# A Framework for Developing a Microchip PIC Microcontroller Based Applications

Shripad G. Desai[1], Vibhor Agarwal[2]
[1]*Assistant professor, Department of Electrical Engineering, Bharati vidyapeeth (Deemed to be university, College of Engineering), Pune*
[2]*Student, Department of Electrical Engineering, Bharati vidyapeeth (Deemed to be university, College of Engineering), Pune*

***Abstract*** **- This paper presents a framework for developing applications based on a Microchip PIC microcontroller (μC). Consisting of hardware and software tools it supports the development and transfer of program code from a personal computer to the microcontroller, and evaluation of its execution on rapid prototyping hardware. 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 MPIC development system (MPICds) thus consists of a programmer, a main board including adapters for a variety of chip packages, and plug-in modules. All these hardware components are designed so that they can be fabricated by any potential user. Selected microcontroller is programmed in the standard ICSP mode using one of the freely available PC programs. 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 high-level tools, where the algorithms are described in the form of different graphical notations, i.e., block diagrams, instead in C or assembly code. Throughout the paper, a special attention is given to the use of framework in the electrical engineering education process. Besides giving some suggestions for the use at different educational levels, the emphasis is on the implication of contemporary design methods, and the hardware's robust design, which is deliberately planned in such a way that even an incorrect use cannot harm any of its components.**

## I.INTRODUCTION

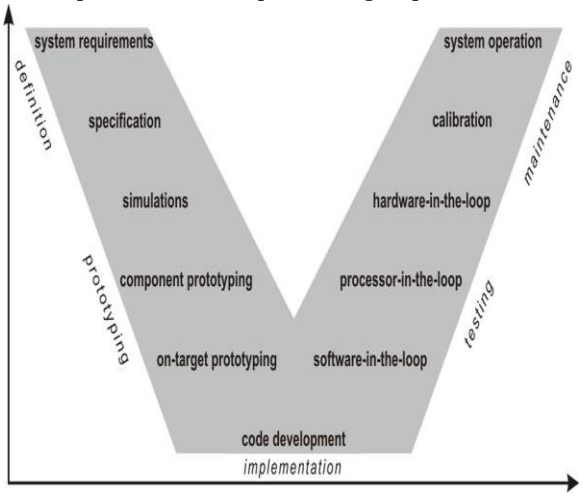
A range of devices we use today in our everyday lives (e.g., telephones, household appliances, cars) contain so called "smart" electronics, which is usually implemented as an embedded system. In its fundament this is a microcomputer designed to control a particular process of the application. Most often, 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, be in 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 master 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.

## II.EMBEDDED SYSTEM DESIGN CYCLE

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 tradeoff the technical, cost, and time-to-market feasibility factors [5, 9]. Typically, design cycle steps 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 Fig. 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.

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 show 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].




V-diagram of an embedded system design flow

## III. MPIC DEVELOPMENT SYSTEM

The intention of the MPICds [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. In the following subsections the assumptions taken into account when designing the MPICds and its key hardware parts are described more in detail.

1.Main board

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

The microcontroller is attached to the main board through a standard 40-pin Textool socket. 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. 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 programmer are connected to the main board through 10-pin IDC connectors. One microcontroller's port with the supply voltage is available on each of the IDC connectors for the connection of plug-in modules. On the IDC connector of the PORTE 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 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.

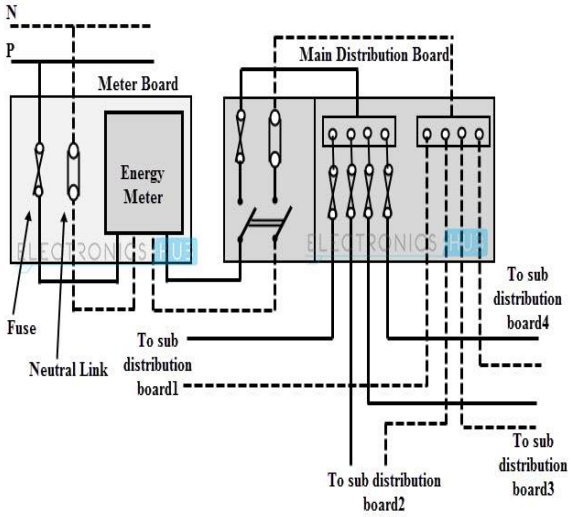



FIG. Electrical scheme of the main board

2.PROGRAMMER

Three particularly important design assumptions were taken into account when designing a programmer's hardware part. These are that the microcontrollers must be programmable in the ICSP mode, that the firmware download can be performed using serial or parallel PC port, and that the accompanying software programming tools are freely available on the Internet. Programmer is therefore designed to be compatible with the JDM [16] and P16PRO [17] programmers, where the download of the firmware is in the case of JDM programmer achieved using COM port, while the P16PRO uses LPT 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.

Electrical scheme of the programmer is depicted in Fig. The circuit consists only of standard elements and can be connected to the main board using IDC connector cable. Similarly, to the main board, 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. For such cases, a Microchip compliant FCC connector is foreseen on the printed circuit board.

The main programmer's element bridging the PC and microcontroller is the high-speed CMOS integrated circuit 74HCT367. It contains six three state non-inverting buffers, which are driven by two enable inputs. Using those, the firmware is transmitted from the PC to a microcontroller and vice versa. The circuit additionally contains some other elements from which the transistors and Zener diodes should be exposed. The latter serve for the adaptation of the COM port voltage levels, which in contrast to LPT port are not TTL compatible. The transistors are used either to assure appropriate current loads or to convert the voltage levels, since in the ICSP programming mode voltages higher than 5 V have to be generated.

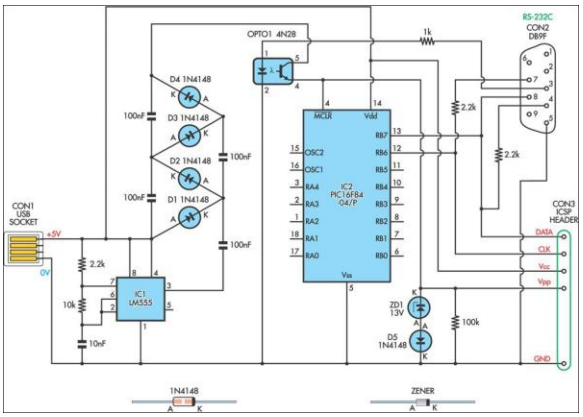



FIG. Electrical scheme of the programmer

## IV. SOFTWARE TOOLS

During the microcontroller-based application development several different software tools are used. When working with the PIC microcontroller family 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.

### 1. Development tools

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 MPICds programmer.

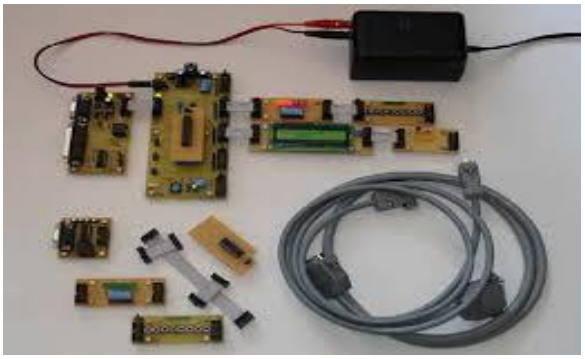


FIG. Photo of the fabricated MPICds

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, 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 mikro C [23] C language compiler deserve to be exposed. Providing an extensive set of common and microcontroller specific libraries (e.g., preprocessor commands, peripheral drivers) they can be used either with their own integrated development environments or integrated with Microchip's MPLAB IDE. While the micro-C 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).

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 Fig. 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 MPIC ds 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 tool. 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]. Some of the tools, more or less following the model-based design paradigm and suitable for the use with the PIC microcontrollers, are described in the following.

## V. FRAMEWORK AS AN EDUCATIONAL TOOL

Provided that during the framework design all the predefined MPICds 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 MPICds 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 microcontrollers, 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.

## VI. CONCLUSION

In this paper the framework consisting of MPIC development system and the supporting software tools were presented together with some proposals for its use in educational courses. Destined to Microchip's PIC microcontroller application design, the MPICds 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 MPICds 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., matrix keyboard, 7- segment LED display, stepper motor module, IR 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 microcontroller-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, issue. 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.