## Creating the first project in

# **mikroC** PRO for ARM<sup>®</sup>



## 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

ARM® and Windows® logos and product names are trademarks of ARM® Holdings and Microsoft® in the U.S.A. and other countries.

# Table of Contents

1. Introduction to mikroC PRO for ARM <sup>®</sup> 04
2. Hardware connection
3. Creating a new project
Step 1 - Project settings07
Step 2 - Add files 10
Step 3 - Include libraries 11
Step 4 - Finishing 12
Blank new project created13
4. Code example
5. Building the source
6. Changing project settings
7. What's next

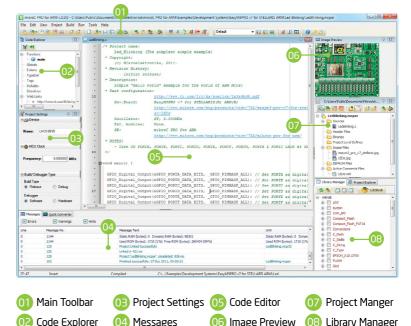
## **1. Introduction to mikroC PRO for ARM®**

**mikroC PRO for ARM**<sup>\*</sup> organizes applications into projects consisting of a single project file (file with the **.mcpar** extension) and one or more source files (files with the .c extension). The mikroC PRO for ARM<sup>\*</sup> compiler allows you to manage several projects at a time. Source files can be compiled only if they are part of a project.

A project file contains:

- Project name and optional description;
- Target device in use;
- Device clock;
- List of 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.



## 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 C 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 schematic

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

## Step 1: Install the compiler

Install mikroC PRO for ARM<sup>®</sup> from the **Product DVD** or download it from the MikroElektronika website:



http://www.mikroe.com/mikroc/arm/

#### Step 2: Start up the compiler

Double click on the compiler icon in the Start menu, or on your desktop to Start up mikroC PRO for ARM<sup>®</sup>. The mikroC PRO for ARM<sup>®</sup> 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**.

<u>P</u> roj	ect	<u>B</u> uild	<u>R</u> un	<u>T</u> ools	<u>H</u> elp
В.	<u>N</u> e	w Projec	:t	Shif	t+Ctrl+N
8	<u>O</u> p	en Proje	ect	Shif	t+Ctrl+O
8	0 <u>p</u>	en Proje	ect Gro	up	
	<u>R</u> ec	ent Pro	jects		•

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.



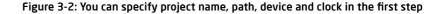


Figure 3-1: Introductory window of the New Project Wizard

## 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, its operating clock frequency, and of course - naming our project. This is an important step, because the 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.

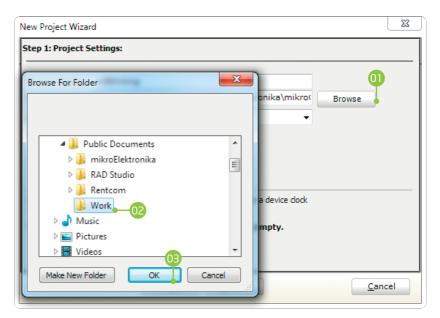
New Project Wiza	rd	×		
Step 1: Project 9	iettings:			
Project Name:	MyProject			
Project folder:	C:\Users\Public\Documents\Work\	Browse		
Device Name:	LM3S9B95 🗸			
Device Clock:	16.000000 MHz			
Enter project name, project folder, select device name and enter a device dock (for example: 96.235). Note: Project name and project folder must not be left empty.				
	ext	<u>C</u> ancel		



### 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:

- OI Click the Browse button of the Project Settings window to open the Browse for Folder dialog.
- Select the desired folder to be the destination path for storing your new project files.
- Click the **OK** button to confirm your selection and apply the new path.

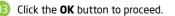




#### Step 1 - Project settings

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

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



New Project Wiza	rd	×		
Step 1: Project 9	settings:			
Project Name:	LedBlinking -01			
	C:\Users\Public\Documents\Work\	Browse		
Device Name:	LM3S9B95			
Device Cloc <u>k</u> :	•16.000000 MHz			
	02			
Enter project name, project folder, select device name and enter a device clock				
(for example: 96.				
Note: Project name and project folder must not be left empty.				
<u></u>				
	◆ <u>B</u> ack <u>N</u> ext ◆	<u>C</u> ancel		

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.

New Project Wizard	×
Step 2: Select files you want to add to project.	
Add File To Project:	
ĺ.	
File Name	Remove All
	Ke <u>m</u> ove An
0	
◆ <u>B</u> ack Next ◆	<u>C</u> ancel

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:



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

2 Click Next.

New Project Wizard	×
Step 3: Select initial state for library manager:	
Include Libraries Include All (Default) Include None (Advanced)	
Selecting all libraries is recommended for beginners. Selecting libraries manually using Library Manager (recommended for advanced users) results in faster compilation.	Library Manager Help
● <u>B</u> ack Next ●	<u>C</u> ancel

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

#### **Step 4 - Finishing**

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

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.



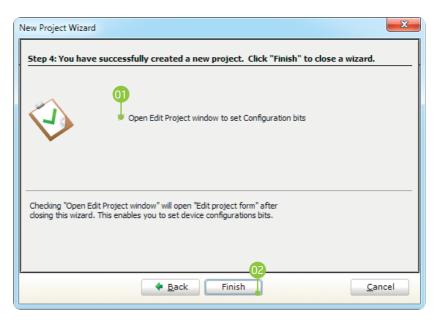
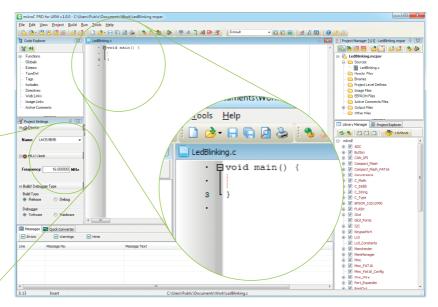


Figure 3-7: Choose whether to open Edit Project window after dialog closes.

#### Blank new project created

New project is finally created. A new source file called **"LedBlinking.c"** is created and it contains the **void main()** function, which will hold the program. You may notice that project is configured according to the settings done in the **New Project Wizard**.

	Project S	Settings 🏻 🏯 🖄		
	🗆 🧼 Devic	e		
Λ				
	Name:	LM3S9B95 👻		
	⊡@ MCU Clock			
V	Frequen	cy: 16.000000 MHz		
	. request			
oebugger Type				



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

## 4. Code example

Time has come to do some coding. mikroC PRO for ARM<sup>®</sup> 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
GPI0_Digital_Output(
    &GPI0_PORTA,
    _GPI0_PINMASK_ALL);
```

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

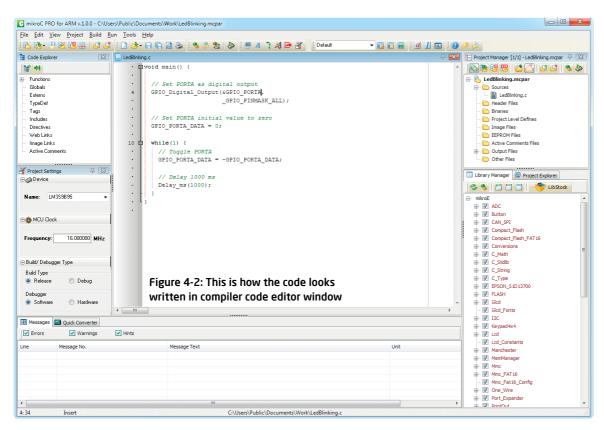
// Set PORTA initial value to 0
GPIO\_PORTA\_DATA = 0;

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.c - source code

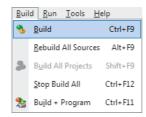
```
void main() {
     // Set PORTA as digital output
     GPIO Digital Output (& GPIO PORTA,
                          GPIO PINMASK ALL);
     // Set PORTA initial value to zero
     GPIO PORTA DATA = 0;
     while(1) {
       // Toggle PORTA
       GPIO PORTA DATA = ~GPIO PORTA DATA;
14
       // Delav 1000 ms
       Delay ms(1000);
```

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



## 5. Building the source

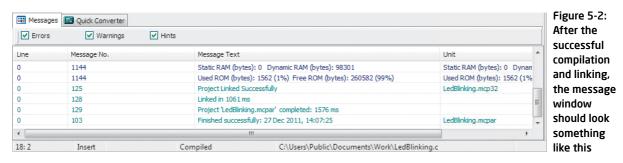
When we are done writing our first LedBlinking code, we can now build the project and create a **.HEX** file which can be loaded into our target microcontroller, so we can test the program on real hardware. "Building" includes compilation, linking and



optimization which is all done automatically. Build your code by clicking on the sicon 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	Туре	Size
📋 LedBlinking.asm	2011-12-27 2:07 PM	ASM File	1 KB
LedBlinking.brk	2011-12-27 11:39	BRK File	1 KB
🖻 LedBlinking.c	2011-12-27 11:44	C File	1 KB
🗿 LedBlinking.c.ini	2011-12-27 11:39	Configuration sett	1 KB
🖻 LedBlinking.cfg	2011-12-27 2:07 PM	CFG File	1 KB
🖻 LedBlinking.cp	2011-12-27 2:07 PM	CP File	1 KB
LedBlinking.dbg	2011-12-27 2:07 PM	DBG File	912 KB
LedBlinking.dct	2011-12-27 2:07 PM	Adobe Illustrator S	652 KB
LedBlinking.dlt	2011-12-27 2:07 PM	DLT File	13 KB
LedBlinking.emcl	2011-12-27 2:07 PM	EMCL File	2 KB
🖻 LedBlinking.hex	2011-12-27 2:07 PM	HEX File	5 KB
📄 LedBlinking.log	2011-12-27 2:07 PM	Text Document	3 KB
🖻 LedBlinking.lst	2011-12-27 2:07 PM	LST File	23 KB
🕎 LedBlinking.mcpar	2011-12-27 2:07 PM	mikroC PRO for A	2 KB
📄 LedBlinking.mcpar_callertable.txt	2011-12-27 2:07 PM	TXT File	1 KB
📄 LedBlinking.user.dic	2011-12-27 2:07 PM	Text Document	0 KB

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



# 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.

Edit Project		01	To change your MCU, just select the
Oscillator Source	MCU and Oscillator		desired microcontroller from the
IOSC Internal Oscillator(default)	Pico and Oscillator		dropdown list.
Crystal Value	MCU Name LM3S9B95		
6.000 MHz(reset value)			
Enable System Clock Divider	Oscillator Frequency [MHz] 16.0000 -02	02	To change your oscillator settings
Disabled			enter the oscillator value and adjust
System Clok Divisor	-Build Type		•
/16	Release      Debug     Size     2000		oscillator configuration registers us-
PLL Power Down			ing drop-down boxes.
PLL Power Down is enabled	-Configuration Registers		
PLL Bypass			
PLL is bypassed 🗸	RCC :\$400FE060 : 0x7803AD1 RCC2 :\$400FE070 : 0x7C06810	03	Several most commonly used oscil-
Internal Oscillator Disable	03 <u>L</u> oad Scheme		lator settings can be loaded using
Internal Oscillator(IOSC) is enabled	Save Scheme		• •
Main Oscillator Disable	Zave Scheme		the provided oscillator "schemes".
Main Oscillator (MOSC) is disabled 🔹			Load the desired scheme by clicking
Auto Clock Gating	Default		the <b>Load Scheme</b> button.
Run-Mode Clock Gating Control used 🔹			
Use RCC2 (when set, overrides the RCC register fields)			
Do not use RCC2 🔹	<u>Q</u> K	04	Select whether to build a <b>Debug</b>
Oscillator Source 2	General Output Settings Cancel		<b>HEX</b> , which is necessary for hardware
IOSC Internal Oscillator(default)	General Output Settings		5
••••••••••••••••••••••••••••••••••••••		J	debugging, or a final <b>Release HEX</b> .

Figure 6-1: Edit Project Window

## 7. What's next?

### More examples

mikroC PRO for ARM<sup>®</sup> comes with over **195 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 gives you a head start in development, and you don't have to do it all 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 date, time and temperature semaphore on 7-segment display, you can combine RTC and temperature sensor examples with Seven Segment Display example and do the job in much less time. All projects are delivered with 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 220,000 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.

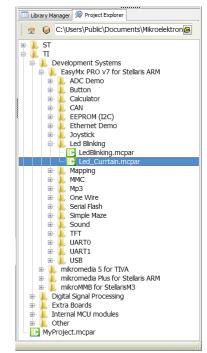


Figure 6-1: Project explorer window enables you to easily access provided examples and load them quickly

#### 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, mikroC<sup>™</sup>, mikroProg<sup>™</sup>, mikroBUS<sup>™</sup>, click<sup>™</sup> boards, EasyMx PRO<sup>™</sup> and mikromedia<sup>™</sup> 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.

Copyright © 2014 MikroElektronika. All Rights Reserved.

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/support If you have any questions, comments or business proposals, do not hesitate to contact us at office@mikroe.com

> Designed by MikroElektronika Ltd. **www.mikroe.com**

> > Creating the first project in mikroC PRO for ARM



## **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 ARM 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 2 SW006023-2 SW007023 MIKROE-730 MIKROE-2401 MIKROE-499 MIKROE-722 MIKROE-724 MIKROE-726 MIKROE-728