SEGA™

Katana CodeScape User Guide

CodeScape for Set 5

User Guide

Version 2.1.0a Build 86

Cross Products Ltd
23 The Calls, Leeds, West Yorkshire, LS2 7EH
telephone: +44 113 242 9814
facsimile: +44 113 242 6163
www.crossprod.co.uk
email sales: enquiry@crossprod.co.uk
email support: support@crossprod.co.uk

Legal Notice

IMPORTANT

The information contained in this publication is subject to change without notice. This publication is supplied "as is" without warranty of any kind, either express or implied, including but not limited to the implied warranties or conditions of merchantability or fitness for a particular purpose. In no event shall Cross Products be liable for errors contained herein or for incidental or consequential damages, including lost profits, in connection with the performance or use of this material whether based on warranty, contract, or other legal theory.

This publication contains proprietary information which is protected by copyright. No part of this publication may be reproduced in any form, or stored in a database or retrieval system, or transmitted or distributed in any form by any means, electronic, mechanical photocopying, recording, or otherwise, without the prior permission of Cross Products Limited.

CodeScape User Guide

Revision History

Version 2.1.0a, 86 July 1998

Release Candidate 1, 7 October 1996, Beta 2, 26 August 1996, Beta 1, 26 July 1996 - 97, 2.0.0, March1998, Version 2.0.5a, May 1998

Contents

Before you begin

Document conventions

NOTE: Notes call attention to important features or instructions.

Typographic conventions in this user guide

Convention	Description
SPACEBAR	Capital letters are used for the names of keys on the keyboard, filenames, and extensions.
ALT+F4	If two or more keys must be pressed simultaneously, the keys are linked with a plus sign (+).
ALT, F, X	If two or more keys must be pressed in succession, the keys are linked with a comma (,).
select this box	Italics are used to denote text boxes and check boxes that are on CodeScape's interface.
emphasis	Bold text is used to denote emphasis.
`input and output`	This font is used to denote, user input, and program output including error messages.
`command-line`	This font is used for command-line options.

This release

The CodeScape 2.1.0 release includes:

- A CD-ROM that contains the CodeScape debugger, and online documentation including context-sensitive help.
- A printed hardware setup guide, a software reference manual, and two tutorials.

NOTE: Contact Technical Support if any of these items are missing.

Audience

This manual is for programmers that write for targets under Windows® 95, or Windows NT™ 3.51/ 4.0. It will also be of use to technical support and software test engineers.

This guide

Using and configuring the interface introduces CodeScape's debugging environment and how to use it. It describes the commands on the menu bar, toolbars, and shortcut menus. It explains how to set up and use windows and regions for your project.

Interacting with target processors explains how to connect to, initialize and reset a target processor. This includes: restarting a program, and saving and loading binary parts of a program.

Working with sessions explains how to use the commands for working with sessions. This includes how to: open new and existing sessions, save a session, save a session with a new name, close a session, view and use the recently used files list, and exit CodeScape.

Working with projects describes how to set up and use your project build environment including how to set up an MS-DOS or Windows editor. It also tells you how to build your project within CodeScape and what to do if CodeScape returns errors after a compile.

Debugging project files explains the various ways you can debug your project files, including: stepping source code, setting watch and data breakpoints, and simulating assembly code.

Using the command-line describes all of the command-line commands.

The CodeScape software

CodeScape is a fast, intuitive, Windows-based debugging tool that runs on Windows® 95, or Windows NT™ 3.51/4.0 machines.

To run, CodeScape requires:

- An IBM™ PC or compatible with Pentium™ 90 processor or above.
- Windows® 95 or Windows NT™ 3.51/4.0.
- 32 MB or more of RAM.

CodeScape's debugging features let you find, isolate, and fix bugs in your original source, or disassembled code.

Run CodeScape to:

1) Edit your project files.
2) Compile and link your project files.
3) Debug your project and test it for errors.
4) Then do one of the following:
    - If debugging returns errors, repeat steps 1 to 3 above.
      -OR-
    - If debugging does not return errors, Build your project.

Using and configuring the interface

You can control CodeScape using a mouse or keyboard. CodeScape has many useful toolbars that can be docked, floating, or hidden. Region specific shortcut menus are available for the most used functions.

The user interface consists of:

- The Menu bar.
- Toolbars.
- Windows.
- Regions.

The commands on the menu bar

File menu ALT+F

The commands in the File menu are for working with sessions, resetting the target, loading program files, restarting program file execution, and saving and loading binary information. Use Save binary and Load binary to move large blocks of data in and out of memory.

The File menu also displays a list of recently used session files. You cannot hide this list or change the number of files displayed.

NOTE: When you load a new session, or exit CodeScape, a message appears prompting you to save any changes to the current session.

Edit menu ALT+E

The commands in the Edit menu are for cutting and pasting in the Edit region, and for performing searches. The Edit menu becomes available when you open a window, create a region, or load a session.

View menu ALT+V

The commands in the View menu are for showing and hiding toolbars, and configuring regions.

Project menu ALT+P

The commands in the Project menu are for configuring the current project and building it from within CodeScape.

Debug menu ALT+D

The commands in the Debug menu are for controlling program execution, stepping code, and using breakpoints. You can set the cursor to the PC (program counter) and vice-versa. The default origin is set to the value of the PC. You can also lock the view to an Expression that contains the PC, a register, or memory.

You can simulate a target’s processor operations on your computer. All of CodeScape’s debugging functions are available when the Simulator is running. You can also run the Profiler to examine the run-time behavior of your program file.

Region menu ALT+R

The commands in the Region menu are for splitting (creating new) regions, changing a region’s type, and updating all regions. The Region menu is available when you open or create a window, region, or session.

Window menu ALT+W

Use the Window menu to open a new window. When you open a new window, the Edit and Region menus appear on the Menu bar, and additional commands appear on the Window menu which are for arranging multiple windows in CodeScape.

The Window menu also displays a list of region types and highlights the currently active region. When you have more than one region in a window frame, the active region within that frame appears in the list. You cannot hide this list or change the number of regions displayed.

Help menu F1

Use the Help menu to get on-line Help and view CodeScape version information.

Region specific shortcut menus

Each region has two shortcut menus:

1) The Region Type menu has commands for changing the region’s type. To see the menu:
   - Press CTRL+SHIFT+F10.

     -OR-
   - CTRL+Right-click anywhere in the region.
2) The Region Actions menu has region specific commands, and global commands for controlling program execution or manipulating breakpoints. To see the menu:
   - Press SHIFT+F10.

     -OR-
   - Right-click anywhere in the region.

The commands on the toolbars

The toolbars provide access to the main debugging functions. To use the *Toolbar Configuration* check box to show or hide toolbars:

- Right-click the status bar.
  -OR-
- Click View, Toolbar.

Then select or deselect toolbars from the list.

Toolbars and their uses

Use the:	For:
Breakpoint toolbar	Point and click access to the most common breakpoint actions.
Debug toolbar	Point and click access for debugging actions.
Processor Combo toolbar	Point and click access for selecting a target processor, configuring a target processor, and loading program files.
Project Build window	Viewing status information about a project build.
Region toolbar	Setting and changing a region’s type.
Region Combo toolbar	Setting the rate at which a region’s display is updated, and changing a region’s type.
Splitter toolbar	Splitting regions using the mouse.
Standard toolbar	Point and click access for opening a new window, and creating, opening, and saving Sessions.
Target window	Viewing the current processor for the current target. The toolbar also provides point and click access for loading program files and configuring the current processor.
Target Combo toolbar	Viewing and configuring the current target.
Edit toolbar	Point and click access to the editing actions.

NOTE: The Target and Project Build windows can be docked at the top and bottom of the main window, or left free floating.

View, hide, dock, and move toolbars

NOTE: For more information see Toolbars and their uses.

View toolbars

Do one of the following:

- Right-click the status bar.
  -OR-
- Right-click on a blank area of any toolbar. Select the toolbar.
  -OR-
- Click View, Toolbar… Select the *toolbar* check box. Click OK.

Hide toolbars

1) Right-click on a blank area of any toolbar.
2) Clear the toolbar from the list.

If the toolbar is undocked:

- Right-click on the toolbar title bar and click Hide.
  -OR-
- On the toolbar title bar, click ☒ .

If the toolbar is docked:

1) Click View, then point to Toolbar…
2) Clear the *toolbar* check box. Click OK.

Dock toolbars

Do one of the following:

- Drag the toolbar to an edge of the main window.
  -OR-
- Double-click the title bar. The toolbar will be docked at its last docked position.

Move toolbars

1) Do one of the following:
   - On the toolbar title bar, right-click and click Move.

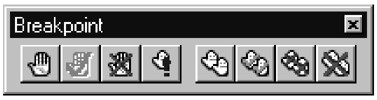
     -OR-
   - Click the toolbar title bar.
2) Drag the toolbar to the required position.

NOTE: YoucannotdocktheTargetwindowortheBuildwindowifyou are still pressing CTRL.

Commands on each toolbar

The commands on the toolbars provide access to the main debugging functions. You can also use the keyboard shortcuts for most debugging operations. Access keys support all operations; see Appendix A.

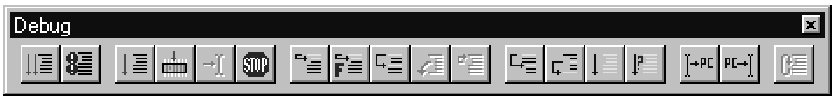
Breakpoint toolbar



Commands on the Breakpoint toolbar

To issue this command:	Click:	Press:
Toggle Breakpoint		F5
Enable Breakpoint		none available
Disable Breakpoint		none available
Configure Breakpoint(s)		CTRL+F5
Reset All Breakpoints		ALT+F5
Enable All Breakpoints		CTRL+SHIFT+F5
Disable All Breakpoints		CTRL+ALT+F5
Remove All Breakpoints		SHIFT+F5

Debug toolbar

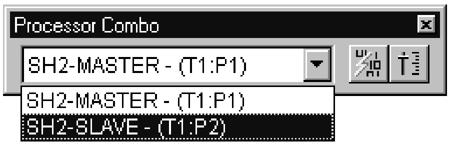


Commands on the Debug toolbar

To issue this command:	Click:	
Run all Processor(s)		CTRL+F9
Stop all Processor(s)		none available
Run		F9
Run to Address		SHIFT+F9
Run to Cursor		ALT+F9
Stop	STOP	F9
Single Step (into)		F7
Forced Step (into)	F	none available
Step Over		F8
Step Out		CTRL+F8
Unstep		CTRL+F7
Step Run In		SHIFT+F7
Step Run Out		SHIFT+F8
Step Run		
Step Run Until		

To issue this command:	Click:	
Set Cursor to PC	I→PC	CTRL+SHIFT+P
Set PC to Cursor	PC→I	CTRL+ALT+P
Restart		CTRL+SHIFT+R

Processor Combo toolbar



SH2-MASTER - (T1:P1)

SH2-MASTER - (T1:P1)
SH2-SLAVE - (T1:P2)

The Processor Combo toolbar shows the current processor for the current target. The toolbar also provides point and click access for loading program files and configuring the current processor.

Commands on the Processor Combo toolbar

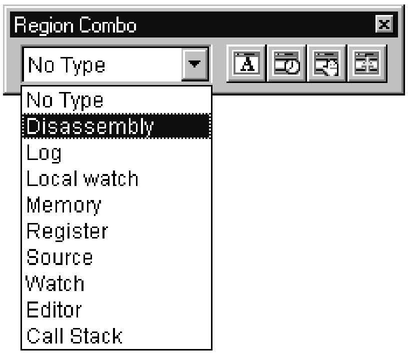
To issue this command:	Click:	Press:
Load Program File		CTRL+SHIFT+C
Configure Processor		none available

Project Build window

The Project Build window appears and automatically displays the specified build utility's output about the build. You can dock the Project Build window at the top and bottom of the main window, or leave it free floating. Any standard format errors and warnings are shown in the Project Build window. You can scroll through the information as it is generated, or press F4 to move through any listed errors one at a time. If you use:

- CodeScape's Edit region it automatically opens your project file at the line containing the first error or warning. You can then use the Project Build window to navigate to all subsequent errors. If there is no active Edit region CodeScape creates one for you.
- An external editor, double-click an entry in the Project Build window to invoke the editor and open the source file at the line containing the error or warning. Some external editors do not support this option and will open without displaying the line at which the error occurred.

Region Combo toolbar

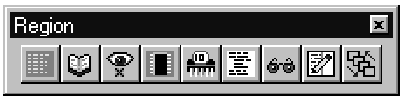


The Region combo toolbar lets you set the rate at which a region's display updates, and change a region's type.

Commands on the Region Combo toolbar

To set this command:	Click:
Region configuration	A
Window update rate	
Stop all window updates	
Update all regions	

Region toolbar



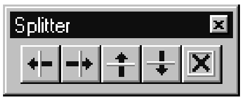
The Region toolbar lets you set or change a region's type.

Commands on the Region toolbar

To create this region:	Click:	Or press:
Disassembly	▦	ALT+1
Log	📖	ALT+2
Local Watch	👁	ALT+3
Memory	▮	ALT+4
Register	▥	ALT+5
Source	☰	ALT+6
Watch	👓	ALT+7
Edit	📝	ALT+8
Call Stack	⧉	ALT+9

NOTE: To stop the display from updating in all regions press CTRL+SHIFT+U.

Splitter toolbar

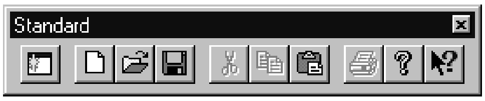


The Splitter toolbar lets you split existing regions to create new regions.

Commands on the Splitter toolbar

To issue this command:	Click:	Press:
Split Left	←	CTRL+SHIFT+LEFT ARROW
Split Right	→	CTRL+SHIFT+RIGHT ARROW
Split Up	↑	CTRL+SHIFT+UP ARROW
Split Down	↓	CTRL+SHIFT+DOWN ARROW
Delete Region	☒	CTRL+D

Standard toolbar

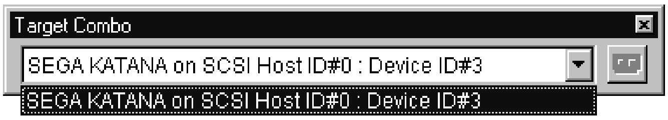


The Standard toolbar provides point and click access for opening a new window, and creating, opening, and saving sessions.

Commands on the Standard toolbar

To issue this command:	Click:	Press:
New window	🗔	CTRL+N
New Session	🗋	CTRL+SHIFT+N
Open Session	📂	CTRL+O
Save Session	💾	CTRL+S
Cut	✂	CTRL+X
Copy	⎘	CTRL+C
Paste	📋	CTRL+V
Print	🖶	CTRL+P
About Box	?	none available
Help	↖?	F1

Target Combo toolbar



SEGA KATANA on SCSI Host ID#0 : Device ID#3

SEGA KATANA on SCSI Host ID#0 : Device ID#3

The Target Combo toolbar shows the current target and lets you to configure it.

Commands on the Target Combo toolbar

To issue this command:	Click:
Target Configure	⊡

NOTE: The Serial Setup button only appears if you are connected to a target with a serial port.

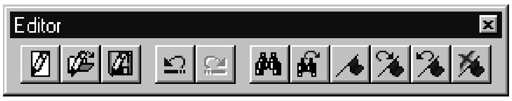
Target window

The Target window shows all the targets that CodeScape is connected to and the processors available in each target. The processor status for each target is shown in the Target processor display.

When you create a new window, CodeScape uses the target information from the selected processor on the target. The Target window provides point and click access for selecting a target processor.

NOTE: The Target window can be docked at the top and bottom of the main window, or left free floating.

Edit toolbar



The Edit toolbar provides point and click access to the editing actions.

To issue this command:	Click:	Press:
Create a new Editor file.		none available
Open an existing Editor file.		none available
Save the current Editor file.		none available
Undo the last action.		CTRL+Z
Redo the last action.		none available
Search for a string.		CTRL+F
Replace the current selection.		none available
Toggle a Book Mark on or off.		none available
Move to the next Book Mark in the file.		none available
Move to the previous Book Mark in the file.		none available
Delete all Book Marks.		none available

How windows and regions work

A Window is a frame that you can configure as a Region and split to create multiple Regions.

A Region lets you view information about your project.

To view a project's regions simultaneously you can:

- Open and close, minimize and maximize, cascade, and tile multiple windows.
- Split windows into multiple regions to display different types of information such as memory contents and source code.
- Resize windows.
- Resize regions by moving the Splitter bars.
- Proportionally resize a window's regions.

Configuring regions

Use the Region Configuration dialog box to configure fonts and colors for each region type, each individual region, and each processor. This lets you to differentiate between processors, show associated regions, and represent changes in memory.

NOTE: Synchronize the cursors in the Source and Disassembly regions to compare your source code with its compiled assembly code.

Using windows

Open a new window

- Click Window, then click New window.
  -OR-
- Click 🗔 on the Standard toolbar.
  -OR-
- Press CTRL+N.

Minimize a window

- On the window title bar click 🗕 .
  -OR-
- On the System menu, click Minimize.

Maximize a window

- On the window title bar click 🗖 .
  -OR-
- On the System menu, click Maximize.

Close a window

- On the window title bar click ✕ .
  -OR-
- On the System menu, click Close.
  -OR-
- Press CTRL+F4.

NOTE: If you delete the only region in a window, the window is deleted as well.

Close all windows

- Click Window, then click Close all Windows.

Move a window

1) Click the window’s title bar.
2) Drag the window to the required position.

Move between windows

- Press CTRL+TAB.

Resize a window

1) Point to the window boarder.
2) Click and drag the window outline to the required size.

To proportionally resize a region in a window:

1) Click Window, then select Proportional resizing.
2) Point to the window’s border.
3) Click and drag the window to the required size.

Load the current session when CodeScape restarts

- Click Window, then click to select Load last session on startup.

Cascade all windows

- Click Window, then click Cascade.

Tile all windows

- Click Window, then click Tile.

Arrange Icons

To arrange all minimized region windows at the bottom of the session window:

- Click Window, then click Arrange Icons.

Using regions

Change a region’s type

To change a region’s type:

- Click Region, then point to Type, then click a region type.
  -OR-
- On the Region Combo box, select a region type from the drop down list.
  -OR-
- Click a Region Type icon on the Region toolbar.
  -OR-
- CTRL+Right-click, then click a region type.

Navigate a region

To scroll through a region:

- Use the LEFT ARROW and RIGHT ARROW keys to move the cursor a single character at a time.
- Use the UP ARROW and DOWN ARROW keys to move the cursor up and down a line at a time.
- Use PAGE UP and PAGE DOWN to move up and down a page at a time.
- Use the HOME and END keys to move to the first visible line and the last visible line of a file.

To move through a region’s fields:

- To move the cursor to the next field, press TAB.
- To move the cursor to the previous field, press SHIFT+TAB.

NOTE: At the end of a field the cursor will move to the next field; at the end of the last field the cursor will move to the next line.

Move between regions

To move to the region to the left:

- Click anywhere in the region to the left.
  -OR-
- Press CTRL+LEFT ARROW.

To move to the region to the right:

- Click anywhere in the region to the right.
  -OR-
- Press CTRL+RIGHT ARROW.

To move to the region above:

- Click anywhere in the region above.
  -OR-
- Press CTRL+UP ARROW.

To move to the region below:

- Click anywhere in the region below.
  -OR-
- Press CTRL+DOWN ARROW.

Create new regions

To create a new region:

- Open a new window (CTRL+N).
  -OR-
- Split an existing region.

Split regions

To split a region to the left:

- Click Region, point to Split, then click Left.
  -OR-
- Click ← on the Splitter toolbar.
  -OR-
- Press CTRL+SHIFT+LEFT ARROW.

To split a region to the right:

- Click Region, point to Split, then click Right.
  -OR-
- Click → on the Splitter toolbar.
  -OR-
- Press CTRL+SHIFT+RIGHT ARROW.

To split a region above:

- Click Region, point to Split, then click Up.
  -OR-
- Click ↑ on the Splitter toolbar.
  -OR-
- Press CTRL+SHIFT+UP ARROW.

To split a region below:

- Click Region, point to Split, then click Down.
  -OR-
- Click ↓ the Splitter toolbar.
  -OR-
- Press CTRL+SHIFT+DOWN ARROW.

Delete a region

Make the region active, then do one of the following:

- Click Region, click Delete.
  -OR-
- Click ☒ on the Splitter toolbar.
  -OR-
- In the region, press CTRL+Right-click and click Delete Region.
  -OR-
- Press CTRL+D.

NOTE: If there is only one region in a window, deleting it deletes the window as well.

To update the display in all current regions:

- Click Region, then click Update all regions now.
  -OR-
- Press CTRL+U.

Configuring regions

Region Configuration dialog box

In the Region Configuration dialog box are tab commands for configuring fonts and colors, and setting the update rate for a single region, a region type, and each processor.

To Configure an active region:

- Right-click, click Properties…
  -OR-
- On an active region’s title bar, double-click ▤.

The Region Configuration dialog box appears.

Do the following:

1) Set the Mode and specify any commands in the *Target processor* text box.
2) Specify any options in the Target, Processor, or Region type lists.
3) Then do one of the following:
    - Click Apply to view your configuration changes without leaving the dialog box.
      -OR-
    - Click OK to set the configuration changes for your project.

Using the Region Configuration dialog box

To configure:	Select:
Thecurrentlyactiveregion in a project.	Apply to active region only.
All regions of a selected type on all processors.	Apply to all regions of selected type. Then select a Region type.
A specific Target Processor,andRegiontype.	Apply to all regions of the selected target. Then select a Target Processor, and a Region type.

Set the color and font

Use the Color and Font tab commands to differentiate between processors, show associated regions, and represent changes in memory.

To set the color attributes for the specified Mode:

1) Select the Color tab.
2) Select the attribute whose color you want to change.
3) Set the region Foreground color.
4) Set the region Background color.
5) Click OK.

To set the font type and size for the specified Mode:

1) Select the Font tab.
2) Click Change font.
3) Specify the region Font, Font style, and Size.
4) Set the Effects you require.
5) Click OK.

Set the region update rates

Use the Update Rate tab to specify when CodeScape will update information in each region.

If the update rate for a region's display interrupts the target causing jitter in your program, set the Foreground and Background sliders to Min.

To specify when CodeScape will update information in a region:

1) Select the Update tab.
2) Drag the Foreground slider to set the update rate for when the region has focus. Set the slider to Max to continually update the display (approximately 14Hz). Set the slider to Min to update the display at approximately 1/10th of the Max setting.
3) Drag the Background slider to set the update rate for when the region does not have focus. Set the slider to Max to continually update the display. Set the slider to Min to prevent updates to the display.

Target window

NOTE: The Target window can be docked at the top and bottom of the main window, or left free floating.

When you run CodeScape it scans for valid targets and displays them in the Target window. The Target window shows all the targets that CodeScape is connected to and the processors available in each target. The processor status for each target is shown in the Target processor display.

When you create a new window, CodeScape uses the target information from the selected processor on the target.

Using the shortcut menu on the Target window

Controlling target processor execution

Right-click, then click:	To:
Configure Processor…	Set the update rate for the current processor.
Simulate Processor	Run the Simulator.
Execution	Run, stop, and restart your program. Run your program until it executes a specified address. Run all of your program files simultaneously. Stop all of your programs running simultaneously.Use the single stepping commands, or run the step commands.
Breakpoints	Add, enable, disable, configure, reset, or remove data breakpoints.
Reset Target	Perform a soft reset or a hard reset. If you reset the target you are prompted to reload the Program File.
Load Program File	Download a program file to the selected processor on the target.

Target Processor display

To show the processor(s) for a target:

- Double-click on the target status line.
  -OR-
- Click ⊞ .

To hide the processor(s) for a target:

- Double-click on the target display line.
  -OR-
- Click ⊟ .

Project Build window

The Project Build window appears and automatically displays the specified build utility’s output about the build. You can dock the Project Build window at the top and bottom of the main window, or leave it free floating.

Any standard format errors and warnings are shown in the Project Build window. You can scroll through the information as it is generated, or press F4 to move through any listed errors one at a time. If you use:

- CodeScape’s Edit region it automatically opens your project file at the line containing the first error or warning. You can then use the Project Build window to navigate to all subsequent errors. If there is no active Edit region CodeScape creates one for you.
- An external editor, double-click an entry in the Project Build window to invoke the editor and open the source file at the line containing the error or warning. Some external editors do not support this option and will open without displaying the line at which the error occurred.

Using the shortcut menu on the Project Build window

Configure, make, then build a project

Right-click, then click:	To:
Setup Project…	Specify file locations for making a project current and building it.
Setup Editor…	Specify the editor that you want to use for your project.
Make	Make your project current by building it.

The Source region

In the Source region there are commands for debugging your original source code.

When you edit your source code the changes are displayed in the corresponding Source region when the display is updated. A * appears in a Source region’s title bar if you edit your source code and do not re-build the program file. Always save any changes that you make to a file edited in an external editor before using the Make option to compile and build your project in CodeScape.

NOTE: Place the mouse pointer over a variable or expression to quickly view its’ value.

NOTE: Before you edit a program file from a UNIX target, convert it to a DOS readable format using a utility such as to_dos (use to_unix to return the file to a UNIX format).

NOTE: If no debug information appears in a Source region, compile all source files for your project with debugging turned on.

Using the shortcut menu in a Source region

The commands on the shortcut menu are for debugging in the region and configuring the source view. Right-click anywhere in the region to access the shortcut menu.

Copy in the Source region

1) In the Source region, select the text you want to copy by highlighting it.
2) Right-click, then click Copy.

The selection is copied, then pasted to the clipboard.

Lock the display origin to an expression

1) On the Source region title bar click 🔒.
   The Goto Address… dialog box appears.
2) Enter an expression for the region origin.
3) Click OK.

Configuring the Source view

Right-click, then click:	To show the:
Show Address	Corresponding address for the first line of code generated by the source code line.
-OR-	
Show Line Nos.	Line numbers for each line of source in the left-hand column.

Accessing the debugging commands in a Source region

Right-click, then select:	To click commands to:
Execution	Run, stop, and restart your program. Run your program to the cursor position, or until it executes a specified address. Run all of your program files simultaneously. Stop all of your programs running simultaneously.Use the single stepping commands, or run the step commands.
Breakpoints	Toggle a breakpoint on or off. Enable, disable, configure, reset, and remove breakpoints.

Setting the cursor and the display in a Source region

Right-click, then click:	To:
Set Cursor to PC	Show the source code from the value of the PC.
Set PC to Cursor	Change the PC at the current cursor position.
Goto Address…	Enter an expression for the region origin to go to.
Goto Source File…	Select the required source file.
Tools	Search in the Source region. Find the next item in the search.
Tab Width…	Enter a value to set the tab size in spaces.
Properties	Configure fonts and colors. Set the update rate for a single region, a region type, and each processor. Change the tab settings.

Synchronize the cursors in a Source and Disassembly region

1) In the Source region:
   - Right-click and click Synchronize Cursor.

     -OR-
   - On the Source region title bar click .
2) In the Disassembly region:
   - Right-click and click Synchronize Cursor.

     -OR-
   - On the Disassembly region title bar click .

The cursors for the Disassembly and Source regions are now synchronized. When you move the cursor in the region with focus, the cursor in the synchronized region shows the corresponding line of code.

NOTE: You can only synchronize regions that are in the same window and are connected to the same target processor.

Goto an address

1) Do one of the following:
   - Click Edit, then click Go To (CTRL+G).

     -OR-
   - Right-click, click Goto Address.

   The Goto Address dialog box appears. (This dialog box works in the same way as the Expression Evaluator.)
2) Enter an expression for the address to go to.
3) Click OK.

Go to a source file referenced in the program file

1) Right-click and click Goto Source File.
   The List Files in Program File dialog box appears.
2) Select the required source file.
3) Click OK.

If the path is incorrect an error message appears in the Source region. Click Project, then click Edit Source Path and enter the correct path for the source files.

NOTE: Code is not generated for data-only files, or if the -g command is not set when compiling. If code is not generated an error message appears.

Evaluate a specific expression

1) Select an expression in the region.
2) Right-click, click Evaluate...

Change the tab settings

1) Right-click, point to Properties then click Tab Width...
   The Change Tab Size dialog box appears.
2) Enter a value for the number of spaces used to represent a tab.
3) Click OK.

Search in the Source region

1) Right-click in the Source region, click Tools, then click Find.
2) Type the Search string in the *Find what* text box.
3) To search for whole words and not parts of a larger word, select the *Match whole word only* check box.
4) If the search is case sensitive, select the *Match case* check box.
5) Click OK.

The search will start from the current cursor position and continue until the end of the file.

NOTE: Right-click, then click Find next to continue searching for the same item.

The Disassembly region

In a Disassembly region there are commands for debugging your program at instruction level (assembly code).

NOTE: Place the mouse pointer over a variable or expression to quickly view its' value.

Using the shortcut menu in a Disassembly region

The commands on the shortcut menu are for debugging in the region and configuring the disassembly view. Right-click anywhere in the region to access the shortcut menu.

Copy in the Disassembly region

1) In the Disassembly region, select the text you want to copy by highlighting it.
2) Right-click, then click Copy.

The selection is copied, then pasted to the clipboard.

Lock the Disassembly region

You can lock the view origin to the PC, a register, or a memory location.

1) On the Disassembly region title bar click . The Goto Address dialog box appears.
2) In the *Expression* text box, enter a valid expression:
   - Value of the PC to lock the view to the PC. Click OK.
     -OR-
   - Name of the register to lock the view to a register. Click OK.
     -OR-
   - In the *Expression* text box, enter the address of the memory location to lock the view to a memory location. Click OK.

NOTE: To unlock the view origin, click again.

Configuring the disassembly view

Right-click, then click:	To show the:
Show Address	Location address of the disassembled code.
Show Labels	Symbolic label replacement of the disassembled code.
Show Opcode Words	Op-code in words for the disassembled region.
Show Hexadecimal	Operand values in hexadecimal.
Show Uppercase	Instructions in upper case.
Show Symbols	Operand values as symbols.
Show EAs & Lits	Effective address and literals.

Accessing the debugging commands in a Disassembly region

Right-click, then select:	To:
Execution	Run, stop, and restart your program. Run your program to the cursor position, or until it executes a specified address. Run all of your program files simultaneously. Stop all of your programs running simultaneously.Use the single stepping commands, or run the step commands.
Breakpoints	Toggle a breakpoint on or off. Enable, disable, configure, reset, and remove breakpoints.

Setting the cursor and the display in a Disassembly region

Right-click, then click:	To:
Set Cursor to PC	Show the source code from the value of the PC.
Set PC to Cursor	Change the PC at the current cursor position.
Goto Address…	Enter an expression for the region origin to go to.
Tools	Search in the Disassembly region. Repeat the last search run. Specify an address to disassemble to a file.
Properties	Configure fonts and colors. Set the update rate for a single region, a region type, and each processor. Change the tab settings.

Synchronize the cursors in a Disassembly and Source region

1) In the Disassembly region:
   - Right-click and click Synchronize Cursor.

     -OR-
   - On the Disassembly region title bar click .
2) In the Source region:
   - Right-click and click Synchronize Cursor.

     -OR-
   - On the Source region title bar click .

The cursors for the Disassembly and Source regions are now synchronized. When you move the cursor in the region with focus, the cursor in the synchronized region will show the corresponding line of code.

NOTE: You can only synchronize regions that are in the same window and are connected to the same target processor.

Goto an address

1) Do one of the following:
   - Click Edit, then click Go To (CTRL+G).

     -OR-
   - Right-click, click Goto Address.

   The Goto Address dialog box appears.
2) Enter an expression for the address to go to.
3) Click OK.

Evaluate a specific expression

1) Select an expression in the region.
2) Right-click, click Evaluate...

Search in the Disassembly region

1) Right-click in the Disassembly region, point to Tools, then click Find.
2) Type the Search string in the *What am I searching for:* text box.
3) Type the Start address in the *Search from:* text box.
4) If the search is case sensitive, select the *Case sensitive* check box.
5) Select one of the following radio buttons:
   - Length (the amount of data).

     -OR-
   - End Address.
6) Type the search item in the text box below.
7) Select one of the following radio buttons: All fields (default), Words, Opcode, OpSrc, OpDest, or Label (address).
8) Click OK.

NOTE: Right-click then click Find next to continue searching for the same item.

Specify an address to disassemble to a file

This general purpose dialog box is for writing a block of memory or disassembly in hexadecimal to a file.

1) Right-click in the Source region, point to Tools, then click Disassemble to File.
2) In the *Destination Filename* text box, enter the name of the file to write to.
3) In the *Start Address* text box, enter the start address in hexadecimal.
4) Do one of the following:
    - Select Length and enter the length in hexadecimal.

      -OR-
    - Select End Address and enter the end address in hexadecimal.
5) Click OK.

The Call Stack region

Use the Call Stack region to view a list of active function calls. Viewing the Call Stack can help you trace the course of function execution. When the target stops, for example if a breakpoint occurs, CodeScape displays the name, label, or address of the current function at the top of the list in the Call Stack region. Execution trace history is shown below the current function with its start point at the bottom of the list.

To use the Call Stack to navigate to a specific function call in active Source, Disassembly, Watch, and Local Watch regions:

- Double click on a function in the Call Stack region.
  CodeScape highlights the function as it occurs in the active regions.

Using the shortcut menu in a Call Stack region

Right-click, then select:	To click commands to:
Show Parameter Names	Toggle the display of the function parameter on or off.
Show Parameter Types	Toggle the display of the function parameter types on or off.
Show Parameter Values	Toggle the display of the function parameter values on or off.
Show Parameter Registers	Toggle the display of the function parameter registers on or off.
Show Octal	Display function values in octal.
Show Decimal	Display function values in decimal.
Show Hexadecimal	Display function values in hexadecimal.
Execution	Run, stop, and restart your program. Run your program until it executes a specified address. Run all of your program files simultaneously. Stop all of your programs running simultaneously.Use the single stepping commands, or run the step commands.
Breakpoints	Toggle a breakpoint on or off. Enable, disable, configure, reset, and remove breakpoints.
Properties	Configure fonts and colors. Set the update rate for a single region, a region type, and each processor.

The Watch and Local Watch regions

The Watch and Local Watch regions display variables and expressions, one per row.

Each row has four columns, the expression appears in the third column and its value (if applicable) appears in the fourth column. If the:

- First column contains a '.' you can place a watch point on the expression.
- Second column contains a '+' you can expand the expression.
- Second column contains a '-' you can collapse the expression.

In a Watch or Local Watch region, you can:

- Highlight changes in data values between execution steps.
- Edit the value of an expression in the Watch region and the Local Watch region.

NOTE: You can only edit the actual expression in a Watch region.

C++ name demangling in a Watch or Local Watch region

C++ name de-mangling is performed on all variable names. This means that you can enter the symbol for a name as it appears in your original source. You can browse data to:

- Expand and collapse branches of the hierarchical view of the structure.
- See exactly where the structures are in memory.
- Edit the values of any variables.

All C types are supported including:

- structs
- unions
- arrays
- enumeration (enum)
- floats / double

Expanding expressions

When you expand an expression, each child expression is indented and shown directly below the parent.

For example:

```
parent
child
child
```

Expressions are added to expanded:

- Pointers, to show the dereferenced item.
- Arrays. An expression is added for each element of the array.
- Structures. An expression is added for each member.

The Watch region

In the Watch region you can enter variables and expressions. The scope of variables in a Watch region is global. If an expression goes out of scope during program execution, a message appears.

- If an expression's value can be determined, as is the case for a static variable, its value is shown in the region.
- If an expression's value cannot be determined, no value is shown.
- If an expression comes back into scope, its value is shown.

Using the shortcut menu in a Watch region

Right-click, then select:	To click commands to:
Cut	Cut the current selection in the Editor file and paste it to the clipboard.
Copy	Copy the current selection in the Editor file and paste it to the clipboard.
Paste	Insert the contents of the clipboard at the current cursor position.
Delete	Delete part of a structure.
Open / Close	Expand / collapse a structure or array.
Insert	Insert a new watch expression.
Append	Insert a variable at the end of the current list.
Show Octal	Display watch expressions in octal.
Show Decimal	Display watch expressions in decimal.
Show Hexadecimal	Display watch expressions in hexadecimal.
Edit Watch Value…	Modify the value of a variable or watch expression.
Execution	Run, stop, and restart your program. Run your program until it executes a specified address. Run all of your program files simultaneously. Stop all of your programs running simultaneously.Use the single stepping commands, or run the step commands.

Right-click, then select:	To click commands to:
Breakpoints	Toggle a breakpoint on or off. Enable, disable, configure, reset, and remove breakpoints.
Highlight Changes	See where in memory an expression changed.
Properties	Configure fonts and colors. Set the update rate for a single region, a region type, and each processor.

Browsing data in a Watch region

Add a symbol or variable

In a Watch region do one of the following:

- Right-click, click Insert to add a symbol or variable at the current cursor position.
  -OR-
- Right-click, click Append to add symbol or a variable at the end of the current list of variables.
  -OR-
- Press return to enter a new symbol or variable at the current cursor position.

Expand a structure or array

Select the structure or array you want to expand, then:

- Click on ‘+’.
  -OR-
- Press SPACEBAR.
  -OR-
- Right-click and click Open/Close.

Collapse a structure or array

Select the structure or array you want to collapse, then:

- Click on ‘-’.
  -OR-
- Press SPACEBAR.
  -OR-
- Right-click and click Open/Close.

Editing variables in a Watch region

Modify the value of a variable or watch expression

Do one of the following:

- Select the value to be changed. Press ENTER.
  -OR-
- Press CTRL+ALT+E.

The Expression Evaluator dialog box appears. Enter a valid expression. Click OK.

Edit a variable's data value (structure, array or union)

1) Double-click the value to be edited.
2) Edit the value.
3) Do one of the following:
   - Press ENTER.
     -OR-
   - Press CTRL+ALT+E to display the Expression Evaluator dialog box.

To delete a parent expression and all child expressions:

1) Expand the structure or array.
2) Select the component you want to delete, then:
   - Right click and click Delete.
     -OR-
   - Press DELETE.

NOTE: If you delete a parent expression, any child expressions are also removed from the region.

To delete a parent expression and move all child expressions up one level:

1) Expand the structure or array.
2) Select the component you want to delete.
3) Press SHIFT+DELETE.

The Local Watch region

The Local Watch region automatically displays all local variables in view from the current position of the PC (program counter). Variables are automatically added to the display when they are in the scope of a function.

NOTE: Ifthereismorethanonevariableofthesamenameinscope, all but the inner most variable of that name are shown in gray.

NOTE: If there are two variables in the same scope with the same name they are shown in italics.

Using the shortcut menu in a Local Watch region

Right-click, then select:	To click commands to:
Copy	Copy the current selection in the Editor file and paste it to the clipboard.
Delete	Delete the current selection.
Open / Close	Expand / collapse a structure or array.
Show Octal	Display local watch expressions in octal.
Show Decimal	Display local watch expressions in decimal.
Show Hexadecimal	Display local watch expressions in hexadecimal.
Edit Local Value	Modify the value of a variable or watch expression.
Execution	Run, stop, and restart your program. Run your program until it executes a specified address. Run all of your program files simultaneously. Stop all of your programs running simultaneously.Use the single stepping commands, or run the step commands.
Breakpoints	Togglea breakpoint on or off. Enable, disable, configure, reset, and remove breakpoints.
Highlight Changes	See where in memory an expression changed.
Properties	Configurefonts and colors. Set the update rate for a single region, a region type, and each processor.

Browsing data in Local Watch region

Expand a structure or array

Select the structure or array you want to expand, then:

- Click on ‘+’.
  -OR-
- Press SPACEBAR.
  -OR-
- Right-click and click Open/Close.

Collapse a structure or array

Select the structure or array you want to collapse, then:

- Click on ‘-’.
  -OR-
- Press SPACEBAR.
  -OR-
- Right-click and click Open/Close.

Editing variables in a Local Watch region

Modify the value of a variable or watch expression

1) Do one of the following:
    - Select the value to be changed. Press ENTER.
      -OR-
    - Press CTRL+ALT+E.

   The Expression Evaluator dialog box appears.
2) Enter a valid expression.
3) Click OK.

To delete a parent expression and all child expressions:

1) Expand the structure or array.
2) Select the component you want to delete, then:
   - Right click and click Delete.
     -OR-
   - Press DELETE.

NOTE: Ifyoudeleteaparentexpression,anychildexpressionsare also removed from the region.

To delete a parent expression and move all child expressions up one level:

1) Expand the structure or array.
2) Select the component you want to delete.
3) Press SHIFT+DELETE.

The Memory region

Use the Memory region to view the target's memory contents from a specific address. In a Memory region you can view memory as ACSII characters, Bytes, Words, or Longs. Write protect an area of memory to prevent memory contents changing in the current memory region.

NOTE: As you scroll through a Memory window the slider speed increases. Click on the top (or bottom) of the slider to return it to the centre position.

Using the shortcut menu in a Memory region

Right-click, then select:	To click commands to:
Display Bytes	Display memory as bytes.
Display Words	Display memory as words.
Display Longs	Display memory as longs.
Display Quadwords	Display memory as quadwords.
Display ASCII	Display the ASCII value for each byte memory.
Highlight Changes	See where the target's memory changed.
Set Bytes Per Line…	Display a specific number of bytes per line.
Edit ASCII	Change an ASCII value in the Memory region.
Edit Memory Value	Change a value in the Memory region.
Follow Pointer	Follow a pointer in memory.
Goto Address…	Set the origin.
Write Protect	Toggle write protect.
Execution	Run, stop, and restart your program. Run your program to the cursor position, or until it executes a specified address. Run all of your program files simultaneously. Stop all of your programs running simultaneously.Use the single stepping commands, or run the step commands.
Breakpoints	Toggle a breakpoint on or off. Enable, disable, configure, reset, and remove breakpoints.

Right-click, then select:	To click commands to:
Tools	Search for a pattern in memory. Repeat the last search. Fill a range of memory with data. Write a block of memory in hexadecimal to a file.
Properties	Configure fonts and colors. Set the update rate for a single region, a region type, and each processor.

Viewing Memory

View Memory regions

Do one of the following:

- Click Region, point to Type and click Memory.
  -OR-
- On the Region toolbar, click ▣.
  -OR-
- In any region, CTRL+Right-click and click Memory.

Set the origin

1) Right-click and click Goto Address…
   The Goto Address… dialog box appears.
2) Enter an address or symbol for the new origin.
3) If you enter an invalid symbol a warning appears with an command to invoke the Origin dialog box.
4) Click OK.

NOTE: The origin is initially set to the value of the PC. You can also set the origin to an expression.

Always display memory from a specified address

1) Do one of the following:
   - Click Edit, then click Go To… (CTRL+G).

     -OR-
   - Right-click and click Goto Address…

   The Goto Address… dialog box appears.
2) Type or select a memory location or expression from the Expression list.
3) Select Lock, click OK.

Follow a pointer in memory

1) Select the memory location holding the value of the pointer.
2) Right-click and click Follow Pointer (CTRL+T).

The Memory region origin changes to display memory from the location specified by the value of the pointer.

Write a block of memory in hexadecimal to file

1) In the Memory region, right click, point to Tools and then click Hex Dump to File. The Hex Dump to File dialog box appears.
2) In the *Destination Filename* text box, enter the name of the file to write to. In the *Start Address* text box, enter the start address in hexadecimal.
3) Then do one of the following:
   - Select Length and enter the length of the memory block in hexadecimal.

     -OR-
   - Select End Address and enter the end address in hexadecimal.
4) Click OK.

The specified block of memory is written to a file.

Editing memory

Change values in the Memory region

1) In a Memory region, do one of the following:
   - Use the + and - keys to increment or decrement the current value.

     -OR-
   - Double-click or press ENTER, then type over the existing byte, word, or long value.

     -OR-
   - Right-click, click Edit memory value…

     -OR-
   - Press CTRL+ALT+E.

   The Expression Evaluator dialog box appears.
2) Enter a valid expression.
3) Click OK.

NOTE: Make sure you enter valid values for the current radix.
CodeScape displays all Memory values in hexadecimal.

Filling memory with specific data

To fill a range of memory with a specific data:

1) In a Memory region, right-click and select Tools…, click Fill...
2) Enter a value in the *Fill Expression* text box.
3) Enter a value in the *Start Address* text box.
4) Select End address, or Length.
5) Enter a value in the text box below.
6) Select the Mode as Text (ASCII), Byte, Word, Long, or Quad.
7) Do one of the following:
   - Select *Convert Native Endian*, to show the real memory value.

     -OR-
   - Deselect *Convert Native Endian,* to store memory as byte sequences.
8) Click OK.

Searching memory

To define and search an area of memory for a specified pattern of data:

1) In a Memory region, right-click and select Tools…
2) Click Find. The Find In Memory dialog box appears.
3) Enter a search string in the *Find Pattern* text box.
   In Binary, Octal, Decimal, and Hex modes the search pattern is delimited by either commas or semi-colons (optional).
4) Enter a value in the *Start Address* text box.
   The default value is the start address of the region. If you have not run a search, the address at the start of the current memory block is used.
5) Select End Address, or Length.
   The default value for the end address is the last displayed byte in the Memory region.
6) Enter a value in the text box below.
7) Select the Width as either Byte, Word, or Long.
8) Select Forward or Reverse to specify the direction of the search.
9) Click OK.

NOTE: If a specified search is not valid, ‘Invalid Address’ appears in the field(s) that require editing.

NOTE: If a match is found its address appears. A search skips over any sensitive areas such as invalid memory areas, write-only memory, and memory reserved for the monitors.

Width

Width aligns the search pattern with the data in the target’s memory. This specifies how the search pattern and the memory contents are compared. The allowable width depends on the search mode selected.

Valid mode and width combinations

For this mode:	Valid widths are:
Binary	Binary, Word, Long
Decimal	Byte
Hex	Binary, Word, Long
Text	N/A

These patterns are equivalent with Hex and Byte widths set:

```
FF,FF,FF,FF,34,DC
FF;FF;FF;FF;34;DC
FFFFFFFF34DC
```

The \ specifier

Use the \ specifier to include special characters in text searches.

NOTE: Always enclose a text search string in quotes.

Example

The pattern "How are you\?" searches for "How are you?"

The ? wildcard

The wild card character ‘?’ can be used in Binary and Hex modes. Use ‘?’ to specify a nibble in Hex mode and a bit in Binary mode that always results in a successful match.

Example

In Hex mode:

FF?F matches FF0F,FF1F,FF2F,...,FFFF

In Binary mode:

????1111 matches 00001111,00011111,...,11111111

The @ wildcard

The wild card character, ‘@’, can be used in Text modes. Use ‘@’ to specify a double-byte character.

Automatic padding

The search pattern is automatically left-padded for the Binary, Decimal, and Hex modes. The padding type is either ‘0’ or ‘?’ depending on the delimiter used.

Delimit with commas

Delimit a search pattern with commas (the default) to left-pad it with zeros. For example, in Hex mode with Byte width:

FFFFFFFF34DC and FF,FF,FF,FF,34,DC do the same search.

The comma separator in is implied in the first search pattern. More examples, in Hex mode and Word width, are:

f,87d,a automatically pads to000F,087D, 000A

f87da automatically pads to000F,87DA

, automatically pads to0000

NOTE: A single comma used on its own produces the pattern 0000.
Use this feature carefully. For example: ,7 automatically pads to
0000,0007

Delimit with semi-colons

Delimit a search pattern with semi-colons to pad it with the ‘?’ wild card. Examples, in Hex mode and Word width, are:

f;8d;a automatically pads to???F,?87D,???A

f;87da automatically pads to???F,87DA

; automatically pads to????

Equivalent search patterns

Use the comma and semi-colon delimiters to do the same search pattern in different ways. The following patterns are the same when the Hex and Byte widths are set:

```
FF,FF,FF,FF,34,DC
FF;FF;FF;FF;34;DC
FFFFFFFF34DC
```

You can mix the comma and semi-colon delimiters to produce precise search patterns. For example, in Hex mode and Long width: f;f0f0f0f0,fffffff?,7; pads to:
???????F,F0F0F0F0,FFFFFFF?,???????7

The Register region

The Register region shows the contents of a processor’s register block and flags.

Using the shortcut menu in a Register region

Right-click, then select:	To click commands to:
Increment Register	Apply the current Increment Value (1 is the default) to the contents of the register.
Decrement Register	Apply the current Decrement Value (1 is the default) to the contents of the register.
Change Inc/Dec Value…	Change the Increment/Decrement Value.
Highlight Changes	See where changes occurred during the last operation.
Write Protect	To prevent data from being written to the currently active Register region.
Edit Register	Change the selected register value.
Column Format	Display registers in two, or four columns. Select Auto to tell CodeScape to choose.
Show Banked Registers	Toggle the banked register display on or off.
Show Float Registers	Toggle the floating point register display on or off.
Execution	Run, stop, and restart your program. Run your program to the cursor position, or until it executes a specified address. Run all of your program files simultaneously. Stop all of your programs running simultaneously.Use the single stepping commands, or run the step commands.
Breakpoints	Toggle a breakpoint on or off. Enable, disable, configure, reset, and remove breakpoints.
Tools	Save the current Register Block. Restore the last saved Register Block.
Properties	Configure fonts and colors. Set the update rate for a single region, a region type, and each processor.

View the Registers region

Click Region, point to Type and click Register.

Change the display format

The registers are displayed in the available area by default. You can set the display to two or four columns.

Display registers in two columns

Do one of the following:

- Right-click, point to Column Format, then click 2 Columns.
  -OR-
- Press CTRL+2.

Display registers in four columns

Do one of the following:

- Right-click, point to Column Format, then click 4 Columns.
  -OR-
- Press CTRL+4.

Display registers in the available area

Do one of the following:

- Right-click, point to Column Format, then click Auto Format.
  -OR-
- Press CTRL+0.

Edit register values

Change the value of a register

1) Move the insertion point to the register value you want to change.
2) Do one of the following:
    - Use + or - to increment or decrement the current value.

      -OR-
    - Type the new value at the insertion point.

      -OR-
    - Double click a register, type a value or expression, press ENTER. The Expression Evaluator dialog box appears.

      -OR-
    - Press CTRL+ALT-E to invoke the Expression Evaluator dialog box.

NOTE: Any alphanumeric characters are shown in upper case.

Change the Increment/Decrement Value

1) Right-click and click Change Inc/Dec Value…
   The Register Increment/Decrement dialog box appears.
2) Type a value for the amount by which to increment or decrement a register. Click OK.

Write protect a register

1) Right-click in the region.
2) Then do one of the following:
    - If Write protect is not selected, click Write Protect.

      -OR-
    - If Write protect is selected, exit the shortcut menu.

NOTE: Write protect only stops register values being changed in the region that is write protected. If you create and edit new Register regions, the changes appear in the write protected region.

Enter an expression for the instruction at the current PC

1) Double-click the register value, then:
    - Enter the expression. Press ENTER.

      -OR-
    - Right-click and click Edit Register… (CTRL+ALT+E).

   The Register Evaluation dialog box appears.
2) Enter the expression.
3) Click OK.

NOTE: If you enter an invalid expression, the Register Evaluation dialog box appears showing the Invalid Register Expression.

Save the state of the registers

Right-click, point to Tools, then click Save Register.

NOTE: The register states for each target processor are saved one at a time and cannot be stacked. The state of the registers is stored internally to CodeScape, not in a file.

Retrieve the state of the registers

Right-click, point to Tools then click Restore Register.

Hitachi target processor register region display

General registers

The Register region shows the values of Hitachi’s 16 general registers (Rn) numbered R0-R15.

R0 works as a fixed source register or destination register in some instructions, and as an index register in:

- Indirect indexed register addressing mode.
- Indirect indexed GBR addressing mode.

R14 works as the frame pointer during debugging.

R15 works as a hardware stack pointer (SP) during exception processing.

Control registers

The Register region shows the values of the SR (Status Register), GBR (Global Base Register), and VBR (Vector Base Register).

System registers

The Register region displays the MACH and MACL (high and low multiply and accumulate registers), PR (Procedure Register), and PC (Program Counter). The MACH and MACL registers store the results of multiply and accumulate operations. The PR stores a return address from a subroutine procedure.

To change the value of a status register:

1) Move the insertion point to the register value you want to change.
2) Do one of the following:

- Use + or - to increment or decrement the current value.

  -OR-
- Type the new value at the insertion point.

  -OR-
- Press CTRL+ALT+E to invoke the Expression Evaluator dialog box.

NOTE: Any alphanumeric characters are shown in upper case.

To set the status register (SR, SSR, or FPSR) flags:

1) Move the insertion point to the flag you want to set.
2) Then:
   - Press 1 set the current flag.
     The bit's flag appears in upper-case.
     -OR-
   - Press 0 to clear the current flag.
     The bit's flag appears in lower-case.

NOTE: Press SPACEBAR to toggle the status registers.

Flags shown in the Registers region after a Target Processor reset

This flag:	Represents this value:
T bit	The MOVT, CMP/cond, TAS, TST, BT (BT/S), BF (BF/S), SETT and CLRT instructions use the T bit to show true (1) or false (0). The ADDV/C, SUBV/C, DIV0U/S, DIV1, NEGC, SHAR/L, SHLR/L, ROTR/L and ROTCR/L instructions also use the T bit to show carry/borrow or overflow/underflow.
S bit	Used by the multiply/accumulate instruction.
Bits2,3and10-31 (Reserved bits.)	Always reads as 0 and must be written as 0.
Bits I3-I10	Interrupt mask bits.
M and Q bits	Used by the DIV0U/S and DIV1 instructions.

SEA and DEA

The Register region shows the value of the SEA (Source Effective Address) and DEA (Destination Effective Address). The SEA and DEA show the source effective address (read from) on the left-hand side and the contents of that address (write to) on the right-hand side.

Highlight recently changed attributes

Recently changed attributes are shown briefly in red (default).

To use another color to highlight a changed attribute, do the following:

1) Click View, then click Properties.
2) Specify the Mode.
3) Select the Color tab.
4) In the Attribute region, select Highlight data changed.
5) In the Effects region, select a new Foreground color.
6) Click OK.

NOTE: You cannot highlight changed attributes by setting a different Background color.

The Log region

The Log region automatically appears and displays all messages generated by the current target.

For example, you can use printf () in your code to output a message to the Log region when a breakpoint triggers.

NOTE: Text strings longer than 132 characters are truncated when displayed in the Log region.

Using the shortcut menu in a Log region

Right-click, then select:	To click commands to:
Configure Log…	Configure the current Log window.
Print…	Print the contents of the current Log window.
Save To File…	Save the contents of the active Log window to file.
Execution	Run, stop, and restart your program. Run your program to the cursor position, or until it executes a specified address. Run all of your program files simultaneously. Stop all of your programs running simultaneously.Use the single stepping commands, or run the step commands.
Breakpoints	Toggle a breakpoint on or off. Enable, disable, configure, reset, and remove breakpoints.
Reset Log	Clear the contents of the Log window.
Properties	Configure fonts and colors. Set the update rate for a single region, a region type, and each processor.

Print the log

1) Make the Log region active.
2) Do one of the following:
   - Click File, then click Print (CTRL+P).

     -OR-
   - Right-click and click Print.

NOTE: CodeScape uses your Windows default printer. To specify a different printer or print settings, click File, then click Print Setup…

Save the log

Right-click and click Save to File.

Clear the Log region

Right-click and click Reset Log.

Configure the Log region

1) In the Log region, right-click and click Configure Log.
   The Log Window Settings dialog box appears.
2) Enter a name for the log. The name is used only to title the log region.
3) To set the Log Type, select:
   - Any (default) to accept breakpoints and standard output.

     -OR-
   - Standard.

     -OR-
   - Breakpoint.
4) Drag the slider to set the size of the buffer. The buffer size can be in the range 10-1,000 lines. The default size is 255 lines.

The Edit region

The default editor is CodeScape's Edit region where you can manage, edit, and print source files.

When you edit your source code the changes are displayed in the corresponding Source region when the display is updated. A * appears in a Source region's title bar if you edit your source code and do not re-build the program file. Always save any changes that you make to a file edited in an external editor before using the Make option to compile and build your project in CodeScape.

Any standard format errors and warnings are shown in the Project Build window. You can scroll through the information as it is generated, or press F4 to move through any listed errors one at a time. If you use:

- CodeScape's Edit region it automatically opens your project file at the line containing the first error or warning. You can then use the Project Build window to navigate to all subsequent errors. If there is no active Edit region CodeScape creates one for you.
- An external editor, double-click an entry in the Project Build window to invoke the editor and open the source file at the line containing the error or warning. Some external editors do not support this option and will open without displaying the line at which the error occurred.

NOTE: Before you edit a program file from a UNIX target, convert it to a DOS readable format using a utility such as to_dos (use to_unix to return the file to a UNIX format).

Using the shortcut menu in an Edit region

Right-click, then select:	To click commands to:
New	Create a new Editor file.
Open…	Open an existing Editor file.
Save	Save the current Editor file.
Save As…	Save the current Editor file with a specific name.
Undo...	Undo the last action.
Redo...	Redo the last action.
Cut	Cut the current selection in the Editor file and paste it to the clipboard.
Copy	Copy the current selection in the Editor file and paste it to the clipboard.
Paste	Insert the contents of the clipboard at the current cursor position.
Toggle Boomark	Toggle the current bookmark on or off.
Tabs…	Enter a new tab value.
Find…	Search for a string.
Replace…	Replace the current selection.
Go To…	Change the line number of the origin address.
Properties	Configure fonts and colors in the region.
Syntax Highlighting	Turn syntax coloring on or off. Turn case sensitivity on or off. Specify a color for any or all of the following items in your code: keywords, quotes, comments, default text, and the background.

Opening and saving files

Open an existing file

1) Right-click and click Open.
   The File Open dialog box appears.
2) Select the required file. Click Open.

Open a new file

- Right-click and click New.

Save a file

- Right-click and click Save.

NOTE: If you use the save command for an un-named file, the File Save As dialog box appears.

Save a file with a new name

1) Right-click and click Save As.
   The File Save As dialog box appears.
2) Enter a new name for the file. Click Save.

Search and replace

Perform searches

1) Move the insertion point to where you want to start searching from.
2) Do one of the following:

- Click Edit, then click Find...

  -OR-

- Right-click, click Find...

The Find dialog box appears.

3) In the Find What text box, enter the search string.
4) In the Direction field, select Up, or Down.
5) Select any of the following options:

- Match Case to find only strings that match the case of the characters in your search string exactly.
- Regular expression if you entered a regular expression in the Find What text box.
- Wrap around search to continue searching after the end of the document has been reached.

6) Do one of the following:

- Click Find Next to continue searching without replacing a found item.

  -OR-

- Click Mark All to add a bookmark to all lines containing your search string.

Search for and replace a string

1) Move the insertion point to where you want to start replacing from.
2) Do one of the following:

- Click Edit, then click Replace...

  -OR-
- Right-click, click Replace...

The Replace dialog box appears.

3) In the Find What text box, enter the search string.
4) In the Replace With text box, enter the new string.
5) In the Direction field, select Up, or Down.
6) Select any of the following options:

Match Case to find only strings that match the case of the characters in your search string exactly.

- Regular expression if you entered a regular expression in the Find What text box.
- Wrap around search to continue searching after the end of the document has been reached.

7) Do one of the following:

- Click Find Next to continue searching without replacing a found item.

  -OR-
- Click Replace to replace the first instance of your search string.

  -OR-
- Click Replace All to replace all instances of your search string.

Cutting and pasting text

Move text

1) Select the text that you want to move by highlighting it.
2) Click Edit, then click Cut (CTRL+X).
3) Put insertion point where you want to paste the information.
4) Click Edit, then click Paste (CTRL+V).

The text is removed from the original location and appears in its new location.

Copy text

1) Select the text you want to copy by highlighting it.
2) Click Edit, then click Copy (CTRL+C).
3) Put insertion point where you want to paste the information.
4) Click Edit, then click Paste (CTRL+V).

The information is copied from its original location and appears in its new location.

Using bookmarks

You can set bookmarks to mark frequently accessed lines in your source file. Bookmarks are removed when the file containing them is closed or reloaded. Bookmarks store only the current line, not the column offset of the cursor. When a line containing a bookmark is deleted, the bookmark is also removed.

To set a bookmark:

1) Move the insertion point to the line where you want to set a bookmark.
2) Right-click, then click Toggle Bookmark.
   An indicator appears in the margin next to the text.

To set a bookmark at all lines that contain a specific string:

1) Right-click, then click Find.
2) In the Find What text box, enter the search string.
3) Click Mark All.
   An indicator appears in the margin of each line that contains the specified string.

To remove a bookmark:

1) Move the insertion point to the line containing the bookmark you want to remove.
2) Right-click, then click Toggle Bookmark.
   The indicator disappears from the margin next to the text.

Interacting with target processors

In the File menu are the commands for: working with target processors, adding files to a project, restarting programs, and saving projects.

NOTE: If you load a session file on a target that is different from the type it was created on, CodeScape loads the session without loading the program file.

You cannot add or remove targets during a session.

Connecting to a target processor

Initialize a target

When you run CodeScape it automatically connects to all detected targets (and prompts you to load the monitor if necessary).

Reset a target

You may be prompted to reset the target. If this occurs, do a soft reset. This restores the state of the target and re-initializes the monitors.

NOTE: You may be prompted to reload the Program File after resetting the target.

To do a Soft Reset:

- Click File, point to Reset, then click Soft Reset.
  -OR-
- In the Target window, right-click and point to Reset Target then click Soft Reset.

If a Soft Reset fails, you will be prompted to do a Hard Reset. This will reset the target and reload the monitors. You will be prompted to reload your project after a Hard Reset.

To do a hard reset:

- Click File, point to Reset, then click Hard Reset.
  -OR-
- In the Target window, point to Reset Target, then click Hard Reset.

NOTE: You will be prompted to reload the monitor after a Hard Reset.

Set the processor update rate

Set the processor update rate to tell CodeScape when to update information to the target.

If the update rate interrupts the target causing jitter in your program:

1) On the Processor Combo toolbar click A.
   The Processor Update Rate dialog box appears.
2) Select the Target that you want to set the update rate for.
3) Select the Processor on which your program is loaded.
4) Do one of the following:
   - Set the slider to Min.

     -OR-
   - Select Disable updates to this processor, to stop all region displays from updating.
5) Click OK.

NOTE: When you set this option it automatically overrides the region update rate set in the Region Configuration dialog box.

Add files to a project

Load a program file

1) Do one of the following:
    - Click File, then click Load Program File (CTRL+SHIFT+C).

      -OR-
    - In the Target window, right click and point to Load Program File…

   The Load Program File dialog box appears.
2) Select the Target from the *Target* list box.
3) Select the Processor from the *Processor* list box.
4) In the Program File text box, do one of the following:
    - Click browse to select a file.

      -OR-
    - Type the path and file name in the *Program File* text box.
5) In the *Load Options* text box, select one of the following radio buttons: Load Binary Only, Load Symbols/Debug only, or Load Both Binary and Symbols/Debug.
6) Select Reset Options to specify the options you want to use for re-loading the program file on the target. The default is Reset Enabled, Soft Reset.
7) Select Enable Run Options (disabled by default) then select:
    - Run, to run the specified program file when it is loaded.

      -OR-
    - Run to and enter an address or expression to run to, when the program file is loaded.
8) Click OK.

Saving and loading binary

Move large blocks of data in and out of memory

Use Save binary and Load binary to move large blocks of data in and out of memory. This is useful for loading and saving bitmaps, or processor specific code to a selected area of memory.

1) Do one of the following:
   - Click File, then click Load binary.

     -OR-
   - Click File, then click Save binary.

   The Write Binary to Memory dialog box appears.
2) Enter the Source file name.
3) Specify the start address.
4) Do one of the following:
   - Select End, then in the text box below, specify the end address.

     -OR-
   - Select Length, then in the text box below, specify the length of the address.
5) Click OK.

NOTE: You cannot write Binary to sensitive areas of memory such as invalid memory areas, read-only memory, and memory reserved for the monitors. If a sensitive area of memory is within a specified range, a message appears prompting you that the area of memory was skipped.

Load the binary part of a program file

1) Do one of the following:
   - Click File, then click Load Program File… (CTRL+SHIFT+C).

     -OR-
   - In the Target window, right click and point to Load Program File…

   The Load Program File dialog box appears.
2) Select the Target from the *Target* list box.
3) Select the Processor from the *Processor* text box.
4) Enter the location of the program file in the *Program File* text box.
5) Click Load Binary Only.
6) Click OK.

Load the symbolic debugging information part of a program file

1) Do one of the following:
   - Click File, then click Load Program File (CTRL+SHIFT+C).

     -OR-
   - In the Target window, right click and point to Load Program File…

   The Load Program File dialog box appears.
2) Select the Target from the *Target* list box.
3) Do one of the following:
   - Select the Processor from the Processor list box. Type the name and path of the program file in the *Program File* text box.

     -OR-
   - Click Browse and find the required file.
4) Click Load Symbolic Debugging information Only.
5) Click OK.

Restarting a program

Restart loads the binary part of the current program file and resets the PC to the entry point (if known). If the program file’s last modification time has changed, symbolic information is loaded.

Load the binary part of the current program file

- Click Debug, point to Execution then click Restart (CTRL+SHIFT+R).
  -OR-
- In the Target window, right-click and point to Execution, then click Restart.
  -OR-
- On the Debug toolbar, click ☰.
  -OR-
- Right-click in any region, click Execution, then click Restart.

Working with sessions

The commands for working with sessions are on the File menu.

CodeScape automatically saves the following debug information when you save a session:

- The configuration of Windows and Regions.
- The Project build information.
- The path for locating source files.
- The directory that contains CodeScape's source files.
- Which program files to load on each target processor.
- Any breakpoints that have been set.

NOTE: The next time that you open the session CodeScape will automatically load this information for you.

NOTE: If you load a session file on a target that is different from the type it was created on, CodeScape loads the session without loading the program file.

NOTE: You cannot add or remove targets during a session.

File menu commands

To open a new session:

1) Click File, Session New.
   The New dialog box appears.
2) Enter a file name using the extension. SSN.
3) Click OK.

To open an existing session:

1) Click File, Session Open...
   The Open dialog box appears.
2) Select a location in the *Look in* text box.
3) Select a file in the *File name* text box.
4) Click OK.

NOTE: If you open a new session you must load a program file.

To save a session:

- Click File, Session Save.

To save a session with a new name:

1) Click File, Session Save As…
   The Save As dialog box appears.
2) Enter a location the in the *Save In* text box.
3) Enter a new Session file name in the *File name* text box.
4) Click OK.

To close a session:

- Click File, Session Close.

NOTE: You may be prompted to save the current session before CodeScape opens a new session.

Recently used files list

The File menu displays a list of recently used session files.

To open a recent session file:

1) Click File.
2) Click the Session that you want to open from the list.

NOTE: You cannot hide this list or change the number of files displayed.

To exit CodeScape:

- Click File, Exit.

When you exit CodeScape, it prompts you to save the following debug information:

- The configuration of Windows and Regions.
- The Project build information.
- The path for locating source files.
- The directory that contains CodeScape's source files.
- Which program files to load on each target processor.
- Any breakpoints that have been set.

NOTE: The next time that you open the session CodeScape can load this information for you.

Working with projects

Use the Project menu to configure, make, and build your project.

The commands on the Project menu let you set up the following:

- A project build environment.
- An editor.
- The path for locating source files.
- A directory for CodeScape's source files.
- The fileserver refresh rate.

NOTE: CodeScape uses a project file (for GNU.C this is a makefile) to link compiled source files when you build your project.

Setting up a project build environment

To configure a project, specify the project build utility that you want to use, then provide it with a command line, a filename, and an environment file.

To configure, make, and build a project:

1) Click Project, then click Setup Project.
2) Do one of the following:
   - Type the project's name and path location in the Project File text box. (For example, makefile.)

     -OR-
   - Select a recent project from the Project File list. (For example, makefile.)

     -OR-
   - Click Browse, then search for a project file. (For example, makefile.)
3) Enter the project build utility's name and path location in the Program Build text box. (For example, make, or SNASMSH2.)
4) Type any command line parameters that should be passed to the make command in the Command Line Modifiers text box. (For example, -f for GNU make.)
5) Enter the project environment file's name and path location in the Environment File text box. This may be the same directory as the program build file.
6) Click OK to accept the project build environment.
7) Make the project current and build it in one of the following ways:
   - Click Project, then click Make.

     -OR-
   - On the Project Build window, right-click, click Make.

     -OR-
   - Press CTRL+M.

The Project Build window appears and automatically displays the specified build utility's output about the build. Any standard format errors and warnings are shown in the Project Build window.

If a build error occurs, double-click an entry to invoke the editor and open the source file at the line containing the error or warning. To advance to the next error or warning press F4. Some external editors do not support this option and will open without displaying the line at which the error occurred.

The environment file

To create an environment file:

1) Start an MS-DOS window and create the environment that you require to run the project build file that may use, for example, Hitachi C, GNU C, or SNASMSH2.
2) Create a file that contains the environment strings required by the project build file. For example, to create a file that contains the environment strings required to run the Hitachi tools, type : **`set>hitachi.env`**

The variables you can set for GNU C

Use the variable:	To specify:
TMPDIR	the directory to use for temporary files.
GCC_EXEC_PREFIX	a prefix to use in the names of the subprograms executed by the compiler.
COMPILER_PATH	path of the compiler.
LIBRARY_PATH	the path of the library.
C_INCLUDE_PATHCPLUS_INCLUDE_PATHOBJC_INCLUDE_PATH	
DEPENDENCIES_OUTPUT	how to output dependencies for the makefile.

To make, and build a project:

Make the project current and build it in one of the following ways:

- Click Project, then click Make.
  -OR-
- On the Project Build window, right-click, click Make.
  -OR-
- Press CTRL+M.

The Project Build window appears and automatically displays the specified build utility's output about the build. Any standard format errors and warnings are shown in the Project Build window. If a build error occurs, double-click an entry to invoke the editor and open the source file at the line containing the error or warning.

NOTE: The Target and Project Build windows can be docked at the top and bottom of the main window, or left free floating.

Setting up an editor

The default editor is CodeScape's Edit region where you can edit existing files and create new ones, but you can configure CodeScape to use an external editor.

CodeScape supports the following external editors: Notepad, MS-DOS Editor, Multi-Edit for Windows, Multi-Edit for DOS, Codewright, Brief, and Vi for MS-DOS/UNIX. You can add and remove editors in this list.

When you select a default external editor, CodeScape displays the following:

- The manufacturer's default installation path location in the Editor Path text box.
- The command to invoke the editor in the Editor Command text box.
- The editor's command line parameter to go to a line. For example, if you select Multi Edit for Windows, %f /L%l appears in the Editor Arguments text box.

When you open a file in the editor, CodeScape replaces %f with the file's name, and %l with the first line of that file. (Older versions of CodeScape use xxxxx instead of %l to represent the first line of a file.)

If a build error occurs when you make and build your project, CodeScape replaces %l with the line number containing the error or warning and displays it in the Project Build window. You can scroll through the information as it is generated, or press F4 to move through any listed errors one at a time.

If the editor you select does not support this option, leave this field blank.

NOTE: To remove an editor: in Editor Name list select the editor that you want to remove, click Remove. To edit the string, without removing the editor from the list, press Delete.

Setting up an external editor

1) Click Project, then click Setup Editor.
2) Enter the name of the editor in the Editor Name list.
3) Enter the editor's path location in the Editor Path text box.
4) Enter the command followed to invoke the editor in the Editor Command text box.
5) Enter %f, then the editor's command line parameter to go to a line, then %l in the Editor Arguments text box.

   When you open a file in the editor, CodeScape replaces %f with the file's name, and %l with the first line of that file. (Older versions of CodeScape use xxxxx instead of %l to represent the first line of a file.)

   If a build error occurs when you make and build your project, CodeScape replaces %l with the line number containing the error or warning and displays it in the Project Build window. If the editor you select does not support this option, leave this field blank.
6) Do one of the following:

   - If you selected a Windows based editor, select the Editor Is Windows Based check box.

     -OR-
   - If you selected an MS-DOS editor, clear the Editor Is Windows Based check box.
7) Click OK.

   If you have set up a new editor, CodeScape automatically adds it to the Editor Name list when you click OK.

NOTE: To remove an editor: in Editor Name list select the editor that you want to remove, click Remove. To edit the string, without removing the editor from the list, press Delete.

NOTE: Always save files edited in an external editor before using CodeScape's Make option to compile and build your project in CodeScape.

Setting up the project commands

Set the path for locating source files

In the Source File Search Path dialog you can: set, change, or remove a directory path name for CodeScape to look for your program's project files.

1) Click Project, then click Edit Source Path…
   The Source File Search Path dialog box appears.

2 Do one of the following:

- Type in the Path of the Source files.

  -OR-

- Click Browse and select the required directory.

2) Click Add.

Remove a path from the Source File Search Path

1) Click Project, then click Edit Source Path…
   The Source File Search Path dialog box appears.
2) Do one of the following:
   - Type in the path of the Source files.

     -OR-

   - Click Browse and select the required directory.
3) Click Remove.

Set a directory for CodeScape's source files

Set or change relative path name of the default directory for your fileserver based operations in the Set fileserver directory dialog box. (This can be the same as the directory you set in Source File Search Paths.)

Do the following:

1) Click Project, then click Set FileServer Root Directory.
2) Enter the Path of the default directory that you wish to use for fileserver operations.
3) Click OK.

NOTE: If the display update rate interrupts the target when it is loading information to the fileserver directory on you computer, click Project then select Enable Fileserver Optimization.

NOTE: Any configuration commands you set are saved in a session file when you exit including software breakpoints and watches, and program update rates.

Enable fileserver refresh rate optimization

If the display update rate interrupts the target when it is loading information to the fileserver directory on you computer, click Project then tick *Enable Fileserver Optimization*.

This disables updates to the display while data is being transferred from the target to your computer. This is useful if, for example, if you are loading a large bitmap from the target to the fileserver directory on your computer.

NOTE: You can toggle this option on or off.

Debugging

Debugging operations are:

- Tracing through your program code.
- Checking variables and structures.
- Adding and configuring breakpoints to control program execution.
- Simulating a target's processor operations to optimize tight assembler loops in your program.
- Profiling the run-time behavior of a program file to find out where it spends its time, and how functions are called when it executes. You can use information generated by the Profiler to identify any inefficient sections of code.

NOTE: Before you edit a program file from a UNIX target, convert it to a DOS readable format using a utility such as to_dos (use to_unix to return the file to a UNIX format).

NOTE: Thetoolbarsprovideaccesstothemaindebuggingfunctions. Use the Toolbar Configuration check box to show or hide toolbars.

Running and stopping programs

Run a program

1) In the Target region, select the processor where your program is loaded.
2) Do one of the following:
   - Click Debug, then click Run (F9).
     -OR-
   - Right-click, point to Execution then click Run.
     -OR-
   - On the Debug toolbar, click .

NOTE: Program execution will run until you stop it, or if an error occurs.

NOTE: When your program is running you can stop it by pressing F9.

Run all processors simultaneously

Do one of the following:

- Click Debug, point to Execution then click Run All (CTRL+F9).
  -OR-
- Right-click, point to Execution then click Run All.
  -OR-
- On the Debug toolbar, click .

NOTE: Program execution will stop at a breakpoint, or if an error occurs.

Run a program to the cursor position

In an active Source or Disassembly region, do one of the following:

- Click Debug, point to Execution then click Run to Cursor (ALT+F9).
  -OR-
- Right-click, point to Execution then click Run to Cursor.
  -OR-
- On the Debug toolbar, click →I.

You can add breakpoints using Run to Cursor at any time during program execution, program execution stops when a breakpoint is encountered. Program execution will stop at the cursor position, or at a breakpoint, or if an error occurs.

Run a program until it executes a specified address

1) Do one of the following:
   - Click Debug, point to Execution then click Run to Address (SHIFT+F9).
     -OR-
   - Right-click, point to Execution then click Run to Address.
     -OR-
   - On the Debug toolbar, click ⌨.
2) Type an address in the *Expression* text box of the Run to Address/Instructions dialog box.
3) Click OK.

The address is the name of a function or, an expression that resolves to an address.

You can add breakpoints using Run to Address at any time during program execution, program execution stops when a breakpoint is encountered. Program execution will stop at the specified address, or at a breakpoint or if an error occurs.

Stop a program

To stop a program running, in the Target region, select the processor on which your program is loaded, then do one of the following:

- Right-click in the Target window, point to Execution then right-click then click Stop.
  -OR-
- On the Debug toolbar, click STOP.
  -OR-
- Click Debug, point to Execution then click Stop.
  -OR-
- Press F9.

NOTE: The processor will stop immediately.

Stop all programs simultaneously

To stop all programs running simultaneously, do one of the following:

- Right-click in the Target window, point to Execution then, click Stop All.
  -OR-
- On the Debug toolbar, click ▤.
  -OR-
- Click Debug, point to Execution then click Stop All.

NOTE: All processors will stop immediately.

Stepping into (tracing) code

Trace functions are available either on the debug toolbar or on shortcut keys.

You can choose either source level tracing in an active Source region, or instruction level in an active Disassembly region. If you trace without a Source or Disassembly region open, tracing acts as if a Disassembly region is open.

NOTE: You can trace a program at any time while debugging. All trace operations immediately stop program execution.

NOTE: All trace operations can be interrupted by breakpoints.

Single step a line of source code

Do one of the following:

- Click Debug, then click Step (F7).
  -OR-
- On the Debug toolbar click .

In an active Disassembly region (or any non-source region), the target executes the instruction at the PC.

In an active Source region, the target executes the instruction at the PC. It stops when all low-level assembly instructions generated by the single source instruction have been executed. This includes:

- All instructions for a source macro instruction.
- Any C instructions that generate several assembler instructions.

Subsequent source lines (called by the current source line) may be in a different function or file, as determined by the execution flow.

NOTE: Execution trace history is generated when single stepping.

NOTE: A trap instruction is treated as a subroutine (a BSR or JSR). Trap 32 is reserved by CodeScape and treated as a single instruction. Use Forced Step Into to step into the trap 32 routine. Stepping into trap 32 may cause the monitor to fail.

Animate Step Run

Select Animate Step Run (default) to update all regions as each instruction executes.

- Click Debug, then click Animate Step Run.

CodeScape will trace instructions and display updated information in the active window until a breakpoint occurs, or until another command is issued, for example start/stop.

Step Run Until...

1) Do one of the following:
   - Click Debug, then click Step Run Until…
     -OR-
   - On the Debug toolbar click, .
2) Enter an expression to run to in the Expression Evaluator.

CodeScape will trace instructions and display updated information in the active window until a breakpoint occurs, or until another command is issued, for example start/stop.

NOTE: If the expression evaluates to a zero result, tracing continues.

Force step a line of source code at the disassembly level

Do one of the following:

- Click Debug, then click Forced Step Into (SHIFT+F7).
  -OR-
- On the Debug toolbar click .

In a Disassembly region, Forced Step Into causes individual assembly instructions to be traced one at a time where this is normally not allowed, for example stepping into a trap 32.

In a Source region, the target executes the instruction at the PC with the current register values then stops at each individually generated assembler instruction. Each individual assembler instruction is traced using the disassembly-level Single Step instead of the source-level Single Step.

Step into mechanism, that is a Trap, Line-A, Line-F, or subroutine is entered and program execution halted inside. In the case of a single source instruction generating many assembly instructions you will need to press SHIFT+F7 several times on the source instruction before progressing to the next source instruction.

Undo a step

- Click Debug, then click Unstep (CTRL+F7).
  -OR-
- On the Debug toolbar click, .

CodeScape keeps a history of trace actions. Trace history is built-up and discarded automatically.

When you Unstep, only the current state of the processor and memory contents are untraced. You can Unstep:

- Instructions that are executed as a series of individual disassembly instructions.
- Traces in Source and Disassembly regions as long as there is a trace history left.

NOTE: You cannot use Unstep if blocks of instructions are stepped over.

NOTE: Where code is stepped over, all trace history to this point is lost.

Stepping over code

Execution trace history is generated when stepping over. This is useful when you need to undo a step operation. Step Over performs a single step if Step Over is not relevant in the current context.

Step over a line of source code

Do one of the following:

- Click Debug, then click Step Over (F8).
  -OR-
- On the Debug toolbar click .

In disassembled code, the target executes the instruction at the PC then stops. A Trap, JSR or BSR is treated as a single instruction and program execution halted on the next instruction in memory when the routine is complete.

In source code, the target executes the instruction at the PC then stops when the source file reference has changed. When stepping over a function call, the entire function is executed. Execution is halted on the next source line.

NOTE: *You cannot step over conditional branches.*

NOTE: *Execution trace history is generated when stepping over.*

Stepping in and out of code

Use Step Run In to run to then stop at the start of each successively nested function calls. Use Step Run Out to run to and stop after each successively nested function call has completed.

Step Run In

- Click Debug, then click Step Run In (SHIFT+F7).
  -OR-
- Right-click in a region, click Execution, then click Step Run In.
  -OR-
- On the Debug toolbar click .

CodeScape will run to the start of the next function and then stop.

Step Run Out

- Click Debug, then click Step Run Out (SHIFT+F8).
  -OR-
- Right-click in a region, click Execution, then click Step Run Out.
  -OR-
- On the Debug toolbar click .

CodeScape will run to the end of the current function and then stop.

Interrupting program execution

Stop a program running

Do one of the following:

- In the Target window, right-click and click Stop.
  -OR-
- On the Debug toolbar, click STOP.
  -OR-
- Press F9.

NOTE: All trace operations immediately stop program execution.

Breakpoints

CodeScape has extensive software and hardware debugging features including breaking on data accesses within memory ranges and on external peripheral access.

NOTE: All breakpoint operations can be performed at any time through the Configure breakpoint(s) dialog box.

Adding breakpoints

You can add a breakpoint in Source, Disassembly, Memory, and Watch regions. You can add breakpoints at any time during program execution. Program execution stops when a breakpoint occurs. Add breakpoints using the menus, shortcut menus, or toolbars.

To add a code breakpoint:

1) Right-click, click Goto Address.
2) On the Breakpoint toolbar, click ✋ to set a breakpoint.
3) Right-click, click Execution, click Run. Your program will run until the breakpoint occurs.

NOTE: You can set a maximum of two Hardware breakpoints for each SH2 processor on a Sega Saturn target.

Add a breakpoint at the current cursor position

In a Source or Disassembly region you can add code breakpoints. In a Watch or Memory region you can add data breakpoints.

In any region, place the cursor in the required position then:

- Click Debug, then point to Breakpoints then click Toggle Breakpoint (F5).
  -OR-
- Right-click, point to Breakpoints then click Toggle Breakpoint.
  -OR-
- On the Breakpoint toolbar, click ✋.

When a breakpoint is set and enabled in a Source or Disassembly region, the breakpoint set icon, ✋, appears in the first column. When a breakpoint is disabled, the breakpoint disabled icon, ✋, appears in the first column.

When a watched variable is visible in the Watch region, the watched variable icon appears. In a Memory region the background color of the specified address changes.

A breakpoint is set with the following default behavior:

- Code breakpoint execution is halted once it has been triggered and no other action, such as logging, is performed. Code breakpoints are implemented in hardware if a ROM address is encountered or software otherwise.
- Watch breakpoints are triggered by any read or write data access to hardware. A message appears when the breakpoint has been triggered and all conditions have been met by default.

All breakpoint locations are tested to make sure that they are placed and configured correctly. If a problem is found a message appears prompting you to re-configure the breakpoint.

NOTE: To change the default behavior of a breakpoint see To Configure a Breakpoint.

NOTE: You can add a breakpoint only to a line that generates code. (Shown by a '.' in column one of a Source region or Watch region, or at any point in the Disassembly region.)

Removing breakpoints

In any region, place the cursor on the required breakpoint then:

- Click Debug, point to Breakpoints then click Toggle Breakpoint (F5).
  -OR-
- Right-click, point to Breakpoints then click Toggle Breakpoint.
  -OR-
- On the Breakpoint toolbar, click ✋.
  -OR-
- In the Configure breakpoint(s) dialog box (CTRL+F5), select the breakpoint that you want to disable then click Remove.

The breakpoint set icon, ✋, will disappear from the code window.

Remove all breakpoints

In any region, place the cursor on the required breakpoint then:

- Click Debug, point to Breakpoints then click Remove all Breakpoints (SHIFT+F5).
  -OR-
- On the Breakpoint toolbar, click ✖.
  -OR-
- Right-click, point to Breakpoints then click Remove all Breakpoints.
  -OR-
- In the Configure breakpoint(s) dialog box , click Remove All (CTRL+F5).

Enabling and disabling breakpoints

Enable a disabled breakpoint

In any region, place the cursor on the required breakpoint then:

Click Debug, point to Breakpoints then click Enable Breakpoint.
-OR-

- Right-click, point to Breakpoints then click Enable Breakpoint.
  -OR-
- On the Breakpoint toolbar, click ✋.
  -OR-
- In the Configure breakpoint(s) dialog box (CTRL+F5), click Code Settings and select Breakpoint Enabled.

The breakpoint set icon will change from ☒ to ✋ to show that the breakpoint is enabled.

Disable an enabled breakpoint

In any region, place the cursor on the required breakpoint then:

- Click Debug, point to Breakpoints then click Disable Breakpoint.
  -OR-
- Right-click, point to Breakpoints then click Disable Breakpoint.
  -OR-
- On the Breakpoint toolbar, click ☒.
  -OR-
- In the Configure breakpoint(s) dialog box (CTRL+F5), click Code Settings and clear Breakpoint is Enabled.

The breakpoint set icon will change from ✋ to ☒ to show that the breakpoint is disabled.

Enable all breakpoints

Do one of the following:

- Click Debug, point to Breakpoints then click Enable all Breakpoints (CTRL+SHIFT+F5).
  -OR-
- Right-click, point to Breakpoints then click Enable all Breakpoints.
  -OR-
- On the Breakpoint toolbar, click .

Disable all breakpoints

Do one of the following:

- Click Debug, point to Breakpoints then click Disable all Breakpoints (CTRL+ALT+F5).
  -OR-
- Right-click, point to Breakpoints then click Disable all Breakpoints.
  -OR-
- On the Breakpoint toolbar, click .

Resetting breakpoints

Reset all breakpoints

Do one of the following:

- Click Debug, point to Breakpoints then click Reset all Breakpoints (ALT+F5).
  -OR-
- Right-click, point to Breakpoints then click Reset all Breakpoints.
  -OR-
- On the Breakpoint toolbar, click .

NOTE: Resetting all breakpoints sets all conditional values, including the current count, to their starting conditions.

Reset the trigger count for a breakpoint

- In the Configure breakpoint(s) dialog box select the breakpoint, click Reset.

Reset only the current value of the count for a breakpoint

- In the Configure breakpoint(s) dialog box select the breakpoint, click General Conditions and click Reset Current.

Configuring breakpoints

CodeScape enables breakpoint configuration including data accesses within memory ranges and breakpoints on external peripheral devices.

To configure a breakpoint:

- Click Debug, point to Breakpoints then click Configure Breakpoint(s)… (CTRL+F5).
  -OR-
- Right-click, point to Breakpoints then click Configure Breakpoint(s)…
  -OR-
- On the Breakpoint toolbar, click .

NOTE: You can add a breakpoint and configure it manually using the Configure breakpoint(s) dialog box.

NOTE: Watch breakpoints trigger on data access, and code breakpoints trigger on the fetch-execute phase of the instruction cycle.

The Configure breakpoint(s) dialog box

In the Configure breakpoint(s) dialog box you can:

- Add, remove, and configure code and watch breakpoints.
- Enable or disable a breakpoint, set its location, and the resources it will use.
- Specify when a breakpoint will occur.
- Configure a prompt for when a breakpoint occurs.

Using the Code Settings tab

Code breakpoints trigger on instruction execution. When a code breakpoint triggers the PC is at the same instruction in the pipeline. The Code Settings tab becomes available when you add or select a code breakpoint to configure.

1) Do one of the following:
   - Select a code breakpoint to configure from the list.

     -OR-
   - Click code to add a code breakpoint to configure.
2) Select Breakpoint Enabled (default), to enable a breakpoint.

   You may be prompted to re-configure a disabled breakpoint. This can occur during code execution, restoring sessions, or when attributes could not be validated when configuring commands within this dialog box. A disabled breakpoint does not affect code execution or use any hardware resources.
3) Specify the position in memory where the code will stop on execution. In the *Location Expression* text box:
   - Enter the required expression.

     -OR-
   - Click Define. The Breakpoint Location Expression dialog box appears. Evaluate the expression to set the location address.
4) Then do one of the following:
   - Select C/C++, to use C/C++ expression syntax.

     -OR-
   - Select Assembly, to use SHx assembly language syntax.
5) In the Implementation mechanism group box:
   - Select Automatic and CodeScape will manage breakpoint resources. Breakpoints are implemented in software by default. If this is not possible then hardware resources are used.

     -OR-
   - Select Software to specify a software breakpoint.

     -OR-
   - Select Hardware to set a hardware breakpoint that is specific to your target processor.

NOTE: You can set a maximum of two Hardware breakpoints for each SH2 processor on a Sega Saturn target.

Using the Watch Settings tab

Watch (data) breakpoints trigger on memory data access. When a Watch breakpoint triggers the PC is several instructions ahead of that breakpoint in the pipeline.

The Watch Settings tab becomes available when you select or add a watch breakpoint to configure.

1) Do one of the following:

   - Select a watch breakpoint to configure from the list.

     -OR-
   - Click Watch to add a watch breakpoint to configure.

2) Select Breakpoint Enabled (default), to enable a breakpoint.
   You may be prompted to re-configure a disabled breakpoint. This can occur during code execution, restoring sessions, or when attributes are not validated during command configuration in this dialog box. A disabled breakpoint does not affect code execution or use any hardware resources.

3) Specify the position in memory where the breakpoint is accessed. In the Location Expression text box:

   - Enter the required expression.

     -OR-
   - Click Define. The Breakpoint location expression dialog box appears. Evaluate the expression to set the location address.

4) Select Include Data Condition to change the Watch Access breakpoint into a Watch Data breakpoint that uses the features of the UBC (User Break Controller). Enter the required Data Expression, then click Define. The Breakpoint watch data expression dialog box appears. Evaluate the expression to set the location address.

5) Then do one of the following:

   - Select C/C++, to use C/C++ expression syntax.

     -OR-
   - Select Assembly, to use the Assembler’s expression syntax.

6) In the Implementation mechanism group box:

   - Select Automatic and CodeScape will manage breakpoint resources. Breakpoints are implemented in software by default. If this is not possible then hardware resources are used.

     -OR-

- Select Software to specify a software breakpoint.

  -OR-

- Select Hardware to set a hardware breakpoint that is specific to your target processor.

7) Under Access Size, enter the Access Size required (the default is Any). When you use Toggle to add a watch breakpoint its size, if known, will be used instead of Any.
8) Under Access Type, select the Access Type required. The default is Both read and write access.

NOTE: If you place a watch (data) breakpoint on a member of a union it will trigger for all members of that size, regardless of type. This also applies to anonymous unions, except that two members of the same size appear as two variables sharing the same address in memory.

Using the General Conditions tab

The General Conditions tab is for defining conditions that must be valid before a breakpoint is triggered. You can condition a breakpoint by memory access type and data value, and confirm that it executed on the correct trigger count.

1) To use a conditional expression select Include Conditional Expression.
2) Do one of the following:

   - Enter a valid expression in the Include Conditional Expression text box.

     -OR-
   - Click Define to open the Breakpoint condition expression dialog box, then define the expression.
3) Do one of the following:

   - Select C/C++, to use C/C++ expression syntax.

     -OR-
   - Select Assembly, to use the Assembler's expression syntax.

   The expression is evaluated for a logical result where a value of zero represents false and non-zero values represent true.
4) Select Include Trigger Count Condition to include the trigger condition. The condition is true when the Current Count reaches the specified Trigger Count value.
5) Enter the value for the Current Count to reach to make the Trigger Count Condition true.
6) Under Counters, check that the value in the Current box matches the value you set in the Trigger box. Click Reset Current to return the current count to zero.
7) Select when to increment the count. The default is to increment the Current Count whenever the breakpoint occurs or is evaluated.
8) If both expression and count conditions are included, select when to break in the expression. The default is OR.

Using the Trigger Actions tab

Use the commands on the Trigger Actions tab to specify how CodeScape responds when a breakpoint has triggered.

Select any or all of the following radio buttons:

- Select Halt execution when conditions match to stop the program executing when the breakpoint conditions have been met. Clear this check box to continue execution after all other requested actions have been performed.
- Select Single shot - breakpoint is discarded when conditions match to discard the breakpoint after it has been triggered and all conditions have been met.
- Select Message box prompt when conditions match. CodeScape will display a message when the breakpoint has been triggered and all conditions have been met.
- Select Beep when conditions match. Your computer will beep when the breakpoint has been triggered and all conditions have been met.
- Select Cause processor simulation to and specify whether the Simulator should Start or Stop when the breakpoint has been triggered.
- Select Log Expression and choose either to produce a log when the breakpoint has been triggered or every time. Enter a valid Log expression.

NOTE: If there is no Log region for the Target Processor, CodeScape creates one.

Using the Advanced tab to specify options for a code breakpoint

NOTE: The Location Address text box is read-only. To set the location, click the Code Settings tab.

NOTE: The ASID Mask Selector field is set to its default state and cannot be configured. It will be enabled in future releases.

On the Advanced tab are commands for using the Hardware Implementation Mechanism. These commands apply only to Watch breakpoints and Code breakpoints.

1) Select Location Mask to specify which bits of the Location Address to mask out. Set Location Mask bits to 1 to ignore the corresponding Location Address bit, 0 otherwise.
2) In the *Break Mode* text box select either:
    - Before Execution.

      -OR-
    - After Execution.

Using the Advanced tab to specify options for a watch breakpoint

NOTE: The Location Address text box is read-only. To set the location, click the Code Settings tab.

NOTE: The ASID Mask Selector field is set to its default state and cannot be configured. It will be enabled in future releases.

On the Advanced tab are commands for using the Hardware Implementation Mechanism. These commands apply only to Watch breakpoints and Code breakpoints.

1) Select Location Mask to specify which bits of the Location Address to mask out. Set Location Mask bits to 1 to ignore the corresponding Location Address bit, 0 otherwise.
2) In the *Data Mask* text box, set Data Mask bits to 1 to ignore the corresponding Data Address bit, 0 otherwise.
3) In the Bus cycle field, select the bus cycles to include, either CPU, or Peripheral (DMA), or both.

Using the Global tab to specify the debug environment for Hitachi SH4-EVA processors

On the Global tab are commands for setting the target processor's debug environment. The Global tab appears when you connect to an SH4-EVA target processor and you can specify any of the available options.

1) In the Global ASE Break Conditions for SH4-EVA CPU field:

    - Select Enable on-chip access detection and CodeScape will generate an on-chip I/O exception.

    The values displayed are the last on-chip address accessed, and the last on-chip data access when the exception occurred.

    - Select Enable break after LDTLB instruction execution and CodeScape will generate an LDTLB instruction break.

    The values displayed are the last PTEH loaded, and the last PTEL loaded into the MMU.

2) In the Global UBC Exception Handler Option field, select Use DBR vector (default).

    CodeScape will use the debug stub default exception handler for UBCs. This lets you define exception handling routines in your program, and to modify the VBR without affecting the behavior of UBC breakpoints.

Breakpoint expression format

CodeScape has a powerful expression formatting facility for controlling the display of expressions in Log windows. Formatting is controlled with formatting expressions which work in a similar way to the C ‘print(f)’ function. The expressions consist of a formatting string followed by any number of comma separated expressions. The expressions are numbered from 0 and can be any valid slot expression referencing register names or memory locations. The syntax for a formatting expression is:

["FormattingString"|FormattingString,] [Expression]...

Format specification

Start a specification with a ‘%’ symbol and follow with one or more command modifiers. Terminate the specification with a with a format specifier. The formatting string is one or more format specifications.

To separate multiple specifications:

- Use spaces.
  -OR-
- Enclose the sequence in quotes and separate each specification with a comma.

The syntax for the format specification is:

```
%[Pointer][Width][Repeat]Specifier
```

% denotes the start of a format specification. The Specifier controls the optional modifiers that affect the expression’s display.

Command modifiers controlled by the Specifier

This expression item:	Denotes an command modifier that:
Pointer	Repositions the parameter pointer.
Width	Specifies the display width of the expression.
Repeat	Specifies the number of items to display.

Each expression must have the same number of operands as format specifications (‘%’ characters). Insufficient operands will cause CodeScape to generate an error. All expression operands must evaluate. If an operand evaluates to a section relative address the string appears as "SectionName:Value".

Format specifier character

The format specifier character is for controlling pointers and formatting instructions that affect the display of an expression.

Format specifier characters and their effects

Use this character:	For this effect:
D, d	Decimal signed integer.
C, c	ASCII character.
U, u	Decimal unsigned integer.
O, o	Octal unsigned integer.
X, H	Hexadecimal unsigned integer using ‘A’-‘F’.
x, h	Hexadecimal unsigned integer using ‘a’-‘f’.
S, s	Pointer to null terminated ASCII string.
T,t	Displays the time in the form HH/MM/SS
!	Display parameter expression as a string.
I, i	Pointer to instruction to disassemble.

Using the pointer parameter, examples

Use this character:	For this effect:
%d	Format parameter as a decimal signed integer.
%u	Format parameter as a decimal unsigned integer.
%H	Format parameter as a hexadecimal unsigned integer (using ‘A’ ‘F’).

The pointer modifier

A parameter pointer holds the position of the current expression, the first expression starting at position 0. The command pointer modifier repositions the parameter pointer and follows directly after the '%' symbol. The modifier consists of a decimal number, optionally preceded by a '+' or '-' symbol and terminated with a '#' symbol.

Syntax

[+|-]Number#

The symbol syntax of the pointer modifier

Use this character:	To:
+|-	Reposition the parameter pointer relative to its current position.
Number	Denote an absolute value for the parameter pointer or the size of the relative movement if used in conjunction with '+' or '-'.

NOTE: If you set the pointer to a value before the first pointer parameter, CodeScape sets it to the first parameter. If you set the pointer to a value after the last parameter, subsequent specifiers become invalid and are copied verbatim into the display string.

Examples

A formatting string to show three parameters as decimal signed integers in reverse order:

```
%2#d %1#d %0#d
```

A formatting string that displays its parameter in hexadecimal, and then in decimal:

```
"%0#x,%-1#d"
```

The width modifier

The optional width modifier specifies the display width for the expression. It follows the # modifier (or '%' symbol if no pointer modifier is specified). The width is either a decimal number or the value of the next parameter expression.

Syntax

[-][Number|*]

The symbol syntax of the width modifier

The symbol syntax:	Denotes that the field width is:
-	Left justified. If the '-' symbol is not set, the field is right justified.
Number	A decimal number. Prefix the number with a zero to pad the display field with zeroes.
*	Set by the value of the next parameter expression.

NOTE: For '%s' formats, the width specifies the maximum number of characters to display.

Using the width modifier, examples

The symbol syntax:	Denotes that the field width is:
%4x	As a 4 digit right justified hexadecimal unsigned integer (using 'a'-'f').
%-8s	As an 8 character left justified string.
%08X	As an 8 digit hexadecimal unsigned integer (using 'A'-'F') and pad with zeroes.
%3#-15S	As the 4th parameter as a 15 character left justified string.
%*s	As a string according to the value of the next parameter.
%4#*d	As a 4 digit right justified decimal signed integer according to the value of the next parameter.

The repeat modifier

The optional repeat modifier controls the number of items displayed and follows the pointer and width modifiers (if specified). The modifier consists of a '@' symbol followed by an optional size modifier and terminated with the number of items to be displayed.

Displayed items are separated with:

- A comma if the format specifier is decimal or octal.
- Spaces if the specifier is hexadecimal.
- No commas or spaces if the specifier is characters.

The endianess of the target processor is preserved when fetching multi-byte items.

NOTE: The repeat modifier has no effect if the format specifier is a string or instruction.

Syntax

@[Size]Number

where:

The symbol syntax of the repeat modifier

The symbol syntax:	Signifies:
@	The start of the repeat modifier.
Size	The size of items fetched from memory.
Number	A decimal number denoting the number of items to be displayed.

Optional command-line characters

The character:	Signifies a:
b	Byte
w	Word
t	Triple
l	Long

Simulating a target

The Simulator is an optimizing tool for Hitachi SH series processors. It uses real targets for the Memory and Register regions.

When you single step in a Simulator region the cursor is shown at the instruction currently executing in the pipeline. During simulation the PC fetches instructions ahead of the current instruction. Some instructions are not executed because of changes in the program flow. For example, instructions fetched after a branch. When you single step in any other CodeScape region, the cursor is shown at the PC (program counter).

The Simulator enables you to optimize timing critical sections of Assembly code by simulating a target's processor operations. For example, you can set breakpoints to simulate a function that is part of a loop in your program.

NOTE: You cannot run the Profiler and the Simulator at the same time.

NOTE: For details about processor pipeline operations, refer to the relevant Hitachi Programming Manual. For a copy of the manual, contact your Hitachi supplier, or connect to the Hitachi Japanese web site at http://www.hitachi.co.jp

NOTE: Memory timings do not model SDRAM banks (6000000-607FFFF, 6080000-60FFFFF).

Using the Simulator's shortcut menu

Select:	To:
Highlight Cache Misses	See in which slot a pipeline operation missed the cache.
Highlight Pipeline Stalls	See in which slot a pipeline operation stalled.
Show Stall Type	Show the type of stall generated.
Show Only Active Stages	Show active / all pipeline stages used.
Show Uppercase	Show instructions in upper case.
Show Symbols	Show operand values as symbols.

Select:	To:
Show EAs & Lits.	Show the effective address and literals.
Print	Print the results of program simulation.
Save to file...	Save the results of program simulation to a file.
Execution	Run, stop, and restart your program. Run your program to the cursor position, or until it executes a specified address. Run all of your program files simultaneously. Stop all of your programs running simultaneously. Use the single stepping options, or run the step options.
Breakpoints	Toggle a breakpoint on or off. Enable, disable, configure, reset, and remove breakpoints.

Running the Simulator

When you run the Simulator it generates information about the pipeline operation for each Assembly instruction. It also highlights any loss of performance in the processor cache and the pipeline. Use the Simulator’s shortcut menu commands to configure the Simulator and access the debugging functions.

In the Target region, select the processor that you want to simulate:

- Select Debug, click Simulate Processor.
  -OR-
- Right-click in the Target region, then click Simulate Processor.
  -OR-
- Press CTRL+ALT+Z.

NOTE: You cannot run the Profiler and the Simulator at the same time.

Running restrictions

The Simulator does not support the following features:

- DMA.
- Timers.
- Division unit.
- Power down mode.
- Memory mapped registers except for the CCR.
- External interrupts.

The Simulator disables external interrupts when it is running. If you use the sleep instruction you cannot wake the Simulator from sleep/standby mode.

Internal exceptions and interrupts are:

- Simulate NOP (no operation) and inform CodeScape of the appropriate exception.
- TRAPA 32 which is used for FileServer operation, software breakpoint operation, and hardware breakpoint operation.
- Address errors, illegal slot, and invoked instructions as reported on the processor status line.

Debugging operations in the Simulator

All of CodeScape's debugging functions are available when the Simulator is running. The debugging functions include commands for: controlling program execution, stepping code, using breakpoints, and setting the cursor to the PC and visa versa.

When you single step in a Simulator region the cursor is shown at the instruction currently executing in the pipeline. During simulation the PC fetches instructions ahead of the current instruction. Some instructions are not executed because of changes in the program flow. For example, instructions fetched after a branch. When you single step in any other CodeScape region, the cursor is shown at the PC (program counter).

Simulation results

During program simulation the Simulator generates information about pipeline operation for each Assembly instruction. You can read any loss of processor performance from the simulation results shown in the Simulator's regions, or by printing the results.

Information generated by the Simulator

When you simulate your project each instruction is executed in a Simulated slot (time). As the Simulator steps through time a linear description of pipeline operation is shown in its regions:

- The Address in memory for each line of source code.
- The Op-code for each instruction.
- The CPU time taken to execute each instruction at an address in memory.
- The Disassembly of the op-code for each instruction.
- Slot information for each stage of pipeline operation:

    1) The vertical cursor indicates the time taken by the processor to execute an instruction for each slot.
    2) The horizontal cursor indicates the instruction that is being allocated time in the active slot.
- Processor status information.

Execution time

The execution time is the CPU time accumulated from the start of an instruction's ‘ex’ (execution) phase to the start of the next instruction's ‘ex’ phase.

Processor status information

Data displayed on the status bar for an active slot

This status area:	Describes:
Diagnosis:	The type of stall encountered and what caused it.
Cache:	Cache memory operation stalls which occur when there is a read/write miss.
System clock:	The total time taken for processor operations upto the current cursor position.

*NOTE: The information generated by the Simulator can be saved in a configuration file with the extension *.sim.*

Pipeline interaction

The Simulator evaluates an instruction’s functionality at the appropriate stage of the pipeline. The following instruction tells the processor to read 32 bits from the address stored in r0, then put the results in r3.

```
mov.L@r0,r3
```

When the instruction executes in a simulated slot (time) the following instruction stages are shown in the simulated pipeline:

```
IF|ID|EX|MA|WB
```

During the instruction’s execution the following operations take place:

- At the IF stage the op-code for the instruction is read from memory.
- At the ID stage the instruction is decoded.
- At the EX stage instruction execution starts, and the contents of register r0 is read.
- At the MA stage memory is accessed at register r0 and the value is stored on the data bus.
- At the WB stage the value stored on the data bus is written back to memory at register r3.

Instruction execution in the Pipeline region

The mnemonic:	Indicates:
IF	Instruction fetch.
if	Dummy instruction fetch where external memory is not accessed.
ID	Instruction decoded / issued (All SH series processors.)Instruction decoded / issued / register read. (SH4 processors only.)
D	Decode stage locked.
d	Register read only. (SH4 processors only.)
EX	Instruction execution.
SX	Execution phase, the SX stage used.
SX*	SX stage locked not used.
NA	Memory not accessed / no operation address.
MA	Memory accessed / operation address.

The mnemonic:	Indicates:
MAm	Memory accessed / multiplier use. (SH2 processors only.)
mm	Multiplier busy. (SH2 processors only.)
WB	Register write back (data stored to registers after operation).
F0	Floating point 0 stage accessed. (Special Stage inner product / transforms).
F1	Floating point 1 stage accessed.
F1*	Floating point 1 stage locked and not accessed.
f1	Floating point 1 stage partial usage (can overlap with other f1's but not F1).
F2	Floating point 2 stage accessed.
F3	Floating point 3 stage accessed. (Special Stage divide / square root).
FS	Floating point store / writeback.
>FPSCR<	Floating point status register updated.

*NOTE: In the *.sim file all of the instructions are represented bythemnemonicslistedaboveexcept,>FPSCR<whichisrepresented by FC, and Mam which is represented Mm.*

Processor operation

The CPU time accumulated by each slot is highlighted to show the state of the processor when an operation went off. Different colors and mnemonics' describe pipeline interaction at each phase.

Processor operation in the Pipeline region

An operation colored:	Indicates that the processor:
Black	Was OK
Red	Stalled
Blue	Missed the cache
Pink	Stalled and missed the cache

Pipeline stalls

Where a stage from one instruction is in contention with a stage of the next or previous instruction a stall occurs. This slows down the operation of the pipeline. Simulation may show a stall in the processor's pipeline.

Remove a stall, in one of the following ways:

- Reorder the instruction sequence to remove an Instruction sequence stall.
  -OR-
- Move an instruction address in memory to remove an Instruction alignment stall.

NOTE: For details about contention in instruction stages and execution states refer to the Hitachi Programming Manual for the 7600 Series.

Pipeline instruction stalls recognized by the Simulator

This symbol:	Shows this type of stall:	If possible, increase the speed of pipeline execution by:
r>	A memory access conflicting with an instruction fetch. (SH1 and SH2 processors only.)	To align instructions that access memory on longword boundaries.
W>	A write back from the registry when a memory access is incomplete.	So that instructions that follow memory loads do not immediately use the same destination register.
x>	A multiplier usage stall. (SH1 / SH2 processors only.)	So that instructions that use the multiplier execute non-consecutively.
i>	An instruction generated stall.	You cannot do anything about this stall type. TRAP, TAS, RTE always stall.
R>	Two instructions trying to lock the same register. (SH4 processors only.)	So that instructions using the same register execute sequentially to ensure that they are not dual issued.
s>	The SX stage of the instruction being in use.	So that the instruction that locks the SX stage executes before instructions that use the SX stage non-consecutively.
f>	A floating point pipeline stall, caused by multiple use of the: F0, or F1, or F3 stages.	So that it uses instructions in the F0, or F1, or F3 stages once.
c>	One or more control group instructions being dual issued.	You cannot do anything about this stall type.
g>	Instructions of the same type occurring together and causing a dispatch failure.	So that instructions of the same type (such as EX + EX, LS + LS, BR + BR, FE + FE) do not occur together.
?>	An unknown stall type.	

NOTE: For details about pipeline instruction stalls refer to the Hitachi Programming Manual for the 7600 Series.

Reading the results of simulation

The Simulator generates information about pipeline operation for each Assembly instruction during program simulation. Any loss of processor performance appears in the results shown in the Simulator’s regions. You can print the simulation results. (For an example of how to read the simulation results refer to the Simulator tutorial.)

Profiling program files

The Profiler is a powerful analysis tool that lets you examine the run-time behavior of program files written for Hitachi SH series processors.

You can configure the Profiler to analyze your program with two levels of detail: statistical and trace. The Profiler can help you to find out where your program spends its time, and how functions are called when it executes. You can use information generated by the Profiler to identify any inefficient sections of code.

NOTE: To ensure accurate results, set the debug stub to run with the cache off when you trace profile. To do this, run DACHECK. For more information refer to the Help supplied with DACHECK.

NOTE: You cannot run the Profiler and the Simulator at the same time.

Opening the Profiler

In the Target region, select the processor that you want to profile:

- Click Debug, then click Profiler.
  -OR-
- Right-click in the Target region, then click Profiler.
  -OR-
- Press CTRL+ALT+X.

Using the Profiler's shortcut menu

Select:	To:
File	Load or save program profile information. Program profiles are saved using the extension *.prf.
Enable Profiler	Start or stop profiling your program file.
Trace Tree Profile Display	Find out for each function, the functions that called it, and the functions it called.
Function Profile Display	Find out for each function, the functions that called it, and the functions it called. Also, how much time your program spent in each function, and how many times each function is called.
Function Profile Filter	Arrange the view to show one of the following: all functions, all tagged functions, or all untagged functions.
Untag All	Untag all currently tagged functions.
Sort	Arrange the column view of the active Function Profile Display.
Source Display	View your program's original source code.
Disassembly Display	View your program at instruction level (assembly code).
Rename Function...	Enter a new name for a specific function.
Profiler Display Setup...	Specify the profile display options.
Setup...	Specify options for Statistical Profiling, or EVA Trace Profiling.

NOTE: To ensure accurate results, set the debug stub to run with the cache off when you trace profile. To do this, run DACHECK. For more information refer to the Help supplied with DACHECK.

NOTE: If you choose Trace Tree Profile Display, you must run the Profiler before you run your program.

NOTE: To set a tag on a specific function, double-click it's entry in the Profiler.

Options on the Profiler's toolbar

To issue this command:	Click:
Start profiling the current program file.	ON
Stop profiling the current program file	OFF
Toggle the display between a Trace Tree Profile and a Function Profile.	
Switch the display from Show All functions to Show All Tagged functions.	
Switch the display from Show All Tagged functions to Show All Not Tagged functions.	
Switch the display from Show All Not Tagged functions to Show All functions.	
Go to the next tagged function in the list.	↓
Display the program file's original source code.	
Display the program file at instruction level (assembly code).	
Toggle the sort options.	a b

NOTE: To ensure accurate results, set the debug stub to run with the cache off when you trace profile. To do this, run DACHECK. For more information refer to the Help supplied with DACHECK.

NOTE: If you want to Trace Profile, you must run the Profiler before you run your program.

NOTE: To set a tag on a specific function, double-click its entry in the Profiler.

The profile display types

You can view the profile of your program using:

- The Trace Tree Profile Display.
  -OR-
- The Function Profile Display.

The Trace Tree Profile Display

The Trace Tree Profile Display tells you for each function; the functions that called it, and the functions it called. A Trace Profile also shows, the total amount of time your program spends executing each function, and how much time it spends in each function and its children.

NOTE: To ensure accurate results, set the debug stub to run with the cache off when you trace profile. To do this, run DACHECK. For more information refer to the Help supplied with DACHECK.

The Function Profile Display

The Function Profile Display lists the functions called. You can use the sort options on the shortcut menu to view the profile relative to: function hit, function count (and children), function clock cycle (and children). The Function Profile Filter lets you specify how you view tagged functions.

NOTE: To set a tag on a specific function, double-click its entry in the Profiler.

Setting the display

To specify the display options for the current profile:

1) Right-click in the Profiler, then click Profiler Display Setup.
   The Profiler Display Setup dialog box appears.
2) In the Column Data text box select one of the following options:

   - Display Both.
     -OR-
   - Display Counts / Cycles.
     -OR-
   - Display Percent.
3) In the Columns text box, select any of the following options:

   - Display Hits.
   - Display C1.
   - Display C1 + Children.
   - Display C2.
   - Display C2 + Children.
4) Click OK.

Configuring the Profiler

The options on the Profiler Setup Dialog let you specify how your profile information is generated.

To configure the Profiler:

1) Right-click in the Profiler, then click Setup.
   The Profiler Setup Dialog appears.
2) Do one of the following:

   - Select Statistical Profiling then specify the Sample Period, and how long you want the Profiler to run for.

     Statistical Profiling lists only the number of times a function or line of source code is called (hits), but it is the fastest way to profile your program.

     -OR-

   - Select EVA Trace Profiling then select the Performance Measurement Counters you want to use.

     EVA Trace Profiling lets you generate function and trace profiles providing more detail about how specific functions are called.
3) Select the Interrupt Profile Trace Filter to prevent the profiler generating data on interrupt or exception subroutines.

NOTE: For details about the Performance Measurement Counters options, refer to the Hitachi Programming Manual for your processor. For a copy of the manual, contact your Hitachi supplier.

NOTE: To ensure accurate results, set the debug stub to run with the cache off when you trace profile. To do this, run DACHECK. For more information refer to the Help supplied with DACHECK.

Debug operations in the Profiler

All of CodeScape's debugging functions are available when the Profiler is running. The debugging functions include commands for: controlling program execution, stepping code, using breakpoints, and setting the cursor to the PC and visa versa.

NOTE: You can only set one Start breakpoint and one Stop breakpoint for the Profiler.

Searching for a function

To search for a function:

1) Move the insertion point to where you want to start searching from.
2) Type the Search string in the current profile.
   The Profiler automatically finds, and displays the nearest match.
3) To continue the search press ENTER.

NOTE: You can search for strings, whole words, or parts of words.

NOTE: The Profiler looks for exact matches first, then the nearest matches in descending order.

Changing the name of a specific function

To rename a function:

1) Select the function that you want to rename.
2) Right-click, click Rename Function...
   The Rename Function dialog box appears.
3) Enter the new name for the selected function.
4) Click OK.

Expressions

The expression evaluator dialog is used for several operations, including: Edit Register, and Goto Address. In the dialog you can use the C/C++ expression evaluator or the Assembler’s expression evaluator.

C/C++ expressions

The C/C++ expression evaluator accepts expressions in a C-like format.

Operator precedence

Operator	Type	Usage	Description
( )	Primary		Parenthesis Brackets
[ ]	Primary	pointer[expr]	Subscripting
.	Binary	object.member	Member selection
->	Binary	pointer->member	Member selection
sizeof()	Unary	sizeof(expr)	Size of object.
sizeof()	Unary	sizeof(type)	Size of type
-	Unary	- expr	Unary Minus
+	Unary	+ expr	Unary Plus
~	Unary	~ expr	Bitwise NOT
!	Unary	! expr	Logical NOT
*	Unary	* expr	De-reference
&	Unary	& lvalue	Address of
*	Binary	expr *expr	Multiply
/	Binary	expr/expr	Divide
%	Binary	expr % expr	Modulo (remainder)
+	Binary	expr + expr	Add (plus)
-	Binary	expr - expr	Subtract (minus)
<<	Binary	expr << expr	Shift Left
>>	Binary	expr >> expr	Shift Right
<	Binary	expr < expr	Less than

Operator	Type	Usage	Description
<=	Binary	expr <= expr	Less than or equal
>	Binary	expr > expr	Greater than
>=	Binary	expr => expr	Greater than or equal
==	Binary	expr == expr	Equal
!=	Binary	expr != expr	Not Equal
&	Binary	expr & expr	Bitwise AND
^	Binary	expr ^ expr	Bitwise Exclusive OR
|	Binary	expr | expr	Bitwise Inclusive OR
&&	Binary	expr && expr	Logical AND
||	Binary	expr || expr	Logical Inclusive OR

Operands that the C/C++ operators act on

Operand	Definition
Constants (Floating or Integer)	Constants can be: hexadecimal numbers prefixed with ‘Ox’. Octal numbers prefixed with ‘O’, or unsigned numbers postfixed with a ‘U’. Characters, for example ‘A’, are not accepted.
Registers	The name of a valid register.
Symbols	Symbol names take into account their type. For example a variable defined as (char chr = ‘A’) would return ‘A’ when evaluated. To get the address of the object ‘&chr’ is required.

Operator limitations:

- Typecasts. Typecasts of basic type, such as int, float, unsigned int, int *, char *, are valid. Typecasts to user defined type such as, struct basic *, are not valid.
- Scope operator, '::'. The scope operator is valid as part of a class element name, for example, c_basic::print.
- Assignment operators, such as =, +=, *=, ++, --, are not implemented in this release.
- File/line number format is not implemented in this release.

Assembler expressions

The assembler expression evaluator is fully compatible with SNASM2.

Operator precedence:

Operator	Type	Usage	Description
( )	Primary	(expr)	Parenthesis Brackets
[ ]	Primary	[expr]	Address of
-	Unary	- expr	Negative expr
+	Unary	+ expr	Positive expr
~	Unary	~ expr	Bitwise NOT
<<	Binary	expr << expr	Shift left
>>	Binary	expr >> expr	Shift right
&	Binary	expr & expr	Logical AND
!	Unary	! expr	Logical NOT
|	Binary	| expr	Logical Inclusive OR
^	Binary	^ expr	Logical Exclusive OR
*	Binary	expr * expr	Multiply
/	Binary	expr / expr	Divide
%	Binary	expr % expr	Modulo (remainder)
+	Binary	expr + expr	Add (plus)
-	Binary	expr - expr	Subtract (minus)
=	Binary	expr = expr	Equals
<>	Binary	expr <> expr	Not Equals
<	Binary	expr < expr	Less Than

Operator	Type	Usage	Description
<=	Binary	expr <= expr	Less Than or Equals
>	Binary	expr > expr	Greater Than
>=	Binary	expr >= expr	Greater Than or Equals

Operands that the assembly operators act on

Operand	Definition		
Constants (Integer)	Constants can be defined in several operators to denote different radix:		
	Variable Hex Decimal Binary	X_<number>	where X is a single digit base prefix '$' or '0x' postfix 'h' prefix '#' postfix 'd' prefix '%' postfix 'b'
Registers	The name of a valid register.		
Symbols	Symbols are evaluated to labels, so a variable of type (char chr = 'A'), would return the address of (label to) the variable A when evaluated. Labels can be qualified by: '.b', '.w', '.l' for byte, word or long respectively [symbol]@b, [symbol]@w, [symbol]@l :<number> for the filename line number		

Expression evaluator dialog box (ALT+E)

The Expression Evaluator is a general purpose dialog box used for several operations, including: Edit Register, and Goto Address.

The options on the Expression Evaluator

Use the:	To:
Expression Combo box	Edit an existing expression, or select one from the history list.
Result box	View the results of an expression evaluation including any error messages.
ExpressionFormatradio buttons	Select C/C++, or Assembly as the expression format.
Default radix radio buttons	Select binary, octal, decimal, or hex, as the radix to use for the expression, or specify another radix in the Other text box. For C expressions this permits only control of the output radix.
Evaluate button	Evaluate an expression in the Expression Combo text box.
Symbol button	Use the Symbol Completion dialog box to search for a symbol from those available in the program file.
File button	View a list of all the files used to build the program file in the List Files in Program File dialog box. The dialog box also provides access to the address for "file:line number" information.
Lock check box	Lock the current expression to a file or symbol.

Symbol Completion dialog box (ALT+S)

Use the Symbol Completion dialog box to search for a symbol in the program file.

Options on the Symbol Completion dialog box

Use the:	To:
Search String text box	Enter the first few characters of the symbol to search for.
Only Search For Symbols Within Scope check box	Search for symbols in scope (select the check box), or to search for symbols in the whole program (deselect the check box).
Possible Completions text box	View a list of all symbols that match the current Search String.
Lookup button	Click Lookup to start another search.
OK button	Accept the current search string.
Cancel button	Ignore current search string.

Using the command-line

Use the command-line commands to specify how CodeScape will run. For example, CodeScape can run from another application such as the Codewright editor, or from a batch file.

Running CodeScape from the command-line

To run CodeScape from the command-line, type CodeScape then one or more optional switches. Always separate switches a space, but do not use spaces within the argument of a switch.

The command-line syntax is:

```
        codescape[Switch]...
```

Command-line switches

Run the CodeScape Help file and view information on using the command-line.

```
        [-|/]?
```

Run CodeScape without the splash screen appearing.

```
        [-|/]nologo
```

Crosslib Verbose mode. This switch displays additional information in the Log window as each command of the Cross Products Fileserver library (LIBCROSS) executes.

```
        [-|/]c
```

Use Project Info. This switch invokes CodeScape using the session file specified by SessionFile. The session file contains information on how to connect to targets. It includes: the object files in use for each target, update rates, breakpoints, watch expressions, log expressions, and window positions and displays. If no memory ranges are specified in the session file CodeScape will look for them in DEFAULT.SSN.

```
        [-|/]i=Session
```

Use this switch to specify which target (t#) and processor (p#) to use, and which program file to load. The processor is identified by its processor ID # (0-7) where 1=Master and 2=Slave. The program file is specified by ProgramFile. Using this switch to specify an object file for a target will override the setting in the session file.

Commands are downloading the binary from the object file (b), and suppressing debug information (n).

```
        [-|/]t#p#[b][n]:[ProgramFile]
```

NOTE: The n command does not require code or symbols. If you use the n command without the b command it will have no effect.

Files used by CodeScape

Filename	Description
SessionFile	This file contains the information needed to restore a previous debugging session.
ProgramFile	The object file. This contains binary and optionally, source level debug and symbol table information produced by the assembler or compiler.

To change file or folder properties:

1) Click the file or folder whose properties you want to change.
2) On the File menu, click Properties.

NOTE: You can drag a file's icon into a document, or even drag a shortcut icon.

Using the command-line, examples

Example: 1

To invoke CodeScape and restore a debugging session according to the information contained in a session file SESSION.SSN, type:

```
        codescape-i=session
```

Example: 2

This example invokes CodeScape and restores the debugging session. The binary (...b...) from the file TEST.COF is downloaded to target 7 (-t7...:test) but no symbolic information is loaded (...n...).

```
        codescape/t7p1bn:test
```

Example: 3

The n command does not require code or symbols. If you use the n command without the b command it will have no effect. The following example illustrates how not to invoke CodeScape:

```
        codescape-t1p1n:test
```

Appendix A: Frequent operations

Keyboard shortcuts are available for frequently used debugging operations. All operations are supported by Access keys which are shown on each menu item by an underlined letter.

There are Keyboard shortcuts and Access keys for options on:

- The Menu bar.
- Each region's shortcut menu.

Shortcut and Access keys on the Menu bar

To use the keyboard to access items that appear on the Menu bar:

- Press F10, select an item with the cursor keys, press ENTER.
  -OR-
- Press the menu’s keyboard shortcut, select an item with the cursor keys, press ENTER.

File menu ALT+F

Using the keyboard to access File menu commands

To:	Pressthiskeyboard shortcut:
Create a new session.	CTRL+SHIFT+N
Open an existing session.	CTRL+O
Close an open session.	none available
Save an open session.	CTRL+S
Save a session with a specific name.	none available
Reset the target with a soft reset.	none available
Reset the target with a hard reset.	none available
Load a program file.	CTRL+SHIFT+C
Restart a session.	CTRL+SHIFT+R
Save the binary part of a file.	none available
Load the binary part of a file.	none available
Print contents of the active region.	CTRL+P
Setup the printing commands.	none available
Exit CodeScape.	ALT+F4

Edit menu ALT+E

Using the keyboard to access Edit menu commands

To:	Pressthiskeyboard shortcut:
Undo the last operation.	CTRL+Z
Cut the current selection.	CTRL+X
Copy the current selection.	CTRL+C
Paste an item from the clipboard.	CTRL+V
Find a search item.	CTRL+F
Find the next search item.	F3
Replace a search item.	none available
Goto a specified address.	CTRL+G

View menu ALT+V

Using the keyboard to access View menu commands

To:	Pressthiskeyboard shortcut:
Select or clear a toolbar from the display list.	none available
Show the Status bar.	none available
To change the properties of the active region.	none available

Project menu ALT+P

Using the keyboard to access Project menu commands

To:	Pressthiskeyboard shortcut:
Setup the current project commands.	none available
Setup the editor commands.	none available
Build the current project.	CTRL+M
Edit path for a project's source files.	none available
Set the FileServer directory.	none available
ToggleFileserveroptimizationonor off.	ALT+P

Debug menu ALT+D

Using the keyboard to access Debug menu commands

To:	Pressthiskeyboard shortcut:
Access the processor execution commands.	none available
Run all processors simultaneously.	CTRL+F9
Stop all processors.	none available
Run a processor.	F9
Run a processor until it executes a specified address.	SHIFT+F9
Run a processor to the cursor position.	ALT+F9
Stop processor execution.	F9
Single step a line of code.	F7
Forced Step Into a line of code.	none available

To:	Pressthiskeyboard shortcut:
Step over a line of code.	F8
Undo a step.	CTRL+F7
Enable Animate Step Run.	none available
Step Run Into a line of code.	SHIFT+F7
Step Run Out of a line of code.	SHIFT+F8
Step Run a line of code.	ALT+F7
Step Run Until.	ALT+F8
Restart processor execution.	CTRL+SHIFT+R
Access breakpoint operations.	none available
Toggle a breakpoint.	F5
Enable a breakpoint.	none available
Disable a breakpoint.	none available
Configure breakpoint(s).	CTRL+F5
Reset all breakpoints.	ALT+F5
Enable all breakpoints.	CTRL+SHIFT+F5
Disable all breakpoints.	CTRL+ALT+F5
Remove all breakpoints.	SHIFT+F5
Set the cursor to the PC.	CTRL+SHIFT+P
Set the PC to the cursor.	CTRL+ALT+P
Goto Address...	CTRL+G

Region menu ALT+R

Using the keyboard to access Region menu commands

To:	Press this keyboard shortcut:
Access the region split commands.	none available
Split the active region to the left.	CTRL+SHIFT+LEFT ARROW
Split the active region to the right.	CTRL+SHIFT+RIGHT ARROW
Split the active region up.	CTRL+SHIFT+UP ARROW
Split the active region down.	CTRL+SHIFT+DOWN ARROW
Delete the active region.	CTRL+D
Access the region type list.	none available
Create a disassembly region.	ALT+1
Create a log region.	ALT+2
Create a local watch region.	ALT+3
Create a memory region.	ALT+4
Create a register region.	ALT+5
Create a source region.	ALT+6
Create a watch region.	ALT+7
Create an edit region.	ALT+8
Create a call stack region.	ALT+9
Update the display in all current regions.	CTRL+U

NOTE: To stop the display from updating in all regions press CTRL+SHIFT+U.

Window menu ALT+W

Using the keyboard to access Window menu commands

To:	Pressthiskeyboard shortcut:
Create a new window.	CTRL+N
Cascade all windows.	none available
Tile all windows.	none available
Arrangeallminimizedregionwindows at the bottom of the session window.	none available
Proportionally resize windows.	none available
Load the last loaded session file the next time CodeScape is run.	none available
Close all windows.	none available

Help menu ALT+H, or F1

Using the keyboard to access Help menu commands

To view:	Pressthiskeyboard shortcut:
The Help contents page.	F1
The Help topics list.	F1
CodeScape version information and copyright.	none available

Shortcut and Access keys on the shortcut menus

Press TAB to move to and activate a region, then:

- Press SHIFT+F10, select an item with the cursor keys, then press ENTER.

Source region

Using the keyboard commands in the Source region

To:	Press this keyboard shortcut:
Synchronize the cursor.	none available
Show the corresponding address for the first line of code generated by the source code.	CTRL+A
Show line numbers for each line of code.	CTRL+L
Access the processor execution commands.	none available
Run all processors.	CTRL+F9
Stop all processors from running.	none available
Run a processor.	F9
Run a to a specific address.	SHIFT+F9
Run a processor to the current cursor position.	ALT+F9
Stop processor execution.	F9
Single step a line of code.	F7
Forced step over a line of code.	SHIFT+F7
Step over a line of code.	F8
Undo a step.	CTRL+F7
Step Out of a line of code.	CTRL+F8
Enable Animated Step Run.	none available
Step Run In to a line of code.	SHIFT+F7

To:	Press this keyboard shortcut:
Step Run Out of a line of code.	SHIFT+F8
Step Run a line of code.	ALT+F7
Step Run Until.	ALT+F8
Restart processor execution.	CTRL+SHIFT+R
Access breakpoint operations.	none available
Toggle a breakpoint.	F5
Disable a breakpoint.	none available
Configure breakpoint(s).	CTRL+F5
Reset all breakpoints.	ALT+F5
Enable all breakpoints.	CTRL+SHIFT+F5
Disable all breakpoints.	CTRL+ALT+F5
Remove all breakpoints.	SHIFT+F5
Set the cursor to the PC.	CTRL+SHIFT+P
Set the PC to the cursor.	CTRL+ALT+P
Go to a specified address.	CTRL+G
Go to a specific source file.	CTRL+SHIFT+F5
Access the region’s tools.	none available
Change the properties of the active region.	none available
Set the color and font for the active region.	none available
Set the tab width for the active region.	CTRL+T
Set the region update rate for the active region.	none available

Disassembly region

Using the keyboard commands in the Disassembly region

To:	Pressthiskeyboard shortcut:
Synchronize the cursor.	none available
Show the location address of the disassembled code.	CTRL+A
Show the symbolic label replacement of the disassembled code.	CTRL+B
Show the opcode words for the disassembled region.	CTRL+W
Show the operand values in hexadecimal.	CTRL+H
Toggle the display between uppercase and lowercase.	CTRL+L
Show the symbols for each line of code.	CTRL+Y
Show the effective address and literals.	CTRL+I
Access the processor execution commands.	none available
Run all processors.	CTRL+F9
Stop all processors.	none available
Run a processor.	F9
Run a processor to a specified address.	SHIFT+F9
Run a processor to the current cursor position.	ALT+F9
Stop processor execution.	F9
Single step a line of code.	F7
Force step into a line of code.	none available
Step over a line of code.	F8
Undo a step.	CTRL+F7
Enable Animated Step Run.	none available
Step Run In to a line of code.	SHIFT+F7

To:	Pressthiskeyboard shortcut:
Step Run Out of a line of code.	SHIFT+F8
Step Run a line of code.	ALT+F7
Step Run Until.	ALT+F8
Restart processor execution.	CTRL+SHIFT+R
Access breakpoint operations.	none available
Toggle a breakpoint.	F5
Enable a breakpoint.	none available
Disable a breakpoint.	none available
Configure breakpoint(s)	CTRL+F5
Reset all breakpoints.	ALT+F5
Enable all breakpoints.	CTRL+SHIFT+F5
Disable all breakpoints.	CTRL+ALT+F5
Remove all breakpoints.	SHIFT+F5
Set the cursor position to the PC.	CTRL+SHIFT+P
Set the PC to the cursor position.	CTRL+ALT+G
Go to a specified address.	CTRL+P
Search in the active region	CTRL+F

Call Stack region

Using the Keyboard commands in the Call Stack region

To:	Pressthiskeyboard shortcut:
Show Parameter Name	none available
Show Parameter Types	none available
Show Parameter Values	none available
Show Parameter Registers	none available
Cycle through the available bases. (Octal, Decimal, and Hexadecimal.)	CTRL+H
Access the processor execution commands.	none available
Run all processors.	CTRL+F9
Stop all processors.	none available
Run a processor.	F9
Run a processor to a specified address.	SHIFT+F9
Run a processor to the current cursor position.	ALT+F9
Stop processor execution.	F9
Single step a line of code.	F7
Force step into a line of code.	none available
Step over a line of code.	F8
Undo a step.	CTRL+F7
Enable Animated Step Run.	none available
Step Run In to a line of code.	SHIFT+F7
Step Run Out of a line of code.	SHIFT+F8
Step Run a line of code.	ALT+F7

To:	Pressthiskeyboard shortcut:
Step Run Until.	ALT+F8
Restart processor execution.	CTRL+SHIFT+R
Access breakpoint operations.	none available
Toggle a breakpoint.	F5
Enable a breakpoint.	none available
Disable a breakpoint.	none available
Configure breakpoint(s)	CTRL+F5
Reset all breakpoints.	ALT+F5
Enable all breakpoints.	CTRL+SHIFT+F5
Disable all breakpoints.	CTRL+ALT+F5
Remove all breakpoints.	SHIFT+F5

Watch region

Using the keyboard commands in the Watch region

To:	Press this keyboard shortcut:
Cut the selection and paste it to the clipboard.	CTRL+X
Copy the selection and paste it to the clipboard.	CTRL+C
Paste the contents of the clipboard at the cursor.	CTRL+V
Delete part of a structure.	DELETE
Expand/collapse a structure or array.	SPACE
Insert a new watch expression.	CTRL+I
Insert a variable at the end of the current list.	CTRL+A
Toggle the display through the available bases (Show Octal, Show Decimal, Show Hexadecimal).	CTRL+H
Modify the value of a variable or watch expression.	CTRL+ALT+E
Search for a symbol or a value.	CTRL+G,ALT+S
Access the processor execution commands.	none available
Run all processors.	CTRL+F9
Stop all processors from running.	none available
Run a processor.	F9
Run a to a specific address.	SHIFT+F9
Stop processor execution.	F9
Single step a line of code.	F7
Forced step over a line of code.	SHIFT+F7
Step over a line of code.	F8
Undo a step.	CTRL+F7
Step Out of a line of code.	CTRL+F8

To:	Press this keyboard shortcut:
Enable Animated Step Run.	none available
Step Run In to a line of code.	SHIFT+F7
Step Run Out of a line of code.	SHIFT+F8
Step Run a line of code.	ALT+F7
Step Run Until.	ALT+F8
Restart processor execution.	CTRL+SHIFT+R
Access breakpoint operations.	none available
Toggle a breakpoint.	F5
Disable a breakpoint.	none available
Configure breakpoint(s).	CTRL+F5
Reset all breakpoints.	ALT+F5
Enable all breakpoints.	CTRL+SHIFT+F5
Disable all breakpoints.	CTRL+ALT+F5
Remove all breakpoints.	SHIFT+F5
Search for a symbol in the Expression Evaluator.	ALT+S

Local Watch region

Using the keyboard commands in the Local Watch region

To:	Press this keyboard shortcut:
Copy the selection and paste it to the clipboard.	CTRL+C
Delete an expression or part of an expression structure.	DELETE
Expand / collapse a structure or array.	SPACE
Toggle the display through the available bases (Show Octal, Show Decimal, Show Hexadecimal).	CTRL+H
Modify the value of a variable or watch expression.	CTRL+ALT+E
Access the processor execution commands.	none available
Run all processors.	CTRL+F9
Stop all processors.	none available
Run a processor.	F9
Run a processor to a specified address.	SHIFT+F9
Stop processor execution.	F9
Restart processor execution.	CTRL+SHIFT+R
Access breakpoint operations.	none available
Configure breakpoint(s)	CTRL+F5
Reset all breakpoints.	ALT+F5
Enable all breakpoints.	CTRL+SHIFT+F5
Disable all breakpoints.	CTRL+ALT+F5
Remove all breakpoints.	SHIFT+F5
Highlight changes.	none available
Change the properties of the active region.	none available

To:	Press this keyboard shortcut:
Search for a symbol in the Expression Evaluator.	ALT+S

Memory region

Using the keyboard commands in the Memory region

To:	Pressthiskeyboard shortcut:
Display the ASCII value for each byte of memory.	CTRL+A
Display memory as bytes.	CTRL+B
Display memory as words.	CTRL+W
Display memory as longs.	CTRL+L
Display memory as quadwords.	CTRL+Q
See where the target's memory changed.	none available
Toggle write protect.	CTRL+SHIFT+W
Set bytes per line.	CTRL+ SHIFT+L
Change a value in the Memory region.	CTRL+ALT+E
Access the processor execution commands.	none available
Run all processors.	CTRL+F9
Stop all processors.	none available
Run a processor.	F9
Run a processor to a specified address.	SHIFT+F9
Stop processor execution.	F9
Single step a line of code.	F7
Force step into a line of code.	none available
Step over a line of code.	F8
Undo a step.	CTRL+F7
Enable Animated Step Run.	none available
Step Run In to a line of code.	SHIFT+F7

To:	Pressthiskeyboard shortcut:
Step Run Out of a line of code.	SHIFT+F8
Step Run a line of code.	ALT+F7
Step Run Until.	ALT+F8
Restart processor execution.	CTRL+SHIFT+R
Access breakpoint operations.	none available
Toggle a breakpoint.	F5
Enable a breakpoint.	none available
Disable a breakpoint.	none available
Configure breakpoint(s)	CTRL+F5
Reset all breakpoints.	ALT+F5
Enable all breakpoints.	CTRL+SHIFT+F5
Disable all breakpoints.	CTRL+ALT+F5
Remove all breakpoints.	SHIFT+F5
Follow a pointer in memory.	CTRL+F
Goto a specific address in memory.	CTRL+G
Toggle write protect.	CTRL+SHIFT+W
Access the region’s tools.	none available
Search for a pattern in memory.	CTRL+F
Continue a search.	F3
Fill a range of memory with specific data	none available
Write a block of memory in hexadecimal to a file.	none available
Change the properties of the active region.	none available

Register region

Using the keyboard commands in the Register region

To:	Pressthiskeyboard shortcut:
Apply the current Increment Value (1 is the default) to the contents of the register.	+
Apply the current Decrement Value (1 is the default) to the contents of the register.	-
Change the Increment/Decrement Value.	none available
See where changes occurred during the last operation.	none available
Prevent data from being written to the currently active Register region.	CTRL+SHIFT+W
Change the selected register value.	CTRL+ALT+E
Access the column format commands.	none available
Display registers in two columns.	CTRL+2
Display registers in four columns.	CTRL+4
Tell CodeScape to set the column format.	CTRL+0
Show banked registers.	CTRL+B
Show float registers.	CTRL+L
Access the processor execution commands.	none available
Run all processors.	CTRL+F9
Stop all processors.	none available
Run a processor.	F9
Run a processor to a specified address.	SHIFT+F9
Run a processor to the current cursor position.	ALT+F9
Stop processor execution.	F9
Single step a line of code.	F7

To:	Pressthiskeyboard shortcut:
Force step into a line of code.	none available
Step over a line of code.	F8
Undo a step.	CTRL+F7
Enable Animated Step Run.	none available
Step Run In to a line of code.	SHIFT+F7
Step Run Out of a line of code.	SHIFT+F8
Step Run a line of code.	ALT+F7
Step Run Until.	ALT+F8
Restart processor execution.	CTRL+SHIFT+R
Access breakpoint operations.	none available
Toggle a breakpoint.	F5
Enable a breakpoint.	none available
Disable a breakpoint.	none available
Configure breakpoint(s)	CTRL+F5
Reset all breakpoints.	ALT+F5
Enable all breakpoints.	CTRL+SHIFT+F5
Disable all breakpoints.	CTRL+ALT+F5
Remove all breakpoints.	SHIFT+F5
Access the region’s tools.	none available
Save the current registers block.	none available
Restore the current registers block.	none available
Change the properties of the active region.	none available

Log region

Using the keyboard commands in the Log region

To:	Pressthiskeyboard shortcut:
Configure the current log window.	none available
Print the contents of the current log window.	CTRL+P
Save the contents of the current log window to file.	CTRL+S
Access the processor execution operations.	none available
Run all processors.	CTRL+F9
Stop all processors.	none available
Run the selected processor.	F9
Run the selected processor to a specific address.	SHIFT+F9
Stop the selected processor.	F9
Restart the selected processor.	CTRL+SHIFT+R
Access breakpoint operations.	none available
Configure breakpoint(s)	CTRL+F5
Reset all breakpoints.	ALT+F5
Enable all breakpoints.	CTRL+SHIFT+F5
Disable all breakpoints.	CTRL+ALT+F5
Remove all breakpoints.	SHIFT+F5
Clear the contents of the log window.	none available
Change the properties of the active region.	none available

Edit region

Using the keyboard commands in the Edit region

To:	Pressthiskeyboard shortcut:
Create a new Editor file.	none available
Open an existing Editor file.	none available
Save the current Editor file.	none available
Save the current Editor file with a specific name.	none available
Cut the current selection in the Editor file and paste it to the clipboard.	CTRL+X
Copy the current selection in the Editor file and paste it to the clipboard.	CTRL+C
Insert the contents of the clipboard at the current cursor position.	CTRL+V
Enter a new tab value.	none available
Undo the action.	none available
Search for a string.	none available
Replace the current selection.	none available
Change the line number of the origin address.	none available
Change the properties of the active region.	none available