Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

Fast Operating Bit-Byte PLC

Mirosław Chmiel, Edward Hrynkiewicz

Institute of Electronics, Silesian University of Technology,
Akademicka 16, 44-100 Gliwice, Poland, tel. +48 32 237 14 95, fax+48 32 237 22 25
<Miroslaw.Chmiel, Edward.Hrynkiewicz>@polsl.pl

Abstract: The paper presents a few different approaches to optimizing operation speed of Programmable Logic Controllers. First approach optimizes architecture of the bit-byte CPU, second optimizes program execution, next one data exchange between bit and byte processors and the last one optimizes input/output servicing. All of them lead to a two processor's bit-byte architecture, which support of concurrent execution of bit and byte computation tasks. In such architecture the processors can operate without waiting one for the other.

1. INTRODUCTION

The Programmable Logic Controllers (PLCs) have been introduced at the end of the sixties. Nowadays the PLCs are commonly used in different areas and applications. One of basic parameters that determine performance of Programmable Logic Controllers is a time needed to execute one thousand of instructions. This is why designing and development of a unit that would enable to execute of a control program in extremely short time is becoming a very important task. Owing to its constructional features, such a unit should not only cover all the possible functional requirements but also make possible to take maximum benefits from the provided features. Trying to propose a cost-effective solution that is fast enough and assures withstanding imposed time limits, being relatively not expensive a bit-byte structure of a controller CPU should be taken into consideration (Aramaki, *et al.*, 1997, Chmiel and Hrynkiewicz, 1999; 2001; Michel, 1990). Thanks to their special operation features such structures enable achievement of satisfactory results in case of both binary signal processing, owing to inclusion of a dedicated bit processor, as well as handling with analogue signals, which is carried out by a standard, cheap micro controller. Such a structure includes two separate components: a binary (bit) unit and a byte unit. Therefore, a control program has to be subdivided into two parts. What's more, such an approach is justified by the fact that special features of instructions for the two processors shall be different. The bit processor executes every instruction during a single clock cycle, whereas for the byte processor every single instruction is equivalent to a procedure that must be developed in a certain assembler language, a high-level C-type or Pascal language (Donandt, 1989, Getko 1983).

In the papers the authors consider improving of PLC operation speed by:

- Developing a hardware structure of bit-byte CPU;

- Improving inter processors information exchange;

- Improving a way of control program execution;

- Introducing modified structure of input/output modules;

- Introducing modified structure of timers and counters modules.

2. STRUCTURE OF FAST BIT-BYTE CENTRAL PROCESSING UNIT

It is well known that in the simplest case each programmable control system might be realised as microprocessor device. But we have to remember about applications in which we are going to use a logic controller. Those applications force special requirements and constraints. To control a real object it is necessary to process a great number binary inputs and calculates binary outputs while standard microprocessors (or microcontrollers) operate mainly on bytes. Instruction list of these devices is optimised for operation on bytes or words (some of them can carry out complicated arithmetical calculation) variables that are not required in this case. Logic instructions like AND or OR on individual bits take the same amount of time as the instructions on a bytes as it is necessary to extract suitable bits from the word(s). When we take under consideration the binary inputs and outputs it is necessary to realise that they reach number of thousands. In such cases parallel computation of all inputs and outputs is impossible. Therefore the inputs and outputs must be scanned and updated sequentially. If we would like to achieve good control parameters the instructions on bits should be done very quickly. Creation of specialized bit-processor, which can carry out bit instruction very fast, is fully reasonable (Chmiel, *et al.*, 2004). If there is a need of computation of byte data for example from analogue to digital converters or external timers, it is required to use of additional 8,16 or even 32 bits processor or micro controller. General structure of that device was presented in Fig.1 (Chmiel and Hrynkiewicz. 2006a, b).

Basic parameter that was taken under consideration was program execution speed. Program execution speed is mainly limited by access latency of both processors to the internal (e.g. counters, timers) and external (e.g. inputs and outputs)

process variables. The program memory and the instruction fetching circuitry also influence system performance. In order to support cooperation of both processors without conflicts and maintain their concurrent operation following assumptions were made (Chmiel and Hrynkiewicz, 2005):

1. Separate address buses for bit and byte processors;
2. Separate data buses: 1 bit wide for bit processor and 8 bit wide for micro controller;
3. Separate control buses with:
   - read and write signals for byte processor;
   - read and write signals for bit processor;
   - refresh signal for latching up states of all inputs and outputs at once and error signal for immediate switching off of all external modules of controller.

Additionally in basic solution the following problems were taken for research and design works. These problems are structure of a memory (memories), instruction fetching by both processors, information exchange rules between processors and access to common resources (timers, counters and flags).

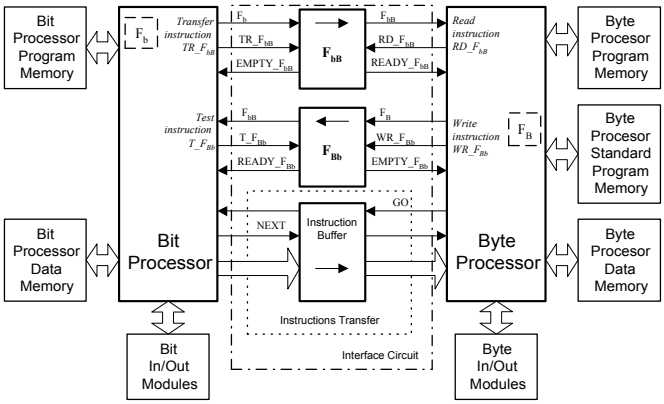

Fig. 1. Block diagram of the two-bus CPU

3. INTER PROCESSORS INFORMATION EXCHANGE

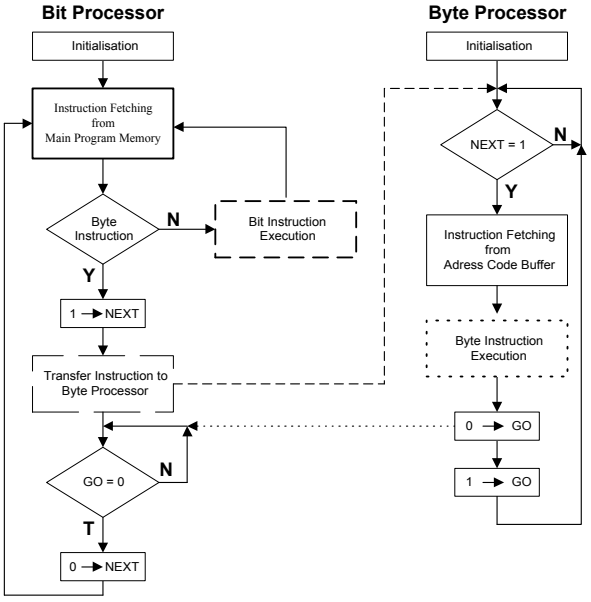

Fig. 2. Program execution in bit-byte CPU

Bit processor delivers instructions to the byte processor through the Instruction Buffer informing about it by means of binary signal (NEXT). On the other hand byte processor after accepting of an instruction sends to the bit processor other binary signal (GO) – Fig.2 (Chmiel and Hrynkiewicz, 2006a).

Such exchanging of the conditional flags doesn't require postponing of program execution. Proper data transfer is maintained by handshake registers that controls data flow among processors and also synchronize program execution. Always when condition result is passed from one processor to another pair of instruction must be executed. When data is passed from the bit processor to the byte processor those are TR_F_{bB} and RD_F_{bB}. Transfer in opposite direction requires execution of instructions WR_F_{Bb} and T_F_{Bb}. In Fig.3 is presented inter-processors condition passing algorithms (Chmiel and Hrynkiewicz, 2006b).

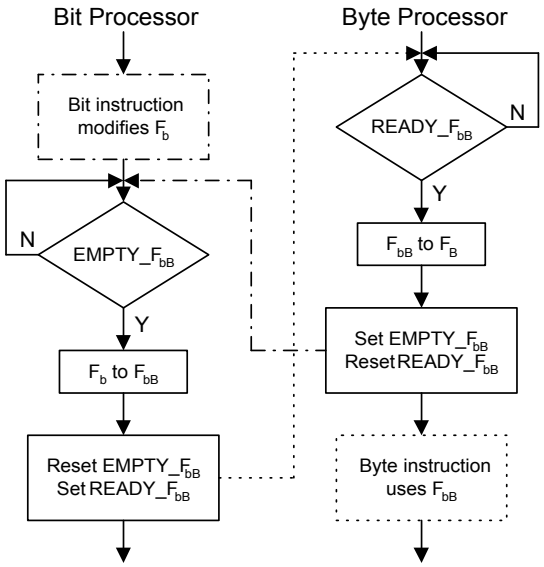

Fig. 3. Example of bit to byte condition passing algorithm

4. PROGRAM EXECUTION

Let us take into account the control program shown in Fig.4. The program consists of different kinds of instructions:

- Bit instructions stored in main program memory executed usually in one clock cycle;

- Byte instruction that requires byte processor program memory access (type I);

- Complex instructions that require processor co-operation while result of operation depends on both processor calculations performed in proper order (type II).

It can be suggested that three following byte instructions can be initiated once by bit processor. This approach allows for shortening of an instruction execution time. The same optimisation can be applied to next two byte instructions. Proposed modification results in shortening of main program. Execution time of byte instruction group is also reduced, as

they are now stored in byte processor memory. Overall execution time of program loop is also reduced.

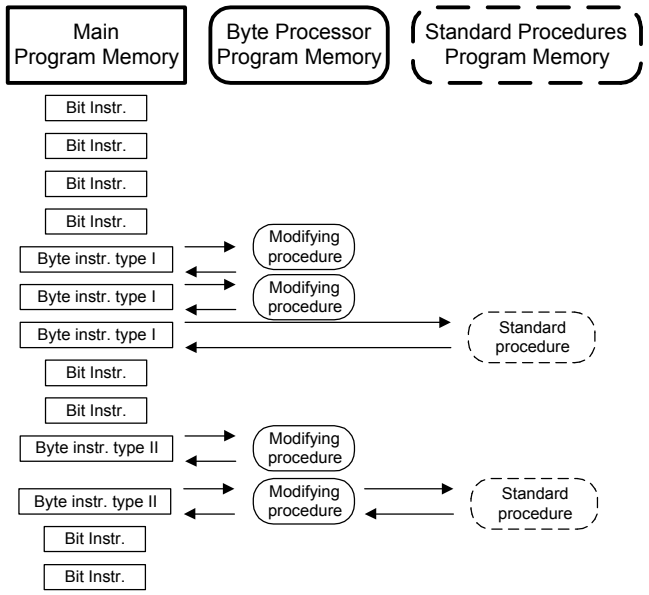

Fig. 4. Instruction allocation case 1

Additionally it can be noticed that first two byte instructions are independent from following bit instructions. As those instructions are independent from each other they can be executed concurrently with these bit instructions. Following situation was considered in Fig.5. As it is presented execution time was radically reduced by concurrent operation of bit and byte processors. There are also operations those result influences on operation of opposite processor. This is very important problem of data exchange between processors.

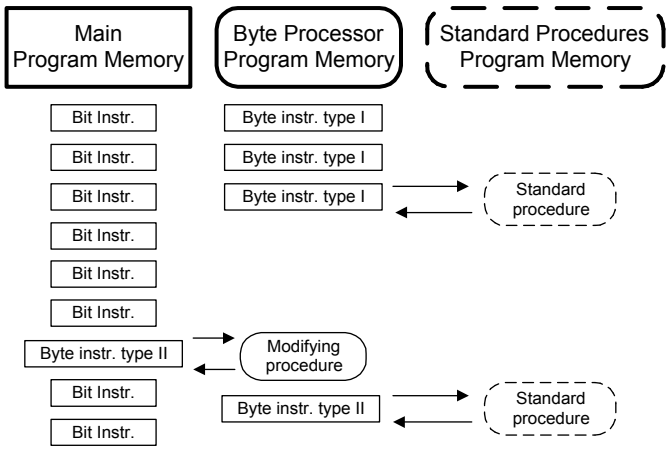

Fig. 5. Instruction allocation case 2

As it was shown the studies on an information exchange between the processors of the bit-byte CPU of a PLC leads to the CPU hardware/software solution which significantly increases a program execution speed. The most interesting result is the possibility of fully parallel work of both processors without waiting one for the other. Such mode of CPU operation become possible thanks to realising that for considered processor the other processor can be treated in the same way as a controlled object.

Let us consider the block diagram of the CPU from Fig.1 in which the instruction buffer was removed while each processor owns instruction memory. Processors are synchronised by state of condition flip-flops F_{Bb} and F_{bB}. Instructions to the processors are delivered from its local memory so that they don't have to wait until instruction is delivered from common program memory. Processors enter wait state only when they attempt to read empty condition register or overwrite not read condition in register. Program execution is synchronised by conditional execution of program fragment that depends on result delivered from opposite processor.

In order to avoid of long wait states program should be written and compiled in that way that load of both processors is equally distributed. Further optimisation can be achieved by increasing number of flip-flops that pass logic condition between processors or implementation of common memory area used for information exchange.

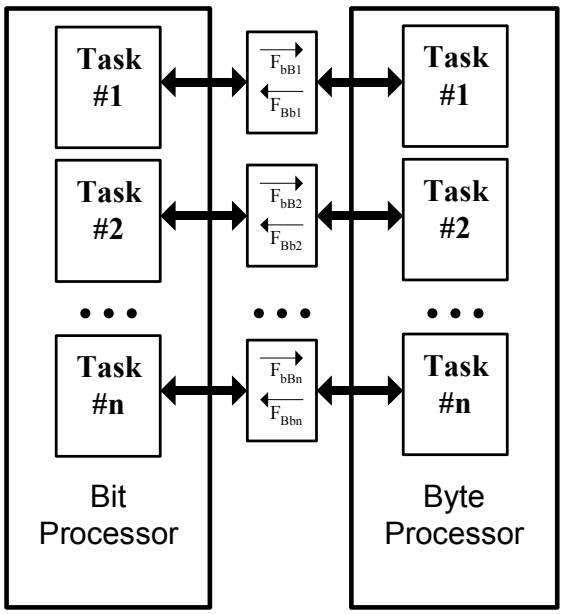

Fig. 6. The idea of multitasks control program execution

A program consists of two parts – one for bit and the other for byte processor, the both parts of control program may be divided in number of mutually independent tasks with respect to data exchange between the processor units. For each task a pair of condition flip-flops should be assigned so as the processors have not to wait for emptying of condition flip-flop during control program execution. This way we obtain an effect equivalent to two smaller programs processed in a quasi-concurrent manner. From the point of view of the interprocessors information exchange such an operation of the PLC would be fully parallel. If required, however, operation of any of the processor could be suspended to wait for the other unit, but only within a specific task – the second task could be further processed. The idea is schematically explained on Fig.6. The diagram shows a case, in which we

have *n* tasks, mutually independent. Every of the units processes *n* tasks, which require mutual data exchange. Besides, every of the units can process any number of additional tasks, which don't require information exchange.

Preserving serial program execution by each processor cause that conditions are generated in the same order. That allows for designing specific hardware with extremely fast access to condition bits. The solution is based on a set of D flip-flops or dual ports memory locations. Their content is written by one of the processors while second reads its contents. The discussed approach eliminates conflicts and reduces access time.

It can be noticed that presented solution allows for unconstrained operation of both processor while processors doesn't have to wait for condition passing. In case of disproportion of program execution time in processors some flags are updated and examined more frequently than the other (Chmiel, *et al.*, 2005; Chmiel and Hrynkiewicz, 2005).

It seems, that the problems, if they could ever reveal, would concern situations of large discrepancies between processor scan times – one of the units reads and writes back flip-flop states several times faster, than the other. It means that during control program loop executed by the one processor the second one may change a state of his condition flip-flop for a few times. It causes that the parts of control program executed by the first processor utilise the different states of the condition flip-flop. Rule of serial program execution is violated. Both processors operate asynchronously to each other. They observe markers area like an I/O space - in order to perform possible for execution actions in response for changing markers.

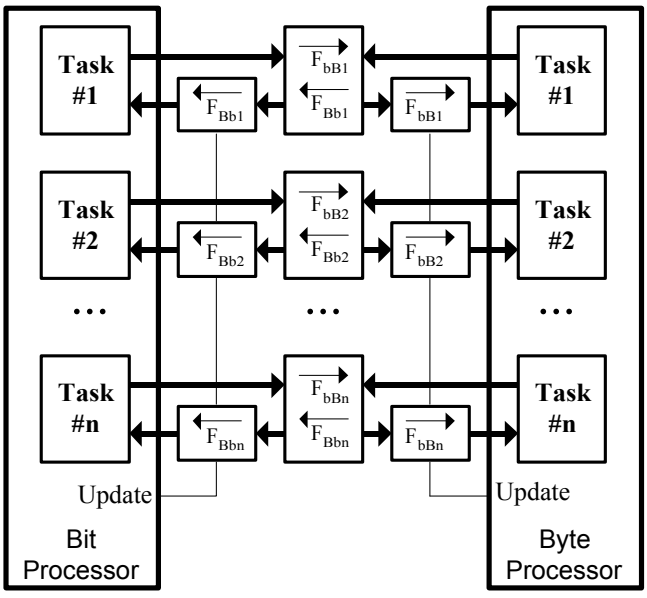

Fig. 7. CPU block diagram for multitasks control program with auxiliary buffer

I/O devices, from the point of view of a CPU, work slowly. It is however possible, that during one program scan a state of input changes several times. In such a case the input changes will not be registered. Now similar problems arise inside our CPU, with data exchange between the processor units. To avoid the situation, that one of the unit's processes different tasks with different states of one of the condition flip-flops, an additional buffering should be applied. The additional buffering consists in organising condition bit data exchange in such a way, that every of the processors performs writes to its own condition register at random moments, while the other would decide, when the data should be copied to its own memory. The copying would be organised in a similar manner, as input scanning and output updating. Such a mechanism would cause, that the data stored in the additional register would constitute for both processors the same set of signals (with respect to the data access mechanism), as ordinary I/Os. I/O states can also change during a program scan, but the processor is capable of acquiring this information only before the next program scan or, alternatively, the information is lost, if the changes last to short. Every of the processor units would refresh from its side the data stored in the condition flip-flops, while the other unit, after completing a program scan, by generating an "update" signal, would initiate copying of the condition flip-flop states to its own image memory. The copying would be performed together with input scanning and PII updating. The idea of the condition flip-flop register with auxiliary buffer is presented in Fig.7.

5. TIMERS AND COUNTERS SERVICING

In separate way can be considered timers and counters which from their nature are located on the point of function of both parts of presented PLC CPU. They are implemented as the software modules for microprocessor but in most cases are utilized by bit processor. An operation of timers and counters may be organized on the base of described above mechanism of condition exchange. To making the operation of system faster the special instruction for timers and counters should be created. These instructions should involve in timer and counter servicing both processors. Bit processor sets or resets status bit of given timer/counter while byte processor completes their functions. Bit processor executes its control program without waiting for finishing of an operation of byte processor. The status of timers and counters is stored in data memory of byte processor. The copy of status is sent to special buffer register which outputs are located in the bit processor discrete input area. Thank to this arrangement checking of each timer or counter state occupies one clock period. The structure which make possible an access to the timers and counters for both processors independent is shown in Fig.8 (Chmiel, 2006).

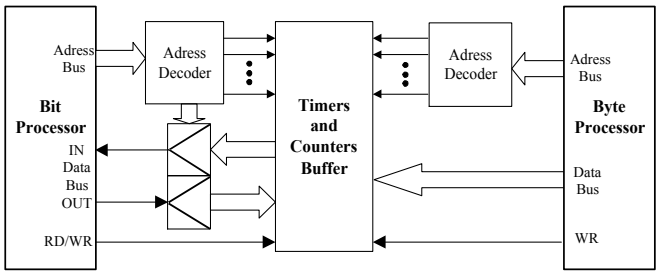

Fig. 8. Hybrid Timer and Counter function realisation

6. PROCESS INPUTS/OUTPUTS SERVICING

The way of microprocessors access to peripheral modules has an important influence on the operating speed of a controller.

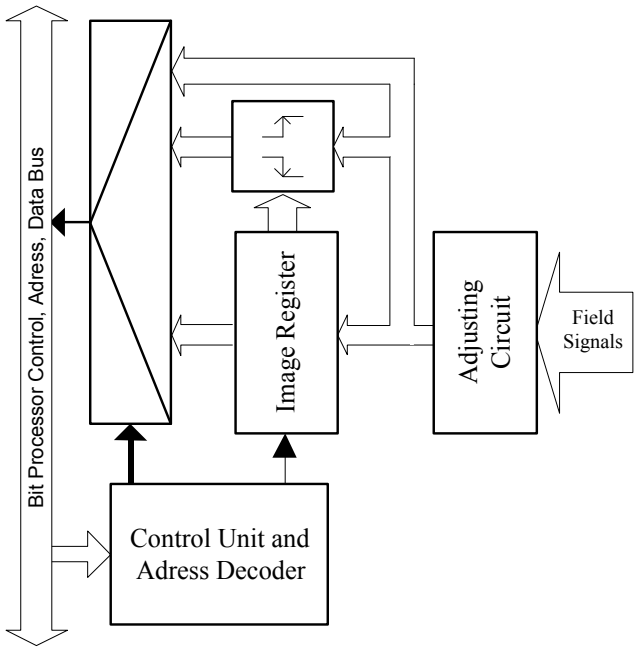

Fig. 9. Binary Inputs Module Schematic

One approach is to use a dedicated microprocessor that is responsible for updating state of the inputs and outputs of the micro controller. In the presented paper, however, some other solution has been applied that does not require the above-mentioned dedicated microprocessor. In addition, it should speed up operation of the central unit of the micro controller. Input and output units are equipped with special mechanisms that facilitate creation of process image input and output (PII, PIQ) directly in the units. There are two ways the central unit could determine the rhythm of updating the process image input and output: after each program scan execution and when requested by the current command. So, depending on the type of the command, one is able to read the input state that has been stored at the beginning of the current program loop or to read the current state of the input.

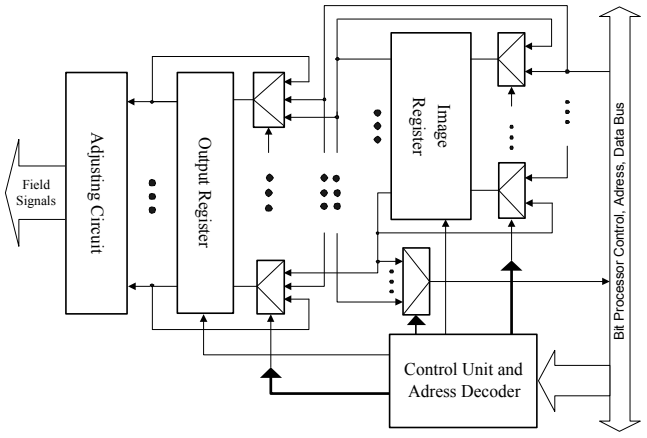

Fig. 10. Binary Outputs Module Schematic

The state of the output can be stored in the process image register and the state of the latter is then copied to the output register at the end of the current iteration of the program loop. One can also directly set a new value at the given output what immediately updates the sate of the process image output register. The units are equipped with edge detection mechanisms what additionally reduces time required for program execution. Block diagrams of binary I/O units are presented in Fig.9 and Fig.10. For the sake of legibility, the edge detection circuit is not shown in the schematic of the output unit (Chmiel, 2006).

6. CONCLUSIONS

As it was shown the studies on an information exchange between the processors of bit-byte CPU of a PLC and on a way of control program execution lead to the CPU hardware solution which significantly increases a program execution speed. The most interesting result is the possibility of fully parallel work of both processors without waiting one for the other. Such mode of CPU operation become possible thanks to realising that for considered processor the other processor can be treated in the same way as a controlled object. Finally block diagram of PLC Central Processing Unit and peripheral modules are presented in Fig.11.

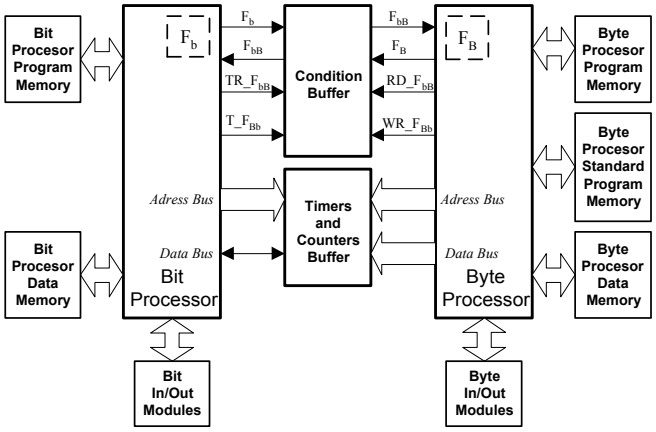

Fig. 11. Block diagram of bit-byte PLC CPU

At this moment we have investigated in practical applications the ideas presented in Fig.1, Fig.5, Fig.8, Fig.9 and Fig.10. The comparison of control program execution time for ignition burners in steel plane furnace for different industry controllers and the controller built on base of these ideas is listed in Table1. Serial mode means that one processor waits for the other after each condition exchange while the parallel mode means that the processors work in concurrent way as far as it is possible exploiting all above mentioned ideas. The comparison was performed for the circuit that contained only the CPU, to avoid influence of the time necessary for communication with I/O modules. The scan times don't contain the "empty" loop scan time, either. This enabled avoiding influence of execution time of system functions. The program used as the example was a part of a practical application – the control system for a metal sheet pickling line at a Columbus Stainless ironworks (South Africa) (Chmiel, *et al.*, 1997).

Table 1. Control program execution time comparison for the few PLCs

PLC	Nr of bit instr.	Nr of byte instr.	Execution time [ms]
S5-100U	1030	140	9.0
S5-115U	1030	140	1.9
S7-224	780	50	2.8
S7-313	1000	115	2.7
S7-315	1000	115	1.5
Modicon A984	-	-	8.0
Serial mode	1050	280	1.7
Parallel mode	850	140	1.1

REFERENCES

Aramaki, N., Y. Shimokawa, S. Kuno, T. Saitoh, H. Hashimoto (1997), „A new Architecture for High-performance Programmable Logic Controller", Proceedings of the IECON'97 23rd International Conference on Industrial Electronics, Control and Instrumentation, IEEE part vol.1, pp.187-190, New York, USA.

Chmiel, M. (2006). "A Response Time of the PLCs Reducing", Krajowa Konferencja Elektroniki, KKE'06, Darłówko Wschodnie Poland, vol.1, pp.161-166 (in polish)

Chmiel, M., E. Hrynkiewicz (1999). „Parallel Bit-Byte CPU Structures of Programmable Logic Controllers", International Workshop ECMS, Liberec, Czech Republic.

Chmiel, M., E. Hrynkiewicz (2001). „Remarks on Parallel Bit-Byte CPU Structures of Programmable Logic Controllers" International Workshop on Discrete Event System Design, DESDes, Przytok near Zielona Góra, Poland.

Chmiel, M., E. Hrynkiewicz (2005). *Design of Embedded Control Systems*, Section V, (Adamski M. A., A. Karatkevich, M. Węgrzyn), pp.231-242, Springer Science + Business Media, Inc.

Chmiel, M., E. Hrynkiewicz (2006a). „Improving of Concurrent Operation of the Bit-Byte PLC CPU", International Conference on Programmable Devices and Systems, PDeS'06, pp.15-20, Brno, Czech Republic, February 14-17.

Chmiel, M., E. Hrynkiewicz (2006b). „How to Reduce a Response Time of the PLCs", The 7th International Conference on Technical Informatics, ConTI'06, Timisoara, Romania, 8-9 June, vol. 2, pp.95-100.

Chmiel, M., E. Hrynkiewicz, A. Milik (2004). „Remarks on Programming of a Bit Processor Used in Bit-Byte CPU of a PLC", The International Workshop on Discrete-Event System Design, DESDes'04, September 15-17, Dychów near Zielona Góra, Poland, pp.129-133.

Chmiel, M., E. Hrynkiewicz, A. Milik (2005). „Concurrent operation of the processors in Bit-Byte CPU of a PLC", Preprints of the IFAC World Congress, Prague, Czech Republic, July 3-8.

Chmiel M., A. Malcher, A. Nowara, „A Control System for a Metal Sheet Pickling Line" (in polish), Maszyny Technologie Materiały, Sigma 1997.

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

Getko, Z. (1983). „Programmable systems of binary control", Elektronizacja, WKiŁ, Warsaw, Poland, (in polish).

Michel, G. (1990), Programmable Logic Controllers, Architecture and Applications, John Wiley & Sons, West Sussex, England.