ARM-based Microcontroller Platform for Teaching Microcontroller Programming

Jan Dolinay, Petr Dostálek and Vladimír Vašek

***Abstract*—**This paper describes new microcontroller platform based on 32-bit ARM Cortex M microcontroller. This platform was developed at our faculty to replace older board with Freescale HCS08 microcontroller for use in lessons of microcontroller programming. Main component of the platform is a low cost evaluation board with Arduino-compatible pin layout FRDM-KL25Z. Both the hardware and software used for the course are described in this paper.

***Keywords*—**ARM, Arduino, FRDM-KL25Z, HCS08, Kinetis, microcontoller.

I. Introduction

THERE is no commonly accepted hardware and software platform for teaching microcontroller programming and related subjects. There are hundreds of types of microcontrollers from many manufacturers which can be used for this task. Typically, each school makes its own decision about the platform. Some use factory-made evaluation boards; some create their own custom boards.

However, in recent years there are two words which appear more and more in this area – Arduino and ARM. Arduino is a prototyping platform with microcontroller which allows very easy programming. The platform is composed of the hardware (board with microcontroller), program library for this microcontroller and integrated development environment (IDE) [1]. The Arduino has become de facto standard for prototyping and is very popular in the do-it-yourself community. It is also increasingly popular as a tool for teaching programming of embedded systems. The opinions whether this platform is suitable for such purpose vary greatly, from very positive to very negative [2], [3], [4]. In summary, it can be said that the platform is really easy to use and has wide community of users worldwide which makes it suitable for the task. The students are able to write programs with interesting outputs with less effort than with a typical microcontroller platform. This significantly improves their motivation to learn and they are more likely to become really interested in the subjects and create their own projects [2]. Also the availability of large number of example programs, libraries and documentation, created by the community of Arduino users represents great benefit of this platform – the learners are not dependent on the teacher and course materials as the only source of information but are able to find solutions on their own in the community. They can also easily explore any topic of interest in more details than it would be possible to cover in course materials.

On the other hand, the skills students gain with Arduino are not easily transferable to other microcontroller platforms [4]. The Arduino programming interface (API) is very easy to use, but does not adhere to any consistent standard or design pattern. In our opinion, given the current state of the industry, a graduate who intends to work with embedded systems professionally needs to understand and be able to handle the microcontroller (MCU) at the level of peripheral registers. This is because this is still the only common denominator for controlling peripherals of any MCU from any vendor. The knowledge of specific software library or graphical configuration tool is applicable only as far as this library or tool is widespread in the industry. And generally there is no such library or tool with really wide use except perhaps the CMSIS standard for ARM microcontrollers [5]. ARM is a company which designs computer processors and licenses these designs to the manufacturers (vendors), who actually produce the processors. The ARM architecture is dominating also the market with microcontrollers. This can be seen in the shift from 8-bit and 16-bit to 32-bit MCUs in recent years, with the 32-bit MCUs being more and more based on ARM technology. According to [6] the 32-bit segment became the leading segment of the MCU market in 2010. This is based on the amount of sales. If the number of units is compared, then the 16-bit MCUs are dominating the market followed by 4 and 8-bit MCUs. The number of 32-bit MCU units sold in 2012 was at about 50 percent of that of 16-bit MCUs. By 2017, 32-bit MCUs are expected to account for 55 percent of microcontroller sales. In terms of unit volumes, 32-bit MCUs are expected to account for 38 percent of microcontroller

This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic within the National Sustainability Programme project No. LO1303 (MSMT-7778/2014) and also by the European Regional Development Fund under the project CEBIA-Tech No. CZ.1.05/2.1.00/03.0089.

Jan Dolinay is with the Department of Automation and Control Engineering, Tomas Bata University in Zlin, Faculty of Applied Informatics, nám. T. G. Masaryka 5555, 76001 Zlín, Czech Republic (e-mail: dolinay@fai.utb.cz).

Petr Dostálek is with the Department of Automation and Control Engineering, Tomas Bata University in Zlin, Faculty of Applied Informatics, nám. T. G. Masaryka 5555, 76001 Zlín, Czech Republic (e-mail: dostalek@fai.utb.cz).

Vladimír Vašek is with the Department of Automation and Control Engineering, Tomas Bata University in Zlin, Faculty of Applied Informatics, nám. T. G. Masaryka 5555, 76001 Zlín, Czech Republic (e-mail: vasek@fai.utb.cz).

shipments in 2017.

This all means that ARM should become also the standard for engineering education and it does seem to be becoming such standard, at least in terms of the hardware. As for software, there is ARM standard for parts of embedded applications called CMSIS. However, CMSIS cannot be considered a complete framework for developing embedded software yet. It is not fully supported by many MCU vendors and it does not cover all the areas needed to create complete program. So at least for the time being this cannot be used as the common denominator for teaching MCU programming skills.

As already mentioned, the Arduino can be considered a standard of its own, but it mostly applies to the do-it-yourself community. It is seldom used (and usable) in industrial applications.

However, it is true that Arduino represents sort of a standard in the hardware layout of a microcontroller prototyping board and microcontroller vendors now often provide their evaluation boards with pin layout compatible with the Arduino boards. This is also the case of the FRDM-KL25Z board used in our design [7].

The above mentioned facts play key role in considering what tools to use when planning MCU programming course, together with the purpose of the course, i.e. the intended future career of the course participants.

At our faculty we have two courses which deal with microcontroller programming for two different study programs. First course is for students who specialize in security technologies and management and the second course is for students of information technology. For the first course Arduino platform is suitable according to our opinion. It offers the right tools to fulfill the aim of the course, which is to introduce the students into embedded systems and show them the possibilities. With easy-to-use platform like Arduino students can quickly build projects which have interesting, real-world outputs and motivate them to learn more.

The second course should produce future software engineers able to deal with microcontrollers at more general level. For such course we believe more traditional approach with C language is more suitable. We needed to create hardware tool for this course which would allow such approach while keeping up with the latest developments in the industry. The results are discussed in the next section.

II. HARDWARE DESIGN

As mentioned in the introduction, we have two different courses of microcontrollers programming at our faculty with two different purposes. Recently, a decision has been made to update the equipment and contents of these courses to reflect new trends in the industry. Previously, we used the same tools in both courses. While this has obvious advantages of easy maintenance of the equipment and course materials, it was also clear that the lessons suitable for students with strong programming background are too difficult for the students of the non-programming, security-technology program. These students were often put off by the difficulties of working with the full-featured IDE and programming in C language. Even though we provide supporting libraries which simplify access to the peripherals, for many students this was not sufficient and their motivation to learn was low.

A. *Evaluation kit with HCS08 microcontroller*

In previous years we used evaluation kit M68EVB908GB60, which can be seen in figure 1. It contains microcontroller with 8-bit HCS08 core [8], to be exact it uses HCS08GB60 MCU.

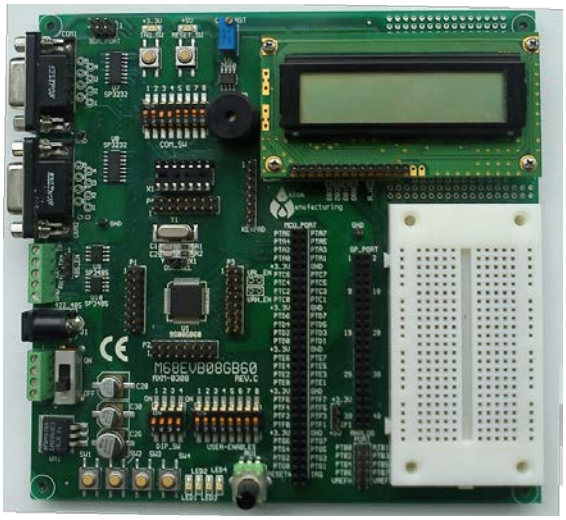

Fig. 1 Evaluation kit with Freescale HCS08GB60 microcontroller

The kit is out of production for several years, but the MCU itself was discontinued only recently, so we were able to repair the kits damaged during the course by replacing the MCU. However, it is clear that our supply of the MCUs would not last long and the kits would need to be replaced just for this reason in a year or two anyway. In the following section this older kit will be briefly described to provide better understanding of the requirements on the new kit. As already mentioned, the core of the kit is 8-bit microcontroller HCS08GB60 with 60 kB of flash and 4 kB of RAM memory. The MCU itself contains many internal peripherals such as two timers, two asynchronous serial interfaces (UART), analog-to-digital converter and others. The board adds external peripherals useful for education, such as LCD display (2 lines by 16 characters), four push buttons and four LEDs, potentiometer and a buzzer. There is also a so-called MCU port, which provides access to all the GPIO pins of the MCU and together with the breadboard (in the bottom right part of the picture) allows easy prototyping of students own circuits. We also use the MCU port as a means for connecting expansion modules which can offer other peripherals. These modules include, for example, a stepper motor (see figure 2), heat plant (figure 3), graphical LCD display, digital-to-analog converter, etc. Some of the modules were described in [9]. The same kit is also used in our labs for other courses [13], [14].

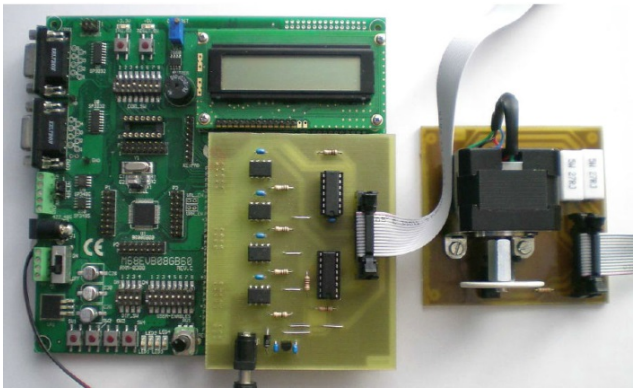

Fig. 2 Evaluation kit with Freescale HCS08GB60 microcontroller and external module with stepper motor

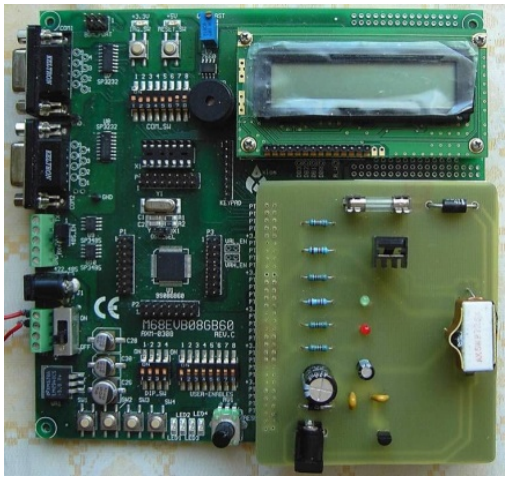

Fig. 3 Evaluation kit with Freescale HCS08GB60 microcontroller and model of heat plant

As a part of the course we also teach programming with simple real-time operating system (RTOS). We have developed such simple system for the older kit [10], [11].

The kit is connected to the development PC through serial port (RS232). There is a small monitor program in the ROM of the MCU which allows downloading user applications into the MCU memory and also source level debugging with breakpoints, variable watch etc.

B. Requirements for the new board

When defining desired features of the new educational board, the experience with the old board was also taken into account, besides the latest trends in the industry.

The main points for the new design can be summarized as follows:

- Microcontroller with 32-bit ARM core
- LCD display on board
- Several push buttons and LEDs
- MCU connector compatible with previous kit
- Low cost
- Easy repairs

Reasons for the above points are given here. ARM architecture has become a standard in the industry and is replacing 8-bit microcontrollers even in simple, low-cost applications.

LCD display, push buttons and LEDs are the basic means of communication of the MCU with the outside world. We prefer to include these directly on the evaluation board as opposed to a "minimalistic" approach known, for example, from the Arduino boards. Such approach means that the board contains minimal number of external peripherals, basically just the power supply, MCU, programming interface and connectors for accessing the MCU pins. This results in lower price and allows students to learn about the hardware together with software. For example, to see a blinking LED, one has to connect the LED and resistor. This is certainly good for understanding the topic better. On the other hand, for courses focused mainly on programming, this hardware interaction may be unnecessary and distracting and it also makes the course difficult to maintain – the equipment such as LEDs, push buttons, resistors, wires etc. has to be dealt with. For these reasons we prefer to have the basic peripherals on-board. Obviously, this does not limit the possibility to connect external peripherals if desired.

On the old kit with HCS08 there is a so-called MCU port which allows access to the MCU pins. As mentioned earlier we have several educational modules which connect to this connector. It was desirable to make it possible to use these modules also with the new kit. This means that a connector with physically compatible layout and functionally compatible pins of the MCU connected is required. It presented rather complicated task to connect the pins as required to maintain functional compatibility with the older kit's connector, but it was solved successfully.

Low cost was not a topmost priority, but naturally, it was also considered. Together with requirement for easy repair of damaged kits this lead to the decision to use factory-made MCU board which will be attached to the printed circuit board (PCB) of the kit, rather than using the MCU directly. There are many low-cost boards with ARM MCUs on the market nowadays. With cost starting even below $10, it is economically more suitable to use such factory-made board with the MCU and supporting electronics including programming interface than trying to use these small parts directly on PCB.

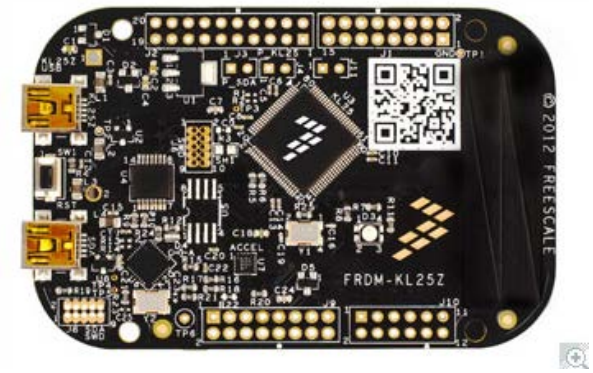

Fig. 4 Freescale FRDM-KL25Z board used in the kit [7]

We chose Freescale FRDM-KL25Z board [7] as the main part of the new educational kit. The board is shown in figure 4.

This board is member of a family of low cost evaluation boards called Freedom platform [12]. The board we used contains KL25Z128VLK4 microcontroller with ARM M0+ core, 128 kB of Flash and 16 kB of RAM. The board includes programming and debugging interface (openSDA). Layout of the board is compatible with the layout of the Arduino platform [1], see figure 5.

The compatibility with Arduino was not a requirement but seems advantageous. The platform is extremely popular in the community of do-it-yourself users and often used also for education. There are also numerous expansion boards (shields) available for this platform, which follow the same pin layout. Therefore, it is nice feature of the new kit, that the main board has Arduino-compatible layer and allows connecting the expansion boards designed for Arduino. Also the FRDM-KL25Z board can be detached from the kit and used separately and vice-versa another board with Arduino-compatible board could theoretically be connected to the educational kit as a new MCU component.

Fig. 5 Arduino Uno board [1]

III. New Educational Kit

Main features of the new development kit were already mentioned in the previous chapter. The basic components are shown in block diagram in figure 8. The actual physical placement of these components in the board can be seen in figure 6 and the real prototype of the kit in figure 7.

As can be seen in the figures, central component of the kit is the FRDM-KL25Z board with microcontroller and programming interface. This board connects to the host computer via USB cable. It is attached to a connector on the main board of the kit, so it can be easily replaced in case of damage. The FRDM-KL25Z board itself contains several interesting peripherals – an RGB LED, touch slider and three-axis accelerometer. On the main board there are many other peripherals useful for education:

- LCD display
- MCU port
- Four push buttons
- Three LEDs
- Potentiometer
- Rotary encoder
- RS-232 port
- RS485 port
- Real-time clock circuit PCF8583P
- Temperature sensor LM75AD
- Humidity sensor HIH6130
- EEPROM memory 25LC640A, connected via SPI bus, 64 kbit.

The text LCD display has two lines with 16 characters per line. MCU port allows connecting our educational models developed for the previous development board used in the lessons. There is also one serial port available on the board in addition to the virtual serial port provided by the programming interface over USB and also one RS-485 port. This way all three UART modules available in the microcontroller are utilized.

For learning about analog-to-digital converter there is a potentiometer connected to one of the input pins of the MCU. There is also rotary encoder connected to a timer channel, which makes it possible to experiment with input capture function of the timer module in the MCU.

The temperature and humidity sensor together with real-time clock (RTC) circuit are connected to the MCU via I2C bus; and there is also external EEPROM memory connected via SPI bus. Thus students can learn how to communicate via these busses and also how to work with the connected devices and sensors.

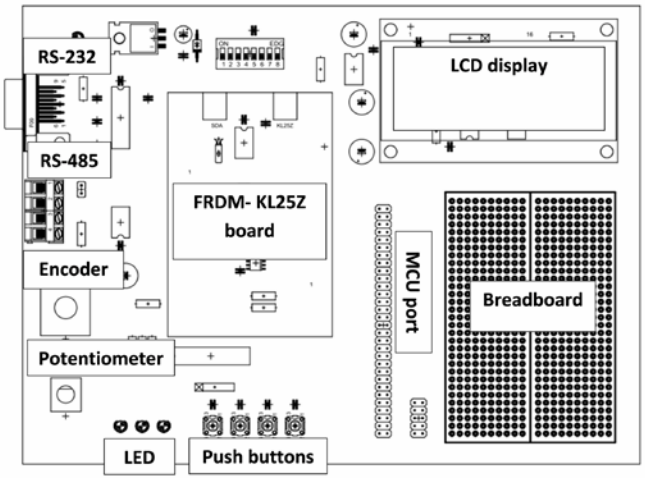

Fig. 6 Layout of the peripherals of the new kit

IV. Software Support

Besides the hardware, we also needed to provide a software framework for students to deliver example programs and peripheral drivers. This is important part, because starting software development without such support is very hard and time-consuming.

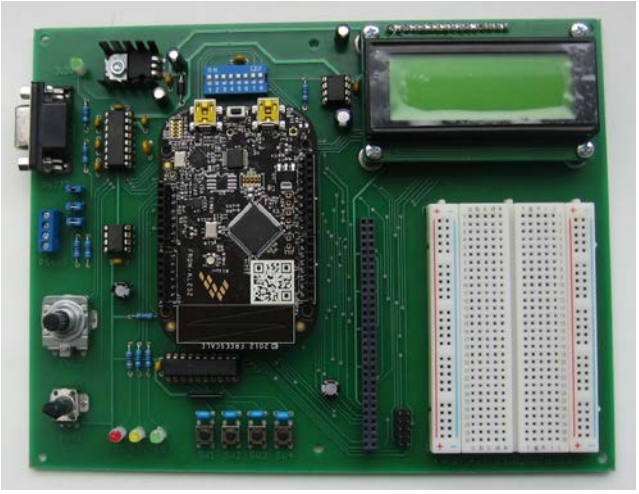

Fig. 7 Prototype of the kit; FRDM-KL25Z board attached

The obvious option would be to use already existing framework, and if no such framework can be used, then develop our own. The following options were evaluated:

- ARM mbed platform (embed.org)
- Freescale Kinetis SDK
- Our own framework

A. *ARM mbed platform*

The ARM mbed is a platform which speeds up the creation of devices based on ARM microcontrollers. It is popular in the do-it-yourself community, although not as much as Arduino. It also targets slightly different community of users with more programming experience. The platform is well designed and covers number of ARM-based microcontrollers and evaluation boards from different vendors, including the FRDM-KL25Z board used in our case.

For our purpose the main drawback of this platform is that the programs are developed using online (cloud) tools. This is not acceptable for our purpose. We cannot guarantee that the internet connection will be available all the time and also our ability to teach the course should not depend on an external website. Therefore we require a standard, offline tools. For some MCUs it is possible to export the mbed projects to such offline tools, but this option is not available for the MKL25 used in our board. The option of setting up the environment manually to build the programs with mbed framework proved to be too complicated to be usable for our purpose.

B. *Freescale Kinetis SDK*

Freescale, the manufacturer of the microcontroller used in our board, offers free software library for its ARM based MUCs and evaluation boards called Kinetis Software Development Kit (SDK). It is a set of drivers for the peripherals of the MCU.

For our purpose the SDK is little too complicated to set up and use. The course is intended to start for very simple programs, such as blinking an LED, and for such programs the programs with SDK would probably be harder to understand than without it. The benefits would only show in more complex programs, which are not the main focus of the course. Moreover, the SDK is limited to Freescale MCUs and it does not adhere to the ARM CMSIS standard for peripheral drivers. It creates its own standard, which is not likely to be adopted by any other MCU manufacturer. Considering this, the learning would be too vendor-specific; the students would spend most of the time at the lesson learning how to work with some peripheral driver specific to this SDK.

C. *Our software framework*

After considering the above mentioned options we decided to implement our own framework with example programs. The main reason is that we need to cover wide range of example applications, from very simple blinking LED with direct register access to applications using real-time operating system. Naturally, drivers for basic peripherals should also be included in the framework. But these drivers need to be as simple as possible so that students do not spend much time learning how to work with them. This is especially important for the peripherals the students need early in the course, such as serial interface (UART) and display.

Later in the course more complex drivers can be used for specialized peripherals, e.g. I2C module. These should preferably be compatible with the ARM CMSIS driver definition, because this is a vendor-independent standard which can be usable in the real world. As already mentioned CMSIS driver implementation is not provided for our MCU. Therefore we decided to implement simple, but compatible version of such drivers ourselves.

The framework is called utb_frdm_vyuka. The name consists of name of the university (UTB, or TBU in English), name of the board (frdm) and Czech word for teaching, vyuka.

It is available on github.com as utb_frdm_vyuka. There are four main parts of the framework:

- Example projects
- Drivers
- CMSIS drivers
- FreeRTOS support

The example programs provide complete projects with required drivers imported, paths set etc. for the whole range of applications used in the course.

Currently there are 22 projects, which cover the following topics:

- Blinking LEDs and reading switches using ready-made functions in Arduino style, such as pinWrite and pinRead.
- Blinking LEDa and reading switches using direct access to GPIO registers of the MCU.
- Working with analog-to-digital converter (ADC) using direct register access.

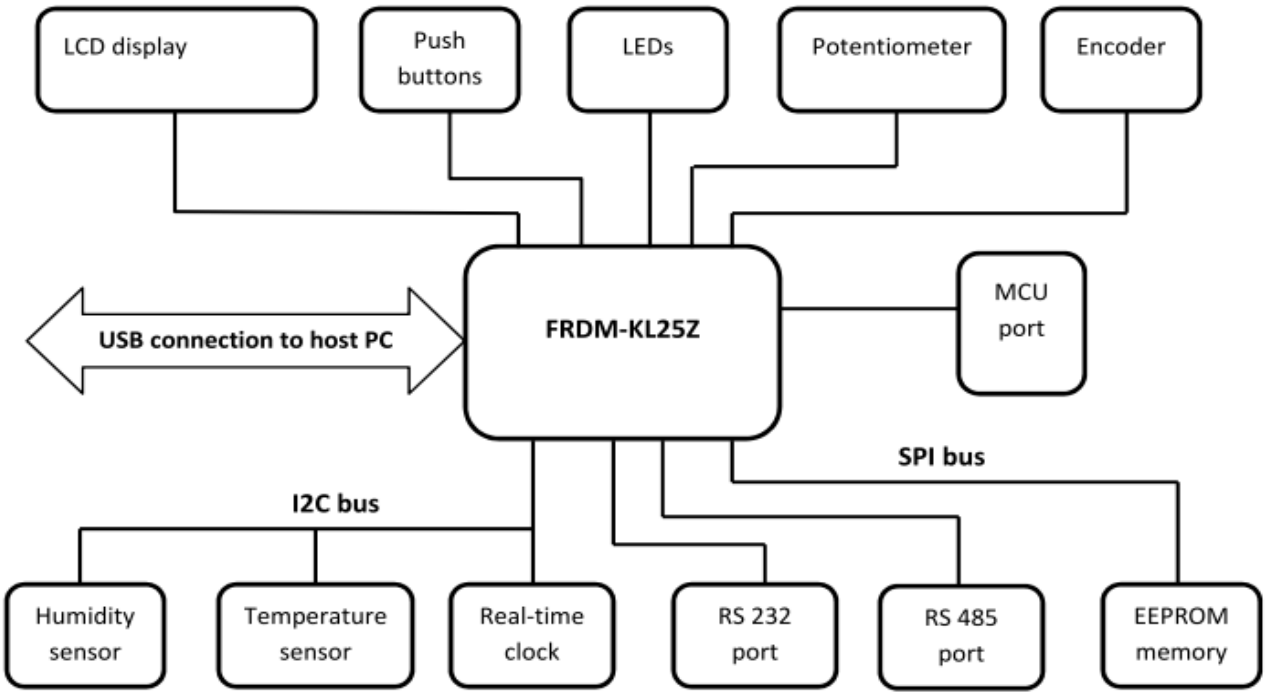

Fig. 8 Main components of the new kit based on FRDM-KL25Z

- Using TPM timer module for generating periodic interrupts, measuring time and generating PWM signal – direct register access.
- Using asynchronous serial communication (UART) with provided simple driver.
- Using I2C bus with provided driver with ARM CMSIS interface.
- Using FreeRTOS in simple multitasking applications and principles of mutual exclusion.

The simple drivers provided with the package cover the commonly used peripherals of the kit. There drivers include:

- GPIO driver for the LEDs and push buttons on the kit.
- LCD driver for the LCD display.
- SYSTICK driver for simple delay with length given in milliseconds.
- UART driver for the UART communication interface.

Besides these simple drivers with our own interface, there is also category of drivers which follow the ARM CMSIS Driver specifications. Currently only one driver is implemented – driver for I2C interface. This interface is used to connect some interesting peripherals on the kit – temperature and humidity sensors, real time clock (RTC) and an accelerometer.

V. Usage in the Microcontroller Programming Course

Starting in the winter semester of 2015 we use the above described evaluation kit and software framework in our course of microcontroller programming. There are total of 12 kits available which corresponds to the number of workplaces in the course lab.

We use Freescale's Kinetis Design Studio as the development environment in the course, mainly because it is free, so the students can use it also at home and because it is based on Eclipse, which the students are acquainted with from other programming courses.

There is one important difference regarding the kit to host computer interaction when compared to the older kit based on HCS08. The new board is connected to the host PC via USB interface. Due to the nature of the USB bus, each kit is a unique device for the host computer. This means that each kit has to be "paired" with one host computer in the lab. Otherwise, if the kits were allowed to "float" freely among the host computers, Windows device driver would need to be installed each time particular kit is connected to a particular host computer for the first time. This is not easy to accomplish given the fact that the students should (and do) work in limited user accounts on the computers. Even if the driver installation would be solved in some way that administrator credentials would not be needed, it would still be undesirable to end up with twelve devices (kits) installed on each computer in the lab. Therefore we adopted a simple scheme where each computer is assigned a number and the same number is written on the microcontroller kit. This way each kit is associated with its host computer and can be used only with this computer. This can be seen as a disadvantage when compared with the older solution which used RS232 serial line and there were no restrictions on the computers where each kit could be connected.

The hardware works without problems so far and also the software applications seem to provide good backing for the lessons. The lessons are organized as follows:

Lesson 1 shows how to control LED and read switches together with software delays (busy loop). The programs use simple GPIO driver which provides functions for writing to port pins and reading the pins. This way the students can concentrate on the higher-level program logic and learn the basic procedures for working with the IDE, such as building the project, uploading the program into the MCU and debugging it.

Lesson 2 covers similar topics as lesson 1, but this time the students are required to handle the pin operations using the peripheral registers of the MCU. This lesson is followed by a small project which controls a model of a real-world device (e.g. washing machine). The model is controlled by digital outputs of the MCU, for example, to turn on the pump, rotate the drum or turn on the heater. The model also provides feedback to the MCU by means of digital inputs, for example, two sensors of water level (half and full) and sensors of temperature (30, 40, 60 and 90 °C). Working on this project the students obtain some practical experience with the GPIO registers before moving on to more complicated peripherals.

Lesson 3 shows how to work with analog to digital converter (ADC) which is an on-chip peripheral in the MCU. The ADC module is used as a relatively simple example of a peripheral which can be controlled via its registers relatively easily. This lesson also introduces the concept of a driver for the MCU peripherals. The students are required to design and write their own driver for the ADC.

Lesson 4 explains the role of time in embedded programs. It covers use of the timer module with timer overflow interrupt.

Lesson 5 shows example of a program which communicates with the host computer via asynchronous serial line (UART). The program sends data to the computer and processes commands sent from the computer.

This lesson is followed by a second project, which is focused on practicing the skills obtained so far. In this project the students are required to create device which communicates with the host computer via serial line and also with one of the kits peripherals via I2C bus. The peripherals available are temperature sensor, humidity sensor and triple axis accelerometer. A driver for the IC2 is provided for this task.

Lesson 6 introduces real-time operating system FreeRTOS. The students will create applications with several tasks running in parallel and should also understand how mutual exclusion and task synchronization is handled. This lesson is then followed by third, last project. In this project the students are to create application which controls one of the models connected to the MCU port of the kit, as described earlier. Examples of these models are stepper motor, heat plant, graphical display and several others.

In future, the course should also cover creating programs with low power consumption, i.e. using the MCU low power modes.

VI. Conclusion

This article described our new educational kit for teaching microcontroller programming. This kit was developed to update the equipment used for courses of microcontroller programming at our faculty. It replaces older board with Freescale HCS08 microcontroller. The new board is based on 32-bit microcontroller with ARM M0+ core from Freescale's Kinetis family. A low-cost board FRDM-KL25Z is used as the main component of the kit. It is attached to our custom printed circuit board with other peripherals, such as display, push buttons, LEDs, external temperature and humidity sensors, RTS, EEPROM memory, etc.

A software framework was also created for this kit which consists of example projects and simple drivers for on-board peripherals. This framework makes it easier for the students to start developing their own applications.

Currently, the kits and accompanying software framework are used in our course of microcontroller programming.

References

[1] *Arduino, Open-source electronics prototyping platform*, Available: http://www.arduino.cc.

[2] P. Jamieson, Arduino for Teaching Embedded Systems. Are Computer Scientists and Engineering Educators Missing the Boat?. Available: http://www.users.muohio.edu/jamiespa/html_papers/fecs_11.pdf.

[3] P. Bender, K. Kussmann, "Arduino based projects in the computer science capstone course," *Journal of Computing Sciences in Colleges*, vol. 27, no. 5, pp. 152-157, May 2012.

[4] *Why Arduino is not the right educational tool*, Available: http://www.hackvandedam.nl/blog/?p=762.

[5] *ARM Ltd., CMSIS - Cortex Microcontroller Software Interface Standard*. Available: http://www.arm.com/products/processors/cortex-m/cortex-microcontroller-software-interface-standard.php

[6] *MCU Market on Migration Path to 32-bit and ARM-based Devices*, Available: http://www.icinsights.com/news/bulletins/MCU-Market-On-Migration-Path-To-32bit-And-ARMbased-Devices.

[7] *Freescale Semiconductor, FRDM-KL25Z: Freescale Freedom Development Platform for Kinetis KL14/15/24/25 MCUs.* Available: http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=FRDM-KL25Z

[8] *Freescale Semiconductor, CPU08 Central Processor Unit Reference Manual, rev.4.* Available: http://www.freescale.com.

[9] J. Dolinay, P. Dostálek, V. Vašek, "Educational models for lessons of microcontroller programming", in *Proc. 11th International Research/Expert conference TMT 2007*, Hammamet 2007, pp. 1447-1450.

[10] J. Dolinay, V. Vašek, P. Dostálek, "Implementation and Application of a Simple Real-time OS for 8-bit Microcontrollers," in *Proc. 10th WSEAS International Conference on ELECTRONICS, HARDWARE, WIRELESS and OPTICAL COMMUNICATIONS (EHAC '11)*, Cambridge 2011, pp. 023-026.

[11] J. Dolinay, V. Vašek, P. Dostálek, Utilization of Simple Real-time Operating system on 8-bit microcontroller. *International Journal of Mathematical Models and Methods in Applied Sciences*, 2011, vol. 5, no. 4, pp. 789-796, May 2011.

[12] *Freescale Semiconductor, Freescale Freedom Development Platform.* Available: http://www.freescale.com/webapp/sps/site/overview.jsp?code=FREDEVPLA

[13] L. Pekař, P. Dostálek, Z. Oborný, "Microcomputer I/O Converter and Control Unit for Heating Systems" in *In Latest Trends on Systems. Volume II*, Rhodes 2014, pp. 310-314.

[14] P. Dostálek, V. Vašek, J. Dolinay, Simple microcontroller based mains power analyzer device. *International Journal of Circuits, Systems and Signal Processing*, 2013, vol. 7, no. 4, pp. 214-221, July 2013.