Computer Science • Vol. 8 Special Edition • 2007

Adam Piłat*, Wojciech Grega*

HARDWARE AND SOFTWARE ARCHITECTURES FOR RECONFIGURABLE TIME-CRITICAL CONTROL TASKS

The most popular configuration of the controlled laboratory test-rigs is the personal computer (PC) equipped with the I/O board. The dedicated software components allows to conduct a wide range of user-defined tasks. The typical configuration functionality can be customized by PC hardware components and their programmable reconfiguration. The next step in the automatic control system design is the embedded solution. Usually, the design process of the embedded control system is supported by the high-level software. The dedicated programming tools support multitasking property of the microcontroller by selection of different sampling frequencies of algorithm blocks. In this case the multi-layer and multitasking control strategy can be realized on the chip. The proposed solutions implement rapid prototyping approach. The available toolkits and device drivers integrate system-level design environment and the real-time application software, transferring the functionality of MATLAB/Simulink programs to PCs or microcontrolers application environment.

Keywords: *execution profiling, real-time control, rapid prototyping*

ARCHITEKTURY SPRZĘTOWE I PROGRAMOWE DLA REKONFIGUROWALNYCH KRYTYCZNYCH CZASOWO ZADAŃ STEROWANIA

Najbardziej popularną konfiguracją wykorzystywaną do sterowania systemami laboratoryjnymi jest komputer klasy PC, wyposażony w kartę wejść/wyjść. Dedykowane oprogramowanie pozwala na realizację wielu zadań definiowanych przez użytkownika. Typowa funkcjonalność karty może być dostosowana w sposób programowy do potrzeb użytkownika. Ta platforma sprzętowo-programowa pozwala na opracowanie i zweryfikowanie działania algorytmów sterowania. Kolejnym krokiem w projektowaniu układu sterowania jest wykorzystanie docelowego systemu dedykowanego. Dostępne oprogramowanie i sterowniki urządzeń pozwalają na wykorzystanie funkcjonalności pakietu MATLAB/Simulink podczas uruchomienia algorytmów na PC lub w jednostce mikrokontrolera.

Słowa kluczowe: *diagnoza wykonania zadań, sterowanie w czasie rzeczywistym, szybkie prototypowanie*

* AGH University of Science and Technology, Kraków, Poland, ap@ia.agh.edu.pl, wgr@ia.agh.edu.pl

1. Introduction

Digital control and monitoring are typical industrial real-time computer implementations. They use a variety of technological platforms as PCs, PLCs and microcontrollers. The hardware architecture is represented by a number of real-time tasks defined inside of a real-time operating system implemented on the control computer. In practical applications the control or monitoring algorithm is coded and runs as a real-time task under control of the real-time operating system. The task is failed if the real-time competitions are not completed in the time period defined by the event and the deadline relative to the event. The real-time performance of the control programs is affected by complexity of the control algorithm, i.e. more complex algorithm will require more time to ensure that all the calculations required to implement the algorithm are finished within one control interval. For systems reacting at signals from external environment we have one additional design parameter, it is a discretisation step – usually related to the sampling period. Except some special cases [1] shorter sampling time results in a higher performance of the control, i.e. model response better suits to the plant data. This leads to well-known conclusion: the real-time characteristics of controllers must be compatible with the timing constraints of the applications. Using the multitasking property of the real-time operating systems the controller can realize several tasks with different sampling or control frequency. The problem of selecting the control/sampling task frequencies to optimize the system control performance subject to real-time environment constraints was reviewed by Magalhaes [2] and lately addressed by Seto [3] and Grega [4]. In Seto formulation each task was characterised by a Performance Loss Index as a function of sampling frequency. If the performance of the system decreases as a result of a slow-down beyond limits of the real-time system it is desirable to extend the hardware architecture by introducing a multiprocessor architecture or user-defined, reconfigurable FPGA core. This approach should be followed by a proper distribution of the tasks between software and hardware layer. As real-time systems are typically interacting at a low level with physical hardware, it is reasonable to implement in FPGA typical low level functions, like A/D conversion or PWM signal generation.

The following real-time architectures are discussed in the next sections:

- personal computer equipped with FPGA based I/O board,
- embedded system exemplified by MPC565 microcontroller.

Both platforms implement rapid prototyping approach. The available toolkits and device drivers integrate system-level design environment and the real-time application software, transferring the functionality of MATLAB/Simulink programs to PCs or microcontrolers application environment. The general concept that comes within the scope of the code generator is as follows [5]. MATLAB tools are used to design, test and analyse a model of a control system. Once the project is complete the appropriate input/output drivers are added to generate an executable code directly from the Simulink block diagram. The code generator software translates the controller

diagram into its equivalent code representation, adding functions that are unique to the target controller. External mode operation of Simulink allows on-line tuning parameters of target controller. In the paper the basic characteristics of the hardware components and real-time systems are given for both system architectures. The operating systems are described in the context of digital realization of control loop. The method of monitoring of real-time tasks of the operating system for embedded controller is proposed. The control architectures were tested and compared using the laboratory magnetic levitation test rig [6, 7]. The magnetic levitation control can be categorized as time critical process: any slow-down of the real-time operation results in unstable behavior of the process.

2. Hardware and software components

2.1. Personal Computer

The most popular configuration of the controlled laboratory test-rigs is the PC equipped with I/O board. The dedicated software components allows to conduct a wide range of user-defined tasks. The typical configuration functionality can be customized by PC hardware components and their programmable reconfiguration.

The tested solution is based on two processors hardware architecture and Windows NT technology. Processor-intensive applications that use multiple processes or are multithreaded with asynchronous execution are well suited to multiprocessor systems. Systems requiring heavy computation capability, including detailed calculation for scientific applications, complex graphics, computer aided design based modeling, or electrical-engineering design might also demand multiprocessor systems. Assigning the applications to the selected processor the user can improve the performance of the machine. In the experimental setup two Pentium III 850 MHz processor computer was running Windows 2000 operating system (Fig. 1). The rapid prototyping environment based on MATLAB/Simulink ver. 7.1 with Real-Time Workshop (RTW), Real-Time Windows Target (RTWT) [8, 9] and external C compiler were used. The RTWT is a PC solution for prototyping and testing real-time systems and uses a single computer as a host and target. Integration between Simulink external mode and RTWT allows to use the Simulink model as a graphical user interface for signal visualization and parameter tuning. Typical applications for RTWT include real-time control prototype of the control system, real-time hardware-in-the-loop simulation – prototype controllers connected to a physical plant. The RTWT uses a small real-time kernel to ensure that the real-time application runs in real time. The real-time kernel runs at CPU ring zero (privileged or kernel mode) and uses the built-in PC clock as its primary source of time for timer interrupts. The RTWT is equipped with scheduler that runs the executable. The number of tasks is equal to the number of sampling periods in the Simulink model with multitasking mode. The maximum number of tasks is 32 and faster tasks have higher priorities than slower tasks.

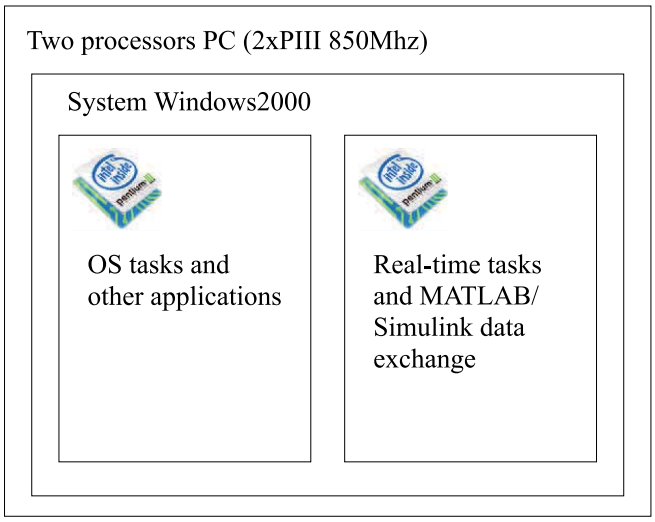

Fig. 1. Multiprocessor architecture for control tasks

2.2. RTW multitasking real-time mode

Real-Time Workshop supports the multitasking real-time code generation (Fig. 2). It makes possible to define the sampling or control period for the selected task. By this way he controller can realize several tasks with different sampling or control frequency.

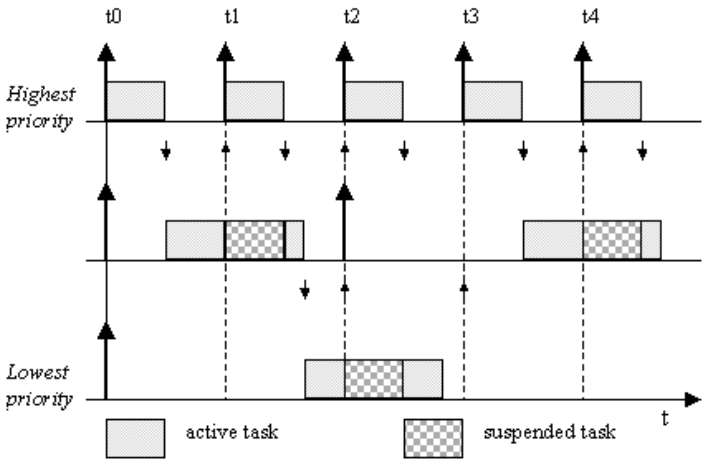

Fig. 2. Tasks execution in the multitasking mode

The control algorithm is defined by the set of Simulink blocks with defined sample rate. This parameter is common for every block or defined separately for single or group of blocks. The sample rate of any block must be an integer multiple of the base (i.e. the fastest) sample rate. The base sample rate is determined by the fixed step defined for the model solver. In a multitasking environment, the block with the fastest sample rates are executed by the task with the highest priority, the next slowest blocks are executed by a task with the next lower priority, and so on. Time available in between the processing of high priority tasks is used for processing lower priority tasks. Finally, using the RTWT up to 32 tasks can be executed. From the programmers point of view to ensure the appropriate data flow between task fired with different sample rates it is necessary to use Zero-Order Hold and Unit Delay block when transitioning from faster to slower blocks or in reverse order respectively. In the new version of Simulink the block Rate Transition can replace previously discussed blocks.

2.3. PC platform application

In this case the controlled external process is connected to the PC via dedicated FPGA based I/O board. The actuator unit of the MagLev system is controlled by the pulse width modulation (PWM) signal and measurement are realized by 12-bit A/D converters (Fig. 3). The FPGA unit is programmed in the custom way.

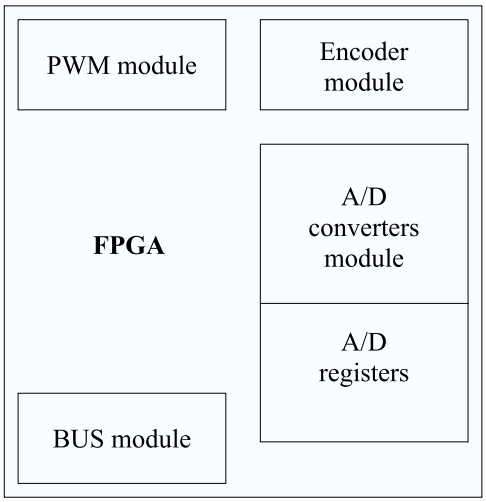

Fig. 3. User-defined FPGA core

The PWM signal generation task was transferred from the software layer of PC to the FPGA unit. The parameters of the PWM signal (frequency and duty cycle) are controlled by the set of registers placed in the FPGA. The A/D converters are handled by the FPGA chip to obtain fastest data acquisition. The converted data is

stored in the FPGA registers. Board registers can be read by the algorithm written as C-coded s-function and executed in the real-time as the RTW task. In every sampling period, when the real-time code is executed the user defined s-function, realized as device drivers are called. The maximum sampling frequency for the RTW/RTWT operation is mostly limited by the interfacing blocks. It was observed and analyzed that the I/O board interface and device drivers code strongly affect the time resources. For the installed ISA I/O board and task of 12 analogue channels to read, the maximal sampling frequency achived was 2kHz. For the same task, but with dedicated FPGA based I/O PCI board programmed in the custom way the maximal sampling/controlling frequency was 10 kHz. In both cases about 35% of free MATLAB resources were left for parameters tuning and signals monitoring purposes.

2.4. MPC565 embedded solution

The MPC565 is an 32-bit embedded microcontroller from Freescale Semiconductor [11], containing a 56 MHz PowerPC core, 1 MB of Flash memory on a single silicon chip (Fig. 4). This combination is ideal for high-performance automotive applications, as well as other control time-critical applications, which requires to monitor a number of analog inputs, generate PWM signals and to run complex control application. Additional Computing Muscle [11] is capable of powering complex applications with its PowerPC core, floating point unit (FPU) and third-generation time processor units (TPU3).

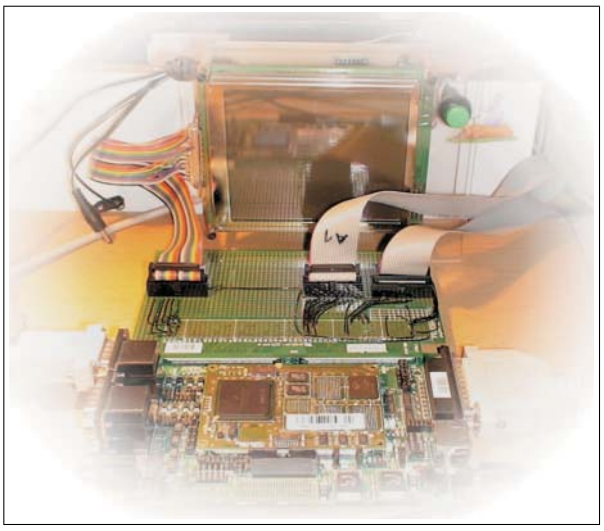

Fig. 4. Embedded system based on MPC565

Each TPU includes its own RISC core and memory system, which allows it to operate as its own microcontroller dedicated to timing functions, capable of processing up to 20 million instructions per second with a 40 MHz system clock. With a 56 MHz

system clock, up to 28 million instructions per second can be performed. Off-chip serial communications are handled by queued serial multichannel modules (QSMCM) that offer UART and SPI functionality. The MPC566 offers code compression in addition to all the other features on the MPC565, saving you up to 50 percent in memory space.

For the rapid development purposes the Embedded Target toolbox for Motorola MPC555 (ETMPC) [12] was applied (Fig. 5). Using ETMPC with the Real-Time Workshop Embedded Coder (RTWEC) [10] one can obtain a complete set of tools for developing embedded applications for the Motorola MPC56x processors. In conjunction with Simulink, Stateflow, and the RTWEC, the ETMPC allows to design and model dynamic system and algorithms on the host PC, and next, to compile, download, run and debug the generated code on the target hardware, seamlessly integrating with industry-standard compilers and development tools for the MPC5xx. For the first time use with MATLAB the dedicated boot code must be installed into the target hardware. The generated and downloaded code can be placed in the RAM or FLASH memory.

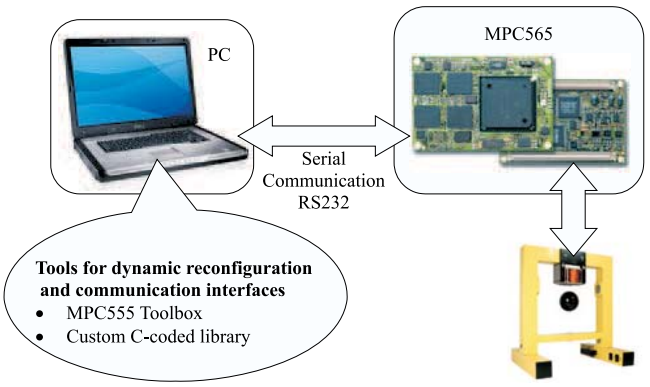

Fig. 5. Host target control architecture and communication method

The MPC microcontroller can be fully programmed using Simulink blocks, but for custom tasks there exists possibility to create C-coded s-functions using Embedded Coder Toolbox. The generated program can run on any Electronic Control Unit (ECU) that is based on the MPC5xx processor. The ETMPC Simulink blocks represents the dedicated MPC5xx's MIOS, TPU, QADC and TouCAN modules, provided with drivers for the on-chip analog input, digital I/O, PWM, serial and CAN devices. To access custom I/O resources it is necessary to write device drivers and integrate them with the automatically generated code. Figure 6 presents an example of Simulink control project for magnetic levitation system. The Embedded Target for Motorola MPC555 supports processor-in-the-loop (PIL) cosimulation, a technique that allows to evaluate how accurate a candidate control system operates on the actual target processor selected for the application.

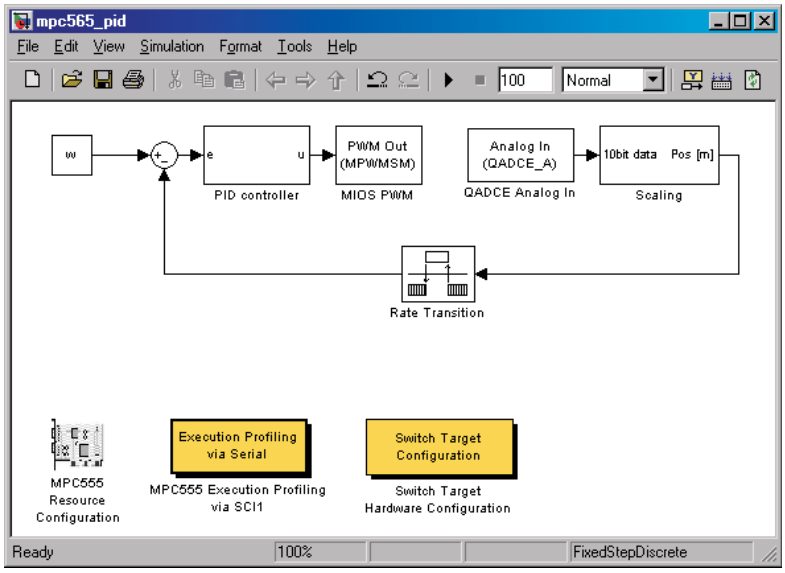

Fig. 6. PID Control loop algorithm designed in Simulink and prepared for MPC565 code generation

2.5. Analyzing real-time tasks in MPC565

The Embedded Target for Motorola MPC555 provides a set of utilities for recording, uploading and analyzing execution profile data for timer-based tasks and asynchronous Interrupt Service Routines (ISRs). With these utilities it is possible to generate a graphical display that shows when timer-based tasks and interrupt service routines are activated, preempted, resumed and completed. The Embedded Target generates a report with information on:

- maximum number of overruns for each timer-based task since model execution had started,
- maximum turnaround time for each timer-based task since model execution began,
- analysis of profiling data for timer-based tasks and asynchronous interrupts over a period of time.

Task turnaround time is the elapsed time between start and finish of a task. If the task is not preempted then the task turnaround time is equal to the task execution time. Task execution time is that part of the time between task start and finish when the task is actually running and not preempted by another task. Task overruns occur when a timer task does not complete before the same task is scheduled to run again. Depending on how the real-time scheduler is configured, a task overrun may be handled as a real-time failure. Alternatively, a small number of concurrent task

overruns may be allowed in order to accommodate cases when a task occasionally runs longer than normal to complete.

To analyze the performance of the PID closed loop for magnetic levitation system (Fig. 6) the model was decomposed into three parts, operating with custom sample and control rates:

- T0AD – sample rate for Analog In and Scaling blocks (AD task),
- T0PID – sample rate for PID, Des Val. and PWM blocks (PID task),
- T0EP – sample rate for Execution Profiling (EP task).

Four experiments (A-D) were conducted in order to estimate the shortest sample and control rates. (Tab. 1). A, C are experiments with single task for control loop. B, D are experiments with multitask for control loop. The time-critical sample rate for the program loaded to the RAM and diagnosed by execution profiling tool is equal to 64 μs. Thus, the sample rate of the model blocks must be an integer multiplication of 64 μs. This fact strongly affects the digital controller design.

Table 1
Experimental results

Experiment	T0AD [μs]	T0PID [μs]	T0EP [μs]
A	960	960	1920
B	192	960	1920
C	64	64	1920
D	64	128	1920

The experiments C and D are time-critical for this hardware architecture. The reason is that the program is executed in the RAM memory. The execution profiling module operates properly for the embedded application designed for the RAM memory only. Figure 7 shows two tasks (control and data acquisition) pending during experiment B.

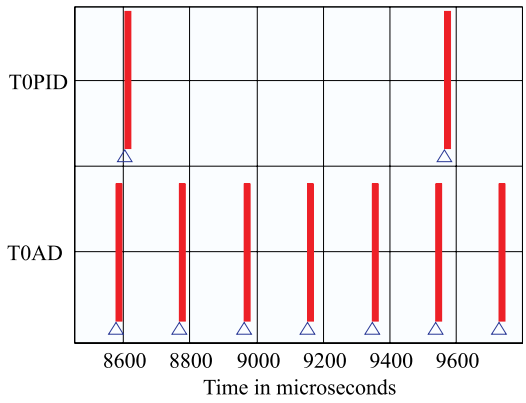

Fig. 7. Tasks execution example diagram

Five samples are collected between the PID task. The task is fired at the time stamp marked by the triangle and is pending the amount of time indicated by the bar width. One can notice that all tasks were finished in a time slot. The controller performs any mathematical calculations based on acquired measurements. Figures 8 and 9 show maximum turnaround and execution times.

The time values were measured by microcontroller timer incremented with the resolution of 0.4 μs. The important observation is that concurrent task overruns does not appear. It means, that each task was executed in the time slot defined by the sample rate. The sample rate is appropriate. The task with the lowest priority is executed longer when time-critical task have highest priorities. Total execution time of the control loop is longer than in multitasking mode.

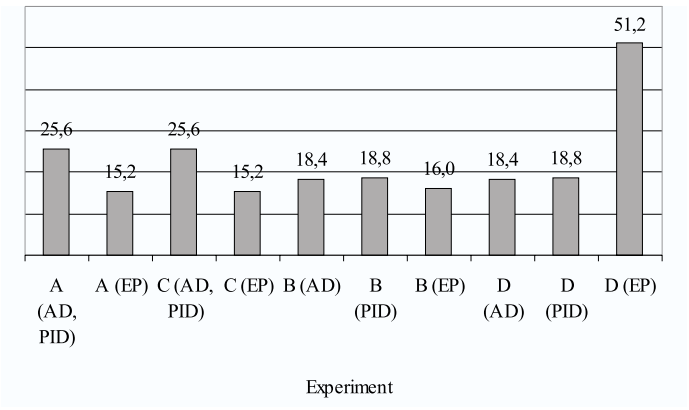

Fig. 8. Maximum turnaround time [μs]

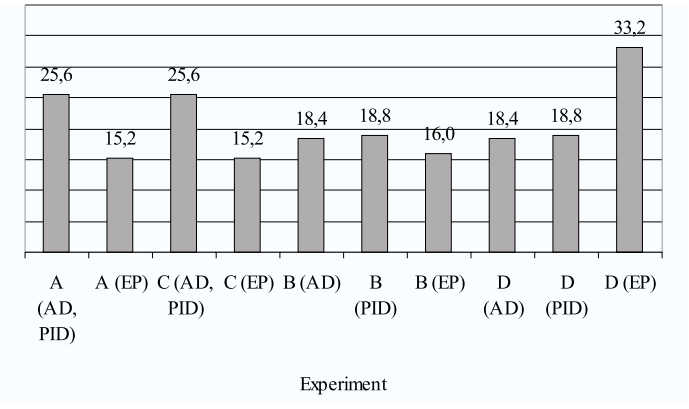

Fig. 9. Maximum execution time [μs]

2.6. Communication

The communication mechanism between embedded controller and host system opens a possibility to change the control strategy, modify the controller parameters and upload the collected data for analysis and future interactions. The communication allows to create hierarchical structure of the control system. The used hardware offers three communication channels: serial, CAN and Ethernet protocols. Due to hardware resources the serial communication is used in the experiments. At the host system PC computer the MATLAB or custom user application is applied to communicate via serial port. The MPC565 board devices are ready for serial communication and direct connection with the host system using RS232 standard. The Embedded Toolbox contains hardware drivers for serial devices. However, the communication protocol and communication tasks must be designed and implemented in both platforms. For example, to monitor the signal processing performed by the microcontroller, the communication library allows to collect the specified number of data and transmit to the host system. To support data exchange between microcontroller and host PC a special serial communication frame format was designed and implemented.

2.7. Visualization task

For the direct visualization of MPC565 controller operation the LCD panel is used (Fig. 10).

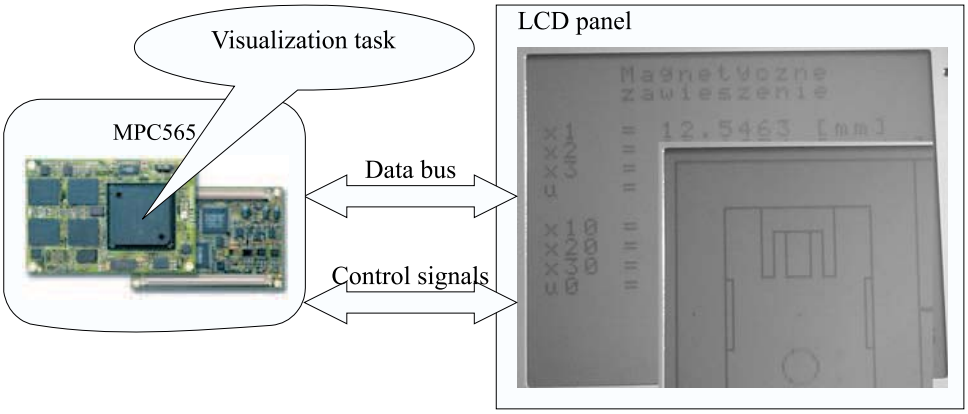

Fig. 10. Process visualization

This device can operate in text or graphic modes. The internal panel memory is organized to store information for both modes separately and the display contents can be toggled. The LCD panel is connected to the MPC565 microcontroller via parallel 8 bit data bus and control lines. From the programming point of view the LCD panel was programmed by a set of functions organized as library. These functions allows to form data and control signals in LCD panel standard, prepare general view of the

user interface and update the necessary information. The visualization task should have the lowest priority than control and data exchange tasks, thus is configured with the lowest sampling time. The refresh rate is strongly influenced by the amount of information to be transmitted to the LCD panel memory updated on the LCD screen.

3. Conclusions

Two real-time control architectures were analysed in the paper. The PC platform gives possibility to simply design and check the control algorithms idea. The embedded solution has more restrictions regarding to microcontroller resources and architecture. Using reconfigurable devices some components of the control task can be transformed to the hardware layer. The programmable reconfiguration allows to change some parameters dynamically while the system is operating. Monitoring and analyzing of real-time tasks in the operating system of embedded controller is supported by Embedded Target Toolbox utilities. To achieve reconfiguration the interfacing and communication algorithms are necessary for data exchange. The rapid prototyping method available for both architectures supports the control application design. For custom projects it is necessary to program some interfaces individually. Thus, the low level C-language programming is required.

Acknowledgements

This paper was supported by reseach grant of AGH.

References

[1] Iserman R.: *Digitale Regelsysteme*. Berlin, Springers-Verlag 1988

[2] Magalhaes A. P.: *A Survey on Estimating the Timing Constraints of Hard Real-Time System*. Design Automation for Embedded Systems, vol. 1, 1996, 213–230

[3] Seto D., Lehoczky J. P., Sha L., Shin K. G.: *On Task Scheduling in Real-Time control Systems*. Proc. of the IEEE Real-Time Systems Symposium, 1996

[4] Grega W.: *Performance evaluation of model-reference control*. 7th IEEE International Conference on Methods and Models in Automation and Robotics, Międzyzdroje 2001, 407–412

[5] Grega W., Kołek K.: *Simulation and Real-time Control: from Simulink to Industrial Applications*. Proc. of 11th IEEE International Conference on Computer Aided Control System Design, Glasgow, 2002, 104–109

[6] Piłat A.: *Control of Magnetic Levitation Systems*. Doctoral Thesis, AGH Institute of Automatics, Cracow, Poland, 2002

[7] Grega W., Piłat A.: *Comparison of Linear Control Methods for AMB System*. International Journal of Applied Mathematics and Computer Science, vol. 15, No. 2, 2005, 101–111

[8] Real-Time Workshop, Users Guide, MathWorks, Inc. 2005
[9] Real-Time Windows Target, Users Guide, MathWorks, Inc. 2005
[10] Real-Time Workshop Embedded Coder, Users Guide, MathWorks, Inc. 2005
[11] MPC565/MPC566 Users Manual, Motorola Inc. 2002
[12] Embedded Target for Motorola MPC555, Users Guide, MathWorks, Inc. 2005