Creating the first project in

mikroPascal PRO for FT90x

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. Installation

mikroPascal PRO for FT90x is a full-featured Pascal compiler for FT90x devices from FTDI Chip®. It features an intuitive IDE, powerful compiler with advanced optimizations, lots of hardware and software libraries, and additional tools that will help you in your work. The compiler comes with a comprehensive Help file and lots of ready-to-use examples designed to get you started in no time. Compiler license includes free upgrades and product lifetime tech support, so you can rely on our help while developing.

Download and install the compiler

Download and install the mikroPascal PRO for FT90x compiler from the MikroElektronika website:

www.mikroe.com/mikropascal/ft90x

2. Activating compiler license

MikroElektronika compiler license (full or time trial) is the permission to use the MikroElektronika compiler with all its features. License is granted to the customer by MikroElektronika as a licensor. The customer must purchase the license rights and then apply a **License Activation Key** to unlock the compiler and remove the demo limit. The customer is bound to comply with the Terms of Usage defined in the Software License Agreement.

What is an activation key?

It's a uniqe sequence of 20 characters and numbers delivered to the user upon purchase. The Activation Key is located at the back of the License Activation Card. At all times, Activation Key remains the property of MikroElektronika.

What is software activation?

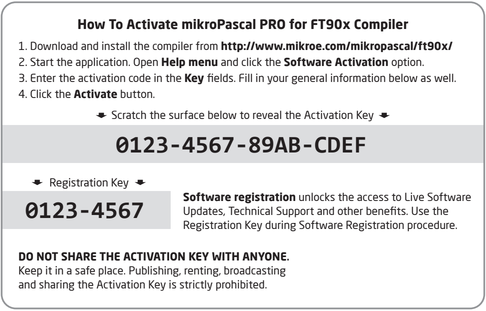
Activation is a process of validating of the Activation Key. Successful Activation removes the Demo Limit and unlocks all software features.

How to perform the activation?

1. Start the application. Open the **Help menu** and click the **Software Activation** option.

2. Enter the **Activation Key** in the **Key** fields. Type in your general information in the fields below as well.

3. Click the **Activate** button.



How To Activate mikroPascal PRO for FT90x Compiler

1. Download and install the compiler from **http://www.mikroe.com/mikropascal/ft90x/**
2. Start the application. Open **Help menu** and click the **Software Activation** option.
3. Enter the activation code in the **Key** fields. Fill in your general information below as well.
4. Click the **Activate** button.

▼ Scratch the surface below to reveal the Activation Key ▼

0123-4567-89AB-CDEF

▼ Registration Key ▼

0123-4567

Software registration unlocks the access to Live Software Updates, Technical Support and other benefits. Use the Registration Key during Software Registration procedure.

DO NOT SHARE THE ACTIVATION KEY WITH ANYONE.
Keep it in a safe place. Publishing, renting, broadcasting and sharing the Activation Key is strictly prohibited.

Figure 2-1: Back side of the license activation card

What is software registration?

Registration is a process which establishes a unique connection between MikroElektronika as a Licensor and the customer as a Licensee. By registering a copy of the Compiler, the customer is granted access to Live Updates, Technical Support and other benefits.

Why should I keep my activation key a secret?

Publishing, renting, public performance, broadcasting or otherwise disclosing the Activation Key to a third party is strictly prohibited. By doing so a customer may loose all benefits granted by registration. In case of severe violations of Software License Agreement, MikroElektronika reserves the right to delicense the customer and request the removal of the Activation Key from the customer's computer.

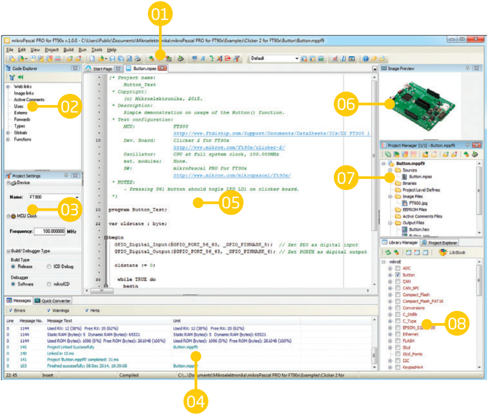
3. Overview

mikroPascal PRO for FT90x organizes applications into projects consisting of a single project file (file with the **.mppf9** extension) and one or more source files (files with the .mpas extension). The mikroPascal PRO for FT90x 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 two microcontroller LEDs blink, which will be easy to test.



Project Settings

Device

Name: FT900

MCU Clock

Frequency: 100.000000 MHz

Build/ Debugger Type

Build Type: ● Release ○ ICD Debug

Debugger: ● Software ○ mikroICD

```
(* Project name:
    Button_Test
 * Copyright:
    (c) Mikroelektronika, 2015.
 * Description:
    Simple demonstration on usage of the Button() function.
 * Test configuration:
    MCU:             FT900
                     http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT900_1
    Dev. Board:      Clicker 2 for FT90x
                     http://www.mikroe.com/ft90x/clicker-2/
    Oscillator:      CPU at full system clock, 100.000MHz
    ext. modules:    None.
    SW:              mikroPascal PRO for FT90x
                     http://www.mikroe.com/mikropascal/ft90x/
 * NOTES:
    - Pressing P61 button should togle LED LD1 on clicker board.
 *)

program Button_Test;

var oldstate : byte;

begin
  GPIO_Digital_Input(@GPIO_PORT_56_63, _GPIO_PINMASK_5);  // Set PE0 as digital input
  GPIO_Digital_Output(@GPIO_PORT_56_63, _GPIO_PINMASK_4); // Set PORTH as digital output

  oldstate := 0;

  while TRUE do
    begin
```

☑ Errors ☑ Warnings ☑ Hints

```
Line  Message No.  Message Text                                             Unit
0     1144         Used RX: 12 (38%)  Free RX: 20 (62%)                     Used RX: 12 (38%)  Free RX: 20 (62%)
0     1144         Static RAM (bytes): 5  Dynamic RAM (bytes): 65521        Static RAM (bytes): 5  Dynamic RAM (bytes): 65521
0     1144         Used ROM (bytes): 1096 (0%)  Free ROM (bytes): 261048 (100%)  Used ROM (bytes): 1096 (0%)  Free ROM (bytes): 261048 (100%)
0     145          Project Linked Successfully                              Button.mppf9
0     140          Linked in 15 ms
0     141          Project 'Button.mppf9' completed: 31 ms
0     103          Finished successfully: 08 Dec 2014, 19:39:08             Button.mppf9
```

22: 45 Insert Compiled C:\...\Documents\Mikroelektronika\mikroPascal PRO for FT90x\Examples\Clicker 2 for

01 Main Toolbar

02 Code Explorer

03 Project Settings

04 Messages

05 Code Editor

06 Image Preview

07 Project Manger

08 Library Manager

4. Hardware connections

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 GPIO pins. In order to see the example in action, it is necessary to connect the target microcontroller according to schematics shown on **Figure 4-1**. In the project we are about to write, we will use **LED1** and **LED2** provided on **clicker 2 for FT90x** development board.

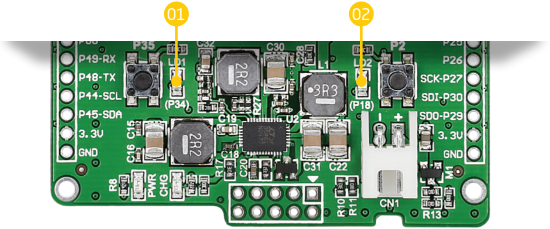
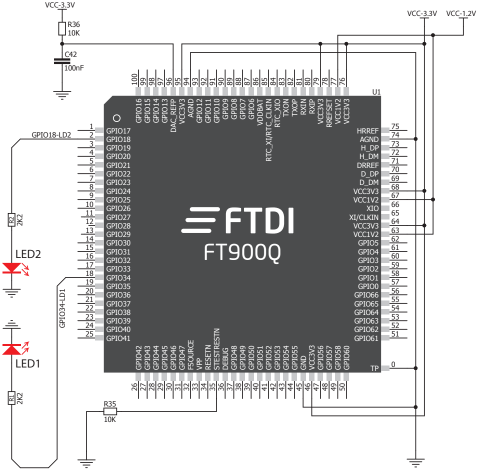
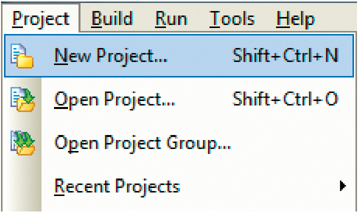


Figure 4-1:
Hardware connection schematics



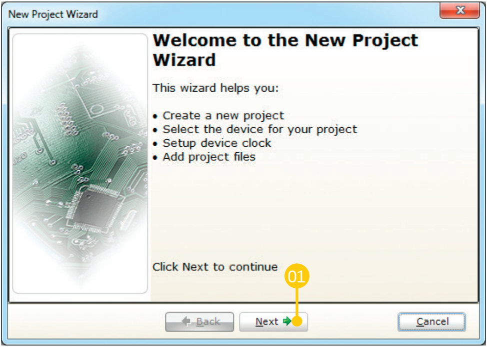
5. Creating the first project

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 5-1**) 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 5-1:
Introductory window of the New Project Wizard



Welcome to the New Project Wizard

This wizard helps you:

- Create a new project
- Select the device for your project
- Setup device clock
- Add project files

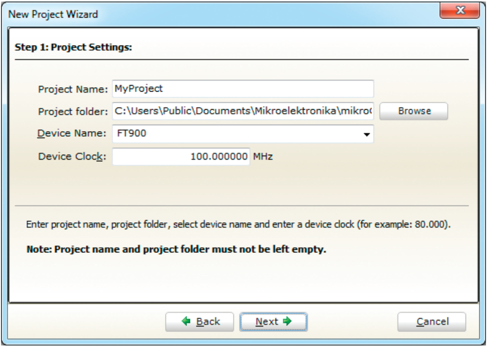
Click Next to continue

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, 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 FT900 as the choice for this project.

Figure 5-2: You can specify project name, path, device and clock in the first step



Step 1: Project Settings:

Project Name: MyProject

Project folder: C:\Users\Public\Documents\Mikroelektronika\mikro(

Device Name: FT900

Device Clock: 100.000000 MHz

Enter project name, project folder, select device name and enter a device clock (for example: 80.000).

Note: Project name and project folder must not be left empty.

Step 1 - Project Settings

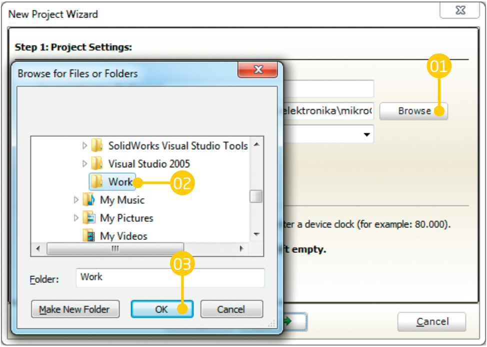
If you do not want to use the suggested path for storing your new project, you can **change the destination folder**. In order to do that, follow a simple procedure:

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 5-3: Change the destination folder using Browse For Folder dialog



New Project Wizard

Step 1: Project Settings:

elektronika\mikro(

ter a device clock (for example: 80.000).

t empty.

Browse for Files or Folders

Folder: Work

Step 1 - Project Settings

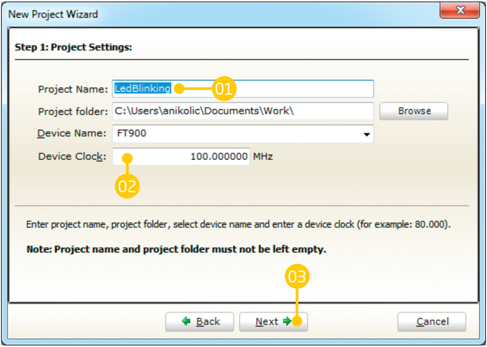
Once we have selected the destination project folder, let's do the rest of the project settings:

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 **100MHz clock**. Clock speed depends on your target hardware. Always make sure to specify the exact clock (**Fosc**) that the microcontroller is operating at.

03 Click **Next** to proceed.

Figure 5-4: Enter project name and change device clock speed if necessary



New Project Wizard

Step 1: Project Settings:

Project Name: LedBlinking

Project folder: C:\Users\anikolic\Documents\Work\

Device Name: FT900

Device Clock: 100.000000 MHz

Enter project name, project folder, select device name and enter a device clock (for example: 80.000).

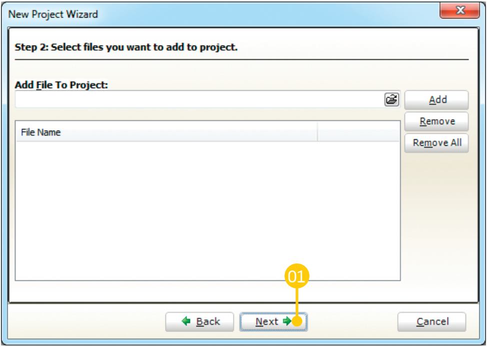
Note: Project name and project folder must not be left empty.

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



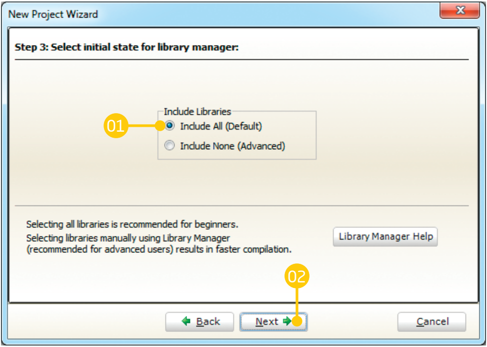
Step 3 - Include Libraries

The 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 **600 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 5-6: Include all libraries in the project, which is a default configuration.



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.

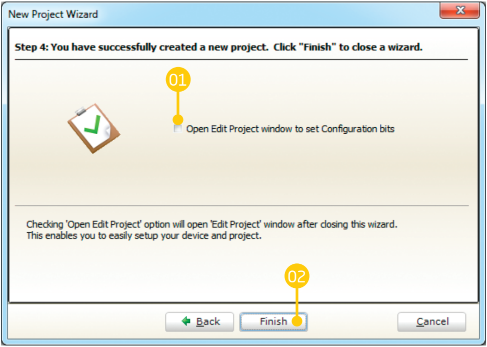
Step 4 - Finishing

After the configuring is done, this 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 other configuration 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. **Therefore, leave the checkbox unchecked.**

02 Click **Finish**.

Figure 5-7: Choose whether to open Edit Project window after dialog closes.



New Project Wizard

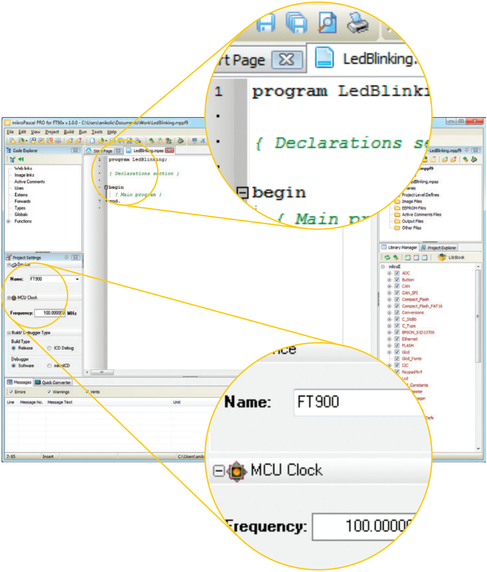
Step 4: You have successfully created a new project. Click "Finish" to close a wizard.

□ Open Edit Project window to set Configuration bits

Checking 'Open Edit Project' option will open 'Edit Project' window after closing this wizard.
This enables you to easily setup your device and project.

Blank new project created

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



```
program LedBlinking;

{ Declarations section }

begin
  { Main program }
end.
```

Project Settings

Device

Name: FT900

MCU Clock

Frequency: 100.000000 MHz

Build/ Debugger Type

Build Type
● Release ○ ICD Debug

Debugger
● Software ○ mikroICD

Figure 5-8:
New blank project is created with your configuration

6. Code example

Time has come to do some coding. First thing we need to do is to initialize the GPIO to act as digital output.

```
//Set LED1 & LED2 to be digital outputs
GPIO_Pin_Digital_Output(_GPIO_PIN_NUM_60);
GPIO_Pin_Digital_Output(_GPIO_PIN_NUM_17);
```

We can now initialize both LEDs with logic zeros:

```
//Turn OFF both LED1 & LED2
GPIO_PIN60_bit := 0;
GPIO_PIN17_bit := 0;
```

Finally, in a **while ... do** loop we will toggle each GPIO pin value, and put a 500 ms delay, so the blinking is not too fast.

```
while true do
begin
  GPIO_PIN60_bit := not GPIO_PIN60_bit;
  Delay_ms(500);
  GPIO_PIN17_bit := not GPIO_PIN17_bit;
  Delay_ms(500);
end;
```

LedBlinking.mpas - source code

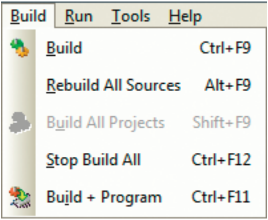
```
program LedBlinking;
begin
  //Set LED1 & LED2 to be digital outputs
  GPIO_Pin_Digital_Output(_GPIO_PIN_NUM_60);
  GPIO_Pin_Digital_Output(_GPIO_PIN_NUM_17);

  //Turn OFF both LED1 & LED2
  GPIO_PIN60_bit := 0;
  GPIO_PIN17_bit := 0;

  //Toggle LEDs one by one
  while true do
    begin
      GPIO_PIN60_bit := not GPIO_PIN60_bit;
      Delay_ms(500);
      GPIO_PIN17_bit := not GPIO_PIN17_bit;
      Delay_ms(500);
    end;
end.
```

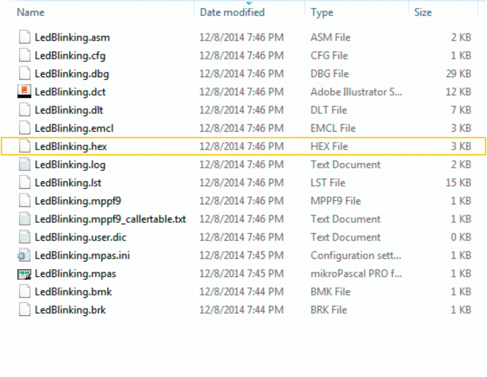
7. Building the source

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. Compiler automatically creates necessary output files. **LedBlinking.hex** (**Figure 7-1**) is among them.

Figure 7-1:
Listing of project files after building is done



Name	Date modified	Type	Size
LedBlinking.asm	12/8/2014 7:46 PM	ASM File	2 KB
LedBlinking.cfg	12/8/2014 7:46 PM	CFG File	1 KB
LedBlinking.dbg	12/8/2014 7:46 PM	DBG File	29 KB
LedBlinking.dct	12/8/2014 7:46 PM	Adobe Illustrator S...	12 KB
LedBlinking.dlt	12/8/2014 7:46 PM	DLT File	7 KB
LedBlinking.emcl	12/8/2014 7:46 PM	EMCL File	3 KB
LedBlinking.hex	12/8/2014 7:46 PM	HEX File	3 KB
LedBlinking.log	12/8/2014 7:46 PM	Text Document	2 KB
LedBlinking.lst	12/8/2014 7:46 PM	LST File	15 KB
LedBlinking.mppf9	12/8/2014 7:46 PM	MPPF9 File	1 KB
LedBlinking.mppf9_callertable.txt	12/8/2014 7:46 PM	Text Document	1 KB
LedBlinking.user.dic	12/8/2014 7:46 PM	Text Document	0 KB
LedBlinking.mpas.ini	12/8/2014 7:45 PM	Configuration sett...	1 KB
LedBlinking.mpas	12/8/2014 7:45 PM	mikroPascal PRO f...	1 KB
LedBlinking.bmk	12/8/2014 7:44 PM	BMK File	1 KB
LedBlinking.brk	12/8/2014 7:44 PM	BRK File	1 KB

8. Changing project settings

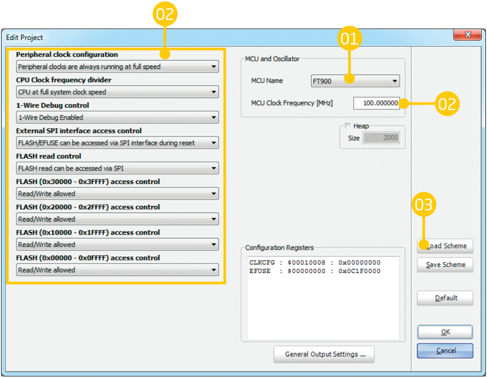
If you need to change the target microcontroller or clock speed, you don't have to go through the new project wizard all over again. This can be done quickly in the **Edit Project** window. You can open it using **Project->Edit Project [CTRL+SHIFT+E]** menu option.

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

Figure 8-1:
Edit Project Window



Peripheral clock configuration
Peripheral clocks are always running at full speed

CPU Clock frequency divider
CPU at full system clock speed

1-Wire Debug control
1-Wire Debug Enabled

External SPI interface access control
FLASH/EFUSE can be accessed via SPI interface during reset

FLASH read control
FLASH read can be accessed via SPI

FLASH (0x30000 - 0x3FFFF) access control
Read/Write allowed

FLASH (0x20000 - 0x2FFFF) access control
Read/Write allowed

FLASH (0x10000 - 0x1FFFF) access control
Read/Write allowed

FLASH (0x00000 - 0x0FFFF) access control
Read/Write allowed

MCU and Oscillator

MCU Name: FT900

MCU Clock Frequency [MHz]: 100.000000

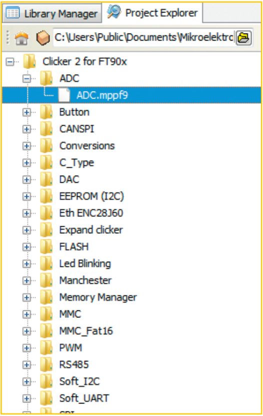
☐ Heap
Size: 2000

Configuration Registers

```
CLKCFG  :  $00010008  :  0x00000000
EFUSE   :  $00000000  :  0x0C1F0000
```

9. More examples

mikroPascal PRO for FT90x comes with over **300 examples** which demonstrate a variety of features. 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. All projects are delivered with working .HEX files, so you don't have to buy a compiler license in order to test them.



```
Clicker 2 for FT90x
    ADC
        ADC.mppf9
    Button
    CANSPI
    Conversions
    C_Type
    DAC
    EEPROM (I2C)
    Eth ENC28J60
    Expand clicker
    FLASH
    Led Blinking
    Manchester
    Memory Manager
    MMC
    MMC_Fat16
    PWM
    RS485
    Soft_I2C
    Soft_UART
```

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™, EasyFT90x™ v7, mikromedia, clicker 2™, mikroBUS™, click™ boards 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.

mikroPascal PRO for FT90x

If you want to learn more about our products, please visit our web site 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
2015.

CTFP in mikroPascal PRO
for FT90x bundle manual
ver 1.01