Real-time requirements meeting in PLC control system

Krzysztof Oprzędkiewicz*

AGH University of Science and Technology, Kraków, Al. Mickiewicza 30, Kraków, Poland

Article history:
Received 12 December 2018
Received in revised form
26 June 2019
Accepted 26 June 2019
Available online 30 June 2019

Abstract
In the paper a problem of real-time requirements meeting at PLC is presented. Basic parameters associated to real time and factors determining their value were discussed. Real-time reliability indices for PLC and methods of their estimation and experimental testing was also proposed. Results were depicted by examples.

Keywords: PLC, 61131 standard, real-time systems

Introduction

PLC controllers are currently the most commonly used in industrial control systems. They perform a lot of tasks in the field of logic, continuous and sequential control and more often in security systems. This is due to many factors: a very extensive market offer, a large "openness" of the platform and the standardization of the design and programming methods of these devices. Currently used PLCs are able to implement any control algorithm or other calculation operation, even with relatively high complexity. The only important limitation to which one should always pay attention is the need to meet the real time requirements while the controller is in operation, and in particular – to provide temporal determinism.

The the PLC based control system is a real-time computer system, but in most cases it is a "hard real time" class system, which means that not meeting the real-time requirements (in particular not exposing a specific signal to the output at a given time) may in a particular situation, have disastrous consequences for the controlled process. In a typical situation, exceeding the given time regimes always results in generating a software error and switching the CPU of the controller into an emergency stop condition, which further complicates the situation.

The need to avoid time errors during the design and subsequent maintenance of the control system with the PLC implies the need for a more accurate analysis of time requirements and avoid any errors already at the software design stage. It is connected with the necessity of making some estimates and possible later experimental tests at the testing and commissioning stage of the control system.

At the same time, it should be noted that the issues discussed above are not analyzed very often. A certain amount of information in this field can be found in the documentation for equipment or literature in the field of PLC programming (eg [4], [5], [6]). Detailed examples of estimations and measurements of experimental time parameters can be found in the author's earlier works, among others in [8], [9], [10], where these works discuss specific cases of a specific system and specific calculation operations.

The purpose of this work is to propose and justify the reliability indicators describing the fulfilment of real-time requirements during the operation of the PLC control system. The general results will be supplemented with experimental examples from real controller systems. In particular, the following issues will be discussed in the work:

- PLC time parameters and factors affecting their value,
- Proposals for real-time reliability indicators for a PLC system,
- Remarks on PLC software optimization in the sense of maintaining given real-time reliability indicators,
- Comments on experimental methods for determining reliability indicators,
- Examples of experimental estimates of the proposed real-time reliability indicators.

Materials and Methods

PLC time parameters and factors affecting their value

The basic time parameters related to the operation of the PLC can be defined on the basis of the program cycle of the PLC (see e.g. [4]). It is shown in Figure 1.

The cycle time T_c is the duration of a single program cycle, from initialization to diagnostics, and through the response time. The response time is the duration from reading inputs to write outputs associated to these inputs (i.e. it is the time to perform a single control loop).

*Corresponding author: kop@agh.edu.pl

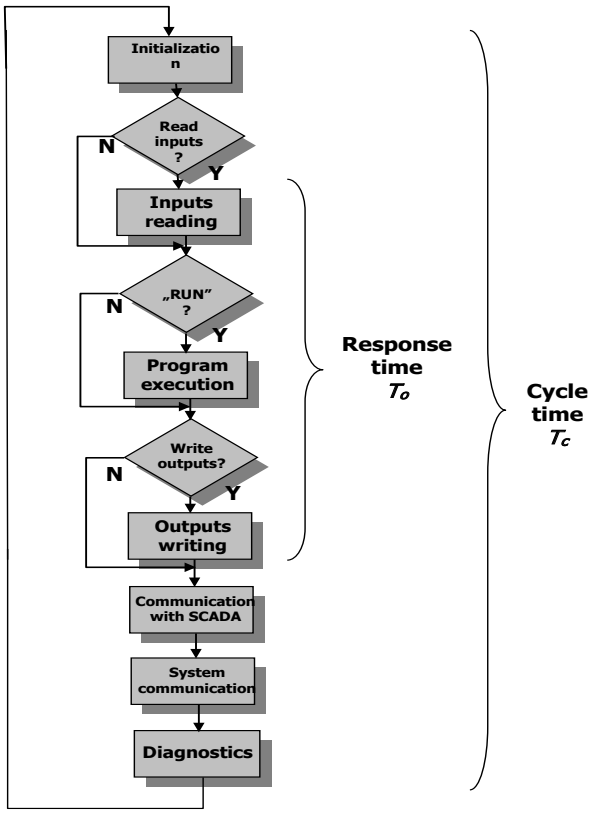

Figure 1. PLC programming cycle, cycle time and response time

It is extremely important that during operation of the PLC system, the values of both cycle time and response times are not constant, only change within a certain range, between the minimum and maximum values, which can be recorded as follows:

$$\begin{cases} T_o^{\min} \le T_o \le T_o^{\max} \\ T_c^{\min} \le T_c \le T_c^{\max} \end{cases} \quad (1)$$

It should be noted that it is possible to estimate only the minimum and maximum values of the cycle time and response time given in relation (1). The estimates of these values are given in the equipment documentation and depend on many factors that will be discussed in the next part of this article. It should be added here that their accuracy is strongly determined by the accuracy of information in this field provided by the manufacturer of the specific equipment.

An example of the estimated response time for the SIEMENS SIMATIC S7 300 PLC system is given in the hardware documentation and has been discussed in previous author's works, e.g. [8]. The minimum and maximum values of the response time can be estimated as follows:

$$\begin{cases} T_o^{\min} = T_R + T_{os} + T_u + T_w + T_{Di} + T_{Do} + T_t \\ T_o^{\max} = 2T_R + 2T_{os} + 2T_u + 2T_w + T_{Di} + T_{Do} + T_t + 2T_{DP} \end{cases} \quad (2)$$

where:

T_R, T_w – means the time of reading and writing in / out,

T_u – means the time of executing the user program,

T_{os} – means the execution time of the operating system (depending on the CPU type)

T_{Di}, T_{Do} are the delay time associated with reading / writing in / out (A / C conversion time and C / A conversion time, depending on the type of transducers),

T_t – means the time of execution of the timers used in the program,

T_{DP} – means the communication time via the PROFIBUS network.

The read times of the T_R inputs and the write of the outputs of the T_w controller depend on the number of reads and write bytes and on the location of the inputs and outputs. It can be saved as follows:

$$T_R + T_W = K + A \cdot LB_0 + B \cdot LB_{1\div3} + D \cdot LB_{DP} \quad (3)$$

In equation (3) LB_0 is the number of bytes to service on rack 0, $LB_{1–3}$ means the number of reads and write bytes on racks 1 – 3 (where rack number 0 is directly connected to the CPU of the controller), LB_{DP} means the number of bytes read and written via PROFIBUS DP, A, B, D, K are constants whose value depends on the type of the central unit and the number of the bus. For example, reading one analog input requires reading two bytes, reading one group of 8 binary inputs requires reading one byte.

The time of execution of the entire user program or its specific fragment (e.g. a single function). Here it can be estimated as follows:

$$T_u = F \sum T_{instr} \quad (4)$$

where F is a coefficient dependent on the type of CPU, eg. for CPU-315 (SIEMENS SIEMATIC S7–300), F is equal to: F = 1.15 and T_{instr} is the time it takes for a single statement to be executed. The value (4) is influenced by the durations of particular instructions. These values are determined by:

1. The type of data on which the instruction operates. Logic type operations on bit sequence data are performed the fastest. Arithmetic operations performed on fixed-point data (types INT and DINT) are performed several times faster than operations performed on floating point data (see eg [5]). A similar situation occurs in the case of comparison operations – also in this case it is much faster performed on fixed-point data.
2. Type of operation performed. For example, the longest is the determination of: the exponential and logarithmic function as well as the trigonometric functions of ordinary and arcus type. In addition – these operations are only possi-

ble to be performed on REAL data. For example, the determination of the ARCSIN function takes about 2.5 [ms] on the SIEMENS SIMATIC S7 300 controller. Detailed information on this subject can be found in the hardware documentation.

3. The language in which the source code of the considered fragment of the software has been prepared. Software written using one of the text languages: Instruction List (assembler) or Structured Text (high level language Pascal type) has much better real time properties than software prepared using graphical language: ladder language, function chart language or Sequence Graph. The relationship between cycle times and responses and the language in which the software was written will be shown in next sections.

In turn, between the response time specified by (2) – (4) and the cycle time of the controller the following relationship takes place:

$$T_s = T_o + T_{DP} + T_{MPI} + T_{int} \tag{5}$$

In (5) T_{DP} is the communication time using the PROFIBUS DP network, T_{MPI} means the time of communication with the programmer or the host computer via the MPI interface and serial port, T_{int} means the total activation time of all interrupts. Equations (2) – (5) can be used to estimate the execution time of a specific set of computational operations, it should be noted that these estimates are very "cautious" and the actual, measured times for specific software are shorter than estimated. An example of comparison of "theoretical" estimates with experimental results is discussed in [8].

The statistical distribution of the cycle time and response time values for the multiple execution of the same calculation task is consistent with the normal distribution. Examples of cycle times and response times distributions for actual controller systems will be provided in the next sections.

Proposals of real-time reliability indicators for a PLC system

In the case of considering real time requirements during operation of the PLC, critical parameters for the correct operation of the system should be considered such time parameters, which failure to comply will result in one of the system's time errors or not to issue a control signal at a given time, which in the case Continuous control systems are equal to the sampling period of the algorithm.

For the above reasons, the following can be used as real-time reliability indicators for a PLC:

- Maximum value of the cycle time of the controller T_c^{max},
- The maximum response time of the T_o^{max} driver,
- Actual maximum execution time for a specific set of instructions, critical for the entire application in terms of speed: T_{kr}.

It can be seen that there is the following relationship between the T_o^{max} and T_{kr} indicators:

$$T_o^{max} > T_{kr} \tag{6}$$

Real-time requirements in the sense of the maximum cycle time will be met if the following relationship is maintained at any time during the controller's operation:

$$T_c^{\max} < T_{hc} \tag{7}$$

Where T_{hc} is the maximum duration of the driver cycle, which must be defined at the hardware configuration stage. Exceeding this value generates an error and causes the CPU to enter the emergency STOP state and the need to perform appropriate procedures to restart the system.

Real-time requirements in the sense of response time and in the sense of time when critical instructions T_{kr} will be executed will be met if at any time during the controller's work:

$$\begin{aligned} T_o^{\max} &< T_p \\ T_{kr} &< T_p \end{aligned} \tag{8}$$

Where T_p is the sampling period of the control algorithm, which is determined in each case based on the dynamics of the controlled process. Not issuing the control signal during this time results in the lack of updating of the signal issued to the controller's output. In case if the signal is a control signal, it will not be updated, and if it is an alarm signal, it will not start any procedure related to its reaction in the given time.

Remarks about PLC software optimization in the sense of maintaining given real time reliability indicators

Maintaining the set levels of the proposed reliability indicators requires a proper approach already at the stage of preparing the source code of the software. The following general notes are relevant to the preparation of the software:

1. The basic computational operations performed on numerical data of the DINT type are much faster than the analogical operations performed on the REAL type data, with the same precision of calculations (the same length of the machine word).
2. The hardware configuration should contain a minimum set of inputs and outputs necessary to perform a specific task. If some installed inputs and outputs are not used, they should be deactivated at the hardware configuration stage, since only deactivation guarantees that they will not be read / written during each cycle.
3. The numerical parameters of the control algorithm, which do not have to be calculated in real time in each

program cycle, should be determined within separate program elements that are triggered when needed during, for example, the auto-tuning experiment, and then stored. Another possibility is to perform calculations of this type outside the controller (e.g. at the SCADA application level) and sending results to the controller via the network.

4. In a situation where it is required to quickly perform specialized calculations of high complexity, it is worth considering the expansion of the controller by intelligent modules with their own processors, dedicated to specific tasks, e.g. axis positioning or temperature control. The speed of operation of such a module is much greater than the speed of the central unit. The disadvantage of this approach is that the programmer is not able to modify the algorithm implemented by such a module (the software is permanently saved and only some of its input parameters can be defined).

Remarks about experimental methods for estimating the considered reliability indices

In case, if the documentation of the equipment does not allow an accurate estimation of the time of the execution of a specific set of computational operations, e.g. based on formulas (2) – (5), it seems advisable to perform simple experimental tests at the application launch stage. Examples of such tests are discussed with details in [8], [9] and will also be shown in this paper. When preparing and conducting experimental tests, estimating the proposed reliability indicators, it is advisable to adopt the following general assumptions:

1. Tests should under no circumstances interfere with the hardware or driver firmware (in order not to lose, for example, the manufacturer's warranty for the equipment).
2. During the preparation and implementation of tests, the specific features and capabilities of specific hardware and software systems should be used, in particular enabling access to the controller's real-time clock. During the tests it is easiest to use the clock in the controller.
3. An important part of the experimental station is the SCADA application cooperating with the controller, which allows you to start and stop experiments and to collect measurement results. In a real practical situation for testing, you can develop an application dedicated to the supervision of a controlled process. After the experiments are completed, additional functions related to real-time supervision can be used to supervise the fulfilment of time requirements during application operation and early detection of possible pre-failure states.

Results and Discussion

The tests discussed were carried out in accordance with the general principles discussed above using two different examples of PLCs, differing in size, computing power and the available programming environment. None of the presented tests used the process interface of the controller – only central units connected with the SCADA application were used for the experiments.

Example 1: Experimental measurement of the cycle time and response time of the GE FANUC VERSAMAX MICRO controller

As a first example, the calculation of the high complexity calculation function on a small GE FANUC VERSAMAX MICRO PLC has been considered. A detailed description of the experiment can be found in [10]. The most important assumptions and results are given below. The aim of the experiment was to determine the T_{kr} time of solving a given computational function of high complexity and to determine the shortest cycle time T_{hc} ensuring the correct operation of the controller while solving this task.

In the considered example, the complex computational task is the kinematics equation for the kinematic chain of the industrial robot IRp6, described by (9):

$$\begin{cases} x_1(t) = l_1(t)\sin(\alpha(t)) + l_2(t)\sin(\alpha(t)+\beta(t)) \\ x_2(t) = l_1(t)\cos(\alpha(t)) + l_2(t)\cos(\alpha(t)+\beta(t)) \end{cases} \quad (9)$$

where:

$x_1(t)$ – means the position on the *X1* axis in the two-dimensional reference system,

$x_2(t)$ – means the position on the *X2* axis in the two-dimensional reference system,

$l_1(t)$ – means the current length of the first robot arm,

$l_2(t)$ – means the length of the robot's second arm (fixed value),

α (t) – means the current angle (in radians) between the first arm and axis X1,

β (t) – means the current angle (in radians) between the second arm and the straight line resulting from the extension of the arm l_1.

The scheme of the experimental set-up is shown in simplified form in Figure 2. The experimental tests were carried out on the GE FANUC VERSAMAX MICRO controller with 23 point version. The SCADA application and the configuration environment were located on a PC class computer that communicated with the PLC via the RS 232 port and the RS 232 – USB converter. All numerical parameters for equation (9) were given in numerical forms from the SCADA application level.

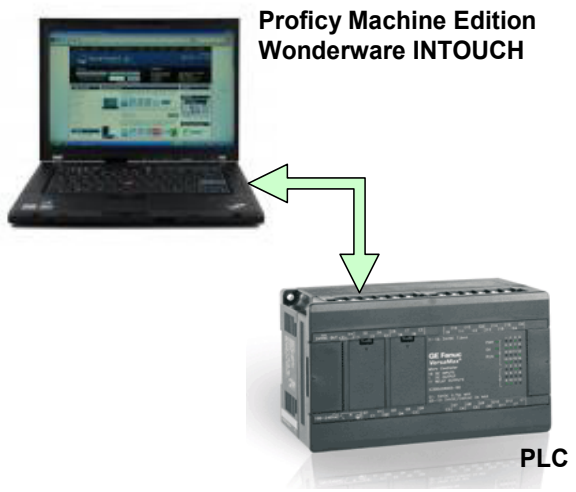

Figure 2. Experimental system with GE FANUC

Experiments were executed with the following assumptions:

- The software has been prepared using the ladder language (it is one of two programming languages available in the system under consideration),
- The program has been divided into sub-programs (these are the only organizational elements of the software available on this platform),
- During the experiments, the controller does not use any inputs or process outputs, starts and stops data collection and input parameters input: e.g. the number of repeats of the N_p loop or the number of samples to be collected takes place only from the SCADA application level,
- The length of the cycle time in the controller is set without limit (parameter SWEEP MODE: NORMAL),
- The recording of measurement results is also made from the SCADA level in a readable format by MS EXCEL,
- The number of executions of the test function determining the value of equations (9) is assumed equal to: N_p = 1000,
- The number of samples collected was taken as 1000.

The results of the experiments are described by the histograms shown in Figures 3 and 4 and Tables 1 and 2. In Figure 3 and in Table 1, the results of measurements of the T_{obl} calculation time are described, and Figure 4 and Table 2 give the results of measurements of the entire cycle time during execution experiments. In both tables, the values of real-time reliability indicators determined in this example are marked. Based on the analysis of histograms from Figures 3 and 4, it can be seen that they can be described by a normal distribution:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)}{2\sigma^2}} \tag{10}$$

In (10) σ is the standard deviation and μ is the median. The function (10) for each situation is plotted on histograms. The searched T_{kr} time is equal to the maximum value of T_{obl} in the considered case.

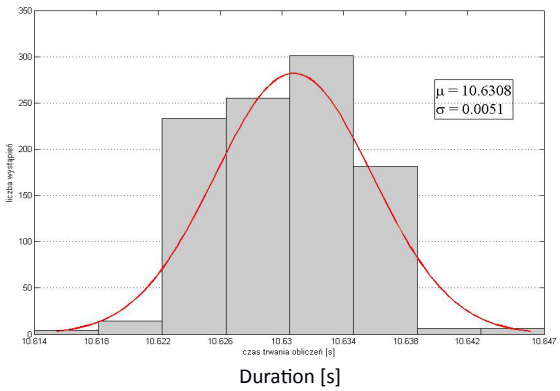

Figure 3. Distribution of time T_{obl} for Example 1

Table 1. Statistical parameters of time T_{obl} in Example 1

Parameter	Value
mean	10.6308 [s]
Maximum: T_{kr}	**10.647 [s]**
Minimum	10.614 [s]
Range	0.033[s]
Median μ	10.63 [s]
Standard deviation σ	0.000051 [s]

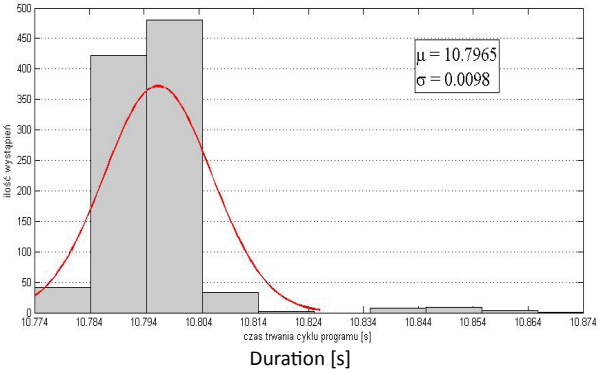

Figure 4. Distribution of time T_c for Example 1

Table 2. Statistical parameters of time T_c

Parameter	Value
mean	10.7965 [s]
Maximum: T_{kr}	**10.876 [s]**
Minimum	10.774 [s]
Range	0.102 [s]
Median μ	10.797 [s]
Standard deviation σ	0.0098 [s]

Based on the analysis of the above results, it can be seen that in the example considered the test function execution is the main part of the CPU load during program execution. This is indicated by the comparison of the time values: T_{kr} and T_{cmax}, where the duration of the entire cycle is only slightly longer than the time of performing the test function.

Based on the above results, you can determine the value of the maximum cycle time determined during the configuration of the T_{hc} equipment that guarantees the correct operation of the controller while solving this task: T_{hc} = 11 [s].

At the same time, it should be noted that the tested controller is in no way suitable for controlling an industrial robot, since during control of the robot the control signal must be exposed in less than 1 [ms].

Example 2: The dependence of the execution time of the calculation operation on the programming language

As the second example, let us consider the relationship between the maximum time of solving a given computing function T_{kr} and the programming language used to record the source code of this function. The tests were performed on medium-sized SIEMENS SIMATIC S7 300 PLC. The scheme of the experimental set-up is shown in Figure 5.

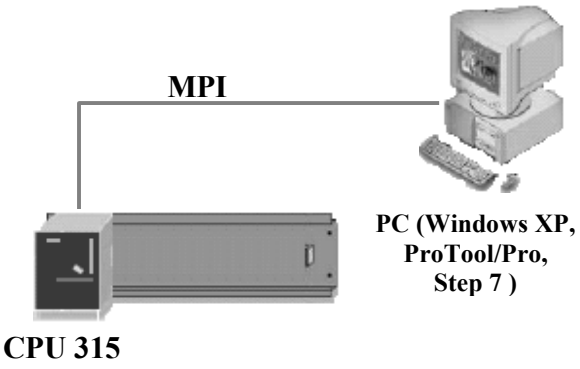

Figure 5. Experimental system with PLC SIEMENS S7 300

The experimental station consists of three main parts:

1. CPU 315, without process interface.
2. PC computer with SIEMENS CP-5611 communication card and STEP-7 configuration software and SCADA ProTool / Pro system for collecting experimental results.
3. Simple MPI network to connect the CPU to the master level.

As part of the test, the calculation of the value of the following test expression was considered:

$$Y:=\text{ASIN}\,(\text{SIN}\,(0.25*\pi) + 0.01) \quad (10)$$

For the study of the time of performing the action (10), the method developed by the author of the work and discussed inter alia in the work [8] was used. This method requires the use of the Sequence Grafting software tool (SFC) and a timer activated concurrently with the procedure being tested is used to measure the execution time of the procedure. The time of execution of the loop containing a large number of performance forms (10) stored in a function built using one of the programming languages is measured. The interrelation of individual elements of the testing program is shown in Figure 6.

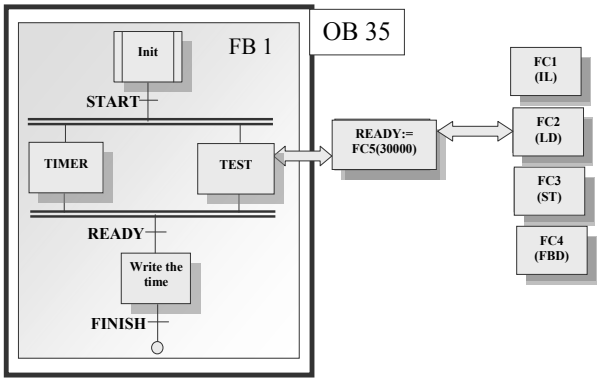

Figure 6. Experimental system with PLC SIEMENS S7 300

The test expression (10) was performed on REAL data. It has been saved in functions FC1 - FC4 type VOID (not returning any result), each of these functions is built using one of the tested PLC programming languages:

- FC1 - using the STL instruction list (assembler),
- FC2 - using the ladder LD language,
- FC3 - using the SCL high-level language,
- FC4 - using the language of FBD function diagrams.

Each of the functions FC1 - FC4 was performed in the FOR.. DO loop in the FC5 function of the BOOL type, returning TRUE after the end of the loop execution. Setting the result of action FC5 to TRUE indicates completion of the loop with the tested activity. The input argument of the FC5 function is the number of loop executions with the function being tested. In the case under consideration, a number of 30,000 loop executions were adopted.

The FC5 function is called in the action associated with the TEST stage of the sequence graph, which is activated in parallel with the TIMER step activating the timer to measure the execution time. Setting the output variable READY to TRUE ends the concurrent stages TIMER and TEST, goes to the next stage and writes the current value of the timer time to the internal memory cell, from which it is then read by the SCADA application and saved in the MS EXCEL file.

Executing an instance of FB1 function block containing a graph is called from the level of the organizational unit OB35, activated by a clock interrupt with a period of 2 [s].

As part of the experimental research for each language, 1000 measurements of the time of performing the function of FC5 were made. The results of the experiments are given in Table 3

and in the histograms in Figures 7a-7d. Table 3 also shows the values of the reliability index T_{kr}, which are equal to the maximum value of the time the function was performed.

Table 3. Values of execution times for testing functions FC1 – FC4 in Example 2:

Language	Minimum time [ms]	**Maximum time (T_{kr})[ms]**	Most probable time [ms]
STL	95	**151**	100
LD	181	**259**	183
STEP7-SCL	86	**166**	100
FBD	189	**428**	191

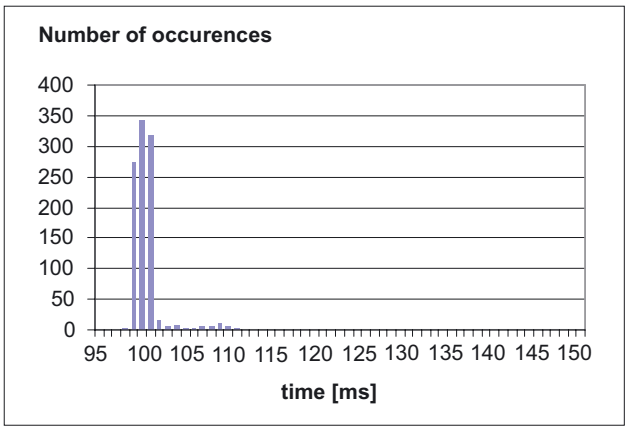

Figure 7a. Distribution of execution Times during use of STL

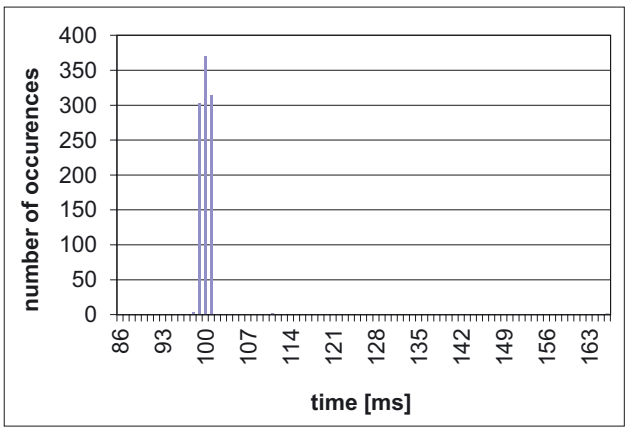

Figure 7b. Distribution of execution times during use of STEP7-SCL

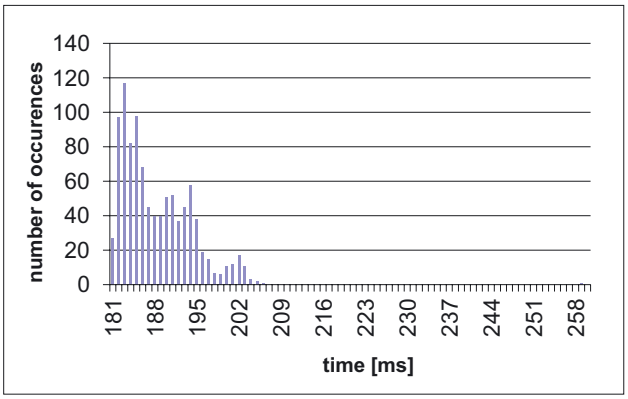

Figure 7c. Distribution of execution times during use of LD

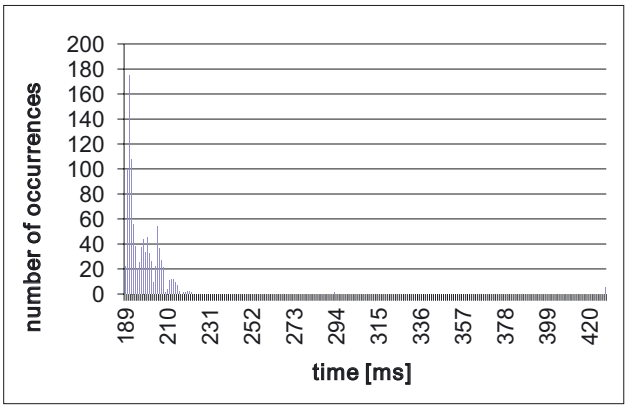

Figure 7d. Distribution of execution times during use of FBD

After analyzing table 3 and histograms 7a–7d, it can be noticed that in the case of the controller system considered, using one of the available text type languages ensures achieving a much lower reliability index T_{kr} and, consequently, a much faster program execution than using graphic languages.

Conclusions

Final remarks for the work can be formulated as follows:

- In the case of PLC control systems it is possible to formulate reliability indicators describing the meeting of real-time requirements and these indicators can be both estimated theoretically, as well as in a simple way determined experimentally using the capabilities offered by a specific controller system.
- New hardware solutions offer much greater opportunities in terms of system speed, while the analysis of its operational reliability should be performed in any situation where high speed controller operation is required while the computational complexity of the algorithm is high.

References

1. Berger H. Automating with STEP 7 in STL and SCL: SIMATIC S7-300/400 Programmable Controllers. SIEMENS; 2012.
2. Berger H. Automating with SIMATIC: Controllers, Software, Programming, Data. SIEMENS 2013.
3. John KH, Tiegelkamp M. IEC 61131-3: Programming Industrial Automation Systems. Concepts and Programming Languages, Requirements for Programming Systems, Decision Making Aids. Springer; 2010.
4. Kasprzyk J. Programowanie sterowników przemysłowych. WNT; 2012.
5. Kwaśniewski J. Programowalny sterownik SIMATIC S7-300 w praktyce inżynierskiej. Wyd. BTC; 2009.
6. Kwaśniewski J. Język tekstu strukturalnego w sterownikach SIMATIC S7-1200 i S7-1500. Wyd. BTC; 2014.
7. Lewis RW. Programming industrial control systems using IEC 1131-3. IEE; 1998.
8. Oprzędkiewicz K. Praktyczne sterowanie systemami dynamicznymi z widmem punktowym i parametrami przedziałowymi. UWND AGH; 2008.
9. Oprzędkiewicz K. Programowe metody diagnostyki spełnienia wymagań czasu rzeczywistego w systemach PLC. In: Trybus L, Samolej S, editors. Metody wytwarzania i zastosowania systemów czasu rzeczywistego. Warszawa: PL Wydawnictwa Komunikacji i Łączności; 2010. p. 163–172.
10. Oprzędkiewicz K. Programowe testy spełnienia wymagań czasu rzeczywistego na sterowniku PLC z wykorzystaniem języka LD. PAR Pomiary Automatyka Robotyka; 2011;15(3):71–75.