mikroICD™

in-circuit debugger

Whether you are a beginner, or a professional, this powerful tool, with intuitive interface and convenient set of commands will enable you to track down bugs quickly. mikroICD™ is one of the fastest, and most reliable debugging tools on the market.

Exploit the full potential of real-time hardware debugging

Start Debugger [F9]

Run/Pause Debugger [F6]

Stop Debugger [Ctrl+F2]

Step into [F7]

Step Over [F8]

Step Out [Ctrl+F8]

Run To Cursor [F4]

Toggle Breakpoint [F5]

Show/Hide Breakpoints [Shift+F4]

Clear breakpoints [Ctrl+Shift+F5]

Jump To Interrupt [F2]

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

Introduction to mikroICD™

What is mikroICD™?

mikroICD™ is a highly effective tool for a Real-Time debugging on hardware level. The mikroICD™ debugger enables you to execute your program on the host microcontroller and view variable values, Special Function Registers (SFR), RAM, CODE and EEPROM memory along with the mikroICD™ code execution on hardware. In order to use **mikroICD™** it is necessary to have the appropriate hardware (**mikroProg™ for PIC®, dsPIC® and PIC32®**) and software (Mikroelektronika compilers for PIC®, dsPIC® or PIC32®).

Key features

Supported in all Mikroelektronika hardware programers for PIC®, dsPIC® and PIC32® (**mikroProg™ for PIC®, dsPIC® and PIC32®**)

Supported in all MikroElektronika compilers for **PIC®**, **dsPIC®** and **PIC32®** (**mikroC™**, **mikroBasic™** and **mikroPascal™**)

Real time step by step debugging
Can monitor **SFR**, **RAM**, **CODE** and **EEPROM** memory

Hardware and software

mikroProg™ programmer

mikroICD™ is included with on-board programmers on mikroElektronika development systems and on **mikroProg™ for PIC®, dsPIC® and PIC32®** stand alone programmer. Microcontroller on target device is connected with programmer via PGC, PGD and MCLR pins. These pins are used for programing purposes and cannot be used as I/O while mikroICD™ is in use. Before using mikroICD™ it is necessary to **program** target microcontroller with a debug-enabled version of your output HEX file.

Compilers

All MikroElektronika compilers (**mikroC™**, **mikroBasic™** and **mikroPascal™**) for **PIC®**, **dsPIC®** and **PIC32®** natively support **mikroICD™**. Specialized **mikroICD™ DLL** module allows compilers to exploit the full potential of fast hardware debugging. Along with compilers, make sure to install the appropriate programmer drivers and **mikroProg Suite™ for PIC®** programming software.

1. Starting compiler

After the appropriate software and hardware is installed and attached to your PC it's time to start the chosen compiler.

mikroC PRO for PIC icon

In this manual we will use **mikroC PRO for PIC** compiler. All other compilers (mikroBasic™ and mikroPascal™ for PIC®, dsPIC® and PIC32®) have the same IDE so using **mikroICD™** is the same for all.

After the compiler is started write a new project or open the existing one.

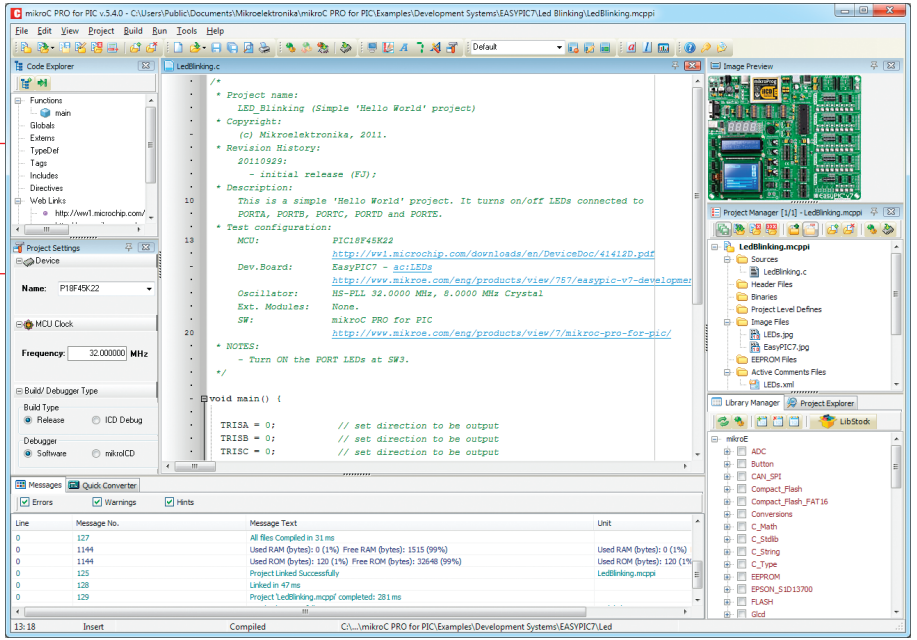


Figure 1-1: mikroC PRO for PIC window

2. Preparing mikroICD™

In order to use mikroICD™, you have to program your microcontroller with debug-enabled .HEX file of your project. This is done in two simple steps:

Step 1 - Enable mikroICD™

Under **Project Settings** - **Build /Debugger Type**, select **ICD Debug** and **mikroICD™** options.

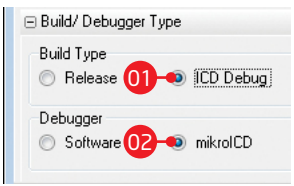


Figure 2-1: Build/Debugger Type options

01 Select **ICD Debug** option to create debug output HEX file.

02 Select **mikroICD™ option** to enable usage of mikroICD™ debugger for debugging

Step 2 - Build your project

Next step is to build your project and to program it to MCU memory. To do that click on **Build > Build + Program [Ctrl+F11]** option or click on  icon in the build toolbar. Compiler will automatically build the program and start mikroProg Suite™ for PIC® software which will program the code into microcontroller.

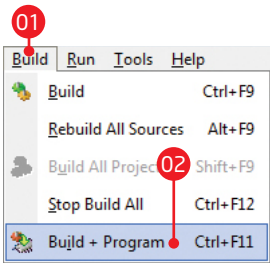


Figure 2-2: Build menu

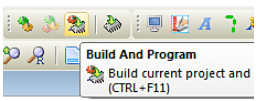


Figure 2-3: Build toolbar

01 Activate **Build** Menu

02 From drop down menu select **Build + Program** option or press **[Ctrl+F11]** on your keyboard

3. Start mikroICD™ debugging

To start **mikroICD™** debugging open the **Run** menu and click the **Start Debugger [F9]** option or  icon from Run toolbar.

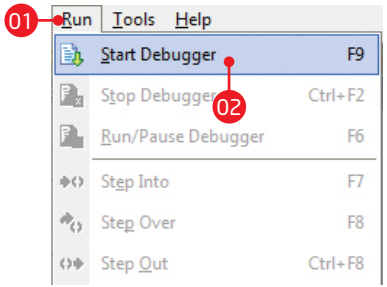


Figure 3-1: Run menu

01 Click the **Run** option

02 From drop down menu select **Start Debugger** option or press **[F9]** on keyboard

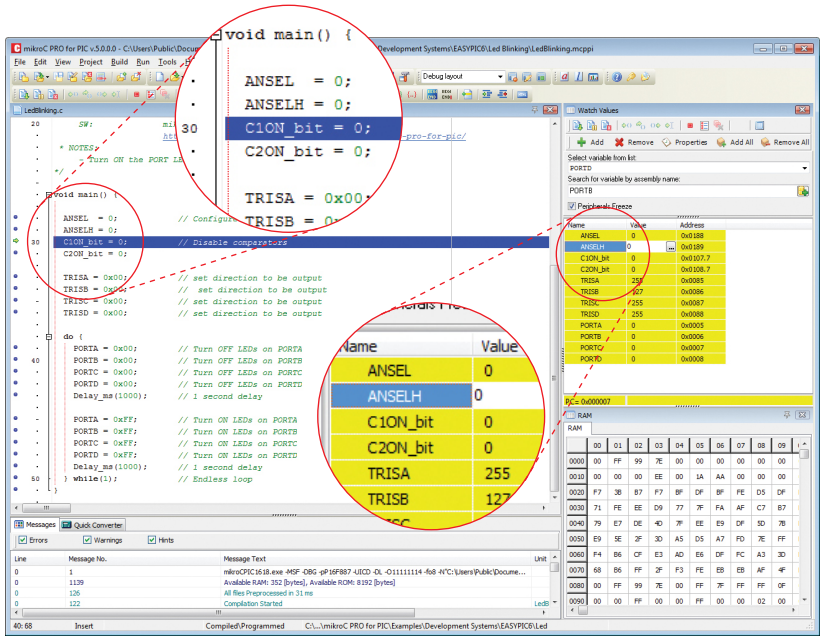


Figure 3-2: mikroC window during debugging

note *Make sure to* **enable power supply** *on your device*

When **mikroICD™** debugging is started a program line which will be next executed is highlighted with a blue strip.

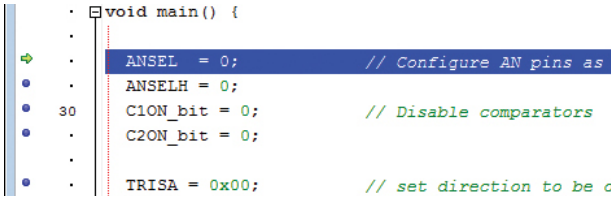


Figure 3-3: Execution line is highlighted

Next step is to select values which will be monitored. Click on **View** -> **Debug Windows** -> **Watch Window [Shift+F5]** or click the icon to open **Watch Values** window.

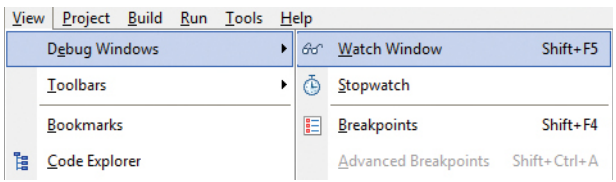


Figure 3-4: Open watch values window

Within **Watch Values** window you can set which registers or variables are going be monitored.

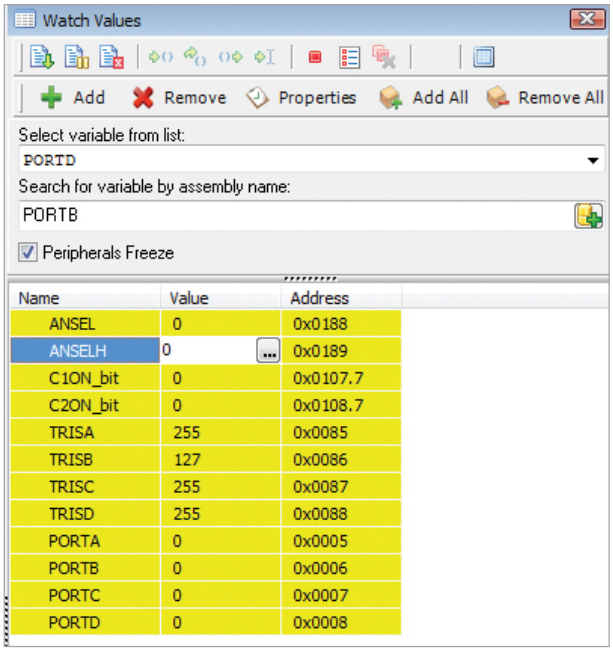


Figure 3-5: Watch values window

4. Watch Window

What are the Watch Variables?

Each special function register (SFR), and user defined variables which are not removed by the optimizer, can be monitored in **Watch Window** during the debugging process. With execution of each program line, values of selected variables are automatically updated. Watch Window also provides the information about the **memory address** and the full **assembly name** of each variable.

Types of variables

Purple colored variables are special **function registers and sbit variables** from the definition file of the selected microcontroller. Black colored variables represent **user defined variables**, or variables used internally by compiler libraries.

Adding Watch Variables

Adding variable for monitoring can be done in two simple ways: by selecting the variable from the drop down list, or by searching for the desired variable using the search box.

Step 1 - Select variable

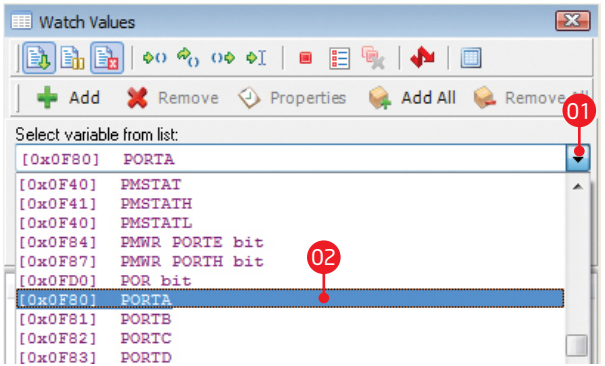


Figure 4-1: Select variable for monitoring

01 Click to show drop down menu.

02 Select desired variable for monitoring

Step 2 - Add variable

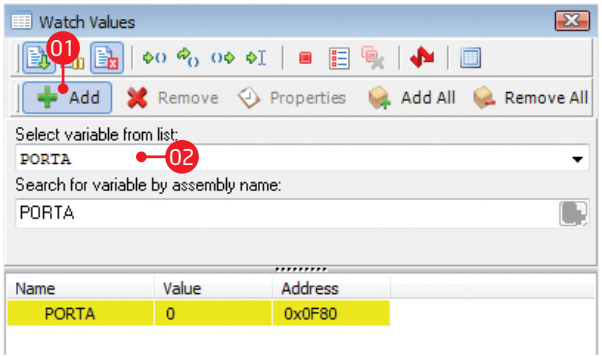


Figure 4-2: Add variable for monitoring

01 Click the **Add** button and selected variable will be added to list

02 List with selected variables

Instead of selecting variable from list you can type in variable assembly name in the search box.

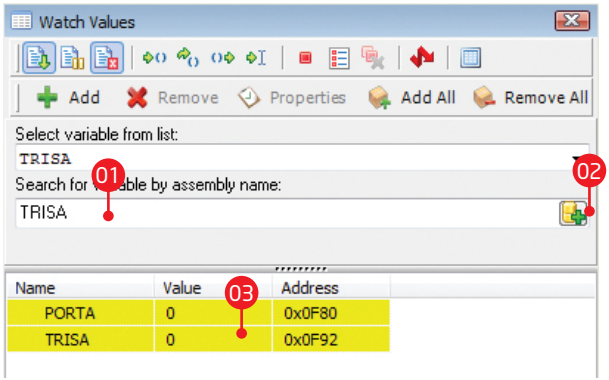


Figure 4-3: Search for variable

01 Type in variable assembly name

02 Click on add variable button

03 Variable is added to list

5. Debugger toolbar

To simplify debugging compiler IDE contains toolbar with icons that allow single click access to **mikroICD™** commands.

Figure 5-1: Debugger toolbar

Debug commands

The first three icons on the toolbar are used for starting/stopping debugger:

- Start debugger [F9]
- Run/Pause Debugger [F6]
- Stop Debugger [Ctrl + F2]

Execution commands

Next set of icons enables you to execute program in real time:

- Step Into [F7]
- Step Over [F8]
- Step Out [Ctrl + F8]
- Run To Cursor [F4]

Managing breakpoints

Last set of icons is related to breakpoints and interrupt option:

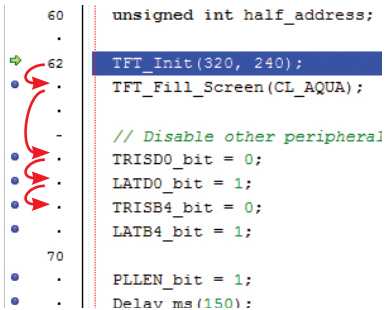
- Toggle Breakpoint [F5]
- Show/Hide breakpoints [Shift+F4]
- Clears breakpoints [Shift+Ctrl+F5]
- Jump to interrupt [F2]

Toolbar Icon	Name	Shortcut	Description
	Start Debugger	[F9]	Starts Debugger.
	Run/Pause Debugger	[F6]	Run/Pause Debugger.
	Stop Debugger	[Ctrl + F2]	Stops Debugger.
	Step Into	[F7]	Executes the current program line, then halts. If the executed program line calls another routine, the debugger steps into the routine and halts after executing the first instruction within it.
	Step Over	[F8]	Executes the current program line, then halts. If the executed program line calls another routine, the debugger will not step into it. The whole routine will be executed and the debugger halts at the first instruction following the call.
	Step Out	[Ctrl + F8]	Executes all remaining program lines within the subroutine. The debugger halts immediately upon exiting the subroutine.
	Run To Cursor	[F4]	Executes the program until reaching the cursor position.
	Toggle Breakpoint	[F5]	Toggle breakpoints option sets new breakpoints or removes those already set at the current cursor position.
	Show/Hide breakpoints	[Shift+F4]	Shows/Hides window with all breakpoints
	Clears breakpoints	[Shift+Ctrl+F5]	Deletes selected breakpoints
	Jump to interrupt	[F2]	Opens window with available interrupts (doesn't work in mikroICD™ mode)

6. Real-Time debugging

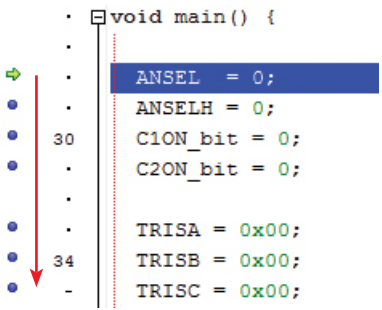
Real-Time debugging enables execution of program in three different ways:

Step by Step



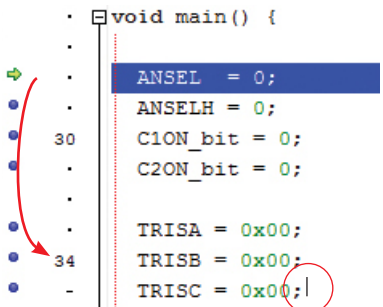
To execute program one line at the time you can use **Step Into** **[F7]** and **Step Over** **[F8]** options

Execute remaining lines



Execution of all remaining program lines is available via **Step Out** **[Ctrl+F8]** option. Debugging will stop when all lines in subroutine are executed.

Execute to cursor



In order to execute program starting from the current line to one where cursor is placed, use option **Run to Cursor** **[F4]**. Program will start execution at the current line (blue strip) and it will stop at line where cursor is placed.

7. Using Breakpoints

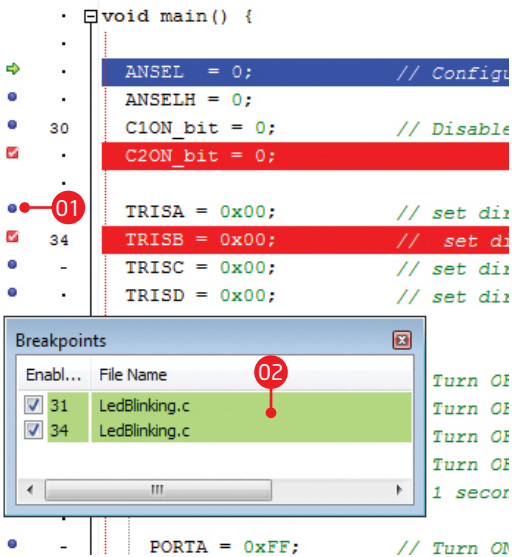


Figure 7-1: Breakpoint selection

01 Click on a blue dot to place a breakpoint

02 Lines 31 and 34 are enabled as breakpoints

The **mikroICD™** enables each program line to be marked with a **breakpoint**. The breakpoint is an intentional stopping or pausing place in the program used for the purpose of debugging.

Breakpoints are placed in the program by clicking the blue dots to the left of the program line or by pressing icon **[F5]**. By selecting the **Run** command icon **[F6]**, the microcontroller will execute the program from the current location (highlighted in blue) until it reaches a breakpoint (highlighted in red). The debugger halts after reaching the breakpoint.

Hardware and software breakpoints

There are two kinds of breakpoints - hardware and software breakpoints. The only visible difference between them is in the speed of program execution before it reaches the specified program line.

Hardware breakpoints are placed within the microcontroller chip and provide considerably faster program execution. The total number of software breakpoints goes up to 16, while the number of hardware breakpoints is much smaller. For example, PIC16® microcontrollers have only one, whereas PIC18® microcontrollers have up to 3 hardware breakpoints. When all hardware breakpoints are used, then remaining breakpoints in the program will be used as **software breakpoints.**

8. Advanced Breakpoints Option

The **mikroICD™** provides the means for using the **Advanced Breakpoints option** with PIC18®, PIC24® and dsPIC® microcontrollers. To enable it, check the Advanced Breakpoints check box within the Watch Values window. To configure the Advanced Breakpoints option it is necessary to start up mikroICD™ **[F9]** and select the **View › Debug Windows › Advanced Breakpoints** option or to use the keyboard shortcut **[Ctrl+Shift+A]**.

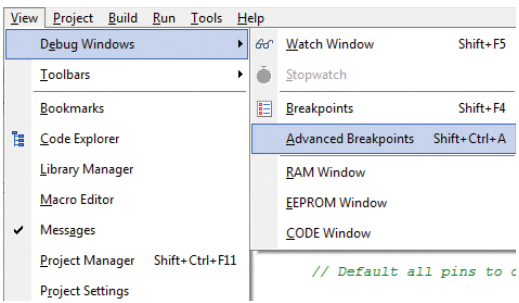


Figure 8-1: Advanced breakpoints menu

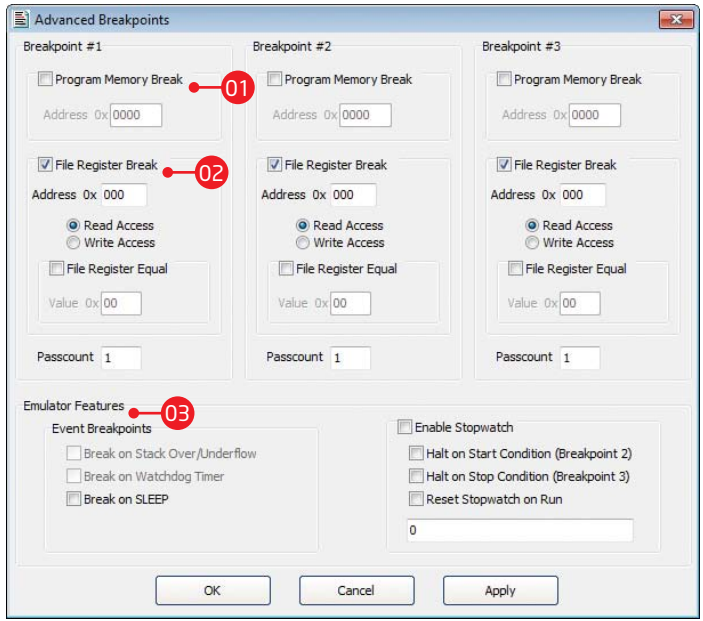


Figure 8-2: Advanced breakpoints window

01 The **Program Memory Break** option is used for placing breakpoints at specified addresses in the program memory. The value entered in the Address field must be in the HEX format.

02 The **File Register Break** option is used for stopping code execution when read/write access to the specified data memory location occurs. If the **Read Access** option is selected, the **File Register Equal** option can be used for setting the appropriate value in the **Value** field. The program execution will be stopped when the value read from the specified data memory location matches the value written in the Value field. All the values entered in the Value field must be in the HEX format.

03 **Emulator Features** enables the usage of **Event Breakpoints** and **Stopwatch**.

Event Breakpoints

Break on Stack Overflow/Underflow : not implemented.

Break on Watchdog Timer : not implemented.

Break on SLEEP : break on SLEEP instruction. SLEEP instruction will not be executed. If you choose to continue the mikroICD debugging [F6] then the program execution will start from the first instruction following the SLEEP instruction.

Enable Stopwatch

To use the Stopwatch define **Breakpoint#2** and **Breakpoint#3** as a Start and Stop conditions and check the Enable Stopwatch checkbox.

Halt on Start Condition (Breakpoint#2) : when checked, the program execution will stop on Breakpoint#2. Otherwise, Breakpoint#2 will be used only to start the Stopwatch.

Halt on Stop Condition (Breakpoint#3) : when checked, the program execution will stop on Breakpoint#3. Otherwise, Breakpoint#3 will be used only to stop the Stopwatch.

Reset Stopwatch on Run : when checked, the Stopwatch will be cleared before continuing program execution and the next counting will start from zero. Otherwise, the next counting will start from the previous Stopwatch value.

9. Disassembly view

During the process of compiling, each program line written in a high-level programming language is replaced with one or more assembly instructions. To display program in the assembly language, select the **View > Listing** option or click icon in toolbar. In this case, the process of simulating and debugging is performed in the same way as if the program is written in a high-level programming language. To toggle between high-level language and assembly language press **[Alt+D]** on your keyboard or click on icon.

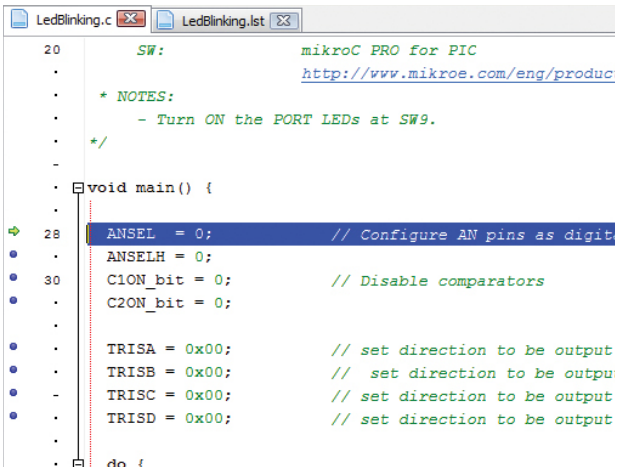


Figure 9-1: High-level programming language

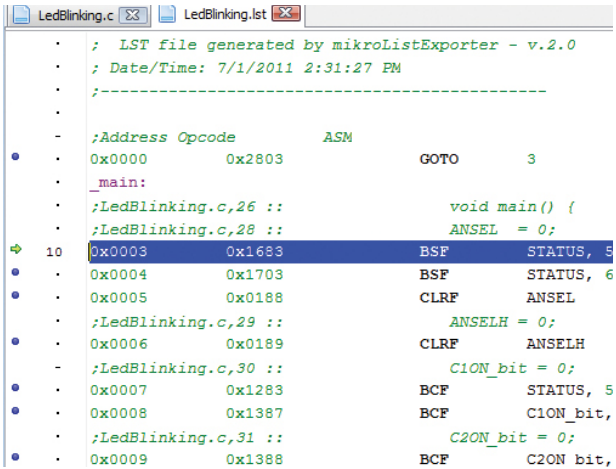
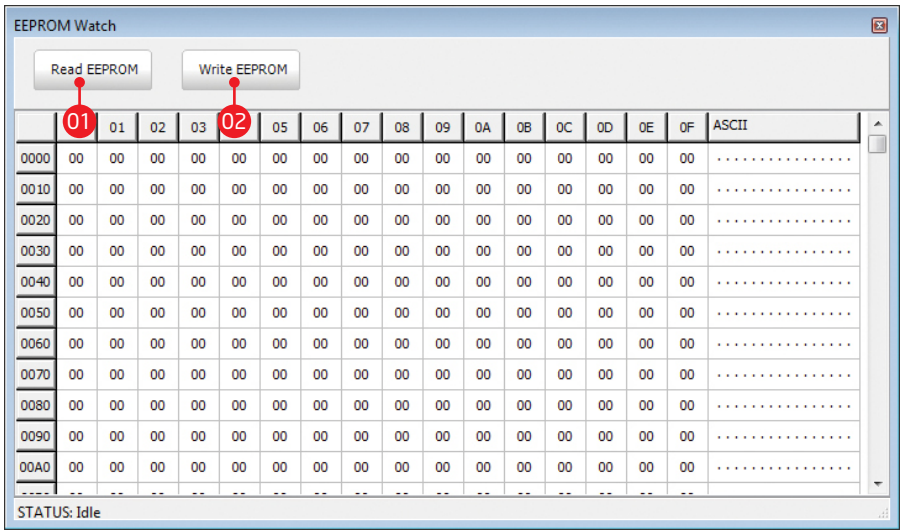


Figure 9-2: Assembly language

10. EEPROM Watch window

You can start **EEPROM Watch window** using **View > Debug Windows > EEPROM Window** option. It shows the values currently stored in the MCU internal EEPROM memory.



01 Click the **Read EEPROM** button to read the contents of microcontroller EEPROM memory which will be shown in the EEPROM Watch window.

02 Click the **Write EEPROM** button to program the data from the EEPROM Watch window into the internal EEPROM memory of the microcontroller.

Figure 10-1: EEPROM watch window

11. RAM window

The **mikroICD™** allows you to view the contents of the microcontroller's RAM memory using the **RAM window**. You can activate it by clicking the **View > Debug Windows > RAM Window** option.

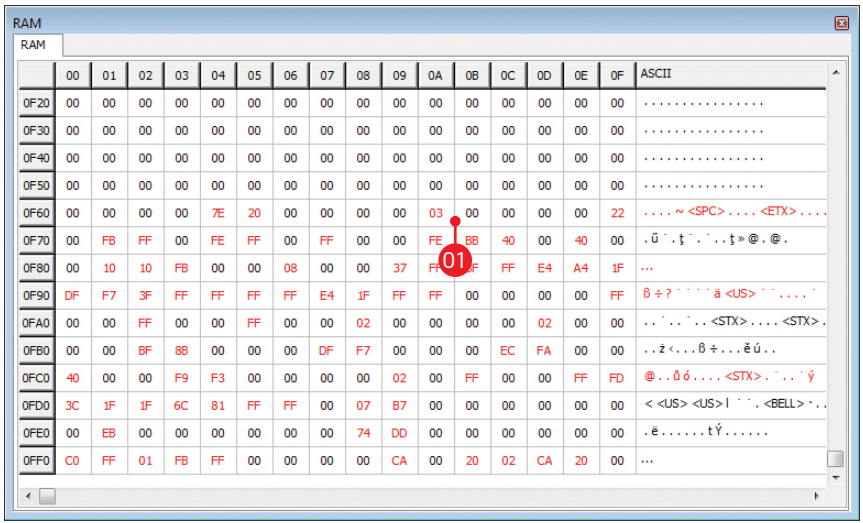


Figure 11-2: RAM window

Unlike the Watch Window option, all memory locations are displayed in a table. The content of each RAM location is displayed in the hexadecimal format and may be changed at any time during the operation of the microcontroller. Changed values are directly written in to the microcontroller by pressing Enter key.

01 In the table cell you can type in value in hexadecimal format. To write typed value into the MCU RAM memory press Enter key on your keyboard.

12. CODE Watch window

The **CODE Watch window** will appear by selecting the **View > Debug Windows > CODE Window** option. It shows the values currently stored in the MCU internal FLASH memory.

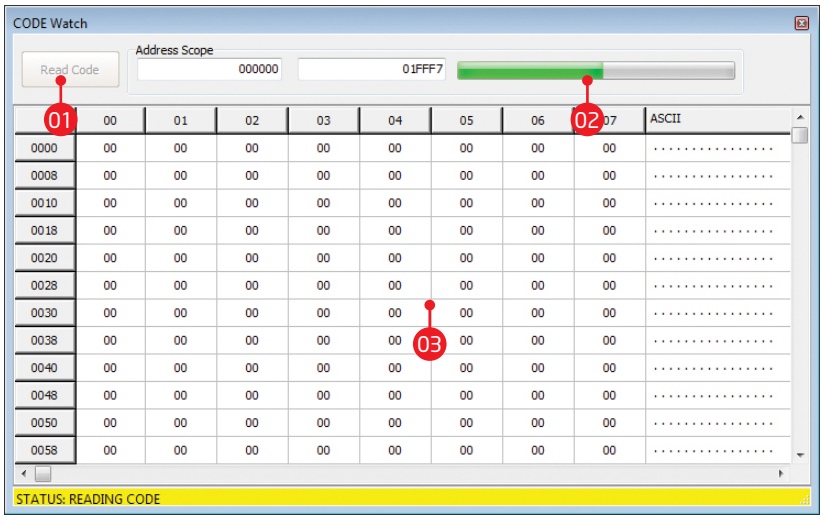


Figure 12-1: CODE Watch

01 Click the **Read Code** button to read content of MCU FLASH memory

02 **Progress bar** monitors code reading process

02 After code reading is finished you can preview it in the table

Notes:

DISCLAIMER

HIGH RISK ACTIVITIES

TRADEMARKS