SOFTWARE AND HARDWARE SOLUTIONS FOR THE EMBEDDED WORLD

MikroElektronika

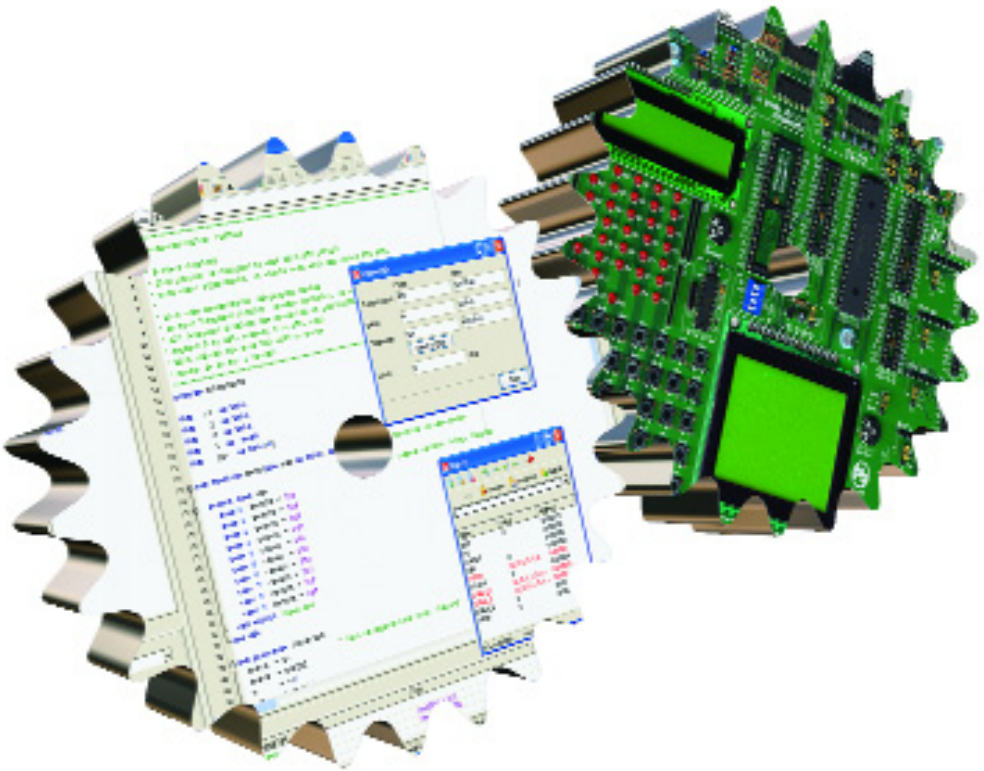
Development tools - Books - Compilers

User's manual

mikroPascal, mikroBasic and mikroC

Compilers IDE

Making it simple

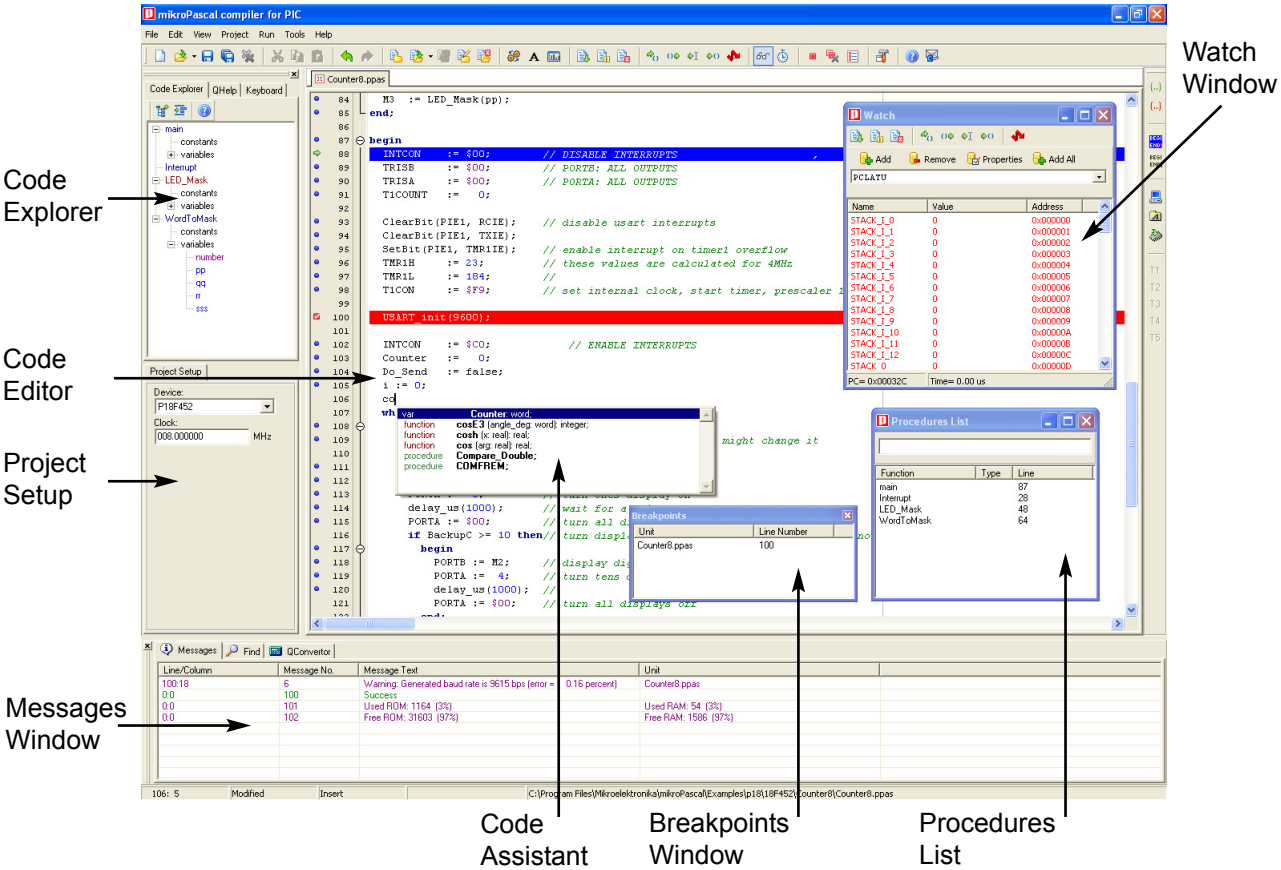


Highly sophisticated IDE provides the power you need with the simplicity of a Windows based point-and-click environment.

Software and Hardware solutions for Embedded World

CONTENTS

Integrated Development Environment



mikroElektronika compilers allows you to quickly develop and deploy complex applications:

- Write your source code using the highly advanced Code Editor

- Use the included libraries to dramatically speed up development: data acquisition, memory, displays, conversions, communications…

- Generate commented, human-readable assembly, and standard HEX compatible with all programmers.

- Inspect program flow and debug executable logic with the integrated Debugger. Get detailed reports and graphs on code statistics, assembly listing, calling tree…

- We have provided plenty of examples for you to expand, develop and use as building bricks in your projects.

CODE EDITOR

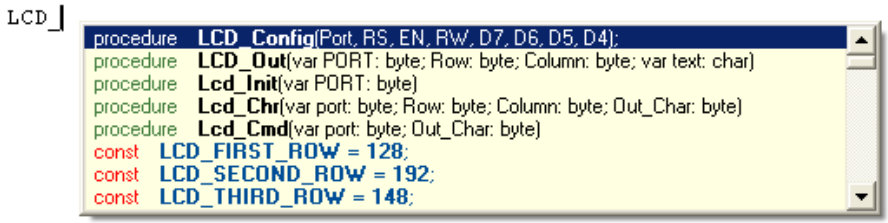
The Code Editor is advanced text editor fashioned to satisfy the needs of professionals. General code editing is same as working with any standard text-editor, including familiar Copy, Paste, and Undo actions, common for Windows environment.

Advanced Editor features include:

- Adjustable Syntax Highlighting
- Code Assistant
- Parameter Assistant
- Code Templates
- Auto Correct for common typos
- Bookmarks and Goto Line

Code Assistant [CTRL+SPACE]

If you type first few letter of a word and then press CTRL+SPACE, all valid identifiers matching the letters you typed will be prompted to you in a floating panel (see the image). Now you can keep typing to narrow the choice, or you can select one from the list using keyboard arrows and Enter.



Parameter Assistant [CTRL+SHIFT+SPACE]

Parameter Assistant will be automatically invoked when you open a parenthesis "(" or press CTRL+SHIFT+SPACE. If name of valid function or procedure precedes the parenthesis, then the expected parameters will be prompted to you in a floating panel. As you type the actual parameter, next expected parameter will become bold.

Code Template [CTR+J]

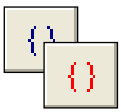
You can insert the Code Template by typing the name of the template (for instance, *whileb*), then press CTRL+J, and the Editor will automatically generate code. Or you can click button from the Code toolbar and select template from the list.

You can add your own templates to the list. Just select Tools > Options from the drop-down menu, or click Tools Icon from Settings Toolbar, and then select Auto Complete Tab. Here you can enter the appropriate keyword, description, and code of your template.

Auto Correct

The Auto Correct feature corrects common typing mistakes. To access the list of recognized typos, select Tools > Options from the drop-down menu, or click the Tools Icon from the Settings Toolbar, and then select the Auto Correct Tab. You can also add your own preferences to the list.

Comment/Uncomment



Comment / Uncomment Icon.

Code Editor allows you to comment or uncomment selected block of code by a simple click of a mouse, using the Comment/Uncomment icons from the Code Toolbar.

Bookmarks

Bookmarks make navigation through large code easier.

CTRL+<number> : Goto bookmark
CTRL+SHIFT+<number> : Set/Clear bookmark

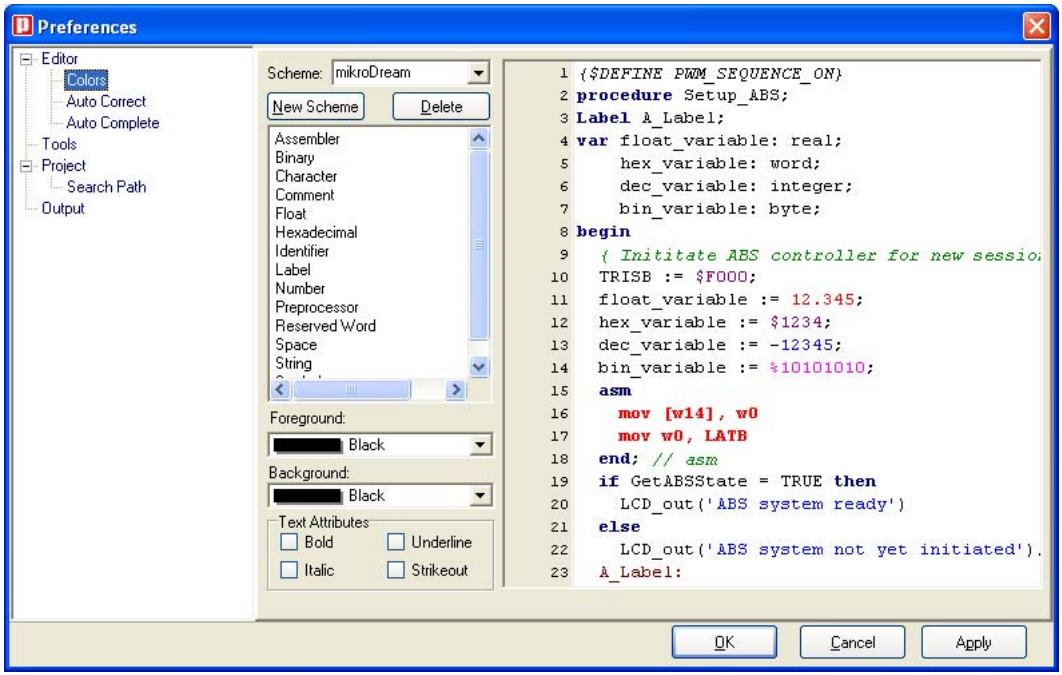
Goto Line

Goto Line option makes navigation through large code easier. Select Search > Goto Line from the drop-down menu or use the shortcut CTRL+G.

Editor Settings

Tools Icon.

You can customize these options from the Editor Settings dialog. To access the settings, click Tools > Options from the drop-down menu, or click the Tools icon.

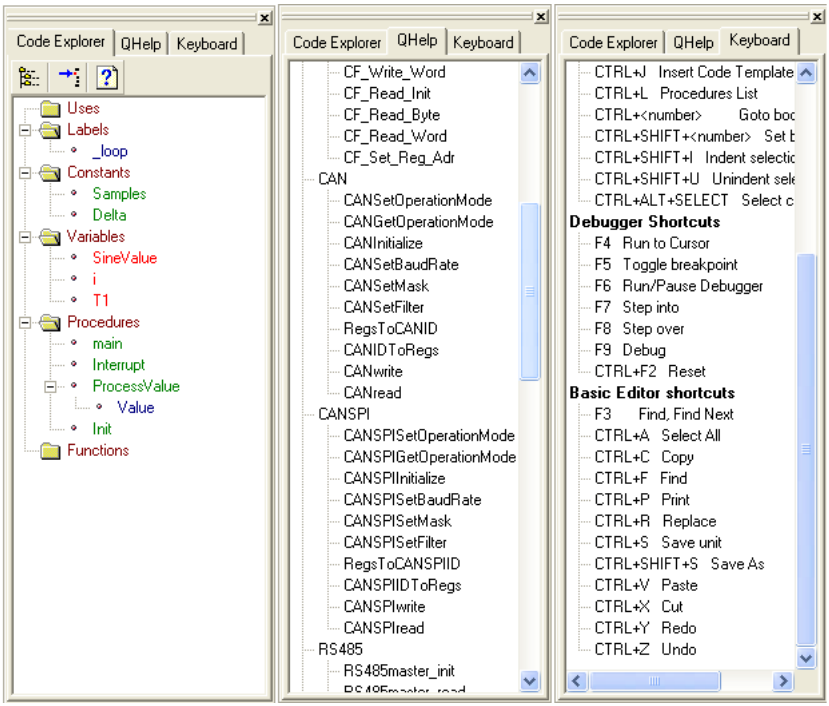


CODE EXPLORER

The Code Explorer is placed to the left of the main window by default, and gives clear view of every declared item in the source code. You can jump to declaration of any item by right clicking it, or by clicking the Find Declaration icon. To expand or collapse treeview in the Code Explorer, use the Collapse/Expand All icon.

Collapse/Expand All Icon.

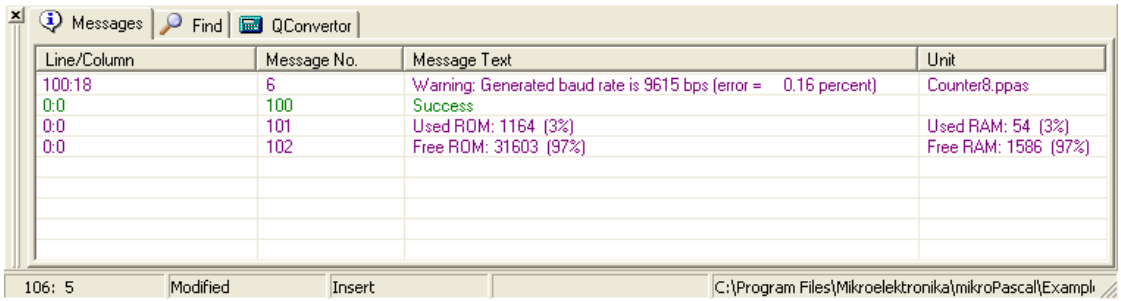
Also, two more tab windows are available in the Code Explorer. QHelp Tab lists all the available built-in and library functions, for a quick reference. Double-clicking a routine in QHelp Tab opens the relevant Help topic. Keyboard Tab lists all the available keyboard shortcuts in mikroPascal, mikroBasic or mikroC compiler.



MESSAGES WINDOW

In case that errors were encountered during compiling, compiler will report them and won't generate a hex file. The Messages Window will be prompted at the bottom of the main window.

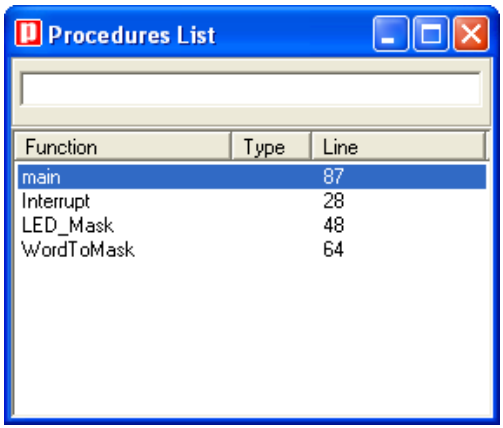
The Messages Window is located under message tab, and displays location and type of errors compiler has encountered. The compiler also reports warnings, but these do not affect generating hex code. Only errors can interefere with generation of hex.



Double click the message line in the Messages Window to highlight the line where the error was encountered.Consult the messages for more information about errors recognized by the compiler.

PROCEDURES LIST

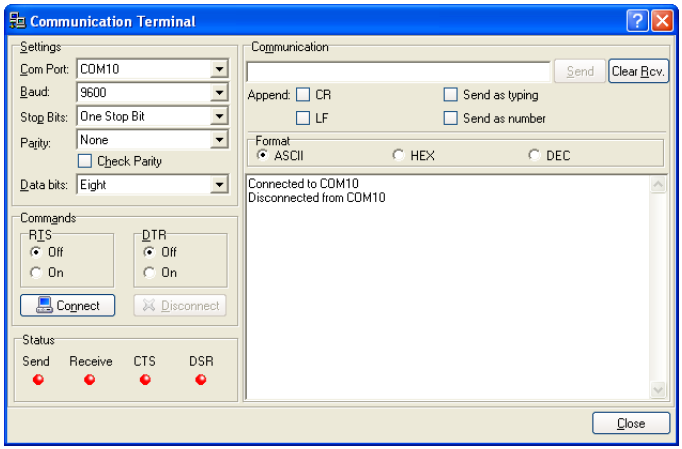
To view procedures list, select View > Procedures List from the drop-down menu or press Ctrl+L shortcut on the keyboard.



INTEGRATED TOOLS

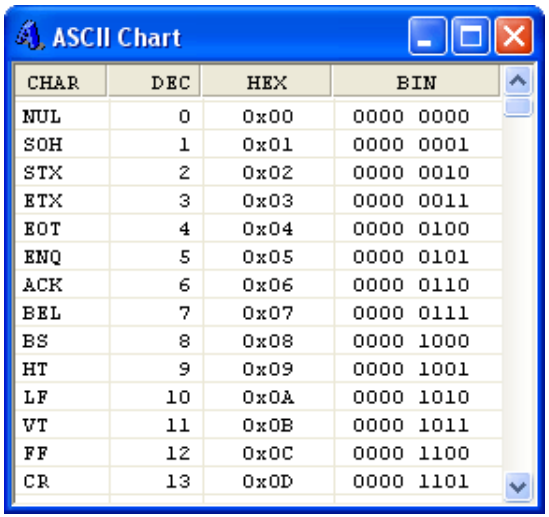
USART Terminal

All compilers includes the USART (Universal Synchronous Asynchronous Receiver Transmitter) communication terminal for RS232 communication. You can launch it from the drop-down menu Tools > Terminal or by clicking the Terminal icon.



ASCII Chart

The ASCII Chart is a handy tool, particularly useful when working with LCD display. You can launch it from the drop-down menu Tools > ASCII chart.

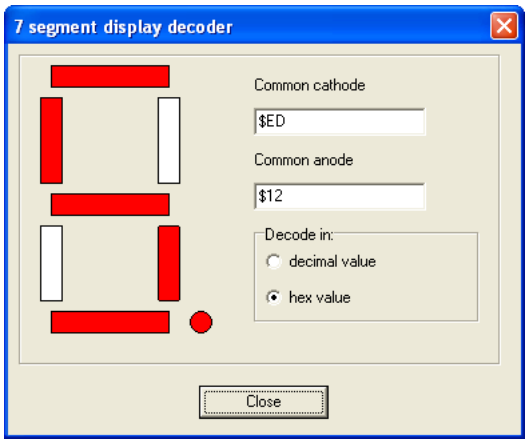


ASCII Chart

CHAR	DEC	HEX	BIN
NUL	0	0x00	0000 0000
SOH	1	0x01	0000 0001
STX	2	0x02	0000 0010
ETX	3	0x03	0000 0011
EOT	4	0x04	0000 0100
ENQ	5	0x05	0000 0101
ACK	6	0x06	0000 0110
BEL	7	0x07	0000 0111
BS	8	0x08	0000 1000
HT	9	0x09	0000 1001
LF	10	0x0A	0000 1010
VT	11	0x0B	0000 1011
FF	12	0x0C	0000 1100
CR	13	0x0D	0000 1101

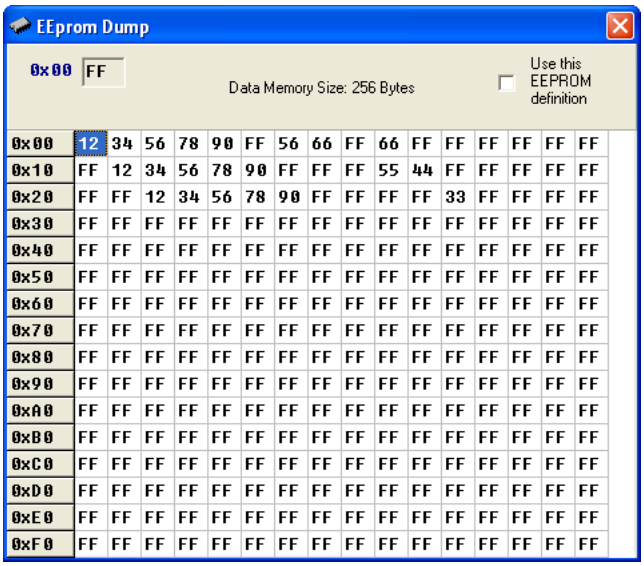
7 Segment Display Decoder

The 7seg Display Decoder is a convenient visual panel which returns decimal/hex value for any viable combination you would like to display on 7seg. Click on the parts of 7 segment image to the left to get the desired value in the edit boxes. You can launch it from the drop-down menu Tools > 7 Segment Display.



EEPROM Editor

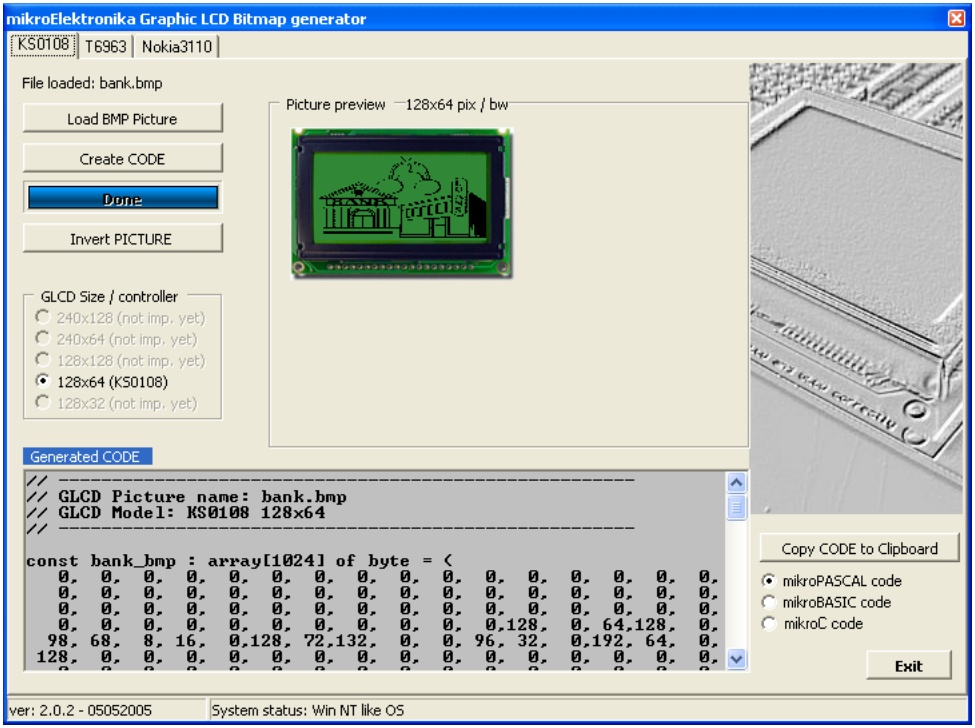
EEPROM Editor allows you to easily manage EEPROM of PIC microcontroller.



0x00	12	34	56	78	90	FF	56	66	FF	66	FF	FF	FF	FF	FF	FF
0x10	FF	12	34	56	78	90	FF	FF	FF	55	44	FF	FF	FF	FF	FF
0x20	FF	FF	12	34	56	78	90	FF	FF	FF	FF	33	FF	FF	FF	FF
0x30	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0x40	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0x50	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0x60	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0x70	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0x80	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0x90	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0xA0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0xB0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0xC0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0xD0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0xE0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0xF0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF

Graphic LCD Bitmap generator

Generates code for mikroPascal, mikroBasic and mikroC compilers from loaded bitmap picture.



KEYBOARD SHORTCUTS

Below is the complete list of keyboard shortcuts available in compilers IDE. You can also view keyboard shortcuts in the Code Explorer, tab Keyboard.

IDE Shortcuts

F1	Help
CTRL+N	New Unit
CTRL+O	Open
CTRL+F9	Compile
CTRL+F11	Code Explorer on/off
CTRL+SHIFT+F5	View breakpoints

Basic Editor shortcuts

F3	Find, Find Next
CTRL+A	Select All
CTRL+C	Copy
CTRL+F	Find
CTRL+P	Print
CTRL+R	Replace
CTRL+S	Save unit
CTRL+SHIFT+S	Save As
CTRL+V	Paste
CTRL+X	Cut
CTRL+Y	Redo
CTRL+Z	Undo

Advanced Editor shortcuts

CTRL+SPACE	Code Assistant
CTRL+SHIFT+SPACE	Parameters Assistant
CTRL+D	Find declaration
CTRL+G	Goto line
CTRL+J	Insert Code Template
CTRL+L	Procedures List
CTRL+<number>	Goto bookmark
CTRL+SHIFT+<number>	Set bookmark
CTRL+SHIFT+I	Indent selection
CTRL+SHIFT+U	Unindent selection
CTRL+ALT+SELECT	Select columns

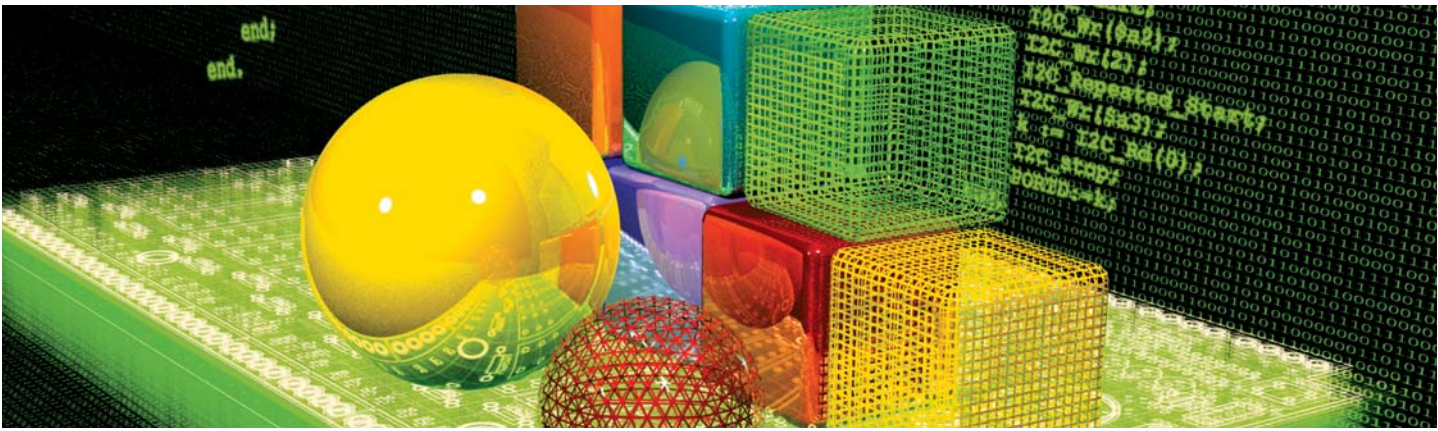
Debugger Shortcuts

F4	Run to Cursor
F5	Toggle breakpoint
F6	Run/Pause Debugger
F7	Step into
F8	Step over
CTRL+F8	Step out
F9	Debug
F2	Jump to Interrupt
CTRL+F2	Reset

mikroPascal and mikroBasic for AVR compilers have additional shortcuts:

Ctrl+F5	Add to Watch list
Shift+Ctrl+F6	View RAM
Ctrl+F6	View Clock
Shift+F6	Edit Registers

CHAPTER 1



Building Applications

QUICK OVERVIEW

Creating applications in mikroPascal, mikroBasic or mikroC is easy and intuitive. Project Wizard allows you to set up your project in just few clicks: name your application, select chip, set flags and get going.

These compilers allows you to distribute your projects in as many units as you find appropriate. You can then share your mikroCompiled Libraries (`.mcl` files) with other developers without disclosing the source code. The best part is that in mikroPascal you can use `.mcl` bundles created by mikroBasic or mikroC (and other way around)!

PROJECTS

mikroPascal, mikroBasic and mikroC organizes applications into *projects*, consisting of a single project file and one or more source files. You can compile source files only if they are part of a project.

Project file carries the following information:

- project name and optional description
- target device
- device flags (config word) and device clock
- list of project source files with paths

New Project

New Project.

The easiest way to create project is by means of New Project Wizard, drop-down menu Project > New Project. Just fill the dialog with desired values (project name and description, location, device, clock, config word) and compiler will create the appropriate project file.

Also, an empty source file named after the project will be created by default. Compilers does not require you to have source file named same as the project, it's just a matter of convenience.

Editing Project

Edit Project.

Later, you can change project settings from the drop-down menu Project > Edit. You can add or remove source files from project, rename the project, modify its description, change chip, clock, config word, etc.

To delete a project, simply delete the folder in which the project file is stored.

SOURCE FILES

Source files containing program code should have following extensions:

mikroPascal for PIC:	`.ppas`
mikroBasic for PIC:	`.pbas`
mikroC for PIC and VR-STAMP:	`.c`
mikroPascal for dsPIC:	`.dpas`
mikroBasic for dsPIC:	`.dbas`
mikroPascal for AVR:	`.apas`
mikroBasic for AVR:	`.abas`

List of source files relevant for the application is stored in project file with following extensions (along with other project information):

mikroPascal for PIC:	`.ppp`
mikroBasic for PIC:	`.pbp`
mikroC for PIC:	`.ppc`
mikroC for VR-STAMP:	`.psc`
mikroPascal for dsPIC:	`.dpp`
mikroBasic for dsPIC:	`.dbp`
mikroPascal for AVR:	`.app`
mikroBasic for AVR:	`.abp`

You can compile source files only if they are part of a project.

Managing Source Files

New File.

Creating a new source file

To create a new source file, do the following:

Select File > New from the drop-down menu, or press CTRL+N, or click the New File icon. A new tab will open, named “Untitled1”. This is your new source file. Select File > Save As from the drop-down menu to name it the way you want.

If you have used New Project Wizard, an empty source file, named after the project with extension defined by compiler, is created automatically. Compiler does not require you to have source file named same as the project, it’s just a matter of convenience.

Open File.

Opening an Existing File

Select File > Open from the drop-down menu, or press CTRL+O, or click the Open File icon. The Select Input File dialog opens. In the dialog, browse to the location of the file you want to open and select it. Click the Open button. The selected file is displayed in its own tab. If the selected file is already open, its current Editor tab will become active

Print File.

Printing an Open File

Make sure that window containing the file you want to print is the active window. Select File > Print from the drop-down menu, or press CTRL+P, or click the Print icon. In the Print Preview Window, set the desired layout of the document and click the OK button. The file will be printed on the selected printer.

Save File.

Saving File

Make sure that window containing the file you want to save is the active window. Select File > Save from the drop-down menu, or press CTRL+S or click the Save icon. The file will be saved under the name on its window.

Save File As.

Saving File Under a Different Name

Make sure that window containing the file you want to save is the active window. Select File > Save As from the drop-down menu, or press SHIFT+CTRL+S. The New File Name dialog will be displayed. In the dialog, browse to the folder where you want to save the file. In the File Name field, modify the name of the file you want to save. Click the Save button.

Close File.

Closing a File

Make sure that tab containing the file you want to close is the active tab. Select File > Close from the drop-down menu, or right click the tab of the file you want to close in Code Editor. If the file has been changed since it was last saved, you will be prompted to save your changes.

COMPILATION

Build Icon.

When you have created the project and written the source code, you will want to compile it. Select Project > Build from the drop-down menu or click the Build Icon or simply hit CTRL+F9.

Progress bar will appear to inform you about the status of compiling. If there are errors, you will be notified in the Error Window. If no errors are encountered, compiler will generate output files.

Output Files

Upon successful compilation, compiler will generate output files in the project folder (folder which contains the project file). Output files are summarized below:

Intel HEX file (`.hex`)
Intel style hex records. Use this file to program MCU.

Binary mikro Compiled Library (`.mcl`)
Binary distribution of application that can be included in other projects.

List File (`.lst`)
Overview of MCU memory allotment: instruction addresses, registers, routines, etc.

Assembler File (`.asm`)
Human readable assembly with symbolic names, extracted from the List File.

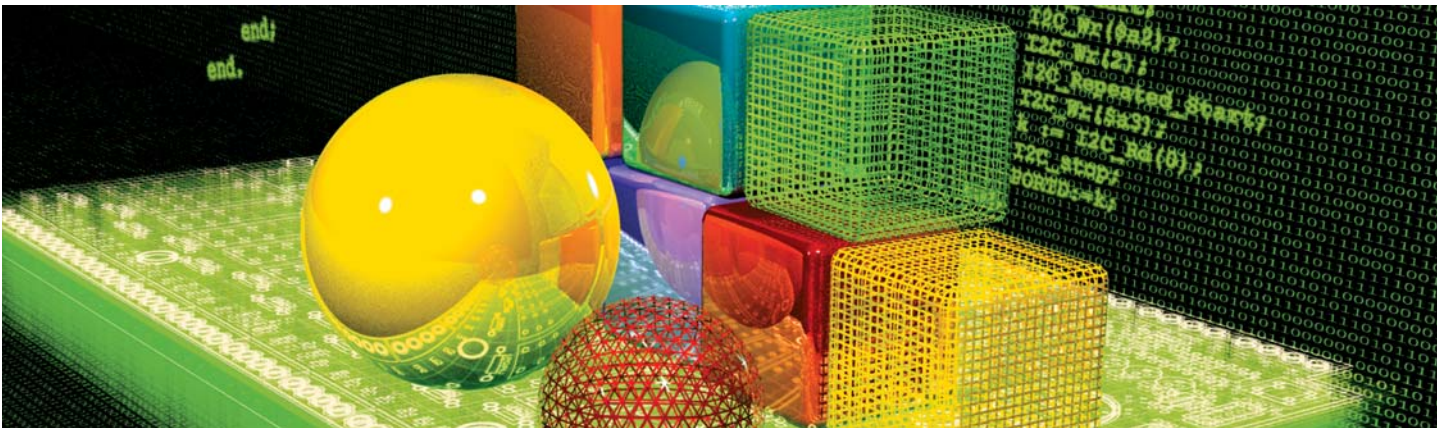
Assembly View

View Assembly Icon.

After compiling your program in mikroPascal, mikroBasic or mikroC, you can click View Assembly Icon or select Project > View Assembly from the drop-down menu to review generated assembly code (`.asm` file) in a new tab window. Assembly is human readable with symbolic names. All physical addresses and other information can be found in Statistics or in list file (`.lst`).

If the program is not compiled and there is no assembly file, starting this option will compile your code and then display assembly.

CHAPTER 2



Debugger

QUICK OVERVIEW

Source-level Debugger is an integral component of mikroPascal, mikroBasic and mikroC compilers. It is designed to simulate operations of PIC, dsPIC, AVR and VR-STAMP microcontrollers and to assist users in debugging program code written for these devices.

DEBUGGER

Source-level Debugger is an integral component of mikroPascal, mikroBasic and mikroC development environment. It is designed to simulate operations of micro-controllers and to assist users in debugging software written for these devices.

Debugger simulates program flow and execution of instruction lines, but does not fully emulate microcontroller device behavior: it does not update timers, interrupt flags, etc.

After you have successfully compiled your project, you can run the Debugger by selecting Run > Debug from the drop-down menu, or by clicking Debug Icon. Starting the Debugger makes more options available: Step Into, Step Over, Run to Cursor, etc. Line that is to be executed is color highlighted.

Start Debugger.

Debug [F9]
Start the Debugger.

Pause Debugger.

Run/Pause Debugger [F6]
Run or pause the Debugger.

Step Into.

Step Into [F7]
Execute the current (single– or multi–cycle) instruction, then halt. If the instruction is a routine call, enter the routine and halt at the first instruction following the call.

Step Over.

Step Over [F8]
Execute the current (single– or multi–cycle) instruction, then halt. If the instruction is a routine call, skip it and halt at the first instruction following the call.

Step Out.

Step Out [Ctrl+F8]
Execute the current (single– or multi–cycle) instruction, then halt. If the instruction is within a routine, execute the instruction and halt at the first instruction following the call.

Run to Cursor.

Run to cursor [F4]
Executes all instructions between the current instruction and the cursor position.

Jump to Interrupt.

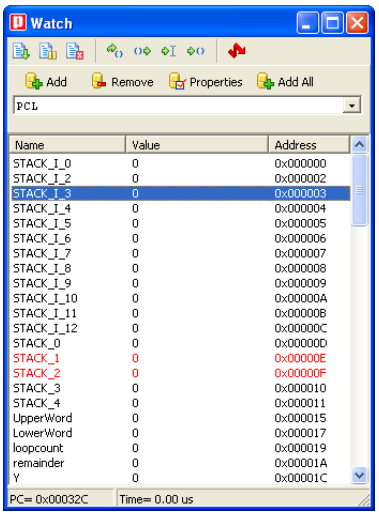
Jump to Interrupt [F2]
Only for PIC microcontrollers. Jump to address `$04` for PIC12/16 or to address `$08` for PIC18 and execute the procedure located at that address.

Toggle Breakpoint.

Toggle Breakpoint [F5]
Toggle breakpoint at the current cursor position. To view all the breakpoints, select Run > View Breakpoints from the drop-down menu. Double clicking an item in window list locates the breakpoint.

Watch Window

Debugger Watch Window is the main Debugger window which allows you to monitor program items while running your program. To show the Watch Window, select View > Debug Windows > Watch Window from the drop-down menu.



The Watch Window displays variables and registers of microcontrollers, with their addresses and values. Values are updated as you go through the simulation. Use the drop-down menu to add and remove the items that you want to monitor. Recently changed items are colored red.

Double clicking an item opens the Edit Value window in which you can assign a new value to the selected variable/register. Also, you can change view to binary, hex, char, or decimal for the selected item.

Stopwatch Window

Debugger Stopwatch Window is available from the drop-down menu, View > Debug Windows > Stopwatch.

The Stopwatch Window displays the current count of cycles/time since the last Debugger action. Stopwatch measures the execution time (number of cycles) from the moment Debugger is started, and can be reset at any time. Delta represents the number of cycles between the previous instruction line (line where the Debugger action was performed) and the active instruction line (where the Debugger action landed).

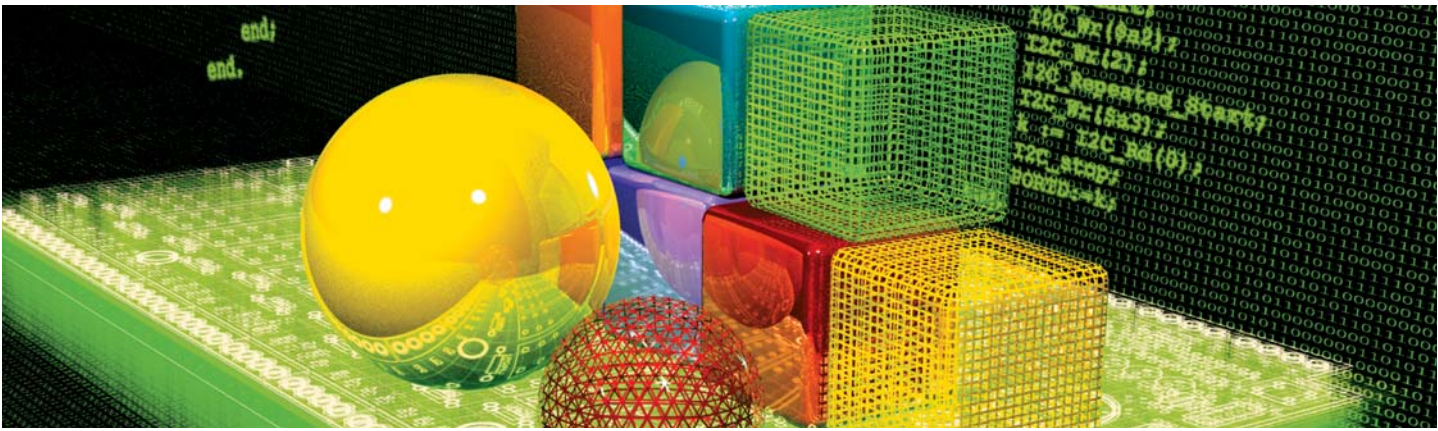
Note: You can change the clock in the Stopwatch Window; this will recalculate values for the newly specified frequency. Changing the clock in the Stopwatch Window does not affect the actual project settings – it only provides a simulation.

View RAM Window

Debugger View RAM Window is available from the drop-down menu, View > Debug Windows > View RAM.

The View RAM Window displays the map of microcontroller's RAM, with recently changed items colored red. You can change value of any field by double-clicking it.

CHAPTER 3



Statistics

QUICK OVERVIEW

After successful compiling, you can review statistics on your code. You can use following tools: Memory Usage Window, Procedures (Sizes) Window, Procedures (Locations) Window, Procedures (Details) Window, RAM Window and ROM Window.

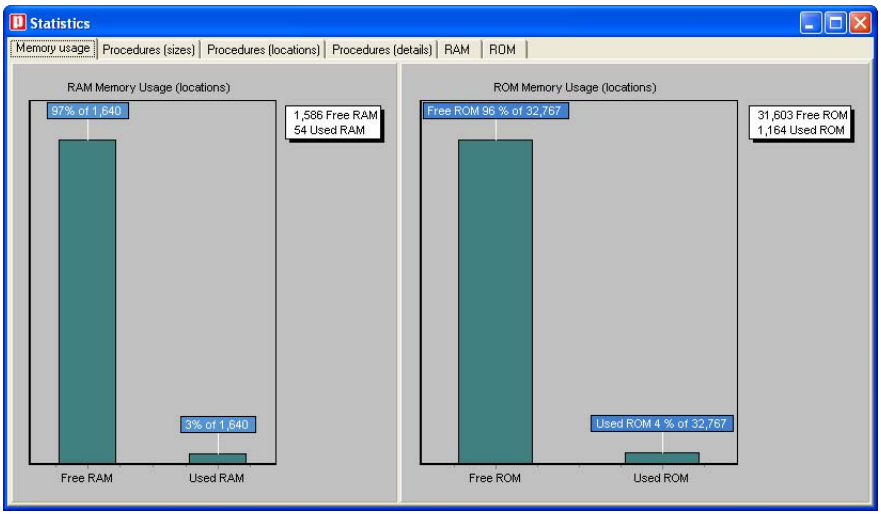
STATISTICS

Statistics Icon.

After successful compilation, you can review statistics of your code. Select View > View Statistics from the drop-down menu, or click the Statistics icon. There are six tab windows:

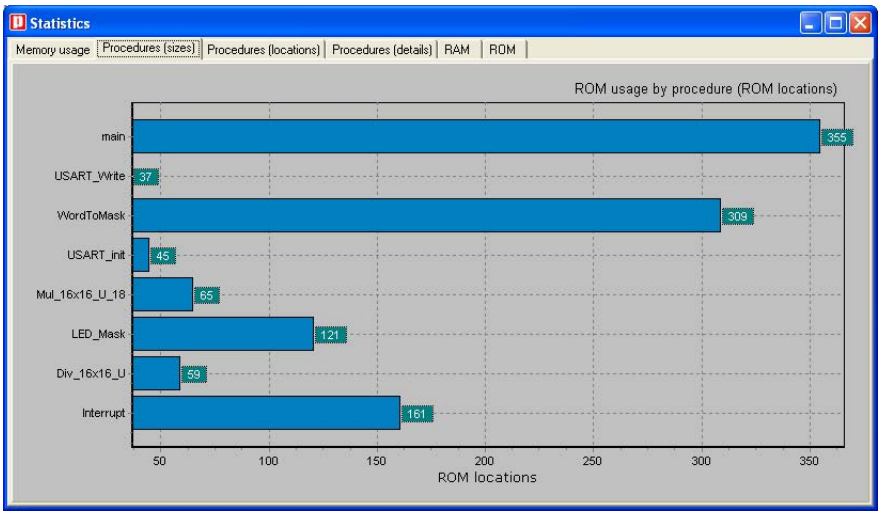
Memory Usage Window

Provides overview of RAM and ROM memory usage in form of histogram.



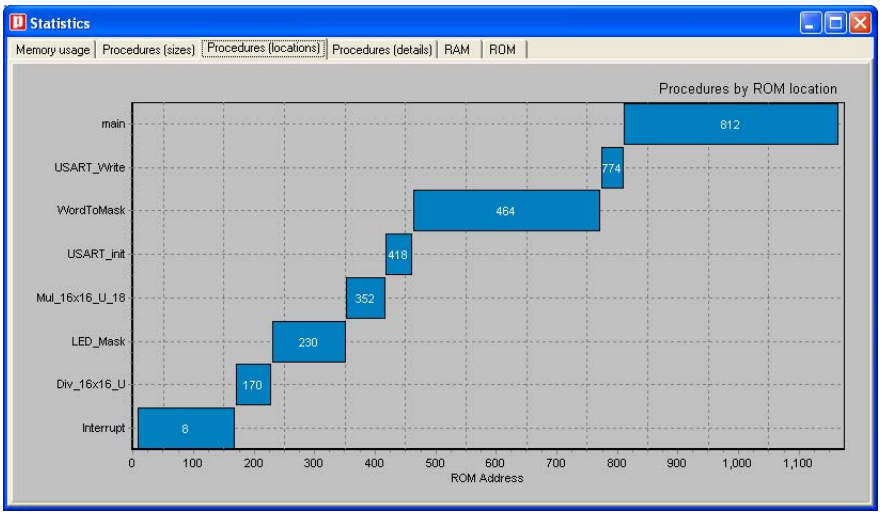
Procedures (Sizes) Window

Displays functions in form of histogram, according to their memory allotment.



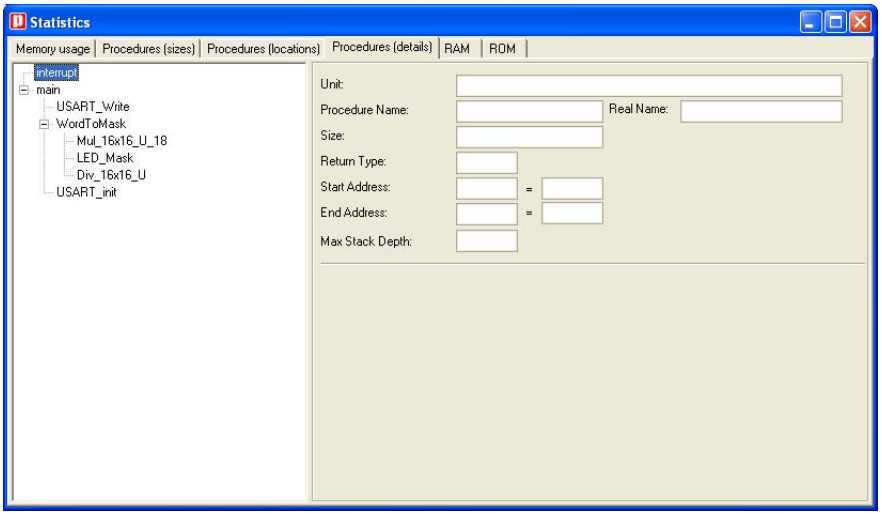
Procedures (Locations) Window

Displays how functions are distributed in microcontroller's memory.



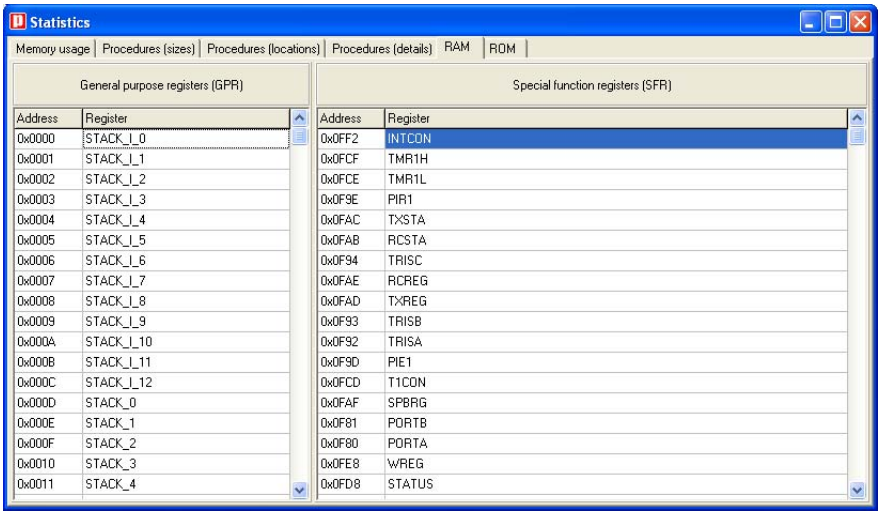
Procedures (Details) Window

Displays complete call tree, along with details for each procedure and function: size, start and end address, calling frequency, return type, etc.



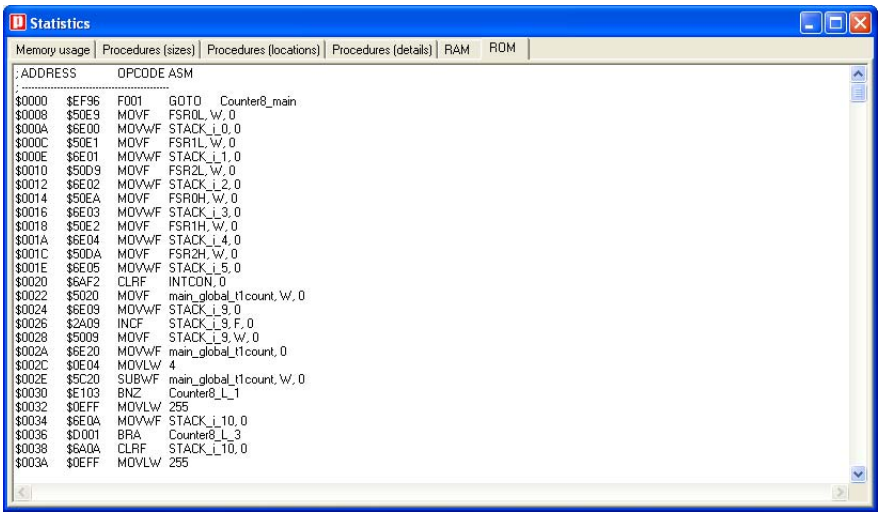
RAM Window

Summarizes all GPR and SFR registers and their addresses. Also displays symbolic names of variables and their addresses.



ROM Window

Lists op-codes and their addresses in form of a human readable hex code.



Second edition
January 2006