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 № 2

«Code VisionAVR C Compiler Introduction and Exercises»

«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. Laboratory practice

a. Control Questions (refers to the steps in Tasks section)

i. How to create a new project?

Steps [i to ii] cover the process of creating a new project in CodeVision AVR.

ii. How to build a project?

Step [ix] shows the process of building a project in CodeVision AVR. In this case the menu option could be easily found in *Project → Build.*

iii. How to configure a project?

Steps [iii to vi] show the process of configuring a new project in CodeVision AVR.

iv. How to compile a project?

Step [ix] shows the process of compile the source code in CodeVision AVR.

b. Tasks

i. Create a new project by selecting: File/New/Select Project.

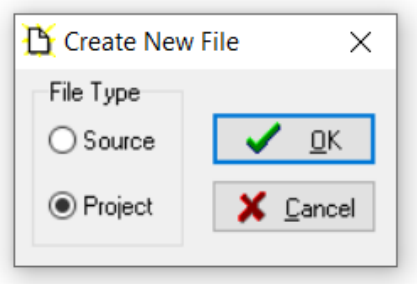

Figure 1.Menu window for creating new sources or projects in CodeVision AVR

ii. Specify that the CodeWizardAVR will be used for producing the C source and project files: Use the CodeWizard? Yes

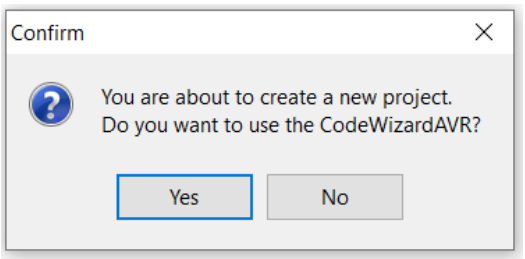

Figure 2. Confirmation window appears to request confirmation about usage of CodeWizardAVR.

iii. *In the CodeWizardAVR windows specify the chip type and clock frequency:*

- *Chip:* ATmega8515
- *Clock:* 3.86 MHz

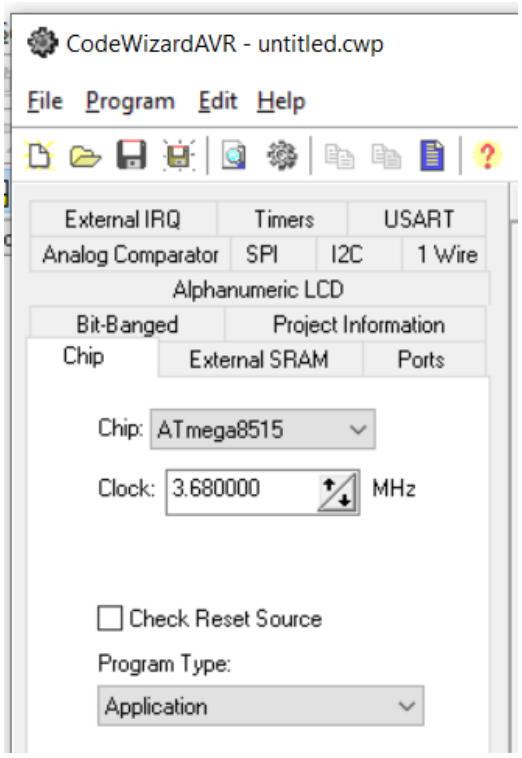

Figure 3. Chip configuration window, in there it is possible select the device and set the master clock frequency.

iv. *Configure the I/O ports:*

- *Ports:* Port B
- *Data Direction:* All pin's port configured as Outputs.
- *Output Value:* 1's

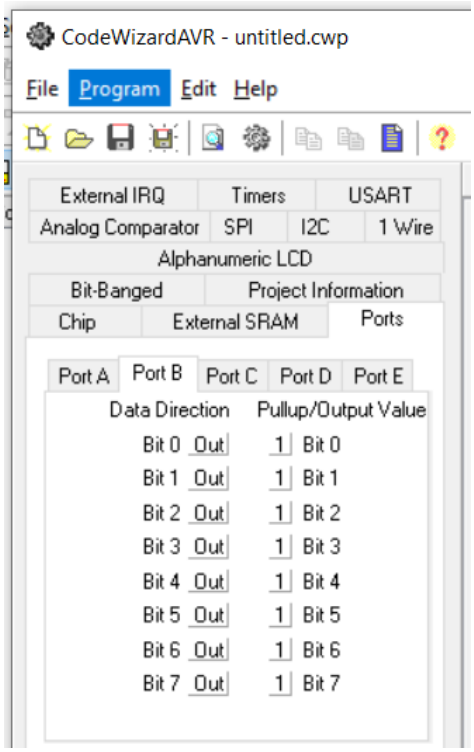

Figure 4. Port configuration menu, those settings correspond to the Port B.

v. *Configure Timer1*
   - *Timers:* Timer1
   - *Clock Value:*3.594 kHz
   - *Interrupt on:* Timer1 Overflow
   - *Value:* 0xF8FB

.

For this laboratory practice, the main goal is toggling a LED two times per second. In order to achieve this goal, the Timer1 embedded into the Microcontroller will be used. Timer1 allows to the microprocessor be interrupted when an overflow is reached by the counter. In the case of the ATmega8515, Timer1 has a resolution of 16-bit, it means that the maximum value of the counter will be 65535 (2^{16} - 1) or 0xFFFF in hexadecimal. Figure 5 shows how many cycles are required to achieve the overflow and then to generate an interruption.

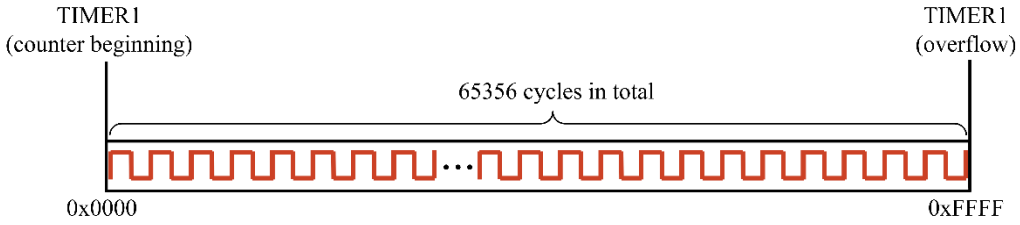

Figure 5. TIMER1 has a counter that starts by default in 0(0x0000) and ends in 65355(0xFFFF), the total number of cycles counted are 65536 (0x10000) before the overflow event.

Before counting cycles, it is needed to define which clock and what the frequency value should be. Whether the count is made using the original master signal of 3.68 MHz (3.68e6 cycles/second) this overpasses the capacity of the counter (65536 cycles). For that reason, in the Figure 6 it is shown the prescaling path (factor 1/1024) using for decreasing the frequency, and the resulting signal has now a frequency of 3.594 kHz or 3594 Hz approximately.

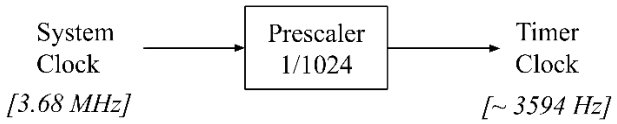

Figure 6. TIMER1 clock source using a prescaler for decreasing frequency.

In order to get two interruptions per second, it is needed to force the counter to start with a different value (different from 0x0000). TCNT1 is the register thar will store this new beginning value. Given the frequency of 3594 Hz (3594 cycles/second), how many cycles should the TIMER1 count before entering an interruption every 0.5 seconds? Answering this if easy as follows:

$$3594 \text{ cycles} \rightarrow 1 \text{ seconds}$$
$$x \text{ cycles} \rightarrow 0.5 \text{ seconds}$$

In the equation, they are required 1797 cycles and then the interruption should occur. On the other hand, it is easy to see that the fixed value to force the counter to start should be the subtraction of 1797 from the total number of cycles that the counter is capable to count (65536 or 0xFFF).

$$TCNT1 = 65536 - 1797 = 63739 \; or \; 0xF8FB$$

Figure 7 shows the cycles distribution required for generating two interruptions per second using a clock signal of 3.594 kHz and a timer of 16-bit resolution.

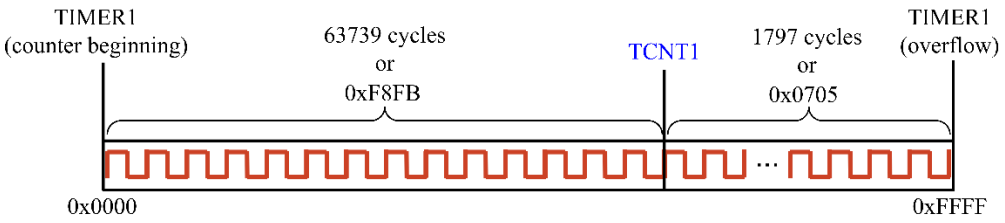

Figure 7. TCNT1 should store the value of 0xF8FB in order to generate an interruption every 0.5 seconds.

Once all the required values for the timer and the interruption settings are understood, next is setting those values in the Wizard as follows in the Figure 8.

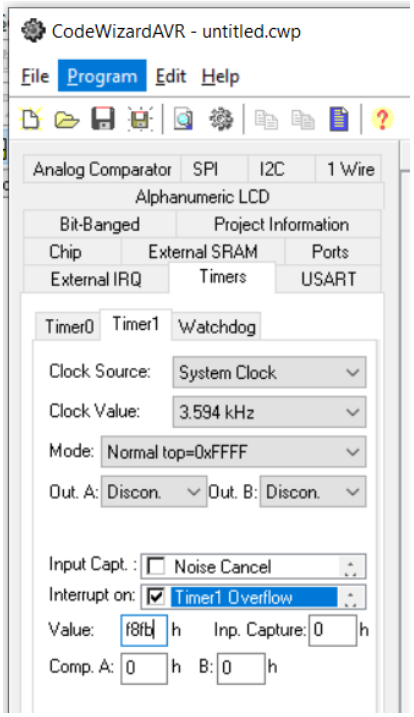

Figure 8. TIMER1 settings, clock sourcing, prescaling and interruption event enabling.

vi. *Generate the C source, C project and CodeWizardAVR project files by selecting: File/Generate, Save and Exit / Create new directory: led.c, led.prj and led.cwp files following the instructions given by the Wizard.*

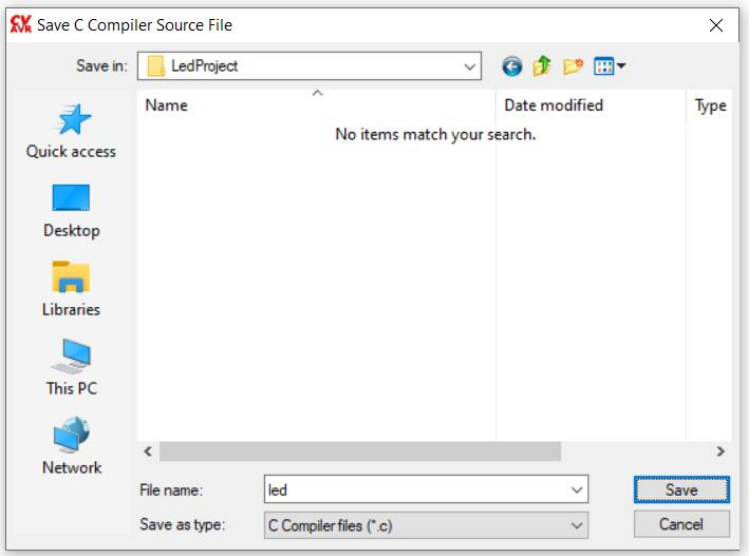

Figure 9. Wizard creates a C file, for this laboratory this file it is named as led.c .

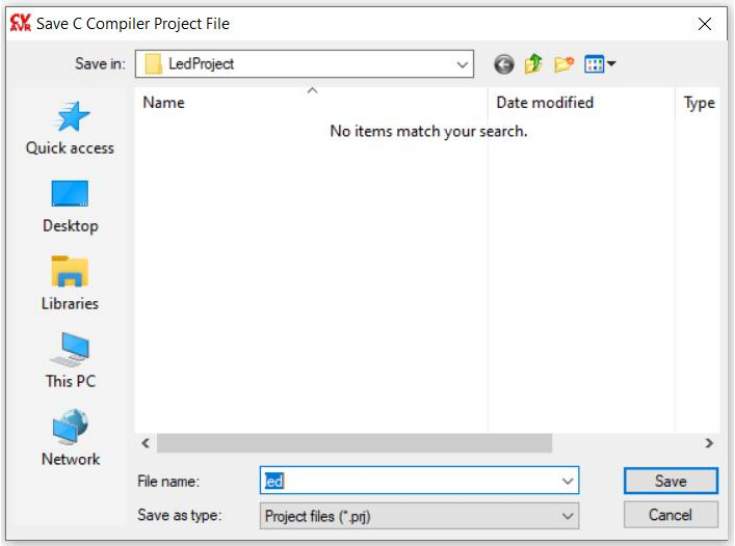

Figure 10. Wizard creates a pjr file, which is the file for the embedded project.

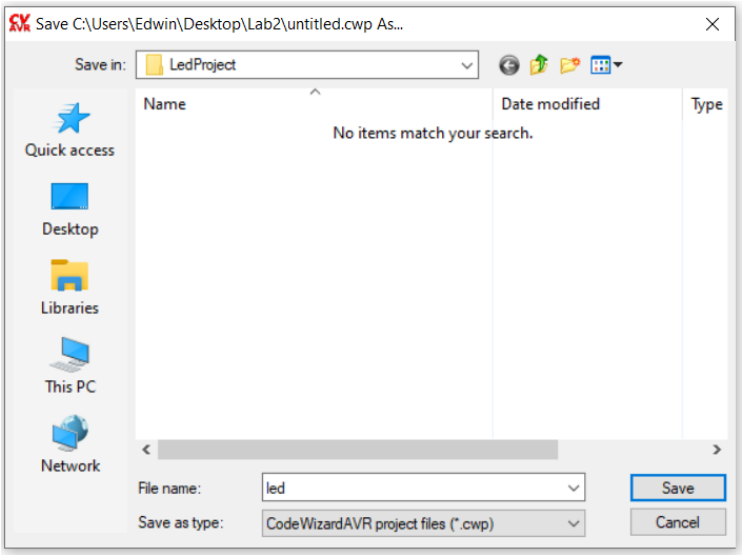

Figure 11. Wizard creates a .cwp file that stores the CodeVisionAVR metadata for this project.

vii. Edit the C source code.

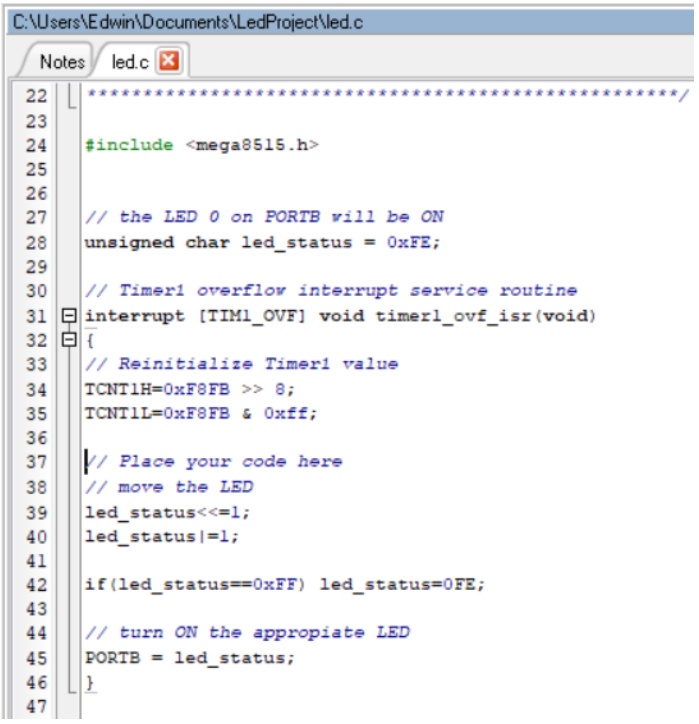

Figure 12. C code for geneating two interruptions in a second, once the interruption ocurrs the LED's value toggles in each event.

viii. *View or modify the project configuration by selecting Project → Configure → After Build → Program the chip → SCK frequency: 230400 Hz.*

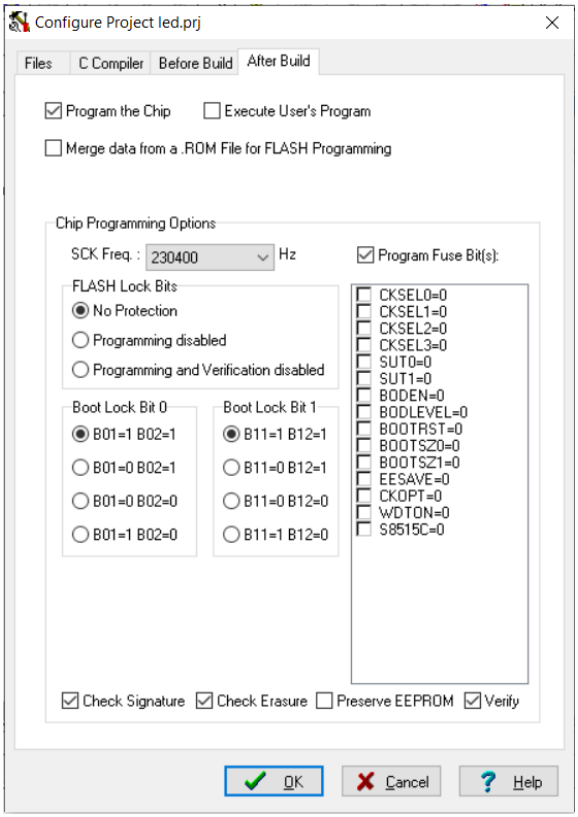

Figure 13. After build settings for programming directly in the board.

ix. *Compile the program by selecting: Project → Build.*

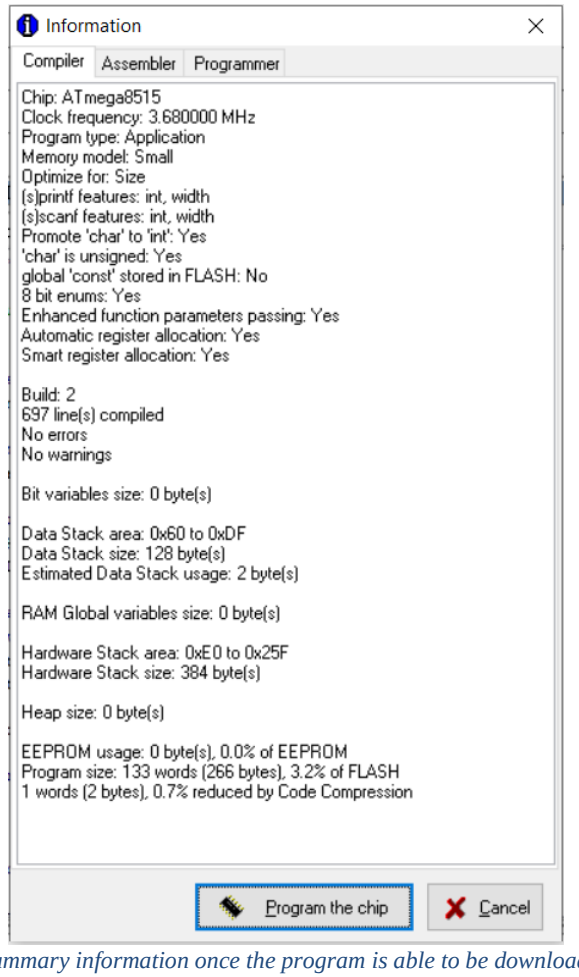

Figure 14. Summary information once the program is able to be downloaded into the Integrated Circuit.

x. *Automatically program the ATmega8515 chip on the STK500 starter kit:Apply power → Information → Program chip.*

   Once you have downloaded the code generated by the Wizard and modified by us in previous steps, you should see blinking a LED whenever you connect it into the PORTB.

2. Summary

In this laboratory practice it was covered those notions for configuring the microcontroller peripherals using a graphical tool that generate automatically the correct values in the associated registers. On the other hand, the TIMER1 was configured in order to generate interruptions into the microprocessor. Interruptions are a core topic in embedded systems that deals with precise applications in real time, due to the fact they could affect the processor status any time it would be desired or required. Finally, using graphical tools to configure the microcontroller are useful for prototyping or industry development, however, is it extremely relevant understand what you are setting, especially in debugging phases.