On random numbers in practice and their generating principles and methods

Mihály Tóth, György Györök
Alba Regia University Center
Óbuda University
Budai Str. 45, H-8000 Székesfehérvár
{toth.mihaly, gyorok.gyorgy}@arek.uni-obuda.hu

***Abstract*—Random numbers are indispensable necessities in many areas of science and technology ranging from Monte Carlo simulations, testing dynamical response of measuring transmitters to secure encryption methods and different security systems. A good number of such systems are applied in car security and physical accessibility of many secure entities from , admission systems to accessibility of databases. Though computer generated random numbers can be used for many applications, they remain basically non-random in the sense that anything that generated by an algorithm is at least theoretically predictable. Beside that type of random number generation could be more costly then the elaborated and practically proved Pseudo-Random Number Generation (PRNG) methods by hardware devices include embedded microcontrollers.**

Firstly we introduce one application then we discuss the concept of randomness and its criteria, then the linear feedback shift registers (LFSR) as traditional devices producing PRNs. In connection of LFSRs we show the polynomial algebraic model of generating PRNs.

In second part of this article we shell show a special decomposition of theoretical background, by two way to microcontroller environment.

I. Introduction

The One Time Pad (OTP) encryption and decryption could be considered in the first approach as handling a standard binary block of a piece of information but those blocks follow each other as a stream cipher do. In the simplest case the encryption process adds an n-bit random number to the same n-bit long input plain text block. That random number is the key that belong to that very block. (There is no encryption if the n-bit key is all zero, consequently all zero key should be avoided.) The decryption process practically is the same. We should be aware that each random key must be used one time only. That is the origin of the name of that method: One time Pad.

In spite of the simplicity of this method in the years of 1930s and 40s a very similar secret communication method was used by the Soviet Union for the so called VENONA project for communicating the details of US Manhattan project. That is not a secret anymore. The practical implementation of such a process needs a serial stream of n-bit random numbers.

There exist as many as 2^n different n-bit binary numbers. If the all-zero n-bit number is excluded we have to generate $2^n - 1$ n-bit numbers which follow each other randomly in a sequence, then the whole sequence repeats. That is what is called pseudo-random (binary) number sequence. The starting n-bit number is the seed of that sequence [1] [2].

II. Golomb's criteria of randomness

Within a random-like bit sequence we call *block* an all 1 bit sequence before and after which there is at least one zero bit. E.g.: 0111110. The name block is independent of the number of 1s in that block. Within a random-like bit sequence we call *gap* an all 0 bit sequence before and after which there is at least one zero bit. E.g.: 100001. The name gap is independent of the number of 0s in that gap. The common name for both block and gap is run. We define the concept of length for runs. There are k identical bits in a k-length run.

Defining randomness of a pseudo-random number sequence (PRNS) two more concepts are needed. One is the *period*, which is the length of the total PRNS. (That is $2^n - 1$).

The other ones are the correlation index and the Autocorrelation function AC(k) which is a single variable function, where k is the measure of shift steps. Since th shift **k** is always an integer, **AC(k)** is not continuous (though we generally map it continuously). Using the concepts introduced above we can define Golomb's criteria of randomness as follows: A PRNS is random-like if, and only if

(a) The 0s and 1s numbers in a period of the bit stream are as well equal as it can be at all.
(b) The number of length $l = 1$ runs in a period must be half of all runs. The number of length $l = 2$ runs in a

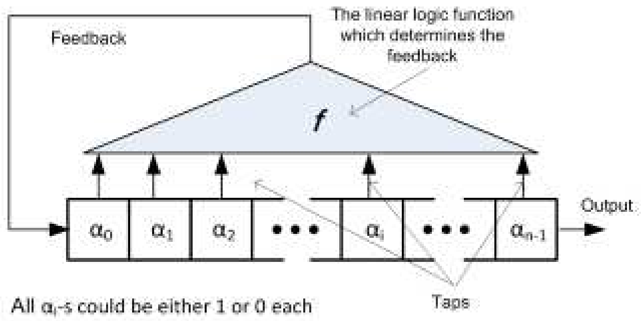
Feedback
The linear logic function which determines the feedback
f
α0
α1
α2
αi
αn-1
Output
All αi-s could be either 1 or 0 each
Taps

Fig. 1. The general concept of Linear Feedback Shift Register (LFSR).

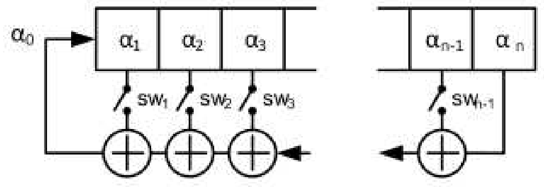

Fig. 2. General switching diagram of a Fibonacci LFSR.

period must be a quarter of all runs and so on; That is the number of length $l = i$ runs let be $1/2^i$;

(c) The value of autocorrelation function AC(k) let be the same for each k, except $k = 0$ An additional condition for that is that the period p must not be a divider of k, but it exists automatically when $k < p$.

The measure of fulfillment of these Golomb [6] conditions show how much a PRNS is random-like.[2].

III. The Linear Feedback Shift Register

It may be stated that the most traditional electronic device of realizing an n-bit PRN is an n-bit shift register with a particular feedback. In fundamental cases the feedback is a *linear logic function*, ie. it consist of linear logic operations only, which are additions (OR and exclusive OR that is XOR. Negation is also a linear operations). Note: There is no logic multiplication among these linear logic operations, ie. AND operation is prohibited. Fig. 1 shows that concept.[3][4]

Not all the taps must exist in a particular case. In a particular state the (binary) content of the register is a particular PRN. Let us denote the logic exclusive OR operation (which is a modulo 2 addition) by a circle-plus symbol: and let that symbol denote the logic gate as well which realizes that operation. Such way the most generalized logic circuit of an LFSR shown in Fig. 2.

The states of switches SW_1, $SW_2 \ldots SW_{n-1}$ determine which cells of the register have effect on the feedback. In other word which taps are active. The states of these switches do not change in a specific application. The rightmost bit always effects the feedback and at least one of the other cells as well. In case of gate-type realization the XOR gates are in serial connection in the feedback line. Consequently their propagation delays are added together resulting a slower operation of the entire device than it could operate in case of having minimal number of XOR gates in the feedback line. We should note, however that in case of modern computer realization that slowing effect does not exist. Still we are looking for realizations in which the number of XOR gates is minimal. This type of LFSR has got simpler algebraic model then other ones. All LFSRs whose circuit structure are the same as Fig. 2 shows called Fibonacci type LFSRs. [5] [6]

IV. The algebraic model of a finite number set and that of an LFSR

The algebraic model of the aforementioned finite number set is a Galois field. All Galois fields contain p^k elements where p must be a prime number. It is also an attribute of any Galois field that it contains one or more elements from which all elements of that very field could be generated by a particular algorithm discussed below. Such an element is a generator (or primitive) element of the given set. The generation process use a so called polynomial algebraic model as follows (1)

$$C(x) = \alpha_0 x^0 + \alpha_1 x^1 + \cdots \cdots + \alpha_i x^i + \cdots \cdots + \alpha_{n-1} x^{n-1}, \quad (1)$$

where α_i takes its actual value from the two element set $\{0,1\}$ and $x^0 = 1$.

A great practical advantage of this model that if we multiply the polynomial by x all coefficients take one step to the right which is a very convenient method of modeling the shift-right operation. When the coefficient α_{n-1} steps to the right, the register would overflow, but for the feedback α_{n-1} will become α_0 if the register is rotated as it is already shown by Fig.1 and Fig.2. We have got the correct shifting model if the multiplication of the polynomial by x is a modulo n multiplication. That is modeling the direct feedback as well. Consequently (2)

$$x \cdot \alpha_{n-1} x^{n-1} = \alpha_{n-1} x^0, \quad (2)$$

and in general (3)

$$x^k \cdot \alpha_{n-1} x^{n-1} = \alpha_{n-1} x^{k \oplus (n-1)}, \quad (3)$$

where $\oplus$ symbol notate here a modulo n addition.

Note that the state of the register will not change if, and only if its content all zero. Because of that fact that state is excluded of all PRNG. The maximal number of all other states can be $2^n - 1$ and that maximum could be realized by choosing an appropriate linear feedback.

V. The switching polynomial and practical considerations

The polynomial (1) is the switching polynomial of an n-bit LFSR. Sometimes called characteristic polynomial as well. This polynomial plays critical rule in generating maximal PRN set and determining the feedback structure of LFSR for a given number of n.

The LFSR could be modeled by the usual way of a synchronous sequential machine (Mealy machine) as well, but the polynomial algebraic modeling is more convenient in case of LFSR. To generate a maximal length of PRN sequence the form of the switching polynomial must be equal to a generator polynomial of the Galois field that is the set of all PRN to be generated. Theoretically any generator polynomial is suitable for generating all $2^n - 1$ members of the maximal PRN set (i.e. Galois field) but there are some practical considerations. Some of them are as follows:

(a) Both α_{n-1} (i.e. the output of the n bit register) and α_0 (i.e. input of the register) coefficients should be included in the polynomial.

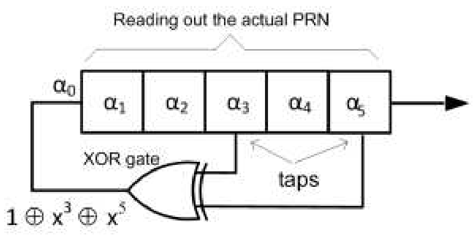

Fig. 3. Five bit Fibonacci LFSR.

(b) Beside those two coefficients at least one of the intermediate coefficients α_k should also be included. Consequently a minimal characteristic polynomial must contain as many as three members. It is not sure, however, that such generator polynomial exists at all in case of a particular length n.
(c) Each existing intermediate coefficient of a generator polynomial determines one tap of the LFSR.
(d) The less number of taps the faster and more reliable operation of the LFSR.
(e) If, and only if a particular n degree polynomial has got more then one generator polynomial, it is advised to choose the one that contains the less members (which is three because of the aforementioned considerations).[7]

Then the general form of a recommended generator polynomial is the following (4)

$$x^n + x^k + 1, \tag{4}$$

It is a rule of thumb applying such form of generator if possible.

Note:

(a) Such form of a generator may not exist for every n.
(b) If it exists at all, deep algebraic considerations needed to determine k. For practical cases one may get the appropriate k from tablets. See for example. The same book publishes a C program for generating primitive polynomials as well [8].
(c) Generating maximal length PRNs the number of taps must be even (incl. the output as one of the taps) and the indices of the taps must be relative primes. In case of one tap plus the output $lcd(n,k) = 1$.
(d) **The generator polynomial determines the feedback structure of the particular n-bit LFSR**.

For demonstration purpose we show and analyze the following 5-bit Fibonacci LFSR. The switching polynomial follows (5)

$$1 + x^3 + x^5. \tag{5}$$

Looking at Fig. 3 we should notice that the tap indices are relative primes.[8]

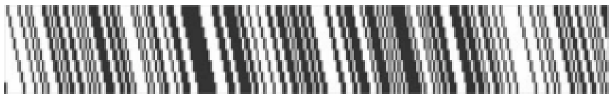

Fig. 4. Representing the operation of an 8-bit Galois LFSR when the seed is 00000001.

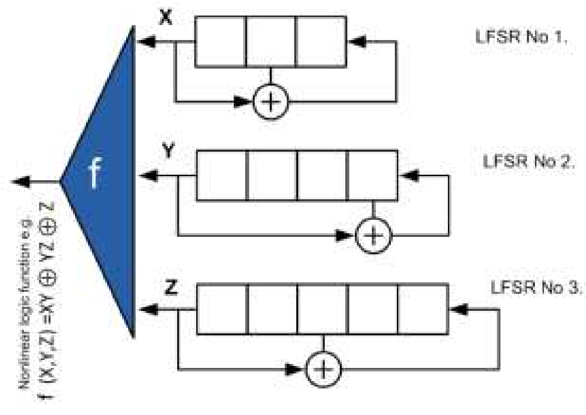

Fig. 5. Nonlinear PRN generator consisting three LFSR.

Just for curiosity we mention that not for all different n exist 3 member generation polynomial, but they may be generated by more member polynomials. For example there are no 3 member polynomials for $n = 12, 13, 14$. In case of an $n = 10$ bit LFSR there are as many as 60 primitive polynomials, but out of them only 2 have 3 members (when $k = 3$ or 7). It also could attract some interest that the autocorrelation function is not always as straight as it is in the above case.

The correcting medicine is to avoid linearity by applying more than one LFSR and combining them by a logic function f (in Fig. 5), which includes nonlinear logic operations, namely logic multiplications (AND operation).

Even the step frequencies of the shift registers are not necessarily equal to each other, but those frequencies must have e a common divisor. The output step-frequency of this arrangement, however, must be equal to that of the highest step frequency register.[8]

VI. Realization in microcontroller environment

There are no miracles, everything what we want to do by microcontroller is time-consuming. Handling of such claims is a natural way by increasing the clock frequency and/or the inner data bus width.

A typical application of embedded microcontrollers is in addition to other functions which may be necessary to produce a random number as well. We assume that the real-time supervision on occasion, the generated random number used for other purposes. Typically, such application is a control of electronic unit, or electro-mechanical device, implement authentication or encryptions occur.[9][10]

Every solution what in section III and V, we propose, typically a long-term application. If an eight bit microcontroller (PIC16F788) has got 8MHz clock frequency, their instruction-

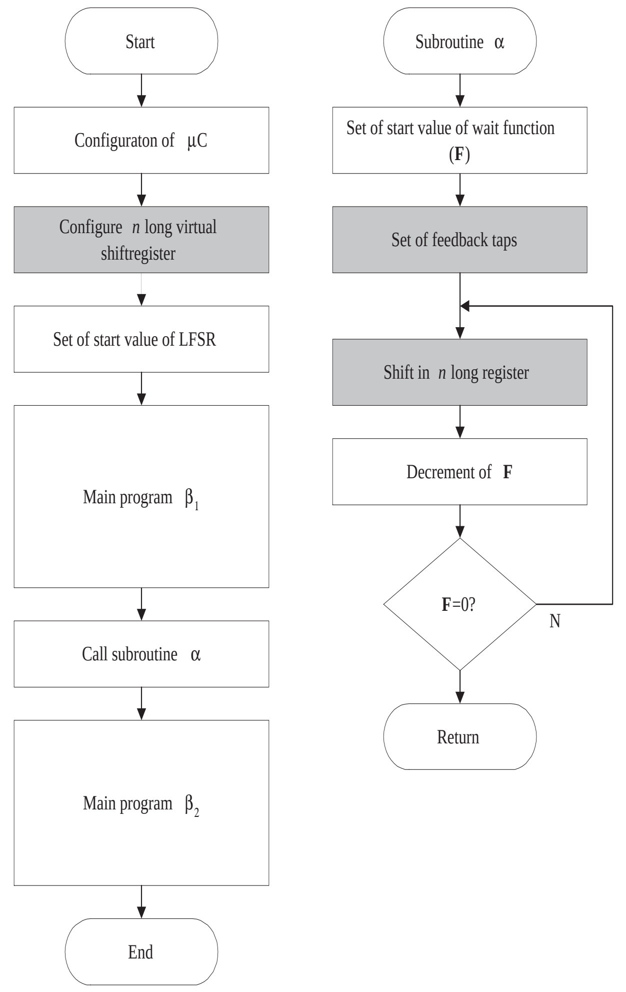

Fig. 6. A main program and waiting subroutine with LFSR function.

time (t_i) is $2,5\mu s$. At an n long virtual shift register there means at least (6);

$$t_s = n \cdot t_i, \tag{6}$$

where t_s is the necessaries time for the total shifting. Of course t_s grown together with n, it can be very significant. The optimal utilization of existing processor resources brings two options, such as;

(a) Latent firmware application,
(b) utilization of hardware abilities of only any firmware contribution of microcontroller.

These two methods can we reduce the microcontroller's resources, thus, the microcontroller is able to achieve better control quality.

VII. Latent firmware solution

The embedded microcontroller is often carried out during the operation delays. NOP instructions by writing them realizing it, often put in loop bodies. There is an opportunity to "blank instruction" replace the program lines in place

```
;------------Main Programm---------------

START
  CLRF TRISD        ;Configuration
                    ;of microcontroller
  BSF  PORTD,7      ;Set of start value
Loop
  NOP               ;"Important" part
  NOP               ;of main program

  CALL ALPHA
  NOP               ;"Important" part
  NOP               ;of main program

  OTO LOOP

;-------------Subroutine-----------------
ALPHA
  CLRF T1           ;Set of wait-time 1
  CLRF T2           ;Set of wait-time 2
LABEL

  RLF PORTD,0       ;Left shift in
                    ;PORTD, result in W
  XORWF PORTD,1     ;Ex-Or PORTD
                    ;and register W

  DECFSZ T1,F       ;Decrement
  GOTO  LABEL       ;of
  DECFSZ T2,F       ;cycle
  GOTO  LABEL       ;variables
  RETURN

;------------General END------------------
  END
```

Fig. 7. Main program and the called waiting subroutine with rejected shifting.

to implement the desired LFSR function, with shifting and feedback function too. Such a solution can be seen in the Fig. 6. In this case, a random number in a register, register array is produced, which then can be read at the appropriate time.[14]

On the Fig. 7 is visible a random number generator algorithm is placed on a waiting subroutine, in assembler language of a popular eight bit Microchip© microcontroller version.[12]

The proposed program coding process and algorithms, unfortunately only a circumstance to be solved. Very carefully to make just such a program, and any change of timing in the operation of the random generation process, the quality may also be affected.[11]

On the Fig. 8 shown in a time functions of the microcontroller outputs, with different triggering conditions of an oscilloscope.

VIII. Utilization of hardware abilities

In the VII chapter we saw a solution, witch can effectively optimize, at a certain application the workload of microcontroller during a random number generation.

The microcontroller's internal peripherals being developed in a way that most of the applications, the expected type of waiting, delaying operation to be relieved of the core logic of controllers. Thus, the available resources be used for other applications.[13]

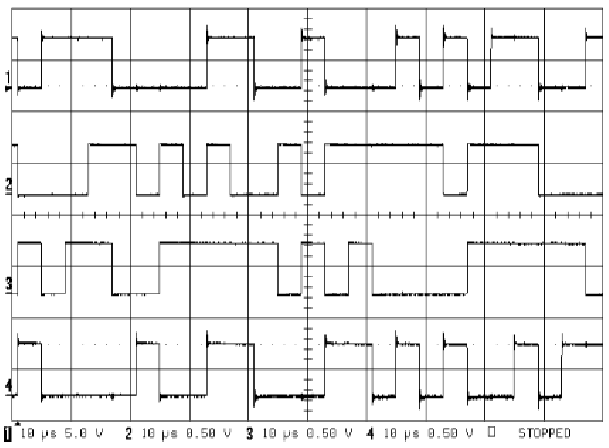

Fig. 8. Time diagram of a microcontroller's parallel port at a random value

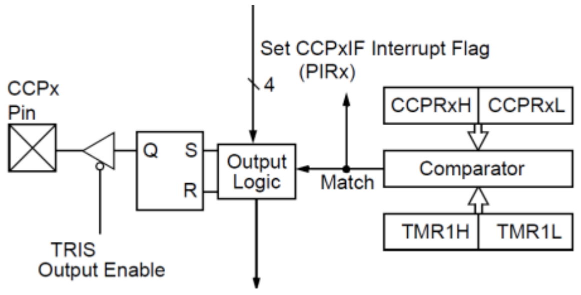

Fig. 9. Block diagram of COMPARE Module of PIC16F887 microcontroller.

In the microcontroller counter type peripheries (TIMER0, TIMER1, TIMER2, CAPTURE) can be found to the serial-parallel conversion is supported, with some parallel serial conversion (also) are used (COMPARE, USART, SPI, I^2C). Fig. 9 shows block diagram of COMPARE module as a typical serial output peripheries, on Fig. 10 seen block diagram of TIMER1 module witch is a typical serial input peripheries.

On the Fig. 11 seen, the connection of the proposed method by which a random number generator can be constructed by suitably chosen micro-peripheries of the microcontroller.[17]

At the internal peripherals of microcontrollers are basically two ways to communicate with the program, a polling process, or a more efficient way the interrupt.

The suggested procedure based on the solution of a previous section (IV). So the proposed random generator 'peripheries' worked itself without program assistance. Main task of sort-time interrupt subroutine is only a reading a parallel data from output of serial-parallel peripheries (TIMER0 module) and generate necessaries virtual feedback combination network (Fig. 1, 2, and write back in parallel-serial device (COMPARE module). Time-consumption during executed the interrupt subroutine is eliminated for the whole random generating.[15] [16]

The Fig. 12 shows an excerpt of the generated random number series.

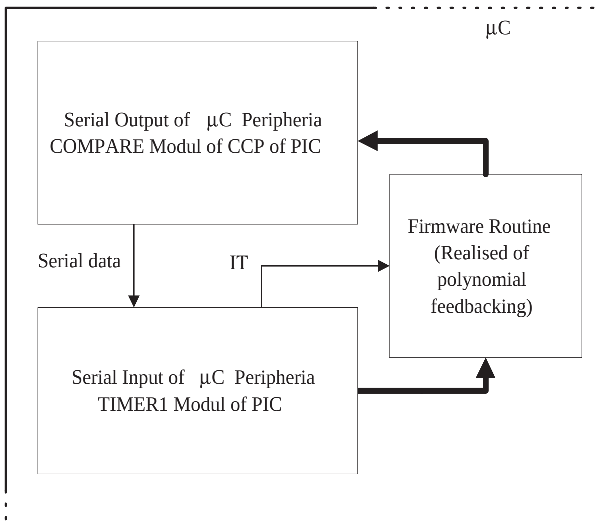
μC
Serial Output of μC Peripheria
COMPARE Modul of CCP of PIC
Firmware Routine
(Realised of
polynomial
feedbacking)
Serial data
IT
Serial Input of μC Peripheria
TIMER1 Modul of PIC

Fig. 11. The practical structure of connected output of parallel-serial and input of serial-parallel type peripheries. Software assistance is in between of peripheries's parallel site.

```
b79a2a1d6f34543ade68a875bcd152eb79a237
a4d6f34548ade21526b79a2a4e6f34549ede79
68a93cbcd1527a79a2a4f6f34548cd1527b00b
a2a4f6234549ec468a93d8ad1527b14a2a4ff6
62a493d8ae1527b15c2a4f62b8549ec572a9a2
3d8ae6527b15cea4f64573e93d8ae7e27b1515
cfc4f62b9fa9ec573f63d8ae7ec7b15cfe6c57
f68d8a1e7ed0b15cfda262b9fb46c573f68c8a
7ed1a1fb468573f68d124ae7ed1a65cfda34eb
fb469e73f68d3ee7edf89fb469f23f681d3e47
d311a7cafda34f94fb469f28f68d3e5a34f940
469f28268d13e506d1a7ca0ca34f941a469f28
68d3a0dc34f9411b869f28d3e506e4a7ca0dca
feb2a1d6f34543ade68a875bcd152eb79a2a4d
f34548ade21526b79a2a4e6f34549ede68a93c
cd1527a79a2a4f6f34548cd1527b09a2a4f623
549ec468a93d8ad1527b1442a493d8ae1527b1
c2a4f62b8549ec572a93d8ae6527bea4f64573
93d8ae7e27b15cfc4f62b9fa9ec573f63d8ae7
c7b15cfe6c573f68d8a1e7ed0b15cfda262b9f
```

Fig. 12. Part of generated binary noise in HEX format.

IX. Conclusion

This article summary and pointed out that a well-known scientific approach of new way of today application necessaries.

We found unusual new application possibilities. We have shown that a procedure for how to optimally adapted to an existing environment, to a real device.

In the future work we will elaborate, and to test of random-number generating application of other type of microcontroller and their micro-peripheries.

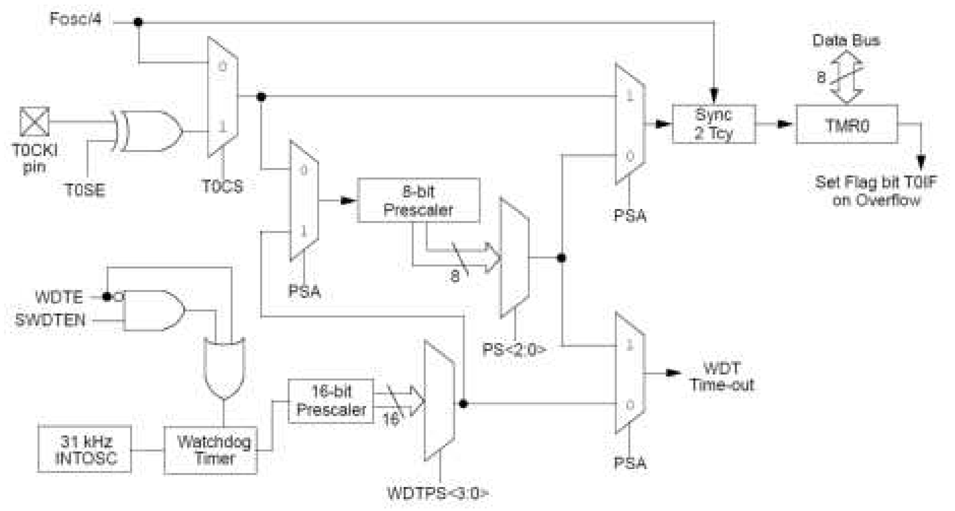

Fig. 10. Block diagram of TIMR0 Module of PIC16F887 microcontroller.

References

[1] Stamp M., Low R. M. Linear Cryptanalysis of Stream Ciphers John Wiley Sons, Hoboke 2007 ISBN: 978-0-470-11486-5

[2] Kasper E. Applied Cryptanalysis www.tca.hut.fi/Studies/T-79.514/slides/S4.Kasper.1c-stream.pdf

[3] Gustavson F.G. Analysis of the Berlekamp-Massey linear feedback shift-register synthesis algorithm. IBM Journal of Research and Development. Vol 20. May 1976. pp 204-212.

[4] Goldberg I., Wagner D. Architectural considerations for Cryptanalytic Hardware www.comms.scitech.susx.ac.uk/fft/crypto/paper.pdf

[5] Fried E. Algebra I., Elemi s lineris algebra. Nemzeti tanknyvkiad, Budapest 2000, ISBN 963 19 1176 4

[6] Golomb S. Shift Register Sequences. Aegean Park Press, 1981. ISBN 0894120484

[7] Wicker S. B. Error Control Systems. Prentice Hall, 1995. ISBN 0-13-200809-2

[8] High Bit Rate Quantum Random Number Generator service. Faculty of Mathematics and Natural Sciences IDepartment of Physics. http://qrng.physik.hu-berlin.de

[9] Fözö, L., Andoga, R., Madarász, L. Mathematical model of a small Turbojet Engine MPM-20. Studies in Computational Intelligence., 2010 Springer-Verlag Berlin Heidelberg, Vol. 313 (2010), pp. 313-322. - ISSN 1860-949X

[10] Fözö, L., Andoga, R., Madarász, L. Situational control, modeling and diagnostics of large scale systems. Towards Intelligent Engineering and Information Technology., 2009 Springer-Verlag Berlin Heidelberg, No. 143 (2009), pp. 153-164. - ISSN 1860-949X

[11] Vámossy, Z. Delphi a gyakorlatban (Mintafeladatok megoldssal). ZAK Kiad, Bicske, 19972000, hrom kiads

[12] Sergyán, Sz. Content-Based Image Retrieval Using Automatically Determined Color Regions of Images. Proc. of 7th International Symposium on Applied Machine Intelligence and Informatics, Herl'any, Slovakia, January 30-31, 2009, pp. 41-45., ISBN 978-1-4244-3802-9, IEEE Catalog Number: CFP0908E-CDR

[13] Ádám, T., Amadou, K., Varga A.,Vásárhelyi, J. Active noise cancellation strategies based on digital signal processing - an overview. Engineering achievements across the global village. Radom : Printing House of the Institute for Terotechnology, 2005. p. 305-312. (The International Journal of Ingenium, ISSN 1363-514X ; 1-4.) International Conference on CAD/CAM, Robotics, and Factories of the Future (21.) (2005) (Krakow)

[14] Gy. Györök, L. Simon Programozható analóg áramkör megszakításos alkalmazása mikrovezrlö környezetben. *Alkalmazott informatika és határterletei Szimpzium, Budapesti Müszaki Föiskola Regionális Oktatsi és Innovcis Központ, 2009. november 6. Székesfehérvár, ISBN ISBN 978-963-7154-94-2.*

[15] Gy. Györök The FPAA Realization of Analog Robust Electronic Circuit. Proc. IEEE Internacional Conference on Computational Cybernetics: ICCC 2009. Palma de Mallorca, Spanyolorszg: pp. 1-5. Paper 10. (ISBN:978-1-4244-5311-5) 2009.11.26-2009.11.29.

[16] Gy. Györök A-class Amplifier with FPAA as a Predictive Supply Voltage Control. CINTI 2008 9th International Symposium of Hungarian Researchers on Computational Intelligence and Informatics, Budapest, Hungary. Budapest, Magyarorszg, 2008.11.06-2008.11.08. pp. 361-368.

[17] Gy. Györök, L. Simon The FPAA Realization of Analog Robust Electronic Circuit. IEEE Internacional Conference on Computational Cybernetics: ICCC 2009. Palma de Mallorca, Spanyolorszg, 2009.11.26-2009.11.29. Palma de Mallorca: pp. 1-5. Paper 10. (ISBN:978-1-4244-5311-5)