AN2719
Application note

Precision improvement techniques for the A/D converter of the STM8 microcontroller

Introduction

This application note describes a selection of hardware and software methods for improving the precision of the on-chip A/D converter of the STM8 microcontroller (STM8 ADC). It is divided into three sections:

- The first section explains the internal design principle of the STM8 ADC.
- The next section lists the main types of error that occur in A/D converters and their sources.
- The last section describes some hardware and software methods for minimizing these errors.

Firmware examples (source codes) are provided with this application note showing how to implement STM8 ADC routines for minimizing measurement errors.

Content

1 STM8 ADC internal hardware

STM8 family microcontrollers include an Analog to Digital Converter of the switched-capacitor type. This ADC type uses the SAR (Successive Approximation Register) principle, by which the conversion is performed in several steps. The number of conversion steps is equal to the number of bits in the ADC converter.

1.1 SAR principle

Figure 1 to *Figure 6* show the first conversion steps.

Figure 1. Basic schematic of switched-capacitor ADC

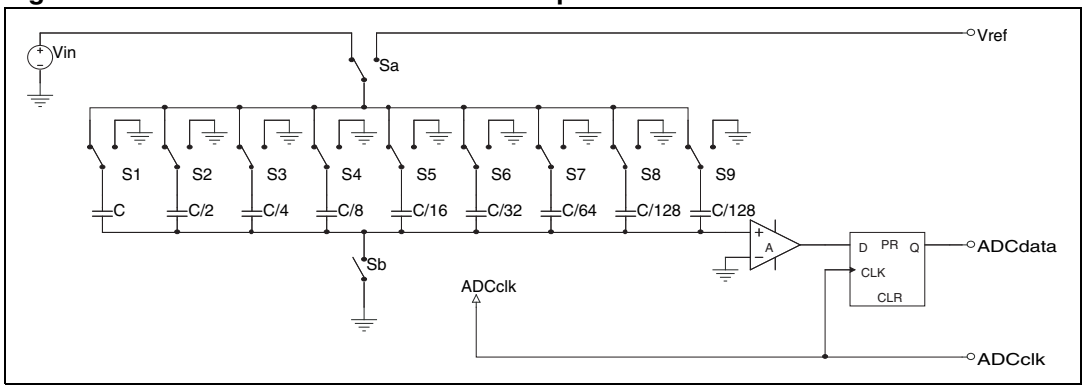


Figure 2. Sample phase

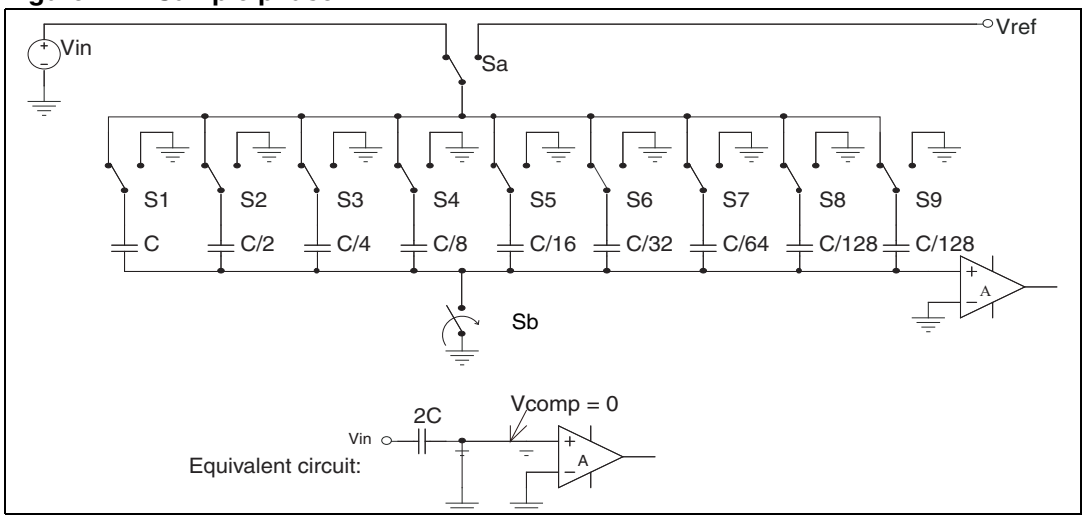


Figure 3. Hold phase

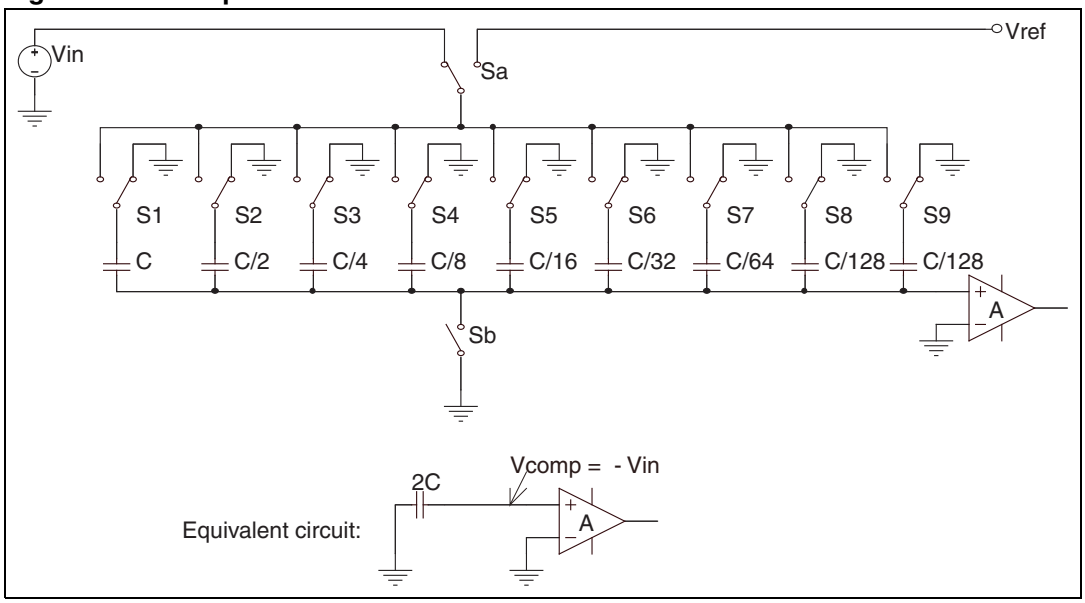


Figure 4. Step 1 compare with $V_{REF}/2$

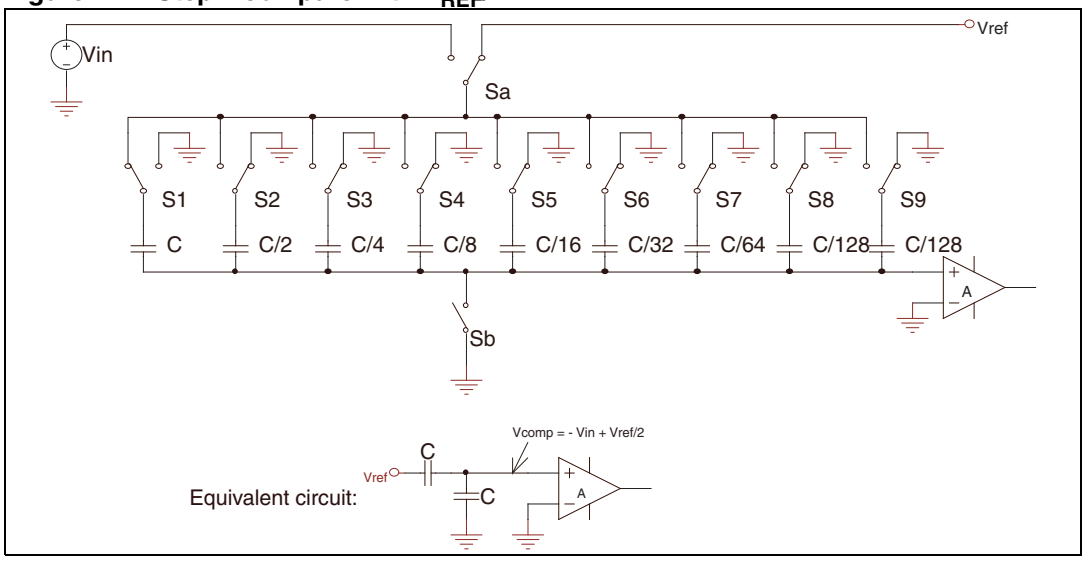


Figure 5. Step 2, if MSB = 1 then compare with 3/4 V_{REF}

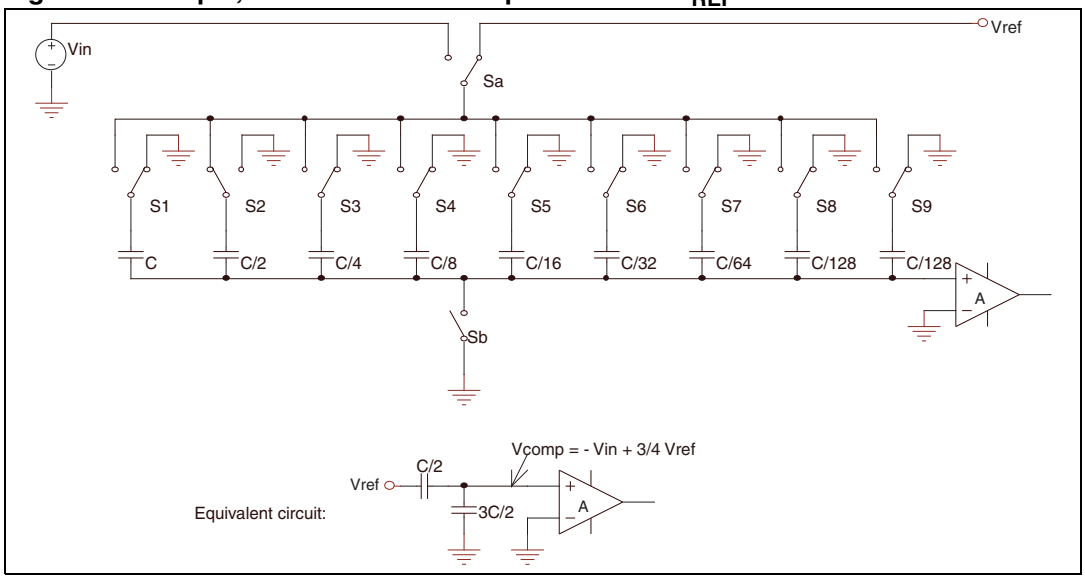
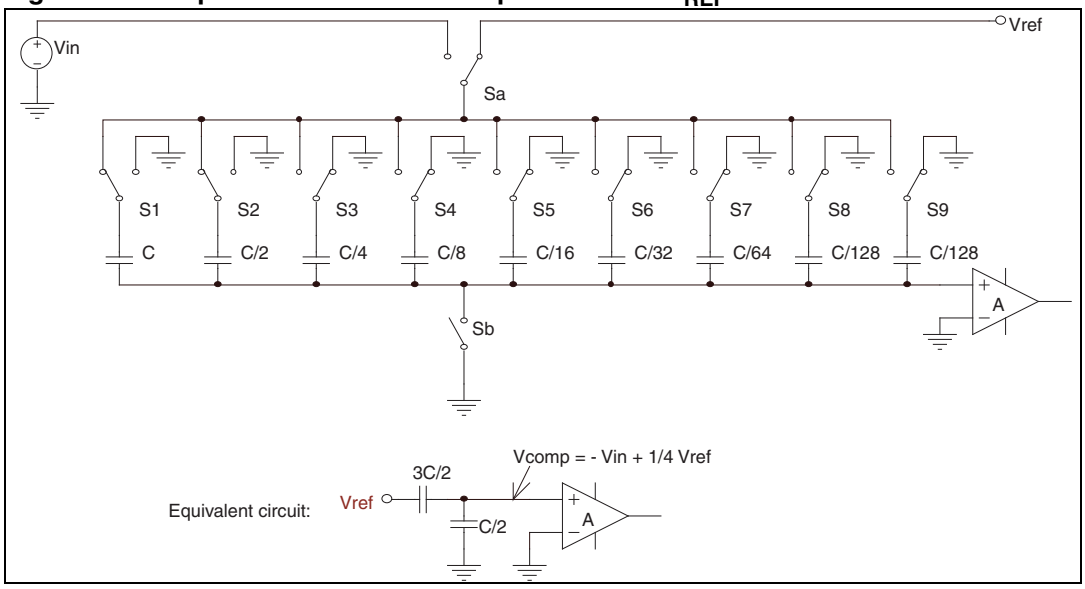


Figure 6. Step 2 if MSB = 0 then compare with 1/4 V_{REF}



1.2 ADC clock

The ADC is driven by a clock derived from the MCU master clock through a programmable divider. This allows you to select the ADC clock speed according to your application requirements. The conversion and sampling speed depends on ADC clock. Each conversion step (described in *Figure 4* to *Figure 6*) is performed in one ADC clock cycle - so 10-bit conversion takes 10 cycles. The sampling period is 3 clocks and the synchronisation period takes 1 clock. The total conversion time is actually 14 cycles.

1.3 Reference voltage

The reference voltage is either internally connected to analog power supply pins or connected to external pins where you can connect a reference voltage source. This reference voltage connection option depends on the given STM8 package and STM8 device type. The reference voltage has big influence to ADC precision, therefore care must be taken with it in the application design (stability, noise, ...).

1.4 Multiplexer

The ADC has an input multiplexer which is used to select one of the STM8 input pins as the analog input to the ADC.

2 ADC errors

2.1 Introduction

This chapter lists the main errors which have an effect on A/D conversion accuracy. These types of error occur in all A/D converters and conversion quality depends on eliminating them. You can find values for these errors specified in the ADC characteristics section of any STM8 datasheet. The datasheets also include sections describing sources of error or methods for minimizing them.

2.2 Linearity errors

2.2.1 Differential nonlinearity

Differential nonlinearity (DNL) shows how far a code is from a neighboring code. The distance is measured as a change in input voltage magnitude and then converted to LSBs. The best ADC performance is specified as "no missing codes". This means that if the input voltage is swept over its range, all output code combinations will appear at the converter output. A DNL error of <±1LSB guarantees no missing codes. With a DNL equal to -1LSB, the ADC does not guarantee to have no missing codes. With a DNL greater than -1, the device has missing codes.

Figure 7. DNL: no missing codes

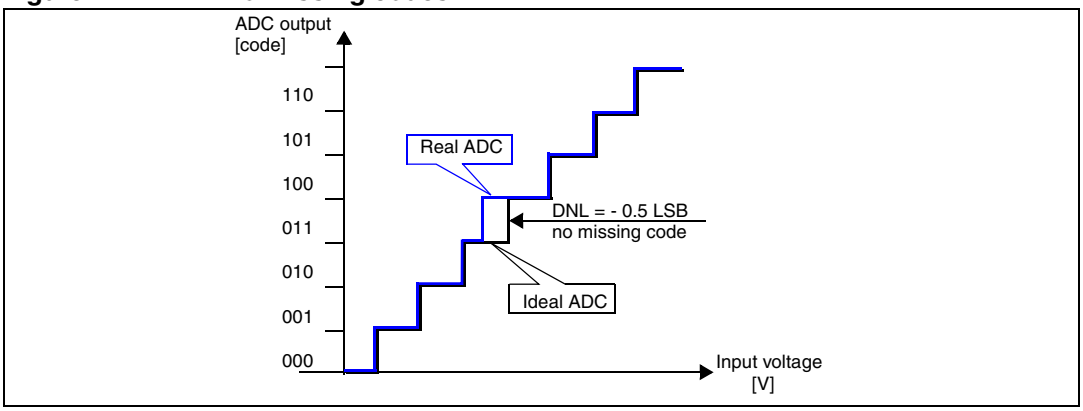
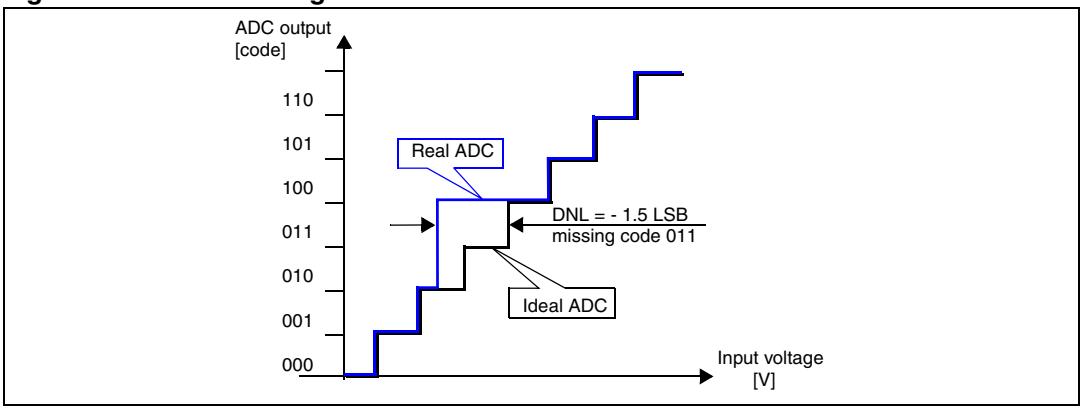


Figure 8. DNL: missing code



2.2.2 Integral nonlinearity

Integral nonlinearity (INL) is defined as the integral of the DNL errors. So, good INL guarantees good DNL. The INL error shows how far from the ideal transfer function value the measured converter result is. For example, an INL error of +/-2LSB in a 10-bit system means the maximum nonlinearity error may be off by 2/1024 or 0.2%. Note that neither INL nor DNL errors can be calibrated or corrected easily.

2.3 Offset error

Offset and gain errors can easily be calibrated by the application firmware. First, apply zero volts to the ADC input and perform a conversion, then the conversion result represents the zero offset error. Then perform a gain adjustment. A subsequent offset error calibration may be required. A useful method for offset and gain calibration is the least square method (which calculates the smallest error in all the used range).

2.4 Gain error

Gain error is defined as the full-scale error minus the offset error. Full-scale error is measured at the last ADC transition on the transfer-function curve and compared to the ideal ADC transfer function.

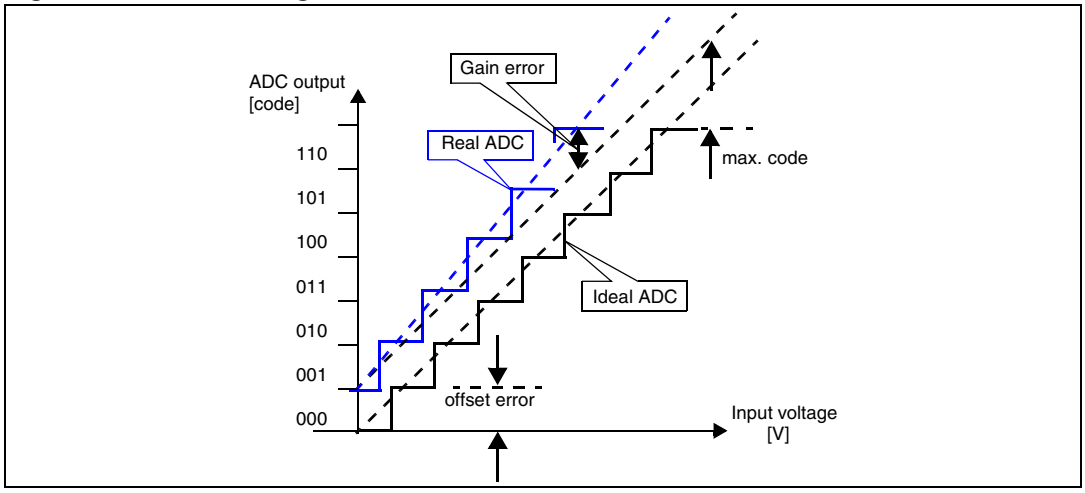
Gain error is easily corrected in firmware with this linear function:

$$y = (k1/k2).x$$

where k1 is the slope of the ideal transfer function and k2 is the slope of the measured transfer function.

Offset error and gain error can decrease dynamic range. For example this can be observed, if a full-scale input voltage is applied and the code obtained is 1010 instead of the ideal 1023 (for a 10-bit converter), or if the full-scale code 1023 appears with an input voltage less than full-scale.

Figure 9. Offset and gain error



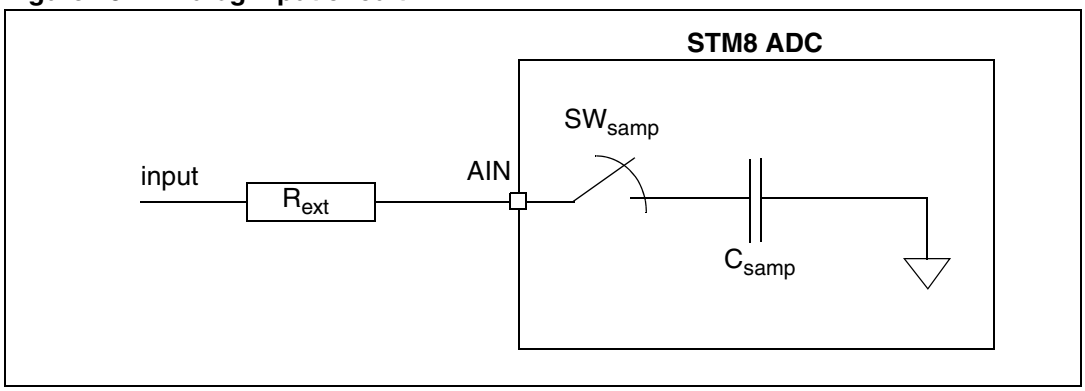
2.5 Hardware design errors

2.5.1 External resistance design error

The input multiplexer has nonzero impedance (R_{mpx} = max. 1kΩ). Sampling is done by switch which has also nonzero impedance (R_{sw} = max. 30kΩ). Both these impedances, together with the sampling capacitance (C_{samp} = max. 3pF) and external signal source resistance (R_{ext}), create a low pass filter. Therefore the external signal source impedance (R_{ext}) must be designed so that this low pass filter cut-off frequency will be not too low in relation to the ADC sampling time (Ts = 3 ADC clocks).

$$(R_{sw} + R_{mpx} + R_{ext}) \cdot C_{samp} \ll T_s$$

Figure 10. Analog input circuit



If the ADC converter has 10-bit precision (n=10) then the maximum precision is 1/2 LSB level (0.5/1024 from full range). So the acceptable error caused by charging C_{samp} through all resistors is 0.5/1024 (during 3 cycles of f_{ADC} clock). Then the maximum resistor value $R_{max}=R_{ext}+R_{sw}+R_{mpx}$ is obtained from following equation:

$$error = \frac{(U_c - U_i)}{U_i} = e^{-\frac{\frac{3}{f_{ADC}}}{R_{max} \cdot C_{samp}}} = \frac{1/2}{n}$$

$$R_{max} = -\frac{3}{f_{ADC} \cdot C_{samp} \cdot \ln\left(\frac{1/2}{n}\right)}$$

for worst case: f_{ADC} = 2 MHz and required 10-bit resolution (n = 10) the maximum serial resistor is:

$$R_{max} = -\frac{3}{(2 \cdot 10^6) \cdot (3 \cdot 10^{-12}) \cdot \ln\left(\frac{1/2}{1024}\right)} \cong 66k\Omega$$

and the maximum external resistor is:

$$R_{ext} = R_{max} - R_{sw} - R_{mpx} = 66 - 30 - 1 = 35k\Omega$$

The external parasitic PCB and package capacitance must also be taken into account. Therefore the maximum resistance of external signal source R_{ext} is lower, and in practice R_{ext} < 20 kΩ is suggested.

2.5.2 Reference voltage source

One of the biggest potential sources of errors in an ADC is the reference voltage. It is important to look at three reference voltage specifications: temperature drift, voltage noise, and load regulation (input resistance of reference input on STM8 ADC converter is >60 kΩ).

2.5.3 Temperature influence

Temperature has a big effect on ADC precision. Mainly two specifications are important: offset drift and gain drift. Those errors can be compensated in microcontroller firmware. One method is to fully characterize the offset and gain drift and provide a lookup table in memory to correct measurement due to temperature change. The ADC in each MCU device must then be compensated individually - therefore this calibration takes additional cost and time. The second method is to recalibrate the ADC when temperature change reaches given values. The effect of temperature on ADC accuracy is usually not specified seperately for microcontrollers because it is lower than other effects. It is included in the values for the other error types.

2.5.4 AC performance

Usually ADC performance is good only if input signals are from DC level up to Nyquist frequency. This is because an ADC design contains not only resistors but also capacitors (and parasitic RLC elements). Therefore frequency response degrades ADC performance if frequency increases. In case of AC measurement, the ADC frequency parameters must be studied; mainly signal-to-noise ratio (SNR), total harmonic distortion (THD) and spurious-free dynamic range (SFDR). Additional parameters can be signal-to-noise and distortion ratio (SINAD) and effective number of bits (ENOB).

3 Methods for precision improvement

3.1 Introduction

ADC converter precision is fixed by its principle, design and implementation but can be also improved by several hardware and/or software methods. Hardware methods improve ADC results by changing the environment around the ADC (PCB design, RC filters, ...) to obtain more accurate ADC output. Software methods try to improve the raw ADC results by postprocessing methods. Some CPU power is required to perform these methods, which can be implemented in the microcontroller firmware.

3.2 Hardware methods

- Analog zooming (use appropriate V_{REF} voltage and V_{REF} offset):
  - select reference voltage between input signal ranges
  - gives full ADC range - min. voltage per bit
- White noise added to measured signal:
  - wobbling of input signal over several bits gives the opportunity to use averaging (if input signal is very stable)
  - white noise gives independence from sampling frequency
- Good hardware design:
  - grounding
  - reference voltage filtering
  - supply filtering
  - preamplifier usage
  - frequency independence
  - …

3.2.1 Analog zooming

Analog zooming is method which improves precision by proper selection of the reference voltage. The reference voltage is selected in the expected range of the signal to be measured.

If the measured signal has an offset, then the reference voltage should also have a similar offset. If the measured signal has defined maximum amplitude, then the reference voltage should also have a similar maximum value. By matching this reference voltage to the measurement signal range we obtain the maximum possible resolution using the full A/D converter output range.

In the STM8 ADC the reference voltage is connected to external pins VREF+ and VREF-. This makes it possible to match the reference voltage to the measured signal range.

To VREF- pin you can connect the minimum measured voltage and to VREF+ pin you connect the maximum measured voltage. Take care to respect the minimum and maximum allowed reference signal levels - these values are specified in the datasheet for the related STM8 device type.

3.2.2 Adding white noise

This method combines hardware and software techniques to improve precision. From a software point of view, this method uses averaging and from a hardware point of view, it uses signal modification/spreading.

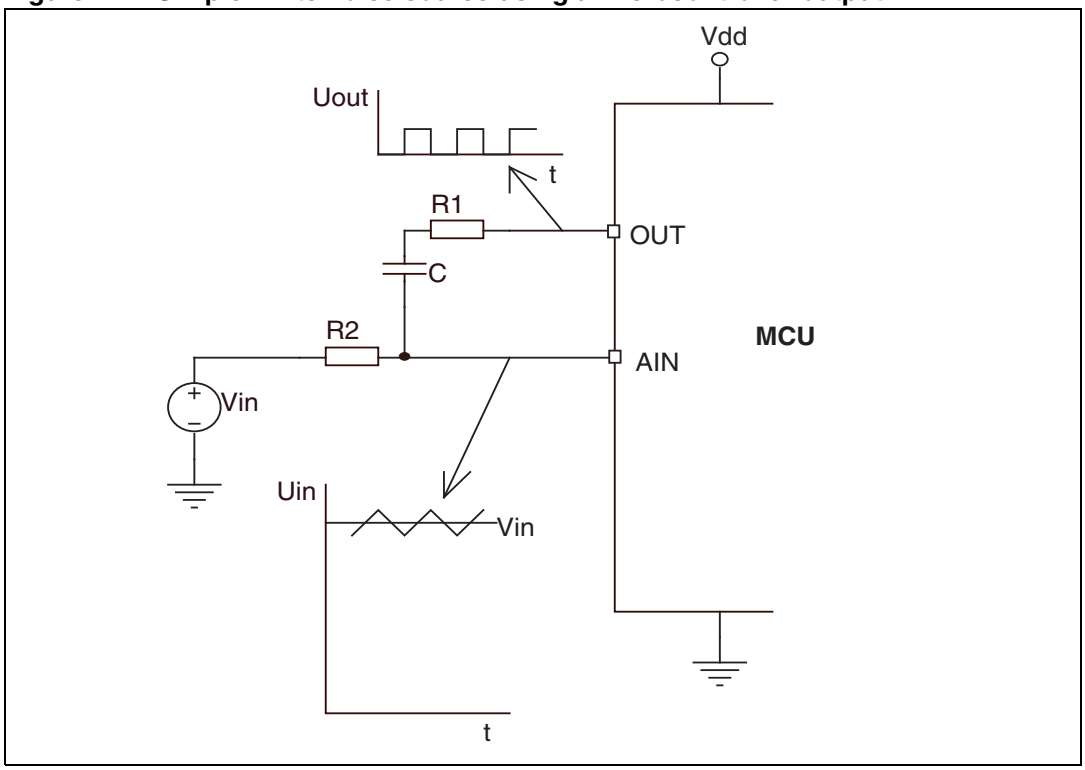
Averaging can be used in cases where the input signal is noisy (some signal change is necessary in order to be able to calculate an average) and the requirement is to obtain the mean value of a signal. A problem can appear when input signal is a very stable voltage without noise. In this case, when the input signal is measured, each sample is the same. This is because the input signal level is somewhere between two ADC word levels (e.g. between 0x14A and 0x14B). Therefore it is not possible to determine the input voltage level more precisely (e.g. if the level is near to 0x14A or near to 0x14B level). The solution is to add noise to the input signal which pushes its level across 1-bit ADC level (so that the signal level changes below 0x14A and above 0x14B level). This causes the ADC results to vary. Applying software averaging to the different ADC results, produces the mean value of the original input signal. This method is also called oversampling and/or dithering.

Adding a white noise signal gives 0.5LSB more each time you double the number of samples. By adding triangle noise, you get +1LSB, which is more efficient. The noise source can be the microcontroller itself.

This can be implemented using a noise generator with RC coupling of this noise to the input signal. Care must be taken to not modify the mean value of the original input signal - so capacitive coupling is used.

A very simple implementation is to design the white noise source as a square (or triangle) source which is generated directly by the STM8 microcontroller (*Figure 11*).

Figure 11. Simple white noise source using a microcontroller output



3.2.3 Hardware design rules

Good hardware design can remove or minimize many external influences on the measured signal and measuring device. Therefore good design is also a method of improving precision, or rather a method of minimizing additional errors caused by the external hardware. The following lists the main rules for minimizing external influences.

Grounding:
Care must be taken in the design of the PCB grounding. Ground paths on a PCB have nonzero resistance (and impedance) therefore current flowing through these paths generates a voltage drop. This voltage drop changes the voltages at points that are important for the ADC. For example it changes the reference voltage on the microcontroller pins slightly or influences the input signal on the microcontroller analog input pins. Good design minimizes this voltage drop. Methods of doing this are: minimizing the ground path impedance and minimizing the current through important ground paths (topology).
To minimize ground impedance: use massive ground paths. To minimize current through sensitive connections: use star topology, use independent PCB lines from input signal pins to microcontroller analog inputs, use independent PCB lines from the reference voltage source to microcontroller reference voltage inputs. Simply: supply current should not flow through signal ground - use signal ground and power ground in design.

Voltage filtering:
To minimize any external influences on the precise voltage levels used by the ADC, it is good to use filters on these voltage sources. Good filters must be applied on the reference voltage and on the supply voltage. Filter design depends on the noise amplitude present on the voltage sources, the external noise disturbance and the required precision.

Preamplifier usage:
If the measured signal is too small (in comparison to the ADC range) then it is good to use external preamplifier. This amplifier can fit the input signal range to the ADC range and can also insert offsets between input signal and ADC input. Care must be taken in the preamplifier design in order not to generate additional errors (for example additional offset, amplifier gain stability, linearity, frequency response, ...).

Frequency independence:
In cases where an AC signal is measured, frequency is an important characteristic of the input signal path. When measuring fast changes in the input signal then the frequency response of the signal path must be flat, even at high frequencies. The parasitic resistance and parasitic capacitance on PCB signal paths must be minimized. Otherwise, when measuring DC signal levels, adding an RC filter to the input signal data path can improve robustness against external noise.

3.3 Software methods

- Averaging samples:
  - averaging decreases speed but can give improved accuracy
- Digital filtering (50/60 Hz suppression from DC value)
  - set proper sampling frequency (the trigger from Timer1 is useful in this case)
  - perform software post-processing on sampled data (comb filter)
- Fast Fourier Transform (FFT) for AC measurements:
  - to show harmonic parts in measured signal
  - slower, due to the use of more computation power
- Calibration of ADC: offset, gain, bit weight calibration
  - decreases internal ADC errors
  - internal ADC structure must be known

3.3.1 Averaging samples

The principle of this method is to increase ADC precision but decrease ADC conversion speed. If the measured analog signal produces unstable ADC values, then the mean value of the given input signal can be obtained by averaging a set of values. Variation can be caused by signal noise or noise generated by the microcontroller itself (high speed digital signals capacitively coupled to the analog input signal).

Averaging is performed by choosing an appropriate number of samples to be averaged. This number depends on the required precision, minimum conversion speed and the level of other ADC errorsl (if another error has a greater influence on ADC precision, then increasing the number of averaging values has no effect on total measurement precision).

The advantage of this method is improving precision without any hardware changes. The disadvantages are lower conversion speed and therefore also lower frequency response (it is equivalent to decreasing effective sampling frequency). Another disadvantage is the need for an analog anti-aliasing filter.

3.3.2 Digital signal filtering

This is modern method which uses digital signal processing techniques. In principle, averaging is also a simple digital filter with a specific frequency response. But if the noise frequency spectrum is known, a digital filter can be designed which minimizes noise influence and maximizes ADC frequency response. For example, if the noise in the measured signal is coming from the 50 Hz power lines, then an appropriate digital filter can mainly suppress the 50 Hz frequency and deliver a precise 10 Hz input signal response.

The disadvantage of this method is that it needs sufficient microcontroller processing power and resources: CPU speed, RAM/ROM memory usage.

3.3.3 FFT for AC measurement

In some specific cases the application needs to know the amplitude of an AC signal with a given frequency. In this case the effective value of an AC signal can also be obtained by using a relatively slow sampling speed (in comparison to the measured signal frequency). For example, when measuring an AC mains signal (which is near-to-sinusoidal and has relatively low harmonics content), it is sufficient to choose a sampling frequency 32 times greater than the mains frequency (50 Hz). In this case you can obtain harmonics up to the

15th order. The amplitude of 15th harmonics in the main signal is very small (the next order harmonics can be neglected). The calculated effective value of the mains signal can be obtained with high precision because the effective values of harmonics are added to the total AC harmonic value as:

$$U_{ef} = \sqrt{U_1^2 + U_2^2 + \ldots + U_n^2}$$

So if the 15th harmonics amplitude is only 1% (0.01) from 1st harmonics (50 Hz) then its contribution to the total effective value will be only 0.01% (because of above equation - square addition: $0.01^2 = 0.0001$).

The principle of this method is therefore to sample the AC signal with a known frequency and then perform FFT postprocessing on the data of each measured period. Because the number of sampling points per measured signal period is small (32 points for example) then the performance needed for FFT processing is not so high (only 32-point FFT for example). If there is no requirement for real-time processing, for example, with a stable input signal shape, and measuring only one period per second, as in the case of the mains signal, then FFT can be calculated even by an 8-bit microcontroller.

Advantages: this method is good for AC measurement of a stable input signal. The disadvantage is the requirement for precise signal sampling. The frequency of the measured signal must be known and the ADC sampling frequency must be set exactly as a 2^n multiplier of the measured frequency. The input signal frequency can be measured by another method. The ADC sampling frequency can be tuned by programming the prescaler and MCU master clock or interpolation can used to insert sample points at the required frequency if sampling is performed with an inaccurate clock.

3.3.4 ADC calibration

This method requires knowledge of internal ADC structure and how the ADC converter is implemented inside the microcontroller. This knowledge is necessary in order to design a physical/mathematical model of the ADC implementation. In case of an unknown ADC implementation we can use standard models - linear or polynomic.

A proper physical model (which is usually a schematic diagram) can be used as the base for describing it mathematically. From the mathematical model each element in the model can be obtained by set of equations (for example, resistor values which represent bit weights). To solve these equations is necessary to perform a set of practical measurements and obtain a set of solvable equations (for example, measurement: input signal versus the proper ADC digital output words).

From the measured values and mathematical computation of the model, all known values of model elements (resistors, voltages, capacitors,...) can be put into the schematic diagram.

So instead of the ADC converter schematic with the designed values you obtain an ADC converter schematic with the real values for a given microcontroller.

Computed model parameters can be stored in the microcontroller memory after calibration and used in postprocessing to correct ADC values.

4 Design rules for minimizing errors

ADC precision can only be significantly improved if external influences on the measurement are minimized. Therefore any precision improvement method also includes rules for correct design.

Main design rules for minimizing design errors:

- Grounding of analog/digital power:
  - star topology
- V_{DDA}, V_{SSA} filtering:
  - RC, LC filtering
  - avoid noise from noisy digital power
- V_{REF} selection - offset, value, precision:
  - reference voltage source precision and stability matched to application precision requirements and ADC capability
  - reference voltage source value and precision in line with the expected measured range
- Source impedance vs. input impedance knowledge:
  - use input buffers for measured signal (if source impedance is too high)
  - impedance in relation with required conversion speed
- External preamplifier usage:
  - for low (and also high) level signals
  - amplifier speed and precision properties
  - amplifier dependency on frequency
- Select appropriate ADC mode, speed, trigger
- Software methods: averaging, FFT, …

5 Conclusion

The Analog to Digital Converter (ADC) in the STM8 microcontroller family is a standard ADC converter which can have common errors. To minimize errors and improve precision, the main methods and application design rules have been described. You can apply these methods to your application.

The choice of method depends on the application requirements and is always a compromise between speed/precision, enough computation power and design topology. Published methods lead to precision improvement but strongly depend on the ADC basic properties.

6 Appendix - source code examples

6.1 Project code example

A firmware package is linked with this document. It contains source codes in C-language which give examples of some of the precision improvement methods described in this application note. This complete project is designed for the *ST Visual Develop* (*STVD*) development environment and *Cosmic* compiler and using the *STM8 evaluation board*. With these examples you can see how the method is implemented in practice and you can test it by comparing ADC precision with the improvement method and without it.

Package name: *STM8_ADC_improvement.zip*
Project name: *STM8_ADC_improvement.stw*

6.2 Source code description

The example consists of a complete STVD project. This project contains source codes in files:

main.c	main program file (MCU initialization, main program loop)
ADCAveraging.c	testing - each file for given method (uses STM8 library)
ADCCalibration.c	
ADCdriverFFT.c	
ADCfncFFT.c	
ADCWhiteNoise.c	
FFT.c	
Filter50Hz.c	
mono_lcd.c	driver for writing to LCD display (on STM8 evaluation board)
stm8_interrupt_vector.c	contains interrupt table address definition
ADCAveraging.h	headers for ADC precision improvements methods
ADCCalibration.h	
ADCdriverFFT.h	
ADCfncFFT.h	
ADCWhiteNoise.h	
FFT.h	
Filter50Hz.h	
globaldefFFT.h	
mono_lcd.h	list of LCD functions
stm8_conf.h	configuration of STM8 library

6.2.1 Program flow

In the *main.c* file is the main program loop - in function *main()*. At the beginning is the basic MCU initialization: GPIO pins (LED diodes, buttons), CPU clock, interrupts. Then the main testing loop begins - it consists of the ADC initialization and series of steps (methods).

Each test step contains a write to the LCD display to display information about the running test and then contains the test routine itself. The test routines are provided in listed files - each file for each method. To continue from one test to the next, the user must press a button (joystick select button on evaluation board). After all tests are done, then the ADC is reconfigured to higher speed and the test loop runs again.

Tests are:

- *TestADCAveraging();*
  - ADC in single mode
  - averaging method applied
- TestADCWhiteNoise();
  - ADC in continuous mode
  - averaging method applied
- TestADCDigitalFilter50Hz();
  - ADC in single mode
  - external timer trigger
  - simple comb 50 Hz filter applied
- TestADCFFT();
  - ADC in single mode
  - external timer trigger
  - 16-point FFT applied
- TestADCCalibration();
  - ADC in single mode
  - software start
  - simple linear model applied (offset and gain calibrated)

6.2.2 Hardware and software requirements

To run the project example you must have installed *STVD* (min. version 3.5.0) and *Cosmic* compiler for STM8 family. Both tools are free for download from STMicroelectronics and Cosmic websites.

The hardware requirement is to have *STM8 evaluation board* with STM8S - 128K chip installed. Instead of a real STM8 chip you can use *STice emulator*.

7 Revision history

Table 1. Document revision history

Date	Revision	Changes
27-May-2008	1	Initial release.
28-Oct-2008	2	Updated *Figure 2*, *Figure 4*, *Figure 5* and *Figure 6*.