Platform for teaching embedded programming

Dolinay J., Dostálek P., Vašek V. and Vrba P.

***Abstract*—**This article describes platform for school courses of programming embedded applications. It integrates several devices on one stand together with supporting software libraries and it is intended to provide easy-to-use platform for lessons. The devices included in the platform are microcontroller development kit, panel PC with Advantech I/O modules and several models of real-world systems. The platform should make it easier for the students to concentrate on the programming problems by providing documented and unified interface between the control system and controlled model of a technological process.

***Keywords*—e**mbedded programming, microcontroller, panel PC, teaching.

I. Introduction

EMBEDDED computer systems can be found literally all around us and their number increases rapidly. Even simple devices which would be made of discrete parts several years ago are now using microcontrollers and other highly integrated circuits. With the increasing usage of microcontrollers, or generally embedded computers, the need for qualified programmers also increases. Universities and colleges must prepare such experts, who will be able to design and program embedded systems efficiently and appropriately. Obviously, the subject is wide and the teaching process is more or less focused on certain parts of the field, based on experience and tradition of the department, etc. In general, it seems desirable to shift the focus from learning about the microcontroller itself to learning about how the microcontroller can be used as a tool to solve practical problems. The student should preferably be exposed to real-world engineering applications, not only to simpler applications [1].

In our courses we also consider it important to allow students to try their skills on a real hardware. In our experience using real devices, such as models of technological processes, real sensors and actuators and so on, makes the lessons much more attractive for students and also their results are better compared to lessons where only simulators and/or computer models are used. We also try to make it possible for the students to work with microcontrollers at home by providing materials and tools for developing their own embedded devices. [2], [3]. Such a simple and affordable device, which can be built by the student, can greatly improve the interest in the subject.

This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic under the Research Plan No. MSM 7088352102 and by the European Regional Development Fund under the project CEBIA-Tech No. CZ.1.05/2.1.00/03.0089).

At our department we teach microcontroller programming and also programming of embedded systems with the help of real-time operating systems. For these lessons we use several teaching aids (models, development kits, etc.), which will be described later, but the common problem we have is that these tools are not unified or compatible. Basically, there are some tools for microcontroller programming and different tools for real-time OS programming and most of these tools have different interfaces and different programming approaches. Last but not least reason is also the physical arrangement of these tools which are just placed on the table and connected to the computer and if there is another lesson of a different course in the classroom, the tools must be put away and later again prepared – which is time consuming. As a better solution we see to put the tools on a stand, which can be permanently connected to the table. The stand should hold all the required devices and prevent discomfort such as mixed cables or even damage of the devices. The content of the platform does not necessarily have to be invariable; it is better if there is option to change some components (e.g. model of a technological process), but the platform should provide common basic functions, such us connection to power supply and the programming interface and allow easy placement of variable components.

II. Design requirements

During the years that we teach microcontroller programming and real-time programming courses, we have used and developed many teaching aids, some of which are now obsolete and some of which are still used. At the beginning we used Motorola 68HC11 microcontroller in a kit containing keyboard, display and programming interface. This kit was also connected to a model of a technological process – a heating plant, which allowed students to develop some real-world applications for measuring and controlling the temperature. This kit was a good tool, but as the 68HC11 was aging we decided to move to newer hardware. However, we found no commercially available ready-to-use device equivalent to the older one. We decided to use M68EVB908GB60 evaluation kit with Freescale HCS08 microcontroller. The kit is equipped with LCD display, push switches, LED diodes and a buzzer. See fig. 1. It contains HCS08 GB60 microcontroller with 60 kB of FLASH memory for the program and 4 kB of RAM memory. The kit is quite well suited for our purpose, but it has also some disadvantages, for example, it is not well protected from damage by users and

its price is too high to be affordable for students to use at home in their own projects. Nevertheless, this seems to be the best option for us from the commercially available development kits, so we plan to stick with it for some time.

We also developed several extension modules which can be attached to this kit, for example, 7-segment, 4 digit multiplexed display and keyboard, DC motor drive and simple heating plant [4].

The kit together with modules allows developing of wide range of applications but it also has some limitations such as that each of the extension modules has more or less original design and software interface, the connection of the module to the kit is fragile and the kit itself is not well protected from damage when just lying on the table. With the experience from using this kit we formulated the following requirements on a new platform for our courses:

- Single robust stand to hold all the components
- Possibility to attach different models of real-world systems through the same interface
- Option to use either microcontroller system or a panel PC to control the models
- Preferably use only commercially available parts for the design; avoid custom-made components

With these requirements in mind the system described in the following chapter was designed.

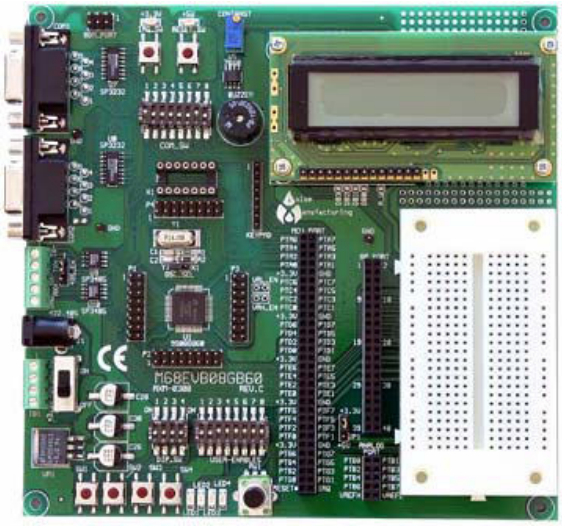

Fig. 1 microcontroller development kit used in lessons

III. Platform Design

This chapter describes the proposed system for teaching courses of real-time programming both on microcontrollers and on PC-based devices (panel PC).

All the devices should be organized on a stand which can be placed on the table and replace the bunch of cables and boards used now. The requirements on the stands are relatively simple –good stability, enough space for the devices and low price. Prototype stand can be seen in fig. 2, but there may be changes to this design later.

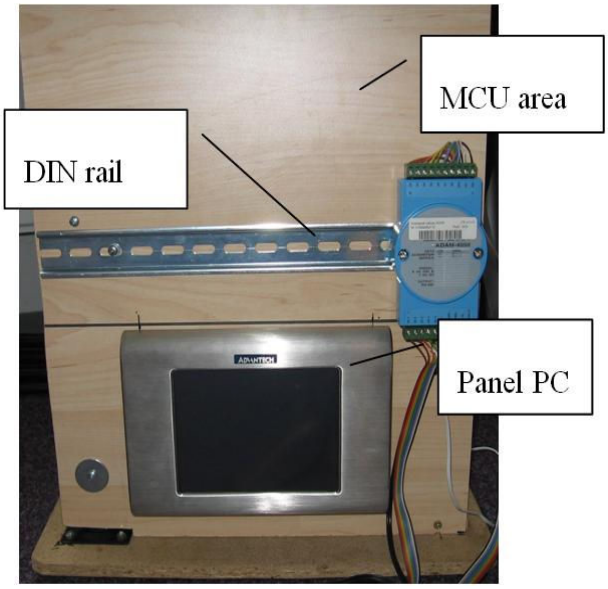

Fig. 2 stand prototype

The stand itself is formed by two wooden boards and a "leg" which is attached to the basement board and holds the front board. The basement board has four rubber spacers on the bottom side to prevent it from sliding on the table. The front board then contains all the equipment, which is attached either directly to the board (the MCU unit), or to a DIN rail (I/O modules and models of real world systems) or placed in a hole in the board (panel PC).

The contents of the stand are described in the following sections.

A. Devices used in the platform

The proposed layout of the devices can be seen in fig. 2. The most visible device is the panel PC which takes up the bottom part of the stand. At the top part there should be the microcontroller unit (for now, M68EVB908GB60 evaluation kit, but with possible replacement in future). Below the MCU unit there is DIN rail for placing models of technological processes and I/O modules Advantech ADAM (used for interfacing the models from panel PC).

Fig. 3 panel PC

The microcontroller kit was described in the previous section. The panel PC can be seen in fig. 3, its parameters are summarized in table 1. It is Advantech PPC-L60T, which was already available at our department.

Table 1 parameters of the Panel PC

CPU	Via Eden CPU running at 400 MHz
RAM Memory	128 MB
Storage	80 GB hard-drive
Screen	6.4“ TFT LCD touch screen
Ports	One RS232 One RS232/RS422/RS485 One USB One VGA One 1 Ethernet 10/100Base-T port

In fact, the choice of the PC is rather arbitrary; it could be a different device from different manufacturer. Our requirement was only to have a PC-based device for courses where multitasking real-time programing on such systems is taught and to be able to control the models of technological processes with this device.

In place of the models of technological processes we intend to use commercially manufactured models intended primarily for teaching PLC programming, but usable also for controlling from microcontroller. These models actually simulate the controlled object (such as wash machine or mixing unit) and provide feedback to the controller system by their outputs and also visually to the user by LEDs. The models use 24V logic for inputs and outputs, so voltage adjustment is required to connect them to the MCU unit. For connections to Panel PC we use Advantech modules, which will be described later, and these modules can work directly with the 24V levels, so no adjustment is necessary. Model of washing machine can be seen in fig. 4 and model of mixer unit in fix. 5. In both the figures is can be noted that there are the LED indicators which show the status of the object, such as the temperature in washing machine (30, 40, 60, 90 degrees C). The status is read by the controlling unit in the form of binary input. For example, there are four binary outputs from the model which report the temperature. If the temperature reaches 60 °C, the corresponding output will read logical 1. On the other hand, when the controlling unit turns on/off some function of the model, e.g. switches on the pump in the washing machine, (by driving the appropriate model input high) the corresponding LED on the model will turn on.

Besides these models we intend to develop our own models also. In the first place, this will be a simple heating plant similar to the one described in [4]. Such models are used at our faculty not only in lessons of programming, but also in lessons on theory of control, where students measure the transfer functions of these plants and verify their design of controllers. It seems to be useful part of the educational process – to allow the students to see several aspects of the use of such model.

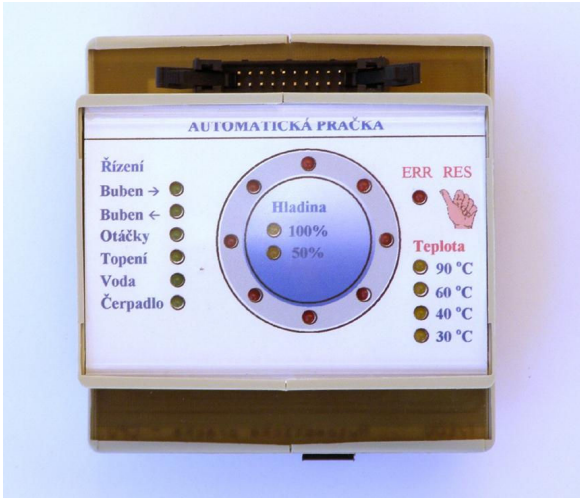

Fig. 4 model of washing machine

Fig. 5 model of mixer unit

One aspect is being a hardware designer and programmer, who needs to create the hardware and software of the control system for such a model, which is essentially the same as creating control system for a real system), and secondly as the user of the device, who does not have to deal with the hardware and software of the system, but has to design and control system itself (choose the proper controller and set its parameters).

Besides the model of heating plant we want to create also some other models, possibly more attractive models, such as DC or servo motor control, etc.

B. Connection of the devices

Now that we have described the devices used in the platform we can move on to the connection between these devices.

The connection can be divided into two main parts:

- Connection of the platform to the computer on which the software is developed (development PC).
- Connection between the models of real-world systems and the control units (panel PC or MCU unit).

All the connections can be seen in fig. 6.

For connecting with the development PC on each workplace, the MCU unit uses its standard programming interface.

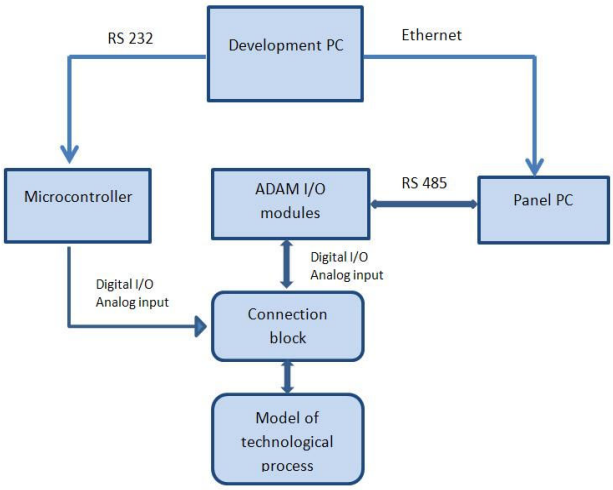

Fig. 6 connection of the components

This interface is RS232 line for the M68EVB908GB60 evaluation kit. It could be USB connection in future when we move on to a new MCU development kit.

The panel PC will be connected with the development PC via Ethernet.

For interfacing the panel PC with the technological processes there are analog and digital I/O modules Advantech. The following modules are used:

- ADAM-4050 - digital inputs and digital outputs.
- ADAM-4017 – analog inputs

The ADAM-4050 features seven digital input channels and eight digital output channels. The outputs are open-collector transistor switches that can be controlled from the computer, for example to control solid-state relays, which in turn can control heaters, pumps and power equipment. The computer can use the module's digital inputs to determine the state of sensors with digital outputs, such as safety switches or remote digital signals [8].

The ADAM-4017 is a 16-bit, 8-channel analog input module that provides programmable input ranges on all channels. This module is offered as a cost-effective solution for industrial measurement and monitoring applications. Its opto-isolated inputs provide 3000 VDC of isolation between the analog input and the module, protecting the module and peripherals from damage due to high input-line voltages. ADAM-4017 offers signal conditioning, A/D conversion, ranging and RS-485 digital communication functions.

The ADAM-4017 uses a 16-bit microprocessor-controlled sigma-delta A/D converter to convert sensor voltage or current into digital data. The digital data is then translated into engineering units. When prompted by the host computer, the module sends the data to the host through a standard RS-485 interface [9]. There is also a “plus” version, ADAM 4017+, which supports Modbus communication protocol.

Both the modules can be seen in fig. 7 and their parameters are summarized in tables 2 and 3.

Fig. 7 advantech ADAM modules

Table 2 parameters of the Advantech ADAM-4050

Inputs	7 digital inputs
Outputs	8 digital outputs open collector Power dissipation 300 mW
Watchdog timer	Yes
Power supply	10 – 30 VDC

Table 3 parameters of the Advantech ADAM-4017

Inputs	8 channels, independently configurable
Input current ranges	4-20 mA +/- 20 mA
Input voltage ranges	0 – 150 mV 0 – 500 mV 0 – 1 V 0 – 5 V 0 – 10 V
A/D converter	16-bit
Isolation	3000 VDC
Power supply	10 – 30 VDC

These ADAM modules are interconnected into RS485 bus together with the panel PC. There is software library for accessing the I/O modules from a student’s programs written in C/C++ which was developed in a diploma thesis [5] and will be described later.

The other side of the ADAM modules is connected to a unified connector which allows connection to the models of technological processes. This connector is the same as the output connector from the MCU unit, so that it is easy to control the models by either the panel PC or by the MCU unit by simply plugging the appropriate connector from the control system into the connector of the controlled model.

One part of the connection is also a connection unit for the ADAM modules. The scheme of this unit can be seen in fig. 8 and the real unit in depicted in fig. 9.

This unit adjusts the signals to and from the model to the ADAM I/O module and also distributes the power supply

voltage to the model and to the ADAM modules. It thus makes the connection mode well-arranged and variable. There is one 20-wires cable from the unit to the model of technological process with standard connector for these models.

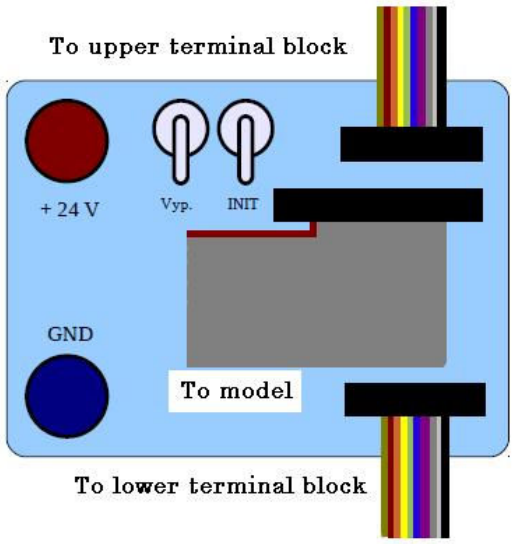

Fig. 8 scheme of the connection unit for the ADAM modules

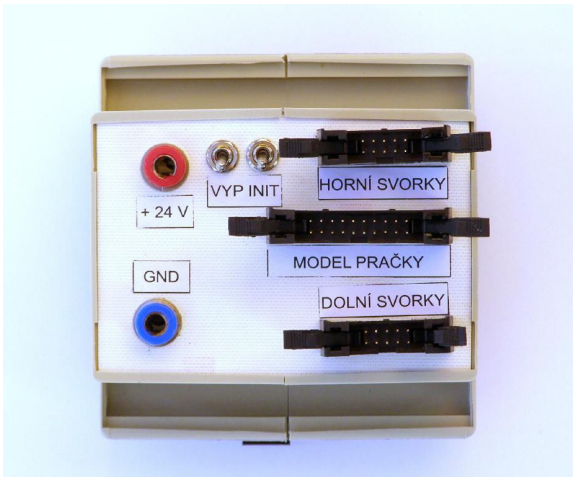

Fig. 9 the connection unit for the ADAM modules

From the unit to the ADAM module there are two 10-wires cables which go to the lower and upper terminal blocks on the ADAM module (fir. 10). All the connectors use locks so that they cannot be connected in a wrong way.

The power supply (24 V) is connected using round connectors which can be seen on the left side of the unit. The voltage is then distributer to the ADAM module and to the model via the cables described above – to the standard pins as required by both the devices.

The connection is currently realized only for the model of the washing machine; the pins are summarized in the tables 4, 5 and 6. However, pin configuration of the ADAM module will be the same for all other models, so the tables 4 and 5 apply for all models.

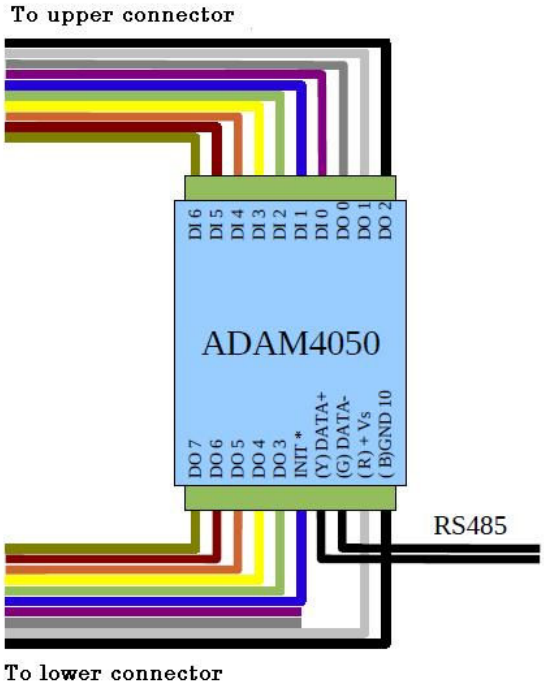

Fig. 10 wiring of the ADAM-4050 module

The pin configuration of the models of technological processes will have different meaning for different models, but physically it will be the same.

Table 4 – pin configuration for the ADAM-4050 module, upper terminal

ADAM-4050 signal	Direction	Meaning	Pin number
DI6	Input	NC	1
DI5	Input	Level 50%	2
DI4	Input	Level 100%	3
DI3	Input	Temperature 90 °C	4
DI2	Input	Temperature 60 °C	5
DI1	Input	Temperature 40 °C	6
DI0	Input	Temperature 30 °C	7
DO0	Output	Rotate right	8
DO1	Output	Rotate left	9
DO2	Output	High speed rotation	10

As mentioned earlier for connection of the MCU unit with the models, it is necessary to perform voltage conversion because the MCU unit uses 3.3 V logic, while the models use 24 V logic. The converter block intended for this purpose was started in the course of a bachelor's thesis, but is it not finished as of now. The prototype uses resistor dividers for inputs to the microcontroller to bring the 24 V level to 3.3 V, and transistor switches to operate the 24V outputs from the converter by the 3.3 V outputs from the microcontroller. In future we would like to implement more advanced version of this converter with opto-isolation.

Table 5 – pin configuration for the ADAM-4050 module, lower terminal

ADAM-4050 signal	Direction	Meaning	Pin number
DO7	Output	NC	1
DO6	Output	Relay	2
DO5	Output	Draining	3
DO4	Output	Filling	4
DO3	Output	Heating	5
Init	Input	Initialization	6
Data+ (Y)	-	RS485	7
Data- (G)	-	RS485	8
+Vs (R)	Input	Power supply (+)	9
GND (B)	Input	Power supply (-)	10

Table 6 – pin configuration for the model of washing machine

Pin	Direction	Meaning	I/O assignment
1	Input	GND	-
2	Input	+24 V	-
3	Input	Rotate right	DO0
4	Input	Rotate left	DO1
5	Input	High speed rotation	DO2
6	NC	-	-
7	Input	GND	-
8	Input	+24V	-
9	Input	Heating	DO3
10	Input	Filling	DO4
11	Input	Draining	DO5
12	NC	-	-
13	NC	-	-
14	NC	-	-
15	Output	Level 50%	DI5
16	Output	Level 100%	DI4
17	Output	Temperature 90 °C	DI3
18	Output	Temperature 60 °C	DI2
19	Output	Temperature 40 °C	DI1
20	Output	Temperature 30 °C	DI0

IV. SOFTWARE FOR THE PLATFORM

Besides the hardware described above, there are also software components of the educational platform. These components should make it easier for students to create their own programs. So far a library was created for communicating with the ADAM modules from program running on the Panel PC and also several helper functions for controlling the models from MCU evaluation kit. We will focus on the library for communication with the ADAM modules via RS485 interface first.

The library is designed in three levels; the lowest level communicating with the hardware and each higher level using the functions offered by the lower level.

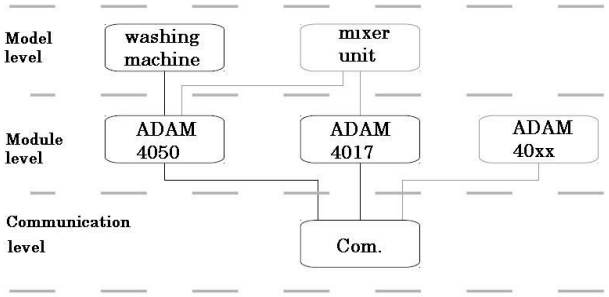

Fig. 11 three-level design of the software library

The design of the library is depicted in fig. 11. As can be seen, at the lowest level, called communication there is one module. This module handles communication via the RS 485 interface using the protocol of ADAM devices. It provides generic functions such as send command or read reply, which are common for all ADAM-40xx devices.

On the middle level, called module level, there are functions specialized for given ADAM module, that is ADAM-4050 and ADAM-4017. These functions provide functions such as writing digital output value for ADAM-4050 or reading analog input from ADAM-4017.

On the top level – model level – there are functions specialized for given model of technological process. Currently only functions for the washing machine are implemented but the library is ready for extension to other models. The functions provide high-level operations on the model, such as “turn heater on” or “get water level”.

Obviously, not all functions may be available to the students. For example, for simple tasks with focus on the control algorithms it may be usable to provide the students with ready-to-use functions at the highest level, so that they can control the model without thinking about communication with the I/O modules or even setting inputs and reading outputs. They can simply use ready functions to turn heater on, etc. and concentrate on the algorithms of controlling the model. On the other hand, for more advanced students it may be enough to provide basic I/O functions such as “set output N to high” or “read input N state” and let the students implement their own higher-level functions and use them in their program.

Let’s now look at the parts of the library in more detail.

A. Communication level

This part of the library is used to handle the serial communication on Windows operating systems. It allows opening and closing the ports and so on. Most of the communication parameters are fixed; their values are required by the ADAM modules. In the user programs it is possible to set the communication speed.

The communication uses protocol ADAM, which is used by Advantech for their ADAM modules. Optionally, there are versions of the modules which communicate using Modbus protocol, but in our case the modules with ADAM protocol are utilized.

The ADAM protocol is plain text protocol with master-slave support which allows data transfer between the ADAM modules and control system (computer). The protocol defines unified command common for all the modules which can be extended by command specific for given module.

The basic set of commands contains commands for identification and parameterization of the modules. The extended set of commands then contains commands specific for each module, such as setting output value.

Each command is terminated by CR character (ASCII code 13).

For example, the command for sending identification string from the device is $AA2<CR>.

The meaning of the characters is:

- $ - leading character
- AA – the address of the device for which the command is intended – as a hexadecimal number.
- 2 – the command itself
- <CR> - the terminating character

Given this information, an example of this command intended for device with address 1 could be:
$012<CR>.
The device replies with message in the following format:
!AATTCCFF<CR>
Where the meaning of the characters is:

- ! – leading char indicating reception of a valid command
- AA – the address of the device which sent the message
- TT – two characters indicating the type of the code sent by the device. Different devices send different codes here.
- CC – code of the communication speed
- FF – various flags describing the state of the device
- <CR> - terminator

Alternatively, if the device does not recognize the command, it can send the following answer:

?AA<CR>

The meaning is quite clear now: ? means that the device does not know this command. AA is the address of the device and <CR> is the terminator character.

Another basic command is to ask the device to send its name. This command has the following form:

$AAM<CR>

The meaning of the parts being:

- $ - leading character
- AA – the address of the device for which the command is intended – as a hexadecimal number.
- M – the command to send identification
- <CR> - the terminating character

The device then replies with message:

!AA(module name)<CR>

The meaning of the message is now quite clear. The (module name) will be, for example, “ADAM4050” for the ADAM-4050 module.

An example of a specific command, which is used by modules with digital outputs, is the command to send the current state of the inputs and outputs of the module:

$AA6<CR>

As can be seen the command follows the general structure described above. The command code is 6.

The answer from the device, if it is ADAM-4050 is in the form of:

!(data_output) (data_input)00<CR>

Where :

- ! indicates that valid command was recognized by the device
- (data_output) are two characters which represent the value of the output register of the device in hexadecimal numbers.
- (data_input) are two characters which represent the value of the input register of the device in hexadecimal numbers.
- 00 – reserved characters
- <CR> - terminator

As can be seen from the above examples, all the values are sent in text representation. If a numerical value is sent, it is sent as two-character string containing the number represented in hexadecimal digits.

B. Module level

This part of the library is used to control the ADAM modules. For the washing machine only the ADAM-4050 module is needed (binary I/O), but there were also functions for the ADAM-4017 (analog inputs) implemented.

The functions in this part allow changing the outputs of the module and reading the inputs. The user does not need to know the details of the communication with the ADAM modules; he/she is shielded from this by the library.

The functions handle initial communication with the ADAM module, including verification that it is the right type of module, setting the address on the RS485 bus, handling inputs and outputs and correct closing of the communication. In all this the functions from the lower, communication level are utilized by the functions on this level.

C. Model level

The functions contained in this part of the library are focused on direct control of the washing machine model. In future there should be equivalent functions created also for the other models. The user can take advantage of functions which turn on/off the inputs of the model and read its outputs, e.g. turn on heater or water pump or read water level. All the physical connections and lower level implementation such as opening port and sending appropriate commands are hidden from the user. This level uses the functions from the lower, module level of the library.

D. Example of the program

To illustrate the use of the library we will show simple program for controlling the washing machine. The code is written in C with the use of the model library. See fig. 12.

```
#include "pracka.h"
int main(int argc, char * argv[])
{
  P_PRACKA model;
  model = PRACKA_inicializace("COM1\0", 9600, 1);
  if(model == NULL) {
    return 0;
  }
  if(!PRACKA_napousteni_zapnout(model)) {
    PRACKA_zavri(model);
    return 0;
  }
  char snimac;
  do
  {
    if(!PRACKA_voda_50(model, &snimac)) {
      PRACKA_zavri(model);
      return 0;
    }
  } while (!snimac);
  if(!PRACKA_napousteni_vypnout(model)) {
    PRACKA_zavri(model);
    return 0;
  }
  PRACKA_zavri(model);
  return 1;
}
```

Fig. 12 simple example program utilizing the library

The #include directive includes the model library for the washing machine "pracka.h".

In the main function we first create variable "model" which will hold the handle to the washing machine object. This variable is of type P_PRACKA, which is defined in the library and represents a washing machine object.

Then the library is initialized by calling PRACKA_inicializace() function. The input parameters are quite self-explanatory. The variable "model" will receive handle to the washing machine object or NULL if the initialization fails. In the case of failure the program is terminated.

Once we have valid handle to the model, we can call other functions operating on the model. In the example, function PRACKA_napousteni_zapnout() is called, which starts filling the washing machine with water.

If any of the library functions fails, it returns 0, which should be tested in the program, as can be seen in the example. Note that in case of error the program not only terminates, but first it correctly closes the connection with the model by call to PRACKA_zavri().

The sample program continues by testing the level of water in the washing machine. As can be seen, it calls function PRACKA_voda_50() which sets the variable "snimac" to 1 or 0 depending on the value read from digital input. When the washing machine reports that the water level reached 50 percent level, the do loop is terminated and the filling of water is stopped by calling PRACKA_napousteni_vypnout(). The program then closes connection with the model and terminates.

V. Conclusion

In this article we described teaching aid – platform for courses of embedded systems programming. This platform should make it easier to teach such courses by organizing all needed devices on one stand and unifying the connection between the control device (computer system or microcontroller) and the controlled device (model of a real process). It is targeted primarily on lessons which deal with programming control systems, such as heating plant or motor control.

The platform consists of a microcontroller unit (MCU evaluation kit), an industrial computer (panel PC), several models of real-world processes, I/O modules and other supporting circuitry for connecting the computer systems to the models.

Included in the platform is also software equipment in the form of libraries of easy-to use functions, which should make it easier for the students to write their own programs. However, these libraries still need to be extended, improved and tested before they can be utilized in the course.

Currently the platform is in the process of development of a prototype. The basic structure and components are defined, but still there is much work left before the platform can be finalized both in the hardware and the software part.

References

[1] Hamrita, T. K., McClendon, R. W., A New Approach for Teaching Microcontroller Courses, International Journal of Engineering Education, Vol.13, No.4, 1997, pp. 269-274.

[2] Dolinay, J., Dostalek, P., Vasek, V., Simple and Cheap Microcontroller Kit for Students, Annals of DAAAM for 2010 & Procceeding of the 21st International DAAAM Symposium, Vienna 2010, pp. 0259, ISSN 1726-9679.

[3] Dolinay, J., Dostalek, P., Vasek, V., Implementation and Application of a Simple Real-time OS for 8-bit Microcontrollers, In procceedings of the 10th WSEAS International Conference on Electronics, Hardware, Wireless and Optical Communications (EHAC'11), Cambridge 2011, pp.023-026, ISSN1792-8133.

[4] Dolinay, J.; Dostalek, P. & Vasek, V. Educational models for lessons of microcontroller programming, Proceedings of 11th International research/expert conference TMT 2007, pp. 1447-1450, ISBN 978-9958-617-34-8, Tunisia, September 2007, Hammamet.

[5] Vrba, P. Connection of Industrial PC with Models of Technological Processes, Diploma thesis at TBU in Zlin, TBU Zlin 2010.

[6] Dolinay, J.; Dostalek, P.; Vasek, V & Vrba, P. Teaching Platform for Lessons of Embedded Systems Programming, Proceedings of 13th International conference on Automatic Control, Modelling & Simulation ACMOS '11, pp. 158-160, ISBN 978-1-61804-004-6, Spain, May 2011, Lanzarote.

[7] Edu-mod Educational models [Online]. Available: http://www.edumat.cz/produkty.php?produkt=edumod

[8] ADAM/4050 product description [Online]. Available: http://www.advantech.com/products/ADAM-4050/mod_02B0D2AD-0BBB-498A-8647-B910CE345125.aspx

[9] ADAM-4017 product description [Online]. Available: http://www.advantech.com/products/ADAM-4017/mod_170C40F4-E6AC-485E-9DF9-1E6EF60F971F.aspx

[10] Dostalek, P.; Vasek, V & Pekar, L. "Self-Tuning Digital PID Controller Implemented on 8-bit Freescale Microcontroller", *International Journal of Mathematical Models and Methods in Applied Sciences*, Issue 4, Vol 4, pp. 274-281, ISSN 1998-0140, 2010.