Creating the first project in

mikroPascal PRO for AVR®

MikroElektronika
DEVELOPMENT TOOLS | COMPILERS | BOOKS

TO OUR VALUED CUSTOMERS

I want to express my thanks to you for being interested in our products and for having confidence in MikroElektronika.

The primary aim of our company is to design and produce high quality electronic products and to constantly improve the performance thereof in order to better suit your needs.

Nebojsa Matic
General Manager

Table of Contents

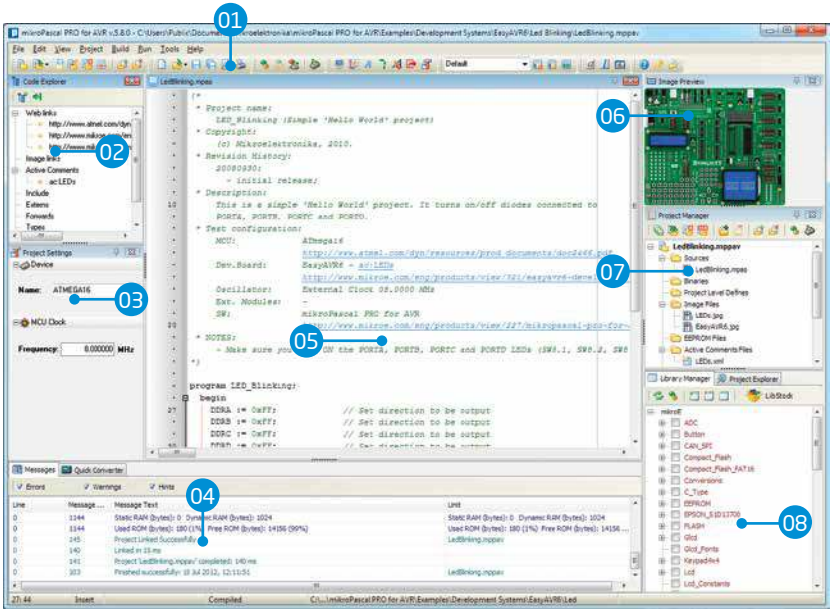
1. Introduction to mikroPascal PRO for AVR®

mikroPascal PRO for AVR® organizes applications into projects consisting of a single project file (file with the **.mppav** extension) and one or more source files (files with the .mbas extension). The mikroPascal PRO for AVR® compiler allows you to manage several projects at a time. Source files can be compiled only if they are part of the project.

A project file contains:

- Project name and optional description;
- Target device in use;
- Device clock;
- List of the project source files;
- Binary files (*.mcl); and
- Other files.

In this reference guide, we will create a new project, write code, compile it and test the results. The purpose of this project is to make microcontroller PORTA LEDs blink, which will be easy to test.



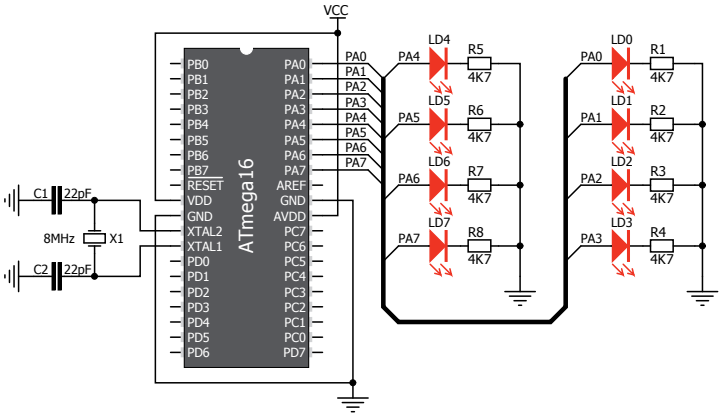
01 Main Toolbar
02 Code Explorer
03 Project Settings
04 Messages
05 Code Editor
06 Image Preview
07 Project Manger
08 Library Manager

2. Hardware Connection

Let's make a simple "Hello world" example for the selected microcontroller. First thing embedded programmers usually write is a simple **LED blinking** program. So, let's do that in a few simple lines of Pascal code.

LED blinking is just turning ON and OFF LEDs that are connected to desired PORT pins. In order to see the example in action, it is necessary to connect the target microcontroller according to schematics shown on **Figure 2-1**. In the project we are about to write, we will use only **PORTA**, so you should connect the LEDs to PORTA only. Eight LEDs are more then enough for demonstration.

Figure 2-1:
Hardware connection schematics



Prior to creating a new project, it is necessary to do the following:

Step 1: Install the compiler

Install the mikroPascal PRO for AVR® compiler from the **Product DVD** or download it from the MikroElektronika website:

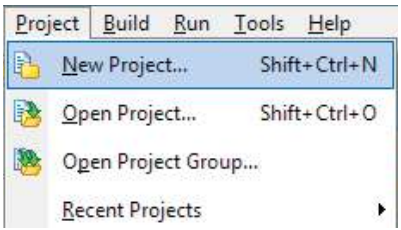
http://www.mikroe.com/eng/products/view/227/mikropascal-pro-for-avr/

Step 2: Start up the compiler

Double click on the compiler icon in the Start menu, or on your desktop to Start up the mikroPascal PRO for AVR® compiler. The mikroPascal PRO for AVR® IDE (Integrated Development Environment) will appear on the screen. Now you are ready to start creating a new project.

3. Creating a New Project

The process of creating a new project is very simple. Select the **New Project** option from the **Project menu** as shown below. The **New Project Wizard** window appears. It can also be opened by clicking the **New Project icon** from the **Project toolbar**.



The **New Project Wizard** (**Figure 3-1**) will guide you through the process of creating a new project. The introductory window of this application contains a list of actions to be performed when creating a new project.

Figure 3-1: Introductory window of the New Project Wizard

01 Click **Next**.

Step 1 - Project Settings

First thing we have to do is to specify the general project information. This is done by selecting the target microcontroller, it's operating clock frequency, and of course - naming our project. This is an important step, because compiler will adjust the internal settings based on this information. Default configuration is already suggested to us at the begining. We have to change the device name to **ATMEGA16** as it is our microcontroller of choice for this project.

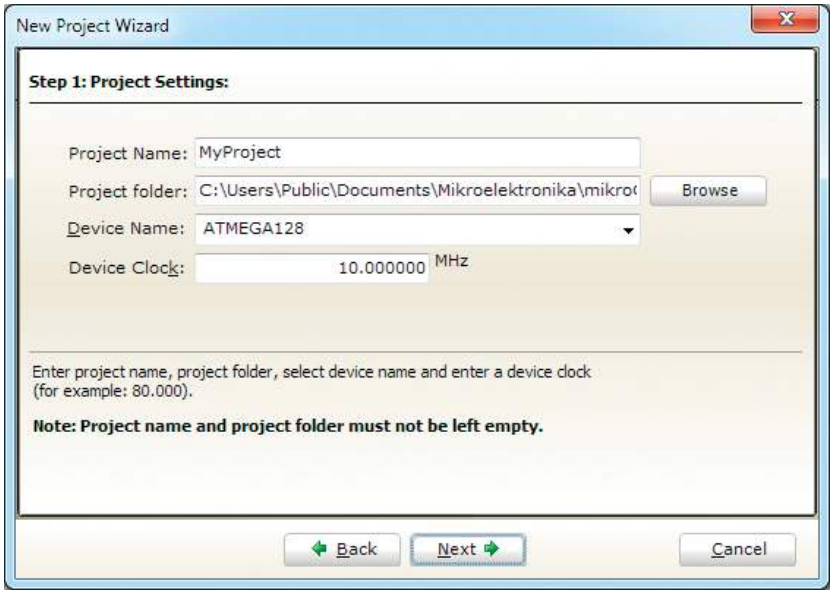


Figure 3-2: You can specify project name, path, device and clock in the first step

Step 1 - Project Settings

If you do not want to use the suggested path for storing your new project, you can **change the destination folder**. In order to do that, follow a simple procedure:

01 Click the **Browse** button of the Project Settings window to open the **Browse for Folder** dialog.

02 Select the desired folder to be the destination path for storing your new project files.

03 Click the **OK** button to confirm your selection and apply the new path.

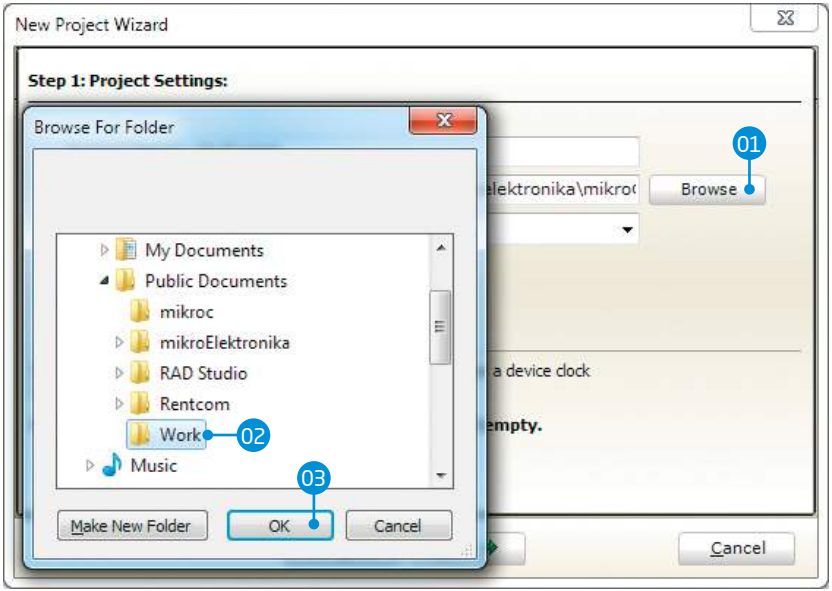


Figure 3-3: Change the destination folder using Browse For Folder dialog

Step 1 - Project Settings

Once we have selected the destination project folder, let's do the rest of the project settings:

01 Enter the name of your project. Since we are going to blink some LEDs, it's appropriate to call the project **"LedBlinking"**

02 For this demonstration, we will use the external crystal **8MHz clock**. Clock speed depends on your target hardware, but however you configure your hardware, make sure to specify the exact clock (**Fosc**) that the microcontroller is operating at.

03 Click the **OK** button to proceed.

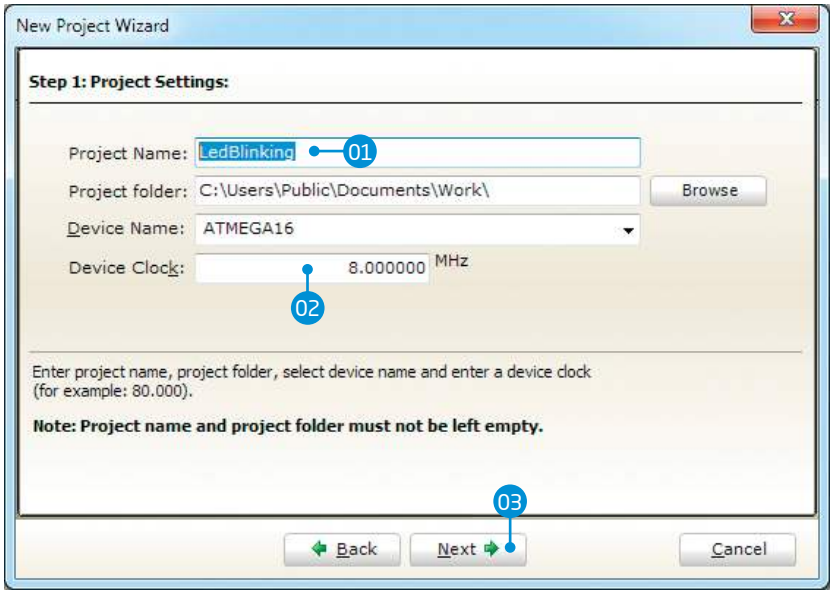


Figure 3-4: Enter project name and change device clock speed if necessary

Step 2 - Add files

This step allows you to include additional files that you need in your project: some headers or source files that you already wrote, and that you might need in further development. Since we are building a simple application, we won't be adding any files at this moment.

01 Click **Next**.

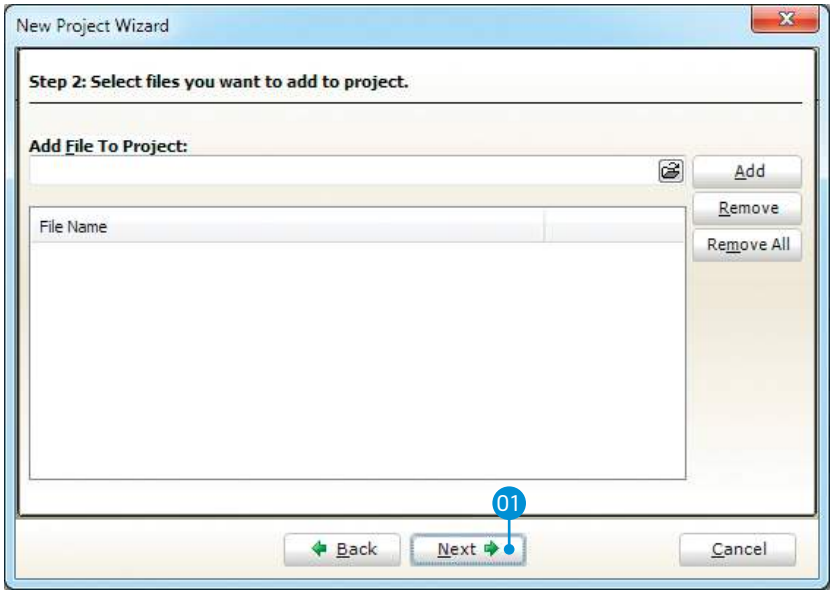


Figure 3-5: Add existing headers, sources or other files if necessary

Step 3 - Include Libraries

Following step allows you to quickly set whether you want to include all libraries in your project, or not. Even if all libraries are included, they will not consume any memory unless they are explicitely used from within your code. The main advantage of including all libraries is that you will have over **500 functions** available for use in your code right away, and visible from **Code Assistant [CTRL+Space]**. We will leave this in default configuration:

01 Make sure to leave **"Include All"** selected.

02 Click **Next.**

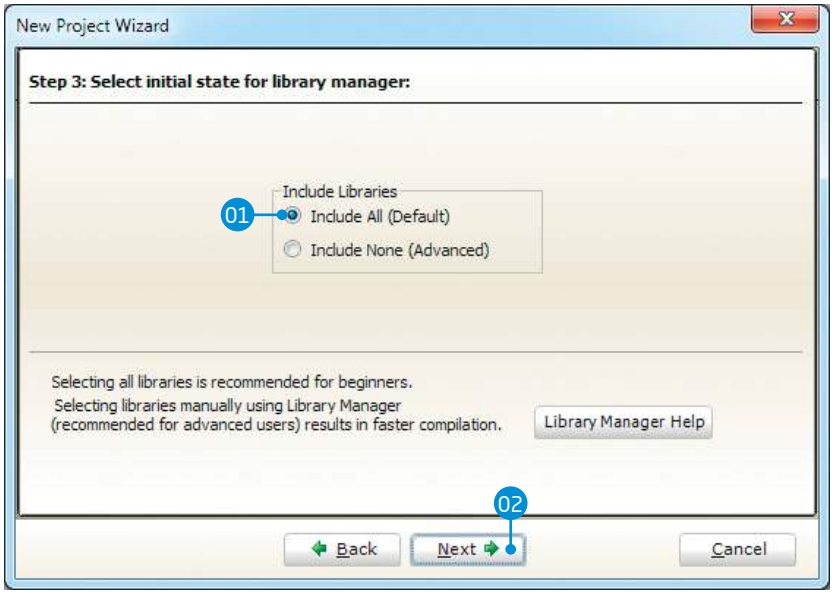


Figure 3-6: Include all libraries in the project, which is a default configuration.

Step 4 - Finishing

After all configuration is done, final step allows you to do just a bit more.

01 There is a check-box called **"Open Edit Project window to set Configuration bits"** at the final step. **Edit Project** is a specialized window which allows you to do all the necessary oscillator settings, as well as to set desired fuse bits. We made sure that everything is described in plain English, so you will be able to do the settings without having to open the datasheet. Anyway, since we are only building a simple application, we will leave it at default configuration (external crystal oscillator). **Therefore, leave the checkbox unchecked.**

02 Click **Finish**.

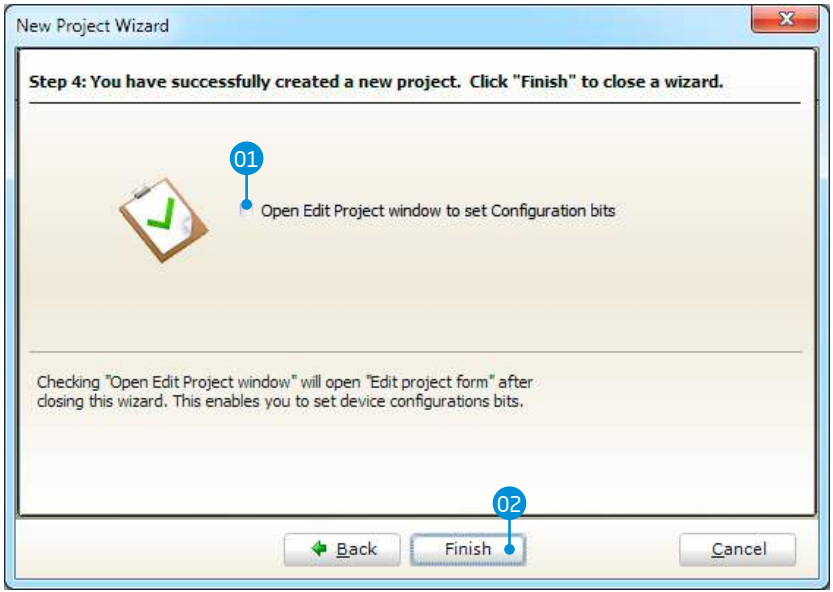


Figure 3-7: Choose whether to open Edit Project window after dialog closes.

Blank new project created

New project is finally created. A new source file called **"LedBlinking.mpas"** is created and it contains the main `begin...end` block, which will hold the program. You may notice that project is configured according to the settings done in the **New Project Wizard**.

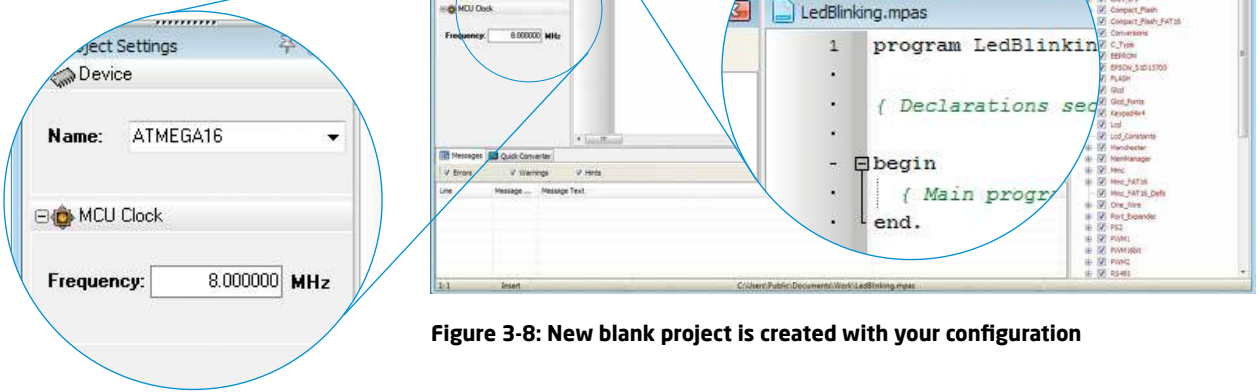


Figure 3-8: New blank project is created with your configuration

4. Code Example

Time has come to do some coding. First thing we need to do is to initialize PORTA to act as digital output. DDRA data direction register, associated with PORTA, is used to set whether each pin acts as input or output.

```
// set PORTA to be digital output
DDRA := 0;
```

We can now initialize PORTA it with logic zeros on every pin:

```
// Turn OFF LEDs on PORTA
PORTA := 0;
```

Finally, in a **while** loop we will invert all bits in PORTA in every iteration, and put a 1000 ms delay, so the blinking is not too fast (see **Figure 4-1**).

```
// Toggle LEDs on PORTA
PORTA := not PORTA;

// Delay 1000 ms
Delay_ms(1000);
```

LedBlinking.mpas - source code

```
1  program LedBlinking;
2  begin
3    // set PORTA to be digital output
4    DDRA := 0;
5
6    // Turn OFF LEDs on PORTA
7    PORTA := 0;
8
9    while TRUE do
10   begin
11     // Toggle LEDs on PORTA
12     PORTA := not PORTA;
13
14     // Delay 1000 ms
15     Delay_ms(1000);
16   end;
17 end.
```

Figure 4-1: Complete source code of the PORTA LED blinking

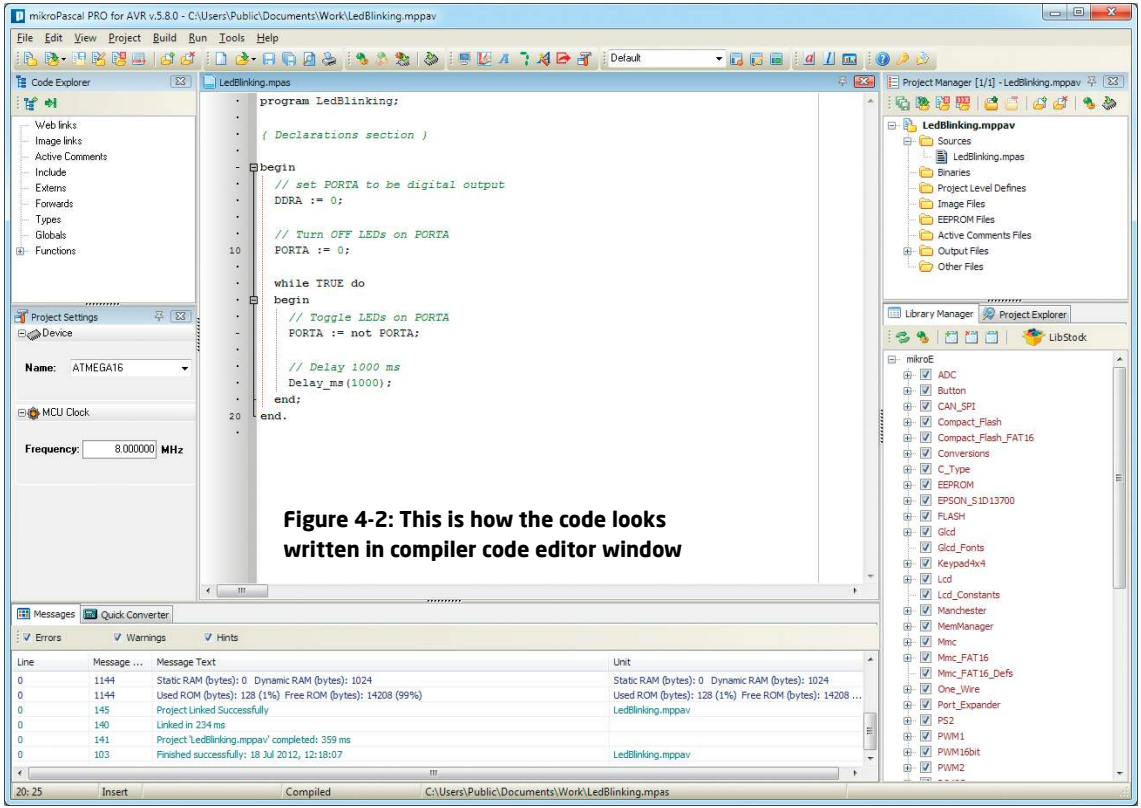
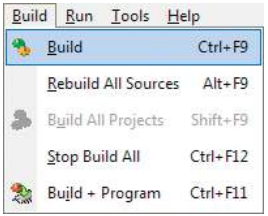
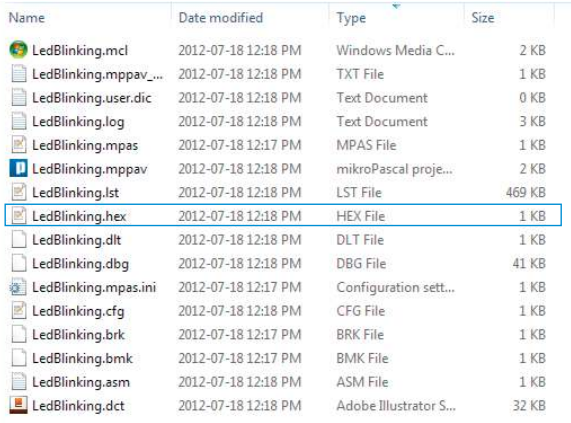


Figure 4-2: This is how the code looks written in compiler code editor window

5. Building the Source

When we are done writing our first LedBlinking code, we can now build the project and create a **.HEX** file which can be loaded into our target microcontroller, so we can test the program on real hardware. "Building" includes compilation, linking and optimization which are done automatically. Build your code by clicking on the  icon in the main toolbar, or simply go to **Build menu** and click **Build [CTRL+F9]**. Message window will report the details of the building process (**Figure 5-2**). Compiler automatically creates necessary output files. **LedBlinking.hex** (**Figure 5-1**) is among them.





Name	Date modified	Type	Size
LedBlinking.mcl	2012-07-18 12:18 PM	Windows Media C...	2 KB
LedBlinking.mppav_...	2012-07-18 12:18 PM	TXT File	1 KB
LedBlinking.user.dic	2012-07-18 12:18 PM	Text Document	0 KB
LedBlinking.log	2012-07-18 12:18 PM	Text Document	3 KB
LedBlinking.mpas	2012-07-18 12:17 PM	MPAS File	1 KB
LedBlinking.mppav	2012-07-18 12:18 PM	mikroPascal proje...	2 KB
LedBlinking.lst	2012-07-18 12:18 PM	LST File	469 KB
LedBlinking.hex	2012-07-18 12:18 PM	HEX File	1 KB
LedBlinking.dlt	2012-07-18 12:18 PM	DLT File	1 KB
LedBlinking.dbg	2012-07-18 12:18 PM	DBG File	41 KB
LedBlinking.mpas.ini	2012-07-18 12:17 PM	Configuration sett...	1 KB
LedBlinking.cfg	2012-07-18 12:18 PM	CFG File	1 KB
LedBlinking.brk	2012-07-18 12:17 PM	BRK File	1 KB
LedBlinking.bmk	2012-07-18 12:17 PM	BMK File	1 KB
LedBlinking.asm	2012-07-18 12:18 PM	ASM File	1 KB
LedBlinking.dct	2012-07-18 12:18 PM	Adobe Illustrator S...	32 KB

Figure 5-1: Listing of project files after building is done

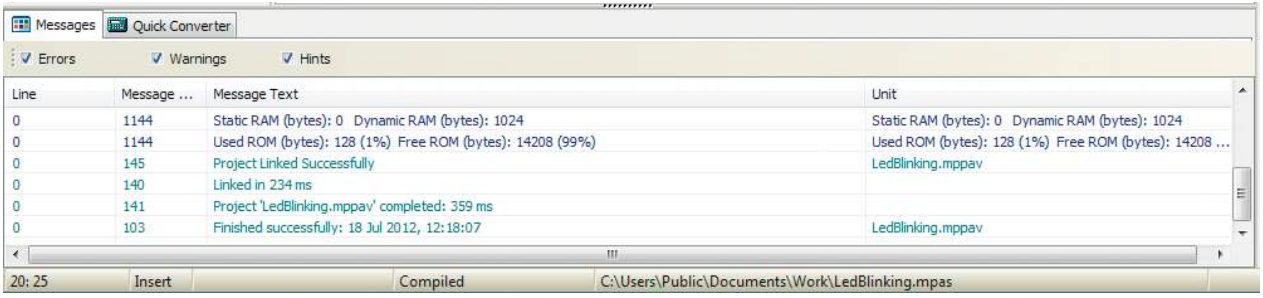


Figure 5-2: After the successful compilation and linking, the message window should look something like this

6. Changing Project Settings

If you need to change the target microcontroller or clock speed, you don't have to go through the new project wizard all over again. This can be done quickly in the **Edit Project** window. You can open it using **Project->Edit Project [CTRL+SHIFT+E]** menu option.

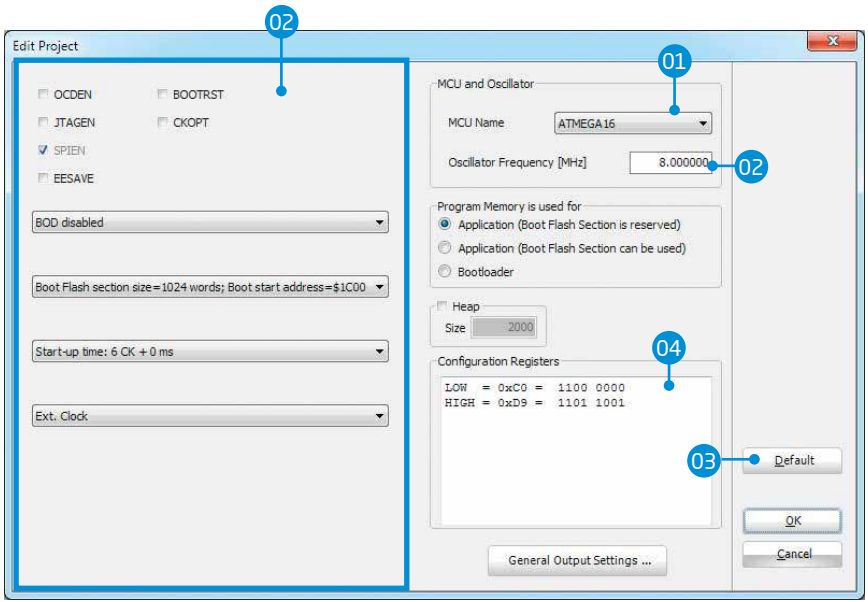


Figure 6-1: Edit Project Window

01 To change your MCU, just select the desired microcontroller from the dropdown list.

02 To change your settings enter the oscillator value and adjust configuration register bits using drop-down boxes.

03 You can always load the default configuration by clicking the **Default** button.

04 For more experienced users there is a box that displays generated values of **LOW** and **HIGH** configuration registers.

7. What's next?

More examples

mikroPascal PRO for AVR® comes with **96 examples** which demonstrate a variety of features. They represent the best starting point when developing a new project. You will find projects written for mikroElektronika development boards, additional boards, internal MCU modules and other examples. This way **you always have a starting point**, and don't have to start from scratch. In most cases, you can combine different simple projects to create a more complex one. For example, if you want to build a temperature datalogger, you can combine temperature sensor example with MMC/SD example and do the job in much less time. All projects are delivered with a working .HEX files, so you don't have to buy a compiler license in order to test them. You can load them into your development board right away without the need for building them.

Community

If you want to find answers to your questions on many interesting topics we invite you to visit our forum at **http://www.mikroe.com/forum** and browse through more than 170 thousand posts. You are likely to find just the right information for you.

On the other hand, if you want to download more free projects and libraries, or share your own code, please visit the **Libstock** website **http://www.libstock.com**. With user profiles, you can get to know other programmers, and subscribe to receive notifications on their code.

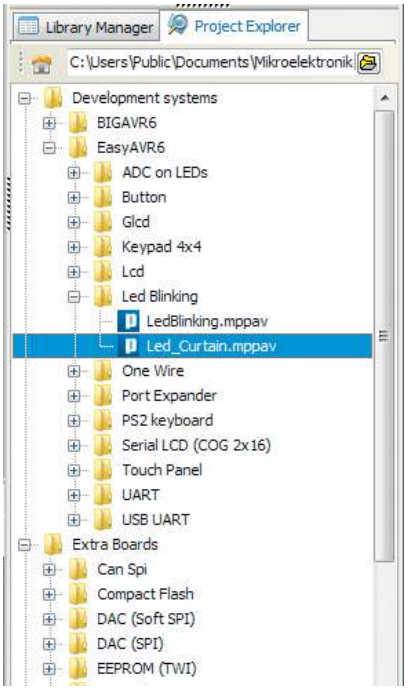


Figure 7-1: Project explorer window enables you to easily access provided examples and load them quickly

DISCLAIMER

All the products owned by MikroElektronika are protected by copyright law and international copyright treaty. Therefore, this manual is to be treated as any other copyright material. No part of this manual, including product and software described herein, may be reproduced, stored in a retrieval system, translated or transmitted in any form or by any means, without the prior written permission of MikroElektronika. The manual PDF edition can be printed for private or local use, but not for distribution. Any modification of this manual is prohibited. MikroElektronika provides this manual 'as is' without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties or conditions of merchantability or fitness for a particular purpose.

MikroElektronika shall assume no responsibility or liability for any errors, omissions and inaccuracies that may appear in this manual. In no event shall MikroElektronika, its directors, officers, employees or distributors be liable for any indirect, specific, incidental or consequential damages (including damages for loss of business profits and business information, business interruption or any other pecuniary loss) arising out of the use of this manual or product, even if MikroElektronika has been advised of the possibility of such damages. MikroElektronika reserves the right to change information contained in this manual at any time without prior notice, if necessary.

HIGH RISK ACTIVITIES

The products of MikroElektronika are not fault – tolerant nor designed, manufactured or intended for use or resale as on – line control equipment in hazardous environments requiring fail – safe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, direct life support machines or weapons systems in which the failure of Software could lead directly to death, personal injury or severe physical or environmental damage ('High Risk Activities'). MikroElektronika and its suppliers specifically disclaim any expressed or implied warranty of fitness for High Risk Activities.

TRADEMARKS

The MikroElektronika name and logo, the MikroElektronika logo, mikroC™, mikroBasic™, mikroPascal™, mikroProg™, EasyAVR6™, mikromedia™ for XMEGA®, mikromedia™ for ATMEGA®, and Ready for AVR® are trademarks of MikroElektronika. All other trademarks mentioned herein are property of their respective companies. All other product and corporate names appearing in this manual may or may not be registered trademarks or copyrights of their respective companies, and are only used for identification or explanation and to the owners' benefit, with no intent to infringe.

If you want to learn more about our products, please visit our website at www.mikroe.com. If you are experiencing some problems with any of our products or just need additional information, please place your ticket at www.mikroe.com/esupport If you have any questions, comments or business proposals, do not hesitate to contact us at office@mikroe.com

Designed by
MikroElektronika,
November 2012.

Creating the first project in
mikroPascal PRO for AVR ver. 2.00