Creating the first project in

mikroPascal
PRO for ARM®

MikroElektronika
DEVELOPMENT TOOLS ∎ COMPILERS ∎ BOOKS

1. Introduction to mikroPascal PRO for ARM

mikroPascal PRO for ARM® organizes applications into projects consisting of a single project file (file with the **.mppar** extension) and one or more source files (files with the .c extension). The mikroPascal PRO for ARM® 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 (*.emcl); 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.
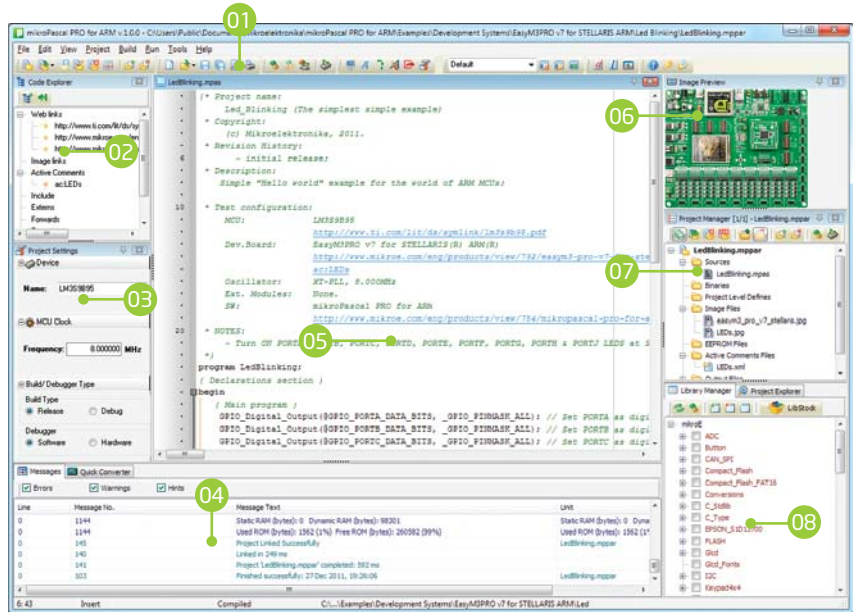
01 Main Toolbar	**04** Messages	**07** Project Manger
02 Code Explorer	**05** Code Editor	**08** Library Manager
03 Project Settings	**06** Image Preview	

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.
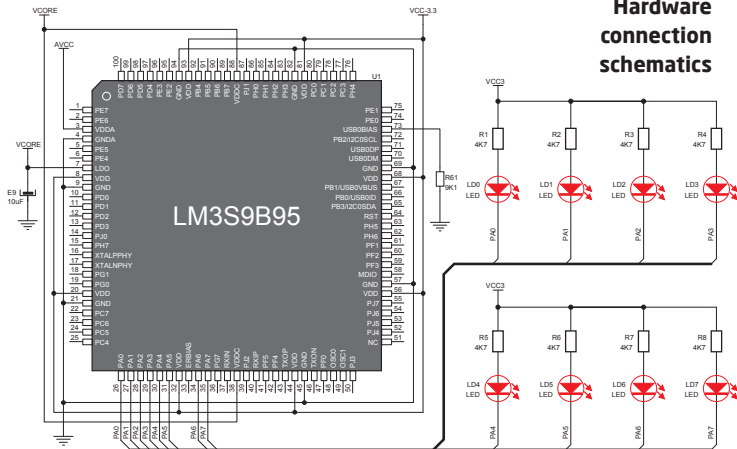

Figure 2-1: Hardware connection schematics

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

Step 1: Install the compiler

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

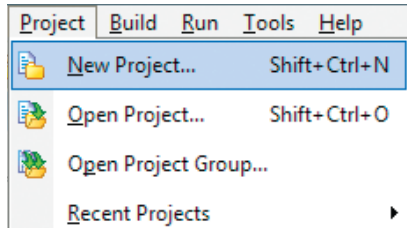**http://www.mikroe.com/eng/products/view/753/mikropascal-pro-for-arm/**

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 ARM® compiler. The mikroPascal PRO for ARM® 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**.

Project	Build	Run	Tools	Help
New Project...			Shift+Ctrl+N	
Open Project...			Shift+Ctrl+O	
Open Project Group...				
Recent Projects			▶	

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.

01 Click **Next**.

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

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 will not change the microcontroller, and we will leave the default **LM3S9B95** as the choice for this project.
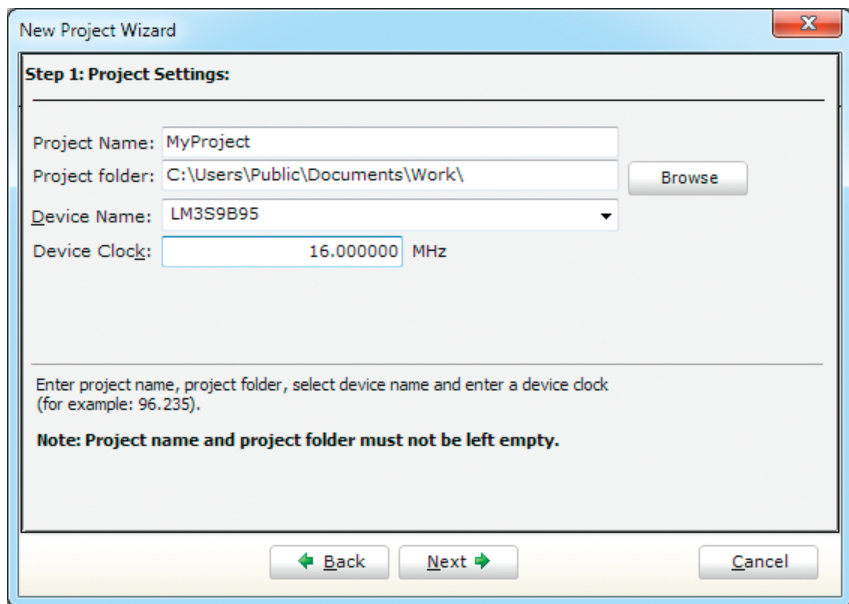
Figure 3-2: You can specify project name, path, device and clock in the first step

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.

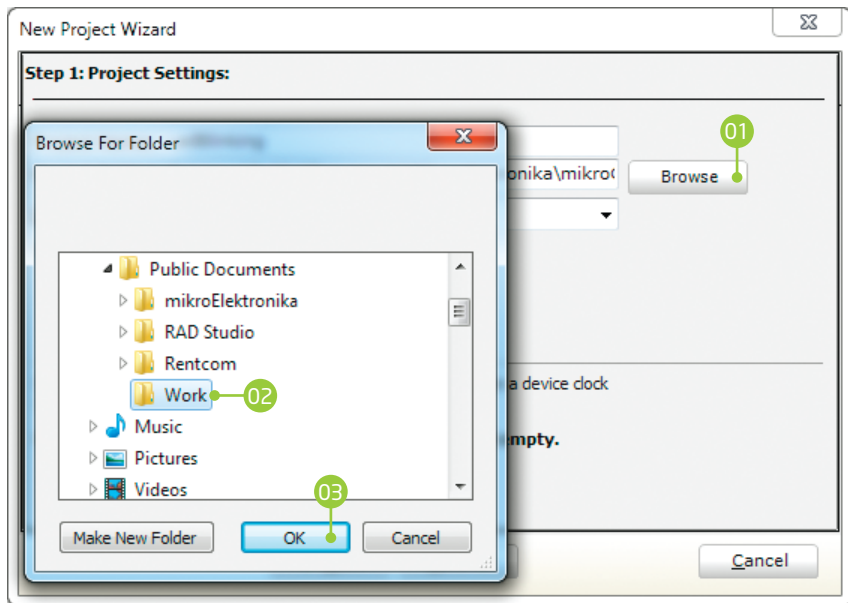**03** Click the **OK** button to confirm your selection and apply the new path.

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 default **16MHz clock**. Clock speed depends on your target hardware, and whether you are using PLL or not. But however you configure your hardware, make sure to specify the exact clock (**Fosc**) that the microcontroller is operating at.
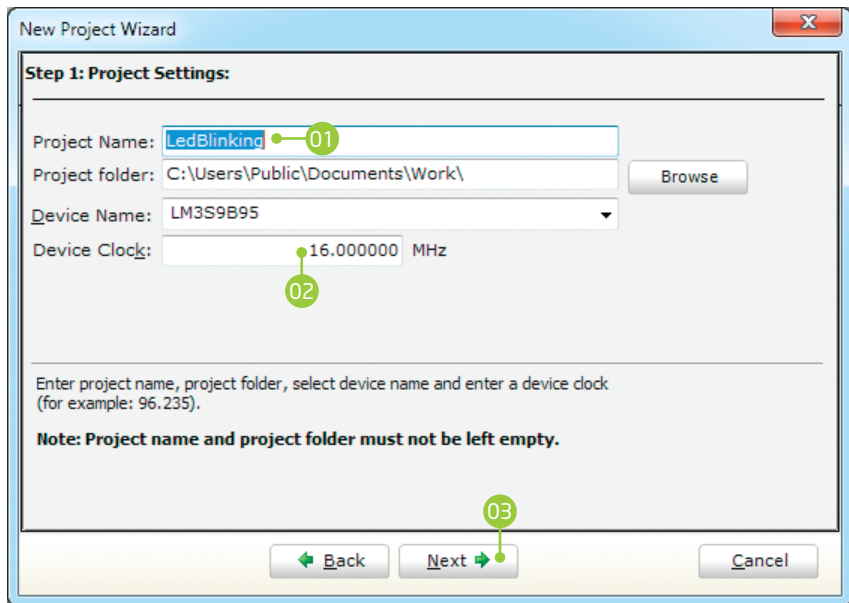
03 Click the **OK** button to proceed.

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.
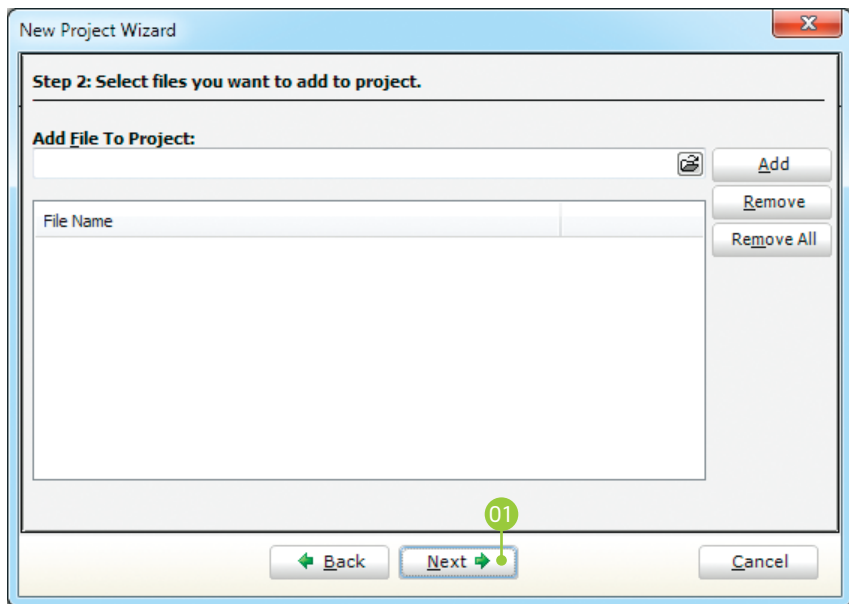
01 Click **Next**.

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 explicitly 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.
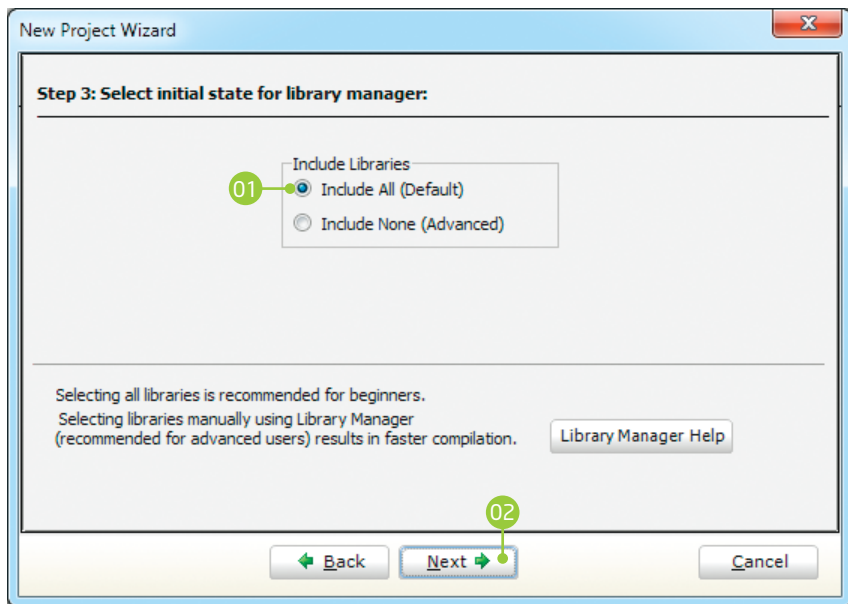
02 Click **Next.**

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 and PLL settings. 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 (internal 16MHz oscillator with PLL disabled). **Therefore, leave the checkbox unchecked.**

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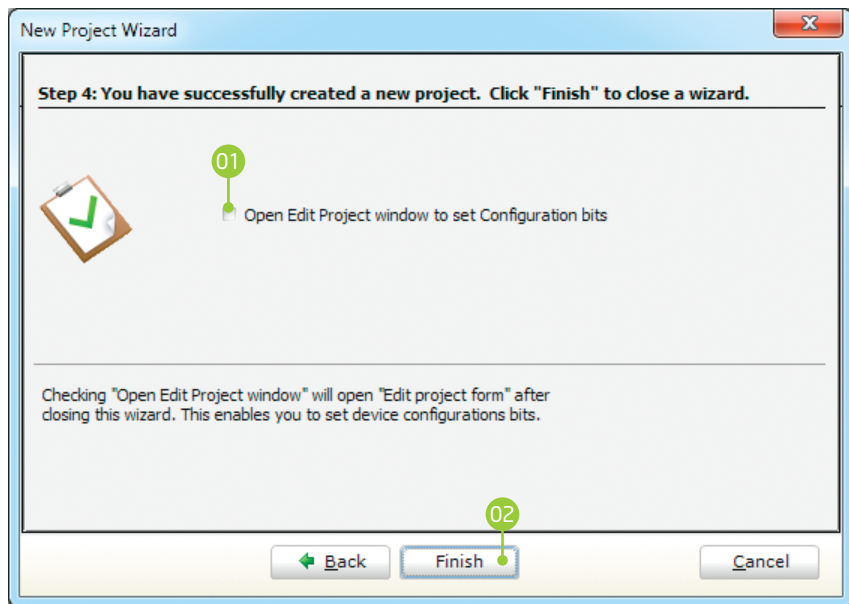
Figure 3-7: Choose whether to open Edit Project window after dialog closes.

Blank new project created

New project is finally created. A new source file called **"LedBlinking.mbas"** is created and it contains the **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**.
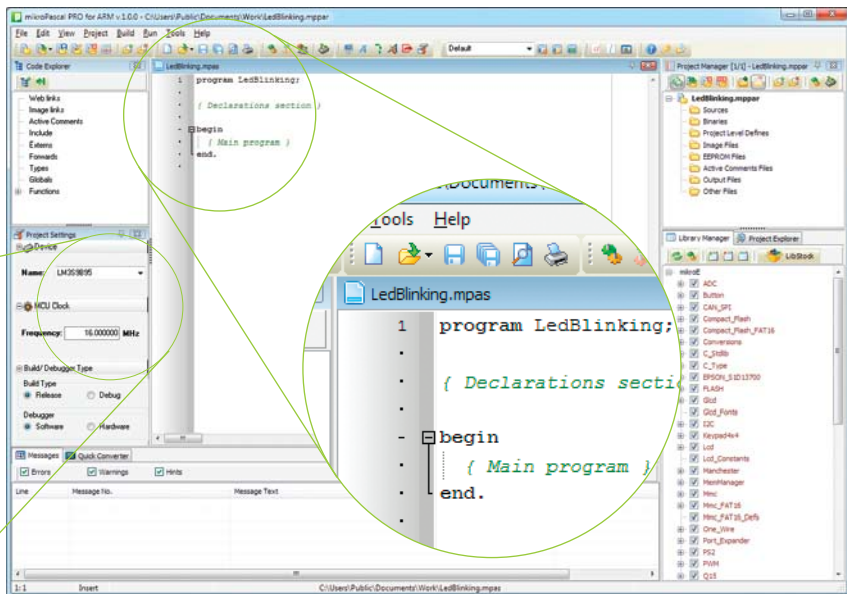

Figure 3-8: New blank project is created with your configuration

4. Code Example

Time has come to do some coding. mikroPascal PRO for ARM® has the unique libraries that enable you to do complicated tasks in a single line of code. Built-in **GPIO library** enables you to set configure each PORT and enable pins that you need, without worrying about complex procedure that this operation requires. To demonstrate this, we will write our first line of code:

```
// Set PORTA as digital output
GPIO_Digital_Output(
  @GPIO_PORTA,
  _GPIO_PINMASK_ALL);
```

Once we have enabled PORTA to act as digital output, we can now initialize PORTA with logic zeros on every PORT pin:

```
// Set PORTA initial value to 0
GPIO_PORTA_DATA = 0;
```

Finally, in a **while()** loop we will toggle the PORTA value, and put a 1000 ms delay, so the blinking is not too fast.

LedBlinking.mbas - source code

```
1    program LedBlinking;
2
3    begin
4      { Main program }
5
6      // Set PORTA as digital output
7      GPIO_Digital_Output(@GPIO_PORTA,
8                          _GPIO_PINMASK_ALL);
9
10     // Set PORTA initial value to zero
11     GPIO_PORTA_DATA := 0;
12
13     while TRUE do
14     begin
15       // Toggle PORTA
16       GPIO_PORTA_DATA := NOT GPIO_PORTA_DATA;
17
18       // Delay 1000 ms
19       Delay_ms(1000);
20     end;
21
22   end.
```

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

File   Edit   View   Project   Build   Run   Tools   Help

```pascal
program LedBlinking;

{ Declarations section }

begin
  { Main program }

  // Set PORTA as digital output
  GPIO_Digital_Output(@GPIO_PORTA,
                      _GPIO_PINMASK_ALL);

  // Set PORTA initial value to zero
  GPIO_PORTA_DATA := 0;

  while TRUE do
  begin
    // Toggle PORTA
    GPIO_PORTA_DATA := NOT GPIO_PORTA_DATA;

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

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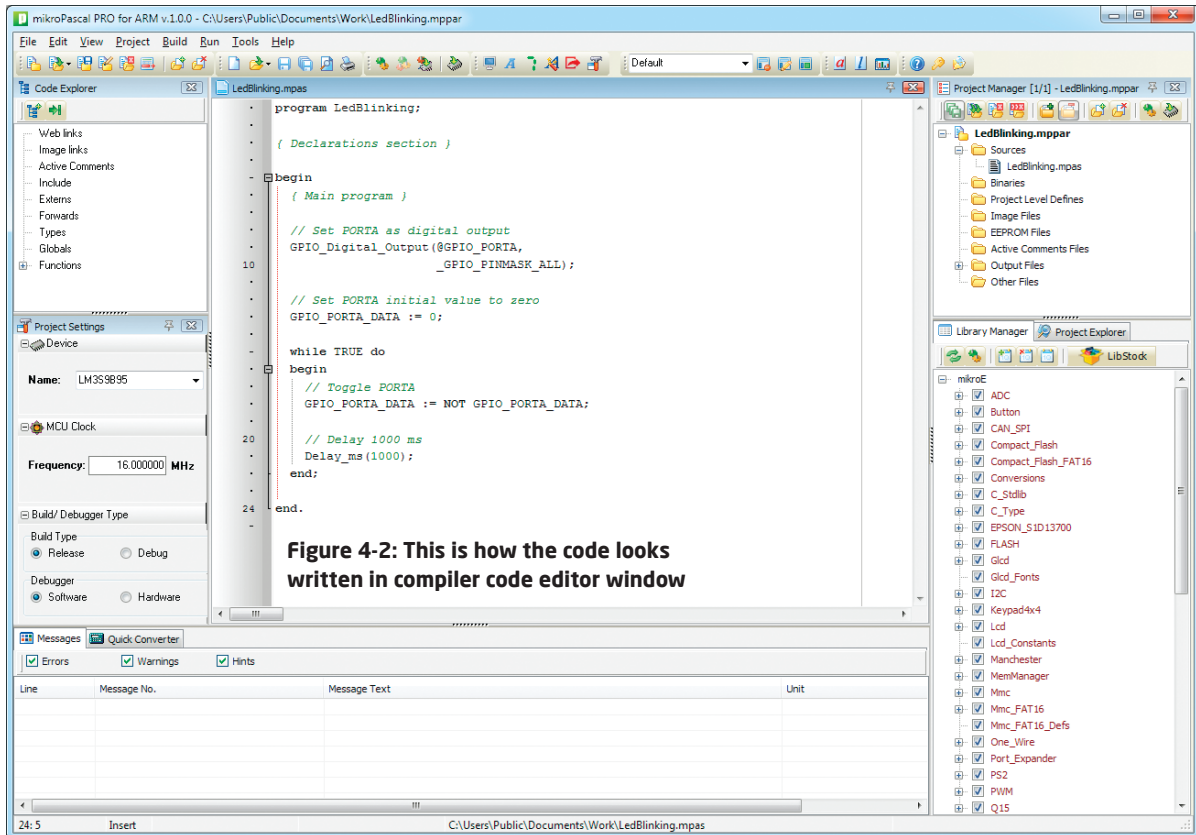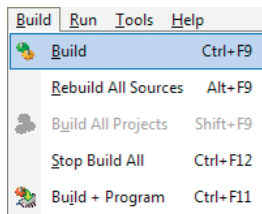
Figure 4-2: This is how the code looks written in compiler code editor window

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 is all done automatically. Build your code by clicking on the ![icon] 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.

Build	Run	Tools	Help
Build			Ctrl+F9
Rebuild All Sources			Alt+F9
Build All Projects			Shift+F9
Stop Build All			Ctrl+F12
Build + Program			Ctrl+F11

Name ▲	Date modified	Type	Size
LedBlinking.asm	2011-12-27 7:40 PM	ASM File	1 KB
LedBlinking.brk	2011-12-27 7:38 PM	BRK File	1 KB
LedBlinking.cfg	2011-12-27 7:40 PM	CFG File	1 KB
LedBlinking.dct	2011-12-27 7:40 PM	Adobe Illustrator S...	625 KB
LedBlinking.dlt	2011-12-27 7:40 PM	DLT File	11 KB
LedBlinking.emcl	2011-12-27 7:40 PM	EMCL File	17 KB
LedBlinking.hex	2011-12-27 7:40 PM	HEX File	4 KB
LedBlinking.log	2011-12-27 7:40 PM	Text Document	3 KB
LedBlinking.lst	2011-12-27 7:40 PM	LST File	24 KB
LedBlinking.mpas	2011-12-27 7:38 PM	MPAS File	1 KB
LedBlinking.mpas.ini	2011-12-27 7:38 PM	Configuration sett...	1 KB
LedBlinking.mppar	2011-12-27 7:40 PM	mikroPascal proje...	2 KB
LedBlinking.mppar_callertable.txt	2011-12-27 7:40 PM	TXT File	1 KB
LedBlinking.user.dic	2011-12-27 7:40 PM	Text Document	0 KB
LedBlinking.dbg	2011-12-27 7:40 PM	DBG File	133 KB

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

Line	Message No.	Message Text	Unit
0	1144	Static RAM (bytes): 0 Dynamic RAM (bytes): 98301	Static RAM (bytes): 0 Dyna
0	1144	Used ROM (bytes): 1398 (1%) Free ROM (bytes): 260746 (99%)	Used ROM (bytes): 1398 (1
0	145	Project Linked Successfully	LedBlinking.mppar
0	140	Linked in 764 ms	
0	141	Project 'LedBlinking.mppar' completed: 1138 ms	
0	103	Finished successfully: 27 Dec 2011, 19:40:08	LedBlinking.mppar

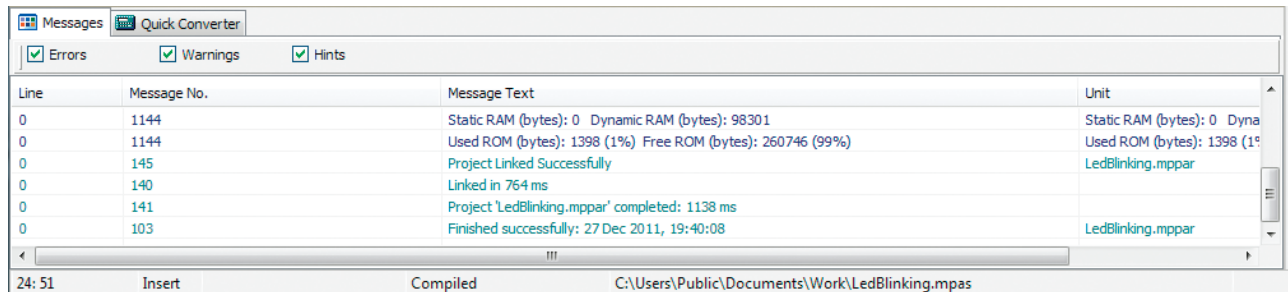| 24: 51 | Insert | Compiled | C:\Users\Public\Documents\Work\LedBlinking.mpas |

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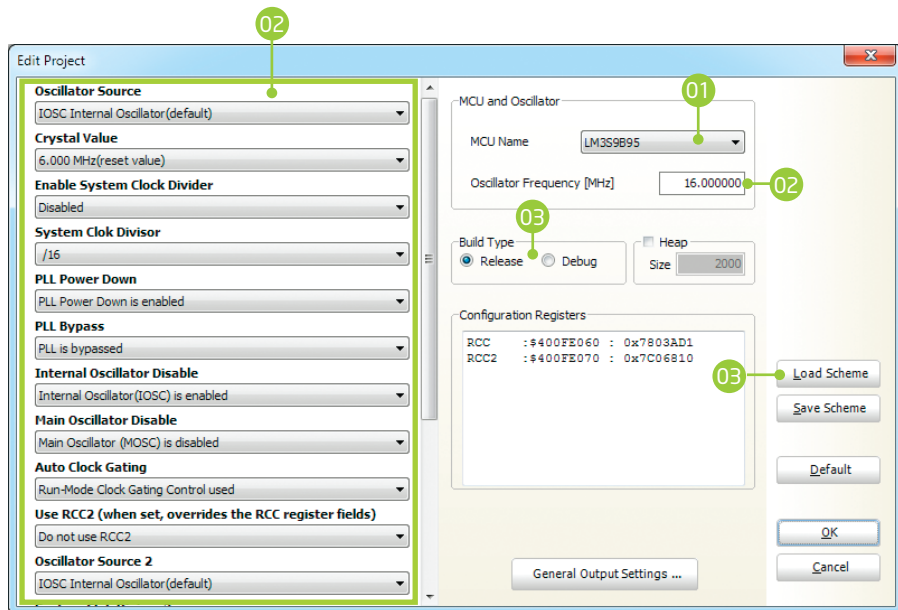
01 To change your MCU, just select the desired microcontroller from the dropdown list.

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

03 Several most commonly used oscillator settings can be loaded using the provided oscillator "schemes". Load the desired scheme by clicking the **Load Scheme** button.

04 Select whether to build a **Debug HEX**, which is necessary for hardware debugging, or a final **Release HEX**.

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

X-ON Electronics

Largest Supplier of Electrical and Electronic Components

Click to view similar products for Development Software *category:*

Click to view products by MikroElektronika *manufacturer:*

Other Similar products are found below :

SRP004001-01 SW163052 SYSWINEV21 WS01NCTF1E W128E13 SW89CN0-ZCC IP-UART-16550 MPROG-PRO535E AFLCF-08-LX-CE060-R21 WS02-CFSC1-EV3-UP SYSMAC-STUDIO-EIPCPLR LIB-PL-PC-N-1YR-DISKID 1120270005 1120270006 MIKROBASIC PRO FOR FT90X (USB DONGLE) MIKROC PRO FOR FT90X (USB DONGLE) MIKROBASIC PRO FOR AVR (USB DONGLE LICEN MIKROBASIC PRO FOR FT90X MIKROC PRO FOR DSPIC30/33 (USB DONGLE LI MIKROPASCAL PRO FOR ARM (USB DONGLE LICE MIKROPASCAL PRO FOR FT90X MIKROPASCAL PRO FOR FT90X (USB DONGLE) MIKROPASCAL PRO FOR PIC32 (USB DONGLE LI SW006021-2H ATATMELSTUDIO 2400573 2702579 2988609 SW006022-DGL 2400303 88970111 DG-ACC-NET-CD 55195101-101 55195101-102 SW1A-W1C MDK-ARM SW006021-2NH B10443 SW006021-1H SW006021-2 SW006022-2 SW006023-2 SW007023 MIKROE-730 MIKROE-2401 MIKROE-499 MIKROE-722 MIKROE-724 MIKROE-726 MIKROE-728