The International Workshop
on Discrete-Event System Design, DESDes'01,
June 27÷29, 2001; Przytok near Zielona Gora, Poland

REMARKS ON PARALLEL BIT-BYTE CPU STRUCTURES OF PROGRAMMABLE LOGIC CONTROLLERS

Mirosław CHMIEL, Edward HRYNKIEWICZ

Institute of Electronics, Silesian University of Technology, Akademicka 16
44-100 Gliwice, Poland, tel. (+48 32) 2371316, (+48 32) 2372461
e-mail: *chmiel@boss.iele.polsl.gliwice.pl*, *eh@boss.iele.polsl.gliwice.pl*

Abstract. *The paper presents some hardware solutions for the bit-byte CPU of a PLC, which are oriented for maximum optimisation of data exchange between the CPU processors. The optimisation intends to maximum utilisation of the possibilities given by the two-processor architecture of the CPUs. The key point is preserving high speed of instruction processing by the bit processor, and high functionality of the byte processor. The optimal structure should enable the processors to work concurrently for as much of the tome as possible, and minimise the situations, when one processor has to wait for the other.*

Key Words. *CPU, Logic Control, PLC*

1. INTRODUCTION

One of the main parameters (features) of Programmable Logic Controllers (PLC) is execution time of one thousand of control commands. This parameter evaluates the quality of PLC. Due to this fact it is important to design and construct a CPU with a structure enabling fast control program execution. The most developed CPUs of PLCs of many well-known manufacturers are constructed as the multiprocessor units. Particular processors in such units execute the commission for them tasks. In this way, we obtain a unit, which makes possible parallel operating of several processors. For such CPU the main problem is the way of task assuming to particular processors and finding a structure of CPU capable to solve such assigning task in practice.

The bit-byte structure of CPU, in which task assignment is predefined, are often met in real solutions. The tasks operating on discrete input/outputs are executed by bit-processor. Such processor may be implemented in programmable structures as PLDs or FPGA [3,4]. It makes the positive effects on user programme execution time (fast operating processor). On the other hand a byte-processor (word-processor) is built on the base of standard microprocessors or embedded microcontrollers. The byte processors are used for control of analogue objects, for numeric data processing and for execution of the operations indirectly connected to user (control) program but connected to the operating system of the programmable controller of a

CPU. Set of such operations consists of timer servicing, reading-out of the input states, setting of the outputs, LAN servicing, communication to the personal computer and so on.

Very interesting problem and difficult for realisation in programmable controller is timer module [5]. A time interval is counted, asynchronously to the program loop execution. It causes difficulties with testing of an end of a counted interval. At long time of program loop execution and short counted time intervals serious errors may occur. The accuracy of time intervals counting may be increased by special program tricks but achieving good results is typically connected to prolongation of control program loop. In some programmable controllers the end of time interval counting interrupts the control program and the service procedure of this interrupt is called however the number of interrupts is typically limited and only a few timers can act in this way. That is why it would be worth to reflect on the way of improving of an accuracy of time counting in the programmable controllers. Another problem is related with this matter. As it was mentioned above the scan time is one of the most important parameters of programmable controllers. However it seems that throughput time more precisely describe the dynamic features of a programmable controller. Naturally it may be said that throughput time is closely linked to the scan time unless a programmable controller does not execute a control program in serial cyclic way. Let us imagine a programmable controller, which operates on the rule based on processing of the segments (tasks) of the control program. These segments are triggered only by the changes of the input signals (input conditions). In this situation one can talk about throughput time (response time) but it would be difficult to talk about the time of program loop execution. It would be only possible to define the mean time of program loop execution for given application. For the application where the signals change sparsely the mean time of program loop execution will be much less than the maximum time evaluated for execution of a whole program. In particular applications the certain group of signals may do changes more often the other signals. The segments of the control program triggered by these signals will be executed often than the other segments. To avoid situation where two or more segments are triggered at the same moment it would be necessary to assign the priorities to the control program segments. The described method of programmable controller operation changes the approach to the preparing of control program but it seems to the authors that in such programmable controller the problems with for example timers will be easier. It is not necessary to observe the moment when time interval will be completed. At the end of the time interval counting the suitable segment may be called and executed. It means that the currently executed program segment should be interrupted. It depends on the priorities assigned to the particular program segments.

Such type of programmable controllers CPUs will be the subject of the future work while in this paper the few proposals of programmable controller bit-byte CPU structures are presented. These are CPUs with serial-cyclic program execution but they are structurally prepared to event-triggered operation.

2. THE REQUIREMENTS FOR PROGRAMMABLE CONTROLLER CPUs

The aim of the work which results are described in the paper was design and implementation of programmable controller CPU based on bit-byte structure. The main design condition was maximum speed of control program execution. This condition should be met rather by elaborating of suitable structure than application the fastest microprocessors. Additionally it was assumed that bit processor will be implemented using catalogue logic devices or programmable structures while as the byte processor will be used the microcontroller 80C320 from Dallas Semiconductor. The CPU should be capable of carrying-out of logic and

arithmetic operations, conditional and unconditional jumps, test states of the inputs, set or reset outputs, timers, counters, and so on.

In the simplest case each programmable control circuit might be realised as microprocessor device. We have to remember about application in which we are going to use constructed logic controller. Those applications force special requirements and constraints. Controlled objects heave a great number of binary inputs and outputs while standard microprocessor (or microcontroller) operate mainly on bytes. Instruction list of those devices is optimised for operation on bytes or words (some of them can carry out complicated arithmetical calculation) variables that are not required in industrial applications. Each task is connected with reading external data, computation and writing computed data to the outputs. Logical instructions like AND or OR on individual bits take the same amount of time. When we take under consideration number of binary inputs and outputs, those in greater units reach number of thousands. In such cases parallel computation of all inputs and outputs is impossible. In this situation all inputs and outputs must be scanned and update sequentially as fast as it is possible. If we would like to achieve good control parameters bits operation should be done very quickly.

Creation of specialised bit processor, which fast can carry out bit operations is fully excused. 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 additional 8, 16 or even 32 bits processor or microcontroller. General structure of that device was presented in [1].

Presented solution consists of two processors. Each of them has its own instruction set. Instruction decoder recognises for which processor instruction was fetched and sends activation signal to it.

Basic parameter that was taken under consideration was program execution speed. Following assumption were made in order to support two processors operation

- Separate address buses for bit and byte processors;
- Two data buses: 1 bit wide for bit processor and 8 bit wide for microcontroller;
- Two control buses with signals RD and WR of microcontroller, IORD and IOWR of bit processor, REFRESH (latches state of all inputs and outputs at once) and ERROR (immediate switch off of all external modules of controller).

3. SELECTED STRUCTURES OF BIT-BYTE CENTRAL PROCESSING UNIT

In this paragraph different conception rules of co-operation bit and byte processor are presented, that allow achieving maximal execution speed by logic controller.

The most typical solution is a circuit with separate program and data memories for both processors. There is also common area of memory through which processors exchange information between them in order to:

- exchange data;
- set and clear flags that request execution of specific tasks instead of exchanging whole instruction.

Other conception is presented in [2]. It based on similar idea as previously presented. This solution assumes common program memory for both processors. Each of them has unique operation codes. One of the processors fetches operation code and recognise it. If fetched

instruction is assigned to it, it is immediately executed; in other cases it is send to the second processor for execution.

The unit is equipped with 3 memory banks for control program:

- main memory;
- standard procedures memory;
- program memory for byte processor.

Such CPU has three states of operation:

- both processors execute control program;
- one processor operates;
- bit processor executes control program while byte processor actualises the timers.

The modification of the above solution, referring to the first conception is the unit where bit processor generates pulses activating the sequential tasks in byte processor. These tasks are stored in suitable areas of byte processor memory.

Finally the CPU structure presented in work [2] was accepted. This structure was additionally equipped with the system of fast data exchange keeping easy way of PLC programming. This system – in simple words – causes that the processors do not wait for finishing their operations but they execute next commands up to the moment when command of waiting for result of operation carried-out by the second processor. The important thing is the suitable program compiler and the way of control program by the user.

Bit processor delivers commands to the byte processor through the command buffer informing about it by means of NEXT=0- signal. On the other hand byte processor after accepting of a command sends to the bit processor EMPTYBUF signal (Fig.1). The processors may transfer one to other the result of recently executed operations through F1B and F2B flip-flop.

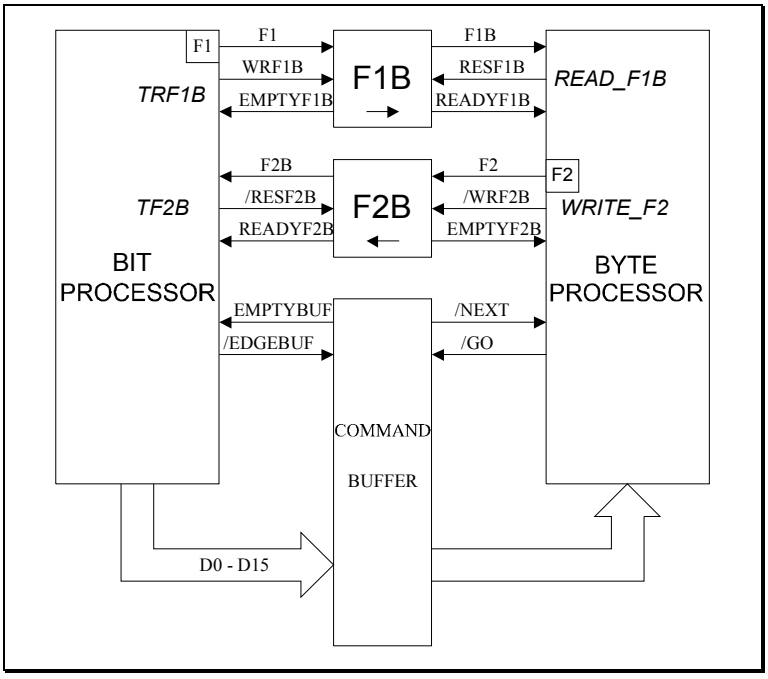

Fig. 1. The block diagram of the CPU based on exchanging of the flags and commands

Two situations cause that the speed of data exchange goes down:

- one processor has not yet execute operation expected by the second processor and this one have to wait for the result (READYF2B=0 or READYF1B=0);
- second processor has not yet received the previous result and the first one can not write the next result (EMPTYF1B=0 or EMPTYF2B=0).

To exclude waiting states the programs has to be written and compiled in such a way to get these two processor working possibly parallel. However in the second case one can take into account the solution basing on increased number of the accessible data exchange flip-flops or on assignment of common memory area for the data exchange purpose.

At that time appears the need for assignment of the marker to every task. One can try to solve the marker problem in the following ways:

- the fixed marker can be assigned to every type (kind) of operation, e.g. comparison instruction (of the byte processor) have to set the marker at the given (particular) address, and, say, the counter increment instruction will use the marker of other address (this solution is not flexible, both processors need for frequent access to the common memory area, or many flip-flops have to be used);
- the successive tasks will use successive markers and this process will repeat periodically after the number of markers is run out. The assignment process can be led automatically by the compiler. However this solution can be applied for the instruction sequences not disturbed by jumps (except the jump to the beginning of program loop).
- the third way is to charge the programmer with the duty of marker assignment. Marker has to contain the operation result, or condition has to be read from that marker. In similar way the Modicon PLCs are programmed where instruction blocks outputs carrying results can be assigned to the marker by programmer himself.

4. SYNCHRONISATION OF PROCESSORS

The designed CPU can work in one of two modes:

- dependent work – the parallel – serial work of processors with transferring of all the necessary data, co-ordinated by the bit processor which is faster. It is basic work mode of the designed CPU. All the mechanisms described for the solution of Fig.1 are made use of;
- independent mode – fully parallel. Both units work fully independent, each one has its own program so there is no waiting for the instruction transfers. There are no data transfers between the two processors. Unfortunately such a mode is applicable only some particular control programs.

5. CONCLUSION

Studies on the information exchange optimisation between the processors of the bit-byte CPU of the PLC have shown the great capabilities and many possible applications of this architecture.

When considering many ways of optimal application of this architecture it seems that quite serious problem is lying in some kind of accepted standard, which describes the way the CPU of PLC is executing the control program should be looked at.

One should go farther in such considerations and try to solve the program execution method taking more task - oriented (problem - oriented) rather than serial – periodical approach.

It seems that probably better results can be obtained when PLC is assumed an event – dependent block (module), which executes particular precisely determined tasks in response to particular constraints i.e. particular elements change of state.

In CPU of that type many problems will be connected with change scanning, because not only inputs and outputs but also markers, timers and counters should be scanned. Other problems will be related to continuous signals.

It seems however that the described architecture "enriched" with an event – dependent control program execution is quite interesting solution of the CPU for PLC.

REFERENCES

[1] Z. Getko, „Programmable systems of binary control", *Elektronizacja*, Zeszyt 18, WKiŁ, Warszawa 1983, (in Polish)

[2] M. Chmiel, E. Hrynkiewicz, „Programmable controllers", *Pomiary Automatyka Kontrola*, No.5, 1994 (in Polish)

[3] M. Chmiel, L. Drewniok, E. Hrynkiewicz, „Single board PLC Based on PLDs", *International Conference on Programmable Devices and Systems*, Gliwice, Poland. November 1995

[4] E. Hrynkiewicz, „Based on PLDs Programmable Logic Controller with Remote I/O Groups", *Third Workshop on Electronic Control and Measuring Systems*, Toulouse, France. June 1997

[5] M. Chmiel, W. Ciążyński, A. Nowara, „Timers and Counters Applied in PLCs", *International Conference on Programmable Devices and Systems*, Gliwice, Poland. November 1995