Problems of effective software development for embedded systems and their ways solutions

A.A. Belykh, Yu.V. Gotovsky
(MPEI (TU), Moscow, Russia)

Nowadays, microcontrollers are used more and more often in embedded systems to solve a huge number of tasks. The microcontrollers themselves can be classified according to a number of characteristics - by the performance of the computing core of the microcontroller (processor) (in MIPS, the number of instructions per second), the processor architecture, the presence of built-in internal memory for data and program code, the requirement for external memory for bootstrapping, the development of the microcontroller's periphery and others. In this paper, we will consider the most widely and often used in various fields of 8-bit microcontrollers (MC). Their frequent use in developments all over the world is due to their low cost and relative simplicity. At the same time, this requires a more "careful" attitude to the quality of the program code, and often the development of software for such MCUs is carried out using the assembly language. Note that the assembly language is not universal and often differs not only from different manufacturers, but also between the MK families of the same manufacturer. However, the criteria, principles and postulates formulated in this article will generally be true for 16- and 32-bit microcontrollers. The use of microcontrollers allows you to transfer the main development costs from the hardware to the software area. This inevitably entails the study of existing and the development of new ways to solve the difficulties faced by programmers in the development of software systems, the development of criteria for effective software development for embedded systems. In addition, rapid improvement (modernization) and the addition of new useful (and sometimes, extremely necessary in a specific task) capabilities in the latest families of microcontrollers forces the release of new versions of devices or simply use them in new developments. Often, in such cases, you have to almost completely rewrite the program text for various reasons - a change, even the slightest, of the instruction system (the expansion of the instruction system with instructions that more efficiently perform a set of actions taking into account the changed architecture of the microcontroller and interaction with memory is quite common) , changes in the internal structure of the microcontroller (registers, internal memory, etc.), changes in the periphery, resulting in a modification of the exchange routines with the outside world or inside the designed device, etc. Moreover, one can imagine a situation when the designer has chosen a certain

a microcontroller for a specific task, began the design and implementation of the project, but then it turned out that this MC was not able to meet the requirements that had changed for a number of reasons (for example, a significant change in the problem statement as a result of a large number of adjustments). Moreover, it turns out that none of the MCUs from this family (by the family we mean microcontrollers similar in architecture and structure, differing mainly in the development of the periphery) does not satisfy the requirements of the modified task. The developer is forced to pick up a completely different MK and start

implementation from scratch. In addition, the new MK may have a completely different assembler (in case they were forced to stop at the MK of another manufacturer), which means that additional time will be required to study it. These problems affect the efficiency of software development for microcontrollers in particular and the design of embedded systems in general.

Efficiency of software development for microcontrollers need to be properly evaluated in order to choose one or another approach to improve this efficiency. The practice of designing embedded systems suggests the main criterion for the effectiveness of development - speed, and, as a numerical expression, the development time (in people hour) of a set of tasks. However, in order to reflect reality as much as possible, it is required to measure the development time of not one system (module), but a series - and over a sufficiently long period of time. And only the entire set of times as a whole can be used to determine the effectiveness of the development process. This condition is necessary due to the fact that the complexity of software design for embedded systems can be unevenly distributed both in time (from device to device) and depending on the specific task. Thus, in order to correctly assess the effectiveness of software development using a particular design tool, it is necessary to measure the time spent on the implementation of each task from the entire set of tasks predetermined (from practice). Note that the time spent on "thinking" the problem is the development of an algorithm for evaluating the effectiveness of development is not taken into account for obvious reasons. In fact, we are only interested in the execution time of a task in a specific assembly language for a specific microcontroller. Another, no less important (and for embedded systems on relatively simple MCUs, probably more important than development time) criterion of software development efficiency is the quality of the program code and, as its numerical expression, the relative software execution time. It should be noted here that we are interested in relative efficiency, that is, comparing the effectiveness of software development using one approach (development tool) and another with which the comparison is made. This means that the embedded system itself (microcontroller) remains unchanged for each approach (development tool). Obviously,

Thus, we can formulate the basic principles of effective software development for embedded systems based on microcontrollers:

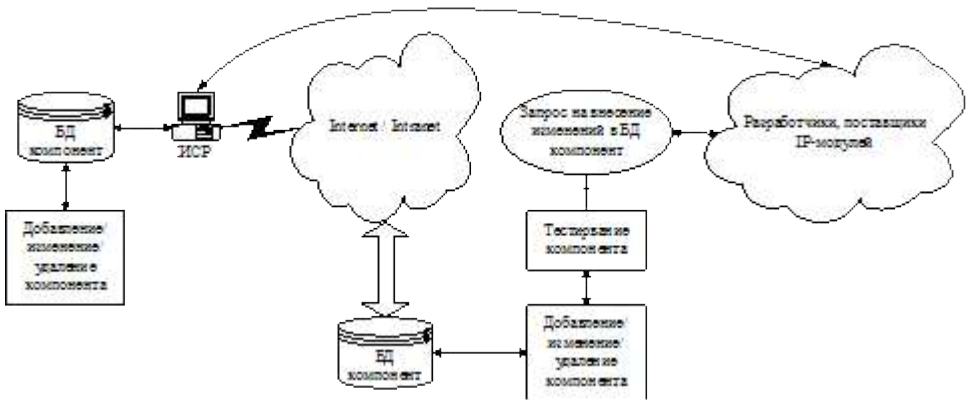
- compilation of fast and predictable (calculated) code programs (due to criticality time fulfillment a certain sequence of actions, time external event); reactions to
- Convenient and fast software development (due to the requirements of a rapidly developing market, the introduction of significant changes in finished devices and the development of new ones);
- small size of the compiled program in machine language (due to the requirements for the occupied size of program and data memory).

One of the options for solving the problems formulated above is the application of the reuse methodology (reuse) program code. In this case, developers focus on implementing their unique technology in the project, and not on implementing frequently used building blocks. In addition, the project can use not only modules (reusable components) developed by this company, but also modules developed by other companies (the market for IP modules (intellectual property modules) - reusable software modules is being created or supplemented).

These components should meet the following minimum requirements:

- portability (it is necessary to demonstrate the successful operation of the module on various processors and their families);
- data independence (there should be no dependence on data external to this module);
- the quality of the documentation (the ratio of comments or description file to the code should be no less than 1: 1);
- as well as requirements for reliability and usage statistics. The use of ready-made components in the development of software for embedded systems imposes additional requirements on the quality of the entire software system - the use of a single or automatically modified programming language, at the same time, the use of reusable software modules during development should not adversely affect the efficiency of the program itself, but should increase efficiency development of the program. Formally, a programming language is understood as a set of texts open for replenishment, written using a certain set of characters - the alphabet of the language (and hence the assembly language mnemonic system).

Thus, the problems formulated above can be solved by using a special software development tool for microcontrollers with support for components (IP modules) in a special programming language. To further improve the efficiency of software development for embedded systems, all this must be integrated into one development environment.



Rice. one.Scheme of interaction between developers, suppliers when using the IDS

The very development of a special programming language, universal

for a certain class of microcontrollers, is a non-trivial task. On the one hand, it should "cover" (in terms of capabilities) all class assemblers, and, on the other hand, the development efficiency (here - runtime) should not fall. Therefore, the programming language can only be assembler and only universal. After a thorough study of a number of programming languages (instruction systems of a certain class of MK), we can conclude about the composition of the mnemonic system of the universal assembler. In addition, an equally decisive factor in efficiency is the development of a compiler and translator from a universal assembly language to the required assembly language of a specific microcontroller. Obviously, it is better to make one translator for all families of assemblers (in this MK class). The proposed integrated development environment (IDE) and interaction scheme are shown in Fig. one.

A.A. Belykh, Yu.V. Gotovsky Problems of effective software development for embedded systems and ways to solve them // X
"IMEDIS", 2004, v.2 - C.381-
387