We are IntechOpen,
the world's leading publisher of Open Access books
Built by scientists, for scientists

6,900
Open access books available

186,000
International authors and editors

200M
Downloads

154
Countries delivered to

Our authors are among the

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS
BOOK CITATION INDEX
INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Microcontroller-based Data Acquisition Device for Process Control and Monitoring Applications

Vladimír Vašek, Petr Dostálek and Jan Dolinay
Tomas Bata University in Zlín
Czech Republic

1. Introduction

Process measurement is one of the most important tasks in the whole control system. It is determined by the fact that control accuracy is fully dependent on how preciously measuring chain works. Present-day there is available number of devices performing data acquisition tasks – standard cards for PCI or ISA bus which are suitable for standard personal computers and its industrial versions and modules for industrial automation usually equipped with RS485, CAN and other interfaces. Independent category is formed by smart sensors incorporating sensor, converter to unified signal and data acquisition device in one embedded system with very compact dimensions and low power consumption. They have number of advantageous features such as automatic diagnostic and calibration, high accuracy and immunity against electromagnetic interference due to short signal paths. On the other hand lower operating temperature range reduces their usage to laboratory applications, automotive and aircraft industry where compact dimensions and low weight are crucial. Quite often occurred situations when it is necessary to measure data in terrain where it is not possible to use standard computer equipped with DAQ card. In these cases laptop computer equipped with portable data acquisition device may be very advantageous. On the market are available devices equipped with USB 2.0 connectivity which can fully functionally substitute PCI cards. But they are not able to work without connected computer which must continually control all DAQ operations. Therefore it is not possible to use them in applications where is required long-term monitoring and archiving process quantities in distant areas without access to mains power.

This contribution proposes design of multi-channel portable data acquisition device based on low cost general-purpose 8-bit microcontroller Freescale 68HC908GP32, which was developed in our department mainly for control and monitoring educational laboratory models. First part deals with hardware design of the DAQ device with focus on description of operation of individual functional blocks. After that follows description of internal software (firmware) based on real-time operating system RTMON for HC08, which was developed on our department especially for microcontroller-based embedded systems with CPU08 main processor core. Next chapters discuss DAQ device software support in form of program libraries for MS Visual C++, Control Web and Matlab/Simulink development environments, which can significantly improve development time of new process control or

monitoring applications. And finally last part deals with verification of the developed DAQ device with control of selected laboratory model.

2. Data acquisition device hardware design

2.1 Hardware overview

Hardware design of the DAQ device is fully adopted to support 16 analog inputs with 12-bit resolution, 8 digital inputs and outputs and one analog output with 12-bit resolution with stress on low power consumption enabling long operation when battery supply is used. The core of the DAQ device is 8-bit general purpose Motorola microcontroller 68HC908GP32 with Von-Neumann architecture which is fully up-ward compatible with the 68HC05 family. On the chip are integrated timer interface with input capture and output compare functions, 8-channel analog-to-digital converter with 8-bit resolution, up to 33 general-purpose I/O pins, clock generator module with PLL, serial communication interface and serial peripheral interface. M68HC908GP32 has implemented several protective and security functions such as low-voltage inhibit which monitors power supply voltage, computer operates properly (COP) counter and FLASH memory protection mechanism preventing unauthorized reading of the user's program. Internal RAM memory has capacity of 512B and FLASH memory 32 KB. Internal clock frequency can be 8 MHz at 5 V operating voltage or 4 MHz at 3 V operating voltage. Microcontroller also supports wait and stop low-power modes (Freescale, 2008)

Central processor unit CPU08 which is the main part of M68HC908GP32 microcontroller is fully object code compatible with M68HC05. This feature allows easy code migration to new architecture providing high speed, low power and better processing capabilities.

Central processor unit features can be summarized in the following points:

- 8 MHz bus speed at 5 V, 4 MHz bus speed at 3 V
- 16-bit stack pointer with new stack manipulation instructions
- 16-bit index register with index register instructions
- 78 new instructions
- Memory to memory moves without using the accumulator
- 16 addressing modes including stack relative
- 64 Kbytes program/data memory space
- Fully static low-voltage/low-power design (Freescale, 2006)

Analog-to-digital conversion is performed by the A/D converter Linear Technology LTC1298. It is micro power, 2-channel, 12-bit switched-capacitor successive approximation sampling A/D converter which can operate on 5 V to 9 V power supplies. Communication with microcontrollers is handled by 3-wire synchronous serial interface. It typically draws only 250µA of supply current during conversion and only 1nA in power down mode in which enters after each conversion (Linear Technology, 1994).

Digital-to-analog circuit utilizes 12-bit D/A converter Burr-Brown DAC7611 with internal 2.435V reference and high speed rail-to-rail amplifier. It requires a single 5 V supply. Power consumption is only 2.5 mW at 5 V. Build-in synchronous serial interface is compatible with variety of digital signal processors and microcontrollers (Burr-Brown, 1998).

2.2 Circuits design

Electronic circuits of the data acquisition device can be divided into the seven main functional blocks as depicted in the Fig.1: analog-to-digital converter with analog

multiplexer circuits, microcontroller circuits, digital-to-analog converter and amplifier circuits, digital I/O driver circuits, serial communication interface RS232 and finally power supply circuits.
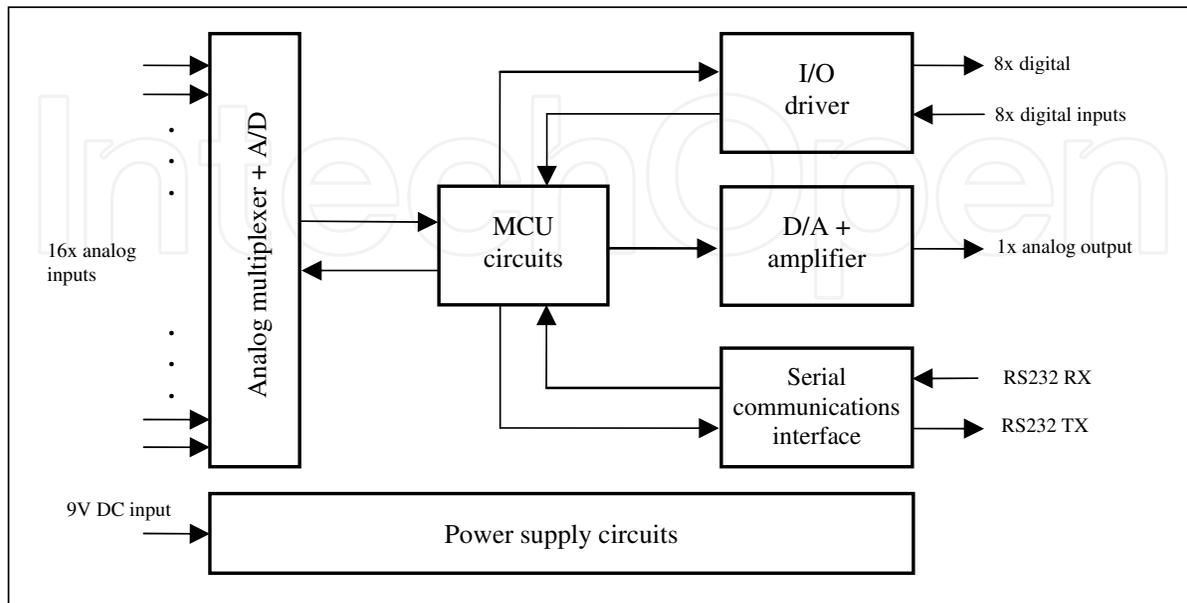

Fig. 1. Block diagram of the DAQ device

MCU circuits (Fig.2) incorporate all electronics circuits needed for correct function of the microcontroller M68HC908. They consists of Pierce crystal oscillator with output frequency of 32.768 kHz connected to OSC1 and OSC2 pins of the MCU and filter network (parts R3, C3, C4) needed by internal phase-locked loop circuit (PLL) allowing programmable selection of output frequency. Reset (RST) and interrupt request (IRQ) pins are permanently connected through pull-up resistors R4 and R5 to high level, because their function is not in data acquisition device used. Reset of the MCU is automatically generated after power on by internal circuits of the microcontroller. Output frequency of the Pierce oscillator is by internal PLL circuit increased to 32 MHz resulting in internal bus clock frequency of 8 MHz which is used as reference frequency of the CPU and most internal peripherals. Ports PTA and PTB are completely dedicated for digital input and output functions except pin PTB0/AD0 which is used in its alternative function as input of analog-to-digital converter for monitoring of accumulator battery voltage. Low voltage condition occurs when supply voltage drops below 6.5 V. So voltage of the one NiMH accumulator in the battery drops to 1.08 V indicating that approximately 80 - 90 % of its capacity is discharged. Correct DAQ device operation is guaranteed at minimum voltage of 5.5 V. Pin PTD3 is connected to the LED diode indicating status of the device. PTD0 to PTD2 pins provides binary selection of active input analog channel, PTC0 to PTC4 pins perform synchronous serial communication with A/D and D/A converters.

Digital input / output driver circuit (Fig.3) has two important functions. Firstly, it protects microcontroller inputs from electrostatic discharge which may occur during handling and connecting DAQ device to the measured object and secondly, it boosts output current from microcontroller pins and protects them against overload or short-circuits. It uses two non inverting 3-state high-speed octal bus buffers 74HC244 with 35mA maximum current output capability per pin.
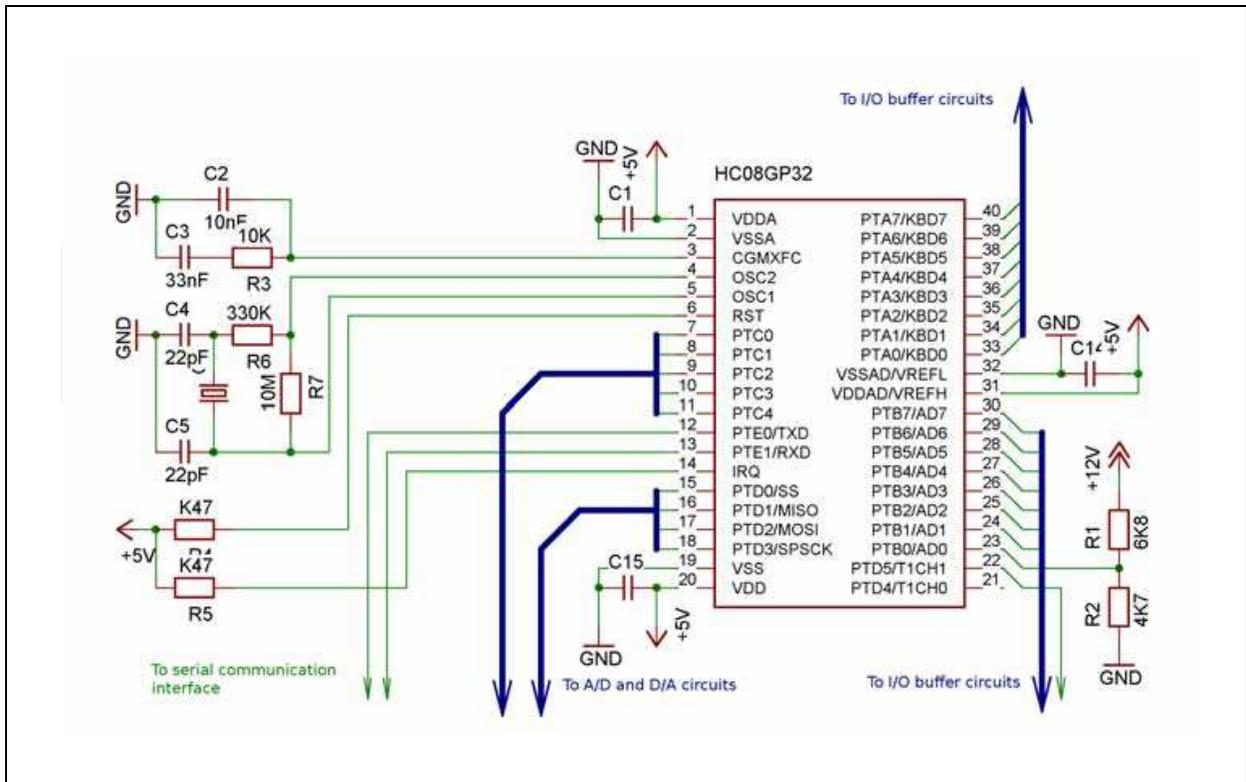
Fig. 2. Microcontroller circuits schematics.
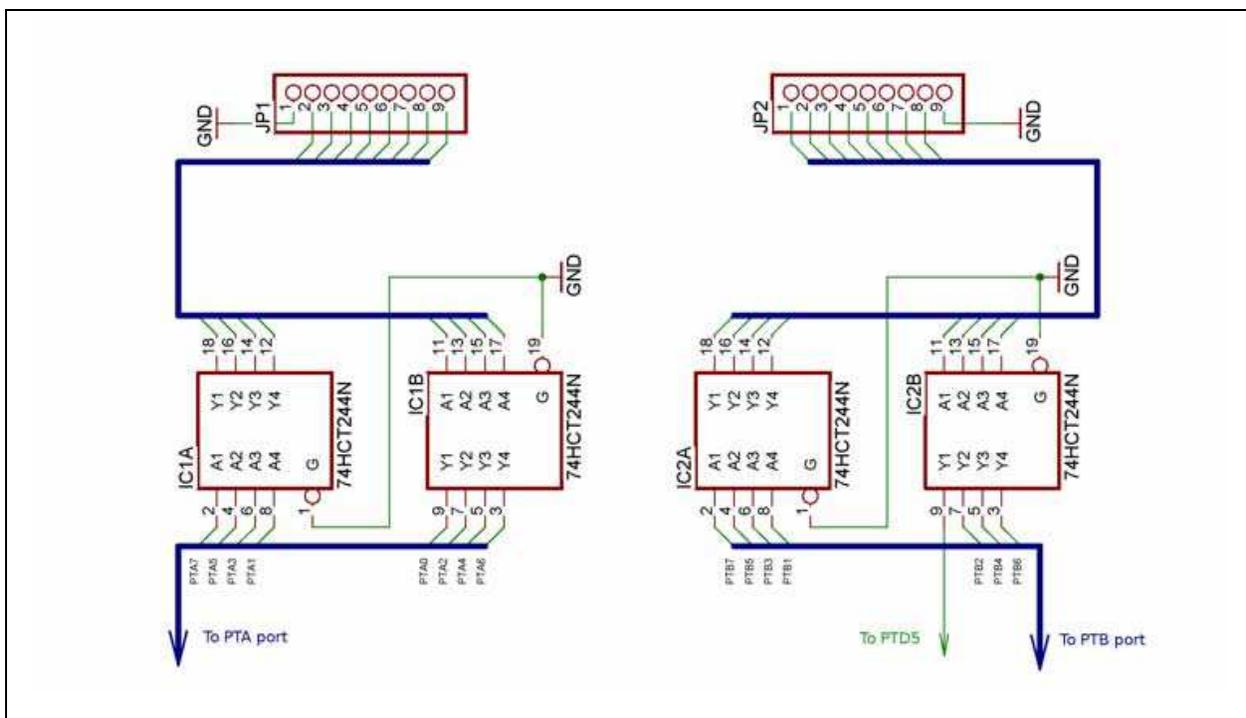
Fig. 3. Digital input/output buffer schematics.

Analog multiplexer, A/D and D/A converter circuits schematics is depicted in the Fig.4. Measured analog voltage signals 0 - 10 V first enters the input dividers performing signal conditioning to voltage range of 0 - 5 V to meet specifications of analog-to-digital converter.
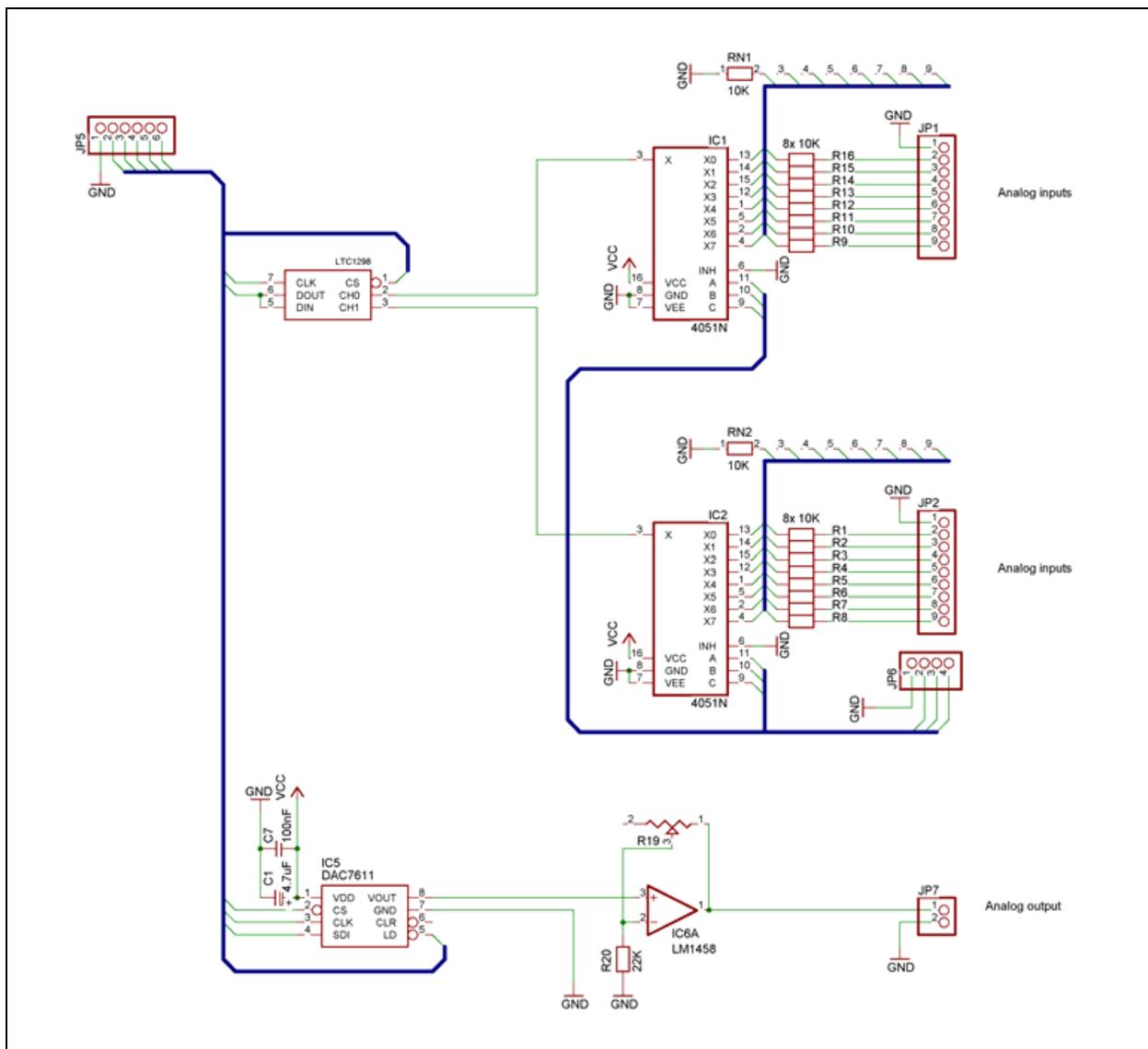
Fig. 4. Analog multiplexer, A/D and D/A converter circuits schematics.

After that they are switched by two 8-channel high-speed analog multiplexers 74HC4051 to the two input analog channels CH0 and CH1 of 12-bit A/D converter LTC1298. Input channel pair selection is realized by MCU pins PTD0, PTD1 and PTD2. Microcontroller communicates with A/D converter using pins PTC1 (data clock), PTC2 (serial data in/out) and PTC4 (chip select). One conversion is done after 16 clock signal periods resulting in maximum of 12.5 kHz sampling rate at clock frequency of 200 kHz.

Analog output circuits are based on digital-to-analog converter DAC7611 with 12-bit resolution. It is controlled by MCU pins PTC0 (chip select), PTC1 (data clock), PTC2 (serial data in) and PTC3 (load/strobe). One D/A conversion takes place after 12 clock periods, output settling time is 7µs to 1 LSB. At the converter output is connected operational amplifier MC1458 in non-inverting configuration amplifying output D/A voltage to the standard range of 0 – 10 V. Exact output range can be adjusted by variable resistor R19.

Data acquisition device contains three independent power supplies. Digital parts (MCU circuits, input/output driver, serial communications interface and D/A converter) are supplied by circuit depicted in the Fig.5. It uses low-drop 5V/1A regulator LM2940 in

manufacturer's recommended wiring enabling correct operation with input voltage down to 5.5 V. Input of the supply is protected against overloading or polarity reversing by fast acting fuse. Analog-to-digital converter is supplied from high-precision voltage reference LM336-Z5.0 which is connected to adjustable current source LM334. Output voltage can be adjusted to the exact 5 V value by variable resistor R18 (Fig.6). Analog output amplifier is supplied by DC-DC converter ICL7662 providing positive and negative voltages for analog output operational amplifier from single supply.
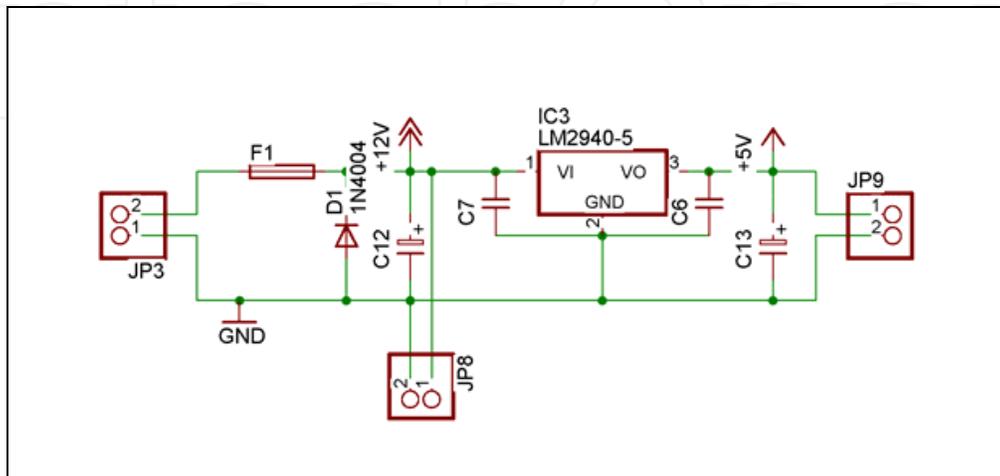
Fig. 5. Power supply for digital circuits schematic
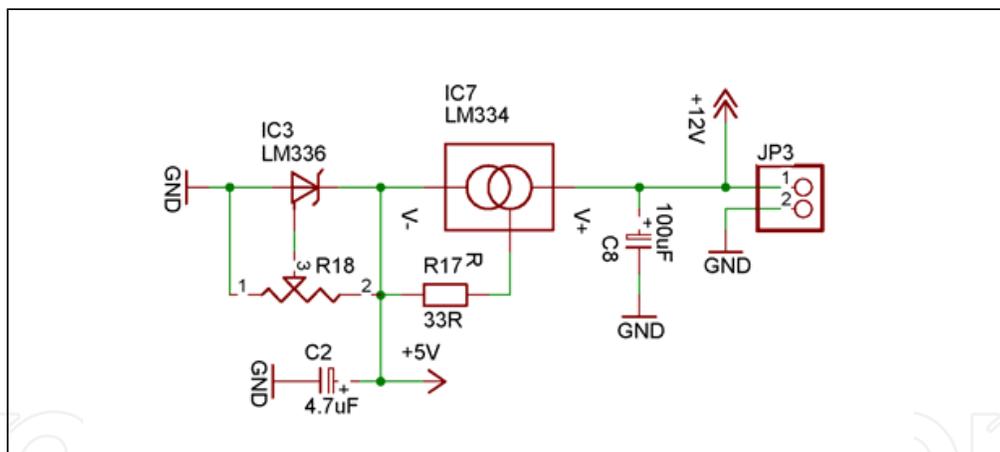
Fig. 6. Power supply for A/D converter schematics

Serial communications interface utilizes standard TTL to RS232 and RS232 to TTL converter MAX232 in manufacturer's reference wiring (Fig. 7). It is connected to microcontroller via pins PTE0/TxD and PTE1/RxD to serial communications module.

Data acquisition device is realized on two printed circuit boards (PCB), which are connected together using single row pin headers. First PCB (Fig.8 – left picture) contains mainly digital circuits – microcontroller, input / output drivers and serial communications interface. It also carries main +5V power supply equipped with low-drop voltage regulator. While second PCB (Fig.8 – right picture) incorporates all analog signal processing circuits – analog-to-digital and digital-to-analog converter, output amplifier, analog multiplexers, DC-DC converter and finally reference voltage source for A/D converter. Device features are summarized in Table 1.
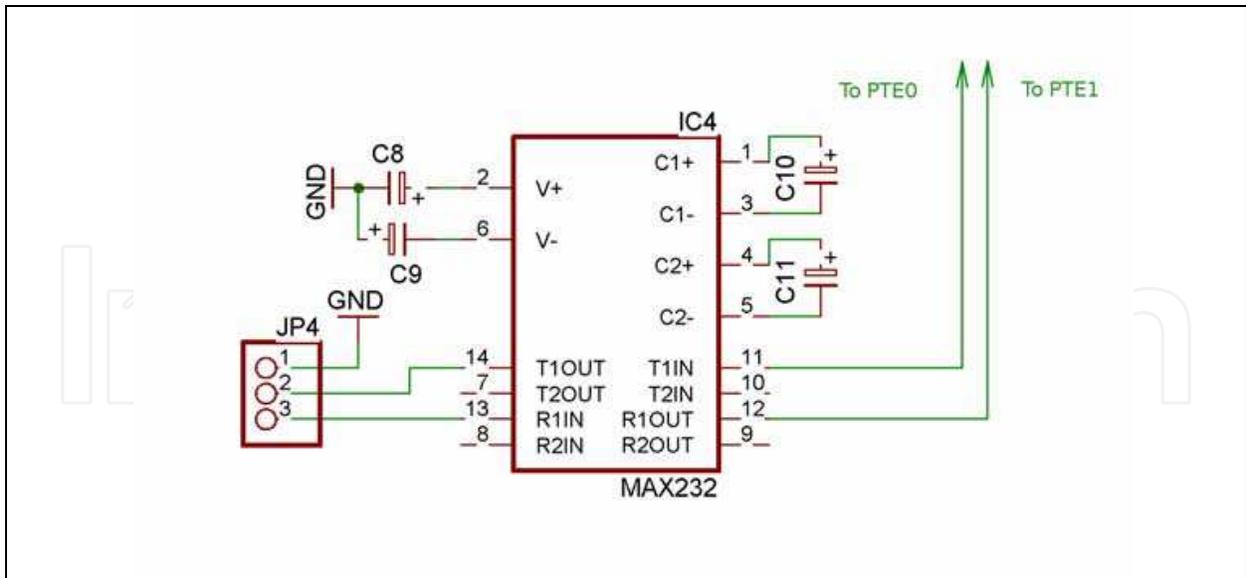
Fig. 7. Serial communication interface schematics
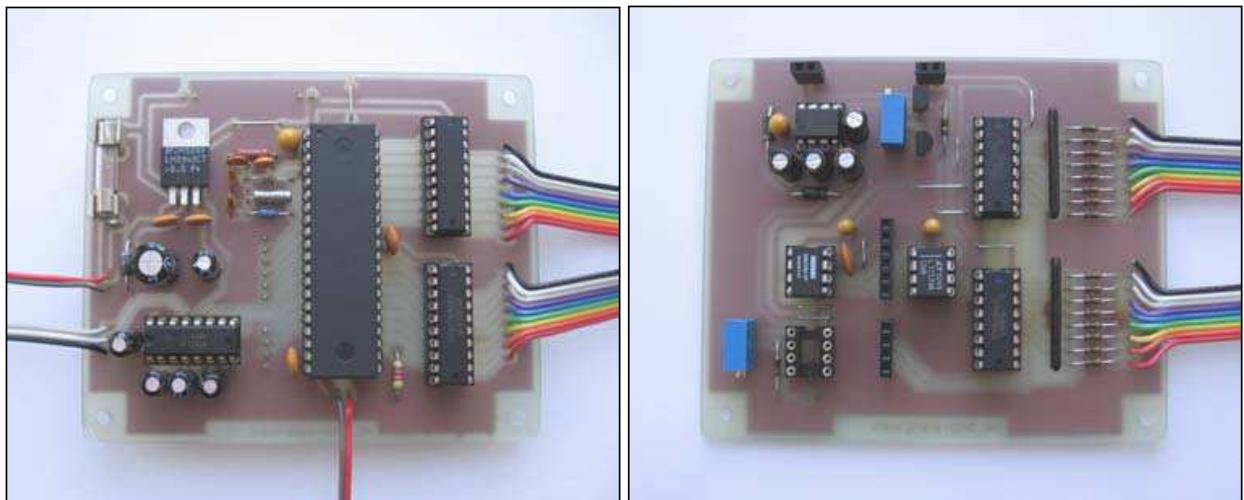
Fig. 8. Finalized DAQ device printed circuit boards

Digital inputs	8 channels, TTL compatible
Digital outputs	8 channels, TTL compatible
Analog inputs	16 channels, 12 bits resolution, input range 0 – 10 V
Analog outputs	1 channel, 12 bits resolution, output range 0 – 10 V
Supply voltage	6.5 to 9V DC
Communication	RS232 interface, 57600 Bd, 8-bit data, 1 start bit, 1 stop bit

Table 1. Technical parameters of the DAQ device

Fig. 9. Photograph of the prototype DAQ unit

3. DAQ device firmware design

DAQ device internal software is based on real-time operating system RTMON for HC08, which was developed on our department especially for microcontroller-based embedded systems with CPU08 main processor core. So software is formed of RTMON core and individual processes which perform all necessary tasks. Each process activity is controlled by operating system core on the basis of process priority and other information stored in the task descriptor. Structure of the DAQ device firmware is depicted in the Fig. 10. There are 4 main processes and 1 interrupt handling routine. RTMON core and program processes functions are described in chapters 4.1 and 4.2.
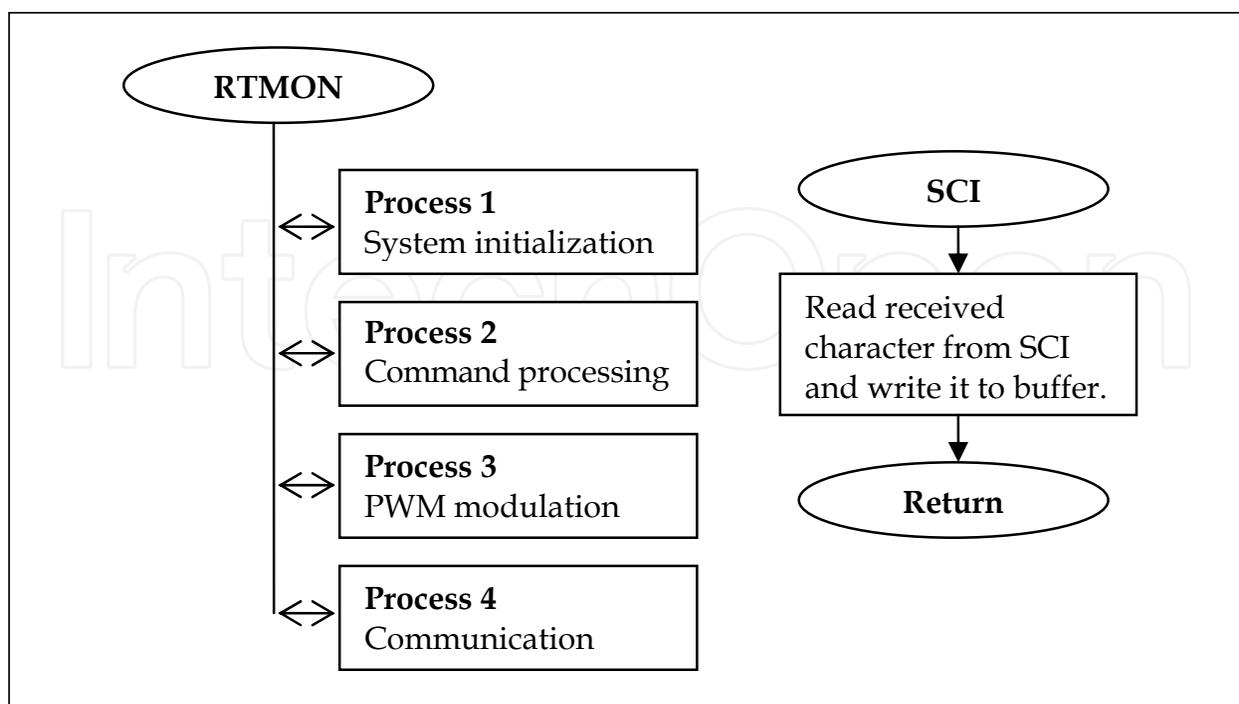
Fig. 10. Internal software structure

3.1 Real-time operating system RTMON

RTMON is preemptive multitasking operating system which is simplified to great extend to allow easy use for programmers. It is written in C language with the exception of small platform-specific code written in assembler. The scheduler assigns time slices to processes based on their priority. The priority is integer in the range 1 to 254. Priority 0 is the highest and is reserved for the RTMON initialization process and priority 255 is the lowest and is reserved for the idle process (called dummy in RTMON).

RTMON allows execution of two different types of processes (tasks): normal processes which execute only once (such process typically contains infinite loop) and periodical process which is started automatically by RTMON with given period. These periodical processes are useful for many applications, for example, in discrete controllers which need to periodically sample the input signal and update the outputs.

For the sake of simplicity of both the implementation and usage, several restrictions are applied. First, the RAM memory for processes and their stacks is statically allocated for the maximal number of processes as defined in configuration file. In the user program, it is not possible to use this memory even if there are fewer processes defined. In case more RAM is needed for the user program, the maximum number of tasks and/or stack-pool size can be changed in configuration file and the RTMON library must be rebuild.

The priority of each task must be unique, so that in each moment one task (the one with highest priority) can be selected and executed on the CPU. The scheduler does not support cyclical switching of several processes with the same priority on the CPU in round-robin fashion; it simply chooses the task with highest priority from the list of tasks which are ready to run. Processes can be created on the fly, but it is not possible to free and reuse memory of a process. No more than the maximal number of processes can be created, even if some processes were previously deleted.

These restrictions, however, do not present any big problem for most applications and allow for small kernel code size and ease of use.

There are only two objects (data structures) which RTMON contains: process (task) and queue. The queues are buffers for transferring data between processes. It would more properly be called mailboxes in our implementation as each queue can contain only 1 message. Several queues can be created, each containing a message (data buffer) of certain size. The size can be specified when creating the queue and is limited by the total size of RAM reserved for all buffers of all the queues (queue pool size). Processes can read and write data to the queue and wait for the queue to become empty or to become full. This allows for use of the queue also as a synchronization object (semaphore).

The RTMON uses timer interrupt which occurs at certain period (e.g. 10 ms) to periodically execute the scheduler, which decides which process will run in the next time slice. The timer interrupt routine is implemented in assembly language. It first stores CPU registers onto the stack and then calls RTMON kernel, which is a C function. The kernel then finds the process with highest priority which is in ready-to-run state and switches the context, so that the code of this process is executed after return from the interrupt service routine. If no process is ready to run, then a special dummy process is executed. This dummy process is contained within RTMON code and does nothing.

The following basic operations can be performed with a process in RTMON. Each operation corresponds to a function in the RTMON library which user program can call:

- create process
- start process

- stop process
- delay process
- continue process execution
- abort (delete) process

For queues there are the following functions:

- create queue (specify size)
- write to a queue with/without waiting
- read from a queue with/without waiting

There are also two functions for controlling the RTMON core:

- initialize RTMON
- end RTMON operation

The RTMON system is used as a precompiled library accompanied by a header file. This simplifies the organization of the project and the build process. User enables RTMON usage in his program by including the header file (rtmon.h) in his source and adding the library to his project.

3.2 Processes functions description

Process 1 is highest priority process which performs DAQ device initialization after power up or reset. It sets all digital outputs to low state (logic 0), setups serial communications interface to communication speed of 57600 Bd, 8-bit data frame, 1 start bit and 1 stop bit, sets analog output to 0 V and finally initializes all necessary data structures. Because of its highest priority no other processes can be switched by OS core into the "run" state. After all initializations are done, process suspends itself.

Process 2 performs all tasks related to command interpretation and execution. It waits for complete command string in the receiver buffer which is handled by serial communication interface (SCI) interrupt routine. This interrupt routine is automatically called when SCI receive one character from the higher-level control system (personal computer or programmable logic controller, for example). When command is completely received in the buffer, process will decode it and executes required action.

Process 3 is periodically activated process performing pulse-width modulation (PWM) on all 8 digital output channels when it is demanded. Its priority is set to higher level than process 2 and process 4 because the PWM is time critical function sensitive to accurate timing. Its resolution is 8-bit allows setting 256 different duty cycles at output. Period of the PWM signal is set to 1000 ms which is optimal value for many controlled systems with higher time constants (heating elements, for example).

Process 4 provides communication via RS232 serial interface with supervisory system. It generates responses to all commands regarding to defined communication protocol include error processing. It has defined lowest priority among of all processes.

3.3 Communication protocol

In order to achieve compatibility with many software platforms, universal ASCII-based communication protocol was chosen. Very advantageous is the possibility to send all implemented commands using generic terminal program that is included in most operating systems. Each command can be divided up to five parts depending on concrete function implementation. Communication starts with character "~" then must follow command name with fixed length of two characters (for example "AO" means set analog output).

After it is the first command parameter with length of one character (channel index) next character is space followed by second parameter (value). Command must be terminated by CRLF sequence.

Command:	~	A	O	0	_	4	.	2	5	CR	LF

- set analog output channel 0 to 4.25 V

Response:	~	A	O	0	=	4	.	2	5	CR	LF

- notification that AO0 was successfully set to 4.25 V
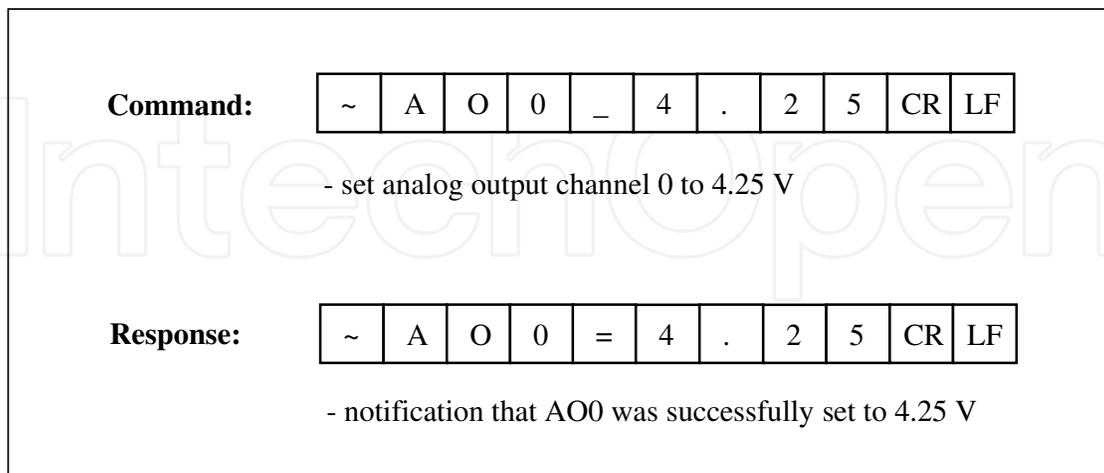
Fig. 11. Communication protocol example

4. Software support

Although communication protocol is very simple and easy to understand it is more comfortable in a control application to call functions which can automatically generate commands for the data acquisition device and consequently process its response. Application developer then does not need to know exact communication protocol and do not need to program it. This simplification results in faster program development and reduction of debugging time. For the portable data acquisition device there were created supporting program libraries for Visual C++, Control Web 5 and Matlab 6.5 (and higher versions with serial port object support) software environments.

4.1 Support libraries for visual C and Matlab

Created libraries incorporate all functions implemented in the device including error processing. For device testing and diagnosis, a DAQ test utility was created. This program can test all functions of the DAQ device and may be very helpful for testing wire connections to the monitored or controlled system. Main window of diagnostic utility is depicted in the Fig.12. Left part of the window "Analog inputs readings" contains 16 edit boxes indicating current voltage levels presented in the analog inputs. Its value is periodically updated with the period of 500 ms. "Digital inputs readings" fields show same information but for digital inputs (only two states 0 and 1). In the window part "Digital output setting" are 8 buttons enabling changing the state of the each digital output. And finally last control "Analog output setting" providing slider for setting the output voltage level on analog output channel.

Matlab 6.5 library has the same functions implemented with only one difference – in place of device handle is a serial port object. Each function is available in separate m-file, so it is very simple to modify them by the user. In the Table 2 are listed all implemented library functions for Visual C++ software environment. Matlab 6.5 library includes similar functions with only one difference – in place of device handle is a serial port object.

Function	Description
HANDLE **Open_device** (const char*)	Opens device connected to specified serial port ("COM1", "COM2", …) and returns device handle.
int **Close_device** (HANDLE h)	Closes device with specified handle.
int **Set_digital_out** (HANDLE h, int output, float value)	Sets specified digital channel (0 to 7) to desired logical value (0 or 1). When floating point value in the range (0; 1) is specified, PWM modulation is activated.
int **Set_digital_out_B** (HANDLE h, int value)	Sets digital channels to value (0 to 255). For example value of 100 sets digital outputs to state 01100100.
int **Set_analog_out** (HANDLE h, int output, float value)	Sets analog output on specified channel to desired value in volts (0 to 10V). Function accepts values in floating-point format.

Table 2. Implemented library functions for Visual C++
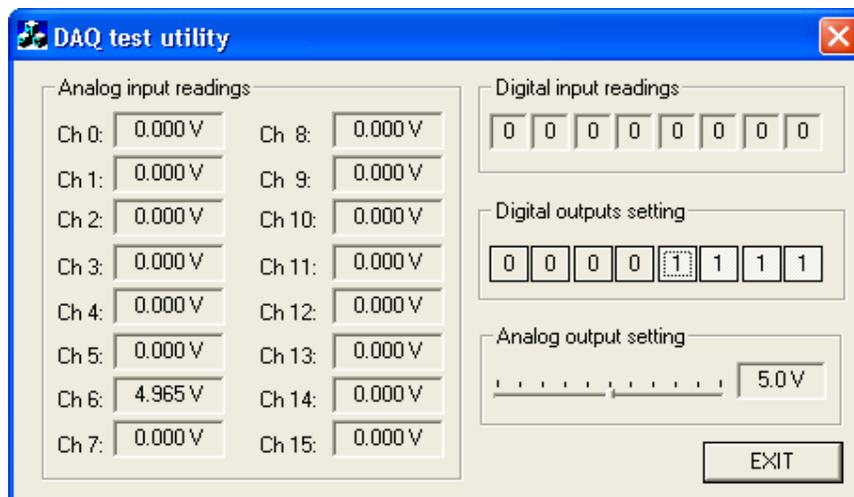
Fig. 12. Main window of the DAQ device test utility

4.2 Driver support for Control Web

Control Web is Rapid Application Development system developed by Moravian Instruments which is suitable for process visualization and real-time control, HMI applications, technological information systems and other applications.

Application design in Control Web 5 is very fast and comfortable due to integrated development environment, which supports more possibilities how to create new application. The basic idea is to build in graphical editor basic components to the larger block, which can gradually form whole system. Each component can be configured in sheet editor enabling transparent parameter settings. Because resulting code is saved in text form and then compiled there is in parallel also a text editor available. Each component can be selected from instruments palette organizing them in subcategories – for example system instruments, flat instruments and so on. Selected category can be expanded if it is possible to

next sub trees. Another type of the object in Control Web is data element. Each data element represents location in system memory, which can save value of the measured quantity, for example (Moravian Instruments, 2005).

Control Web is in standard installation equipped with several drivers which can be divided to two main categories – for demonstration and testing purposes (Virtual Driver, Model driver, Simulation Driver, Simulating Driver) and general drivers for use in real applications (DDE Client Driver, ASCDRV5 driver, TCP/IP driver). Device driver is independent component in a form of dynamically linked library with standardized interface. During development of the Control Web system gradually originated three versions of the interface. Basic interface was defined for Control Web version 3 and must be implemented in every driver. Newer interface version 4 was created with Control Web 2000 and finally newest version 5 was defined for Control Web 5. Backward compatibility is guaranteed by implementation of the basic interface in all higher versions of the interface.

Control Web communicates with driver using channels, which must be defined in the driver map file (file with extension dmf). Each channel is defined by number, direction (input, output) and data type (real, boolean, string, and others). Driver configuration is stored in the parametric text file (extension par) containing specific information for correct driver initialization. All parameters can be easily viewed and modified using "Driver inspector".

On the basis of requirements of the driver interface version 3 a device driver for portable data acquisition device was created. It supports all available hardware functions – there are 16 input real channels, 8 boolean input and output channels and one real output channel. For verification of the correct function of the driver and DAQ device test application was created. It enables to manually set output channels and in 500 ms period reads all input channels. Main window of the test application is in Fig.13.
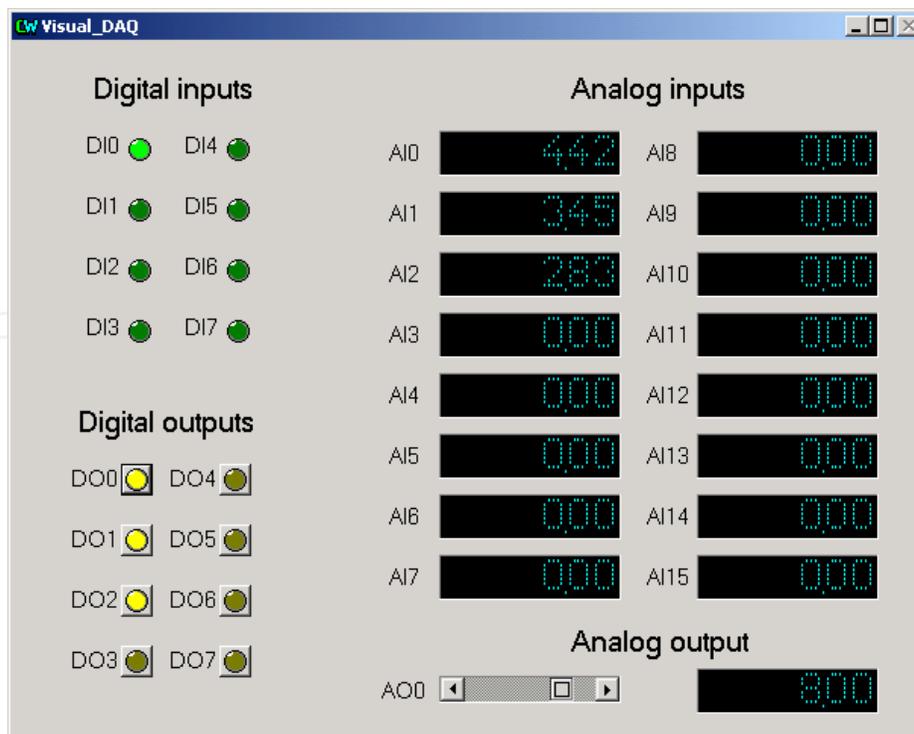

Fig. 13. DAQ test utility for Control Web

5. Verification and results

Designed device was tested on laboratory model of heating plant system with three temperature measurement channels with unified analog outputs 0 – 10 V and two digital input channels for heating element and auxiliary cooling fan control. Main cooling fan and circulation pump speed are controlled by analog inputs 0 – 10 V. DAQ is connected with standard personal computer via serial communications interface.

5.1 Heating plant model with time delay

Time delay systems are represented in many practical applications. Primarily it is in systems where matter is transferred along a certain path with a certain speed. An example of such systems is a belt conveyor apparatus, pipelines for transferring heat, dispensing systems used in chemical industry, etc. From the educational point of view it therefore appears to be useful to acquaint students with a laboratory model of this type of a system.

Educational heating-system model with time delay developed at our institute is based on the principle of transferring heat from source through a piping system using heat transferring media to heat-consuming appliance. The heat transferring fluid is transported using a continuously controllable DC pump 6 into a flow-heater 1 with 750W heat power. The temperature of the liquid leaving the heater is measured using a platinum thermometer T1. Warmed liquid then comes into a thermal insulated coil 2 which is composed of a 15m pipeline. Here a transport delay between 50 and 200 seconds originates depending on the pump speed. Heat exchanger 3 represents a heat-consuming appliance by releasing the thermal energy from the heat transferring fluid to the ambient air. The heat consumption can be set using two fans 4, 5 with adjustable speed. The temperature of the fluid entering and leaving the heat exchanger is measured by thermometers T2 and T3. Expansion tank 7 compensates for the thermal expansion effect of water. Schematic of the model is depicted in Fig.14, its photograph in Fig.15.
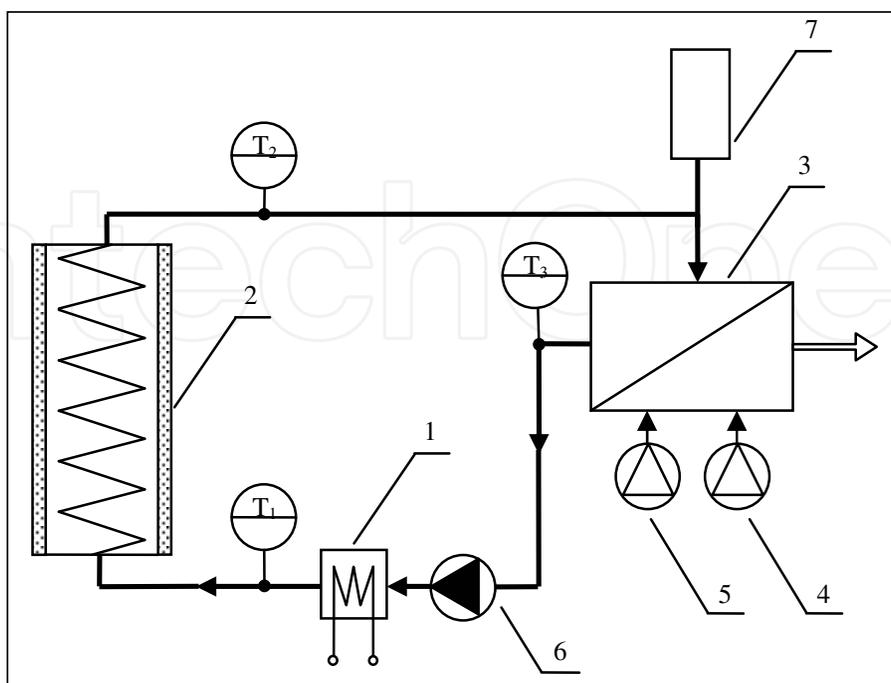

Fig. 14. Block schematic of the heating plant with time-delay

Fig. 15. Educational heating plant model with time-delay

5.2 Model control application for Control Web

Main window of the developed control application in Control Web 5 software environment
is depicted in Fig.16. The left part of the window contains simplified technology schematic
with two control components for pump speed and demanded temperature setting. Pump
speed can be set in the range from 0 to 100 % where 0 % means minimum allowable speed
(it can not be completely stopped due to possible heater damage). Demanded temperature
range is limited to maximum of 85 centigrade.

The right part of the window visualizes all monitored values – temperatures in 3 important
parts of the model, controller actuating signal and demanded temperature value. All these
values are stored in graph with history of 1000 values.

5.3 Model control application for Matlab 6.5

All the tasks related to control and monitoring of the time delay model are served by control
software with graphical user interface running in Matlab 6.5 environment. The software
supports step response measurement of the system, control of the selected controlled
variable using PS, PSD and general linear controller with disturbance introduction
possibility. To allow quick restoring of the time-delay model to initial conditions before next
measurement, cooling function was implemented. All measured data are automatically
saved to the workspace in the matrix form and to the user definable text file with format
suitable for import to spreadsheet processor.
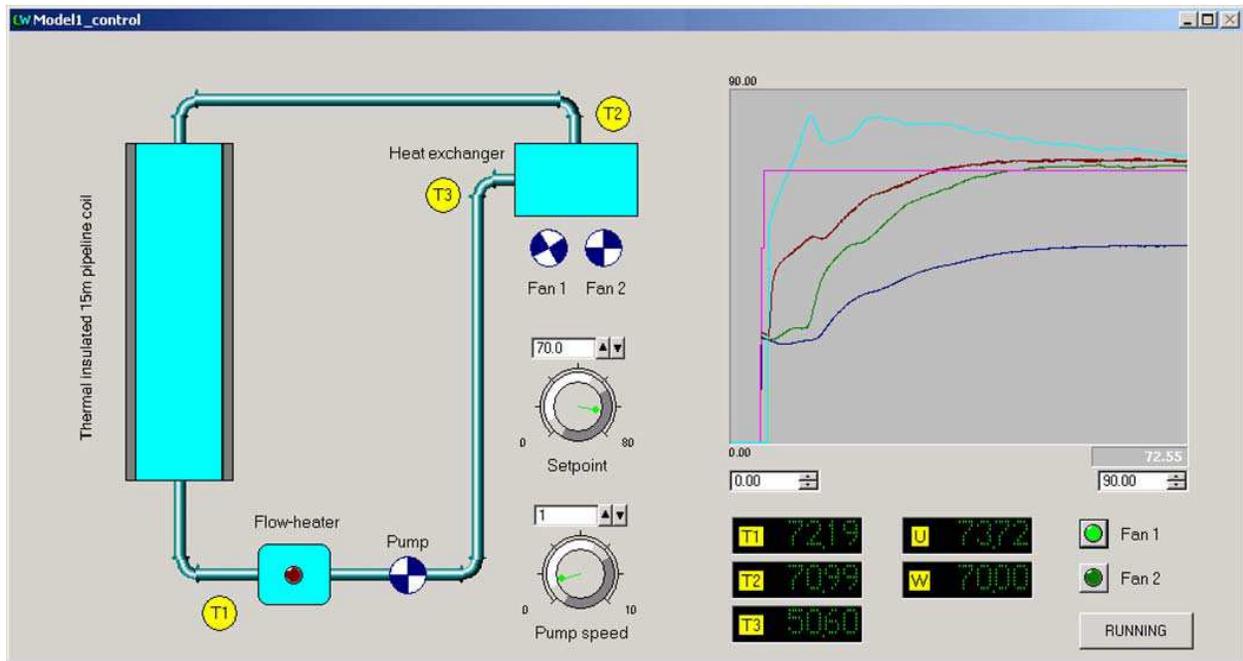
Fig. 16. Heating plant model control application for Control Web 5

After the program is executed by writing command "hmodel" in the command window of the Matlab 6.5 environment the main window depicted in the Fig.17 will appear. The window is divided into two parts – upper part is dedicated to displaying all measured system variables in the form of auto-scale graph.
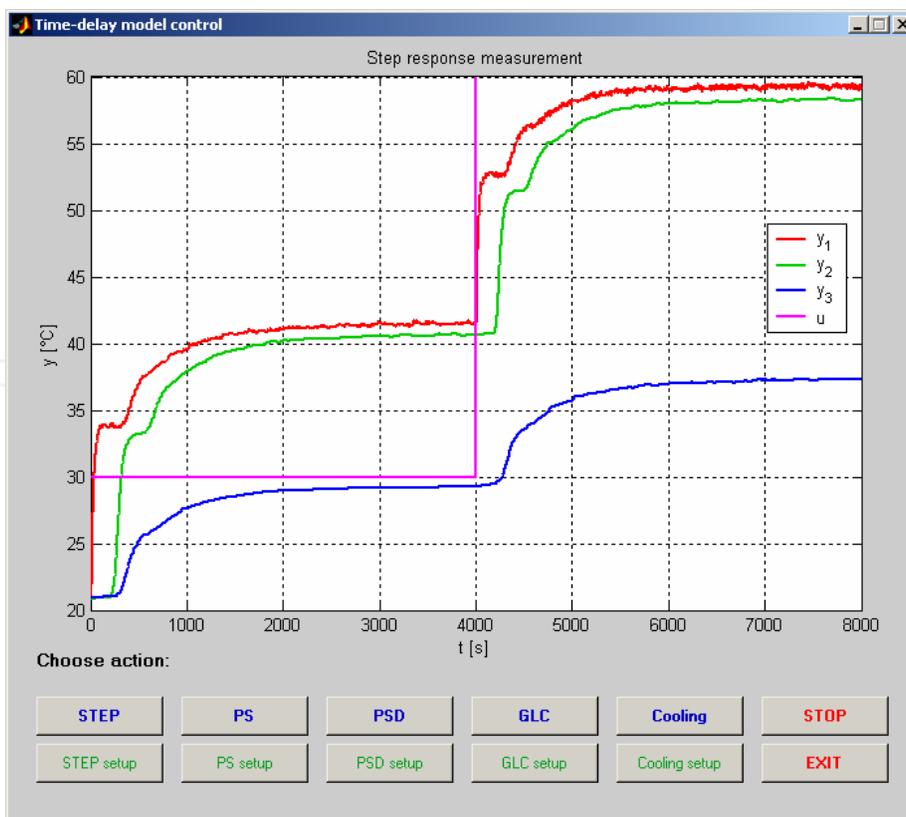

Fig. 17. Heating plant model control application for Matlab 6.5

Below the graph is situated block of buttons for selecting desired program action. These are divided into three categories depending on their function. Buttons with blue labels are used to startup program module which is performing each measurement – for example button "STEP" will initiate measurement of the step response. Green labeled buttons are intended to setup each program module. Button "STOP" interrupts current measurement with saving all measured data to file and workspace as well. Reaction to the stop command is not instantaneous – it can take up to one sampling period of the current measurement. When ALT+F4 key combination or close button is pressed during measurement, all actuating signals are immediately reset and application is closed without data saving.

Pressing "EXIT" button will close the program with resetting all actuating signals to zero values. During measurement all buttons except "STOP" button are disabled in order to prevent unwanted interference to the running experiment.

All measurements during program verification were passed in the following conditions: distilled water as heat transfer medium, pump speed at minimum (maximum time delay), fan 1 on, fan 2 at minimum speed and channel index was set to 2 (temperature before heat exchanger). Step response of the time delay model is depicted in Fig.18. It was measured with actuating signal change from 10 % to 60 % of the maximum value.
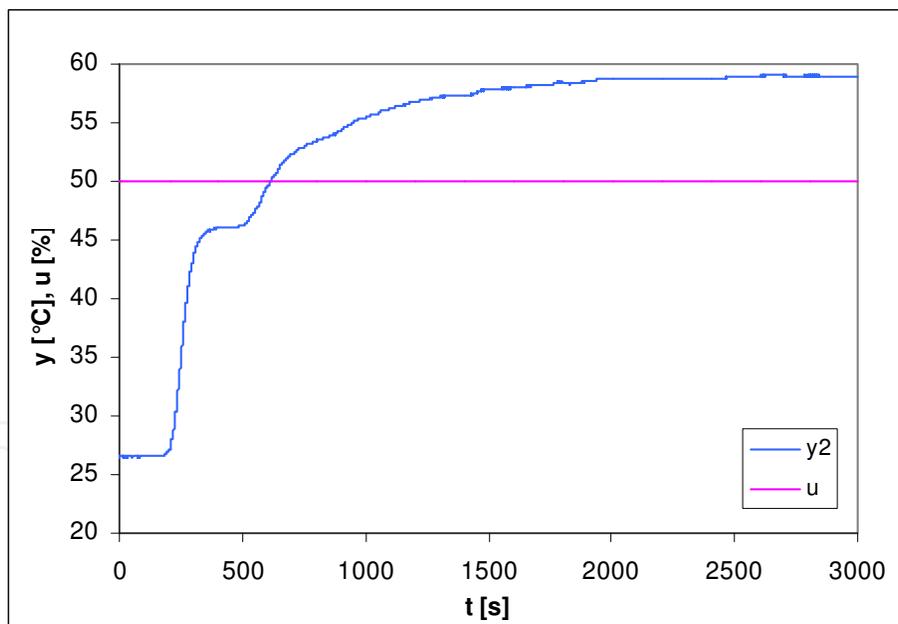

Fig. 18. Measured step response of heating plant model

System was identified with first order transfer function with time-delay (1) (Fig.19). Approximated first order transfer function with time-delay was used for PS controller design with inversion dynamics method published in (Vítečková, 2000). Parameters α and β

was chosen for 5% relative overshoot of the controlled variable. Final PS controller parameters for sampling period of 10 s are: $q_0 = 1.15$, $q_1 = -1.11$.
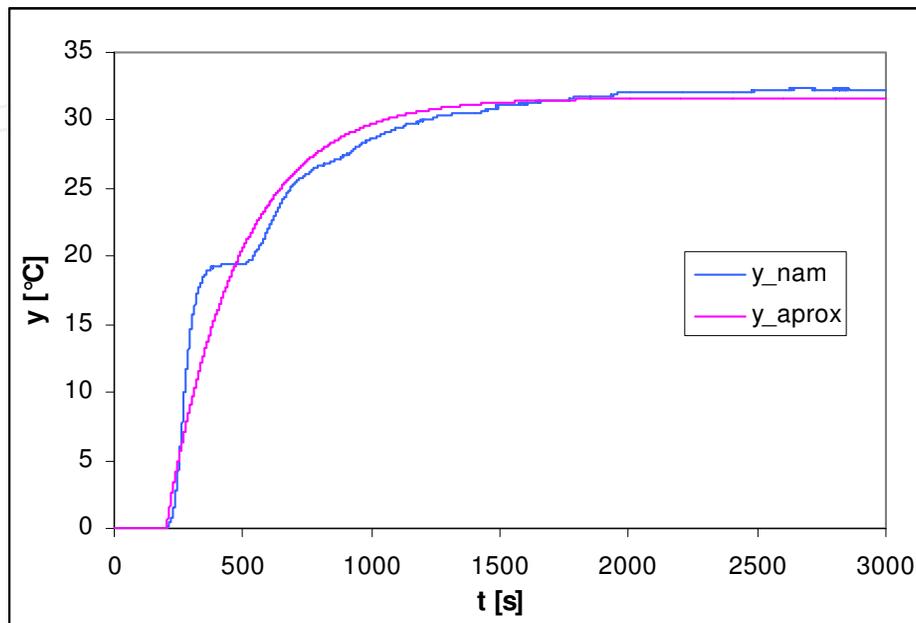

Fig. 19. Step response approximation with first order transfer function with time-delay
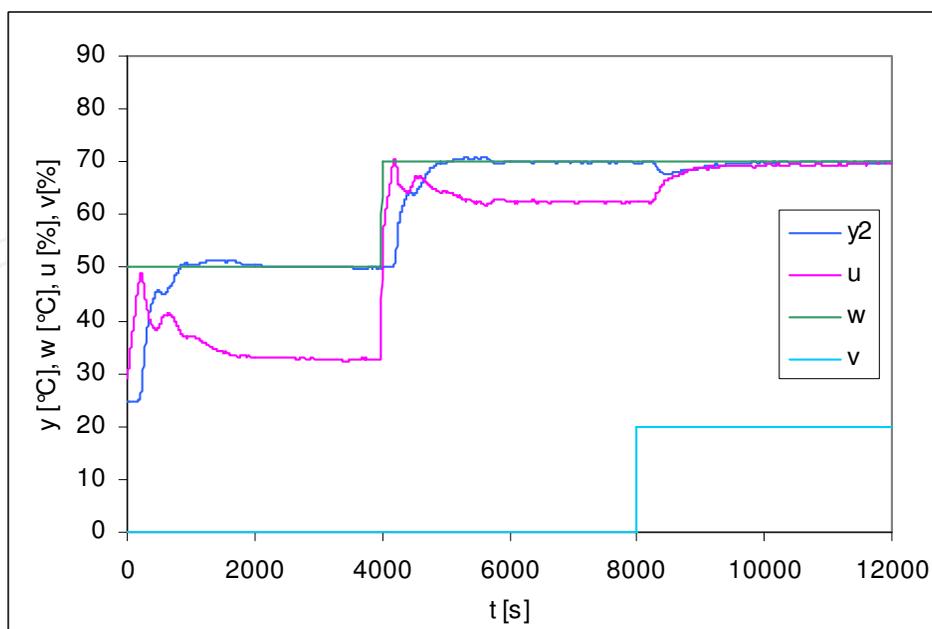

Fig. 20. PS controller control process

$$G(s) = \frac{k}{Ts+1} \cdot e^{-T_D s} = \frac{0.646}{290s+1} \cdot e^{-200s} \tag{1}$$

Resulting control process achieved with PS controller with set point values set to 50 °C at time 0 s and 70 °C at time 4000 s is in Fig.20.

6. Conclusion

The contribution deals with portable data acquisition unit which was developed at our department for control and monitoring related tasks. The device is designed with respect to possible battery operation enabling measurement in areas where power source is not available. It provides sixteen analog inputs with 12-bit A/D conversion resolution with input voltage range 0 – 10 V, eight TTL compatible digital inputs and outputs protected against electrostatic discharge and overloading and one analog output channel equipped with 12-bit D/A converter with output amplifier providing standard voltage output 0 – 10V. Communication with supervision system is realized via standard RS232 asynchronous serial interface which makes DAQ device fully platform independent. It uses universal ASCII-based communication protocol which can be easily successfully implemented in many control and monitoring software environments.

In order to improve development of new software applications with this device a support program libraries for Matlab/Simulink, Visual C++ and Control Web 5 were created. For research and educational purposes control software with graphical user interface running in Matlab 6.5 environment was developed. It supports step response measurement of the system, control of the selected controlled variable using PS, PSD and general linear controller with disturbance introduction possibility.

7. Acknowledgments

8. References

Burr-Brown. (1998). *DAC7611: 12-Bit Serial Input Digital-to-Analog Converter* [online]. 1st edition. Tucson : Burr-Brown, [cit. 2010-01-10]. Available from WWW: <http:// www.burr-brown.com/>.

Freescale. (2008). *M68HC08 Microcontrollers: MC68HC908GP32 Data Sheet* [online]. 1st edition. Chandler : Freescale Semiconductor, [cit. 2010-01-22]. Available from WWW: < http://www.freescale.com/>.

Freescale. (2006). *M68HC08 Microcontrollers: CPU08 Central Processor Unit Reference Manual* [online]. 1st edition. Chandler: Freescale Semiconductor, [cit. 2010-01-20]. Available from WWW: < http://www.freescale.com/>.

Linear Technology. (1994). *LTC1286/LTC1298 Micropower Sampling 12-Bit A/D Converters* [online]. 1st edition. Milpitas: Linear Technology Corporation, [cit. 2010-01-10]. Available from WWW: < http://www.linear.com/>.

Moravian Instruments. (2005). *Control Web 5 software documentation* [online]. 1st edition. Zlín:
        Moravian Instruments, [cit. 2010-01-15]. Available from WWW: < http://
        www.mii.cz/>.
Vítečková M. (2000). *Controller tunning by method of inverse dynamics*. Ostrava: VŠB –
        Technical University of Ostrava, 56p. ISBN 80-7078-628-0.

Data Acquisition

Edited by Michele Vadursi

The book is intended to be a collection of contributions providing a bird's eye view of some relevant multidisciplinary applications of data acquisition. While assuming that the reader is familiar with the basics of sampling theory and analog-to-digital conversion, the attention is focused on applied research and industrial applications of data acquisition. Even in the few cases when theoretical issues are investigated, the goal is making the theory comprehensible to a wide, application- oriented, audience.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Vladimir Vasek, Petr Dostalek and Jan Dolinay (2010). Microcontroller-based Data Acquisition Device for Process Control and Monitoring Applications, Data Acquisition, Michele Vadursi (Ed.), ISBN: 978-953-307-193-0, InTech, Available from: http://www.intechopen.com/books/data-acquisition/microcontroller-based-data-acquisition-device-for-process-control-and-monitoring-applications

INTECH
open science | open minds