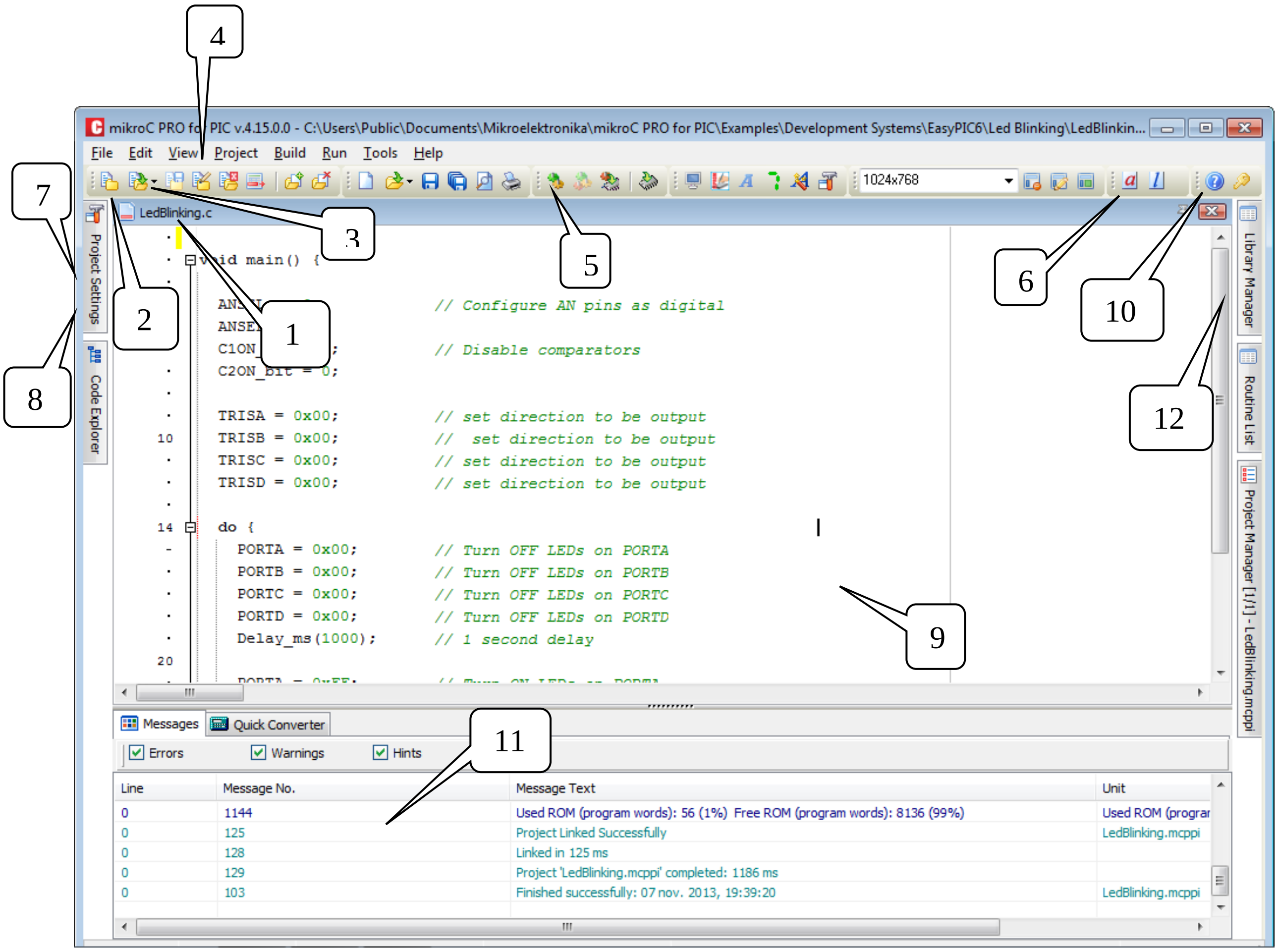
FIRST STEPS WITH THE PIC 16F84A TROUGH EXAMPLES IN MikroC and Proteus

Introduction

Microcontrollers in general and the PICs in particular are components whose real implementation requires programming. The basic programming language of these components is the **assembly language**. Many difficulties related to the programming with this language (the mastering of the instruction set, the length of the source codes, the interpretation not obvious of the instructions…) led to the setting-up of the high-level languages, closer to the human language and easier to handle and understand. Among them, we enumerate MikroC and CCS (for a programming in of C language), PROTON (BASIC), MikroPascal (Pascal)…In this small tutorial, we will learn through examples how to program and simulate a microcontroller with MikroC and Proteus.

Presentation of the interface

We present here the buttons mostly used in this software and their functions.


4
mikroC PRO for PIC v.4.15.0.0 - C:\Users\Public\Documents\Mikroelektronika\mikroC PRO for PIC\Examples\Development Systems\EasyPIC6\Led Blinking\LedBlinkin...
File Edit View Project Build Run Tools Help
1024x768
7
3
LedBlinking.c
5
6
10
2
1
8
12
Project Settings
Code Explorer
Library Manager
Routine List
Project Manager [1/1] - LedBlinking.mcppi
9
Messages
Quick Converter
11

```
void main() {

  ANS                    // Configure AN pins as digital
  ANSE
  C1ON_        ;         // Disable comparators
  C2ON_bit = 0;

  TRISA = 0x00;          // set direction to be output
  TRISB = 0x00;          //  set direction to be output
  TRISC = 0x00;          // set direction to be output
  TRISD = 0x00;          // set direction to be output

  do {
    PORTA = 0x00;        // Turn OFF LEDs on PORTA
    PORTB = 0x00;        // Turn OFF LEDs on PORTB
    PORTC = 0x00;        // Turn OFF LEDs on PORTC
    PORTD = 0x00;        // Turn OFF LEDs on PORTD
    Delay_ms(1000);      // 1 second delay

    PORTA = 0xFF;        // Turn ON LEDs on PORTA
```

☑ Errors ☑ Warnings ☑ Hints

Line	Message No.	Message Text	Unit
0	1144	Used ROM (program words): 56 (1%) Free ROM (program words): 8136 (99%)	Used ROM (prograr
0	125	Project Linked Successfully	LedBlinking.mcppi
0	128	Linked in 125 ms	
0	129	Project 'LedBlinking.mcppi' completed: 1186 ms	
0	103	Finished successfully: 07 nov. 2013, 19:39:20	LedBlinking.mcppi

1 Name of the current file or program

2 **` New Project':** allows to create a new project

3 **` Open Project':** allows to open an already existing project

4 **` Project' Edict:** allows modifying the basic parameters of the project (the PIC name, the name of the project, the frequency of the PIC actually programmed)

5 **` Build Project':** allows to build the project and to generate the file of extension .HEX. It is the file, which is inserted in the PIC for a real implementation. It is this file which will be useful to us in simulation

6 **` View Assembly':** allows to see the assembly version of the program written

7 Shows the name of the PIC we are programming. Ex 16F84A

8 Shows the frequency of the clock to be used during the realization

9 Represents the zone of edition of the source code

10 **` Help':** allows to enter the help of the software MikroC

11 Presents the errors and the warnings relating to the current program. The posted messages give the nature of the error and its position in the source code or say if the program is well typed

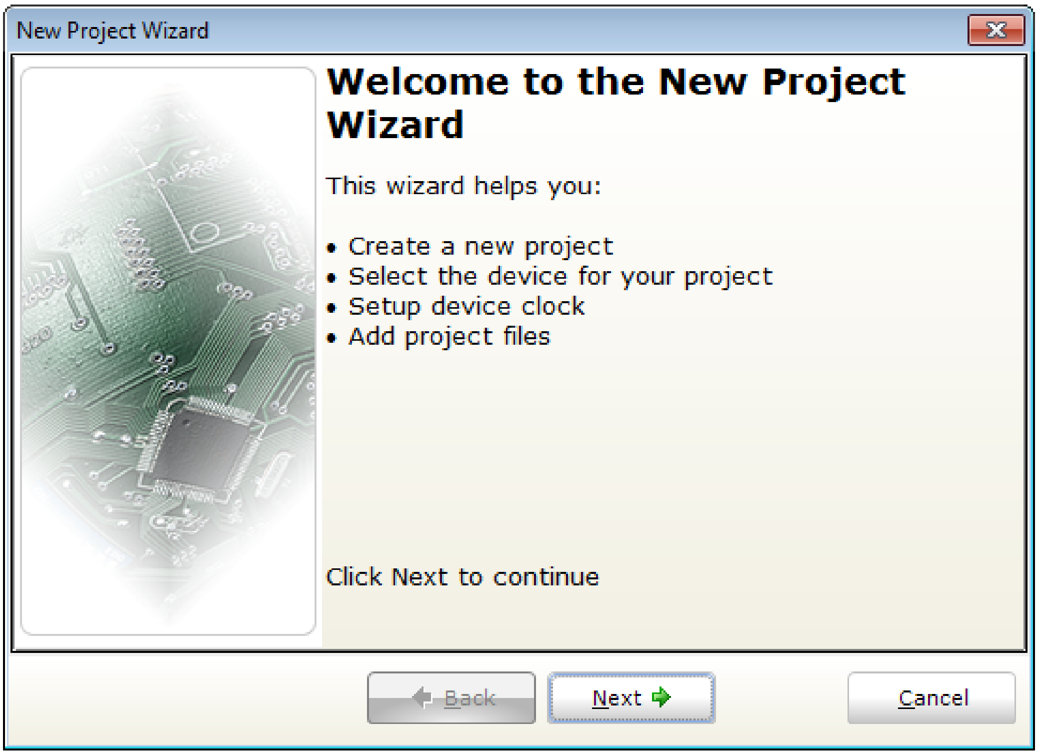
12 The library manager allows to activate or deactivate some MiKroC's libraries

Creation of a new project

During the creation of a new project, a certain number of choices must be operated:

- The name of the project
- The path (it is preferable to create a file in advance where all the files of the project will be safeguarded)
- Name of the PIC to be programmed
- The choice of its frequency of clock
- The activation of certain additional parameters such as the Watchdog, the protection of code
- The type of oscillator (with quartz, RC circuit …)
- And others.

When we click on the button **` New Project' (2),** the following pages appear and all the parameters mentioned above can be located and adjusted.



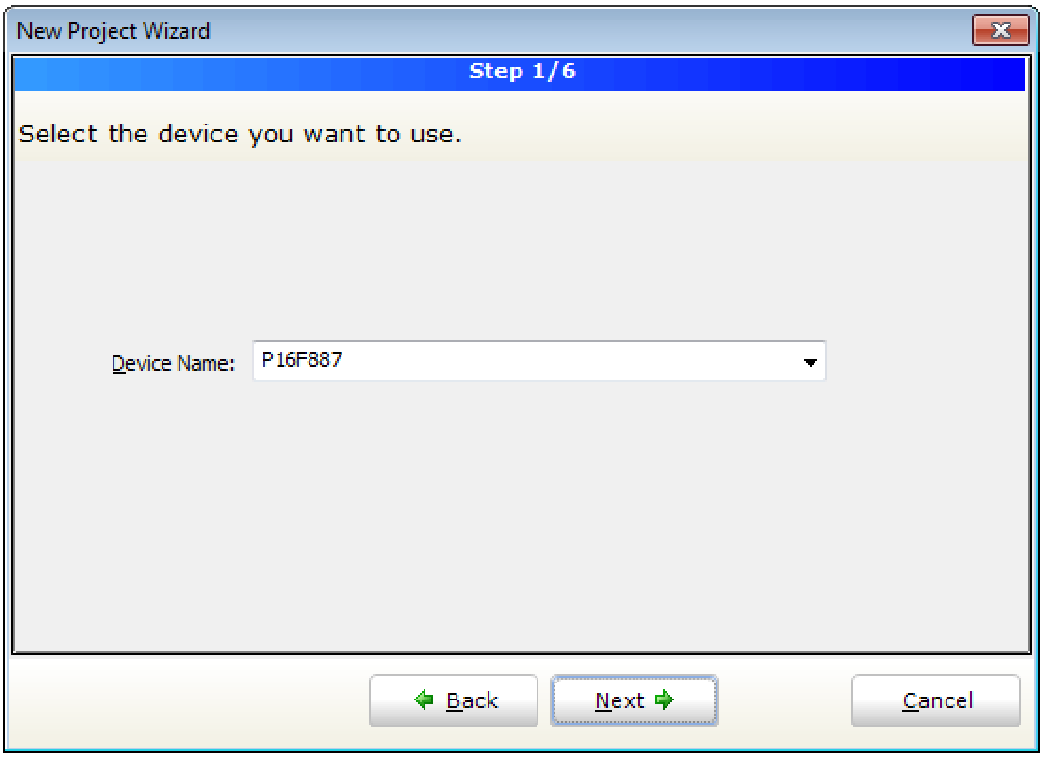
Welcome to the New Project Wizard

This wizard helps you:

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

Click Next to continue

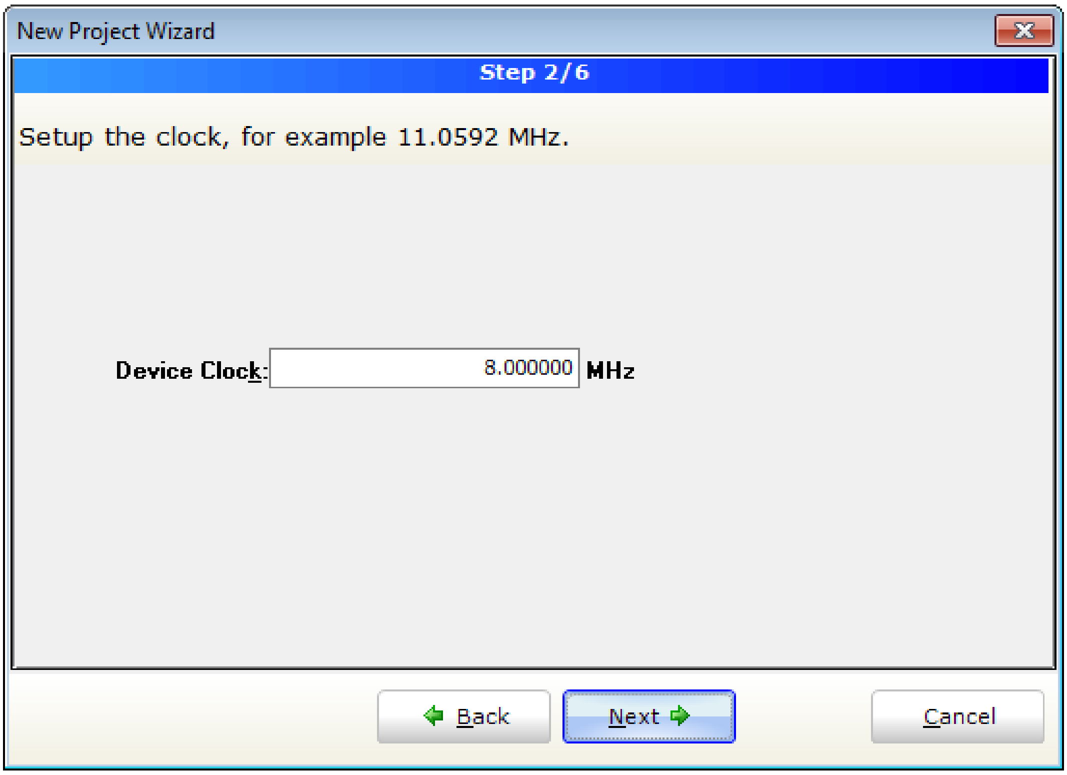
Click on ‘**Next**’



Select the device you want to use.

Device Name: P16F887

Select the microcontroller and Click on ‘**Next**’



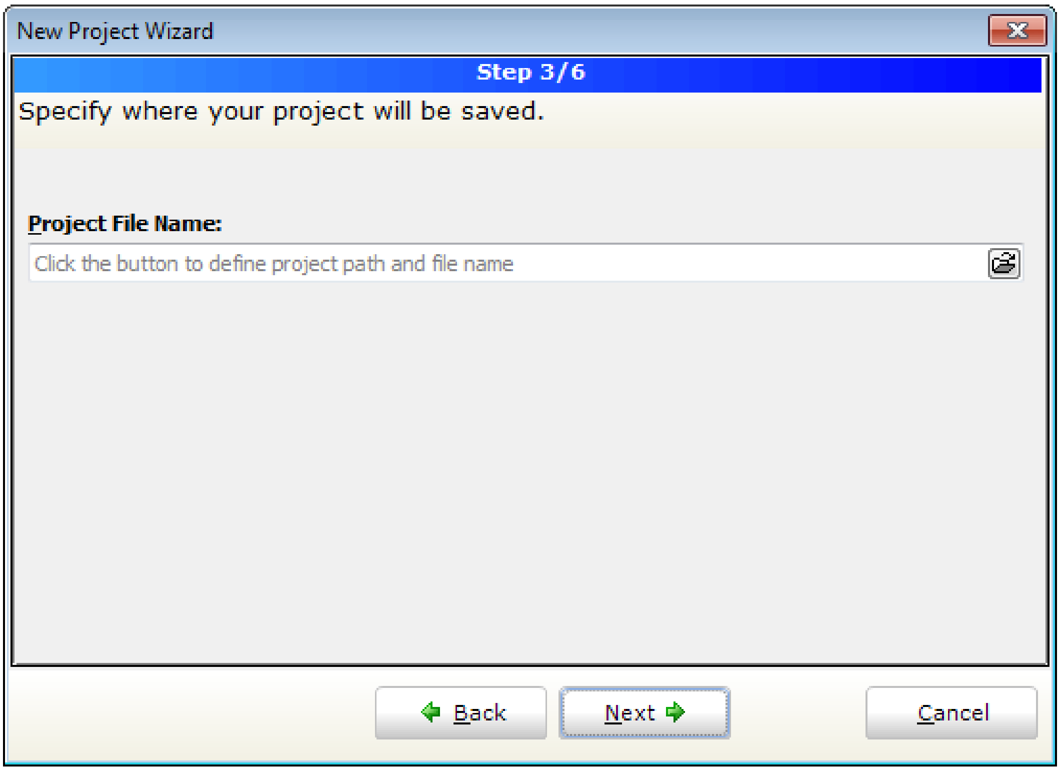
New Project Wizard

Step 2/6

Setup the clock, for example 11.0592 MHz.

Device Clock: 8.000000 MHz

Select the frequency and Click on ‘**Next**’



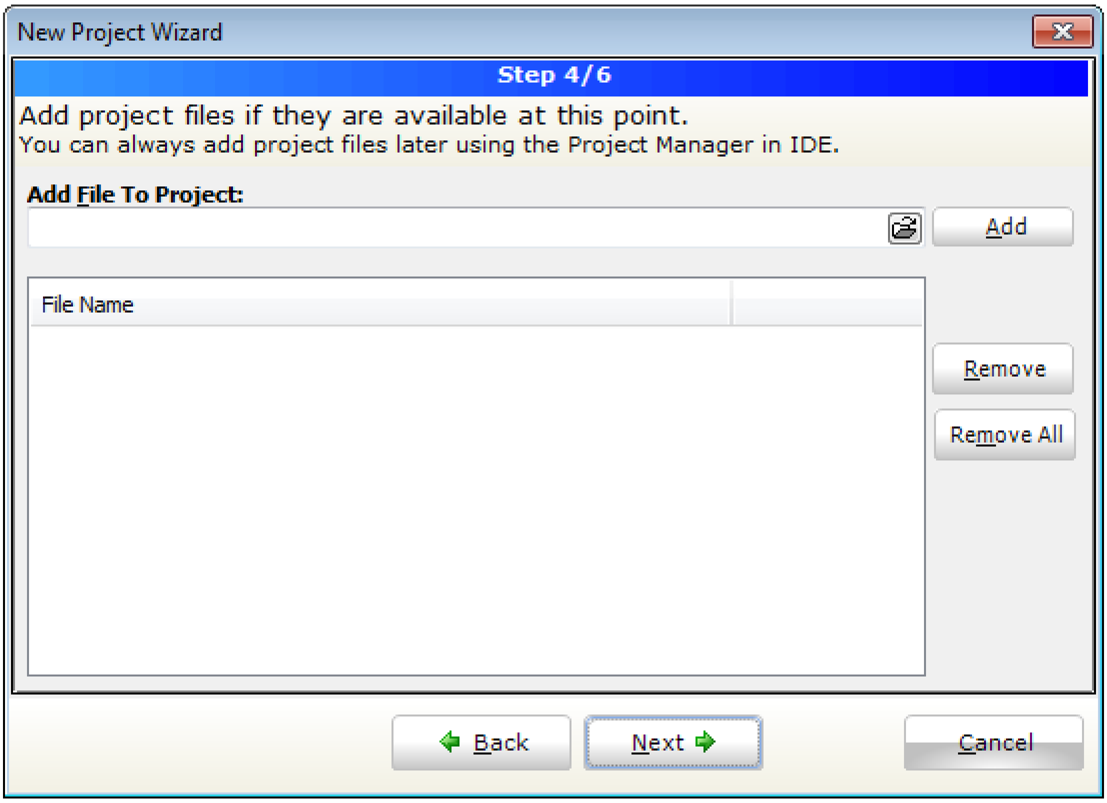
New Project Wizard

Step 3/6

Specify where your project will be saved.

Project File Name: Click the button to define project path and file name

Select the path and give the name of the project



New Project Wizard

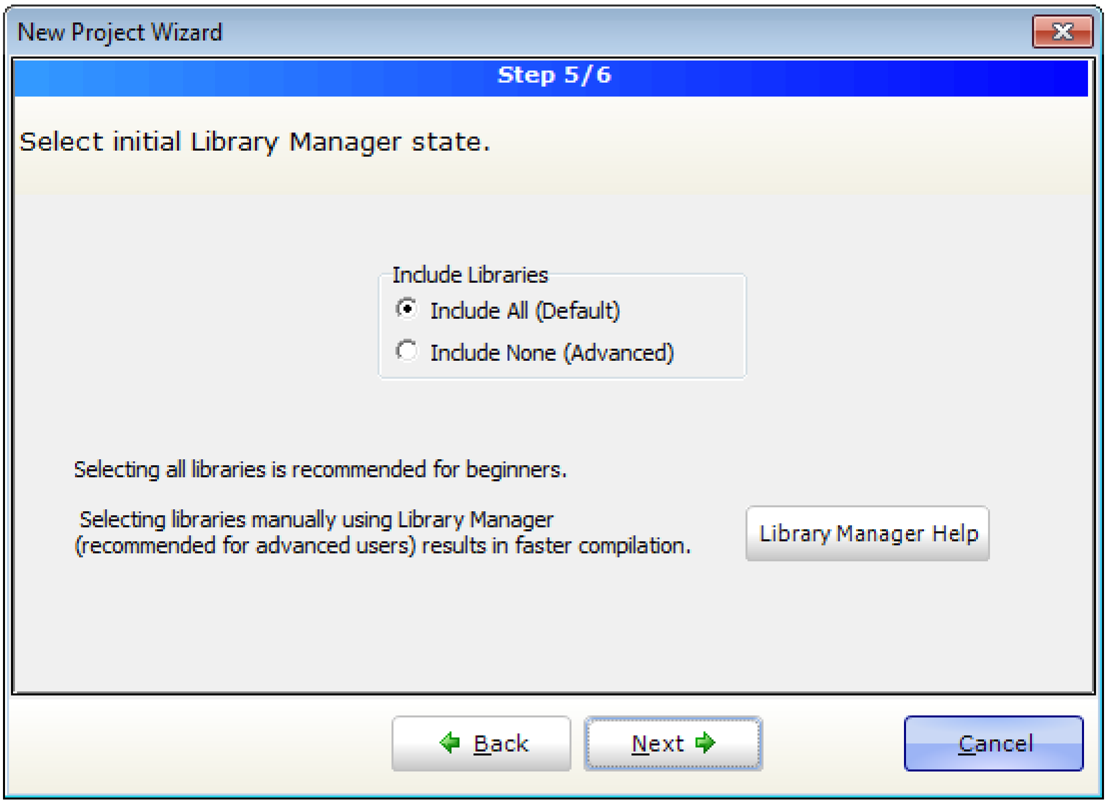
Step 4/6

Add project files if they are available at this point.
You can always add project files later using the Project Manager in IDE.

Add File To Project:

File Name

Click on ‘**Next**’



New Project Wizard

Step 5/6

Select initial Library Manager state.

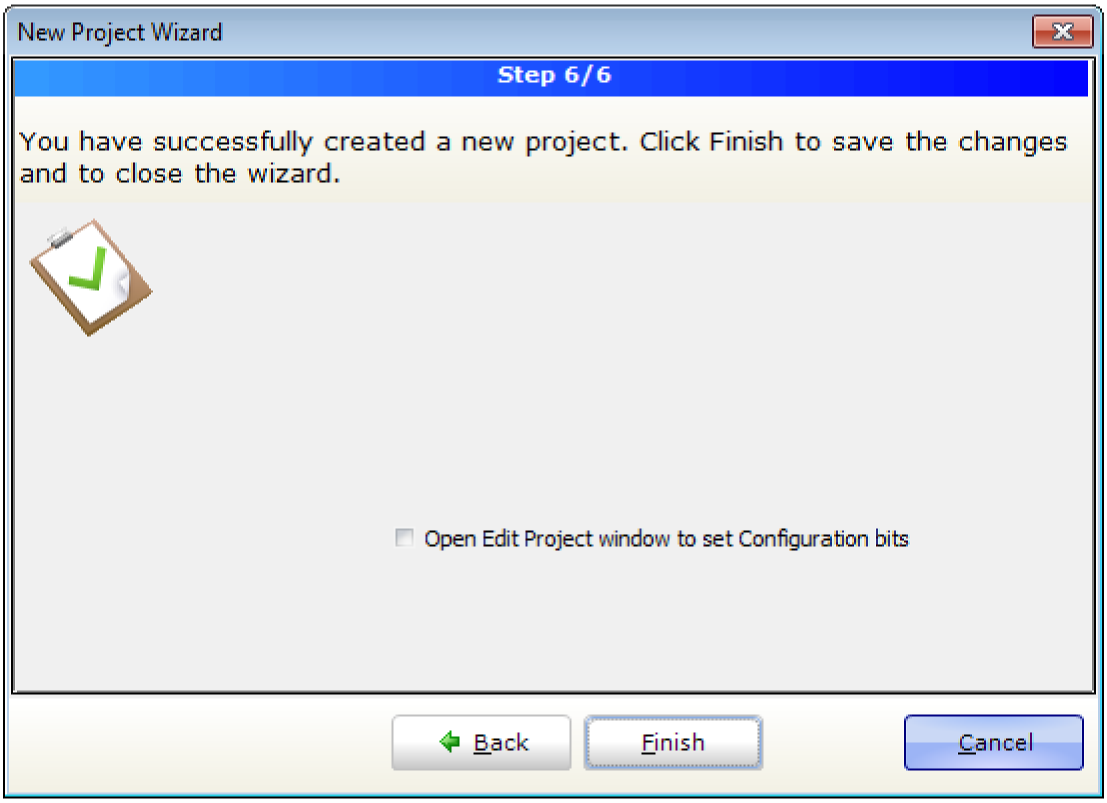
Include Libraries

● 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.



You have successfully created a new project. Click Finish to save the changes and to close the wizard.

□ Open Edit Project window to set Configuration bits

When all is set according to our own needs, click on 'Finish' to validate. One returns in the editing page of MikroC. We can now start typing our source code.