Framework of Designing Multiple Microcontroller based Applications

R.S.K. Selvakumar[1], K.H. Ghazali[2], N.M.K. Nik Yusoff[3], H. Abdul Aziz[4]
Faculty of Electrical & Electronics Engineering
Universiti Malaysia Pahang
26600 Pekan, Pahang
MALAYSIA

inderjit_1791@yahoo.com[1] www.ump.edu.my

Abstract: - **This paper presents a framework for developing applications based on a three type of microcontroller (μC), Freescale HC11, MCS51 and Microchip PIC18. This project consists of the hardware and software implementation that supports the development and transfer process of the program code from a personal computer to the microcontroller, which led to the evaluation of educational based training kit systems. The first part of the paper focuses on hardware design, which is based on a modular approach, i.e. recomposed for the design of each application, in order to ensure maximum adaptability. The Multiple Microcontroller Evaluation Tool (microEVAT) thus consists of a programmer tool, a main board including adapters for a variety of chip packages, and plug-in modules. These are presented in the second part of the paper describing the software part of the framework, which besides programming tools also discusses the code development tools. The stress is given to the use of assembly code and high-level tools, where the algorithms are described in the form of different graphical notations, i.e. block diagrams. Throughout the paper, a special attention is given to the use of framework in the electronics engineering education process.**

Keywords: **- PIC microcontrollers, HC11 microcontrollers, MCS51 microcontrollers Development system, Electronics engineering education**

1. INTRODUCTION

A range of devices we use today in our everyday lives (e.g. telephones, household appliances, cars) contain so called "smart" electronics, which is usually implemented as an embedded system. In its fundament this is a microcomputer designed to control a particular process of the application. Most often, the central part of embedded systems is based upon a microcontroller (μC) or digital signal processing (DSP) device. These are a special kind of microcomputers integrated on a single chip, which in addition to a central processing unit and a memory consists of numerous application dependent peripherals (e.g. AD and DA converters, timers, communication modules, etc.)[1]. Being the most widely manufactured processing devices, microcontrollers play a pivotal role in electronic control systems. Concerning a broad range of potential consumers, they are a product of interest not only in electrical, but also in mechanical and industrial engineering.

With components being ever more complex, a design methodology and the ability to easily test and verify the designed system on a real object are becoming increasingly important. Building of application prototypes, either hardware or software therefore plays an important role in the design process. Since microcontroller based application design can be described as a combination of computer programming and digital circuits constructing [2], it is important to excel both fields at the same time. With the technology advancing very fast, the progress in this field also has to reflect in the pedagogical process [3, 4]. Educational institutions, responsible to provide students with appropriate skills and knowledge, thus need to be aware of the industry needs and practices. On the other hand, industry is the one responsible to make its specifics and requirements well known [5, 6].

Recognizing and acting upon the paradigm shifts in the design of contemporary systems thus calls for novel pedagogical approaches that should be supported by suitable hardware and software tools [1]. For that reason, our research team designed a framework aiming to ease and quicken this process. It is hardware part in the Multiple Microcontroller Evaluation Tool (microEVAT) with modular approach structure [7]. This means that it is recomposed for each application separately. The main hardware parts are the main board with the attached microcontroller and the memory used to download the firmware software from a personal computer (PC) to a microcontroller. In addition to these two major hardware parts several typical plug-in test modules have been designed, intended for the verification of developed programs. The framework's software tools can be divided into those for program code development, and those for programming the microcontrollers. In the latter group only freely available programs were considered, while in the case of development tools we focus on those offering high-level design, with the algorithms described in the form of different graphical representation (e.g. block diagrams, flowcharts) rather than written in C or assembly code.

The rest of the paper is organized as follows. In Section 2 a general embedded system design cycle is briefly introduced. The main characteristics of the framework's hardware parts, i.e. the Multiple Microcontroller Evaluation Tool (microEVAT), are presented in Section 3, while the software parts are described in Section 4. The frameworks' appropriates for educational purposes is discussed in Section 5. Finally, Section 6 concludes the paper with the ideas and plans for further developments of the framework.

2. IMPLEMENTATION OF EMBEDDED SYSTEMS

Due to dramatically increasing complexity of embedded systems over the past decades, developers are facing ever-increasing challenges for their products to stay market competitive [8]. In this context the utilization of systematic design methods is essential in order to aggregate rather than trade-off the technical, cost, and time-to-market feasibility factors [5, 9].

Typically, the implementation of embedded systems can be illustrated using a V-diagram representation. Several versions from different industry fields can be found in the literature, but here we refer to the one applied to the design of embedded systems [5]. As depicted in Figure 1, the general progression of the design steps in time is indicated from left to right. Hence the horizontal axis of the diagram can be thought of as time, but since the design is often an iterative process, the actual development rather cycles between left and right leg of the diagram than proceeds linearly through the steps [10]. The vertical axis represents the level of system components' abstraction, with the top steps representing high-level system view and the bottom steps representing very low-level processes.

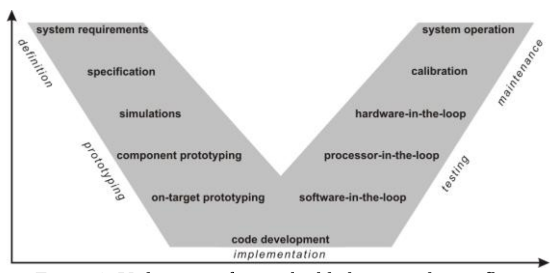


Figure 1: V-diagram of an embedded system design flow

The design starts with specifying the system requirements. These are than processed, and the system is along the left, the decomposition leg of the V-diagram, split into smaller pieces. At the bottom of the V-diagram is the implementation stage, representing the transition from decomposition back to re-composition proceeding up along the right leg of the V-diagram. Here the system's entities are combined back into larger pieces, resulting in a finally assembled system. The goal of each design process is a system working according to the specifications, with the V-diagram as narrow as possible, thereby reducing the time and consequently the cost of design [10].

The framework presented in this paper aids narrowing the embedded system design V-diagram by providing a common set of tools that are used through the prototyping, implementation and testing stages. In prototyping stage first a simulation model is designed that behaves according to the requirements. Then for the validation of feasibility, the design components are prototyped on general prototyping hardware. The timing characteristics are not modelled in this step, and may change in the actual design. Once the components behave appropriately, the on-target prototyping is employed on the envisaged processing device, in our case a microcontroller. Characteristics of code compiler, fixed-point precision effects and interactions among the system's components are validated in this step. When all simulations and prototyping shows the correct operation of a design, a highly optimized code is produced in the implementation stage. Finally, in the testing stage the design is evaluated while moving up along the right leg of the V-diagram. Configurations such as software-in-the-loop, processor-in-the-loop, and hardware-in-the-loop are used in the proceeding steps [8].

3. MULTIPLE MICROCONTROLLER EVALUATION TOOL

The intention of the microEVAT [7] is not to compete with the commercially available platforms, but rather trying to efficiently combine freely available tools and easily accessible components. Likewise, as an educational tool like E-blocks [11] or MILES [12], the system is not meant for any particular curriculum, but can be readily adapted to any secondary school or university course. Because of its modular concept it is also very appropriate for self-learning as it allows construction of actually required components, thus reducing the cost of the system. Figure 2 shows the block diagram of the system.

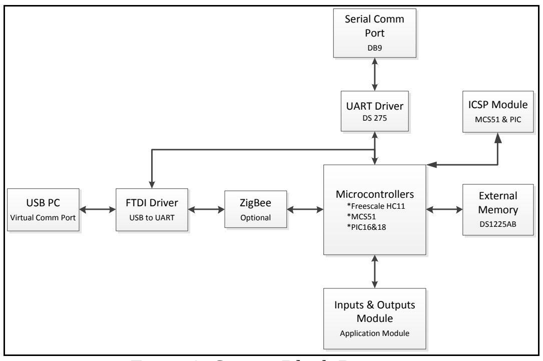


Figure 2: System Block Diagram

In the following subsections the assumptions taken into account when designing the microEVAT and its key hardware parts are described more in detail.

3.1 Designing Assumptions

Bearing in mind that the microEVAT is aimed at education process the Freescale HC11, MCS51 and PIC microcontroller families [13] have been selected from a range of other microcontroller families from different manufacturers due to multiple reasons [1, 14]. Firstly, these microcontrollers are widely available on the market at relatively affordable prices. Moreover, we can choose from a wide range of quality development and programming supporting tools (even freely available), while the transfer of the firmware can be always performed using the same interface. Lastly, they are also very popular in hobby electronic, which may also be an important factor for educational equipment. The following assumptions and limitations that were taken into account when designing the Multiple Microcontroller Evaluation Tool:

- Since the application's software development is usually a cyclic process of writing and testing the program code, its downloading to a μC as well as verification should be supported.
- The system must be structured in a modular fashion to assure maximum flexibility and rapid prototyping.
- No pre-programmed chips should be used in the design.
- All the basic accompanying software tools should be freely available on the Internet.
- The choice of microcontrollers should be limited to those where the transfer of the firmware can be performed using ICSP (In-Circuit Serial Programming) module (a property of a great majority of PIC microcontrollers) [15].
- The selection of microcontrollers is further restricted to those available in the PDIP package types [15].
- The programmer must offer a possibility to download a firmware from PC to microcontroller using FTDI (USB) or serial (COM) port.
- The programmer must be designed in such way that it can be permanently connected to the main board, i.e. when programming or running the application.
- All microcontrollers' I/O pins must be accessible on the main board plug-in module connectors.
- Where needed the setting of microcontroller's operation using external signals should be accessible through jumpers or short-circuit connections.
- Plug-in modules must be designed in such a way that even their incorrect use does not harm any of the system's components.

These assumptions are based on recognition of problems and difficulties that beginners in the world of microcontrollers are often faced with. Consequently, the system designed in such way will not result in the most efficient variant, but will certainly enable someone to get familiar with the entire design cycle.

3.2 Main Board of microEVAT

The main board comprising of three types of microcontroller families represents a central part of the development system, to which all other system components (i.e. the power source, the programming mode, and plug-in modules) are connected as needed. Its structure of electrical scheme is depicted in Figure 3.

These microcontrollers are attached to the main board through a standard 40-pin PDIP, 44 and 52-pin PLCC. A limitation to the microcontrollers with up to 40 pins was made due to the fact that a 40-pin package is the largest PDIP package type in the case of PIC microcontrollers and so with 52-pin for Freescale HC11 and 44-pin for MCS51 microcontrollers. Furthermore, a more frequent variant of pins arrangement was considered, while for microcontrollers with different arrangement or lower number of pins, appropriate adapters were designed.

The main board includes a power supply with Greatz Bridge whose intention is dual. The first is to be used as a classical full wave bridge, so that input voltage can be either AC or DC. The second is to protect the connected power supply in case the programmer has its own power supply connected. Plug-in modules and the memory interface circuit are connected to the main board through wire-wrapping technique (prototype). One microcontroller's port with the supply voltage is available on each of the power supply jack connector for the connection of plug-in modules.

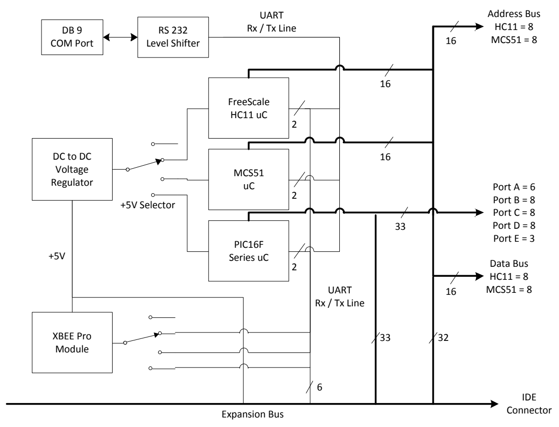


Figure 3: Structure of the Basic interface and power module.

On the main board of the microcontroller additionally the microcontroller’s pins for the connection of external reset circuit, external oscillator and external AD converter supply voltage can be accessed.

Each microcontroller port is connected to the resistor chain, which is used to perform either pull-up or pull-down function. The operation can be set using appropriate jumper connections. On the IDC connector for the programmer attachment all pins that are necessary for the connection of ICSP compatible programmers for PIC microcontrollers are available. Additionally, on the remaining pins, the microcontroller’s USART module pins are available, that can be used when downloading the firmware using bootloaders. The programmer’s virtual connection-disconnection to the microcontroller is driven manually, using a single switch.

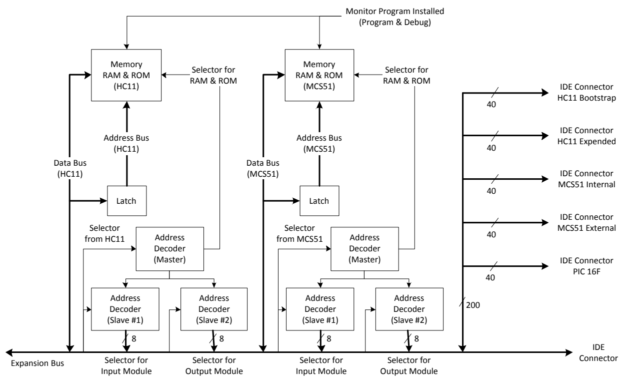
Places for the RC and quartz crystal oscillator are also foreseen on the prototype board. A simple reset circuit with a key as an execution unit is also designed so that it does not disturb the ICSP programmer activity.

3.3 Programmer and Memory Interface

Three particularly important design assumptions were taken into account when designing a programmer’s hardware part. These are that the microcontrollers must be programmable either firmware based (monitor program) or in the ICSP mode, that the firmware download can be performed using serial or USB PC port, and that the accompanying software programming tools are freely available on the Internet.

Programmer is therefore designed to be compatible with the Motorola, Intel and Microchip programmers, where the download of the firmware is in the case of JDM programmer achieved using COM port. ICSP is a synchronous procedure specified by Microchip and used for programming the PIC microcontrollers [18]. It uses five or six microcontroller pins for writing and reading the firmware. The sixth pin must be driven only at microcontrollers which can also be programmed using a second type of programming procedure, i.e. the Low-Voltage ICSP.

The structure of electrical scheme of the memory interface is depicted in Figure 4. The circuit consists only of standard elements and can be connected to the main board using wire wrapping technique. Similarly to the main board, for ICSP programming technique, the programmer has its own power supply for two additional reasons. The first is to protect the microcontroller against damages during programming using a build in 100 mA current limit, and the second is simply to enable its autonomous use, e.g. for programming the microcontroller already attached to the end application. Figure 5 shows the electrical scheme of the programmer.



Monitor Program Installed (Program & Debug)
Memory RAM & ROM (HC11)
Selector for RAM & ROM
Memory RAM & ROM (MCS51)
Address Bus (HC11)
Address Bus (MCS51)
Data Bus (HC11)
Data Bus (MCS51)
Latch
Selector from HC11
Selector from MCS51
Address Decoder (Master)
Address Decoder (Slave #1)
Address Decoder (Slave #2)
IDE Connector HC11 Bootstrap
IDE Connector HC11 Expended
IDE Connector MCS51 Internal
IDE Connector MCS51 External
IDE Connector PIC 16F
Expansion Bus
Selector for Input Module
Selector for Output Module
IDE Connector

Figure 4: Structure of the Memory interface.

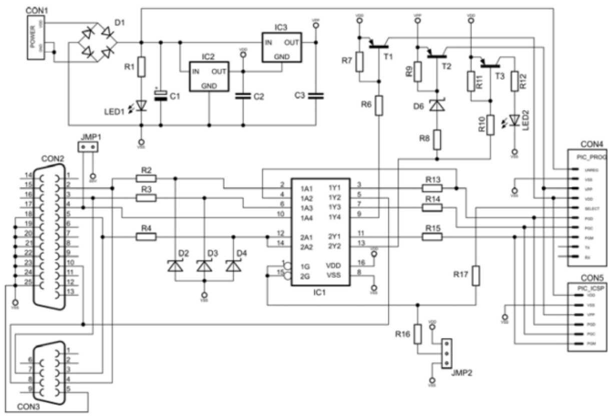


Figure 5: Electrical Scheme of the programmer

3.4 Plug-In Modules

For the purposes of testing and validating programs written into microcontrollers a set of basic test circuits has been designed. We refer to them as plug-in modules as they are connected to the main board for each application design. Some of those are:

- Module 0 – a module to drive a state of a particular microcontroller's input using a Keypad and DIP Switch.
- Module 1 – a module to represent a particular microcontroller's output by using Bar Graph, Multi-segment display and Dot Matrix display.
- Module 2 – an LCD and Graphic LCD module with two 16 X 2 characters alphanumeric lines and 128 X 64 dot pixels.
- Module 3 – a module with high power application such as Stepper Motor and DC Motor.

In order to offer maximum usage flexibility, the modules are designed so that that their final function is adapted by setting jumpers or short-circuit connections. An example of prototype development system is shown in Figure 6.

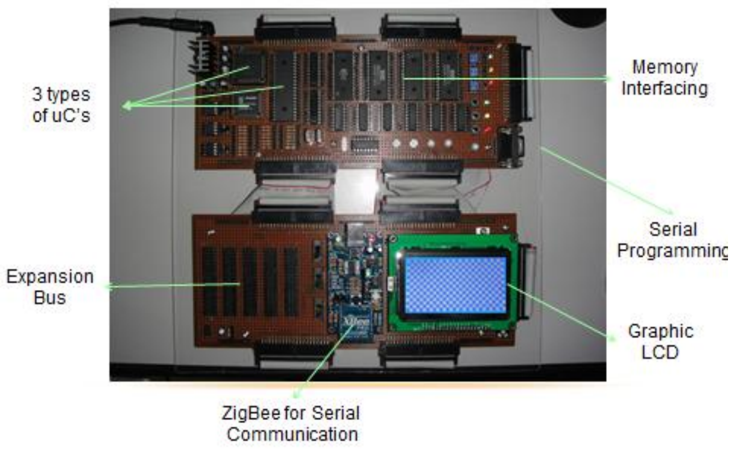


Figure 6: Prototype of the microEVAT with GLCD and ZigBee module interfacing

4. SOFTWARE IMPLEMENTATION

During the microcontroller based application development several different software tools are used. When working with these microcontroller families, we can select among a wide range of quality software tools from different manufacturers, where many of them (including some from Microchip) are in its full functionality freely available on the Internet. We can coarsely divide them into the development tools in which the source code is processed, and the programming tools used to transfer the compiled firmware from a PC to the microcontroller.

4.1 Software Development for Freescale HC11 and MCS51

Development of evaluation tool software is divided into two stages; the monitor program development and the Integrated Development Environment (IDE) program development. Monitor program controls the entire interfacing between a computer and the evaluation tool. Written in assembly language and stored in ROM, monitor program is used to perform object file loading and executing without the hassle of burn-erase-burn method usually associated with EPROM programming. This program will be stored into microcontroller internal EEPROM.

The conceptualization of the monitor program algorithm (used in this evaluation tool) is shown in the Figure 7. The basic function of a monitor program is to read user's input from keyboard. Based on the input, it determines which procedure, either upload S19 or HEX file or execute program, to be executed. Once selected, the instruction is processed by sending the information to the development board for execution. Status of execution is feedback to user on a computer screen.

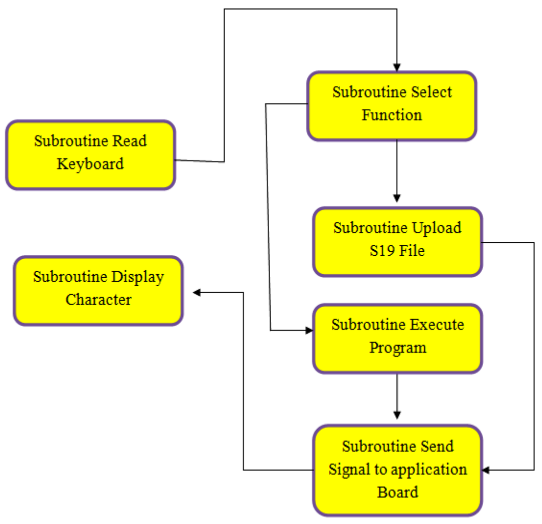
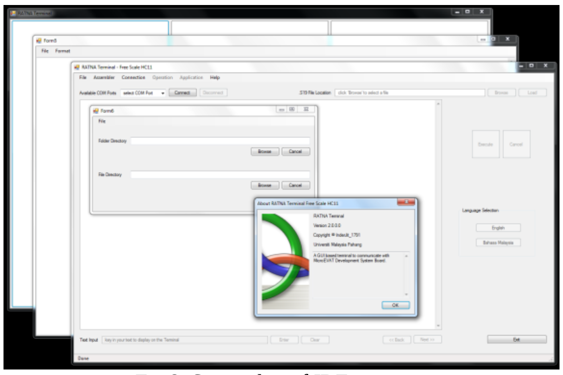

Subroutine Read Keyboard
Subroutine Select Function
Subroutine Upload S19 File
Subroutine Display Character
Subroutine Execute Program
Subroutine Send Signal to application Board

Fig 7: Concept Diagram of Monitor Software

Usually running a monitor program requires the use of communication program such as Hyper-Terminal. Likewise, developing application software for microcontroller requires a text editor and an assembler to write and assemble the code respectively. An IDE program integrating programs such as communication program, cross-assembler and text editor is developed to create user-friendly development environment. This new software is developed to allow users to perform all development activities in the ever familiar window working environment without needing to exit any program. Instead of entering instruction on command line, IDE user interface allows execution of instruction by clicking the appropriate buttons. Figure 8 shows a screenshot of the IDE program.



A GUI based terminal to communicate with MicroEVAT Development System Board.

Fig 8: Screenshot of IDE program

4.2 Development Tool for PIC Microchip

Microchip offers a full range of development tools for its PIC microcontroller family. These include an Integrated Development Environment (IDE) usually dubbed MPLAB [19], consisting of assembler, C compiler, simulator, product selector guide etc. Integrated with these tools there are also programming tools, which normally support only the Microchip's proprietary hardware. For that reason next subsection presents some alternative software tools meant for the use with our microEVAT programmer.

Besides Microchip's development tools we can also choose among development tools from other manufacturers. These are typically compilers from higher-level languages (e.g. C, Basic, and Pascal), real-time operating systems, and tools adapted for operating systems other than Microsoft Windows (e.g. Linux). From a range of higher-level language compilers a HI-TECH PICC [20], IAR Embedded Workbench [21], CCS C [23], and mikroC [23] C language compiler deserve to be exposed. Providing an extensive set of common and microcontroller specific libraries (e.g. pre-processor commands, peripheral drivers) they can be used either with their own integrated development environments (as mention above in section 4.1) or integrated with Microchip's MPLAB IDE. While the mikroC can only be used to develop code for 8-bit microcontrollers (PIC10, PIC12, PIC14, PIC16 and PIC18 families), HI-TECH, IAR, and CCS C tools are also covering the 16-bit microcontrollers (dsPIC and PIC24 families-which not included in microEVAT).

Traditional code development tools, such as those mentioned above, take place late in the design cycle, i.e. in the implementation and test stages (see Figure 1). Such design process is consequently more prone to errors, due to late integration of hardware and software [5]. Having available a real-time prototyping hardware, like microEVAT in the case of PIC microcontrollers, implementation aspects can be addressed earlier in the design cycle, thus supplementing traditional prototyping approaches (e.g. off-line simulations) with real testing [4, 10]. Software tools allowing developers to design simulation models in early design stages and then verifying them on a real hardware through the use of automatic code generation have recently emerged as a model-based design tools. They are not only used in the prototyping stage, but also in the implementation and testing stages, where extensive code generation for different hardware configurations is required [5, 8].

One of the most commonly used model-based design tools is The MathWorks' Simulink platform [24]. It provides an interactive graphical environment in which the algorithms are developed in the form of block diagrams. With the aid of Real-Time Workshop Embedded Coder it can be used to generate the target independent ANSI C code. Microchip's MPLAB IDE plug-in for MATLAB and Simulink enables developers to include the Simulink model in the design of 8-bit microcontroller applications, whose automatically generated code can be included into MPLAB IDE projects.

A similar tool, extending the functionalities of Simulink, is the Embedded Target for 16 bits PIC [25]. It provides the blocks for microcontroller specific peripherals, such as ADC converters, UART and SPI communication, PWM modulation, etc. Models developed with this tool can be compiled and build for 16-bit microcontrollers directly from the Simulink without requiring MPLAB IDE.

In the area of open source tools Evidence's Scilab/Scicos Code Generator for FLEX [26] has recently emerged as a model-based design platform. Similarly to Octave, Scilab [27] is a MATLAB like open source platform for numerical computation, while Scicos [28] is its block diagram modeller and simulator, in which block diagrams are developed as in Simulink. Using built-in code generator the Scicos block diagrams can be exported to ERIKA Enterprise [26] and run on the FLEX board for dsPIC microcontrollers [26]. Although these open source model-based design tools are still in the early development stage, they deserve being mentioned, since they are aiming to cover the entire design cycle.

The last tool our team present in the group model-based design tools is the Flowcode from Matrix Multimedia [11]. It supports simulation and transfer of the program code to a subset of 8-bit microcontrollers, with the code generated from the flowchart type block diagrams.

4.3 Programming Tools

Hardware part of the Multiple Microcontroller Evaluation Tool (microEVAT) can be accompanied using one of the following freely available software programming tools:

- IC-Prog [29] – can be used for programming the microcontroller through the PC's COM port, when the programmer in JDM compatible mode.
- WinPic [30] – fully customizable programmer that can be used to program the microcontroller either using serial or parallel port.
- WP11 [32] – programmer for HC11 based microcontroller that can be used when bootstrap mode operation selected.

5. FRAMEWORK AS AN EDUCATIONAL TOOL

Provided that during the framework design all the predefined microEVAT assumptions were fulfilled and efficient supporting software tools have been selected, the development system resulted in robust equipment, which can be also used for educational purposes in either secondary school or university courses. Moreover, it is not destined only to electrical engineering education, but can be quickly adapted to courses of other engineering branches due its modular hardware and the software tools, not requiring extensive programming skills. Eventually, the framework is also efficient enough to serve experts in designing their μC-based applications.

One of defining characteristics of engineering education is the use of models for problem solving. A variety of graphical representations are thus used to illustrate the behaviour of object of interest. In a sense of microcontroller application design, where mastering of digital circuit construction and programming skills at the same time is essential, circuit schematics are used to represent the hardware, while different forms of block diagrams are used to illustrate the software execution flow. Comparing the product design cycle to the educational process, where a specific topic should be covered from its theoretical and practical point of view within a limited time frame, similar challenges can be identified, and efficient development tools are playing a crucial role in overcoming them [4]. Our framework can aid the education process by offering students to build their hardware models by interconnecting the microcontroller with the microEVAT plug-in modules, and enabling them to easily converge from high-level models to efficient product implementations using contemporary software tools.

In case of using the framework in the secondary school course it can serve as common supporting educational equipment at multiple courses [31]. For instance, the system can be used in courses dealing with basics of microprocessors and micro-controllers, their programming (in assembler, higher-level language or through model whose code is automatically generated), and further in courses concerning the design of additional plug-in modules. The most motivated students can also fabricate the system (or its particular part) by themselves.

At the university level the framework can be used as educational equipment in courses dealing with microprocessor, microcontrollers or embedded systems, covering both the programming and the hardware peripheral design support [1]. Due to the growing need of interdisciplinary education the system is also an attractive solution for students of other engineering sciences, e.g. computer, mechanical, chemical or industrial engineering education. Some subjects that the system can aid covering include computer architectures, digital signal processing, mechatronics, automated measuring, etc.

6. CONCLUSION

In this paper the framework consisting of the Multiple Microcontroller Evaluation Tool (microEVAT) and the supporting software tools were presented together with some proposals for its use in educational courses. Destined to three types of microcontroller application design, the microEVAT is designed as open as possible, so that anyone can fabricate it on his/her own and use it with freely available basic accompanying software. Its design is sufficiently general to be used as a whole or as a component in another system.

It is not focused on a particular microcontroller device, but it enables to select the most appropriate device for each application design. When using as a whole, the microEVAT represents a very robust system, where even an incorrect use does not harm any of the system's components. The development and improvement of the system is still in progress and additional plug-in modules (e.g. keypad, Multi-segment display, stepper motor module, ZigBee communication module, etc.) are being designed. As such the system is particularly appropriate for educational purposes, trying to support those entering to the world of microcontrollers.

References:

[1] A. H. G. Al-Dhaher, Integrating hardware and software for the development of micro-controller-based systems, *Microprocessors and Microsystems*, vol. 25, iss. 7, October 2001, pp. 317-328.

[2] M. Predko, *Programming and customizing PICmicro microcontrollers*, McGraw-Hill, 2000.

[3] P. Caspi et al., Guidelines for a graduate curriculum on embedded software and systems, *ACM Transactions on Embedded Computing Systems*, vol. 4, iss. 3, August 2005, pp 587-611.

[4] W. Wolf and J. Madsen, Embedded systems education for the future, *Proceedings of the IEEE*, vol. 88, iss. 1, January 2000, pp. 23-30.

[5] P. J. Mosterman, Automatic Code Generation: Facilitating New Teaching Opportunities in Engineering Education, *36th ASEE/IEEE Frontier in Education Conference*, October 2006.

[6] D. J. Jackson and P. Caspi, Embedded Systems Education: Future Directions, Initiatives, and Cooperation, *SIGBED Review (Special Issue on the First Workshop on Embedded System Education)*, vol. 2, iss. 4, October 2005.

[7] M. Smolnikar, MPICds – MPIC development system, *http://MPICds.googlepages.com/*, December 2007.

[8] P. J. Mosterman, S. Prabhu and T. Erkkinen, An Industrial Embedded Control System Design Process, *The Inaugural CDEN Design Conference (CDEN'04)*, July 2004.

[9] M. Rupp, A. Burg and E. Beck, Rapid prototyping for wireless designs: the five-ones approach, *Signal Processing*, vol.83, iss. 7, July 2003.

[10] Shortening the Embedded Design Cycle with Model-Based Design, National Instruments Corporation, *http://zone.ni.com/devzone/cda/ tut/p/id/4074*, December 2007.

[11] Flowcode and E-blocks, Matrix Multimedia Limited, *http://www.matrixmultimedia.com/*, December 2007.

[12] L. F. Ferreira et al., MILES: A Microcontroller Learning System combining Hardware and Software tools, *35th ASEE/IEEE Frontiers in Education*, October 2005.

[13] Technical Library CD-ROM 2004/2005 (DS00161Q), Microchip Technology Inc., 2004.

[14] A. Clements, Selecting a processor for teaching computer architecture, *Microprocessors and Microsystems*, vol. 23, iss. 5, October 1999, pp. 281-290.

[15] Product Selector Guide 2005 (DS00148K2), Microchip Technology Inc., 2005.

[16] PIC-Programmer, *http://www.jdm.homepage .dk/newpic.htm*, December 2007.

[17] P16PRO and PICALL, *http://picallw.feniks-pro.com/*, December 2007.

[18] PICmicro 18C Family Reference Manual – In-Circuit Serial Programming (DS39531A), Microchip Technology Inc., 2004.

[19] MPLAB IDE User's Guide (DS51519B), Microchip Technology Inc., 2006.

[20] PICC compiler, HI-TECH Software, *http://www.htsoft.com/products/compilers/picccompiler.php*, December 2007.

[21] IAR Embedded Workbench, IAR Systems, *http://www.iar.com/*, December 2007.

[22] CCS C Compiler, CCS Inc., *http://www. ccsinfo.com/*, December 2007.

[23] mikroC compiler, mikroElektronika, *http://www.mikroe.com/en/compilers/mikroc/ pic/*, December 2007.

[24] Simulink, The MathWorks Inc., *http://www. mathworks.com/products/simulink/*, December 2007.

[25] Embedded Target for 16 bits PIC, *http://www.kerhuel.eu/RTWdsPIC/*, December 2007.

[26] FLEX: Microchip dsPIC evaluation board, Evidence, *http://www.evidence.eu.com/content/ view/114/204/*, December 2007.

[27] Scilab: The open source platform for numerical computation, INRIA, *http://www. scilab.org/*, December 2007.

[28] Scicos: Scilab's block diagram modeler/simulator, INRIA, *http://www.scicos. org/*, December 2007.

[29] IC-Prog Prototype Programmer, *http://www.ic-prog.com/*, December 2007.

[30] WinPic - A PIC Programmer for Windows, *http://freene-thomepage.de/dl4yhf/winpicpr.html*, December 2007.

[31] W.A. Stapleton, Microcomputer Fundamentals for Embedded Systems Education, *36th ASEE/IEEE Frontiers in Education Conference*, October 2006.

[32] WP11 – Windows Programming for HC11, *http://www.tec-i.com*, December 2004.