mikroICD debugger is a highly effective tool for real-time debugging at hardware level. It enables you to view program variable values, Special Function Registers (SFRs) and EEPROM while the program is running. This manual contains practical example on how to create a new project, write and compile code and test the results.
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

1.0. mikroICD Overview............................................................................................................................ 4
   Hardware..................................................................................................................................... 4
   Software...................................................................................................................................... 5
2.0. Using mikroICD................................................................................................................................. 6
   Writing the Program and Setting up the Project for Debugging................................................... 6
   Compiling the Program and Dumping it into the Microcontroller............................................. 7
   Starting up the mikroICD Debugger............................................................................................ 8
3.0. Practical Example of Using mikroICD............................................................................................... 9
4.0. mikroICD Debugger Options......................................................................................................13
   Real-Time Debugging..................................................................................................................13
   Breakpoints.................................................................................................................................13
   Watch Window Option........................................................................................................... 14
   Advanced Breakpoints Option................................................................................................. 15
   View Assembly Option............................................................................................................... 16
   EEPROM Watch Window........................................................................................................ 16
   RAM Window............................................................................................................................. 17
1.0. mikroICD® Overview

The mikroICD (In-Circuit Debugger) is a hardware tool designed for testing and debugging programs on most PIC microcontrollers. It also enables you to monitor the state of all registers within the microcontroller which operates in real environment. In order for the mikroICD debugger to be used, it is necessary to have the appropriate hardware as well as to install additional software.

Hardware

The mikroICD is an integral part of the PICflash programmer intended for use with PIC16, PIC18, PIC24, dsPIC30 and dsPIC33 microcontrollers. It is built into all PIC® development systems designed by MikroElektronika such as EasyPIC6, EasyPIC5, BigPIC5, EasydsPIC4, LV 24-33A etc. Thanks to mikroICD support, the PICflash® programmer is a multifunctional device as it may be used for programming PIC microcontrollers as well as for debugging programs executed in real time. Besides, the PICflash programmer is also available as a stand-alone device used for programming chips built into (soldered on) the target device.

PIC microcontrollers are connected to the programmer through the PGC, PGD and MCLR pins. In case that such programmer is used for the programming only, its hardware will automatically break connection with these pins after loading the HEX code, thus enabling them to be used for other purposes.
In case that the PICflash programmer is also used for debugging (mikroICD is enabled), these pins will be used for communication with the PC and cannot be used for other purposes.
mikroICD Debugger

The process of testing and debugging programs in real environment is performed by monitoring the state of all registers within the microcontroller. The mikroICD debugger also offers functions such as running a program step by step (single stepping), pausing the program execution to examine the state of currently active registers using breakpoints, tracking the values of some variables etc. In this case the mikroICD debugger is connected to the PC all the time so that the PGC, PGD and MCLR/Vpp pins cannot be used for the operation of the target device.

Software

The mikroICD debugger needs the additional software to be installed on the PC for its operation. Such software includes:

**PICflash v7.02** (or later version) is a program used along with the PICflash programmer’s hardware. It enables you to select the microcontroller to be programmed and to set up its mode. You can download it for free from our website at [www.mikroe.com](http://www.mikroe.com).

**Drivers** necessary for the proper operation of the PICflash programmer enable communication between the PC and the PICflash programmer’s hardware.

**Compilers** are programs used for compiling programs written in high-level programming languages into executable file (HEX code). Here is a list of compilers providing the mikroICD support:

- mikroC PRO® 2009;
- mikroBasic PRO® 2009;
- mikroPascal PRO® 2009;
- mikroC® (dsPIC30/33 & PIC24);
- mikroBasic® (dsPIC30/33 & PIC24); and
- mikroPascal® (dsPIC30/33 & PIC24).

The compilers’ demo versions can be downloaded for free from our website at [www.mikroe.com](http://www.mikroe.com)
2.0. Using mikroICD

The mikroICD debugger comes with all PIC and dsPIC compilers designed by MikroElektronika. This manual illustrates and describes its operation in the mikroC PRO for PIC compiler. The principle of the operation is the same for mikroBasic and mikroPascal compilers as well.

Step 1: Writing the Program and Setting up the Project for Debugging

Creating a new project and writing a program in the compiler’s main window should be done first. The next step is to set up the project for debugging using the mikroICD debugger. To perform this, it is necessary to select the following options in the Project Settings window:
Step 2: Compiling the Program and Dumping It into the Microcontroller

The program has to be compiled into the machine code before it is downloaded into the microcontroller. In order to start the process of compiling, click one of the appropriate shortcut icons or select the following option from the compiler’s Project drop-down menu:

Build+Program [Ctrl+F11]

By clicking on this command, the PICflash programmer will be automatically activated after completing the process of compiling and the compiled program (HEX code) will be immediately loaded into the microcontroller’s program memory. The programming progress will be shown in the PICflash programmer’s window to appear on the screen:

NOTE: In addition to the aforementioned build option which causes the program to be automatically compiled and loaded into the microcontroller memory, there are two other build options in the Project drop-down menu:

Build [Ctrl+F9] If the project consists of one file; and
Build All [Shift+F9] If the project consists of several files;

These options are intended for compiling only and do not start up the programming process. Accordingly, when these are used, the HEX code has to be loaded into the microcontroller from within the PICflash program using the Load and Write options. More information on MCU programming using the PICflash program may be found in the PICFlash Programmer manual.
Step 3: Starting up the mikroICD Debugger

After the microcontroller has been successfully programmed, it is time to start up the mikroICD by selecting the Start Debugger option from the Run drop-down menu.

As mentioned before, the mikroICD debugger enables you to directly monitor the state of all registers within the microcontroller. Some of the most frequently used debugger options are: Step Into, Step Over, Run to Cursor and Step Out. For the Watch Values window to appear on the screen, select the View > Debug Windows > Watch Window option.

The Watch Values window showing the state of the microcontroller’s registers and program variables

Icon commands
- Click on some of these options to add/remove selected registers on the list
- A complete list of registers within the programmed microcontroller
- A list of selected registers to be monitored. The state of these registers change during the program execution, which can be viewed in this window
- Double click on the Value field enables you to change data format
3.0. Practical Example of Using mikroICD

Here is a step-by-step illustration of the operation of the mikroICD:

Step 1: Writing the Program and Setting up the Project for ICD Debugging

/* Here is a simple program to demonstrate the operation of the microcontroller. The PORTC port’s pins are configured as digital outputs and their logic state changes once per second. Establishing connection between such port and LEDs will cause LEDs to blink simultaneously */

void main() {
    TRISC = 0x00; // Configure PORTC pins as outputs
    PORTC = 0x00; // Turn OFF LEDs on PORTC

    do {
        PORTC = 0xFF; // Turn ON LEDs on PORTC
        Delay_ms(1000); // 1 second delay

        PORTC = 0x00; // Turn OFF LEDs on PORTC
        Delay_ms(1000); // 1 second delay
    } while(1); // Endless loop
}

When the program has been written, it is necessary to select the appropriate debugging mode before it is compiled into the HEX code, in order to perform debugging using the mikroICD debugger.

First, to debug program, select the ICD Debug option from the Project Settings window.

Then, select the mikroICD option to enable the mikroICD debugger to be used for debugging.
Step 2: Compiling the Program and Dumping It into the Microcontroller

In order to compile the program into the HEX code and automatically dump it into the microcontroller, select the Build+program option [Ctrl+F11] from the Project drop-down menu.

Immediately after completing the compiling process, the PICflash programmer’s window will appear on the screen. In the right bottom corner thereof, there is the Progress bar showing the programming progress. If the Tools > Options > Tools > PICflash Options > Close when finished option is ticked off the PIC flash programmer’s window will be automatically closed after programming.
Step 3: Starting up mikroICD and Line-by-Line Program Execution

When the program is loaded into the microcontroller, its execution in real time can be monitored by using the mikroICD debugger. To start it up, select the Start Debugger option from the Run drop-down menu or click the [F9] button.

Now, the program within the microcontroller will be executed line by line by pressing the [F8] button.

During program execution, the program line to be next executed is highlighted in blue by default.

Only two registers are selected here to be monitored. Use the Add All option to add all registers within the microcontroller to the list of selected registers to be monitored.
The Watch Values window enables you to monitor the state of selected registers and to view how their states change during program execution.

In this example, the first instruction is executed using the Step Over option. In higher programming languages such option executes the whole program line regardless of how many assembly instructions it consists of.

By executing the same instruction (Step Over [F8]) two more times, the 32nd program line which contains the Delay_ms(1000) command will be reached. To perform its execution, it is advisable to use the Run to Cursor [F4] option as it executes the program at full speed.

The Run to Cursor command will cause the program to be executed at full speed until it encounters the program line with cursor placed.
4.0. mikroICD Debugger Options for Advanced Users

The following text describes the advanced options offered by the mikroICD debugger.

Real-Time Debugging

The Step Into [F7] and Step Over [F8] commands enable the program to be executed line by line. Program execution is a slow process in this case, and as such, is suitable for short programs. Unlike them, the Run/Pause Debugger [F6] and Run To Cursor [F4] commands enable the program to be executed in real time and therefore be much faster. The speed of program execution depends on the microcontroller's own clock. By pressing [F6] or selecting the Run/Pause Debugger option, the mikroICD is temporarily halted and the microcontroller executes the loaded program at full speed. Another pressure on the same button reactivates the mikroICD and the program execution stops at reached location. By pressing [F4], the microcontroller will proceed with program execution at high speed until it reaches the line selected by the cursor.

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 space to the left of the program line or by pressing [F5]. By selecting the Run command [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.

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 number of hardware breakpoints is limited, whereas the total number of software breakpoints is unlimited. For example, 16-bit PIC microcontrollers have only one, whereas 18-bit PIC 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.
Watch Window Option

The Watch Window option allows you to monitor the values of program variables as well as the contents of SFRs while the program is running. As soon as the program is loaded into the microcontroller, the Watch Values window appears on the screen. To reopen this window, when removed, select the option View > Debug Windows > Watch Window.

The Watch Values window displays data in three columns: register or variable names, their values and memory addresses. Double click on any variable opens the Edit Value window which allows you to assign it a new value. It is also possible to change data format (decimal, hexadecimal, binary, floating or character) in this window.
Advanced Breakpoints Option

The mikroICD provides the means for using the Advanced Breakpoints option with PIC18 and PIC18FJ microcontrollers. To enable it, tick the Advanced Breakpoints checkbox 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 [Ctrl+Shift+A] shortcut icon.

![Enhanced Breakpoints Page](image)

Program Memory Break Option

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.

File Register Break Option

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.

When the Advanced Breakpoints option is enabled, mikroICD operates in real-time mode, thus supporting only the following set of commands: Start Debugger [F9], Run/Pause Debugger [F6] and Stop Debugger [Ctrl+F2]. After reaching the first breakpoint, the Advanced Breakpoints option can be disabled and the process of debugging can be continued with a full set of commands. The number of advanced breakpoints is equal to the number of hardware breakpoints and depends on the microcontroller in use.
View Assembly Option

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 Assembly option from the Project drop-down menu. 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.

EEPROM Watch Window

The EEPROM Watch window will appear by selecting the View > Debug Windows > EEPROM Window option. It shows the values currently stored in the PIC internal EEPROM memory.

Pressing the Write EEPROM button causes data from the EEPROM Watch window to be loaded into internal EEPROM memory of the microcontroller.

Pressing the Read EEPROM button causes the contents of the EEPROM memory to be read and shown in the EEPROM Watch window.
RAM Window

The mikroICD allows you to view the contents of the microcontroller’s RAM memory in the RAM window by clicking the View > Debug Windows > RAM Window option. 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 into the microcontroller by clicking Enter.
Here is a list of the most frequently used *mikroICD* options:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Function key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start Debugger</td>
<td>Start up debugger</td>
<td>[F9]</td>
</tr>
<tr>
<td>Run/Pause Debugger</td>
<td>Run or pause debugger</td>
<td>[F6]</td>
</tr>
<tr>
<td>Stop Debugger</td>
<td>Stop debugger</td>
<td>[Ctrl+F2]</td>
</tr>
<tr>
<td>Step Into</td>
<td>Execute the current program line, then halts. If the program line executed</td>
<td>[F7]</td>
</tr>
<tr>
<td></td>
<td>calls another routine, the debugger steps into the routine and halts after</td>
<td></td>
</tr>
<tr>
<td></td>
<td>executing the first instruction within it.</td>
<td></td>
</tr>
<tr>
<td>Step Over</td>
<td>Execute the current program line, then halts. If the program line executed</td>
<td>[F8]</td>
</tr>
<tr>
<td></td>
<td>calls another routine, the debugger will not step into it. The whole routine</td>
<td></td>
</tr>
<tr>
<td></td>
<td>will be executed and the debugger halts at the first instruction following the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>call.</td>
<td></td>
</tr>
<tr>
<td>Step Out</td>
<td>Execute all remaining program lines within the subroutine. The debugger</td>
<td>[Ctrl+F8]</td>
</tr>
<tr>
<td></td>
<td>halts immediately upon exiting the subroutine. This option is provided with</td>
<td></td>
</tr>
<tr>
<td></td>
<td>the PIC18 microcontroller family, but not with the PIC16 microcontroller</td>
<td></td>
</tr>
<tr>
<td></td>
<td>family.</td>
<td></td>
</tr>
<tr>
<td>Run To Cursor</td>
<td>Execute the program until reaching the cursor position.</td>
<td>[F4]</td>
</tr>
<tr>
<td>Toggle Breakpoint</td>
<td>During the process of debugging, the program executes until reaching a</td>
<td>[F5]</td>
</tr>
<tr>
<td></td>
<td>breakpoint. The <em>Toggle Breakpoints</em> option sets new breakpoints or removes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>those already set at the current cursor position.</td>
<td></td>
</tr>
<tr>
<td>Show/Hide Breakpoints</td>
<td>To view all the breakpoints in the program, select the <em>Show/Hide Breakpoints</em></td>
<td>[Shift+F4]</td>
</tr>
<tr>
<td></td>
<td>option from the <em>Run</em> drop-down menu or use the <em>Shift +F4</em> shortcut.</td>
<td></td>
</tr>
<tr>
<td>Clear Breakpoints</td>
<td>Clear all breakpoints from the program.</td>
<td>[Ctrl+Shift+F5]</td>
</tr>
</tbody>
</table>
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 weapon 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.
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/en/support.

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