ISSN: 3071-186X Volume 01, Issue 09, October, 2026

Scienta Press Journals

American Innovator
Journal of Emerging Technologies and Research

Software Development and Debugging Tools

Larisa Alexandrovna Korotkova,
Senior Lecturer
Khikmatov Samandar
4th year student
Tashkent State Technical University, Department of Radio Engineering Devices and Systems +998 946905731.

Abstract

This article focuses on effective methods for software debugging, as well as the combined devices and kits used for these processes.

Keywords: Evaluation boards, in-circuit emulators (ICE), software simulator.

This work is Licensed under a Creative Commons Attribution 4.0 International License.

The most effective method for debugging microcontroller programs is the use of specialized professional debugging tools, including:
• in-circuit emulators (ICE) – hardware-software tools capable of replacing the target processor within the actual device;
• software simulators – software tools capable of simulating the operation of a microcontroller and its memory;

• debug monitors – a special program loaded into the memory of the system being debugged.;

• development boards (Evaluation Boards – evaluation boards) – своеобразные конструкторы для макетирования прикладных systems;

• ROM emulators – hardware and software tools that allow replacing the ROM of the debugged device with RAM, into which a program can be loaded from a computer via one of the standard communication channels.[1]

In addition, there are also combined devices and sets. As a rule, a simulator consists of a debugger, a CPU model, and memory. More advanced devices include models of built-in peripheral devices (timers, ports, ADCs, and interrupt systems). The simulator must be able to load

program files in all popular formats, display information about the state of the resources of the simulated microcontroller as fully as possible, and also provide the ability to simulate the execution of the loaded program in various modes. During the debugging process, the model executes the program, and the current state of the model is displayed on the computer monitor screen.

After loading the program into the simulator, the user can run it in step-by-step or continuous mode, set conditional or unconditional breakpoints, monitor and freely modify the contents of the memory cells and registers of the simulated microcontroller. The simulator allows you to quickly check the program execution logic and the correctness of arithmetic operations. Depending on the class of the debugger being used, some simulator models support high-level program debugging. The simulator may also include a number of additional software tools and an external environment interface. The presence of such an interface makes it possible to create and flexibly use a model of the microcontroller's external environment, which functions and influences the program being debugged according to a given algorithm.[2]

In a real system, the microcontroller usually "handles" reading information from the devices (sensors) connected to it, processing it, and issuing control signals to the actuators. To simulate the operation of a sensor in a simple simulator, you need to manually change the current state of the model of the peripheral device to which the sensor is connected in the real system. However, there are a number of modern software simulator developments in which a special file is typically used to simulate external conditions and situations input signals. This file defines the sequence of input signals arriving at the simulated device. For example, for AVR microcontrollers, this input file for the software simulator might look like this:

000000000:00
000000006: F1
000000015:18
000000109:1C
000000203:61
000000250:10
000000344:1F
000000391:71
999999999: ff

where each line contains – the cycle data arriving at a specified port. In some simulator models, this problem of simulating external signals is solved in such a way that the simulator has a built-in tool for creating

models of external devices connected to the microcontroller, including tools for graphically displaying information.

An obvious feature of software simulators is that the programs loaded into them run at a time scale different from the real one. However, the low cost and the ability to debug even in the absence of a mockup of the device being debugged make software simulators a very attractive debugging tool. It should also be noted that there is a whole class of errors that can only be detected using a simulator.[3]

The idea of the unity of software and hardware in systems based on microcontrollers is very important. Combining software development tools with hardware development tools can become an important advantage in device development. So-called integrated development environments significantly facilitate and speed up the process of developing and debugging microprocessor systems. They combine a text editor for writing source code, assembler and C translators, a linker, a debugger, reference information on microcontrollers, and other tools that a developer needs. The configuration of translators, the linker, and other components is carried out not by specifying keys in the command line, but in the form of dialog boxes, where you just need to put “checkmarks” in the required places. The conversion of program source texts into a machine code file is triggered by pressing a single key. [3]

The emergence of integrated software development environments further increased the efficiency of creating programs for microcontrollers, allowing the developer to focus on the essence of the problem being solved and to abstract away from the specific details of its implementation. Several companies produce integrated software development packages; packages from different manufacturers are similar in terms of functionality, but they differ in the service capabilities they offer, the user-friendliness of their operation, and the quality of the generated machine code. [4]

List Of Literature

1. Programmable logic integrated circuits based on CMOS structures and their applications. / P.P. Maltsev, N.I. Garbuzov, A.P. Sharapov, A.A. Knyshev. – Moscow: Energoatomizdat, 2021. – 158 p.
2. Solovyov V.V., Vasiliev A.G. Programmable logic integrated circuits and their applications. – Minsk: Belarusian Science, 2020. – 270 p.
3. Bursky D. Embedded Logic and Memory Find a Home in FPGA. – Electronic Design, 2022, No. 14, pp. 43–56.
4. Chang D., Mazek-Sadowska M. Dynamically Reconfigurable FPGA. – JEEE Transition on Computers, 2020, No. 6, pp. 565–578.