Gaussian criteria?” and “How to generate white noise if needed?”
  - A practical way of detecting the Gaussian criteria of the input signal noise is to see the distribution of a clean DC signal over the ADC codes. The histogram method can be used to verify if the input noise follows a Gaussian distribution. The example in *Figure 2* shows two possible situations.

Figure 2. Histogram analysis



- In the case where external noise dither must be added to the input signal, then the thermal noise generated by a diode or a resistor can be injected into the input signal.
- The input noise must not correlate with the useful input signal and the input signal should have equal probability to be between two adjacent ADC codes. This means that for systems using feedback process, this method does not work.

4.4 Method implementation on the STM32F1 Series, STM32F3 Series and STM32Lx Series devices

This method describes the different steps undertaken to implement and test the oversampling method on the STM32F1 Series, STM32F3 Series and STM32Lx Series devices.

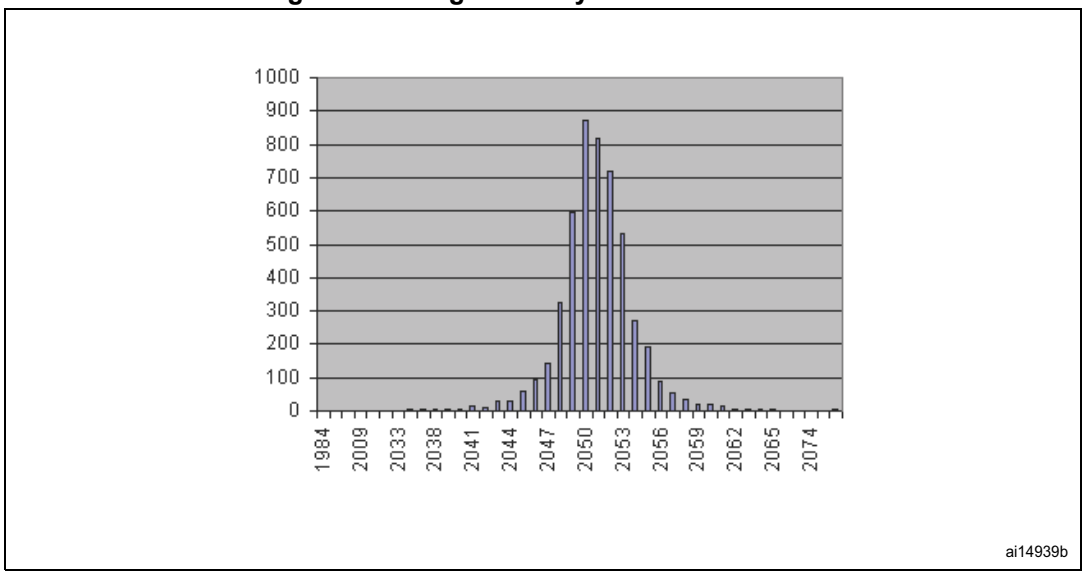
According to the previous section, in order to make this solution work properly, there must be some white noise to make the input signal toggle randomly by 1/2 LSB. For this, the application environment noise must be considered.

The first step consists in computing the ADC thermal noise to conclude if external white noise must be injected into the input signal. In a typical application board, the computed noise does not include only the ADC internal noise but also the possible noise generated by the different board components and the layout. Therefore, this evaluation depends on the application board but the methodology remains the same.

The histogram method is used for different DC input voltages. This input voltage is sampled a large number of times (example 5000). The related distribution can be easily interpreted using a spreadsheet.

For example, for a 1.65 V DC input voltage applied on the STM3210B-EVAL evaluation board, the histogram shown in *Figure 3* is detected.

Figure 3. Histogram analysis for DC = 1.65 V



The ADC thermal noise can be computed from this histogram (although this can be shown, it is not the objective of this application note and the details are not offered here).

In order to carry on this ADC noise test, the user must do the following:

- Uncomment the line `#define Themal_Noise_Measure` in the oversampling.h file
- Configure the `Total_Samples_Number` which is the number of ADC conversion operations. It must be smaller than 65535. The DMA channel is configured to store the number of ADC samples in a RAM buffer. At the end of the transfer, an interrupt is generated and the number of occurrences of each ADC code is computed
- In order to compute the occurrence of the ADC codes, a variable giving the relevant ADC codes is defined

When the code is run, `Relevant_ADC_Samples` ADC samples and their corresponding number of occurrences are displayed on the HyperTerminal. The HyperTerminal configuration is 8-bit data, no parity, 115 200 baud rate. If the effective number of ADC samples found is smaller than the defined `Relevant_ADC_Samples` variable, then 0 is displayed for both ADC code and ADC code occurrences. The user can capture them and build a histogram.

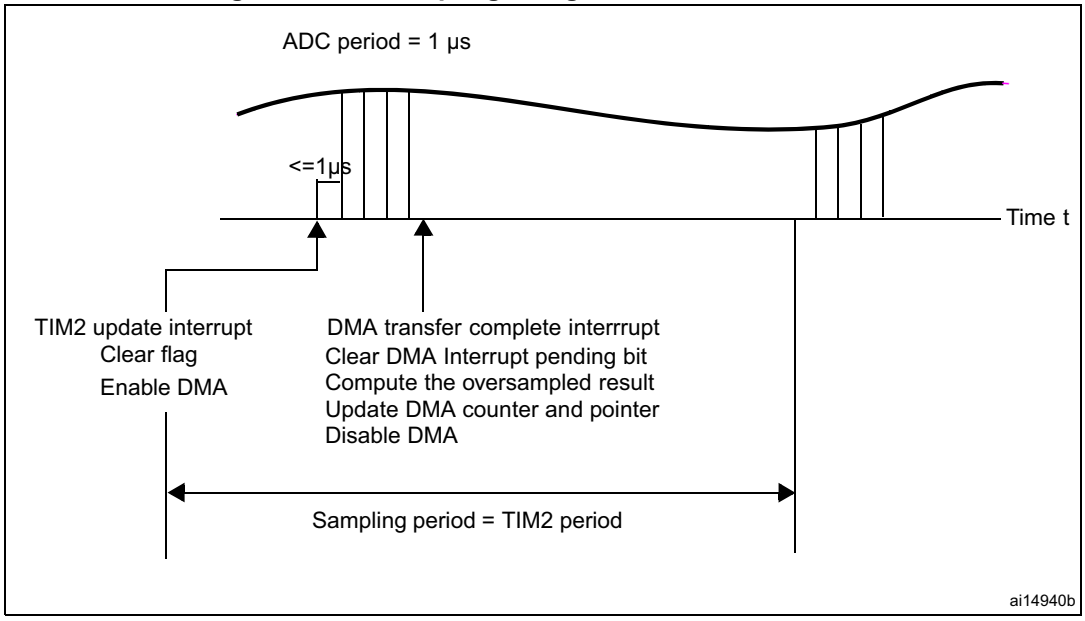
4.4.1 Oversampling using a white noise embedded software flowchart

The STM32F1 Series, STM32F3 Series and STM32Lx Series on-chip ADC conversion frequency is fixed to 1 MHz. The ADC DMA channel is configured to transfer the number of oversampled inputs from the ADC data register to a buffer in RAM. This transfer is configured to occur one time. At the end of the DMA transfer, an interrupt is triggered and the oversampled result is computed.

The general-purpose timer TIM2 is used to generate the input signal sampling frequency. For this, the TIM2 reference clock is configured at 1 µs. Its period determines the input signal sampling period. It is defined in the oversampling.h file as `#define Input_Signal_Sampling_Period`. When the TIM2 update interrupt is triggered, the

DMA is re-enabled and the converted ADC values can be treated.
Figure 4 summarizes the implemented functionality.

Figure 4. Oversampling using a white noise flowchart



The oversampled data are computed in the DMA transfer complete interrupt. For synchronization reasons, it is recommended to read it in the second TIM2 interrupt.

Note that with this implementation, the TIM2 period must be greater than the time required by the ADC to convert OSR samples, and greater than the ADC interrupt execution time.

If the sampling frequency required by the application is exactly OSR µs, then the user is not required to use the timer TIM2 to generate the input sampling frequency. However, the DMA must be configured to be functional in continuous mode and the DMA transfer complete interrupt must be updated accordingly. The oversampled data are usually computed in the DMA transfer complete interrupt.

4.4.2 Oversampling using white noise result evaluation

In order to evaluate the oversampling method, the user must uncomment the `#define Oversampling_Test` line and configure the number of samples with the enhanced resolution.

When this line is uncommented, a buffer is created in the RAM to store the oversampled data. The buffer contents are then displayed on the HyperTerminal. The HyperTerminal configuration must be 8-bit data, no parity and 115 200 baud rate. The user can capture them into a txt file and then compare the expected results to the real ones.

In order to evaluate the new enhanced ADC, a ramp with a 50 Hz frequency and a 1 V amplitude is input in the ADC and sampled using the oversampling algorithm every 100 µs.

The embedded software example related to this method is located in the `WhiteNoiseMethod` folder.

Figure 5. Ramp samples with 1 additional bit

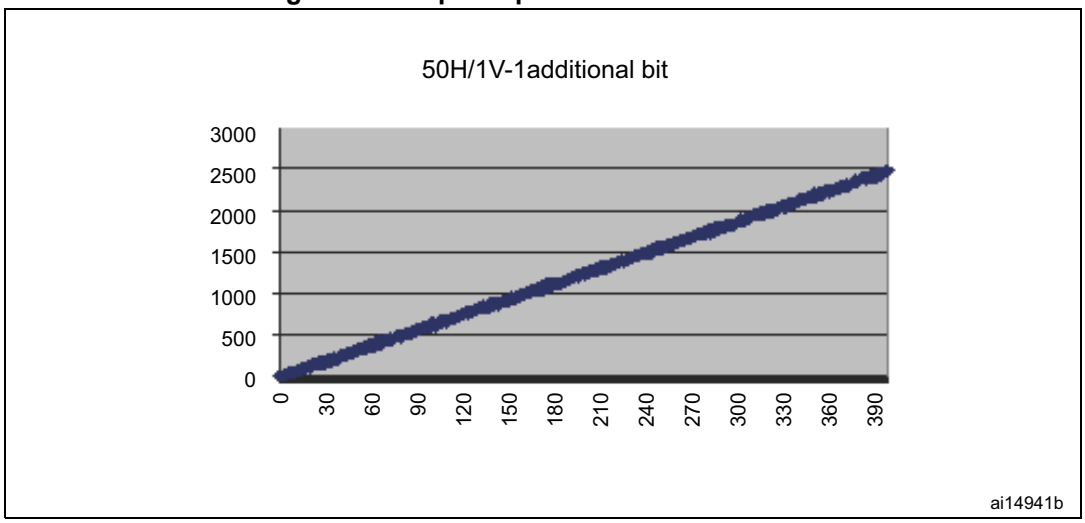
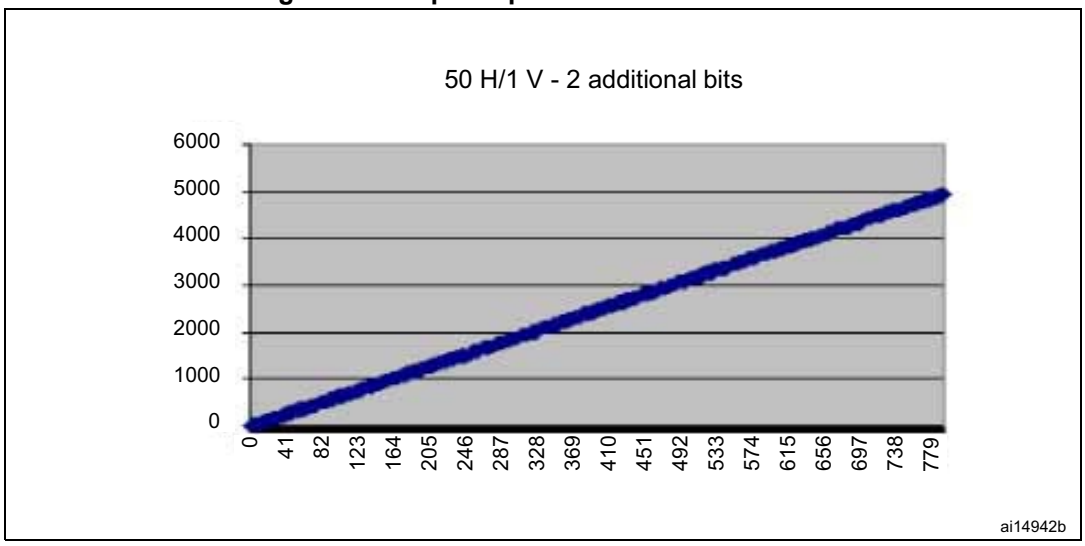


Figure 6. Ramp samples with 2 additional bits



The oversampling algorithm using white noise is run with the same ramp (50 Hz frequency and 1 V amplitude). Both *Figure 5* and *Figure 6* give the ADC oversampled data as a function of time in µs. *Figure 5* is the result of adding one bit while *Figure 6* is the result of adding two additional bits to the ADC on-chip resolution.

When the ramp is sampled without using any extra software resolution, with a 3.3 V reference supply, 1 V corresponds to the digital value 1250.

When one additional bit is added, 1 V is sampled as 2500 and when two additional bits are added, 1 V is sampled as 5000.

This means that the environment contains enough noise for this method to work.

5 Oversampling using triangular dither

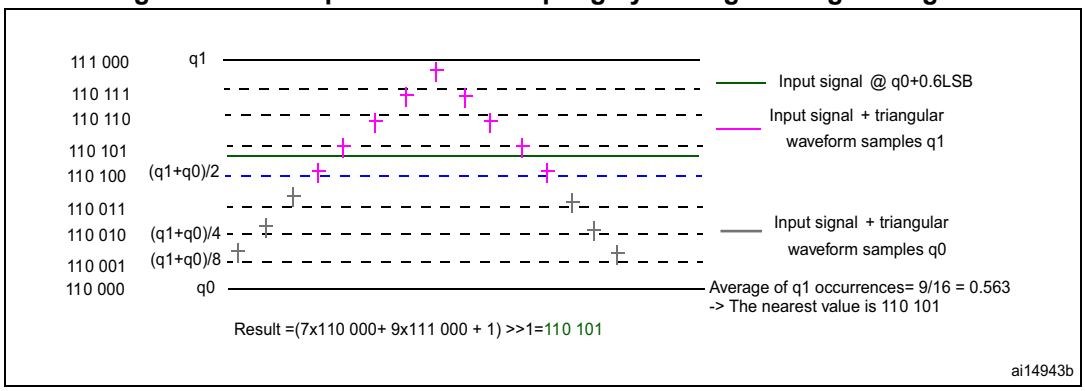
Assuming that the input signal is between two successive quantization steps q0 and q1 during the oversampling period, then the converter may convert it either to q0 or q1. Adding extra p bits of resolution means determining the relative position of the input signal between q0 and q1.

With the addition of an appropriate triangular signal, the quantizer generates a series of q1s and q0s. Averaging the q1 occurrences over a given interval determines the relative position of the input signal between the lower and the higher quantization steps.

The theory states that the best results are achieved when dithering the input signal using a triangular waveform with a period of OSR times the ADC sampling period and an amplitude of n+0.5LSB where n = 0,1,2,3.

The theory behind this methods is quite complicated, so that *Figure 7* serves as an example to illustrate how this method works. In this example, the ADC on-chip resolution is 3 and three extra bits are added by embedded software. The input signal is assumed to have an amplitude of q0+ 0.6LSB (q0 = 6 in this example). In order to add three additional bits, the input signal is sampled 2.2^3 times (16 times).

Figure 7. How to perform oversampling by adding a triangular signal



If the input signal is not correlated with the triangular waveform, then it is demonstrated that the gain in the SNR is equal to

$$SNR_{Gain} = 20.\log\left(\frac{OSR}{2}\right)$$

Therefore, each doubling of the sampling frequency improves the SNR by 6dB and adds 1 ADC bit resolution.

In general, in order to add p-bit extra resolution, the oversampling frequency must be equal to

$$F_{OVS} = 2.2^{p}F_S$$

5.1 When does this method work?

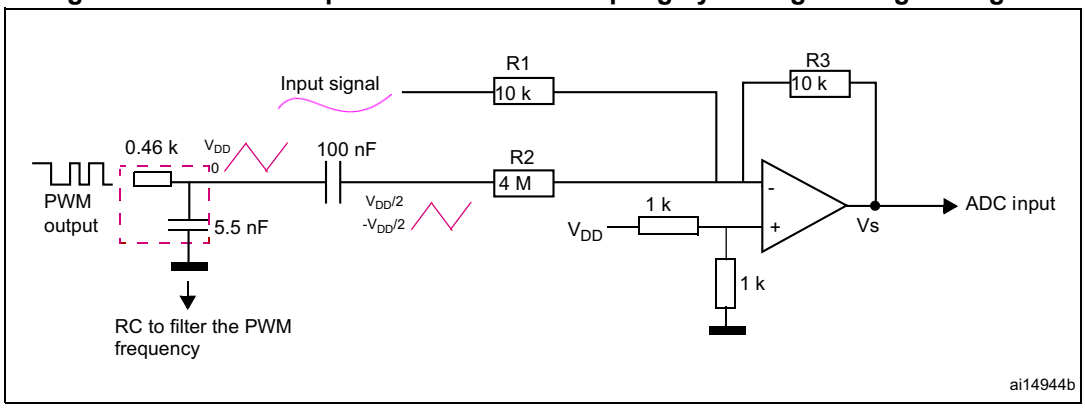
In order to make this method work, the input signal must not vary by more than ±0.5LSB during the oversampling period and must not correlate with the triangular dither signal.

5.2 Method implementation on STM32F1 Series, STM32F3 Series and STM32Lx Series devices

In order to implement the second solution, the following is needed:

- An operational amplifier to perform the sum of the input signal and the triangular waveform. For this, an op-amp inverter/summing stage is required. The ST component LMV321 can be used.
- A triangular waveform with a period of OSR times the ADC conversion rate. The user can either use a signal generator or one of the on-chip timers and an RC network to generate this triangular signal. Indeed, the on-chip timer generates a PWM signal with a duty cycle varying from 0 to 100%. This PWM output can be filtered with an RC filter to generate a triangular signal varying from 0 to V_{DD}. In order to generate an amplitude of 0.5LSB, then the output is first passed through a capacitor (to cut the DC component) and then divided by the prescaler R2/R3 (see *Figure 8*). This prescaler is equal to the ADC number of words.
- The input signal must not be changed after the op-amp. For this reason, R1 should be equal to R3.
- The sum of the input signal and the triangular dither is inverted. For this purpose, a 3.3 V offset is required on the positive entry of the op-amp. After the oversampled data are computed, this offset is subtracted to give the input signal estimation with an extra resolution.

Figure 8. Hardware requirements of oversampling by adding a triangular signal



5.2.1 Oversampling using triangular dither embedded software flowchart

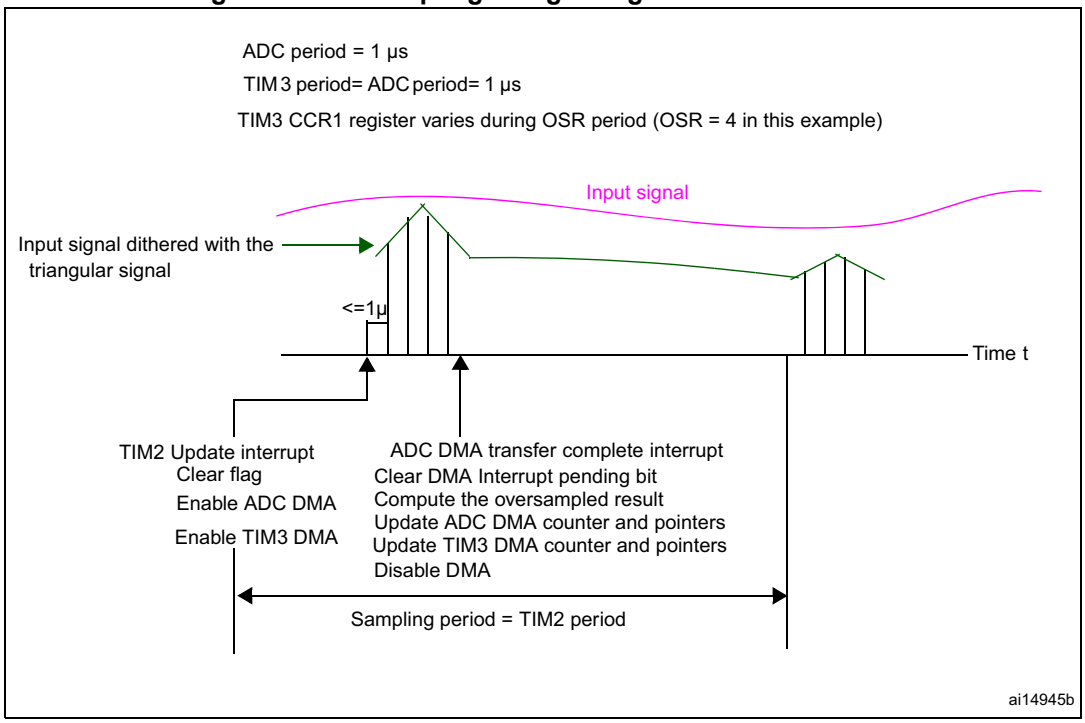
The STM32F1 Series, STM32F3 Series and STM32Lx Series on-chip ADC conversion frequency is fixed at 1 MHz. The ADC DMA channel is configured to transfer the number of oversampled inputs from the ADC data register to a buffer in RAM. This transfer is configured to occur one time. At the end of the DMA transfer, an interrupt is triggered and the oversampled result is computed.

The general-purpose timer TIM2 is used to generate the input signal sampling frequency. For this, the TIM2 reference clock base is configured at 1 µs. Its period determines the input signal sampling period. It is defined in the *oversampling.h* file by `#define Input_Signal_Sampling_Period`.

The triangular dither is generated using the timer TIM3 configured in PWM mode by updating the Capture Compare Register CCR1. The timer TIM3 period must be equal to the ADC conversion rate and CCR1 must be updated OSR times where OSR is the oversampling factor. In order to do this, the possible CCR1 values are first computed and stored into a RAM buffer, then the DMA transfer is used to update the CCR1 register, removing the need for interrupts.
Note that the ADC conversion rate limits the oversampling factor. For example, in the case where the ADC is running at 1 MHz, the STM32F1 Series is operating at 56 MHz. In order to have a period of 1 µs, the auto-reload register of the timer TIM3 must be equal to 55. The maximum number of additional bits is then 4.

When a TIM2 update interrupt is triggered, the ADC and TIM3 DMA are re-enabled and the converted ADC values can be treated to compute the new sample with the extra resolution bits. *Figure 9* summarizes the implemented functionality.

Figure 9. Oversampling using triangular dither flowchart



For this method to work, the input signal must not vary by more than ±0.5LSB during the oversampling period. This means that for STM32F1 Series, STM32F3 Series or STM32Lx Series devices operating from a 3.3 V V_{REF+}, the maximum allowed variations of the input signal during the oversampling period is ~0.4 mV.

On the other side, a triangular waveform with an amplitude of 0.5 LSB means a 0.4 mV amplitude when operating the STM32F1 Series, STM32F3 Series or STM32Lx Series from a 3.3 V V_{REF+}. The application environment must therefore not be very noisy. Any disturbance of the triangular waveform will have an impact on the computed oversampled data.

According to the implementation, the triangular waveform is generated by means of the STM32 timer and an RC filter that cuts the 1 MHz timer frequency. The timer PWM output signal is integrated to provide a triangular signal with a 3.3 V amplitude. The division is done with the ratio R3/R2.

The embedded software related to this method is located in the `TriangularDitherMethod` directory.

6 Comparing the first and second methods

The first method based on oversampling and averaging using white noise provides a half-bit additional resolution for each doubling of the oversampling rate. The maximum input frequency is drastically decreased with the additional number of additional bits.

For applications where this gain is sufficient, it is a good choice. It requires the presence of white noise in the input signal to make the signal toggle between two adjacent ADC codes. In general, the ADC thermal noise is sufficient and there is no need to add external hardware to act as an external white noise source. This makes the solution more cost-effective.

The second method based on dithering the input signal using a triangular waveform and computing its relative position between two quantized steps provides one more bit for each doubling of the oversampling rate. This is twice the improvement given by the first method. To make this method work, the input signal must not correlate with the triangular signal and must not have a variation greater than 0.5LSB during the oversampling period. However, external hardware is needed to add the input signal and the triangular waveform.

Table 1 summarizes the main differences between the two methods. It is not possible to say that one method is better than the other. Each method has its advantages and limitations. The user must select the one that better meets their application requirements (sampling frequency, number of effective bits etc.).

Table 1. Oversampling using white noise vs. oversampling using triangular dither

Implementation conditions	Oversampling using white noise	Oversampling using triangular dither
Oversampling factor to add p bits to he ADC on-chip resolution	4^p	2.2^p
Maximum Input signal frequency	$f_{ADC}max/(2.4^p)$	$f_{ADC}max/(2.2.2^p)$
Dither signal	White noise with an amplitude of at least 1 LSB	Triangular signal with an amplitude of n+0.5LSB
External hardware	External white noise source needed if the input signal noise is not sufficient	Triangular waveform generator: an on-chip timer can be used. In this case, an RC network is used to filter the PWM frequency An op-amp is needed to add the triangular waveform and the input signal

7 Hints

7.1 What is the maximum number of bits that can be added to the on-chip ADC resolution?

It can be easily shown that increasing the on-chip ADC resolution decreases the maximum frequency component of the input signal.

For example, when using the STM32F1 Series, STM32F3 Series or STM32Lx Series ADC at 1 MHz and two additional bits are required by the application, then the maximum input frequency is divided by:

- 16 when using the white noise method (62.5 kHz).
- 4 when using the triangular dither method (125 kHz)

What is the maximum number of bits that can be added to the on-chip ADC resolution?

For the two methods, the estimation of the input signal is done during an oversampling period of OSR times the ADC conversion rate. In the case, the ADC is running at 1 MHz, the input signal estimation is done over OSR µs. The signal must not vary by more than 1/2LSB for the white noise method and, by ±0.5LSB for the triangular waveform method.

- When using the white noise method, the maximum number of bits that can be added to the ADC resolution depends only on the input signal.
- When using the triangular dither method, the maximum number of bits that can be added to the ADC resolution does not depend only on the input signal. In fact, the steps defining the triangular signal depend on the ADC and APB frequencies. The timer period should be equal to the ADC rate:

  $2.2p \leq$ Timer period

  $P \leq \log2$ (Timer period / 2)

In our example, running the ADC with a rate of 1 µs causes the STM32F1 Series to operate at 56 MHz, which means that the timer period must be equal to 55. The maximum number of bits that can be added in this case is 4.

7.2 Taking advantage of STM32 DAC implementation

Some STM32F1 Series, STM32F3 Series and STM32Lx Series devices come with a DAC (digital-to-analog converter) that can be used in the oversampling method to avoid the use of external components.

The DAC can be used in the two oversampling methods as follows:

- In the first method, the DAC can be used to generate a white-noise waveform with programmable amplitude that can be injected into the input signal if noise is not sufficient. The waveform is generated thanks to the implemented pseudo-random algorithm. For more details, refer to the STM32F1 Series, STM32F3 Series and STM32Lx Series reference manuals.
- In the second method, the DAC can be used to generate the triangular waveform. This removes the need for any additional external RC circuitry to filter the timer PWM frequency.

Note: This hint is not implemented in the software given within the application note.

7.3 Taking advantage of the STM32F1 Series, STM32F3 Series and STM32L4 Series dual ADC mode implementation

In some STM32F1 Series, STM32F3 Series and STM32L4 Series devices, the dual ADC mode is an interesting feature that allows two ADCs to convert at the same time.

Using the dual ADC fast interleave mode, the same channel is converted alternately by ADC2 and ADC1. The time separating two successive samples is 7 ADC clock cycles. The input signal is therefore oversampled faster. In the example described in this application note, a sample is obtained every 1 µs. Using the dual ADC fast interleave mode, it is possible to have a sample every 7 ADC clock cycles, that is every 0.5 µs when running the ADC at 14 MHz.

Note: This hint is not implemented in the software given within the application note.

7.4 Taking advantage of the STM32L0 Series hardware ADC oversampling implementation

On the STM32L0 Series and STM32L4 Series devices, the ADC implements the oversampling feature in hardware.

In the hardware oversampling ADC mode, the oversampling (averaging) is computed to the final result with an increased resolution by hardware. The ADC internally performs predefined numbers of data conversions which are averaged to one standard final ADC result. The CPU/software intervention is then decreased which leads to lower microcontroller consumption and less program memory usage. The final oversampled sample is the same as the one got with the software oversampling method.

Note: For a comparison of the hardware and software oversampling usage see 'ADC hardware oversampling for microcontrollers of the STM32L0 Series and STM32L4 Series' application note (AN4629).

Appendix A Quantization error

Assuming that the user has an N-bit analog-to-digital converter (ADC) and a voltage reference V_{AREF}.

The quantum q being the minimum distance between two adjacent ADC codes. It is defined as follows:

$$q = \frac{V_{AREF}}{2^N}$$

The quantization error equation is:

$$|e_q| \leq \frac{q}{2}$$

Assuming that:

- The signal crosses many levels between samples
- The sampling rate is not synchronized to the signal frequency
- The input signal has equal probability of being anywhere in the quantization interval q, leading to a random quantization error

Given the above assumptions, the quantization noise can be approximated to a random variable equally distributed between ADC codes with zero mean. From this assumption, it can be easily demonstrated that the quantization noise variance is given by the following formula

$$\sigma^2 = E(e_q^2) = \int_{-\frac{q}{2}}^{\frac{q}{2}} (P(e_q) \cdot e_q^2) de_q = \frac{q^2}{12}$$

According to the above formula, the quantization noise power depends on the ADC resolution and decreases drastically when the ADC resolution increases.

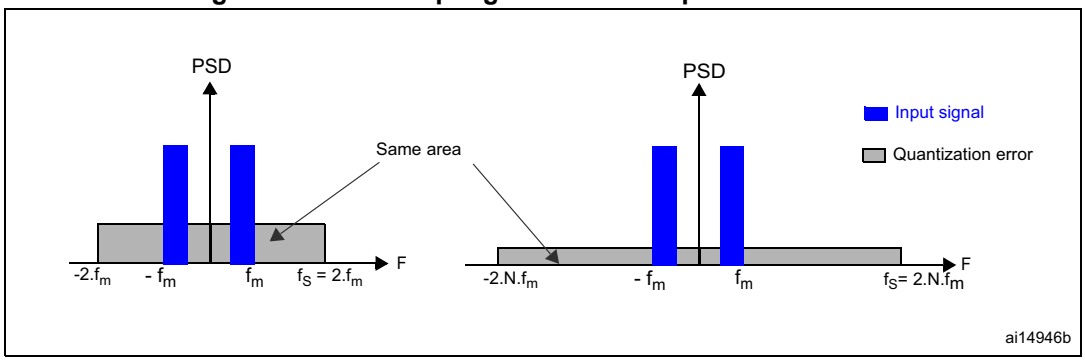
Given an ADC sampling frequency f_S (which is specified according to the MCU), in the case where the Shannon criteria is respected, then the quantization noise power density is equal to

$$PSD = \frac{\sigma^2}{f_s}$$

Let f_m be the maximum frequency component of the input signal. The quantization noise power present in the band of interest is given by

$$\eta_b^2 = \int_{-f_m}^{f_m} \frac{\sigma^2}{f_s} df = \sigma^2 \cdot \frac{2f_m}{f_s} = q^2 \cdot \frac{2f_m}{12.f_s}$$

Figure 10. Oversampling effect on the quantization error



Note that increasing the sampling frequency reduces the in-band quantization noise power and consequently improves the signal-to-noise ratio.

Given the same input signal and sampling it with $2.f_m$ and $f_S = OSR.2.f_m$, the gain in SNR is

$$SNR_{OVS} = 10\log\left(\frac{\sigma_x^2}{\sigma^2 \cdot \frac{2f_m}{f_s}}\right) = SNR_{2.fm} + 10\log(OSR)$$

Revision history

Table 2. Document revision history

Date	Revision	Changes
08-Jul-2008	1	Initial release.
23-Sep-2013	2	Added STM32L1x products Added subsection title *Section 5.2.1: Oversampling using triangular dither embedded software flowchart* in *Section 5.2: Method implementation on STM32F1 Series, STM32F3 Series and STM32Lx Series devices* Updated '3.3V V_{REF+}' and 'R3/R2' in *Section 5.2.1: Oversampling using triangular dither embedded software flowchart* Updated title of *Section 7.2: Taking advantage of STM32 DAC implementation* Updated title of *Section 7.3: Taking advantage of the STM32F1 Series, STM32F3 Series and STM32L4 Series dual ADC mode implementation*
12-Dec-2017	3	Updated the whole application note adding the STM32F3 Series, STM32L0 Series and STM32L4 Series products. Added *Section 7.4: Taking advantage of the STM32L0 Series hardware ADC oversampling implementation*.