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	[illegible]	Edwin G. Carreno
	(group number)	(signature, data)	(full name)

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

Grade ___________________

____
(signature, data)

Kazan 2021

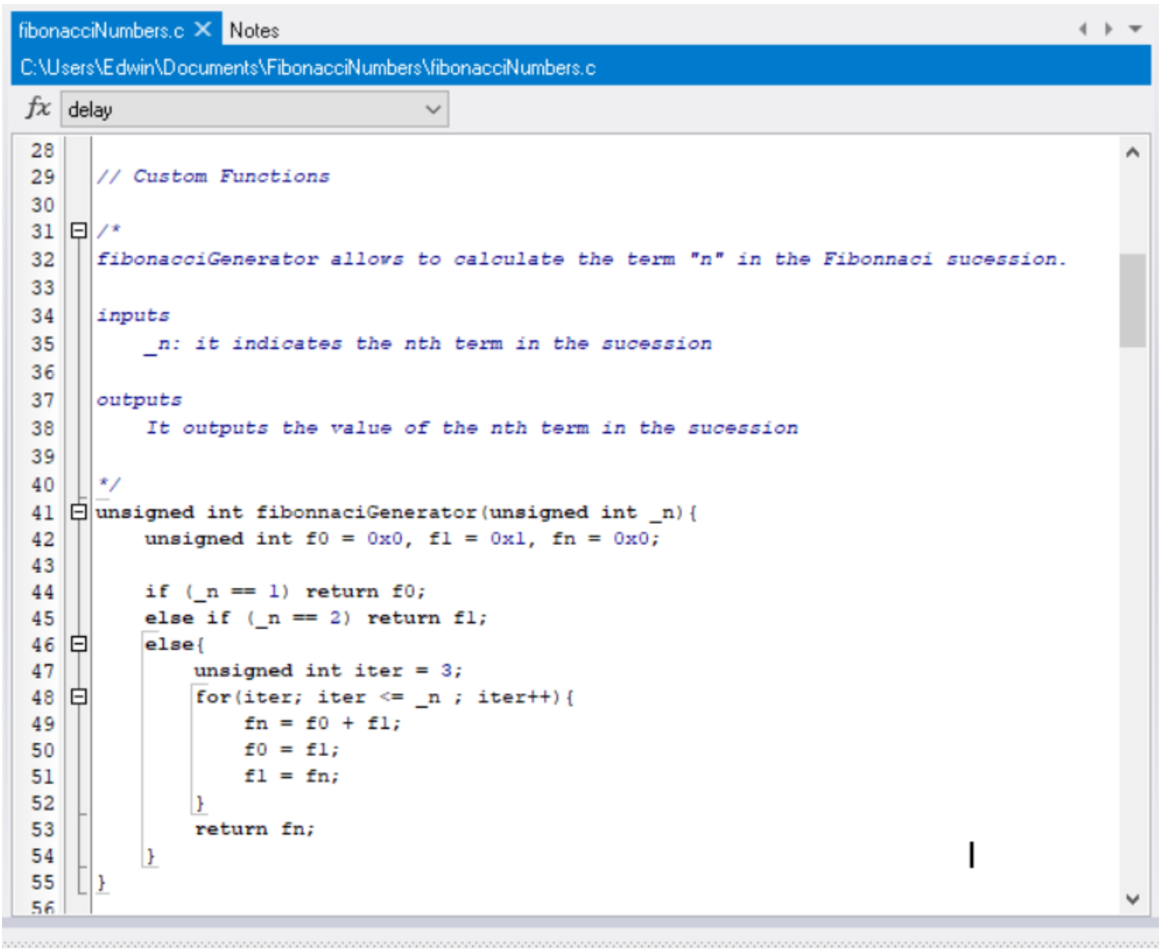
Content

1. Tasks

a. 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:

$$f_0 = 0\,, f_1 = 1\,, \qquad f_n = f_{n-1} + f_{n-2}$$


fibonacciNumbers.c
Notes
C:\Users\Edwin\Documents\FibonacciNumbers\fibonacciNumbers.c
fx
delay

```
// Custom Functions

/*
fibonacciGenerator allows to calculate the term "n" in the Fibonnaci sucession.

inputs
    _n: it indicates the nth term in the sucession

outputs
    It outputs the value of the nth term in the sucession

*/
unsigned int fibonnaciGenerator(unsigned int _n){
    unsigned int f0 = 0x0, f1 = 0x1, fn = 0x0;

    if (_n == 1) return f0;
    else if (_n == 2) return f1;
    else{
        unsigned int iter = 3;
        for(iter; iter <= _n ; iter++){
            fn = f0 + f1;
            f0 = f1;
            f1 = fn;
        }
        return fn;
    }
}
```

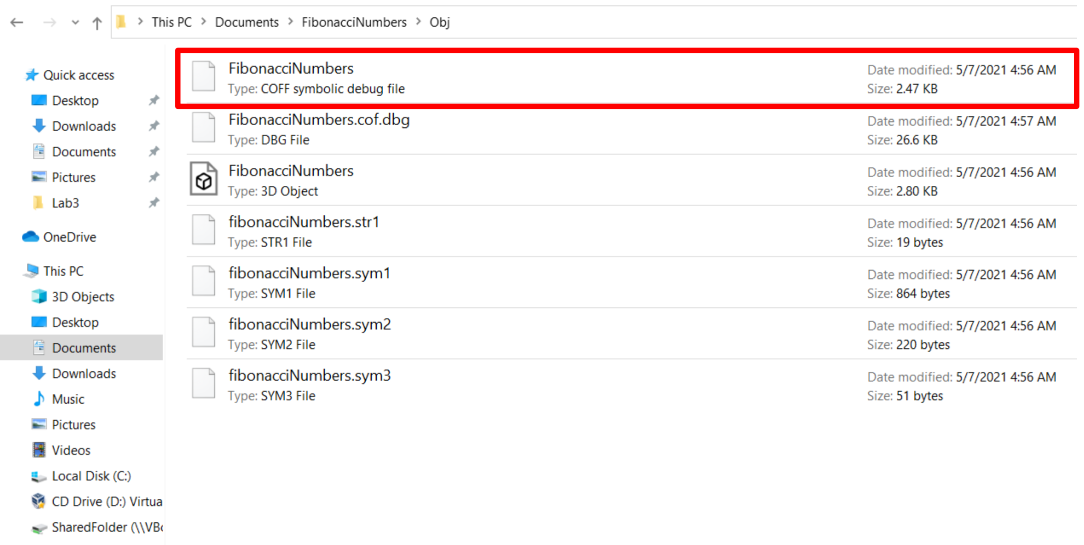
Figure 1. Function fibonacciGenerator calculate the value of the nth term.

b. 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.

c. 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.



FibonacciNumbers	Type: COFF symbolic debug file	Date modified: 5/7/2021 4:56 AM	Size: 2.47 KB
FibonacciNumbers.cof.dbg	Type: DBG File	Date modified: 5/7/2021 4:57 AM	Size: 26.6 KB
FibonacciNumbers	Type: 3D Object	Date modified: 5/7/2021 4:56 AM	Size: 2.80 KB
fibonacciNumbers.str1	Type: STR1 File	Date modified: 5/7/2021 4:56 AM	Size: 19 bytes
fibonacciNumbers.sym1	Type: SYM1 File	Date modified: 5/7/2021 4:56 AM	Size: 864 bytes
fibonacciNumbers.sym2	Type: SYM2 File	Date modified: 5/7/2021 4:56 AM	Size: 220 bytes
fibonacciNumbers.sym3	Type: SYM3 File	Date modified: 5/7/2021 4:56 AM	Size: 51 bytes

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.

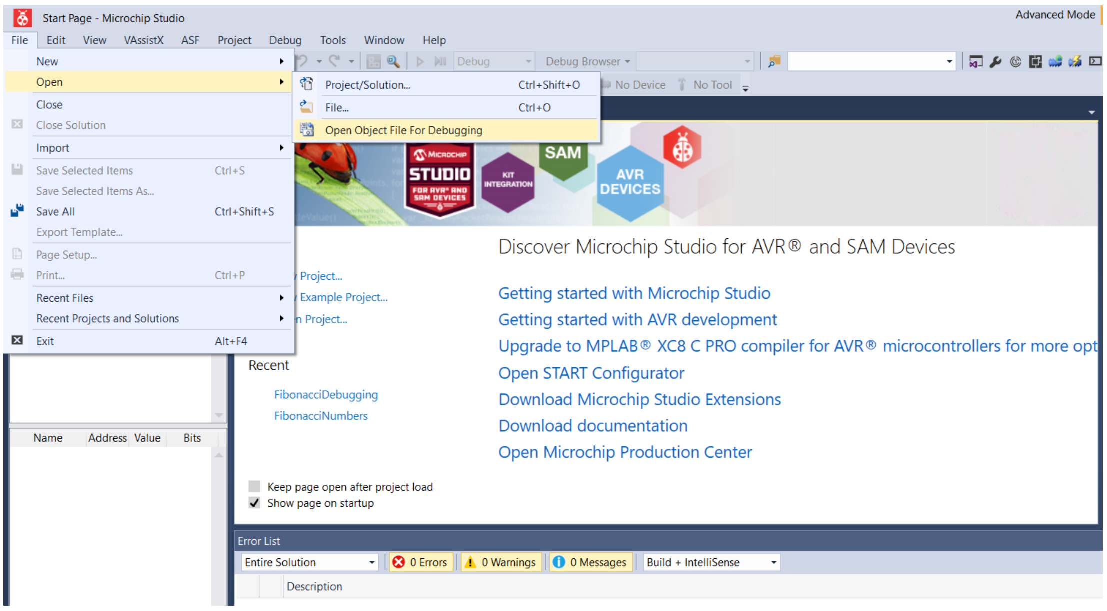
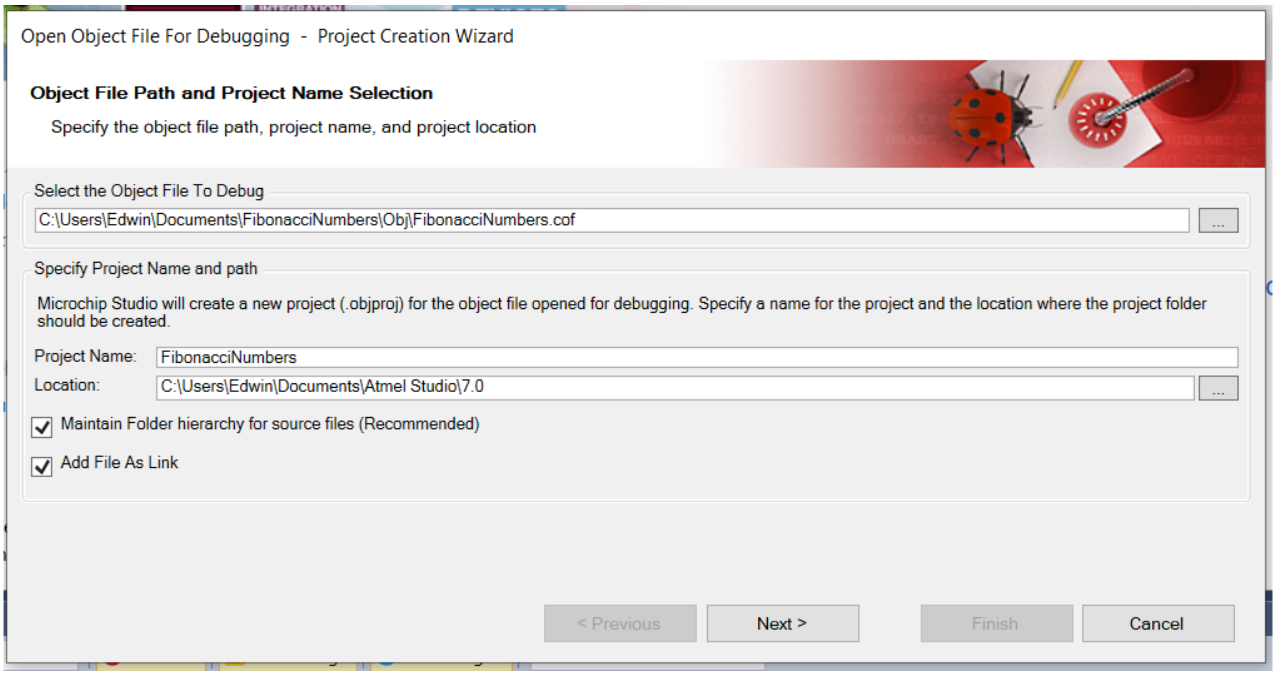

Start Page - Microchip Studio
Advanced Mode
File Edit View VAssistX ASF Project Debug Tools Window Help
New
Open
Close
Close Solution
Import
Save Selected Items Ctrl+S
Save Selected Items As...
Save All Ctrl+Shift+S
Export Template...
Page Setup...
Print... Ctrl+P
Recent Files
Recent Projects and Solutions
Exit Alt+F4
Project/Solution... Ctrl+Shift+O
File... Ctrl+O
Open Object File For Debugging
Debug
Debug Browser
No Device
No Tool
STUDIO FOR AVR® AND SAM DEVICES
KIT INTEGRATION
SAM
AVR DEVICES
Discover Microchip Studio for AVR® and SAM Devices
Getting started with Microchip Studio
Getting started with AVR development
Upgrade to MPLAB® XC8 C PRO compiler for AVR® microcontrollers for more opt
Open START Configurator
Download Microchip Studio Extensions
Download documentation
Open Microchip Production Center
Project...
Example Project...
Project...
Recent
FibonacciDebugging
FibonacciNumbers
Keep page open after project load
Show page on startup
Name Address Value Bits
Error List
Entire Solution
0 Errors
0 Warnings
0 Messages
Build + IntelliSense
Description

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


Open Object File For Debugging - Project Creation Wizard

Object File Path and Project Name Selection

Specify the object file path, project name, and project location

Select the Object File To Debug

C:\Users\Edwin\Documents\FibonacciNumbers\Obj\FibonacciNumbers.cof

Specify Project Name and path

Microchip Studio will create a new project (.objproj) for the object file opened for debugging. Specify a name for the project and the location where the project folder should be created.

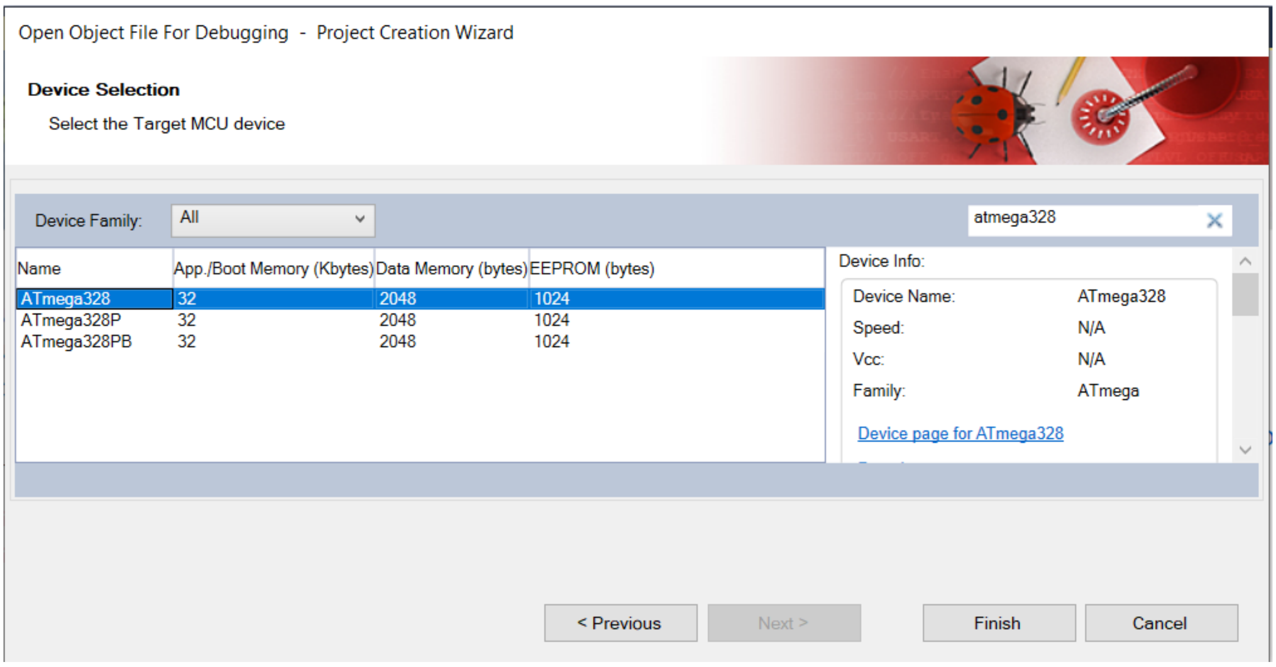
Project Name: FibonacciNumbers

Location: C:\Users\Edwin\Documents\Atmel Studio\7.0

✓ Maintain Folder hierarchy for source files (Recommended)

✓ Add File As Link

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


Open Object File For Debugging - Project Creation Wizard

Device Selection

Select the Target MCU device

Device Family: All

atmega328

Name	App./Boot Memory (Kbytes)	Data Memory (bytes)	EEPROM (bytes)
ATmega328	32	2048	1024
ATmega328P	32	2048	1024
ATmega328PB	32	2048	1024

Device Info:

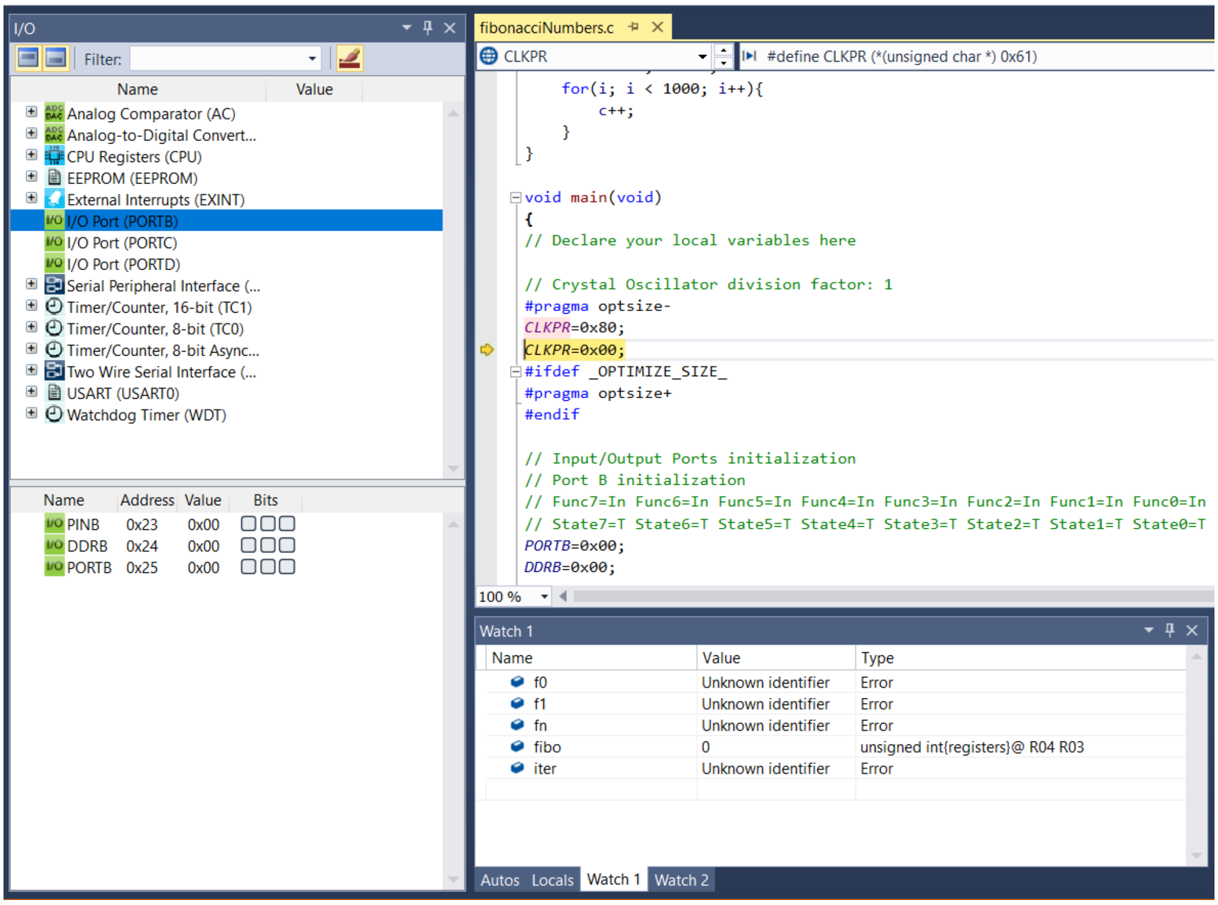
Device Name:	ATmega328
Speed:	N/A
Vcc:	N/A
Family:	ATmega

Device page for ATmega328

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.



Name	Value
Analog Comparator (AC)	
Analog-to-Digital Convert...	
CPU Registers (CPU)	
EEPROM (EEPROM)	
External Interrupts (EXINT)	
I/O Port (PORTB)	
I/O Port (PORTC)	
I/O Port (PORTD)	
Serial Peripheral Interface (...	
Timer/Counter, 16-bit (TC1)	
Timer/Counter, 8-bit (TC0)	
Timer/Counter, 8-bit Async...	
Two Wire Serial Interface (...	
USART (USART0)	
Watchdog Timer (WDT)	

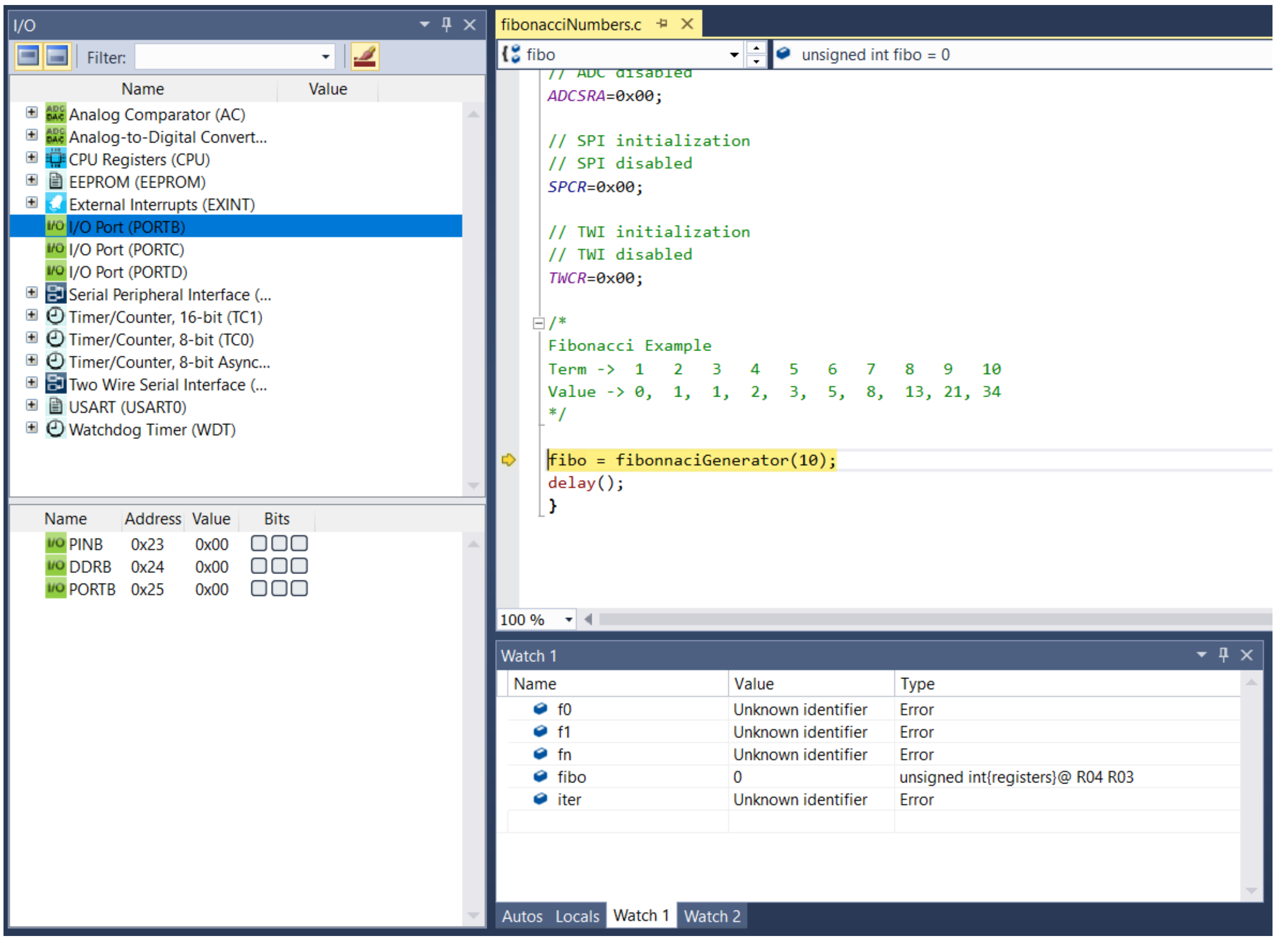
Name	Address	Value	Bits
PINB	0x23	0x00	☐☐☐
DDRB	0x24	0x00	☐☐☐
PORTB	0x25	0x00	☐☐☐

```
        for(i; i < 1000; i++){
            c++;
        }
}

void main(void)
{
// Declare your local variables here

// Crystal Oscillator division factor: 1
#pragma optsize-
CLKPR=0x80;
CLKPR=0x00;
#ifdef _OPTIMIZE_SIZE_
#pragma optsize+
#endif

// Input/Output Ports initialization
// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTB=0x00;
DDRB=0x00;
```

Watch 1

Name	Value	Type
f0	Unknown identifier	Error
f1	Unknown identifier	Error
fn	Unknown identifier	Error
fibo	0	unsigned int{registers}@ R04 R03
iter	Unknown identifier	Error

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



Name	Value
Analog Comparator (AC)	
Analog-to-Digital Convert...	
CPU Registers (CPU)	
EEPROM (EEPROM)	
External Interrupts (EXINT)	
I/O Port (PORTB)	
I/O Port (PORTC)	
I/O Port (PORTD)	
Serial Peripheral Interface (...	
Timer/Counter, 16-bit (TC1)	
Timer/Counter, 8-bit (TC0)	
Timer/Counter, 8-bit Async...	
Two Wire Serial Interface (...	
USART (USART0)	
Watchdog Timer (WDT)	

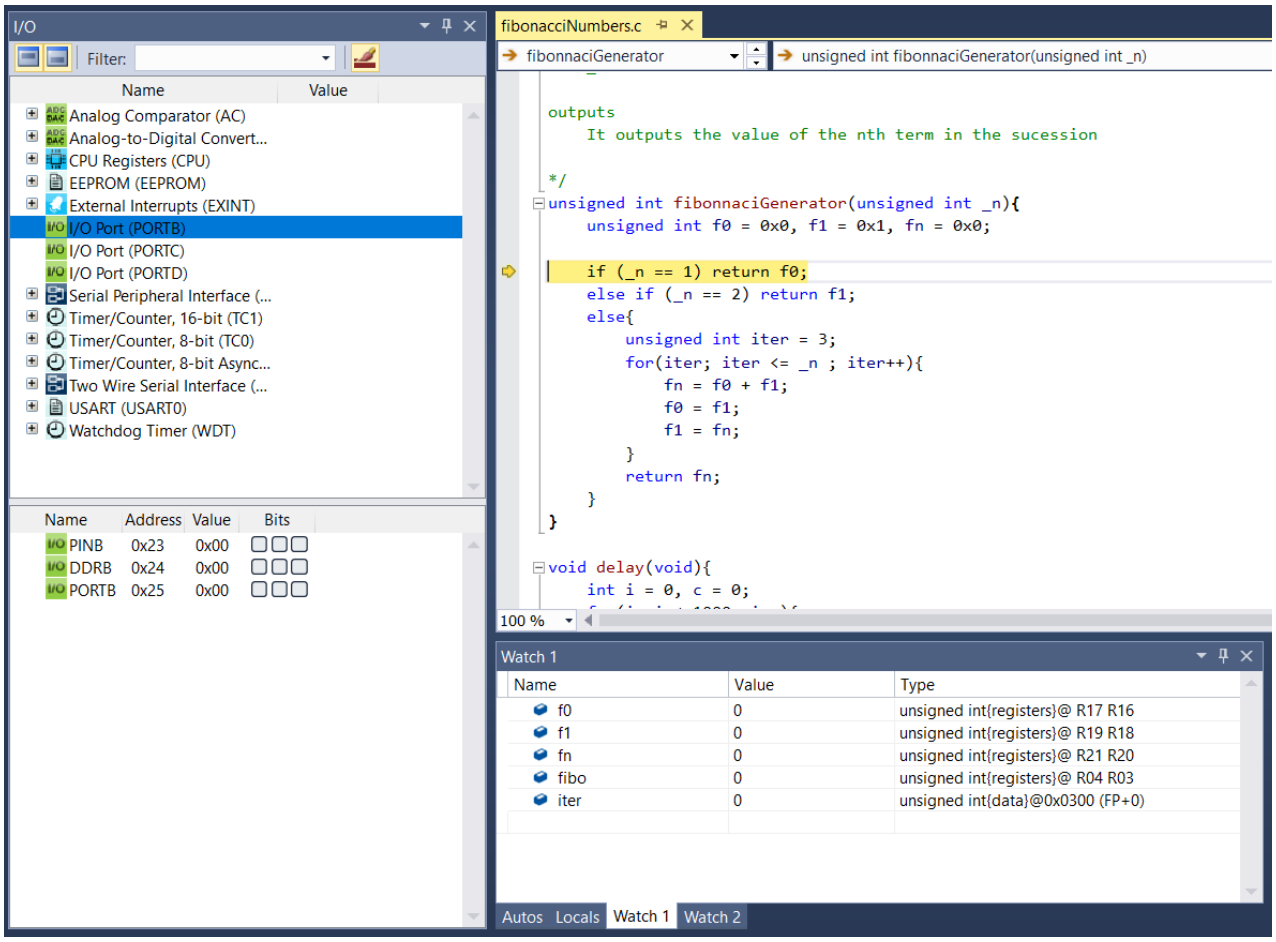
Name	Address	Value	Bits
PINB	0x23	0x00	☐☐☐
DDRB	0x24	0x00	☐☐☐
PORTB	0x25	0x00	☐☐☐

```
// ADC disabled
ADCSRA=0x00;

// SPI initialization
// SPI disabled
SPCR=0x00;

// TWI initialization
// TWI disabled
TWCR=0x00;

/*
Fibonacci Example
Term ->  1   2   3   4   5   6   7   8   9   10
Value -> 0,  1,  1,  2,  3,  5,  8,  13, 21, 34
*/

fibo = fibonnaciGenerator(10);
delay();
}
```

Watch 1

Name	Value	Type
f0	Unknown identifier	Error
f1	Unknown identifier	Error
fn	Unknown identifier	Error
fibo	0	unsigned int{registers}@ R04 R03
iter	Unknown identifier	Error

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


I/O

Filter:

Name	Value
Analog Comparator (AC)	
Analog-to-Digital Convert...	
CPU Registers (CPU)	
EEPROM (EEPROM)	
External Interrupts (EXINT)	
I/O Port (PORTB)	
I/O Port (PORTC)	
I/O Port (PORTD)	
Serial Peripheral Interface (...	
Timer/Counter, 16-bit (TC1)	
Timer/Counter, 8-bit (TC0)	
Timer/Counter, 8-bit Async...	
Two Wire Serial Interface (...	
USART (USART0)	
Watchdog Timer (WDT)	

Name	Address	Value	Bits
PINB	0x23	0x00	☐☐☐
DDRB	0x24	0x00	☐☐☐
PORTB	0x25	0x00	☐☐☐

fibonacciNumbers.c

fibonnaciGenerator → unsigned int fibonnaciGenerator(unsigned int _n)

```
outputs
    It outputs the value of the nth term in the sucession

*/
unsigned int fibonnaciGenerator(unsigned int _n){
    unsigned int f0 = 0x0, f1 = 0x1, fn = 0x0;

    if (_n == 1) return f0;
    else if (_n == 2) return f1;
    else{
        unsigned int iter = 3;
        for(iter; iter <= _n ; iter++){
            fn = f0 + f1;
            f0 = f1;
            f1 = fn;
        }
        return fn;
    }
}

void delay(void){
    int i = 0, c = 0;
```

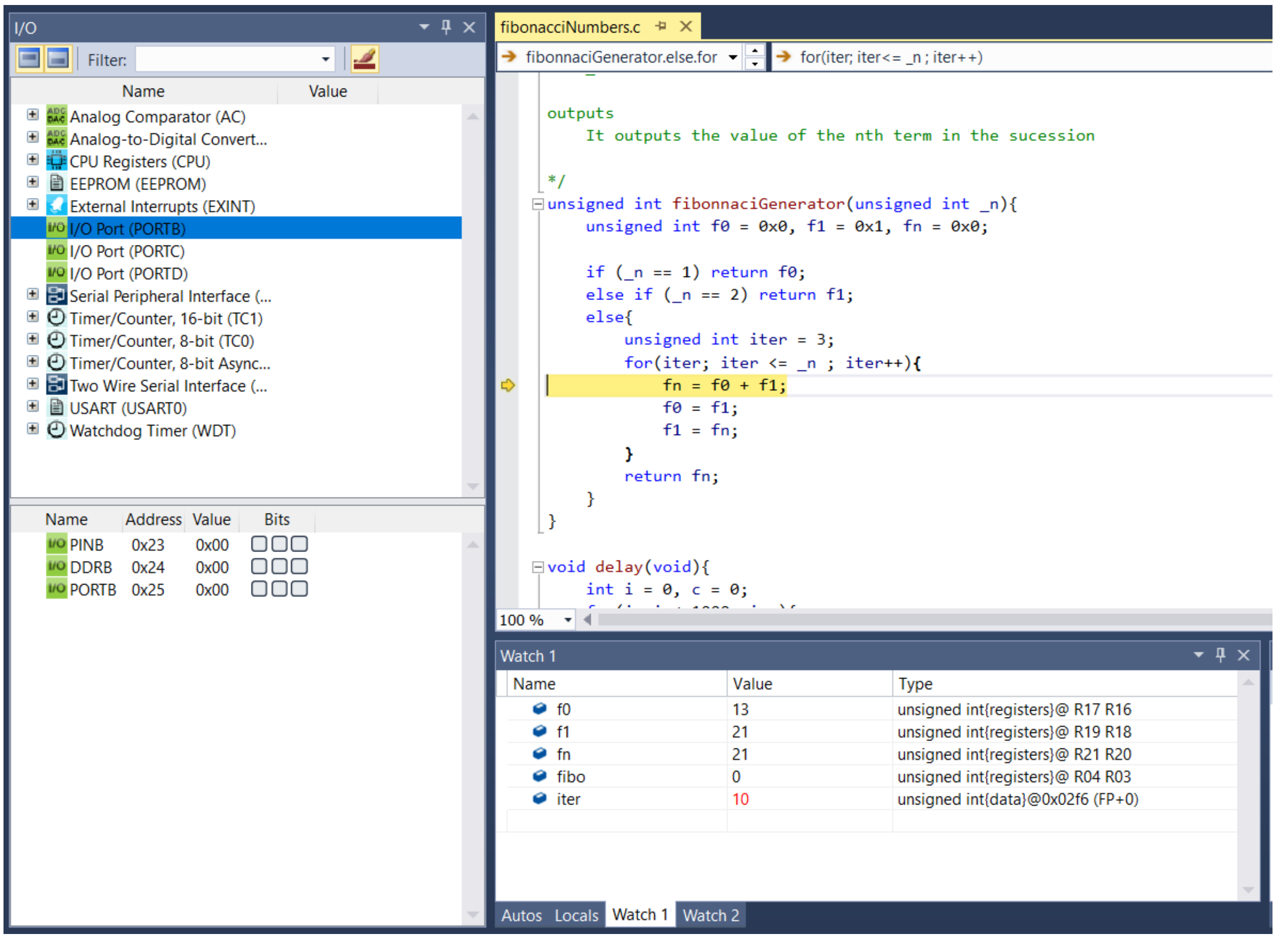
100 %

Watch 1

Name	Value	Type
f0	0	unsigned int{registers}@ R17 R16
f1	0	unsigned int{registers}@ R19 R18
fn	0	unsigned int{registers}@ R21 R20
fibo	0	unsigned int{registers}@ R04 R03
iter	0	unsigned int{data}@0x0300 (FP+0)

Autos Locals Watch 1 Watch 2

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


I/O

Filter:

Name	Value
Analog Comparator (AC)	
Analog-to-Digital Convert...	
CPU Registers (CPU)	
EEPROM (EEPROM)	
External Interrupts (EXINT)	
I/O Port (PORTB)	
I/O Port (PORTC)	
I/O Port (PORTD)	
Serial Peripheral Interface (...	
Timer/Counter, 16-bit (TC1)	
Timer/Counter, 8-bit (TC0)	
Timer/Counter, 8-bit Async...	
Two Wire Serial Interface (...	
USART (USART0)	
Watchdog Timer (WDT)	

Name	Address	Value	Bits
PINB	0x23	0x00	☐☐☐
DDRB	0x24	0x00	☐☐☐
PORTB	0x25	0x00	☐☐☐

fibonacciNumbers.c

fibonnaciGenerator.else.for → for(iter; iter<= _n ; iter++)

```
outputs
    It outputs the value of the nth term in the sucession

*/
unsigned int fibonnaciGenerator(unsigned int _n){
    unsigned int f0 = 0x0, f1 = 0x1, fn = 0x0;

    if (_n == 1) return f0;
    else if (_n == 2) return f1;
    else{
        unsigned int iter = 3;
        for(iter; iter <= _n ; iter++){
            fn = f0 + f1;
            f0 = f1;
            f1 = fn;
        }
        return fn;
    }
}

void delay(void){
    int i = 0, c = 0;
```

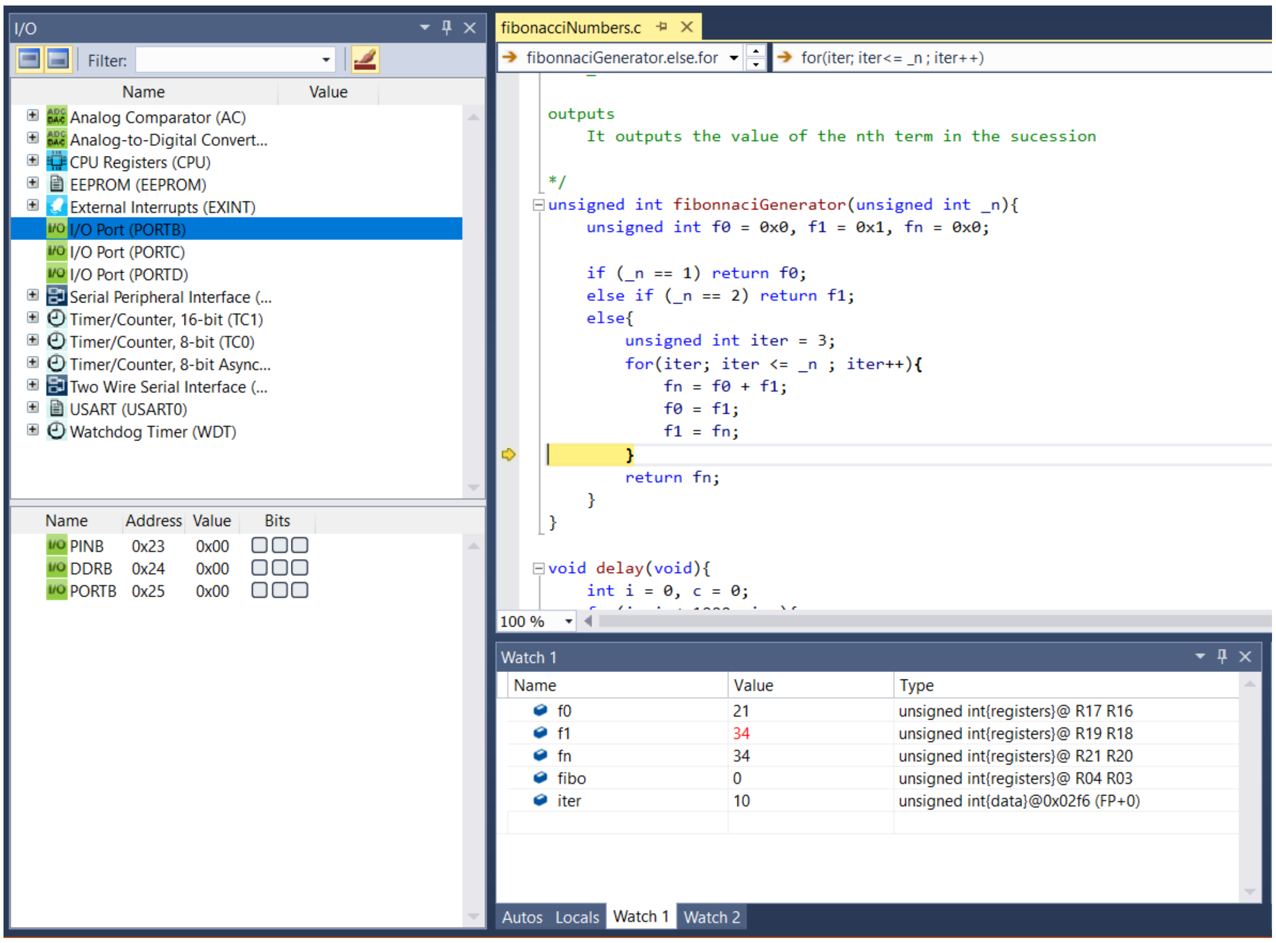
100 %

Watch 1

Name	Value	Type
f0	13	unsigned int{registers}@ R17 R16
f1	21	unsigned int{registers}@ R19 R18
fn	21	unsigned int{registers}@ R21 R20
fibo	0	unsigned int{registers}@ R04 R03
iter	10	unsigned int{data}@0x02f6 (FP+0)

Autos Locals Watch 1 Watch 2

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



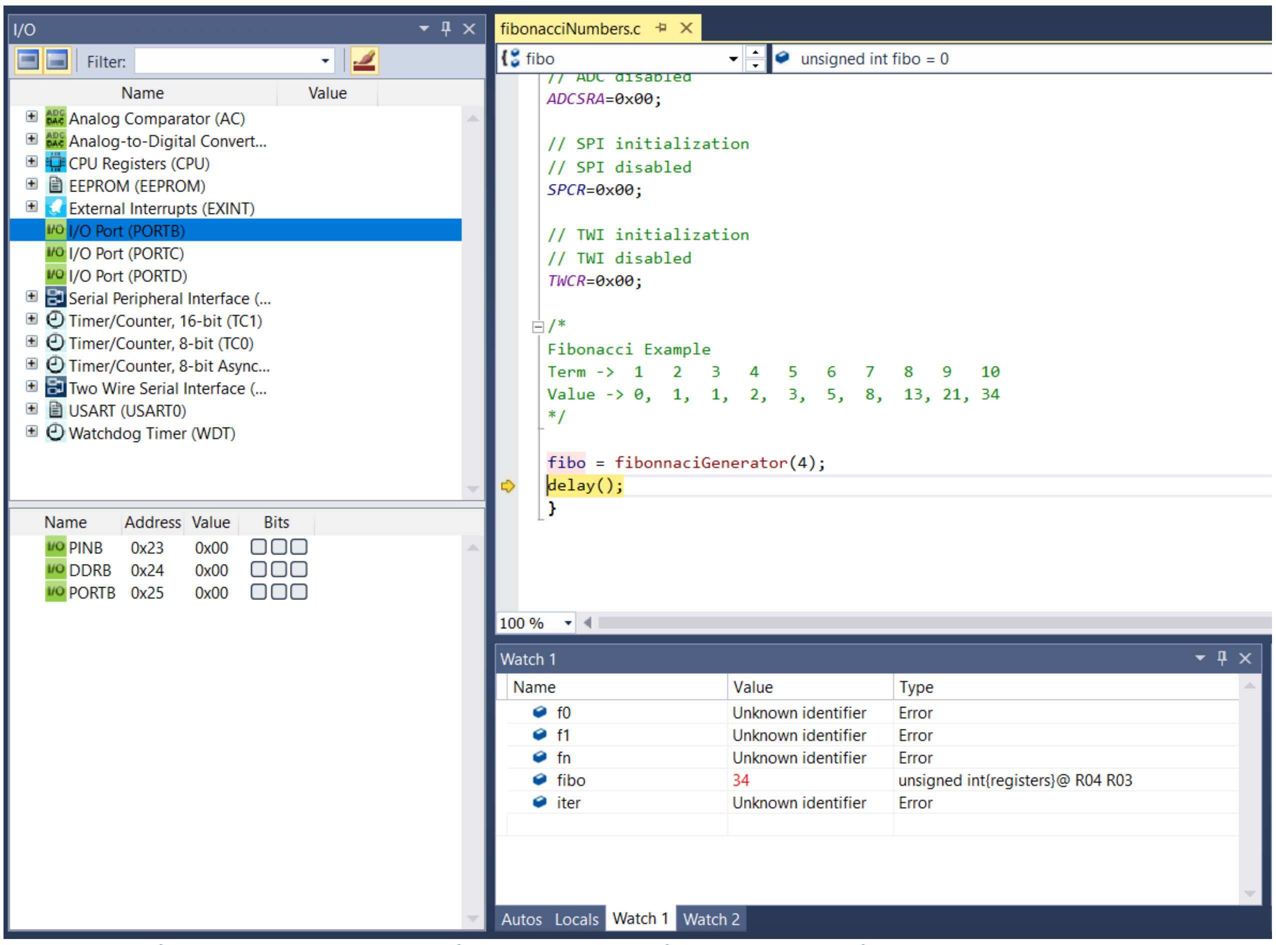
Name	Value
Analog Comparator (AC)	
Analog-to-Digital Convert...	
CPU Registers (CPU)	
EEPROM (EEPROM)	
External Interrupts (EXINT)	
I/O Port (PORTB)	
I/O Port (PORTC)	
I/O Port (PORTD)	
Serial Peripheral Interface (...	
Timer/Counter, 16-bit (TC1)	
Timer/Counter, 8-bit (TC0)	
Timer/Counter, 8-bit Async...	
Two Wire Serial Interface (...	
USART (USART0)	
Watchdog Timer (WDT)	

Name	Address	Value	Bits
PINB	0x23	0x00	☐☐☐
DDRB	0x24	0x00	☐☐☐
PORTB	0x25	0x00	☐☐☐

```
outputs
    It outputs the value of the nth term in the sucession

*/
unsigned int fibonnaciGenerator(unsigned int _n){
    unsigned int f0 = 0x0, f1 = 0x1, fn = 0x0;

    if (_n == 1) return f0;
    else if (_n == 2) return f1;
    else{
        unsigned int iter = 3;
        for(iter; iter <= _n ; iter++){
            fn = f0 + f1;
            f0 = f1;
            f1 = fn;
        }
        return fn;
    }
}

void delay(void){
    int i = 0, c = 0;
```

Name	Value	Type
f0	21	unsigned int{registers}@ R17 R16
f1	34	unsigned int{registers}@ R19 R18
fn	34	unsigned int{registers}@ R21 R20
fibo	0	unsigned int{registers}@ R04 R03
iter	10	unsigned int{data}@0x02f6 (FP+0)

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



Name	Value
Analog Comparator (AC)	
Analog-to-Digital Convert...	
CPU Registers (CPU)	
EEPROM (EEPROM)	
External Interrupts (EXINT)	
I/O Port (PORTB)	
I/O Port (PORTC)	
I/O Port (PORTD)	
Serial Peripheral Interface (...	
Timer/Counter, 16-bit (TC1)	
Timer/Counter, 8-bit (TC0)	
Timer/Counter, 8-bit Async...	
Two Wire Serial Interface (...	
USART (USART0)	
Watchdog Timer (WDT)	

Name	Address	Value	Bits
PINB	0x23	0x00	☐☐☐
DDRB	0x24	0x00	☐☐☐
PORTB	0x25	0x00	☐☐☐

```
// ADC disabled
ADCSRA=0x00;

// SPI initialization
// SPI disabled
SPCR=0x00;

// TWI initialization
// TWI disabled
TWCR=0x00;

/*
Fibonacci Example
Term ->  1   2   3   4   5   6   7   8   9   10
Value -> 0,  1,  1,  2,  3,  5,  8,  13, 21, 34
*/

fibo = fibonnaciGenerator(4);
delay();
}
```

Name	Value	Type
f0	Unknown identifier	Error
f1	Unknown identifier	Error
fn	Unknown identifier	Error
fibo	34	unsigned int{registers}@ R04 R03
iter	Unknown identifier	Error

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.

d. Demonstrate the work of your program to the instructor.

e. 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.

f. 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 2^{15}-1).

2. Control Questions

a. 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.

b. 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.

c. 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.

d. 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.

3. 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.