



---

# **GENTOS: CoreRiver's Total Solutions for Embedded System Development**

---

Preliminary

**Rev. 1.2  
September 2005**

**Copyright CoreRiver Semiconductor Co., Ltd. 2005  
All Rights Reserved**

- ◆ *CoreRiver Semiconductor reserves the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice.*
- ◆ *Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete.*
- ◆ *The CoreRiver Semiconductor products listed in this document are intended for usage in general electronics applications. These CoreRiver Semiconductor products are neither intended nor warranted for usage in equipment that requires extraordinarily high quality and/or reliability or a malfunction or failure of which may cause loss of human life or bodily injury.*

Preliminary



## Table of Contents

<b>1</b>	<b>INSTALLATION .....</b>	<b>9</b>
1.1	SYSTEM REQUIREMENTS .....	9
1.1.1	<i>Hardware Requirements</i> .....	9
1.1.2	<i>Software Requirements</i> .....	9
1.2	SOFTWARE INSTALLATION .....	9
<b>2</b>	<b>DEVELOPMENT ENVIRONMENT .....</b>	<b>11</b>
2.1	WORKSPACES .....	11
2.1.1	<i>Path Specifications</i> .....	11
2.1.2	<i>Working with Workspaces</i> .....	11
2.2	WINDOW MANAGEMENT .....	12
2.2.1	<i>Dockable Windows</i> .....	13
2.2.2	<i>Mini Frame Windows</i> .....	13
2.2.3	<i>Context Menus</i> .....	14
2.2.4	<i>Toolbars</i> .....	14
<b>3</b>	<b>DOCUMENT MANAGEMENT .....</b>	<b>17</b>
3.1	DOCUMENT WINDOWS .....	17
3.1.1	<i>Creating a New Document</i> .....	17
3.1.2	<i>Opening an Existing Document</i> .....	17
3.2	THE EDITOR .....	17
3.2.1	<i>Selecting Text Block</i> .....	18
3.2.2	<i>Bookmarking the Text</i> .....	19
3.2.3	<i>Search and Replace Operations</i> .....	20
3.2.4	<i>Clipboard operations</i> .....	21
3.2.5	<i>Drag and Drop</i> .....	21
3.2.6	<i>The Context Menu</i> .....	22
3.2.7	<i>Printing</i> .....	22
3.2.8	<i>Configuring Editor Options</i> .....	23
<b>4</b>	<b>BUILD MANAGER .....</b>	<b>25</b>
4.1	INTRODUCTION .....	25
4.1.1	<i>Detecting Changes to the Project</i> .....	25
4.2	PROJECT ORGANIZATION .....	25

4.2.1	<i>Project Files</i> .....	26
4.3	COMPILER TOOL-SET INTEGRATION.....	26
4.3.1	<i>Compiler page</i> .....	27
4.3.2	<i>Debugger page</i> .....	27
4.3.3	<i>Assembler page</i> .....	28
4.3.4	<i>Linker page</i> .....	29
4.4	USING AN EXAMPLE WORKSPACE AS A TEMPLATE.....	30
4.5	BUILD SESSION.....	30
4.5.1	<i>Output Window</i> .....	30
4.5.2	<i>Building</i> .....	31
<b>5</b>	<b>DEBUG SESSION</b> .....	<b>33</b>
5.1	DOWNLOAD FILES.....	33
5.2	BREAKPOINTS.....	33
5.2.1	<i>Execution Breakpoints</i> .....	33
5.2.2	<i>Clear All Breakpoints</i> .....	35
5.2.3	<i>H/W Breakpoints</i> .....	35
5.3	SOURCE DEBUGGING.....	35
5.3.1	<i>Go</i> .....	35
5.3.2	<i>Reset &amp; Run</i> .....	36
5.3.3	<i>Stop Debugging</i> .....	36
5.3.4	<i>Stop</i> .....	36
5.3.5	<i>Reset</i> .....	36
5.3.6	<i>Step Into</i> .....	36
5.3.7	<i>Step Over</i> .....	36
5.3.8	<i>Stepi</i> .....	37
5.3.9	<i>Setting Execution Breakpoints</i> .....	37
5.4	DISASSEMBLY WINDOW.....	37
5.5	MEMORY WINDOW.....	38
5.5.1	<i>Opening a Memory Window</i> .....	39
5.5.2	<i>Finding Memory Contents</i> .....	39
5.5.3	<i>Modifying Memory Contents</i> .....	39
5.6	WATCH WINDOW.....	39
5.6.1	<i>Adding Watch Expressions</i> .....	39
5.6.2	<i>Modifying Values</i> .....	39
5.6.3	<i>Watching Complex Expressions</i> .....	39

5.7	SPECIAL FUNCTION REGISTERS WINDOW .....	39
5.7.1	<i>Modifying a register</i> .....	39
<b>6</b>	<b>HARDWARE CONFIGURATION .....</b>	<b>39</b>
6.1	DEBUG HARDWARE AND DEBUG MODE.....	39
6.1.1	<i>PC to Emulator Communication</i> .....	39
6.1.2	<i>Configuration</i> .....	39
6.2	EMULATION WITH TARGET SYSTEM.....	39
6.3	EMULATION WITH TARGET CODE .....	39
<b>7</b>	<b>TECHNICAL NOTES.....</b>	<b>39</b>
7.1	DIFFERENCES FROM A STANDARD ENVIRONMENT.....	39
7.2	COMMON GUIDELINES .....	39
7.3	PORT REPLACEMENT INFORMATION .....	39
7.4	INTERRUPTS .....	39
7.4.1	<i>Interrupt Handling When Stopped</i> .....	39
7.4.2	<i>Debugging Interrupt Routines</i> .....	39
7.5	MEMORY ACCESS .....	39

## List of Figures

Figure 2-1	New Workspace Dialog .....	12
Figure 2-2	GENTOS and the Watch window as a mini frame window.....	14
Figure 3-1	Open File dialog .....	17
Figure 3-2	Block selection.....	18
Figure 3-3	Selected Text in edit window .....	19
Figure 3-4	Source window with bookmarked locations .....	20
Figure 3-5	Find dialog.....	20
Figure 3-6	Replace dialog.....	21
Figure 3-7	Editor's context menu .....	22
Figure 3-8	Print dialog.....	23
Figure 3-9	Editor options pane.....	24
Figure 3-10	Editor options, colors pane .....	24
Figure 4-1	Project Workspace Window showing project hierarchy.....	26
Figure 4-2	Project option dialog, Compiler page .....	27
Figure 4-3	Project option dialog, Debugger page.....	28

---

Figure 4-4 Project option dialog, Assembler page .....	28
Figure 4-5 Project option dialog, Linker page.....	29
Figure 4-6 Output window, Build pane .....	31
Figure 5-1 Execution breakpoint indicated in source.....	34
Figure 5-2 H/W Breakpoint dialog .....	35
Figure 5-3 Disassembly window .....	37
Figure 5-4 Memory Window .....	38
Figure 5-5 Watch Window.....	39
Figure 5-6 Register Window showing CoreRiver MCU's registers .....	39
Figure 6-1 Hardware Configuration dialog, Communication page.....	39
Figure 6-2 Hardware Configuration dialog, ROM setup page.....	39
Figure 6-3 Hardware Configuration dialog, RAM setup page .....	39
Figure 6-4 Hardware Configuration dialog, Clock setup page .....	39
Figure 6-5 Hardware Configuration dialog, Reset setup page.....	39





## 1 Installation

### 1.1 System Requirements

#### 1.1.1 Hardware Requirements

- 486/66 MHz or better Processor
- 32MByte or more memory
- 30MByte or more hard disk space

#### 1.1.2 Software Requirements

- Windows 2000/XP or higher

### 1.2 Software Installation

- Download the install program "GENTOS\_VXXX-2000-install.exe" from download center in CoreRiver Homepage ([www.coreriver.com](http://www.coreriver.com)). XXX means the version number.
- Follow the installation wizard.
- You must install GENTOS at the c:\GENTOS directory.



## 2 Development Environment

### 2.1 Workspaces

GENTOS organizes project development in workspaces. All of the workspace information is stored in a workspace file (.gts extension). The workspace file contains the information on how the files necessary for a successful build of your project are related to each other, along with information on other features that aid development, such as bookmarks, breakpoints, desktop layout etc.

Ideally you will organize your project in a new directory where the workspace file will be located. This is considered to be your working directory. You are free to create sub-directories and place your source files there. All file path information in a workspace is stored relative to the workspace file location, so you will be able to move and copy the workspace to any other directory without disturbing its functionality.

#### 2.1.1 Path Specifications

At various places in GENTOS, especially when configuring project settings, you will need to enter file paths. To assure maximum workspace moveability, GENTOS recognizes and internally maintains three different path specifications:

- Files in the directory or sub-directory of the workspace file. These are stored and displayed in relative form.
- Files that can be stored in relative form of the compiler toolset directory. These are stored in relative form to the compiler toolset path, but displayed in absolute form.
- Files that can not be stored in relative form to either of above referential directories. These are stored and displayed in absolute form.

#### 2.1.2 Working with Workspaces

There is not much you can do in GENTOS without having a workspace open. To get you there, you will either open an existing workspace (one you have created previously or an example workspace shipped with installation files) or create a new one from scratch.

##### 2.1.2.1 Opening an Existing Workspace

- Select the “Open Workspace” sub-menu from the “File” menu
- Browse for the workspace file
- Click the “Open” button after selecting the desired workspace file

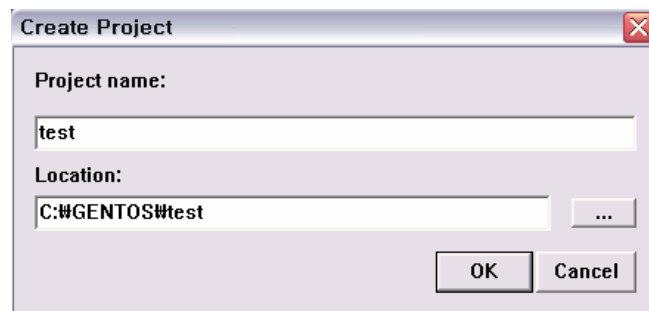
Alternatively you can select a previously opened workspace from the recent list in the “File” menu

- Open the “File” menu
- Four most recently used workspaces are listed on “Recent Workspaces” sub-menu. Select the

workspace of your choice.

### 2.1.2.2 Creating a New Workspace

- Select the “New Workspace” sub-menu from the “File” menu
- After prompted for project directory and name, browse for directory of your choice (an empty new directory is recommended) and enter the name for the new workspace file.



**Figure 2-1 New Workspace Dialog**

A newly created workspace contains only default project and hardware settings.

### 2.1.2.3 Saving Workspaces

You can explicitly save a workspace with the “Save Workspace” sub-menu. You will find this most useful when creating workspaces with the same project and hardware configuration, and sometimes with the same set of project files.

- Select the “Save Workspace” sub-menu from the “File” menu
- Browse for an existing workspace file or the desired directory and enter the name of the new workspace file manually
- Click the “Save” button when finished

## 2.2 Window Management

GENTOS is a multiple document interface (MDI) application, which means that several documents can be viewed and edited at the same time. In this case the term ‘document’ applies to source code files (typically C sources) for your project that you create and edit using the built-in or an external editor, build using the Build Manager and debug with the integrated debugger. The MDI enables you to open and process one or more of your source files, which are displayed in standard MDI child windows.

However, there is plenty of other information to be displayed in specialized windows, like the project workspace window (displaying project hierarchy), the output window (displaying build output), specialized

debugger windows, etc.

Besides standard MDI windows, GENTOS introduces two more window types that can be manipulated easier than MDI windows.

The type of the window can be set from its context menu:

- MDI child: the standard window
- Dockable: a window that can be docked like a toolbar
- Mini Frame: a window that can be floated above other windows or even outside GENTOS application window

### **2.2.1 Dockable Windows**

Dockable toolbars have been around in mainstream windows programs for some years now, and everyone has come to appreciate that the toolbar is always visible and at the same time not occupying any unnecessary desktop real estate.

GENTOS promotes the same concept to specialized windows that you will want to have around all the time. Once they are docked, a document can not obscure them, they are not subject to repositioning during tile and cascade operations and they minimize desktop real estate consumption by not displaying a caption bar.

The area occupied by docked windows and toolbars is out of reach of your documents. When a document is maximized, it will grow to occupy the remainder of GENTOS application window.

Typically you will create your preferred window layout along with colors and fonts and then never feel the need to reposition a window again. Your documents are then best viewed maximized.

#### **2.2.1.1 Docking a Window**

Place the mouse cursor on the border of the selected window. The mouse cursor will change to the drag shape when placed over the window border,

- Hold down the left mouse button while dragging the window to its new location.
- Release the left mouse button.

### **2.2.2 Mini Frame Windows**

A mini frame window can be floated anywhere on the desktop. If placed over GENTOS application window, it will overlap all its MDI and docked windows.

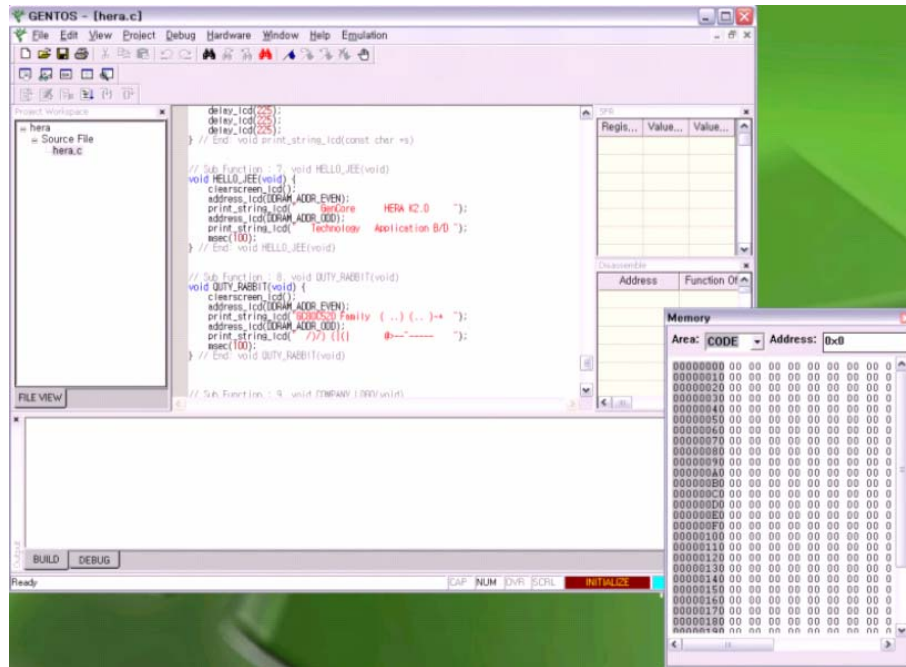


Figure 2-2 GENTOS and the Watch window as a mini frame window

### 2.2.3 Context Menus

Following windows guidelines for user interface design, some GENTOS windows have a context menu, activated either by a right mouse click in the window or with a 'Shift+F10' keyboard shortcut.

You will find context menus richer in options and faster to manipulate than the global menu. Along with toolbars, context menus make a trip to the menu bar rarely.

### 2.2.4 Toolbars

Toolbars provide easy, one-click access to most often used commands, which have been grouped in following categories:

#### 2.2.4.1 Edit Toolbar

The Edit Toolbar contains source file management commands - i.e. file and edit operations.



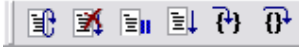
#### 2.2.4.2 Window Toolbar

View Toolbar buttons control specialized windows state.



### 2.2.4.3 Debug Toolbar

Debug Toolbar buttons control debug system operation.







## 3 Document Management

### 3.1 Document Windows

GENTOS is a Multiple Document Interface (MDI) application, which means that several documents can be viewed and edited at the same time. The MDI interface allows opening and processing one or more of your source files, which are displayed in standard MDI child windows.

GENTOS supports only ASCII text type document.

#### 3.1.1 Creating a New Document

To create a new document, select the “New” sub-menu from the “File” menu.

#### 3.1.2 Opening an Existing Document

To open an existing document, select the “Open” sub-menu from the “File” menu. And then, select its name and location (folder).

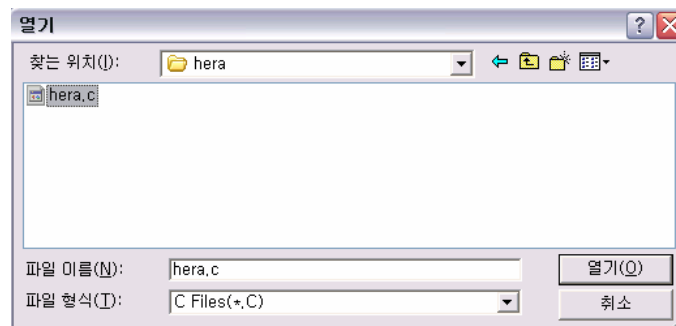


Figure 3-1 Open File dialog

### 3.2 The Editor

The editor is the place where you will probably spend most of your time, both when writing as well as debugging your sources. The integrated editor in GENTOS supports you with basic functions for editing non-document type files, like:

- find and replace operations
- copy, cut and paste via system clipboard
- drag and drop text operation

as well as some advanced programming aid features, like:

- color syntax source coloring
- debug watch tips, etc.

All editor related commands are available from the “Edit” menu.

### 3.2.1 Selecting Text Block

Selecting a block of text is necessary when you wish to copy it to the clipboard, move or copy it to another location using drag and drop, or erasing it.

#### 3.2.1.1 Selecting Text Block with Mouse

- Position the mouse on the place in the editor window where you wish the selected block to begin,
- Press the left mouse button,
- While keeping the left mouse button pressed, drag the mouse to the location where you want the selected block to end,
- Release the left mouse button

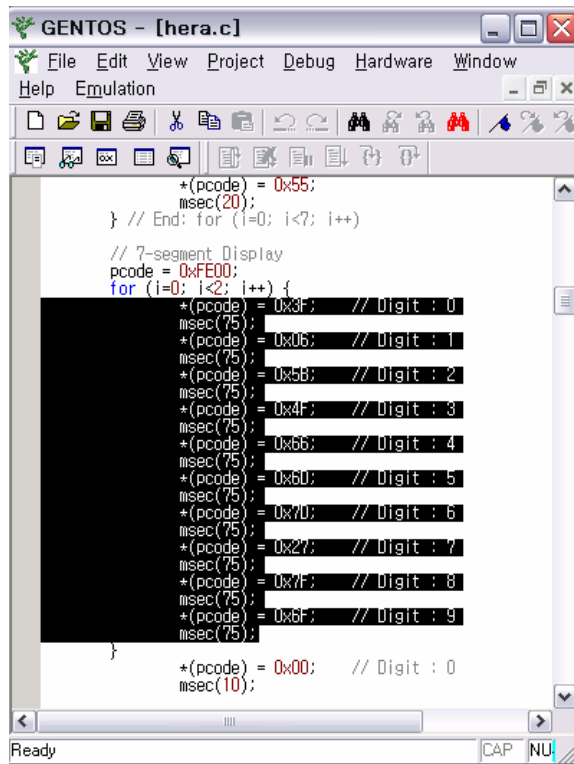
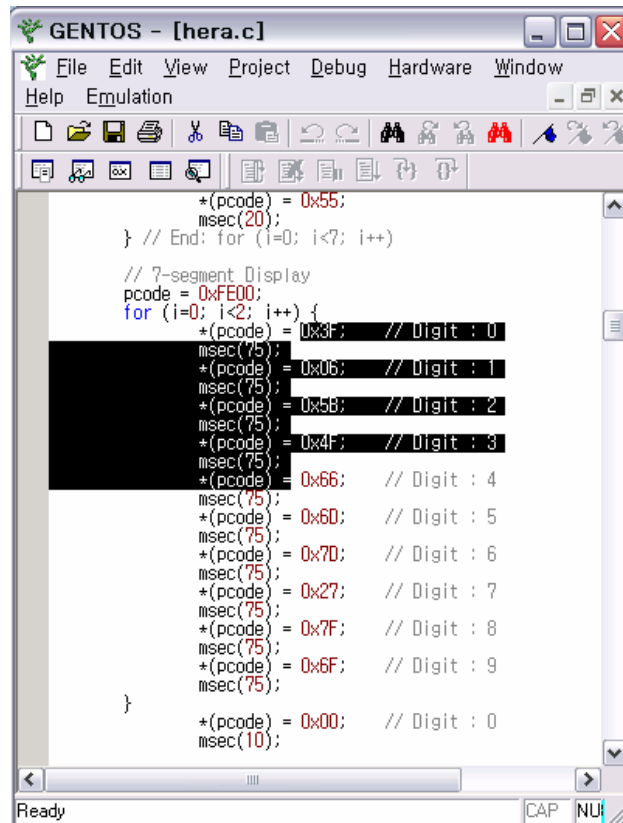


Figure 3-2 Block selection

#### 3.2.1.2 Selecting Text with Keyboard

- Position the insertion point (caret) on the place in the editor window where you wish the selected block to begin
- Press the 'Shift' key

- While keeping the 'Shift' key pressed, use movement keys to select the desired block to end,
- Release the 'Shift' key



```
GENTOS - [hera.c]
File Edit View Project Debug Hardware Window
Help Emulation
*(pcode) = 0x55;
msec(20);
} // End: for (i=0; i<7; i++)

// 7-segment Display
pcode = 0xFE00;
for (i=0; i<2; i++) {
*(pcode) = 0x3F; // Digit : 0
msec(75);
*(pcode) = 0x06; // Digit : 1
msec(75);
*(pcode) = 0x5B; // Digit : 2
msec(75);
*(pcode) = 0x4F; // Digit : 3
msec(75);
*(pcode) = 0x66; // Digit : 4
msec(75);
*(pcode) = 0x6D; // Digit : 5
msec(75);
*(pcode) = 0x7D; // Digit : 6
msec(75);
*(pcode) = 0x27; // Digit : 7
msec(75);
*(pcode) = 0x7F; // Digit : 8
msec(75);
*(pcode) = 0x6F; // Digit : 9
msec(75);
}
*(pcode) = 0x00; // Digit : 0
msec(10);
Ready CAP NU
```

Figure 3-3 Selected Text in edit window

### 3.2.2 Bookmarking the Text

When a text file is being edited, bookmarks can be set to the required locations. This enables you to jump to bookmarks easier and doesn't require you to remember the locations in the text file.

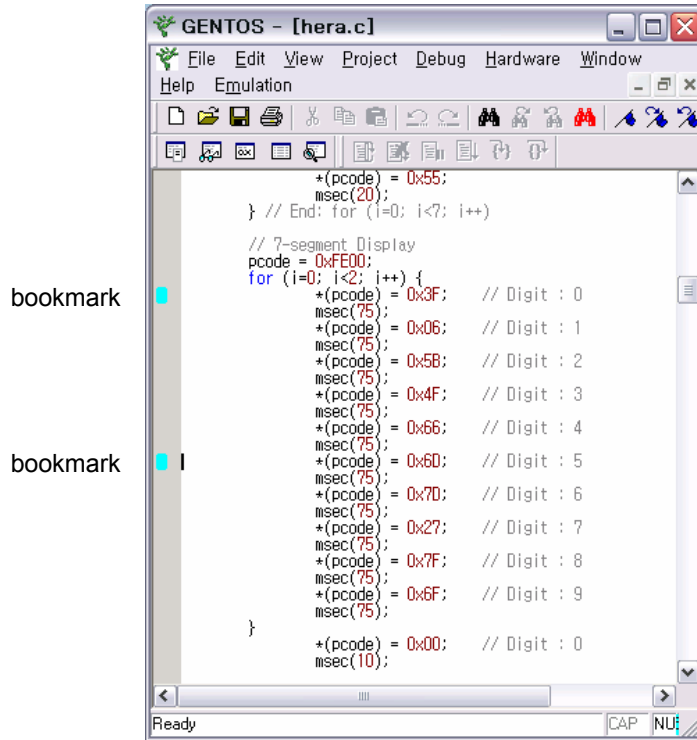


Figure 3-4 Source window with bookmarked locations

### 3.2.3 Search and Replace Operations

Search and replace operations can operate on an entire source file, or just on currently selected text (if any). You can search for whole words – characters in front or behind the searched word must not be identifier characters (letters, digits, and underscore), and it can be case sensitive or insensitive.

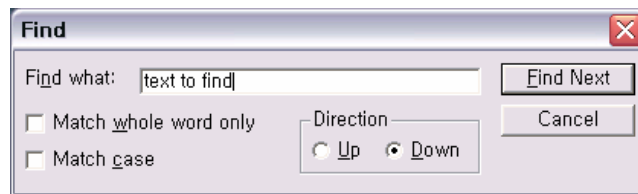


Figure 3-5 Find dialog

#### 3.2.3.1 Finding Text

- Open the 'Find' dialog box,
- Enter the string you wish to search for,
- Specify search options (direction, case, whole words)
- Select the 'Find Next' button to begin search

### 3.2.3.2 Replacing Text

- Open the 'Replace' dialog box,
- Enter the string you wish to search for and the string you wish to replace it with,
- Specify search options (scope, case, whole words)
- Select the 'Replace All' to automatically replace all occurrences, or use 'Find Next' and 'Replace' button confirm replace of individual occurrences

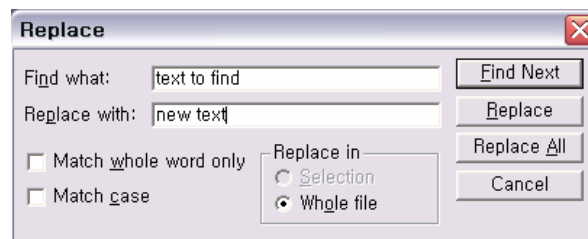


Figure 3-6 Replace dialog

## 3.2.4 Clipboard operations

GENTOS supports windows clipboard copy, paste and cut operations.

### 3.2.4.1 Copying Text to Clipboard

- Select a block of text
- Select the “Copy” sub-menu from the “Edit” menu or press the keyboard shortcut Ctrl+C.

### 3.2.4.2 Cutting Text to Clipboard

- Select a block of text
- Select the “Cut” sub-menu from the “Edit” menu or press the keyboard shortcut Ctrl+X. The selected block will be removed from the editor.

### 3.2.4.3 Pasting Text from Clipboard

- Make sure the desired text is placed in the clipboard.
- Position the insertion point to the place where you wish to insert the text.
- Select the “Paste” sub-menu from the “Edit” menu or press the keyboard shortcut Ctrl+V.

## 3.2.5 Drag and Drop

A quick alternative to clipboard's copy and paste operation when it comes to simple move or copy operations inside GENTOS is the usage of editor's drag and drop ability.

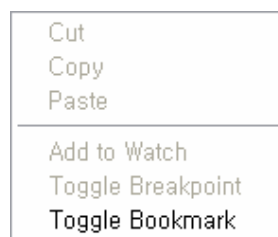
### 3.2.5.1 Moving Text using Drag and Drop

- Select a block of text
- Place the mouse cursor over the selected block of text – the text drag cursor will replace the I beam cursor
- Press the left mouse button
- While keeping the left mouse button pressed, move the mouse to the location where you wish to move the selected block
- Release the left mouse button

### 3.2.5.2 Copying Text using Drag and Drop

- Select a block of text
- Place the mouse cursor over the selected block of text – the text drag cursor will replace the I beam cursor
- Press the left mouse button
- While keeping the left mouse button pressed, move the mouse to the location where you wish to move the selected block
- Before releasing the left mouse button, press the 'Ctrl' key.
- While keeping the 'Ctrl' key pressed, release the left mouse button.

### 3.2.6 The Context Menu

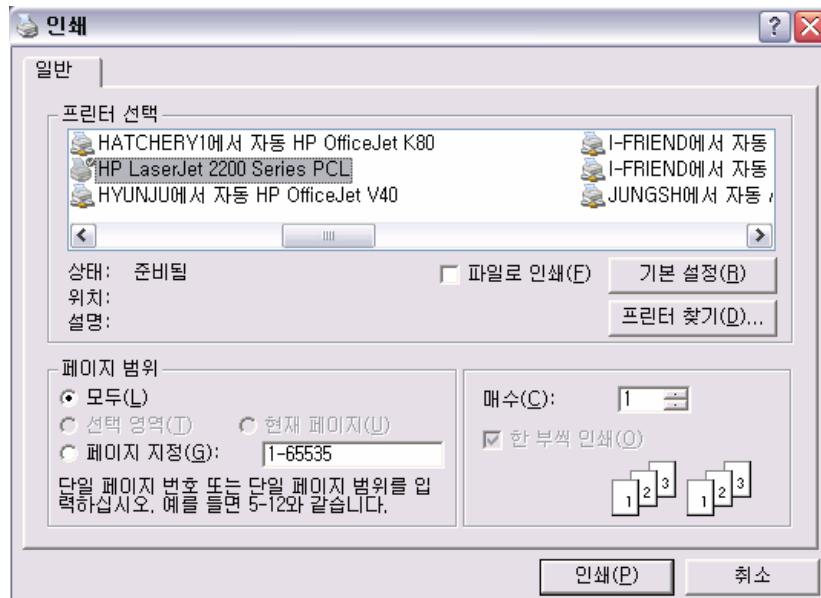


**Figure 3-7 Editor's context menu**

In the editor's popup context menu, you will find editor, build and debug commands that are used most often.

### 3.2.7 Printing

GENTOS supports simple printout of the current editor window or selection contents. In the Print dialog, the printer, its settings and print range are configured.



**Figure 3-8 Print dialog**

#### 3.2.7.1.1 Printer

Shows the printer on which the text will be printed. Use “Settings...” to change the printer or its properties.

#### 3.2.7.1.2 Range

Specifies whether the entire source file or only the current selection should be printed.

#### 3.2.7.1.3 Print

Starts printing.

#### 3.2.7.1.4 Orientation

To change the orientation, select Portrait or Landscape.

### 3.2.8 Configuring Editor Options

Several editor options can be customized. You will find them all in the editor pane of the Options dialog (Edit menu).

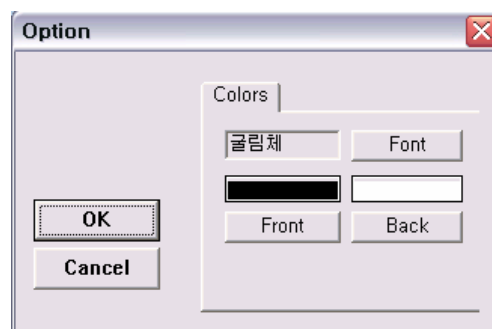


Figure 3-9 Editor options pane

### 3.2.8.1 Customizing Colors and Fonts

The colors can be customized to your specific requirements. This can be done in the “Edit/Options” pane.



Figure 3-10 Editor options, colors pane

For every different view a color and font can be set. The color of the foreground and the background can be selected using the appropriate tabs, and the font using the “Font & Size” tab.



## 4 Build Manager

### 4.1 Introduction

One of the goals in the development of GENTOS was to achieve a tight integration of all stages of program development (edit, build and debug) in a single application.

GENTOS provides its own powerful editing, compiling and debugging capabilities. GENTOS provides several specialized objects that facilitate this integration. You will manage your project hierarchy in the Project Workspace Window, run compilers using the Project Toolbar and configure compiler options in the Project Settings dialog.

In addition GENTOS separates project configuration settings from workspace settings in a separate project configuration file. This file is generated automatically when the workspace file (.gts) is generated.

#### 4.1.1 Detecting Changes to the Project

When the build manager is configured properly, it will check all project files and recompile/assemble those where:

- the object file doesn't exist
- the object file is older than the project file
- any of the header files used by a C project file is newer than the object file
- compiler/assembler settings for the project file have changed

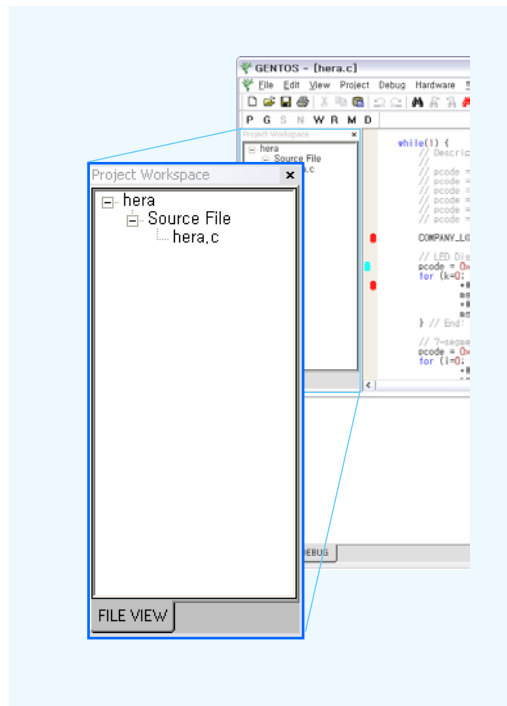
A project will be linked (output file generated) when:

- the output file doesn't exist
- any of the object files is newer than existing output file
- linker settings have changed
- the indirection file has changed

This check is performed before the CPU operation is advanced (download, step, etc.).

### 4.2 Project Organization

GENTOS adds two hierarchical levels between the projects and files that it consists of project files. These two levels are an abstract organizational form, which you can use if you feel the need to, or not use at all – especially for small projects.



**Figure 4-1 Project Workspace Window showing project hierarchy**

### 4.2.1 Project Files

Project files are the meat of your project. By compiling, assembling and linking them, your project is built.

GENTOS divides project files by their extension into four types:

- C-language source files (typically .C and .H extension)
- assembler source files (sometimes .H and .ASM extension)
- object and map files (.EXE or .MAP)
- Intel HEX files (.IHEX)

### 4.3 Compiler Tool-set Integration

Compared to other system development system, GENTOS can maximize its performance of Build Manager by even simple setting. GENTOS ships with ready-made example projects for all major compilers, which you can use as template workspaces for your new projects.

For quick starts and small projects, you can use an example workspace as a template for your workspace, for bigger projects however you will want to have more control over the Build Manager's settings.

You have already learned how to add targets and groups, what remains is the Project Settings dialog, where global, target and project file specific settings can be configured.

### 4.3.1 Compiler page

The compiler page contains options specific to the C compiler. If you are using only assembler, you do not need to set any options on this page.

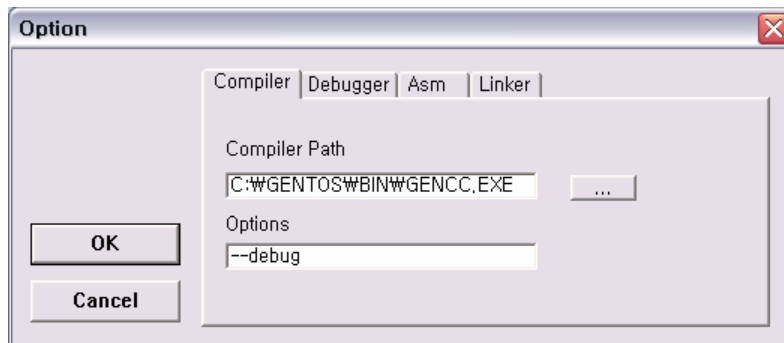


Figure 4-2 Project option dialog, Compiler page

#### 4.3.1.1 Compiler Path

Specifies the path to the compiler's EXE file.

This setting has global scope. The default compiler of GENTOS is "gcc.exe".

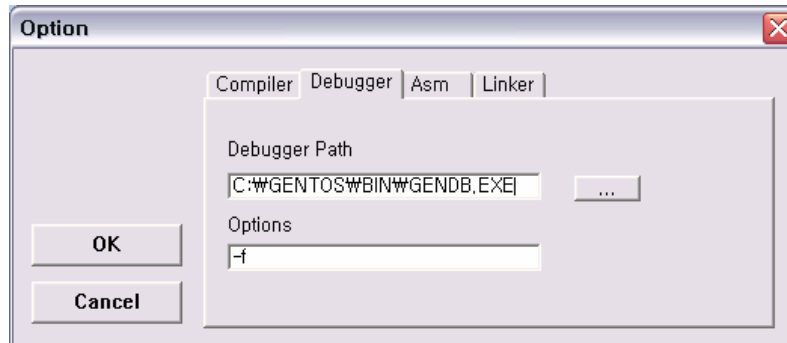
#### 4.3.1.2 Compiler Command Line Options

In this edit field you must enter command line options you wish the compiler to be called with when the selected file is compiled. These will usually be code generation options, conditional defines, etc.

This setting has file scope. The default option is "--debug" for debug information. See the manual "CoreRiver's Software Development Kit (SDK)" for more detail.

### 4.3.2 Debugger page

The debugger page contains options specific to the source-level debugger.



**Figure 4-3 Project option dialog, Debugger page**

#### 4.3.2.1 Debugger Path

Specifies the path to the debugger's EXE file.

This setting has global scope. The default debugger of GENTOS is “gendb.exe”.

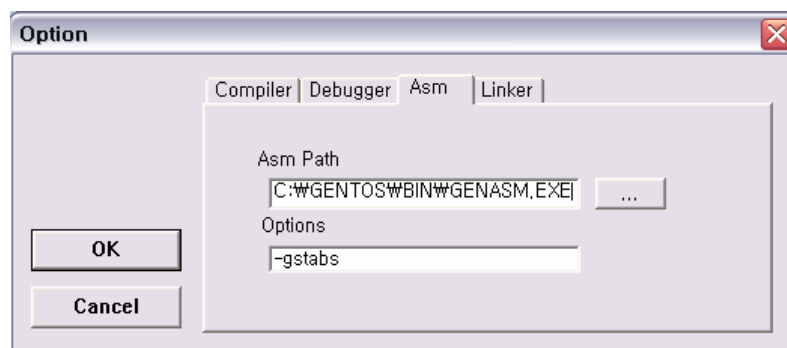
#### 4.3.2.2 Debugger Command Line Options

In this edit field you must enter command line options you wish the debugger to be called with in debugging time.

This setting has file scope. The default option is “-f” for the full file name and line number. See the manual “CoreRiver’s Software Development Kit (SDK)” for more detail.

#### 4.3.3 Assembler page

The assembler page contains options specific to the assembler. If you are using only compiler, you do not need to set any options on this page.



**Figure 4-4 Project option dialog, Assembler page**

#### 4.3.3.1 Assembler Path

Specifies the path to the assembler's EXE file.

This setting has global scope. The default assembler of GENTOS is "genasm.exe".

#### 4.3.3.2 Assembler Command Line Options

In this edit field you must enter command line options you wish the assembler to be called with when the selected file is assembled. These will usually be code generation options, conditional defines, etc.

The default option is "-gstabs" for stabs information. See the manual "CoreRiver's Software Development Kit (SDK)" for more detail.

#### 4.3.4 Linker page

The linker page contains options specific to the linker.

All settings on this page have global scope.

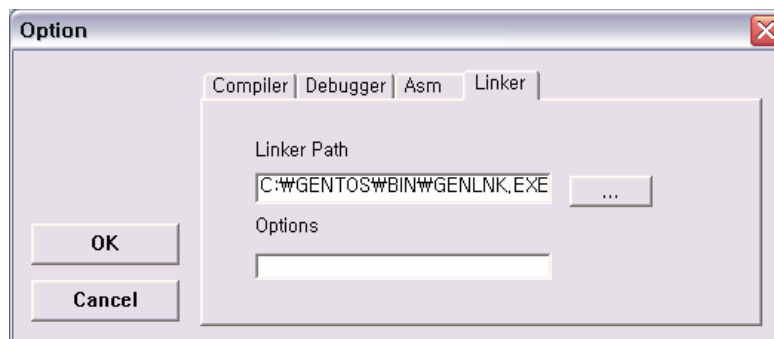


Figure 4-5 Project option dialog, Linker page

#### 4.3.4.1 Linker Path

Specifies the path to the linker's EXE file.

This setting has global scope. The default assembler of GENTOS is "genlnk.exe".

#### 4.3.4.2 Linker Command Line Options

In this field you must enter command line options you wish the linker to be called with. These will usually be only a path to the Indirection File with perhaps some additional options.

See the manual "CoreRiver's Software Development Kit (SDK)" for more detail.

## 4.4 Using an Example Workspace as a Template

You can make a quick start by modifying one of the example projects for the compiler that you use as follows:

- Use Windows Explorer to create a new directory for the new workspace
- Copy any project files you wish to include in your project from the example directory. This could be startup files, interrupt function examples, etc.
- Open the newly copied workspace in GENTOS.
- Remove unused example project files from the project files list
- Add your project files to the project files list
- Select the Build command to rebuild the entire project

## 4.5 Build Session

When the Build Manager processes a project file or links the project it creates a child process with redirected standard handles (STDOUT and STDERR).

The child process runs in a hidden window, so you do not get to see it while it runs. Any captured output is displayed in the output window after the process ends.

### 4.5.1 Output Window

The Output Window is a scrollable, terminal style window that shows raw output emitted by external tools spawned by GENTOS.

#### 4.5.1.1 Build Pane

The Build pane displays build progress and compiler, assembler and linker.

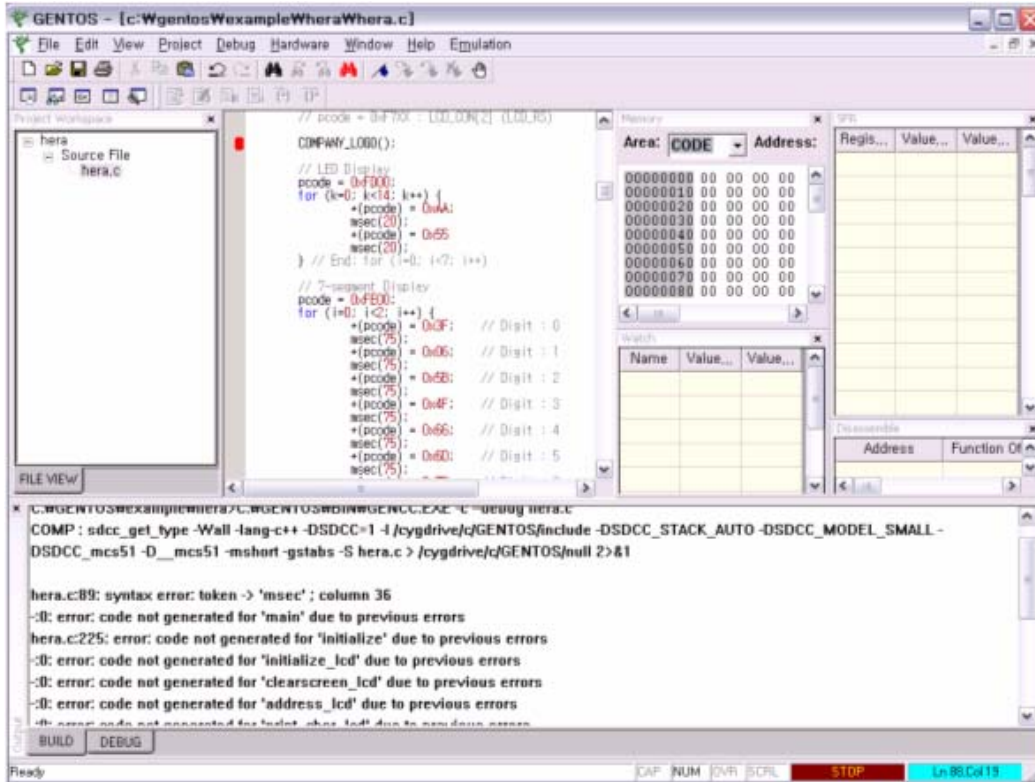


Figure 4-6 Output window, Build pane

The above figure shows a error message. By double clicking on it, the source file of the error is opened and positioned to the location of the error.

## 4.5.2 Building

GENTOS Build Manager can build on all modified project files (make) or all project files (rebuild).

### 4.5.2.1 Building All Modified Project Files

This is probably the build command you will be using most, since Build Manager automatically detects which files need to be built and performs all necessary actions to get the project up to date.

To build all modified project files, select the “Build” sub-menu from the “Project” menu or click the Make button on the Project toolbar

### 4.5.2.2 Building All Project Files

To build all project files, select the “Build” sub-menu from the “Project” menu or click the build button on the Project toolbar.





## 5 Debug Session

### 5.1 Download Files

When working without an emulator, the program for the target CPU is usually burnt into an EPROM, programmed in a FLASH device, etc. In an emulator system the emulator provides the emulation memory, where the program is loaded and executed by the target. This is either true emulation memory on an In-Circuit emulator, memory in simulated target devices, or regular target RAM devices that the debugger can access (usually by means of CPU's BDM or JTAG interface).

The files that you usually use to program the target EPROM or FLASH are loaded into this emulation memory and are called download files. Although you will usually be using only a single download file, you can define any number you wish.

**Note:** for EPROMs usually binary, Intel or Motorola hex files are used. Although you can use them with emulator as well, they do not contain any debug information. To use debug information you need to use linker's output format that contains it. Most linkers generate such output file by default and object to hex converters are then employed to generate PROM-able files.

To configure download files select the "Download Files" sub-menu on the "Debug" menu.

### 5.2 Breakpoints

Besides reading and writing memory, debugging is all about breakpoints.

A breakpoint is a way to stop the CPU. There are various kinds of breakpoints available:

- **Implicit execution breakpoints** are transparent to users, but used heavily for step, step over, run until and run until return operations.
- **Execution breakpoints** are used to break CPU execution before it executes the instruction on the breakpoint address.
- **Hardware specific breakpoints** depend on the capabilities of the attached debugger. Refer to manual "CoreRiver's Software Development Kit (SDK)" for more information.

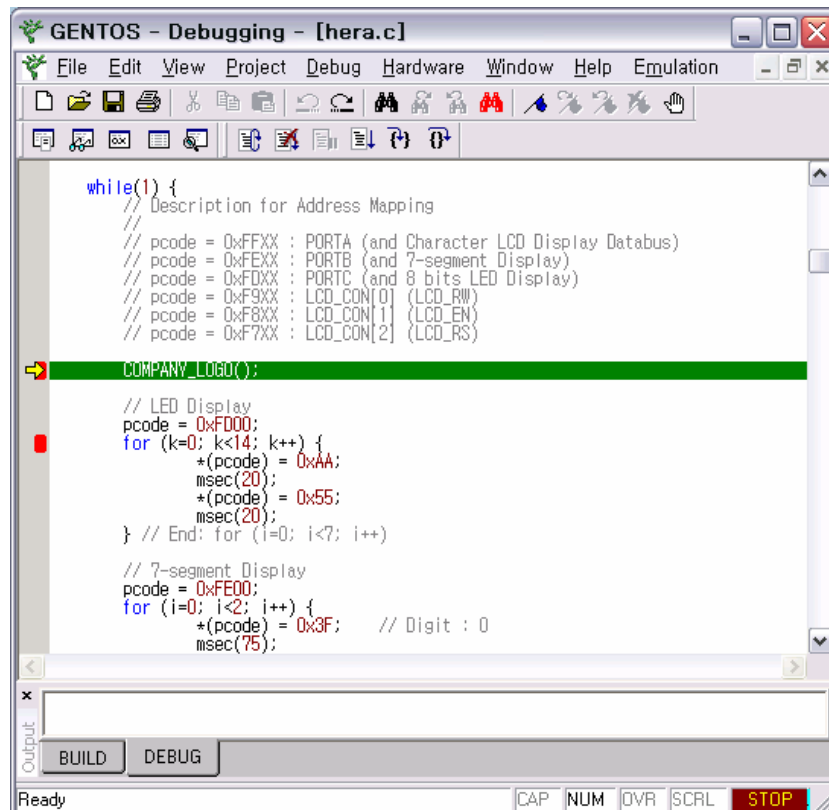
When a breakpoint is hit, the CPU is stopped and GENTOS updates its windows to reflect the new situation.

#### 5.2.1 Execution Breakpoints

Execution breakpoints act when the CPU attempts to execute the instruction at the breakpoint

location.

This is the most commonly used breakpoint type. GENTOS makes extensive use of them to implement functions like 'Step Into', 'Step Over', and 'Run'. These implicitly used breakpoints are not visible to you.



**Figure 5-1 Execution breakpoint indicated in source**

In situations where simple stepping through a program is not enough, you can explicitly set an execution breakpoint.

An execution breakpoint can be set in two ways.

The quick and easy way is to:

- use a shortcut key or,
- select the 'Toggle breakpoint' command from editor's context menu.

This command will set an execution breakpoint at the insertion point of the editor.

### 5.2.2 Clear All Breakpoints

Clear All Breakpoints commands make all breakpoints disable. Especially useful when final application tests are performed.

### 5.2.3 H/W Breakpoints

H/W Breakpoints commands provides a link to a dialog where hardware breakpoints can be configured.

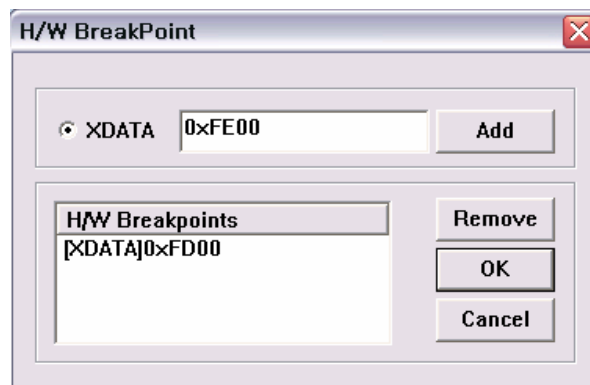


Figure 5-2 H/W Breakpoint dialog

Current GENTOS supports only the hardware breakpoint for accessing the external data memory.

## 5.3 Source Debugging

If your compiler can generate source debug information, you will probably spend most of the time debugging in your own sources.

GENTOS provides you with quick yet powerful functions to control program flow. All commands are available in the Debug menu as well as in the Debug toolbar.

**Note:** no debugging is available until the emulator has been initialized and the program loaded.

### 5.3.1 Go

Go command sets the CPU running.

This command is available if the CPU is currently stopped or when the debugger starts.

### **5.3.2 Reset & Run**

Reset & Run command will place the CPU in reset state, release it from reset and it executes the first instruction.

### **5.3.3 Stop Debugging**

Stop Debugging command exits the debugger.

This command is available only if the CPU stops.

### **5.3.4 Stop**

Stop command stops the CPU immediately.

This command is available only if the CPU is running.

### **5.3.5 Reset**

Reset command will place the CPU in reset state, release it from reset and stop it before it executes the first instruction.

For CPU's that need to perform chip initialization within specified number of clocks after the CPU has been released from reset, use the Reset and Run command, which does not stop the CPU after it is released from reset.

### **5.3.6 Step Into**

Step Into command performs a single program step. When calling a function, the function will be stepped in.

If the CPU is running, step into command causes the CPU to stop when it reaches the next source line.

If interrupt servicing in background has been disabled, no interrupt will be serviced during this step.

### **5.3.7 Step Over**

Step Over command performs a single program step inside the current function. Any function(s) called by executing the current line are not stepped in.

If interrupt servicing in background has been disabled, no interrupt will be serviced during this step.

This command is available if the CPU is currently stopped.

### 5.3.8 Stepi

Stepi command performs a single MCU instruction. When calling a function, the function will be stepped in.

If the CPU is running, step into command causes the CPU to stop when it reaches the next MCU instruction.

If interrupt servicing in background has been disabled, no interrupt will be serviced during this step.

### 5.3.9 Setting Execution Breakpoints

Setting and clearing an execution breakpoint in source is easy. The 'Breakpoint' command available in Debug or "Toggle Breakpoint" command in editor's context menu sets an execution breakpoint at the current insertion point. If a breakpoint is already set on this location, it is cleared.

## 5.4 Disassembly Window

Programmers will mainly use the disassembly window where no source debug information is available. Its functions nearly duplicate source-debugging functions.

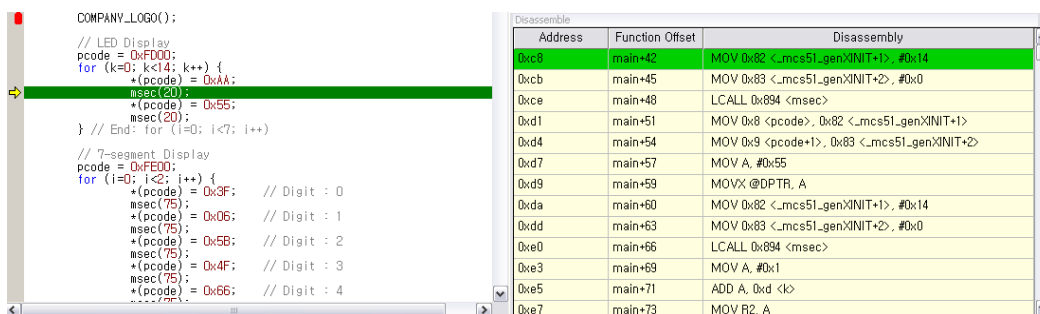


Figure 5-3 Disassembly window

Disassembly window lists the target program in disassembled form. The program execution point and breakpoints are indicated, as well as an arrow indicator with which you can move up and down to view other parts of the program, set and clear breakpoints, run until, etc.

Disassembly pane always displays disassembled instruction, but you can additionally instruct it to display:

- source - if a line symbol's address matches the address of the disassembled instruction, the source line is displayed before the instruction.
- labels - if a code label's address matches the address of the disassembled instruction, the

name of the label is displayed before the instruction.

- symbols - if an absolute location is addressed by an instruction and that address matches a global variable or a code label, the symbol's address is displayed instead of the value.

Example: variable 'var' is located at address 1000h:

```
MOV A, (1000)
```

is replaced by

```
MOV A, (var)
```

- symbol values - if symbols are enabled and you still wish to see the absolute number, enabling this setting will display the value as well. In the previous example this would yield:

```
MOV A, (var) (1000)
```

## 5.5 Memory Window

Memory windows are best suited for a raw view of the CPU's memory. GENTOS can show three memory areas (CODE, IDATA and XDATA) by selecting area.

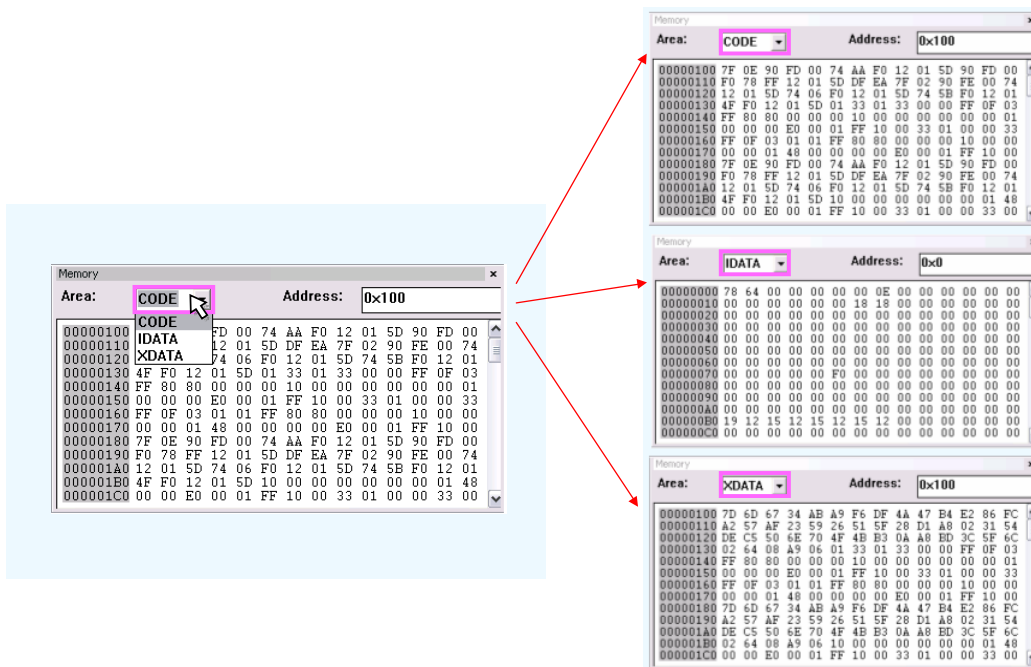


Figure 5-4 Memory Window

In a memory window the following elements are visible:

- Memory area selector - shows which of the CPU's memory areas is currently displayed

- List field - is used to enter the address from which you wish to list the memory.
- Address column - displays the address of the first item in the numerical display area at the same line
- Numerical display area - displays memory contents in binary, hexadecimal or decimal integer or floating point format.
- ASCII display area - displays memory contents in ASCII format

The memory window will also color locations that have changed in the previous action, and indicate the location of the stack pointer when in range.

### 5.5.1 Opening a Memory Window

A memory window is opened by selecting the Memory Window command from the View menu or by clicking the memory window button on View toolbar.

### 5.5.2 Finding Memory Contents

A very helpful function for browsing the memory is to find memory contents. After entering the start address of searching range, the contents of the memory window are updated. Or you can find the wanted memory contents by scroll the memory window.

### 5.5.3 Modifying Memory Contents

Memory contents can be modified directly in the numerical display area.

To modify a location:

- make sure it is visible
- click on it with the left mouse button - a block caret will appear covering the selected location
- enter the new value.

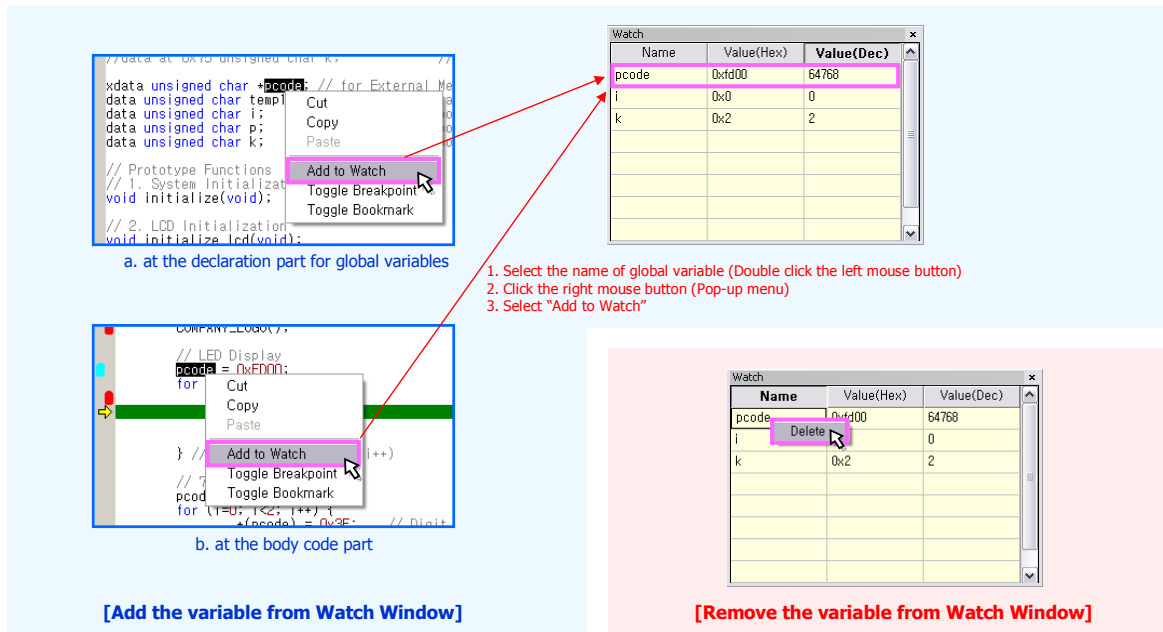
**Note:** If the memory does not change, the CPU might have problems with write access to that location.

When entering value in the hexadecimal display types, individual digits can be modified directly.

## 5.6 Watch Window

The watch window is best suited for watching and modifying high level variables - symbols with

associated type. You do not need to bother with their locations; you only need to specify their name.



**Figure 5-5 Watch Window**

The watch window consists of four functionally identical panes, which are easily switched by clicking on the appropriate pane selector. This way you can configure a larger number of watches without having to constantly scroll.

### 5.6.1 Adding Watch Expressions

To configure a new watch expression you can:

- select Add Watch... command from editor's context menu

or

- select the expression you wish to watch in the editor and drag it to the watch window

### 5.6.2 Modifying Values

GENTOS can only modify values of expressions that evaluate to a target system address (also called Lvalues). You can not modify constant expressions or Rvalues.

To modify value of an expression you can:

- click in the value column of the expression you wish to modify



- enter the new value. This can be any valid watch expression.

**Note:** If the expression does not change, the CPU might have problems with write access to that location.

### 5.6.3 Watching Complex Expressions

Complex data types can be watched by expanding the tree box expression.

Structures, unions and classes members will also be shown. If these are complex as well, they can be expanded further until a simple type is reached.

When expanding a pointer the value that the pointer points to is shown in the expanded leaf. This can again be expanded until a simple type is reached.

## 5.7 Special Function Registers Window

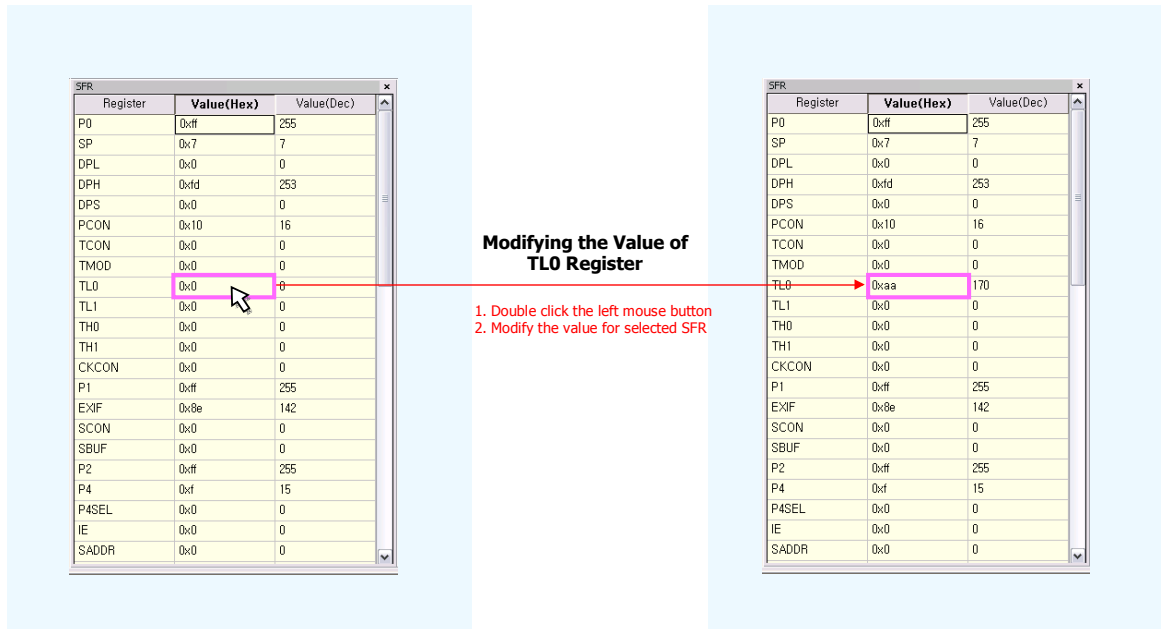
GENTOS provides a specialized window (SFR window) to display CPU's onchip special function registers.

To open the SFR window, select the "Register Window" sub-menu from the View menu, or click the Register window icon on the view toolbar.

The SFR window organizes CPU's registers hierarchically into two levels of groups (groups and sub groups) and two levels of registers (registers and sub registers).

A group is a collection of registers that serve a module on the CPU. If you are currently working on serial communication, you will see all involved registers by expanding the 'Serial Port' group.

When there are many (nearly) identical CPU modules, a group will contain a sub-group for every such module. In the figure below, the 'General Purpose Registers' group contains all 4 register bank sub groups as well as the 'General Purpose Registers' sub group, which duplicates registers of the currently active, register bank.



**Figure 5-6 Register Window showing CoreRiver MCU's registers**

A register corresponds to a CPU's special function register. Along with the name, its current value is displayed.

### 5.7.1 Modifying a register

To modify register value, double click on its value. In the in-place edit field that opens enter the new value using the same number base as the register is displayed in.

**Note:** whenever possible, sub registers are modified without writing the entire register (bit addressable registers). If a sub register can not be accessed independently of its register, the register is first read, sub register bits are adjusted, and the whole register is written back.

## 6 Hardware Configuration

After the Emulator has been connected to the PC, you should also configure necessary settings in the software to work with attached hardware components. To do so, select 'Hardware Configuration' command from the Hardware menu.

### 6.1 Debug Hardware and Debug Mode

On the Hardware Type page you must specify what hardware you are using and, which of its available emulation modes you wish to use.

#### 6.1.1 PC to Emulator Communication

On the communication menu specify the communication port where the Emulator is attached.

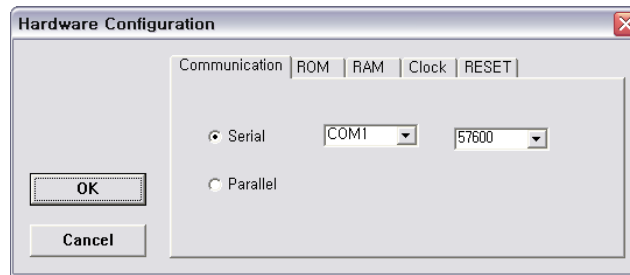


Figure 6-1 Hardware Configuration dialog, Communication page

- Serial (COM) : select this when Emulator is attached to the PC's COM port. Also specify the number of the port. Use as a last resort. Serial communication can be 5-20 times slower than parallel. Baudrate 57600 in COM1 port is default setting.
- Parallel (LPT) : select this when the Emulator is attached to the PC's LPT port. Also specify the number of the port. If possible make sure that your parallel port is configured for bi-directional communication. This will greatly increase Emulator to PC communication speed. But parallel communication is not supported in current GENTOS.

#### 6.1.2 Configuration

All In-Circuit Emulator operation settings are configured in the 'Hardware Configuration' dialog. You will usually configure these options only once per project.

You may find some settings redundant since an Emulator probably knows how much memory it has

or what POD is attached to it, however since settings, which are not detectable rely on these parameters and to allow configuration without an Emulator attached, these parameters have to be specified as well.

### 6.1.2.1 ROM

The ROM Setup page determines the location of ROM: emulator ROM or target ROM. The default location for ROM is emulator.

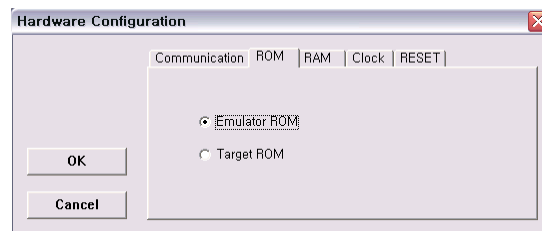


Figure 6-2 Hardware Configuration dialog, ROM setup page

### 6.1.2.2 RAM

The RAM Setup page determines the location of RAM: emulator RAM or target RAM. The default location for RAM is target to use the external access by MOVX instruction.

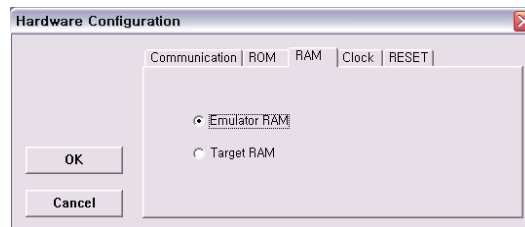
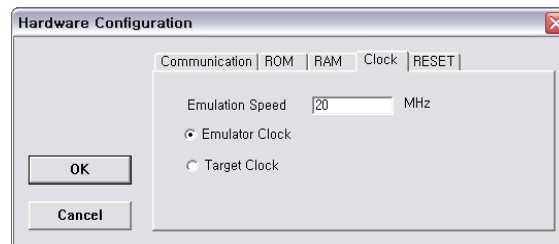


Figure 6-3 Hardware Configuration dialog, RAM setup page

### 6.1.2.3 Clock

The Clock Setup page determines the CPU's clock source and its frequency.



**Figure 6-4 Hardware Configuration dialog, Clock setup page**

**Note:** When either of these settings is set to target the corresponding line is routed directly to the CPU from the target system.

Clock source can be either used from the emulator or from the target. It is recommended to use the emulator clock when possible. When using the clock from the target, it may happen that the emulator cannot initialize any more.

It is dissuaded to use a crystal in the target as a clock source during the emulation. It is recommended that the oscillator be used instead. Normally, a crystal and two capacitors are connected to the CPU's clock inputs in the target application as stated in the CPU datasheets. A length of clock paths is critical and must be taken into consideration when designing the target. During the emulation, the distance between the crystal in the target and the CPU (on the POD) is furthermore increased, therefore the impedance may change in a manner that the crystal doesn't oscillate anymore. In such case, a standalone crystal circuit, oscillating already without the CPU must be built or an oscillator must be used.

When the clock source is set to from emulator, you will be able to use clock from 1MHz to 20 MHz.

**Note:** The clock frequency is the frequency of the signal on the CPU's clock input pin. Any internal manipulation of it (division or multiplication) depends entirely on the emulated CPU.

If the clock source is set to from target, the clock is provided by the target system. In certain application, a clock below 1MHz is used. Since the minimal clock the Emulator can generate is 1MHz, an target clock source must be used and the clock source set to external.

#### 6.1.2.4 Reset

When checked, then the target's RESET line can reset the CPU. This option is available only when the CPU is in running. Default RESET is in emulator.

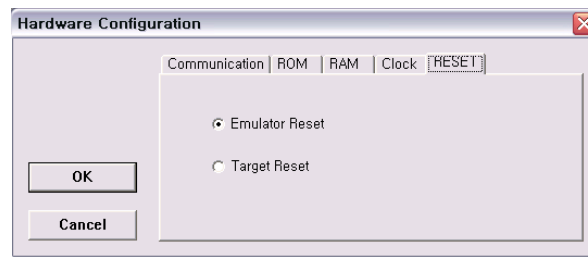


Figure 6-5 Hardware Configuration dialog, Reset setup page

## 6.2 Emulation with Target System

First we will use the example project, already used above.

1. Make sure that both your target system and the Emulator are switched off.
2. Remove the CPU from the target system.
3. Insert the emulation POD in its place. Take special precaution to insert the POD correctly.
4. Connect any extra additional lines
5. Switch on the Emulator
6. Switch on the target system
7. Perform a download.
8. Try out some debugger features.

After you are finished:

1. Switch the target system off
2. Switch the Emulator off

**Important:** To avoid damage to Emulator, you must never operate with Emulator switched off and target system switched on. It is recommended that both Emulator and target system power supply be operated with a single power switch.

## 6.3 Emulation with Target Code

To try whether your target functions with the Emulator:

1. Keep clock set to from emulator, since long lines and adapters that are used between Emulator and target system can distort the clock waveform and fail the emulation.
2. Connect the RESET output line signal on the POD to the target's resetting signal, this way peripheral devices will be reset, whenever the Emulator resets the CPU.

3. Enable target interrupts by installing jumpers between POD signals and target interrupt lines (for example INT0, INT1, INT2, and TINT0, TINT1, and TINT2 respectively).
4. Remove all download files from the download files list. Close the dialog.
5. Execute the 'Reset and Run' command from Debug menu.
6. The hardware should now initialize and when finished, the CPU should be put into running.
7. The target should now behave as if running standalone.





## 7 Technical Notes

### 7.1 Differences from a standard environment

The In-Circuit Emulator and the Active Emulator can emulate a processor or a micro-controller. Beside the CPU, additional logic is integrated on the POD. The amount of additional logic depends on the emulated CPU and the type of emulation. A buffer on a data bus is always used (minimal logic) and when rebuilding ports on the POD, maximum logic is used. As soon as a POD is inserted in the target instead of the CPU, electrical and timing characteristics are changed. Different electrical and timing characteristics of used elements on the POD and prolonged lines from the target to the CPU on the POD contribute to different target (the whole system) characteristics. Consequently, signal cross-talks and reflections can occur, capacitance changes, etc.

Beside that, pull-up and pull-down resistors are added to some signals. Pull-up/pull-down resistors are required to define the inactive state of signals like reset and interrupt inputs, while the POD is not connected to the target. Because of this, the POD can operate as standalone without the target.

### 7.2 Common Guidelines

Here are some general guidelines that you should follow.

- Use target Vcc/GND if possible (to prevent GND bouncing),
- Make an additional GND connection from POD to the target if the Emulator behaves strangely,
- Use the reset output line on the POD to reset the target whenever Emulator resets the CPU,
- No on-chip or external watchdog timers can be used during emulation (unless explicitly permitted). Disable them all.
- When interrupts in background are enabled, take note that the interrupt routine must return in 25 ms, otherwise the Emulator will assume that the program is hung.

### 7.3 Port Replacement Information

In general, when emulating the single chip mode, some ports have to be rebuilt on the POD because original ports are used for emulation – typically ports used as address and data bus in extended mode. Special devices, so called port replacement units, provided already by the CPU vendor or other standard integrated circuits are used to rebuild "lost" ports. Rebuilt ports are logically compatible with original CPU's ports, but electrical characteristics may differ. If a special

device (the port replacement unit (PRU), available from the CPU manufacturer) is available, electrical characteristics don't differ much and usually the user doesn't have to pay attention. The differences may become relevant when standard integrated circuits are used and operating close to electrical limits, e.g. when input voltage level is close to specified maximum voltage for low input level ("0") or specified minimum voltage for high input level ("1") or if, for example, the target is built in the way that the maximum port input current must be considered.

## 7.4 Interrupts

### 7.4.1 Interrupt Handling When Stopped

While the user's program is stopped, interrupts can be disabled or enabled.

#### **Interrupts Enabled When Stopped" checked**

When this option is checked, the Interrupt Enable flag is never modified by the emulator. When the user's program is stopped the emulator doesn't influence the state of Interrupt Enable flag.

**Note:** On all 8 bit CPUs the emulator allows interrupt nesting up to 15 levels in depth, representing no limitations in practice. Nesting will occur only if interrupt servicing is interrupted by another interrupt before the servicing is completed. While any nested interrupt is serviced by the CPU, the emulator has no access to the CPU therefore debug windows cannot be refreshed in the meantime.

To allow background interrupt execution on 8 bit CPUs, interrupt routines must meet the following conditions:

- All CPU registers must be preserved,
- Interrupt routines must return with the corresponding return-from interrupt instruction (RETI, RFI, etc.). Do not assume that your compiler always gets it right. Interrupt routine exiting with jump or call instruction cannot be debugged.
- The return address must not be changed in the interrupt routine.

#### **Interrupts Enabled When Stopped" unchecked**

After the user's program is stopped (STOP), the emulator remembers the current Interrupt Enable flag status and disables interrupts. When the program is set back to run, the emulator restores the interrupts (Interrupt Enable flag) back and proceeds with program execution (RUN).

There is no problem when the 'Run' command is being used, but a problem can occur under certain

conditions when a single step command is being used.

While in stop and executing a single step in the disassembly window there are no problems. During single step in the disassembly window the emulator itself detects any instruction that changes the state of Interrupt Enable flag and handles it correctly.

For example, interrupts are active and the program is stopped. The emulator remembers the Interrupt Enable flag state and disables interrupts. Now the user executes single steps in the disassembly window and, for example, once the SWI instruction (software interrupt) is stepped. At this moment, the CPU pushes the content of the CCR register to the stack, where the Interrupt Enable flag is stored and jumps to the address where the interrupt vector points to. Before the user's program was stopped (from running), the interrupts were active (Interrupt Enable flag) and after the program was stopped, they were disabled (Interrupt Enable flag) by the emulator. Therefore an incorrect Interrupt Enable flag value (CCR) is now pushed to the stack. Since the emulator can detect such an instruction it modifies the stack with the proper Interrupt Enable value. If this would not be done, the program execution would be changed after RETI instruction in the software interrupt routine is executed. Interrupts in the user's program would now be disabled and not enabled as before while the program was running.

When using step in the source window the above-mentioned problem becomes relevant and the user should never forget it. The source step is actually executed with RUN command with prior setting of breakpoint on the required source line. If SWI (software interrupt) occurs during one source step the CCR with disabled interrupts will be pushed to the stack and after returning from software interrupt routine (RETI) the same value is popped up from the stack. When the user re-runs his program, interrupts are disabled and not enabled, as before the user's program was stopped.

During the source step the emulator cannot detect instructions that changes the state of Interrupt Enable flag as it is the case with single step in the disassembly window.

#### **7.4.2 Debugging Interrupt Routines**

An interrupt routine can only be debugged when the interrupt source for this routine has been disabled, otherwise you will keep reentering the routine and thus run out of system stack.

For example, you have an interrupt routine with 10 source lines. Let's say the interrupt routine is called continuously - e.g. that one timer is the source for this interrupt. Breakpoint is set on the first source line of the interrupt routine. Program execution stops on breakpoint. Now the source step is

executed. Source step is actually executed using RUN command with prior setting of breakpoint on the required source line. In this particular case, while the source step is executed, the CPU executes the code and before the source step finishes, a new interrupt call occurs. New values are pushed on to the stack and the CPU stops on the breakpoint again. If you repeat source steps in such interrupt routine new values are pushed to the stack and you can easily run out of stack.

An interrupt source can be disabled in two ways:

- Disable the interrupt process in the stopped mode. The stopped mode is entered whenever CPU is stopped, and the Emulator remains in stopped mode until the Run command is executed. (During Step, Step over, etc. commands, the stopped mode persists).
- Do not place a breakpoint on any instruction in the interrupt routine where interrupts are not yet disabled. Also, you must not step over any instruction that re-enables the current interrupt, but run the program before the instruction is executed.

## 7.5 Memory Access

When monitor access to the CPU's memory is requested, the emulator stops the CPU and instructs it to read the requested number of bytes. Since all accesses are performed using the CPU, all memory available to the CPU can be accessed. The drawback to this method is that memory cannot be accessed while the CPU is running. Stopping the CPU, accessing memory and running the CPU is an option, which, however, affects the real time execution considerably.

The time the CPU is stopped for is relative and cannot be exactly determined. The software has full control over it. It stops the CPU, updates all required windows and sets the CPU back to running. Therefore the time depends on the communication type used, PC's frequency, CPU's clock, number of updated memory locations (memory window, SFR window, watches, variables window), etc.