Edward HRYNKIEWICZ, Mirosław CHMIEL

Politechnika Śląska, Instytut Elektroniki

About Programmable Logic Controller - step by step

Streszczenie. *W artykule krok po kroku przedstawiona jest i objaśniona podstawowa struktura logicznego sterownika programowalnego (PLC) oraz omawione są podstawy programowania takiego sterownika. Autorzy prowadzą czytelnika od prostego cyfrowego układu mikroprogramowanego poprzez kolejne etapy rozbudowy o licznik programu, dekoder instrukcji, moduły wejścia wyjścia, liczniki i układy czasowe, wewnętrzne magistrale układ arytmetyczno logicznywyjaśniając działanie sterownika, ideę programowania i jego wykorzystanie. Ponadto zaprezentowano kilka prostych przykładów zastosowania. (**O programowalnym sterowniku logicznym - krok po kroku**).*

Abstract*. In the paper a structure of PLC and PLC programming are described and explained step by step. The authors start description from simple micro-programmed digital circuit and next developing it with program counter, instruction decoder, input/output modules, counter and timer modules, internal busses and arithmetic logic unit explain an operation of the PLC and the idea of programming and utilization of such devices. A few examples of considered PLC simple applications are presented.*

Słowa kluczowe: mikroprogramowany, struktura programowanego sterownik logiczny (PLC), programowanie sterowników.
Keywords: micro-programmed digital circuit, PLC structure, PLC programming.

Introduction

Programmable logic controllers (PLC) are used for control different processes in industry, automotive, intelligent buildings and other applications. Due to this fact they should be universal. An universality is obtained by re-configurability (the PLCs are equipped with different I/O modules) and programmability (the PLCs are specialized for given control task by configuring and programming). Even though the programmable logic controllers are first applied in industry at seventies years of the last century the questions are often asked about their structure and the way of their operation. There are a lot of publications devoted to programmable logic controllers as for example [1,2,3,4,5,6,7,8,9] but they describe them on very general level. From the point of view of electronic structure a PLC is a especially designed digital system. The idea of such system will be explained by means of following example:

Example 1

Let us consider a control unit for an electric heater. The heater is equipped with push button A for controlling a fan, push button B for controlling a heater and sensor C informing that adjusted temperature has been reached. An operation of the electric heater may be described by the state diagram presented in fig.1.

1 - heater and fan are switched of
2 - fan is switched on
3 - heater and fan are switched on

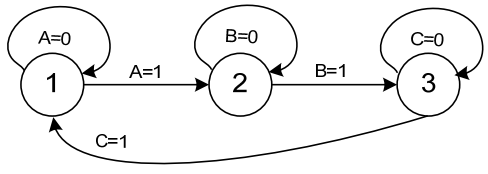

Fig.1. State diagram for example 1.

To design a control unit it is necessary to introduce state encoding:

1 - 01
2 - 10
3 - 11

For such state encoding the state diagram from fig.1. may be implemented in logic circuit shown in fig.2.

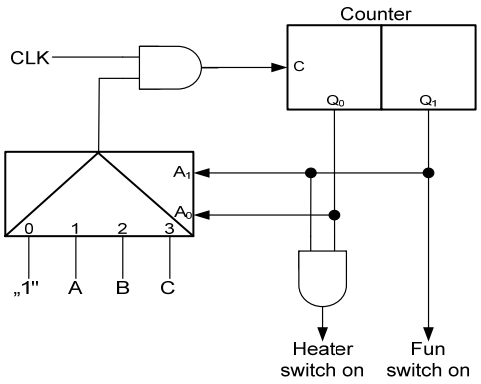

Fig.2. The heater control unit in counter-multiplexer version.

At the moment let us change the structure of the electric heater and its behavior at state 3. Let us equip the heater fun with fan rotor revolution sensor D. The sensor output equal to logic “1” means that the fun operates properly. The fun is out of operation if the sensor output is equal to logic “0”. In state 3 the condition C is checked and if it is equal to logic “0” the system goes to state 3’ in which the condition D is observed. If D is equal to logic “0” the heater is immediately turned off. In opposite situation the system come back to state 3. This is shown in state diagram presented in fig. 3.

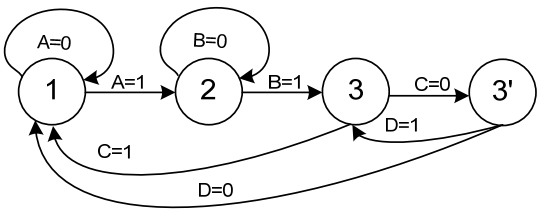

Fig.3. Modified state diagram from Fig.1.

State encoding for modified state diagram can be written as follows:

1 - 01
2 - 10
3 - 11
3’ - 00

If control unit is described by a state diagram consisting of the states from which there are two outputs - one to the same state and other to the next state (as it is seen in fig.1.) - the implementation may be done in counter- multiplexer form. For the state diagram shown in fig.3. such implementation would be complicated and expensive. A better solution is utilizing microprogrammed control circuit shown in fig.4 [10,11].

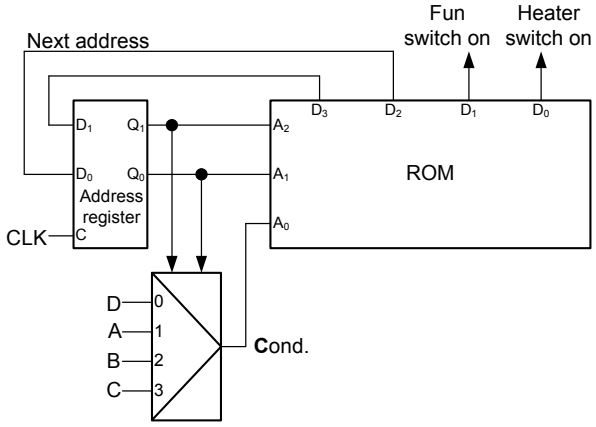

Fig.4. The microprogrammed implementation of the control unit described be the state diagram shown in Fig.3.

The microprogram stored in Read Only Memory (ROM) is written in Table 1.

Table 1. The microprogram representing the state diagram from fig.4.

State	Actual address			Data			
	Q_1	Q_0	Cond.	Next address		Fan	Heater
	A_2	A_1	A_0	D_3	D_2	D_1	D_0
1	0	1	0	0	1	0	0
	0	1	1	1	0	0	0
2	1	0	0	1	0	1	0
	1	0	1	1	1	1	0
3	1	1	0	0	0	1	1
	1	1	1	0	1	1	1
3’	0	0	0	0	1	1	1
	0	0	1	1	1	1	1

The ROM in fig.4 has only four bits output word and three bits address. But when a control task is more complicated, especially when number of states in state diagram is large and number of output control signals is large the ROM capacity become large too. To avoid expanding of a number of output word bits the counter may be used for generating an address instead of storing next address in memory. Moreover the states of output signals may be coded. It means that it is necessary to use decoder (called instruction decoder) at the memory outputs to obtain control signals for particular states of control unit. Some decoded instructions are dedicated for the control unit itself (for example such instruction is “jump” which writes-in to the program counter new address when it occurs in control program). The described modifications are shown in fig.5.

The Structure of a Simple PLC

In solution shown in fig.5. a valid condition is chosen by means of an address of current state. More universal addressing a valid condition may be done directly from the program memory. It means that a condition test instruction is stored in a memory together with the condition address. The states of output signals may be set or reset in the same way. To create possibility for testing the output states and for designing sequential automata the outputs should be connected to the condition multiplexer too, as it was shown in fig.6. The condition flip flop (CFF) is set at the beginning of series of test instructions. Such arrangement supports calculation of logic product (conjunction) of binary variables - inputs and outputs for example. This way we obtain simple programmable logic controller (PLC) which may be called logic or bit processor [12,13,14].

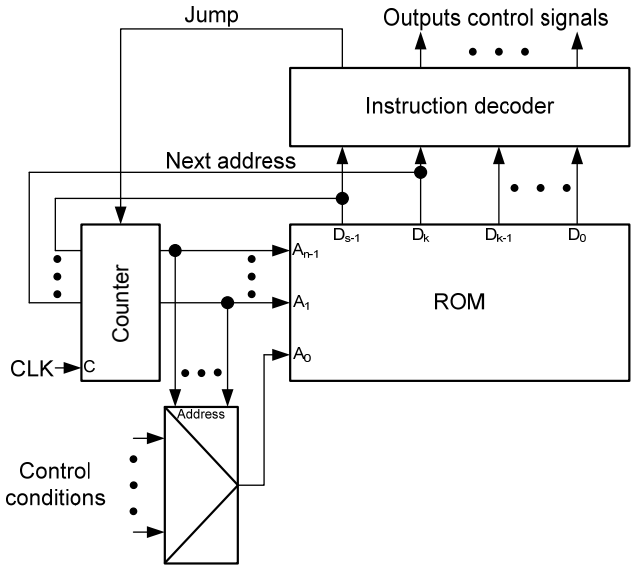

Fig.5. The control unit with program counter and instruction decoder.

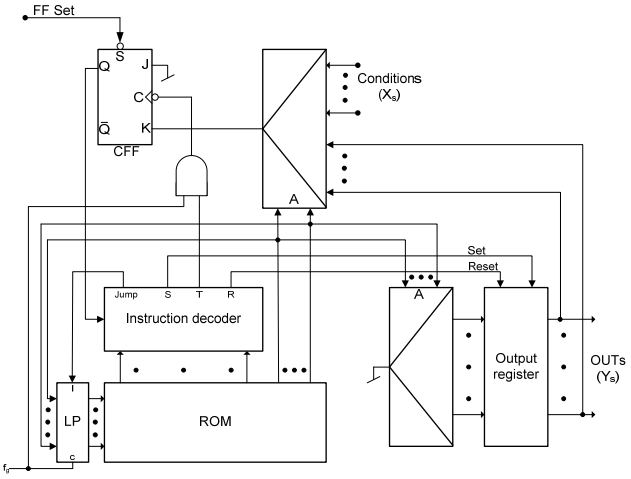

Fig.6. The structure of simple programmable logic controller – logic processor.

PLC Programming

Despite simplicity of presented logic controller its functionality is relatively high. Assuming instruction list presented below one can use it in different industrial applications for control binary objects.

Mnemonic	***Data***
Test I/O	**address**
Set	**address**
Set(CFF=1)	**address**
Set(CFF=0)	**address**
Reset	**address**
Reset(CFF=1)	**address**

Reset(CFF=0)	**address**
Jump	**program line number**
Jump(CFF=1)	**program line number**
Jump(CFF=0)	**program line number**
NOP	

Example 2

Let us use the programmable logic controller instead the relay flip flop with dominative reset (R) (see fig.7).

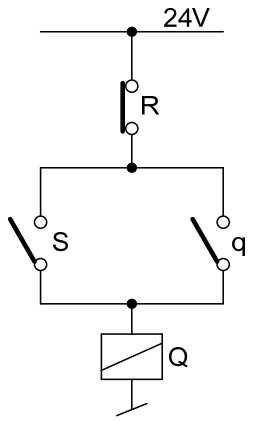

Fig.7. The relay flip flop with dominative reset.

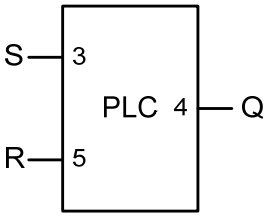

Fig.8. The exemplary addresses of PLC inputs and output.

The inputs S and R are applied to PLC pins 3 and 5 as it is shown in fig.8. The output Q at pin 4 represents the state of the relay Q from fig.7. The control program for which the PLC operates in the same way as relay flip flop has a form as follows:

0.	**Test**	**R**
1.	**Jump(CFF=0)**	**line6**
2.	**Test**	**S**
3.	**Jump(CFF=1)**	**line5**
4.	**Test**	**Q**
5.	**Set(CFF=1)**	**Q**
6.	**Reset(CFF=0)**	**Q**
7.	**Jump**	**line0**

One can notice that at line 7 is jump to line 0. It means that control program is executed in serial-cyclic way. This is typical for programmable logic controllers.

In the system the instructions are represented by the code words. For example the following coding can be used in the considered controller:

Mnemonic	***Code***
Test	**0001**
Set	**0010**
Set(CFF=1)	**0011**
Set(CFF=0)	**0100**
Reset	**0101**
Reset(CFF=1)	**0110**
Reset(CFF=0)	**0111**
Jump	**1000**
Jump(CFF=1)	**1001**
Jump(CFF=0)	**1010**
NOP	**0000**

Therefore taking into account the PLC addresses assigned to the inputs and output the program describing PLC operation as flip-flop with dominating reset (R) may be written as:

			line number	**instr. code**	**addr.**
0.	**Test**	**R**	**0000**	**0001**	**0101**
1.	**Jump(CFF=1)**	**line6**	**0001**	**1001**	**0110**
2.	**Test**	**S**	**0010**	**0001**	**0011**
3.	**Jump(CFF=1)**	**line5**	**0011**	**1001**	**0101**
4.	**Test**	**Q**	**0100**	**0001**	**0101**
5.	**Set(CFF=1)**	**Q**	**0101**	**0011**	**0101**
6.	**Reset(CFF=0)**	**Q**	**0110**	**0111**	**0101**
7.	**Jump**	**line0**	**0111**	**1000**	**0000**

At the moment let us come back to the above considered heater control. Using denotations:

H – heater
F – fun
D – fun operation sensor
R – main switch
A – fun switch
B – heater switch
C – set temperature reached

For PLC input addressing as it is seen in fig.9.

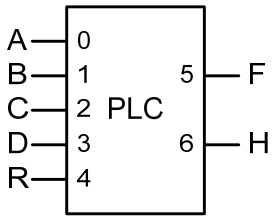

Fig.9. The exemplary PLC input addressing for heater control.

One can write the heater control program in the following form:

			line number	**instr. code**	**addr.**
0	**Test**	**R**	**00000**	**0001**	**00101**
1	**Jump(CFF=1)**	**line15**	**00001**	**1001**	**01111**
2	**Test**	**H**	**00010**	**0001**	**00110**
3	**Jump(CFF=0)**	**line10**	**00011**	**1010**	**01010**
4	**Test**	**D**	**00100**	**0001**	**00011**
5	**Reset(CFF=0)**	**H**	**00101**	**0111**	**00110**
6	**Reset(CFF=0)**	**F**	**00110**	**0111**	**00101**
7	**Test**	**C**	**00111**	**0001**	**00010**
8	**Reset(CFF=1)**	**H**	**01000**	**0110**	**00110**
9	**Reset(CFF=1)**	**F**	**01001**	**0110**	**00101**
10	**Test**	**A**	**01010**	**0001**	**00000**
11	**Set(CFF=1)**	**F**	**01011**	**0011**	**00101**
12	**Test**	**B**	**01100**	**0001**	**00010**
13	**Set(CFF=1)**	**H**	**01101**	**0011**	**00110**
14	**Jump**	**0**	**01110**	**1000**	**00000**
15	**Reset**	**H**	**01111**	**0101**	**00110**
16	**Reset**	**F**	**10000**	**0101**	**00101**
17	**Jump**	**line0**	**10001**	**1000**	**00000**

From Simple to Developed PLC

In practical applications it is necessary to delay some actions of the controller. It happens, for example, when set of conveyors are controlled (a next conveyor is typically switched on few seconds after previous one) or when a buzzer is switched on for a few seconds before machine drive is put into operation. In such application a timer have to be used. For PLC types as considered in this section a timer is hardware module with manually adjustable delay time or a value of time delay can be programmed by means *load* operation (L). Its inputs are located in PLC output area while its state is tested as a typical input. The way of timer connection to PLC structure is shown in fig.10. The instruction *set* starts a timer while instruction *reset* prepares the timer to next using. The example of timer using is presented below:

```
0.    Test          Condition
1.    Jump(CFF=0)   line3
2.    Set(CFF=1)    Timer
3.    Test          Timer
4.    Jump(CFF=0)   line7
5.    Set(CFF =1)   Output
6.    Reset         Timer
7.    ???
```

In some applications a controller should count manufactured elements or external events. To provide a controller with such functionality it should be equipped with a counter module. The value to each a counter should count may be adjusted by means a few positions tomb wheels or it may be programmed by means *load* operation (L). The connection of such counter to the considered PLC structure is shown in fig.10.

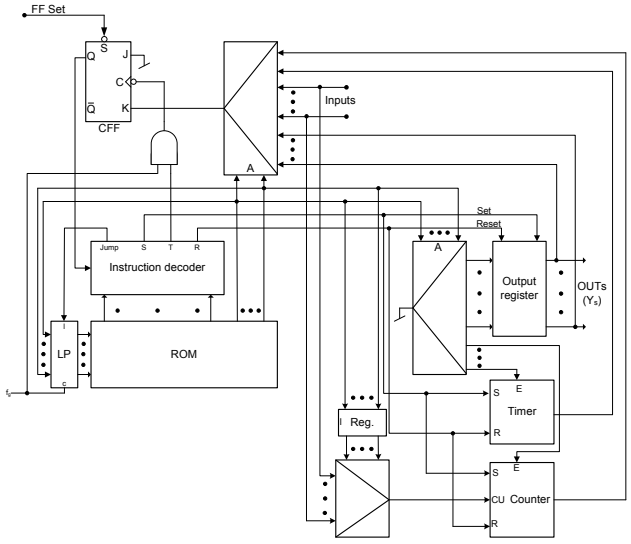

Fig.10. The PLC with a timer and counter.

As it was said before presented programmable logic controller is able to control binary objects. To control continuous (analog) objects the unit should be equipped with AD and DA converters and arithmetic logic unit (ALU) for processing numeric values. The general block diagram of such PLC is shown in fig.11. Beside load instruction (L) a few of new instructions have to be added to the instruction set for control numeric operations. These are addition (“+”), subtraction (“-“) and comparison (Comp). A numeric value is loaded always to the accumulator AC1. Next load causes that new value is written to the AC1 while previous state of AC1 is transmitted to accumulator AC2. The comparison operation compares value in AC1 to value in AC2.

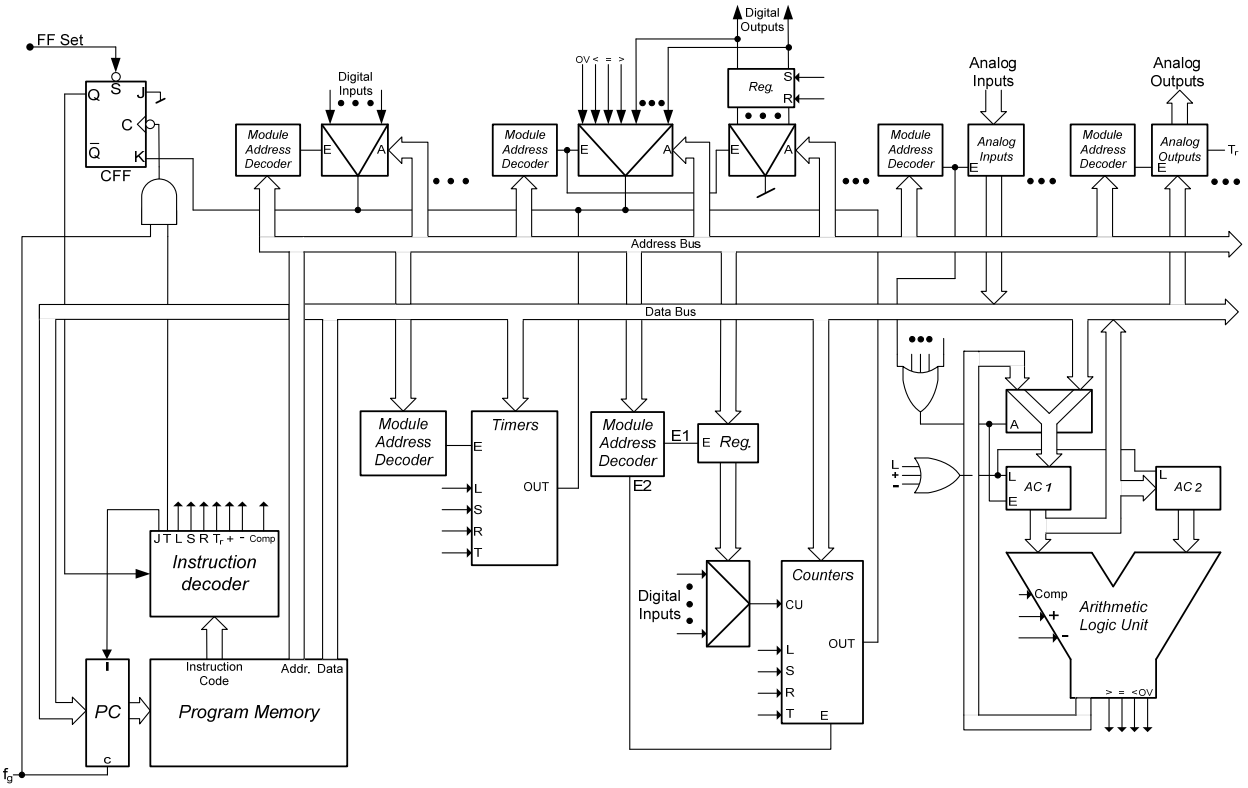

Fig.11. The whole structure of the PLC.

Example 3

Let us write control program which switch on binary output when numeric value of analog input reaches value 150 and switch off this output when the numeric value drops to value 50. It means that it is dual level control with hysteresis (see fig.12). The control program written with using the mnemonics assumed above has the following form:

```
0.    Load          IN
1.    Load          150
2.    Comp
3.    Test          AC1<AC2
4.    Reset(CFF=1)  OUT
5.    Jump(CFF=1)   line11
6.    Load          IN
7.    Load          50
8.    Comp
9.    Test          AC1>AC2
10.   Set(CFF=1)    OUT
11.   Jump          line 0
```

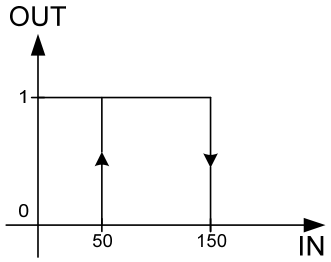

Fig.12. The analog control example.

Conclusions

In the paper the structure of programmable logic controller was described and idea of such unit programming and applications was explained on few simple examples. In fig.11. the arithmetic-logic unit was included for word operation execution as, for example, results of analogue to digital conversion comparison to given value (see example 3). In more sophisticated constructions a microprocessor/ microcontroller may be used for byte/word operation instead of arithmetic logic unit [15]. Such PLC structure is called bit-byte [16,17] because bit operations are executed by above described fast bit-processor while byte/word operations are executed in software way by microprocessor. The bit-byte structure of the PLC one can meet in number of PLCs present on the market [1,4,12]. The PLC structure can be considered as bit-byte even though the manufacturers introduced their product as microprocessor based but they usually offer special module for fast bit operations [13]. The processors in bit-byte structure of PLC can execute their operations concurrently. Very interesting problem is how to put both processors in concurrent operation as far as it is possible? More about this problem a reader can find in [18,19]. It is necessary to added that a PLC programming was standardized. The reader should refer to the standard EN/PN 61131-3 where control program representation forms are described [20]. The paper is the authors replay for questions expressed by the new PLC users who often are not electronic engineers.

Acknowledgement

The works was supported by Ministry of Science and Higher Education of Republic of Poland No, 5391/B/T02/2010/38

REFERENCES

[1] Michel G.: *Programmable Logic Controllers, Architecture and Applications.* John Wiley & Sons, West Sussex, England, 1990

[2] Broel-Plater B.: *Sterowniki programowalne, właściwości I zasady stosowania* (*Programmable Logic Controllers, Features and Rules of Application)*. Seria Tempus. Szczecin, Wydziała Elektryczny Politechniki Szczecińskiej, Poland, 2000

[3] Erickson K.T.: *Programmable Logic Controllers.* IEEE Potentials, vol.15, no.1, Feb.-March 1996, pp.14-17. Publisher: IEEE, USA, 1996

[4] Irwin J.D.: *Industrial Electronics. Handbook*. A CRC Handbook Published in Cooperation with IEEE Press 1997

[5] Kwaśniewski J.: *Programowalne sterowniki przemysłowe w systemach sterowania* (*PLC in Control Systems)*. Fundacja Dobrej Książki, Kraków, Poland, 1999

[6] Mandado E., Marcos J., Perez S.A.: *Programmable Logic Devises and Logic Controllers.* Prentice Hall, London 1996

[7] Warnock I. G.: *Programmable Controllers. Operation and Application*. Hemel Hempdtead, Prentice Hall Int., 1998

[8] Legierski T., Kasprzyk J., Wyrwał J., Hajda J.: *Programowanie sterowników PLC (Programming PLC)*. Wydawnictwo Komputerowe Jacka Skalmierskiego, Gliwice, 2008

[9] Chmiel M., Mocha J., Hrynkiewicz E., Milk A.: *Central Processing Units for PLC implementation in Virtex-4 FPGA*. Proceedings of the 18th IFAC World Congress, August 28-September 2, Milano, Italy, 2011

[10] Łuba T.:*Układy mikroprogramowane (Microprogrammed circuits)*. Oficyna Wydawnicza Politechniki Warszawskiej, Poland, 1996

[11] Molski M.: *Modułowe i mikroprogramowalne układy cyfrowe (Module and microprogrammed digital curcuits)*. WKŁ, Warszawa, Poland, 1986

[12] Getko Z.: *Programmable systems of binary control in PLC*, Elektronizacja, WKiŁ, Warszawa, 1983 (in polish)

[13] Donandt J.: *Improving response time of Programmable Logic Controllers by use of a Boolean coprocessor*, IEEE Comput. Soc. Press. 4:167-169, Washington, DC, USA, 1989

[14] Aramaki N., Shimokawa Y., Kuno S., Saitoh T., Hashimoto H.: A new Architecture for High-performance Programmable Logic Controller, Proc. of the IECON'97 23rd Inter. Conf. on Industrial Electronics, Control and Instrumentation, IEEE part vol.1, pp.187-190, New York, USA, 1997

[15] Milik A., Hrynkiewicz E.: *PID Module for Reconfigurable Logic Controller*. Preprints of the IFAC Workshop on Programmable Devices and Systems, PDeS 2000, Ostrava, Czech Republic, February 2000

[16] Chmiel M., Hrynkiewicz E.: *Remarks on Parallel Bit-Byte CPU structures of Programmable Logic Controllers*. In: Design of Embedded Control Systems, Section V, (Adamski M.A., Karatkevich A., Węgrzyn M), pp. 231-242, Springer Science+Business Media, Inc., 2005

[17] Chmiel M., Mocha J., Hrynkiewicz E.: *A FPGA-Based Bit-Word PLC CPUs Development Platform*, The International IFAC Workshop on Programmable Devices and Embedded Systems, PDeS'10, October 6-7, Pszczyna, Poland, pp. 155-160, 2010

[18] Chmiel M., Hrynkiewicz E., Milik A.: *Concurrent operation of the processors in Bit-Byte CPU of a PLC*, Preprints of the IFAC World Congress, Prague, Czech Republic, July 3-8, 2005

[19] Chmiel M., Hrynkiewicz E.: *Fast Operating Bit-Byte PLC*, Preprints of the 17th IFAC World Congress (on DVD-ROM), Seoul, Korea, July 6-11, pp. 14810-14815, 2008

[20] Norma PN-EN 61131-3:2004(U). *Programmable Controllers - Part 3; Programming Languages*, International Electrotechnical Commission, Geneva, 2004

***Authors**: dr hab. inż. Edward Hrynkiewicz, prof. Politechniki Śląskiej, dr inż. Mirosław Chmiel, Politechnika Śląska, Instytut Elektroniki, ul. Akademicka 16, 44-100 Gliwice,*
e-mail: Edward.Hrynkiewicz@polsl.pl, Miroslaw.Chmiel@polsl.pl