Metrol. Meas. Syst., Vol. 25 (2018) No. 4, pp. 675–687.

METROLOGY AND MEASUREMENT SYSTEMS

Index 330930, ISSN 0860-8229

www.metrology.pg.gda.pl

A RANDOM SIGNAL GENERATION METHOD FOR MICROCONTROLLERS WITH DACs

Zbigniew Czaja, Michał Kowalewski

Gdańsk University of Technology, Faculty of Electronics, Telecommunications and Informatics, G. Narutowicza 11/12, 80-233 Gdańsk, Poland (✉ *zbczaja@pg.edu.pl*, +48 58 347 1487, *michal.kowalewski@eti.pg.edu.pl*)

Abstract

A new method of noise generation based on software implementation of a 7-bit LFSR based on a common polynomial PRBS7 using microcontrollers equipped with internal ADCs and DACs and a microcontroller noise generator structure are proposed in the paper. Two software applications implementing the method: written in ANSI C and based on the LUT technique and written in AVR Assembler are also proposed. In the method the ADC results are used to reseed the LFSR after its each full work cycle, what improves randomness of generated data, which results in a greater similarity of the generated random signal to white noise, what was confirmed by the results of experimental research. The noise generator uses only the internal devices of the microcontroller, hence the proposed solution does not introduce hardware redundancy to the system.

Keywords: noise generators, LFSR, microcontrollers, ADC, DAC.

1. Introduction

At present, to generate a random signal in mixed-signal electronic systems, random number generators are used. In practice, pseudo-random number generators are used due to their simple construction and easy hardware implementation. Especially, well known *Linear Feedback Shift Registers* (LFSRs) [1, 2] are easily implemented in hardware [3–5] as well as in software [6–9].

LFSRs are used not only in cryptography but also in circuit testing for test-pattern generation and signature analysis, especially in built-in self-test techniques and in digital broadcasting systems. Because LFSRs enable a very fast generation of pseudo-random sequences, they can be applied to generating an approximation of white noise used *e.g.* in sound generators and in the measurement equipment for digital transmissions.

Often the mixed-signal electronic systems are controlled by microcontrollers. Therefore, to build the noise generators we can use microcontrollers already present in these systems, and especially their analogue peripheral devices, as it was done for self-testing of analogue parts of such systems [10, 11] and for direct sensor-to-microcontroller interface circuits [12–16]. Thanks to this, we avoid introducing hardware redundancy into the system. Hence, in the paper we propose a new method of random signal generation based on software implementation of LFSR in a microcontroller equipped with internal *Analog-to-Digital Converter* (ADC) and *Digital-to-Analog*

Article history: received on Apr. 24, 2018; accepted on Aug. 14, 2018; available online on Dec. 31, 2018, DOI: 10.24425/mms.2018.124880.

Converter (DAC) circuits. The ADC is used to reseed the LFSR (similarly to that presented in the paper [9]), whereas the random signal is generated by the DAC (Fig. 1).

The method is presented on an example of an 8-bit ATXmega32A4 microcontroller [17], which is equipped with numerous well-working measurement peripheral devices, *e.g.*: one 12-bit 4-channel ADC and one 12-bit 2-channel DAC. The structure of LFSR is based on a common polynomial PRBS7, *e.g.* used in telecommunication for testing of serial data links based on an 8b/10b data encoding scheme [18], which has the form [19]:

$$L(x) = x^7 + x^6 + 1, \tag{1}$$

where the exponents in (1) define the positions of feedback taps of the 7-bit LFSR (see Fig. 6). Thus, the outputs from stages 7 and 6 are modulo-2 added and fed back to the input of the first stage of the register to obtain a $2^7 - 1 = 127$ long sequence.

2. Operating principles

In Fig. 1 we can see the hardware structure of the random signal generator, consisting only of a microcontroller, what is an advantage of the proposed approach. The ADC of the microcontroller serves as the source of entropy that provides real random values used as the reseeding value of the LFSR. The ADC samples the voltage at floating pins ADC0, ldots, ADCn, thus it measures noise; more precisely – interferences induced on these pins by inter alia working digital blocks of the microcontroller. We use a low voltage reference source with a simple structure (a 1 V internal voltage reference [17] stabilized form a supply voltage $V_{cc} = 3.3$ V) to increase the measurement sensitivity of the ADC, which increases the susceptibility of the ADC to noise, what is generally not recommended, but in our case is preferred. As the reseeding value of the 7-bit LFSR (we base on a polynomial PRBS7) seven lower bits of the 12-bit ADC result are taken. The actual random data from the ADC are used to reseed the LFSR after its each full work cycle consisting of 127 steps.

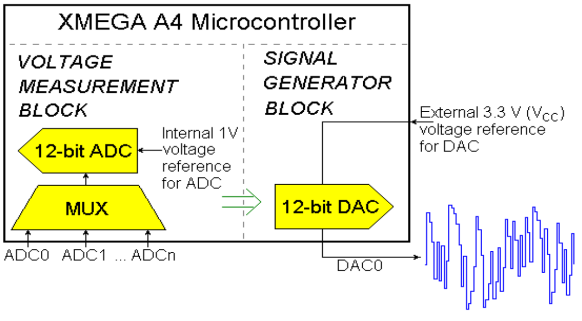

Fig. 1. The hardware structure of the random signal generator based on a microcontroller equipped With internal ADC and DAC circuits.

This solution results from hardware limitations of peripheral devices of an ATXmega32A4 microcontroller [17]. The minimum conversion time for 12-bit resolution of the ADC takes 14 μs, whereas the maximum conversion rate of the DAC is equal to 1 million samples per second. Therefore, the minimum DAC conversion time (1 μs) is at least 14 times less than that of the ADC (we also have to take into account the time needed for servicing the ADC by the software).

Because the DAC works at the maximum speed to generate the best possible approximation of white noise, the 7-bit LFSR cycle takes 127 μs. Thus, the ADC has enough time to measure a new reseed value.

3. Entropy source

As it was mentioned, the ADC is used to generate the actual random data by sampling floating pins ADC0, .. ADCn, *i.e.* essentially by measuring noise.

However, we should remember that there is a risk that for more random data they can become predictable, that is they can degrade to pseudo-random data if not enough entropy is available.

It was tested experimentally. A set of 1024 voltage values sampled every 127 μs with 12-bit resolution at the floating ADC1 input were measured by the ADC (with the internal 1 V reference and the signed-ended positive input channel mode) of the microcontroller and next sent via the UART interface (via the UART/USB converter module) to the *personal computer* (PC). The results are drawn for full 12-bit resolution in Fig. 2a and Fig. 3a, for 7 least significant

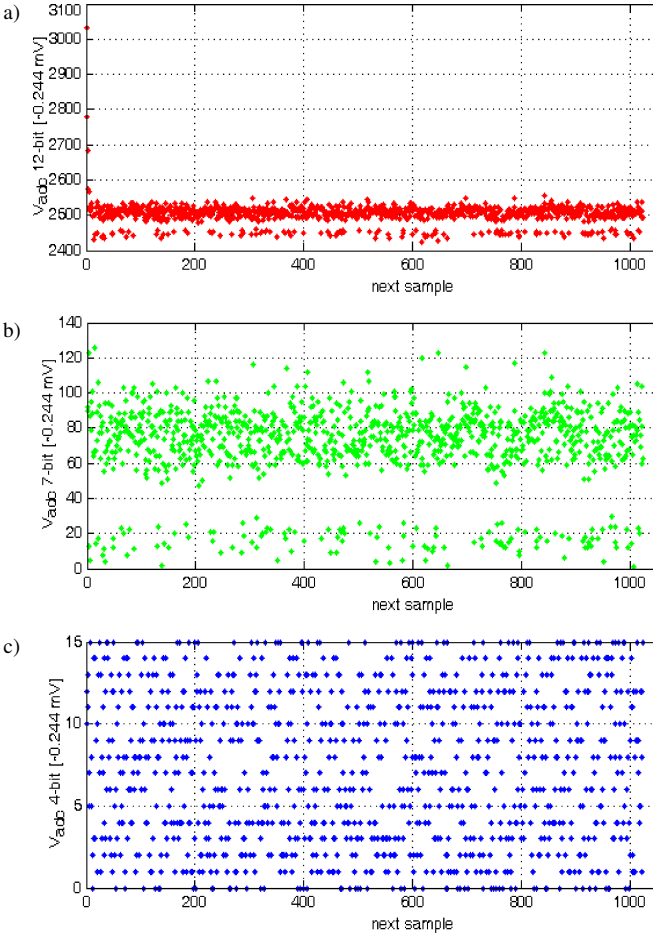

Fig. 2. ADC measurement results: a) for 12-bit resolution; b) for 7 LSBs; c) for 4 LSBs of the measurement results.

bits (LSBs) in Fig. 2b and Fig. 3b, and also for 4 LSBs of the measurement results in Fig. 2c and Fig. 3c.

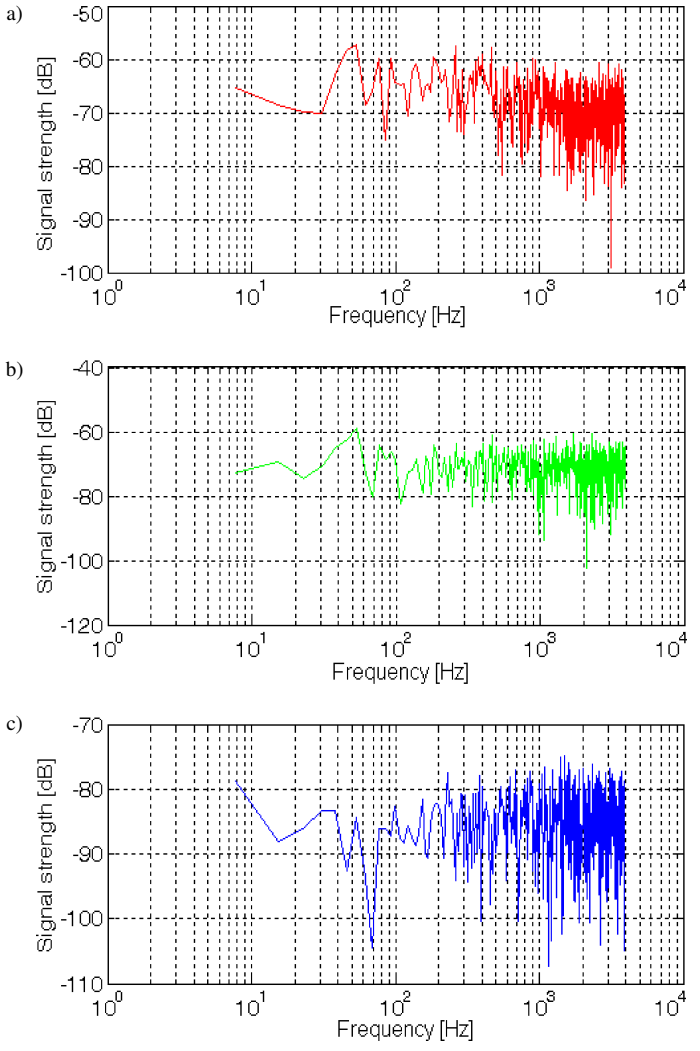

Fig. 3. Spectra for ADC measurement results: a) for 12-bit resolution; b) for 7 LSBs; c) for 4 LSBs of the measurement results.

Histograms of these results are presented in Fig. 4. It is seen in Fig. 2, Fig. 3 and Fig. 4 that the measurement data have a specific distribution depending on the operating conditions and configuration of a given system during measurements. In other words, it is the distribution of interferences at a chosen pin coming from inter alia digital blocks of the microcontroller – *e.g.* a system clock and a core microprocessor – working during measurements. Thus, we can say that they statistically characterize the actual status of the system. This is especially visible for the 12-bit resolution of the ADC (Fig. 2a, Fig. 3a and Fig. 4a). However, it is seen that distributions for 7 LSBs of the measurement results (Fig. 4b), and especially for 4 LSBs of the measurement results (Fig. 4c) are approaching the distribution of white noise.

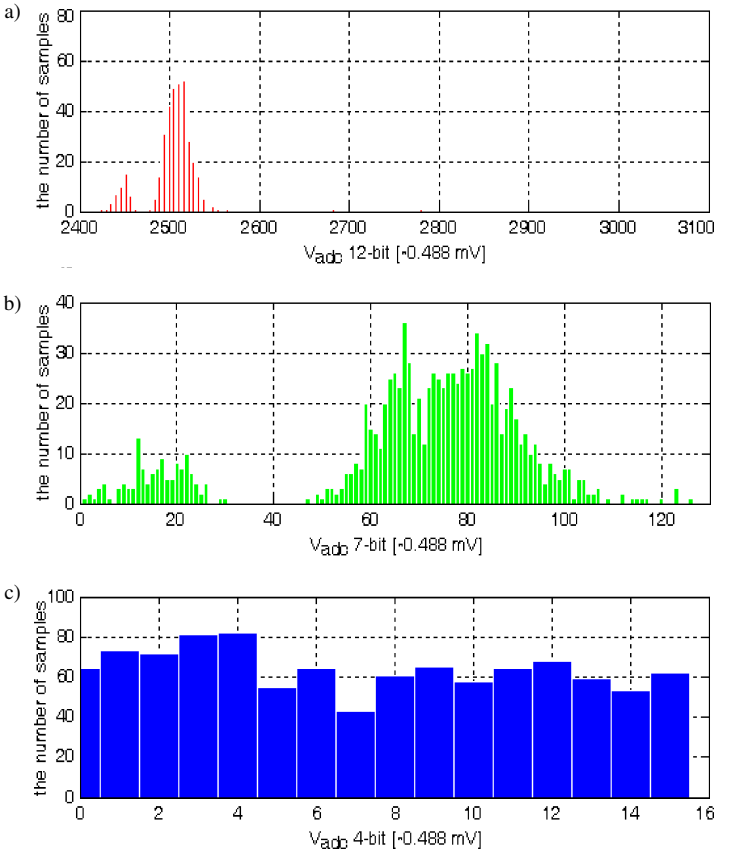

Fig. 4. Histograms for ADC measurement results: a) for 12-bit resolution; b) for 7 LSBs; c) for 4 LSBs of the measurement results.

For the proposed application of noise generator the reading of 7 LSBs of the result was chosen, because it is simple (only the 8-bit ADC result low register *ADCA_CH0RESL* [17] is read and the most significant byte (MSB) of this result is cleared) and takes less than 1 μs, what is very important. It follows from the fact that the reseeding should take exactly 1 μs – the converting time of a single DAC sample, to not distort the generated waveform (harmonics with an amplitude greater than the others do not enter). It should be mentioned that such randomness is sufficient to generate an approximation of white noise.

However, for the needs of cryptography 4 LSBs of n independent measurement results should be used to create, more precisely, to assemble the seed, where $n = \textit{the size of the seed in bits} / 4$, what guarantees true randomness.

4. Software implementation

To fully utilize the capabilities of the DAC, it has to generate a new voltage value at the DAC0 pin every 1 μs. Hence, calculating a new value of the LFSR and writing it to the DAC data register should take less than 1 μs. Because the microcontroller works with a 16 MHz oscillator

($t_{clk} = 0.625\ \mu$s), these operations must fit in the period taken by 16 instruction cycles of the core microprocessor of the microcontroller. This challenge was solved and presented below.

Two new solutions of software applications of the noise generator based on a 7-bit LFSR and a polynomial PRBS7 are proposed in the paper. The first one is based on the *look-up-table technique* (LUT) and is written in ANSI C, whereas the second one is written in AVR Assembler [17] and operates in accordance with the principle of operation of LFSR.

4.1. ANSI C application based on LUT technique

To simplify the algorithm of application written in ANSI C [20] and based on the LUT technique, whose flowchart is shown in Fig. 5, a table *v_table* of uint8_t variables keeps two copies of the set of all 127 7-bit values generated based on the polynomial PRBS7. Thus, it has a double size – but only $2 \cdot 127$ bytes. Additionally, an index i and a variable *end_i* are declared as register variables. Also, incrementing and resetting MSB of i are written in AVR Assembler.

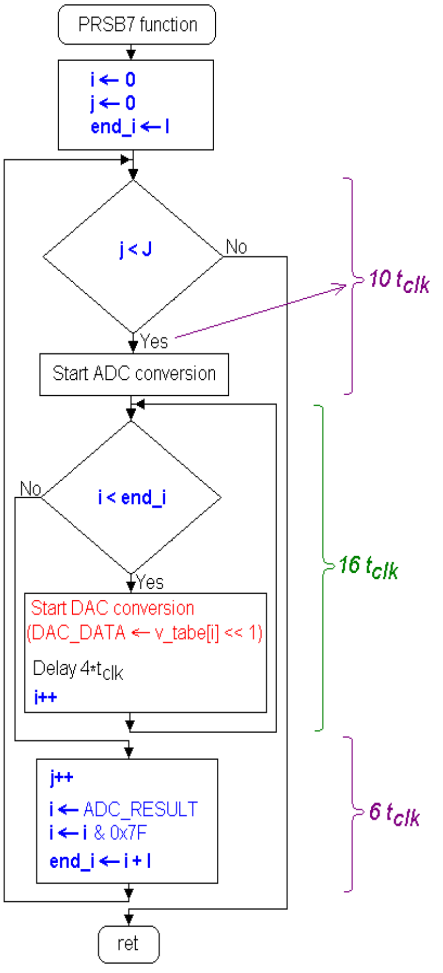

Fig. 5. A flowchart of the algorithm of the PRSB7 function based on the LUT technique.

A variable j determines the number of repetitions of the full 127 μs cycle, *i.e.* the duration of the generated waveform.

The algorithm consists of two loops:

– The external loop is responsible for starting the ADC conversion (the 12-bit result is right justified), incrementing the variable j and checking whether the signal generation should be terminated ($j < J$), and for reseeding the index i (writing the 7-bit ADC measurement result to i and updating the variable *end_i* so that the internal loop should be executed $I - 1 = 127$ times). The execution of the code that supports this loop takes 16 instruction cycles.

– The internal loop also takes 16 instruction cycles. 12 cycles are used to start the DAC conversion by writing to the DAC data register high *DACB_CH0DATAH* [17] a value from *v_table* indicated by index i, incrementing i and checking whether the full 127 μs cycle is finished. A 4 t_{clk} delay is added to ensure that the software triggers the DAC conversion every 1 μs.

4.2. Assembler code implementing LFSR

The idea of the second proposed application written in AVR Assembler [21, 22] is based on the principle of operation of a 7-bit LFSR based on a polynomial PRBS7 (1). Hence, a 7-bit LFSR is implemented in an 8-bit register called LFSR_reg (Fig. 6) (an r16 register was chosen).

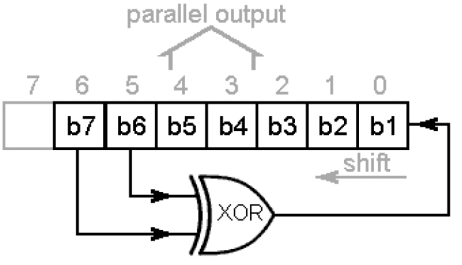

Fig. 6. Implementation of an LFSR based on a PRBS7 in an 8-bit register.

The algorithm of the assembler application is divided into fifth parts (Fig. 7):

– The first – initialisation – part executed only once is responsible for entering the initial values into i, j and *LFSR_reg* register variables and for the first start of the ADC conversion.

– The second part contains instructions that execute: storing a value from *LFSR_reg* in a temporary register *LFSR_temp_reg*, shifting left the contents of *LFSR_reg* and starting the DAC conversion by entering the contents of *LFSR_reg* to the register *DACB_CH0DATAH*, decrementing i and testing whether it is equal to 0, what means that the full 127 μs cycle is finished. The contents of *LFSR_reg* is shifted for two reasons. The first one follows from the fact that we care about using the full voltage range of the DAC, that is 3.3 V (the 12-bit result is left justified). The second one – it is the first operation executed according to the formula (1).

– In the third part the remaining operations of the formula (1) are executed.

– In the fourth part two functions are performed. The first one is updating the seed (the *ADC_reg* variable) and the second one is executing by software a 7 t_{clk} delay to synchronize it with the DAC conversion.

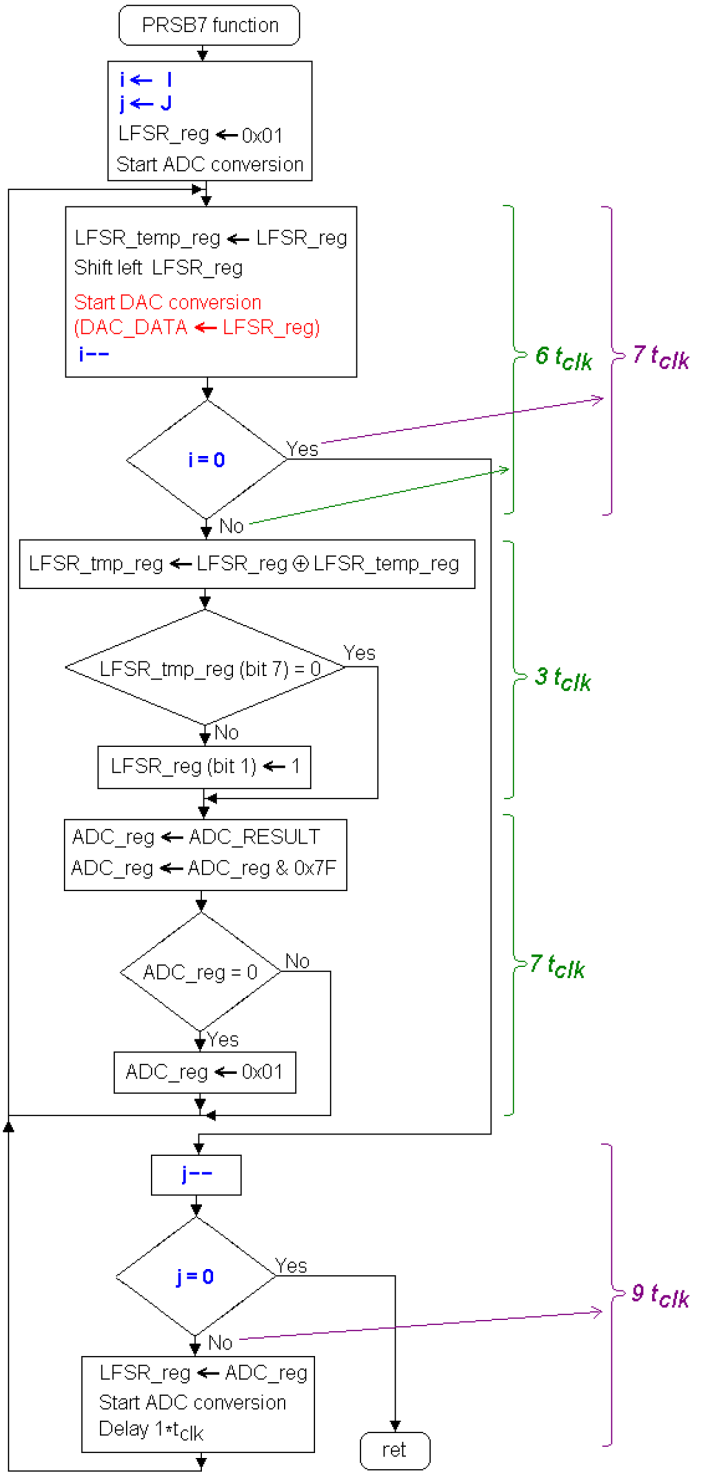

Fig. 7. A flowchart of the algorithm of the PRSB7 function written in AVR Assembler.

– The last (fifth) part is responsible for the control of the duration of the generated waveform (testing the j variable), for reseeding the *LFSR_reg* and starting the next ADC conversion.

In this case the algorithm is also composed of two loops. The internal one consists of the second, third and fourth parts of the algorithm and the external one is assembled from the second and fifth parts. Execution of both loops takes precisely 16 t_{clk}.

5. Experimental results and discussion

In this chapter the results of experimental research of the proposed noise generator methods are presented. A picture of the laboratory stand is shown in Fig. 8. The system consists of a microcontroller XMEGA A4 board (ATXmega32A4 working with a 16 MHz crystal oscillator and a supply voltage $V_{CC} = 3.3$ V), an Agilent U2531A Data Acquisition Board used to sample noise at the DAC output, a digital oscilloscope Agilent MSO-X 3014A used to view the random signal timing and a PC used to control the system and analyse the measurement data.

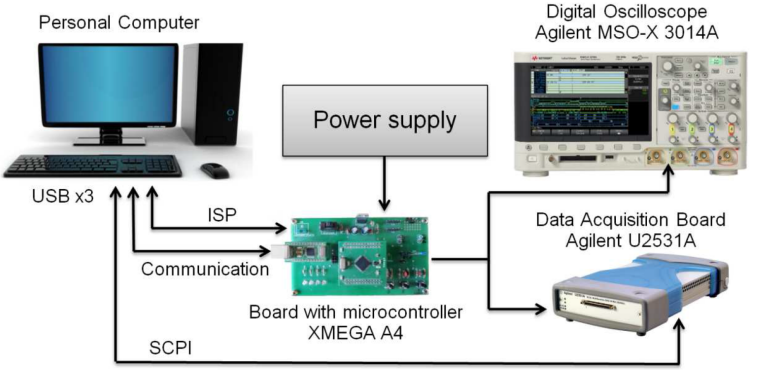

Fig. 8. A scheme of the laboratory stand.

The measurements were carried out for three methods of generating a white noise approximation based on a 7-bit LFSR and a polynomial PRBS7. The first method was used to generate a reference random signal. In this case the random signal was generated with a constant initial value (the constant value of the seed of the LFSR), as it is shown in Fig. 9. In relation to it, the proposed solutions of noise generation are compared and evaluated. The second method uses the ANSI C application based on the LUT technique, and the third one – the application written in AVR Assembler; both are based on the reseeding initial value (Fig. 10).

For each case the random signal was $N = 20000$ times sampled by the U2531A board [23] with 14-bit ADC resolution at a sampling frequency $f_s = 2$ MHz, an acquisition time 10 ms and a voltage reference $V_{ref} = 5$ V. Next, the set of measurement data $\boldsymbol{v}_N = \{v_n\}_{n=0,\ldots,N-1}$ was sent to the PC. The relative amplitude of an individual component V_k of the spectrum $\boldsymbol{V}_K = \{V_k\}_{k=0,\ldots,N-1}$ of the generated random signal was calculated from (2):

$$V_k = 20 \cdot \log_{10}\left(\frac{2}{N} \cdot \sum_{n=0}^{N-1} v_n \cdot e^{\frac{-i \cdot 2 \cdot \pi \cdot k \cdot n}{N}}\right). \tag{2}$$

The determined spectra of the white noise approximations for three generation methods are drawn in Fig. 11, Fig. 12 and Fig. 13, respectively.

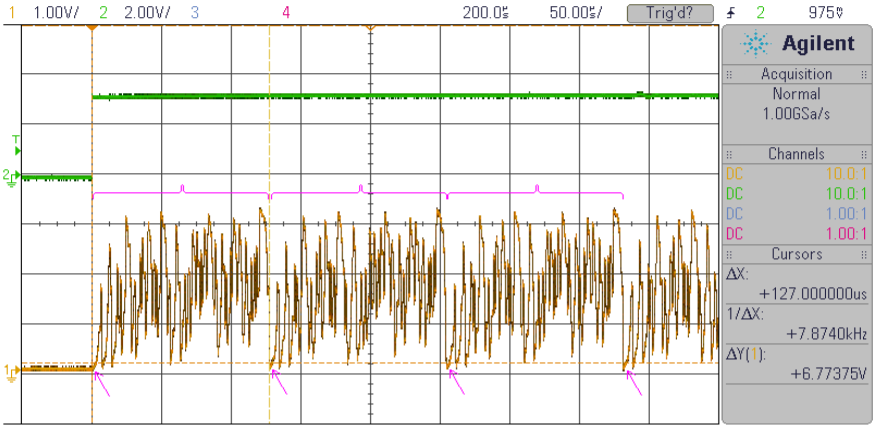

Fig. 9. The white noise approximation timing generated at the DAC output of an ATXmega32A4 microcontroller for a polynomial PRBS7 with a constant initial value.

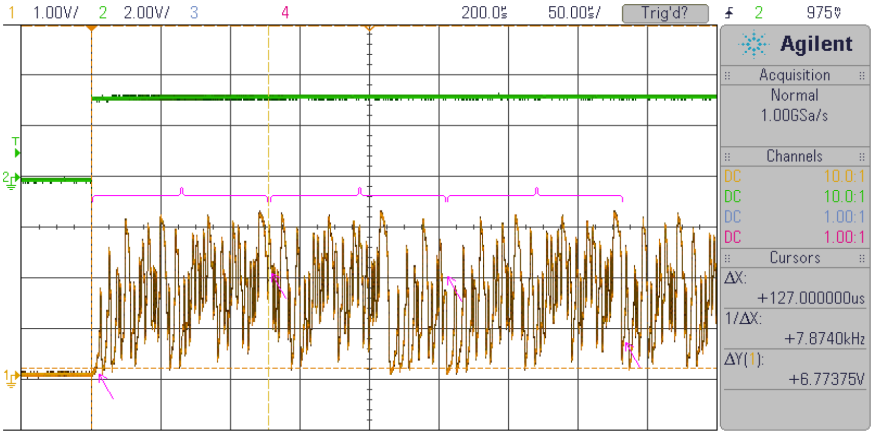

Fig. 10. The white noise approximation timing generated at the DAC output of an ATXmega32A4 microcontroller for a polynomial PRBS7 based on the reseeding initial value.

As it is shown in Fig. 9, the white noise approximation signal for a constant initial value of the LFSR has the character of a periodic signal. For this reason its spectrum differs significantly from the spectrum of white noise, as it is presented in Fig. 11. Among others, we can distinguish several large peaks in this spectrum.

The experimental research shows that thanks to using the reseeding of the LFSR our microcontroller noise generator generates a signal that is more similar to white noise than the one obtained in the case of a constant initial value, as shown in Fig. 12 and Fig. 13.

Comparing the spectra shown in Fig. 12 and in Fig. 13 it is seen that the second one is more similar to a white noise spectrum. Greater deviations from the white noise spectrum appearing on the graph in Fig. 12 result from an occasional slight lack of synchronization of the external loop software (Fig. 5) with the work of the DAC. The execution times of the external and internal loops (Fig. 5) of the application written in ANSI C were exactly determined based on the code

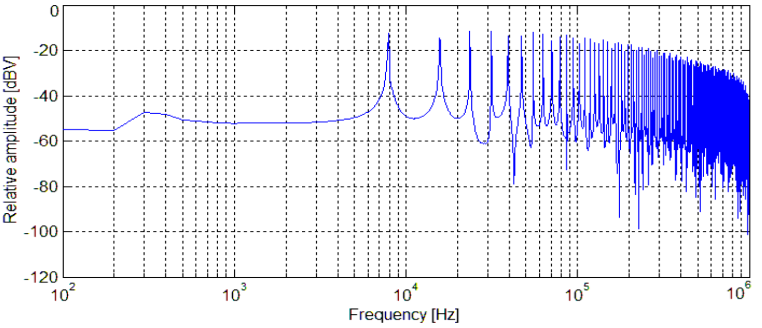

Fig. 11. The spectrum of a signal generated for a constant value of a seed of LFSR for a polynomial PRBS7.

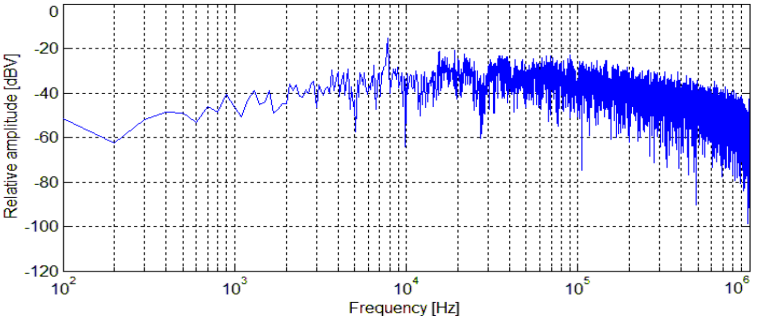

Fig. 12. The spectrum of a signal generated for reseed values of LFSR using the application written in ANSI C and based on the LUT technique.

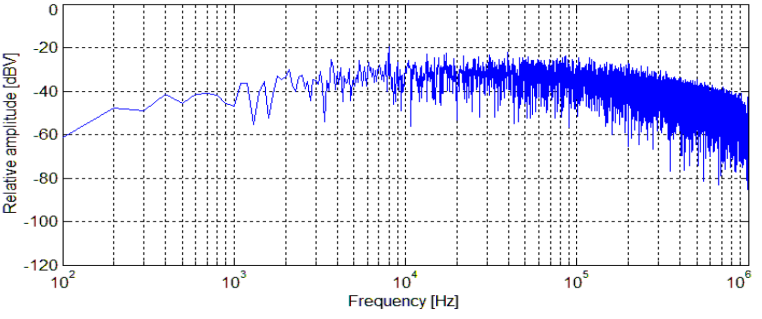

Fig. 13. The spectrum of a signal generated for reseed values of LFSR using the application written in AVR Assembler.

in the LSS file [20]. However, it is not possible to fully control the low-level behaviour of the code execution of a high level language until the end. For this reason the application in AVR Assembler was written.

Summarizing, the application written in ANSI C and based on the LUT technique is very simple and easy to implement in comparison with the application written in AVR Assembler. However, the second method enables to generate a better white noise approximation signal.

6. Conclusions

A new method of random signal generation based on software implementation of LFSRs for microcontrollers with internal ADCs and DACs and a microcontroller noise generator structure are proposed in the paper. Two software applications implementing the method are also proposed: written in ANSI C and based on the LUT technique and written in AVR Assembler. The proposed solution has the following features:

- the LFSR is based on a common polynomial PRBS7;
- the ADC result is used to reseed the LFSR after its each full work cycle, what improves randomness of generated data, which results in a greater similarity of the generated random signal to white noise;
- the random signal is generated by the DAC with its maximum conversion rate (1 MS/s);
- calculating a new value of the LFSR and writing it to the DAC data register takes only 16 instruction cycles of the core microprocessor of the microcontroller, which corresponds to 1 μs. This is one of significant advantages of the proposed method.

It should be emphasized that this has been attained without introducing hardware redundancy to the system, because we have employed only the internal devices of the microcontroller.

Further research in the field of random signal generation will be focused on increasing the degree of signal randomness through the use of several ADC channels and differentiation of the type of signal generated (not only white noise, but also red, pink, blue and violet ones).

References

[1] Saluja, K.K. (1987). *Linear feedback shift registers theory and applications*. Department of Electrical and Computer Engineering, University of Wisconsin-Madison, 4.

[2] Walczak, J., Stępień, R. (2012). Discrete Modeling of LFSR Registers. *Elektryka*, 2(222), 97–104.

[3] D'Alvano, F., Badra, R.E. (1996). A Simple Low-Cost Laboratory Hardware for Noise Generation. *IEEE Transactions on Education*, 39(2), 280–281.

[4] Mita, R., Palumbo, G., Pennisi, S.M., Poli, M. (2002). A Novel Pseudo Random Bit Generator for Cryptography Applications. *The 9th IEEE International Conference on Electronics, Circuits and Systems*, 489–492.

[5] Cypress Semiconductor Corporation. (2015). 8-Bit Pseudo Random Sequence Generator Datasheet, Document Number: 001-13579 Rev. *J.

[6] Mondal, S., Barman, A.D., Datta, A.K. (2012). ARM7 Microcontroller Based Digital PRBS Generator. *International Journal of Electrical, Electronics and Computer Engineering*, 1(2), 55–59.

[7] Walczak, J., Stępień, R. (2010). Microprocessor Based White Noise Generator. *Elektryka*, 2(214), 97–104.

[8] Babu, P., Soumya, S.S., Sudheesh, K., Sujeesh, K., Syamily, P.S. (2014). Design of a Microcontroller Based Random Number Generator. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, 3(2), 7614–7618.

[9] Fimml, P. (2013). HOWTO: A Simple Random Number Generator for the ATmega1280 Microcontroller. *https://ti.tuwien.ac.at/ecs/teaching/courses/mclu_2014/misc/task1-specific-stuff/rand_howto.pdf*.

[10] Czaja, Z. (2013). Self-Testing of Analog Parts Terminated by ADCs Based on Multiple Sampling of Time Responses. *IEEE Transactions on Instrumentation and Measurement*, (62), 3160–3167.

[11] Toczek, W., Czaja, Z. (2011). Diagnosis of fully differential circuits based on a fault dictionary implemented in the microcontroller systems. *Microelectronics Reliability*, 8(51), 1413–1421.

[12] Czaja, Z. (2016). An Implementation of a Compact Smart Resistive Sensor Based on a Microcontroller with an Internal ADC. *Metrol. Meas. Syst.*, 23(2), 255–238.

[13] Czaja, Z. (2012). A microcontroller system for measurement of three independent components in impedance sensors using a single square pulse. *Sensors and Actuators A*, (173), 284–292.

[14] Czaja, Z. (2018). Time-domain measurement methods for R, L and C sensors based on a versatile direct sensor-to-microcontroller interface circuit. *Sensors and Actuators A*, (274), 199–210.

[15] Jevtic, N., Vujo, Drndarevic, V. (2013). Design and implementation of plug-and-play analog resistance temperature sensor. *Metrol. Meas. Syst.*, 20(4), 565–580.

[16] Kokolanski, Z., Gavrovski, C., Dimcev, V., Makraduli, M. (2013). Hardware techniques for improving the calibration performance of direct resistive sensor-to-microcontroller interface. *Metrol. Meas. Syst.*, 20(4), 529–542.

[17] Microchip Technology Inc. (2017). 8-bit Atmel XMEGA AU Microcontroller, XMEGA AU MANUAL. *http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-8331-8-and-16-bit-AVR-Microcontroller-XMEGA-AU_Manual.pdf*.

[18] Tavacoli J. Silicon Driven Signal Integrity Tools. Altera (2005). *ftp://ftp.altera.com/outgoing/download/education/events/2005_highspeed_altera.pdf*

[19] Mutagi, R.N. (1996). Pseudo noise sequences for engineers. *Electronics & Communication Engineering Journal*, 79–87.

[20] Atmel Corporation. (2015). AVR Libc Reference Manual. [online] *https://www.microchip.com/webdoc/AVRLibcReferenceManual/index.html*.

[21] Atmel Corporation. (2016). AVR Assembler. [online] *https://www.microchip.com/webdoc/GUID-E06F3258-483F-4A7B-B1F8-69933E029363/index.html*.

[22] Atmel Corporation. (2016). AVR Instruction Set Manual. *http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-0856-AVR-Instruction-Set-Manual.pdf*.

[23] Keysight Technologies. (2017). U2500A Series USB Modular Simultaneous Sampling Multifunction DAQ Devices – Data Sheet. *https://literature.cdn.keysight.com/litweb/pdf/5991-0651EN.pdf?id=2205971*.