**MINISTRY OF SCIENCE AND HIGHER EDUCATION OF THE RUSSIAN Federation**

Federal State Budgetary Educational Institution of Higher Education

"Kazan National Research Technical University named after A. N. Tupolev-KAI"

(KNRTU-KAI)

Institute of Computer Technologies and Information Security

**Department of Computer Systems**

Report № 3

**«AVR Simulation with the Microchip Studio 7.0 »**

**«Architecture of embedded systems»**



Student 4167 Edwin G. Carreno

(group number) (signature, data) (full name)

Associate professor of the computer systems’ department Daria V. Shirshova

Grade

(signature, data)

Kazan 2021

**Content**

1. Tasks.
2. Control Questions.
3. Summary
4. **Tasks**
5. Write a program, calculating Fibonacci numbers. In a Fibonacci series each number is the sum of the two previous numbers.

The Fibonacci series it is expressed as follows:

Graphical user interface, text, application

Description automatically generated

Figure 1. Function fibonacciGenerator calculate the value of the nth term.

1. Use CodeVision AVR to write a program.

Steps to create the CodeVision AVR program are explicitly explained in the guidelines for Laboratory N° 2 “CodeVision AVR C Compiler Introduction and Exercises”. Current report will not cover those steps.

1. Create a project, build it and simulate with the AVR Studio simulator.

First, when a project in CodeVision is created and compiled, a bunch of files are generated by the program. Those files include executable, linking, objects, etc. For simulation purposes using Microchip Studio or AVR studio the main file that serves as a simulation file is a file with the extension .cof*.* Figure 2 shows the .cof file inside the folder created for the CodeVision project.

Graphical user interface, text, application, email

Description automatically generated

Figure 2. File with extension .cof is the main input for debugging purposes in Microchip Studio if the compiler was the CodeVision C compiler.

Once the .cof file is located, next steps include opening Microchip Studio and entering the File tab, and click on the Open menu, Open Object File for Debugging. Figure 3 shows the menu window to open a file for debugging in Microchip Studio 7.

Text

Description automatically generated

Figure 3. Open Object File for Debugging option allows to Microchip Studio open simulation files as is the case for the .cof file.

Graphical user interface, text, application

Description automatically generated

Figure 4. Project creation wizard window for simulation using debug files.

Graphical user interface, application

Description automatically generated

Figure 5. Selection device window. For laboratory, the microcontroller ATmega 328 was selected in both platforms (CodeVision and Microchip Studio).

Following screenshots shows the process of debugging using the Microchip Studio program. In there, the main steps are related to the Step Into and Step Over functions that serves as controlled points to check step by step what is going on with the simulation. It is relevant to note that the simulation file cannot be modified by Microchip Studio. If you desire to modify it, you should make changes in the CodeVision project, compile and reload the file into the Microchip Studio (it detects any change in the file and request a confirmation to reload the file).

On the other hand, Microchip offers a Watch windows that allows to you check the interest variables in real time, step by step. For this laboratory, this window is used to check if the successive summations are calculated and updated as it is expected according to the Fibonacci formula given at the beginning of this document.

Graphical user interface, text, application

Description automatically generated

Figure 6. Debugging window, in this part you could see the value of the different registers initialized by CodeVision.

Graphical user interface, text, application

Description automatically generated

Figure 7. fibonacciGenerator is called with an input value (10) and the result should be store in another integer value called fibo.

Graphical user interface, text, application

Description automatically generated

Figure 8. Step into function enters to the fibonacciGenerator function. In the Watch window you see different values related to the series calculation.

Graphical user interface, text, application

Description automatically generated

Figure 9. After 9 iterations (nth -1 term = 9) the last value calculated corresponds to the value 21.

Graphical user interface, text, application

Description automatically generated

Figure 10. Last value calculated is the value 34 that corresponds to the 10th term of the Fibonacci series.

Graphical user interface, text, application

Description automatically generated

Figure 11. The fibo variable stores the value of 34 generated by the fibonacciGenerator function. In order to see the last value or trick the debugger, a delay function was introduced it is an empty function.

1. Demonstrate the work of your program to the instructor.
2. Open the Watch window, showing the variables.

Figures 6 to 11 show the Watch windows for each step that it was made in the debugger.

1. Explain the results.

Successive summations in order to calculate the Fibonacci n-th value were made by the simulator embedded into the Microchip Studio IDE. In there a debugger was used to check the value of every single variable created in the program. If this program were uploaded into a microcontroller, it is important to note that the maximum value is limited by the size of the integer variable (usually around 215-1).

1. **Control Questions**
2. **Why do we need a Workspace window?**

A workspace windows is used to check the value of the registers at any time we want to check the value when a debugger is used.

1. **Why do we need a Watch window?**

A Watch window is used to check the value of any variable or register in the program. Usually, those values are updated accordingly to the current step in the debugger.

1. **Why do we need an Output window?**

In order to get relevant information related to the microcontroller, compiling, building, etc. It serves as a start point when a program fails, in this output windows usually serves a feedback in case your program fails in compilation time or execution time.

1. **Why do we need an IDE Toolbar?**

An IDE toolbar is useful to call functions created for different purposes, in the case of the debugger, those functions are Starting the debugger, Stop the debugger, Step Into, Step Out, etc. In an IDE toolbar there are a variety of functions created to develop at different stages of the building process.

1. Summary

Current laboratory guidelines allow to the user to manipulate different platforms in order to compile an embedded C project and to simulate their behavior. CodeVision AVR C Compiler was used to create, initialize and compile a C project. On the other hand, Microchip Studio utilized the .cof file generated by CodeVision in order to perform simulations under the embedded debugger.