Silesian University of Technology

Faculty of Automatic Control,
Electronics and Computer Science

Embedded Systems Laboratory

Exercise 21

AVR Microcontrollers – part I

M.Sc. Eng. Jarosław Paduch

Exercise description:

Requirements:

Hardware: One computer PC

Software: ATMEL AVR Studio - version 7.0 and above.

During the exercise, students are to:

- Create the new GCC-executable project in ATMEL STUDIO version 7.0 or above,

- Compile and link the program,

- Observe program execution in the simulator, step by step.

- Prepare and test program execution with different data,

Report

The report should contain:

- Proper Title page

- The topic of the program

- The source code of the program with comments!!!

- Description and results of program testing

- Conclusions!!!

Report template is given on our website: http:\\www.zmitac.aei.polsl.pl

under sheet "Course MS" (microprocessor systems) / Report and project registration card templates.

ALL Program assumptions:

- The frequency of processor clock is equal to 16MHz,

- Microcontroller ATMega 2560,

- Port A has eight buttons connected to ground,

- Port B has eight diodes connected to a power supply,

-Other ports of the microcontroller are programmed as outputs,

- Somewhere in the program memory, there is a "TAB_ROM" table with unspecified length (from 0 bytes !!), and having the word "BACA" in the end

- Somewhere in the data memory, there is a "TAB_RAM" table – with length from 0 to not more than 100 bytes

-Programs should be written in “C” language,

ToDO:

a. Write a program which copies data from TAB_ROM to TAB_RAM one time only,

b. Write a program which copies data from TAB_ROM to TAB_RAM. Coping should occur just if at least any one button is pressed. If the program runs into the end of one of the tables, it should begin from the beginning of the table.

c. Write a program which only once copies data from TAB_ROM to PORT B (display data on LED's).

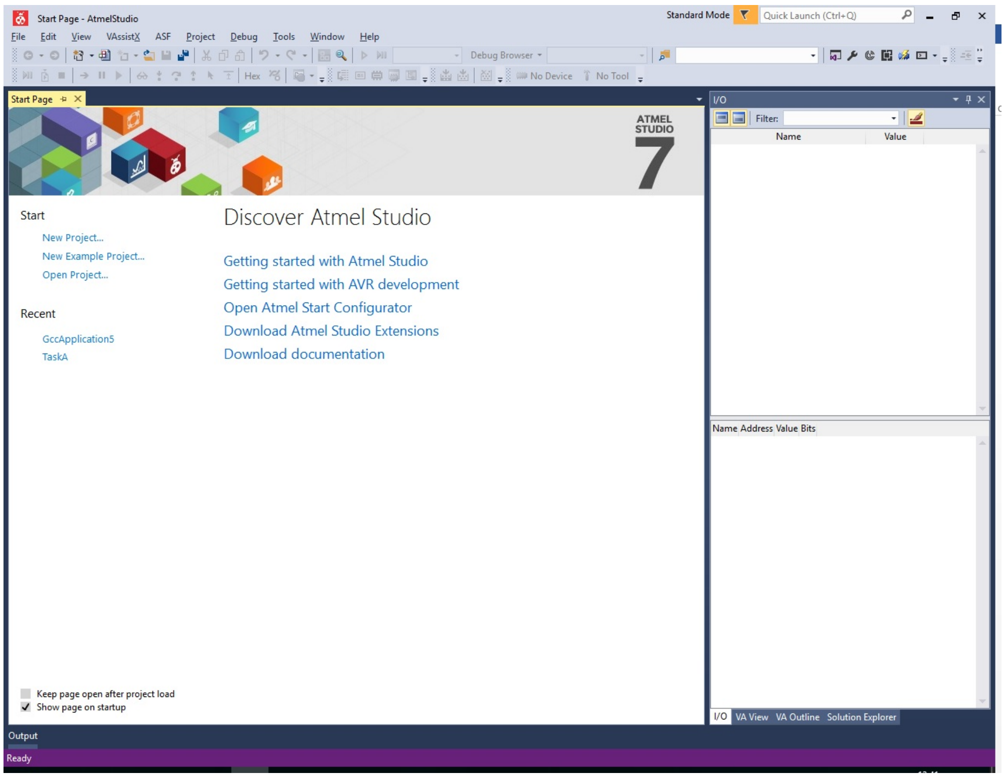
d. Write a program which copies data from TAB_ROM to PORT B (display data on LEDs. If the program runs into the end of the table, it should begin from the beginning.

e. Write a program which copies continuous data from TAB_ROM to PORT B (display data on LEDs). Displaying each byte should be delayed exactly 0.1 seconds,

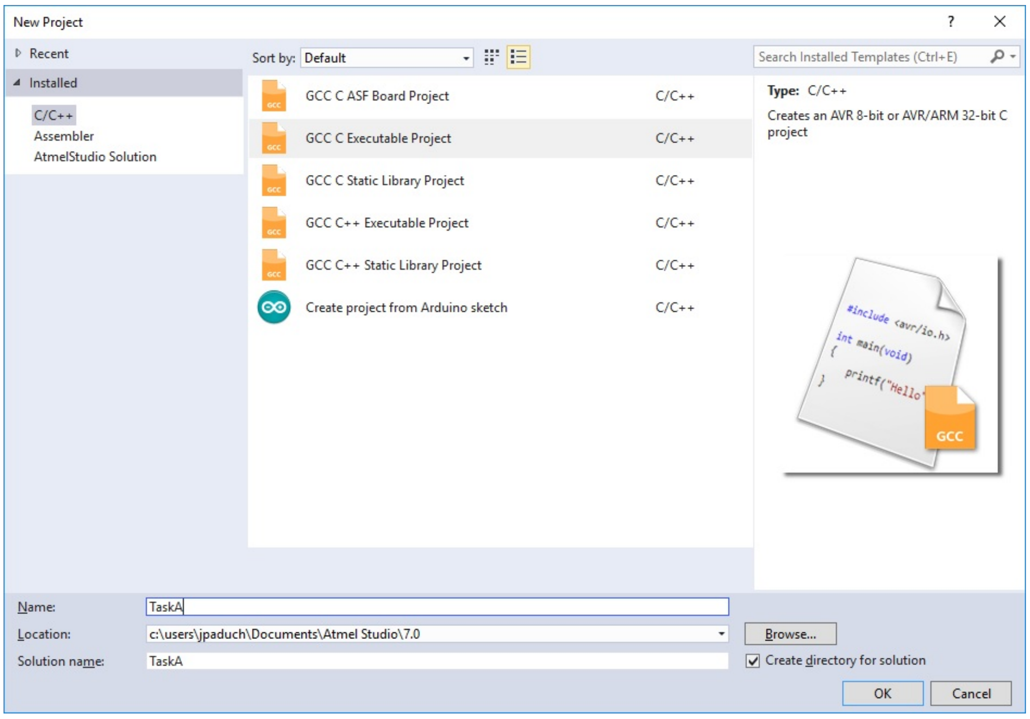
f. Write a program which copies continuous data from TAB_ROM to PORT B (display data on LEDs). Coping should occur only if at least any one button is pressed. Displaying each byte should be delayed exactly 0.2 seconds,

Create a new project:

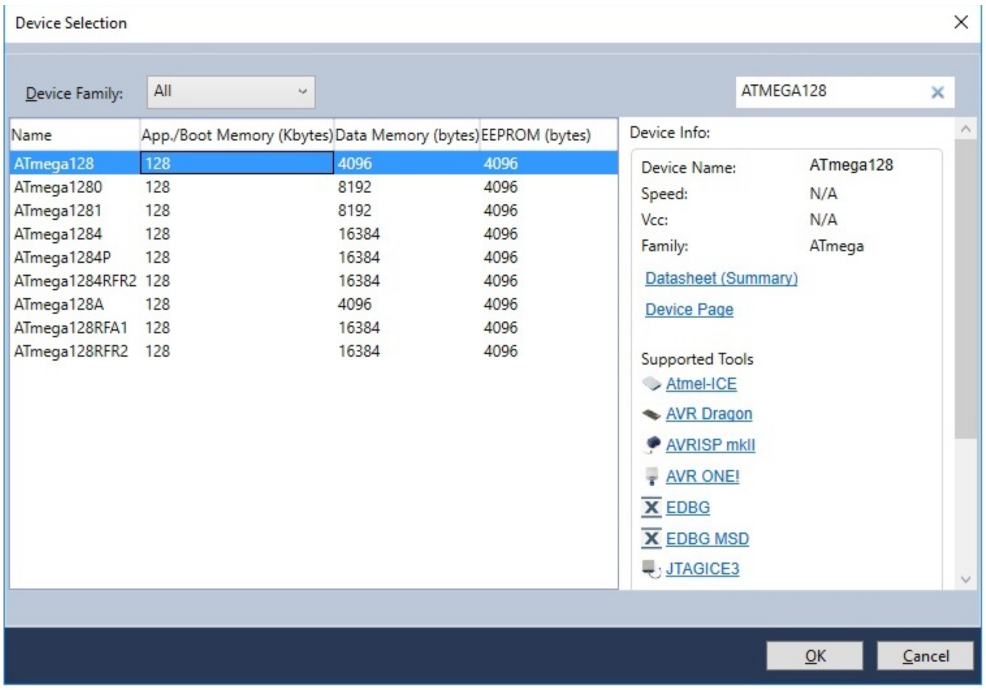
1. Run: Atmel Studio (shortcut is on the desktop)


Start Page - AtmelStudio
File Edit View VAssistX ASF Project Debug Tools Window Help
Start
New Project...
New Example Project...
Open Project...
Recent
GccApplication5
TaskA
Discover Atmel Studio
Getting started with Atmel Studio
Getting started with AVR development
Open Atmel Start Configurator
Download Atmel Studio Extensions
Download documentation
Keep page open after project load
Show page on startup
Output
Ready

2. Create a new project: choose: Gcc Executable Project, change the new project name and then press button OK.


New Project
Recent
Installed
C/C++
Assembler
AtmelStudio Solution
Sort by: Default
GCC C ASF Board Project C/C++
GCC C Executable Project C/C++
GCC C Static Library Project C/C++
GCC C++ Executable Project C/C++
GCC C++ Static Library Project C/C++
Create project from Arduino sketch C/C++
Search Installed Templates (Ctrl+E)
Type: C/C++
Creates an AVR 8-bit or AVR/ARM 32-bit C project
Name: TaskA
Location: c:\users\jpaduch\Documents\Atmel Studio\7.0
Solution name: TaskA
Browse...
Create directory for solution
OK
Cancel

3. Choose proper microcontroller type: ATMEGA128 or ATMEGA 2560 and press button OK.



Name	App./Boot Memory (Kbytes)	Data Memory (bytes)	EEPROM (bytes)
ATmega128	128	4096	4096
ATmega1280	128	8192	4096
ATmega1281	128	8192	4096
ATmega1284	128	16384	4096
ATmega1284P	128	16384	4096
ATmega1284RFR2	128	16384	4096
ATmega128A	128	4096	4096
ATmega128RFA1	128	16384	4096
ATmega128RFR2	128	16384	4096

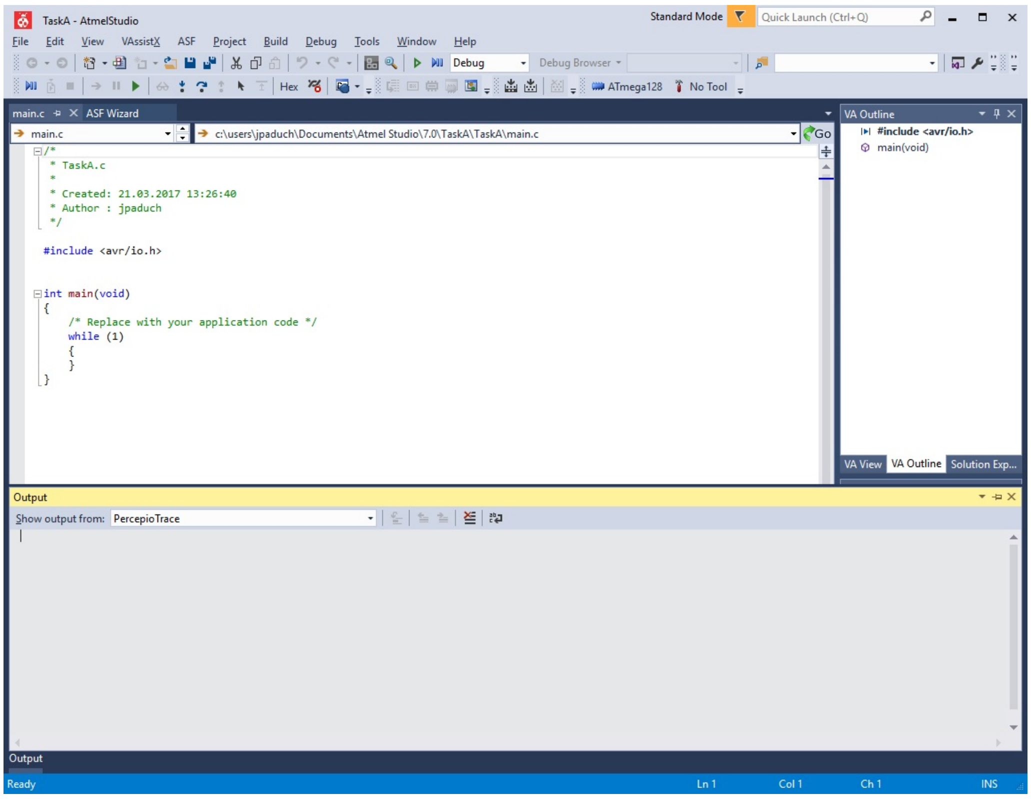
Device Info:

Device Name:	ATmega128
Speed:	N/A
Vcc:	N/A
Family:	ATmega

Datasheet (Summary)
Device Page

Supported Tools
Atmel-ICE
AVR Dragon
AVRISP mkII
AVR ONE!
EDBG
EDBG MSD
JTAGICE3

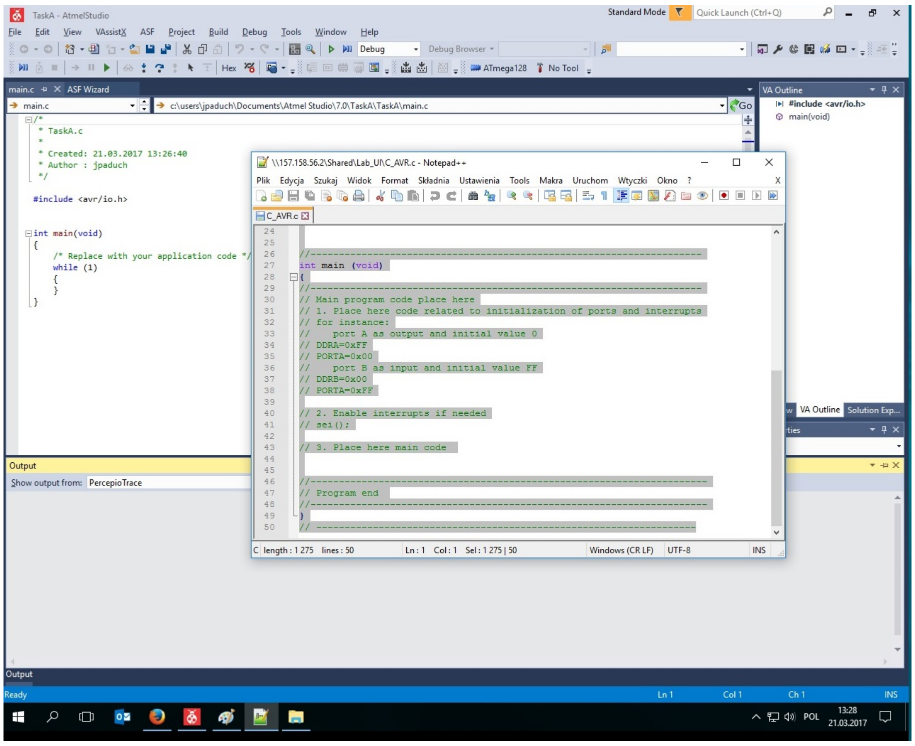
4. And finally, the GCC Project is created with a sample C program



c:\users\jpaduch\Documents\Atmel Studio\7.0\TaskA\TaskA\main.c

```c
/*
 * TaskA.c
 *
 * Created: 21.03.2017 13:26:40
 * Author : jpaduch
 */

#include <avr/io.h>


int main(void)
{
    /* Replace with your application code */
    while (1)
    {
    }
}
```

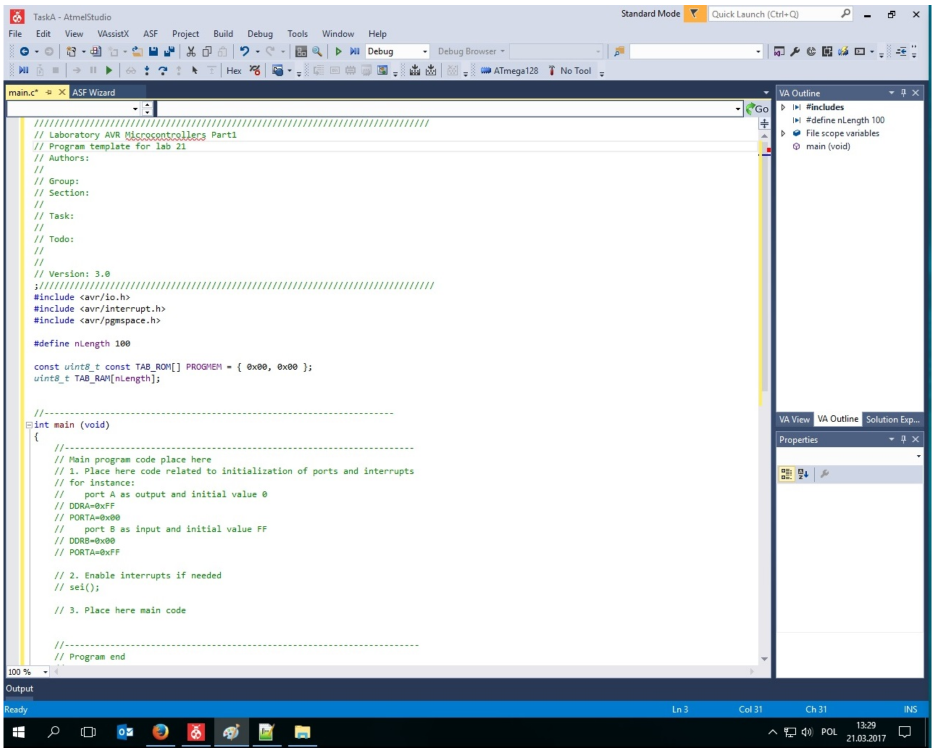
5. Please replace this code and enter the program from this instruction or copy the contents from the file:: S:\Lab_ui\C_AVR.c



```
/*
 * TaskA.c
 *
 * Created: 21.03.2017 13:26:40
 * Author : jpaduch
 */

#include <avr/io.h>


int main(void)
{
    /* Replace with your application code */
    while (1)
    {
    }
}
```

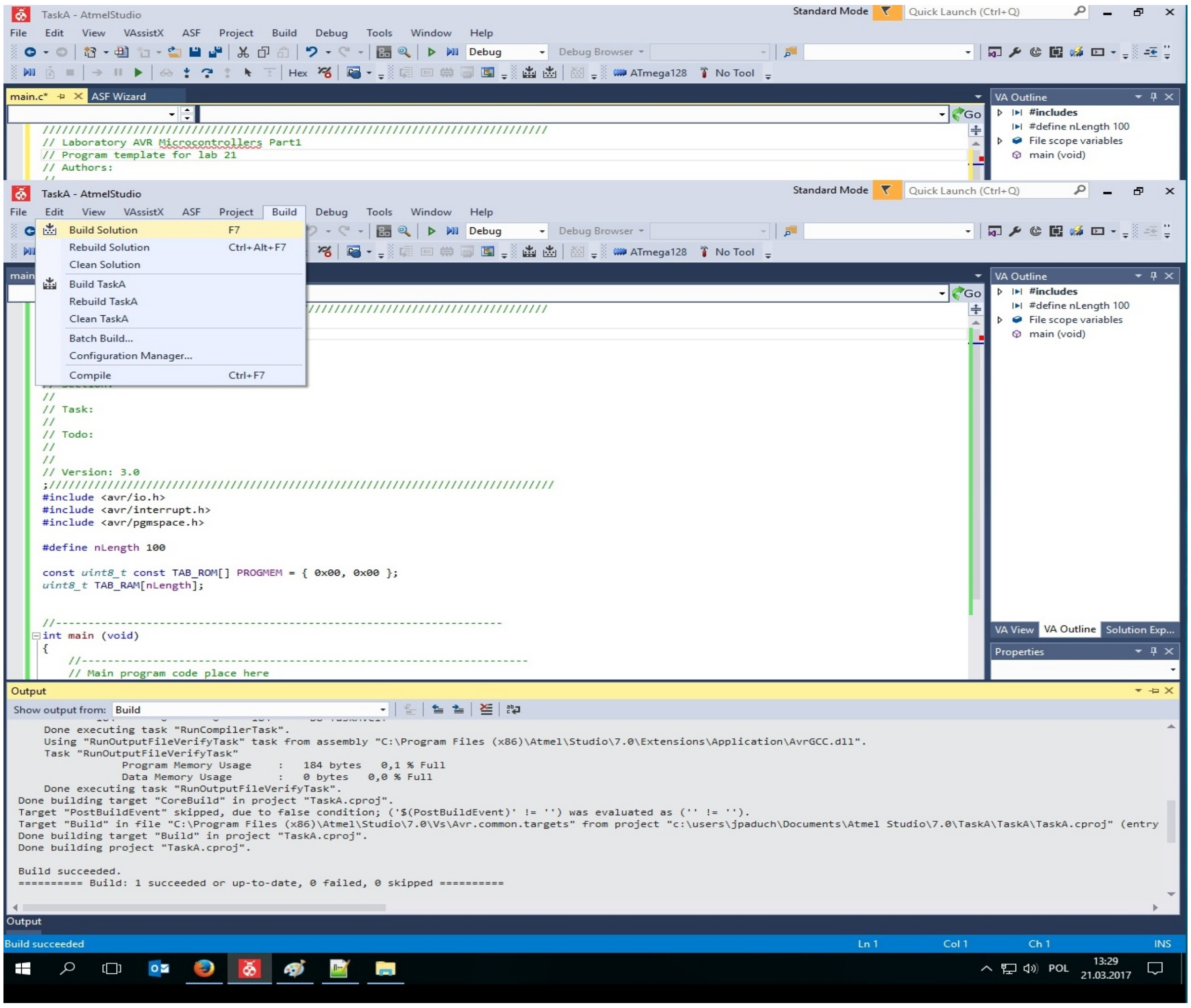
```
//-----------------------------------------------------------------------
int main (void)
{
//-----------------------------------------------------------------------
// Main program code place here
// 1. Place here code related to initialization of ports and interrupts
// for instance:
//    port A as output and initial value 0
// DDRA=0xFF
// PORTA=0x00
//    port B as input and initial value FF
// DDRB=0x00
// PORTA=0xFF

// 2. Enable interrupts if needed
// sei();

// 3. Place here main code


//------------------------------------------------------------------------
// Program end
//------------------------------------------------------------------------
}
// ---------------------------------------------------------------------
```

6. Now all is ready to start writing a program.



```
//////////////////////////////////////////////////////////////////////////////
// Laboratory AVR Microcontrollers Part1
// Program template for lab 21
// Authors:
//
// Group:
// Section:
//
// Task:
//
// Todo:
//
//
// Version: 3.0
;//////////////////////////////////////////////////////////////////////////////
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/pgmspace.h>

#define nLength 100

const uint8_t const TAB_ROM[] PROGMEM = { 0x00, 0x00 };
uint8_t TAB_RAM[nLength];


//-----------------------------------------------------------------------
int main (void)
{
    //-----------------------------------------------------------------------
    // Main program code place here
    // 1. Place here code related to initialization of ports and interrupts
    // for instance:
    //    port A as output and initial value 0
    // DDRA=0xFF
    // PORTA=0x00
    //    port B as input and initial value FF
    // DDRB=0x00
    // PORTA=0xFF

    // 2. Enable interrupts if needed
    // sei();

    // 3. Place here main code


    //------------------------------------------------------------------------
    // Program end
```

7. Building solutions

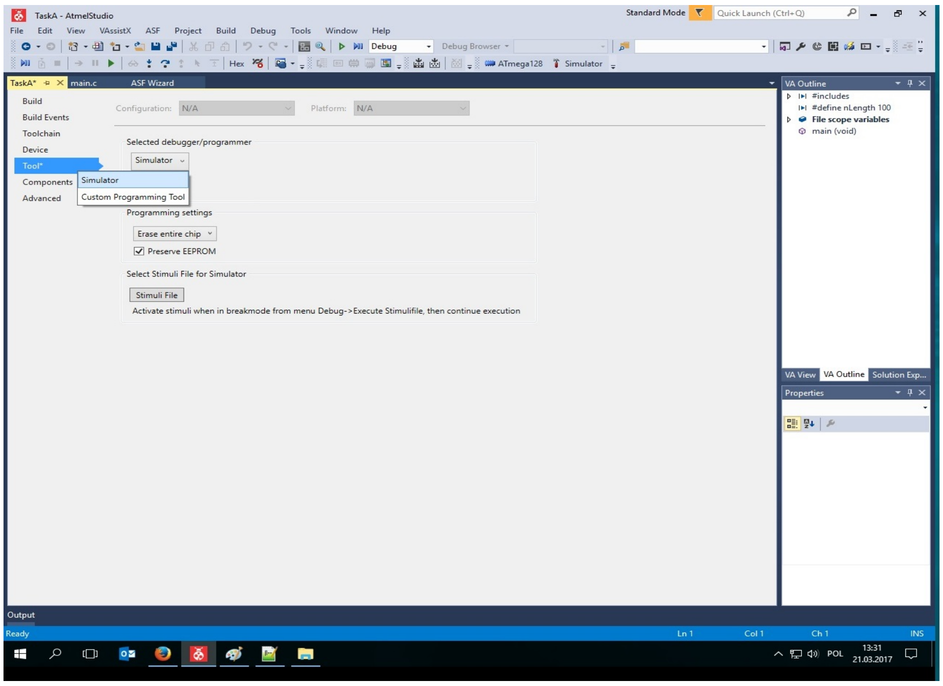


```
//////////////////////////////////////////////////////////////////////////////
// Laboratory AVR Microcontrollers Part1
// Program template for lab 21
// Authors:
```

```
//
// Task:
//
// Todo:
//
//
// Version: 3.0
;//////////////////////////////////////////////////////////////////////////////
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/pgmspace.h>

#define nLength 100

const uint8_t const TAB_ROM[] PROGMEM = { 0x00, 0x00 };
uint8_t TAB_RAM[nLength];


//---------------------------------------------------------------------
int main (void)
{
    //---------------------------------------------------------------------
    // Main program code place here
```

```
    Done executing task "RunCompilerTask".
    Using "RunOutputFileVerifyTask" task from assembly "C:\Program Files (x86)\Atmel\Studio\7.0\Extensions\Application\AvrGCC.dll".
    Task "RunOutputFileVerifyTask"
                Program Memory Usage    :   184 bytes   0,1 % Full
                Data Memory Usage       :   0 bytes   0,0 % Full
    Done executing task "RunOutputFileVerifyTask".
Done building target "CoreBuild" in project "TaskA.cproj".
Target "PostBuildEvent" skipped, due to false condition; ('$(PostBuildEvent)' != '') was evaluated as ('' != '').
Target "Build" in file "C:\Program Files (x86)\Atmel\Studio\7.0\Vs\Avr.common.targets" from project "c:\users\jpaduch\Documents\Atmel Studio\7.0\TaskA\TaskA\TaskA.cproj" (entry
Done building target "Build" in project "TaskA.cproj".
Done building project "TaskA.cproj".

Build succeeded.
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
```

8. Running



Configuration: N/A Platform: N/A

Selected debugger/programmer: Simulator

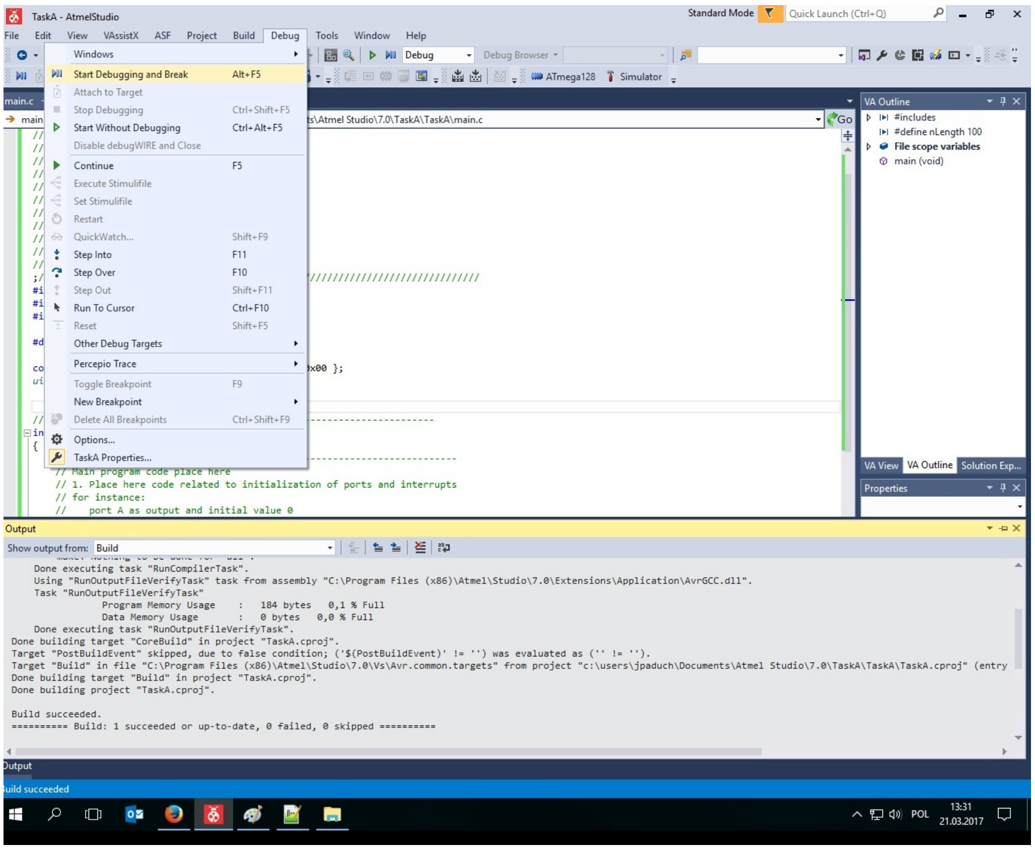
Programming settings: Erase entire chip

☑ Preserve EEPROM

Select Stimuli File for Simulator

Activate stimuli when in breakmode from menu Debug->Execute Stimulifile, then continue execution

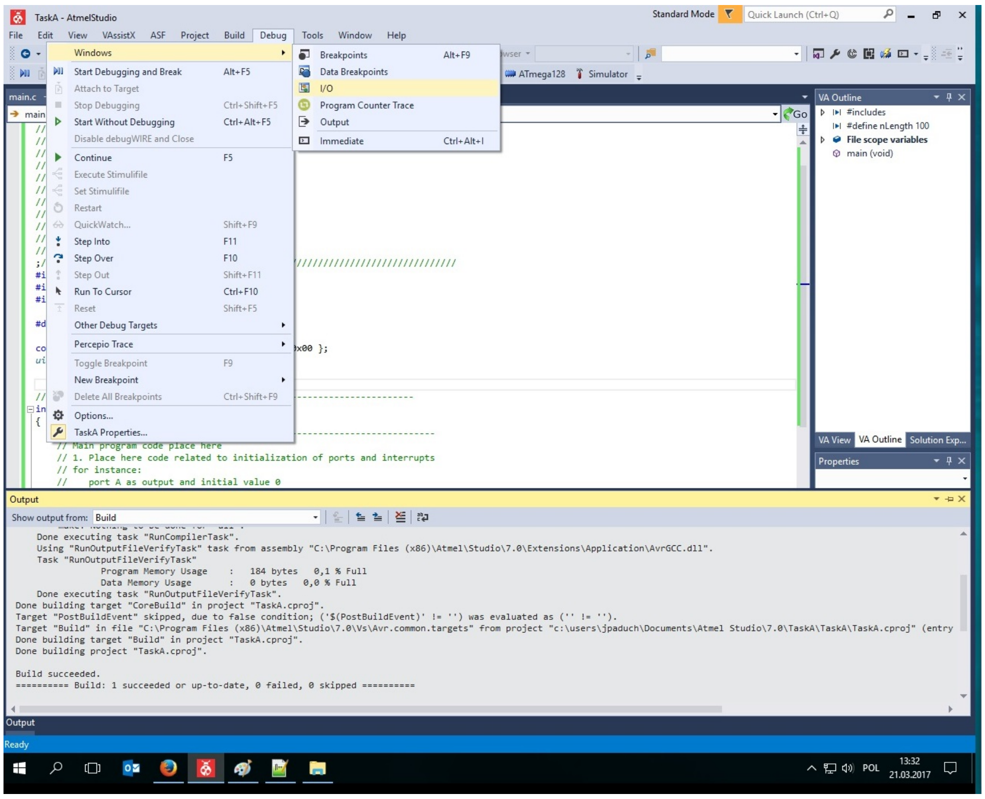
9. Debugging



```
// Main program code place here
// 1. Place here code related to initialization of ports and interrupts
// for instance:
//    port A as output and initial value 0
```

```
    Done executing task "RunCompilerTask".
    Using "RunOutputFileVerifyTask" task from assembly "C:\Program Files (x86)\Atmel\Studio\7.0\Extensions\Application\AvrGCC.dll".
    Task "RunOutputFileVerifyTask"
            Program Memory Usage    :   184 bytes   0,1 % Full
            Data Memory Usage       :   0 bytes   0,0 % Full
    Done executing task "RunOutputFileVerifyTask".
Done building target "CoreBuild" in project "TaskA.cproj".
Target "PostBuildEvent" skipped, due to false condition; ('$(PostBuildEvent)' != '') was evaluated as ('' != '').
Target "Build" in file "C:\Program Files (x86)\Atmel\Studio\7.0\Vs\Avr.common.targets" from project "c:\users\jpaduch\Documents\Atmel Studio\7.0\TaskA\TaskA\TaskA.cproj" (entry
Done building target "Build" in project "TaskA.cproj".
Done building project "TaskA.cproj".

Build succeeded.
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
```

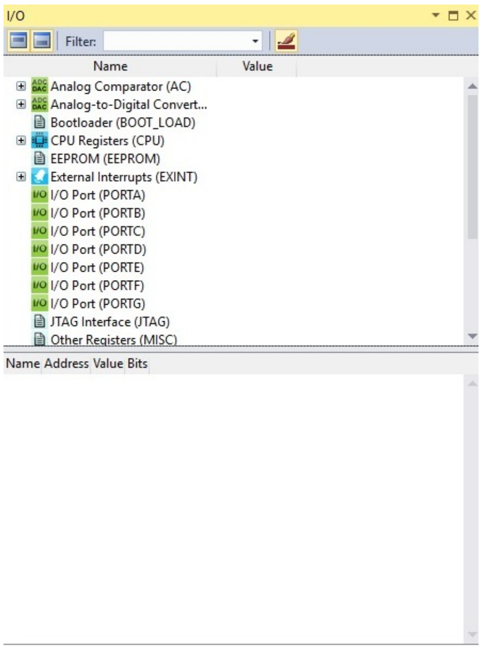
10. Opening additional windows

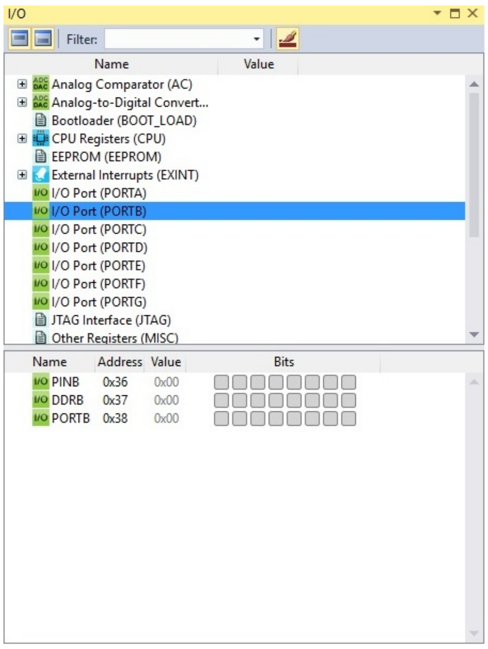


```
// Main program code place here
// 1. Place here code related to initialization of ports and interrupts
// for instance:
//    port A as output and initial value 0
```

```
    Done executing task "RunCompilerTask".
    Using "RunOutputFileVerifyTask" task from assembly "C:\Program Files (x86)\Atmel\Studio\7.0\Extensions\Application\AvrGCC.dll".
    Task "RunOutputFileVerifyTask"
                Program Memory Usage    :   184 bytes   0,1 % Full
                Data Memory Usage       :   0 bytes   0,0 % Full
    Done executing task "RunOutputFileVerifyTask".
Done building target "CoreBuild" in project "TaskA.cproj".
Target "PostBuildEvent" skipped, due to false condition; ('$(PostBuildEvent)' != '') was evaluated as ('' != '').
Target "Build" in file "C:\Program Files (x86)\Atmel\Studio\7.0\Vs\Avr.common.targets" from project "c:\users\jpaduch\Documents\Atmel Studio\7.0\TaskA\TaskA\TaskA.cproj" (entry
Done building target "Build" in project "TaskA.cproj".
Done building project "TaskA.cproj".

Build succeeded.
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
```

Just open window with I/O view





Name	Address	Value	Bits
PINB	0x36	0x00	☐☐☐☐☐☐☐☐
DDRB	0x37	0x00	☐☐☐☐☐☐☐☐
PORTB	0x38	0x00	☐☐☐☐☐☐☐☐

Bibliography

1. Jarosław Doliński - "Mikrokontrolery AVR w praktyce"
2. Andrzej Pawluczuk - "Sztuka programowania mikrokontrolerów. AVR - podstawy"
3. Piotr Górecki - "Mikrokontrolery dla początkujących"
4. http://www.itee.uq.edu.au/~cse/_atmel/AVR_Studio_Tutorial/
5. http://winavr.scienceprog.com/avr-gcc-tutorial/
6. http://imakeprojects.com/Projects/avr-tutorial/
7. http://www.atmel.com/products/avr/
8. http://www.avrfreaks.net/
9. http://pl.wikipedia.org/wiki/Atmel_AVR
10. http://www.elportal.pl/ea/asm_avr.html
11. Jakub Jankowski, Marcin Kania, Mariusz Macheta, Łukasz Strzelecki – Opracowanie na temat mikrokontrolery AVR”

1.C-Code template

```
////////////////////////////////////////////////////////////////////////////////
// Laboratory AVR Microcontrollers Part1
// Program template for lab 7
// Please fill in this information before starting coding
// Authors:
//
// Group:
// Section:
//
// Task:
//
// Todo:
//
//
// Version: 5.0
////////////////////////////////////////////////////////////////////////////////
#include <avr/io.h>
// please look into header pgmspace.h functions to properly read from ROM
#include <avr/pgmspace.h>
// please change this to proper value given by the teacher
#define F_CPU 16000000L
// please correct this line according to the guidelines given by the teacher
#define <delay.h>
#define nLength 100
uint8_t TAB_RAM[nLength];
// please correct end of this table according to the guidelines given by the teacher
uint8_t const TAB_ROM[] PROGMEM = { 0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09,
0x0A, 0x0B, 0x0C, 0x0D, 0x0E, 0x1F,
       0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17, 0x18, 0x19, 0x1A, 0x1B, 0x1C, 0x1D, 0x1F,
0x1F,
       0x20, 0x21, 0x22, 0x23, 0x24, 0x25, 0x26, 0x27, 0x28, 0x29, 0x2A, 0x2B, 0x2C, 0x2D, 0x2E,
0x2F,
       0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x38, 0x39, 0x3A, 0x3B, 0x3C, 0x3D, 0x3E,
0x3F,
       0x40, 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48, 0x49, 0x4A, 0x4B, 0x4C, 0x4D, 0x4E,
0x4F,
       0x50, 0x51, 0x52, 0x53, 0x54, 0x55, 0x56, 0x57, 0x58, 0x59, 0x5A, 0x5B, 0x5C, 0x5D, 0x5E,
0x5F,
       0x60, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x67, 0x68, 0x69, 0x6A, 0x6B, 0x6C, 0x6D, 0x6E,
0x6F,
       0x70, 0x71, 0x72, 0x73, 0x74, 0x75, 0x76, 0x77, 0x78, 0x79, 0x7A, 0x7B, 0x7C, 0x7D, 0x7E,
0x7F,
       0x80, 0x81, 0x82, 0x83, 0x84, 0x85, 0x86, 0x87, 0x88, 0x89, 0x8A, 0x8B, 0x8C, 0x8D, 0x8E,
0x8F,
       0x90, 0x91, 0x92, 0x93, 0x94, 0x95, 0x96, 0x97, 0x98, 0x99, 0x9A, 0x9B, 0x9C, 0x9D, 0x9E,
0x9F,
       0xa0, 0xa1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xaA, 0xaB, 0xaC, 0xaD, 0xAE,
0xAF,
       0xB0, 0xB1, 0xB2, 0xB3, 0xB4, 0xB5, 0xB6, 0xB7, 0xB8, 0xB9, 0xBA, 0xBB, 0xBC, 0xBD, 0xBE,
0xBF,
       0xC0, 0xC1, 0xC2, 0xC3, 0xC4, 0xC5, 0xC6, 0xC7, 0xC8, 0xC9, 0xCA, 0xCB, 0xCC, 0xCD, 0xCE,
0xCF,
       0xE0, 0xE1, 0xE2, 0xE3, 0xE4, 0xE5, 0xE6, 0xE7, 0xE8, 0xE9, 0xEA, 0xEB, 0xEC, 0xED, 0xEE,
0xEF,
       0xFF, 0xFE, 0xFD, 0xFC, 0xFB, 0xFA, 0xF9, 0xF8, 0xF7, 0xF6, 0xF5, 0xF4, 0xF3, 0xF2, 0xF1,
0xF0,
0xFF, 0xFF};

//------------------------------------------------------------------------
int main (void)
{
       //------------------------------------------------------------------------
       // Main program code place here
       // 1. Place here code related to initialization of ports and interrupts
```

```
        // for instance:
        //    port A as input and switching Pull-up resistors on
        // DDRA=0x00
        // PORTA=0xFF
        //    port C as output and initial value FF
        // DDRB=0xFF
        // PORTA=0xFF
        //
        // Writing a logic one to PINxn toggles the value of PORTxn, independent on the value of DRxn
        // Note that the SBI instruction can be used to toggle one single bit in a port.



        // 2. Enable interrupts if needed
        // sei();



        // 3. Place here main code





        //-------------------------------------------------------------------
        // Program end
        //-------------------------------------------------------------------
}
// ----------------------------------------------------------------
```