

# SANKHYA Debugger™

System Level Debugger

**User Guide and Reference Manual**

**THIS DOCUMENT CONTAINS PROPRIETARY INFORMATION OF SANKHYA TECHNOLOGIES PRIVATE LIMITED. Use, duplication and disclosure are subject to license restrictions.**

**(C) Copyright 2004 Sankhya Technologies Private Limited**

**All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means viz., electronic, mechanical, photo-copying, recording, or otherwise, without the prior consent of the publisher.**

**SANKHYA, SANKHYA TECHNOLOGIES, SANKHYA Debugger, Dynamically Targetable Tools Framework, SANKHYA Tools Collection, SANKHYA Varadhi, SANKHYA Software are Trademarks, Service Marks or Registered Trademarks of Sankhya Technologies Private Limited. All other brands and names are the property of their respective owners.**

Part No. 10003163-001

---

---

# SANKHYA Debugger

**User Guide and Reference Manual**

**Sankhya Technologies Private Limited**

**Part No. 10003163-001**

---

**Table 1: Revision History**

Revision number	Revision History	Date
001	SD 1.0 Pre-Beta Release	05 Feb 2004

# Contents

## Preface

<b>Part 1 - User Guide .....</b>	<b>1</b>
<b>1.1 Introduction .....</b>	<b>1</b>
1.1.1 Overview.....	1
1.1.2 Hosts/Targets supported.....	2
1.1.3 Features .....	2
<b>1.2 Invoking SANKHYA Debugger .....</b>	<b>3</b>
1.2.1 Setting up SD host development environment.....	3
<b>1.3 Debugging using SD .....</b>	<b>5</b>
1.3.1 Overview of Debugging Process .....	5
1.3.2 Usage Example of SANKHYA Debugger.....	5
1.3.3 Debugging using customized System and CPU.....	8
<b>1.4 Debugging using Simserver .....</b>	<b>14</b>
1.4.1 Overview of Debugging Process .....	14
1.4.2 A SimServer Client sample.....	14

---

<b>Part 2 - Reference Manual .....</b>	<b>18</b>
<b>2.1 SANKHYA Debugger .....</b>	<b>18</b>
2.1.1 Introduction.....	18
2.1.2 Synopsis .....	18
2.1.3 GUI Features.....	19
2.1.3.1 Main Window .....	19
2.1.3.1.1 Connection Management .....	19
2.1.3.1.2 System Management.....	22
2.1.3.1.3 System Configuration .....	23
2.1.3.1.4 Debug.....	26
2.1.3.1.5 Breakpoint Management.....	29
2.1.3.1.6 Windows Management .....	32
<b>2.2 SANKHYA Debugger Commands .....</b>	<b>41</b>

## Preface

**Sankhya Debugger (SD)** is a system level debugger for debugging embedded software applications running on simulated and real target systems.

This document contains 2 parts:

Part-1 'User Guide' provides the user with the description on how SANKHYA Debugger can be used to debug a embedded application for MIPS32 and ARMv4 targets.

Part-2 'Reference Manual' provides detailed description of the SANKHYA Debugger GUI features, functions and debugger commands with examples.

### **Audience**

This document intends to provide the users of SANKHYA Debugger an in depth coverage of SANKHYA Debugger tool. It explains the steps to debug an application using SANKHYA Debugger for MIPS and ARM.

The user is expected to have basic knowledge of microprocessor systems with good understanding of the target processor architecture and instruction set.

### **Notational Conventions**

The guide follows the following conventions

- % - The 'percentage' sign denotes a Unix environment.
- > - The 'greater than' symbol represents a DOS/Windows environment.
- Italic* - The words given in italics represents an option.

- `code` - The guide differentiates the normal text from a program code through this color. Any code, part of a code, input, output and command line statements in this document, will be represented using this color.
- `->` - Indicates the sequence of menu commands to perform an operation.
- `...` - Indicates that some portion of the material has been removed to simplify the description.
- `[ ]` - Indicates an optional argument that can be used in the command line.

---

# USER GUIDE

**Sankhya Technologies Private Limited**

---

---

# *Part 1 - User Guide*

---

## **1.1 Introduction**

**Sankhya Debugger (SD)** is a system level debugger, for debugging embedded software applications running on simulated and real target systems. SD can be used to describe, simulate and debug the target systems in an user friendly environment. SANKHYA Debugger provides embedded system developers a GUI framework that supports system level debugging features such as system simulation and switching between multiple CPUs/Systems.

### **1.1.1 Overview**

Sankhya Debugger includes the following tools for MIPS (MIPS32-ISA) target and ARM (ARMV4) target.

- `sdmips` - SANKHYA Debugger for MIPS.
- `simsvermips` - Simulator Server for MIPS.
  
- `sdarm` - SANKHYA Debugger for ARM.
- `simsverarm` - Simulator Server for ARM.

The User Interface of Sankhya Debugger includes the main window, register window and other GUI components to display the state of the system and the selected CPU in which debugging is performed.

SANKHYA Debugger communicates with the target system, which could be a simulator or a real target, through a server agent. Communication between SANKHYA Debugger and the server agent is established using CORBA paradigm.

SANKHYA Debugger is available with simulator servers for MIPS32 and ARMv4 targets. Simserver is a CORBA Server which is basically a wrapper server for SANKHYA Simulator libraries that acts as the target server agent for the debugger. Simserver can be easily integrated into RTL verification environments through a standard CORBA based interface specified in simserver.idl. Please refer “Part II - Reference Manual” for more information on the usage of simserver.

Real Target debugging support can be added by using appropriate Background Debug Mode (BDM) servers.

### **1.1.2 Hosts/Targets supported**

The following hosts are supported by SANKHYA Debugger.

- Windows NT 4.0

The following targets are supported by SANKHYA Debugger.

- MIPS32
- ARMv4

### **1.1.3 Features**

The following are the features of SANKHYA Debugger.

- Basic Assembly level debugging
- System Level debugging

- Remote debugging
- Compatible with GCC and STC compiler tools
- Supports command file processing
- Supports standard debugger commands
- Separate standard debugger windows
  - Code window
  - Register window
  - Memory window
  - Breakpoint window

## 1.2 Invoking SANKHYA Debugger

### 1.2.1 Setting up SD host development environment

To work with the SANKHYA Debugger, the following environment variables should be set.

SD\_HOME - Directory path pointing to the root of installation

In order to set these variables and PATH variable on Windows, run sd.cmd. In a Windows Command Prompt, type

```
> <SD_HOME>\sd.cmd
```

where,

<SD\_HOME> - Directory where Sankhya Debugger is installed.

By default, Sankhya Debugger is installed in C:\SANKHYA\SD.

SANKHYA Debugger can be invoked from the command line as follows.

For ARM,

```
> sdarm
```

For MIPS,

```
> sdmips
```

When sdmips is invoked, SANKHYA Debugger for MIPS will be opened as shown in figure 1.1

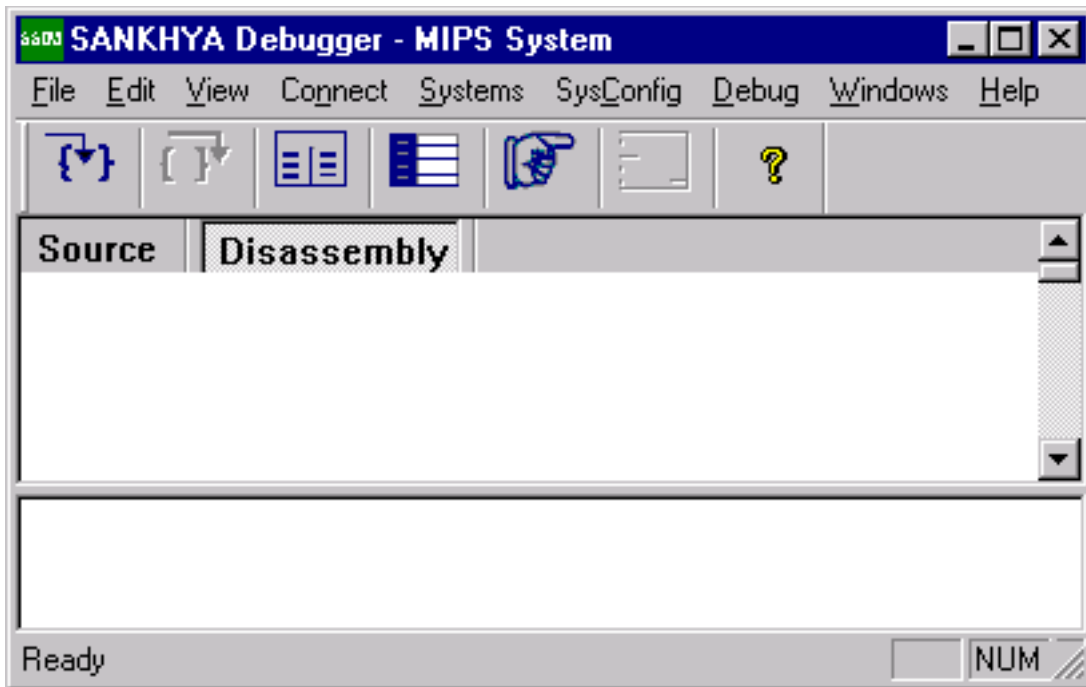


Figure 1.1 SDMIPS Main Window

On invoking SANKHYA Debugger for MIPS, sdmips connects to a single CPU

based system by default, with a memory block specified in the configuration file ‘mips\_sys.ini’. Similarly SANKHYA Debugger for ARM uses the configuration file ‘arm\_sys.ini’.

## 1.3 Debugging using SD

### 1.3.1 Overview of Debugging Process

Here is the complete overview of the steps to be followed for debugging an application using SANKHYA Debugger.

Step-1. Build the application.

Step-2. Create command files.

Step-3. Debug the application

- a. Connect to target system and configure the system
- b. Load the application
- c. Debug the loaded application

The following section explain each step specified in the overview in detail for a simple ARM sample.

### 1.3.2 Usage Example of SANKHYA Debugger

Sankhya Debugger samples for the target ‘tgt’ (arm/mips) can be found under the directory %SD\_HOME%\samples\sd\{tgt}. The ‘sort’ sample performs sorting of values in descending order. The above sample is generated using GCC tools and is available in gcc directory.

**Step-1:** Build the application.

The ‘**sort**’ sample explained below uses the application `sort.x` generated by the `gcc` compiler. Please refer the Compiler manual for more information on building the sample using GNU C Compiler.

**Step-2:** Create command files as below.

Contents of `sort1.cmd`:

```
memset 0xb9e0 0x1000
b 0x8150
b 0x9f18
run
```

Contents of `sort2.cmd`:

```
run
```

**Step-3:** Debug the application.

Following are the steps on how to use the ARM debugger for debugging the ‘`sort`’ application.

**Step-3a:** After setting up the SD environment, invoke the Debugger for ARM.

```
> sdarm
```

**Step-3b:** Configure the CPU memory region for the ‘`sort`’ application. Click “SysConfig -> CreateMemoryRegion”, a dialog box will pop up as in figure 1.6.

Fill the memory region details as follows in the “CreateMemoryregion” dialog box.

CPUName : Select ‘ARMv4-1’ from the combo box  
MemoryBlock : Select ‘m’ from the combo box  
StartAddress : Type 0x0  
Attributes : Choose “rw” from the combo box

**Step-3c:** Load the executable sort.x. Click “Debug -> LoadFileToTarget” or Press “Ctrl+L”. Select ‘sort.x’ in the opened file dialog box.

**Step-3d:** Include the command file sort1.cmd. Click “Debug -> IncludeCommands”. Select ‘sort1.cmd’ in the file dialog box.

**Step-3e:** Open the Memory window by clicking “Windows -> Memory” or using the hotkey “Alt+F1” .

In Memory window:

Type 0xc9d0 in “Start:” and click ”Update”. Now the memory window displays the unsorted input values in ascending order.

**Step-3f:** Include the command file sort2.cmd.

Click “Debug -> IncludeCommands”. Select ‘sort2.cmd’ in the file dialog box.

In Memory window:

Type 0xc9d0 in “Start” and click “Update”. Now the memory window will display the sorted values in descending order.

Note: Please refer samples directory for information on how to run a sample using Sankhya debugger for MIPS. In the above sample, debugging is performed using default System and CPU.

### 1.3.3 Debugging using customized System and CPU

Sankhya Debugger provides support to debug in the following modes.

- Single System, Single CPU
- Single System, Multiple CPU

Multiple CPUs can be created in a single System and different applications can be loaded in the respective CPUs and debugging can be performed. By default, sankhya debugger will simulate a single CPU based system.

The following are the steps to create a system and CPU and debug the application in SANKHYA Debugger.

**Step1:** Create a System for the application.

Invoke “Connect->CreateSystem” A dialog box titled “SystemDialog” will be opened as in figure 1.2.

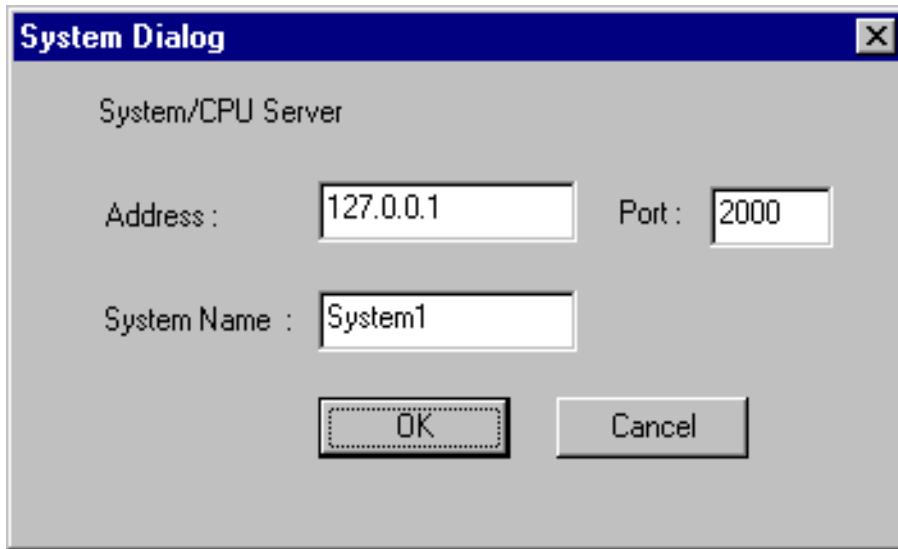


Figure 1.2 Create System Dialog

Specify the following details in the dialog box opened and click ‘OK’

Address : IP address of the host in which the simulator server or BDM server (CORBA) is running.

Port : Port in which the Server is listening.

System Name : Name of the system.

By default, Sankhya debugger will start and connect to the simulator server in the port ‘2000’.

Simulator server can be started in the user specified port also as below.

For MIPS,

```
> simservermips --VaradhiPORT 2004
```

For ARM,

```
> simserverarm --VaradhiPORT 2004
```

“Connect->CreateSystem” command can use the above specified port number for creating a system and debugging the application in the created system.

**Step2:** Connect to the created System by invoking “Connect->ConnectSystem”. A dialog box titled “Connect System” will pop up as in Figure 1.3.

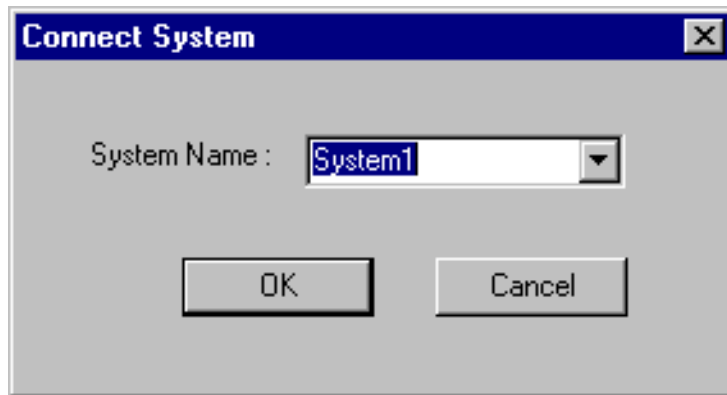


Figure 1.3 Connect System

Choose the system created(System1) in Step 1 and click ‘OK’. Now the ‘Systems’ Menu in the main window will display the connected system “System1”.

**Step3:** Create and set the CPU for the connected system.

Invoke “SysConfig->CreateCPU” . A dialog box titled “Create CPU...” will pop up as shown below.

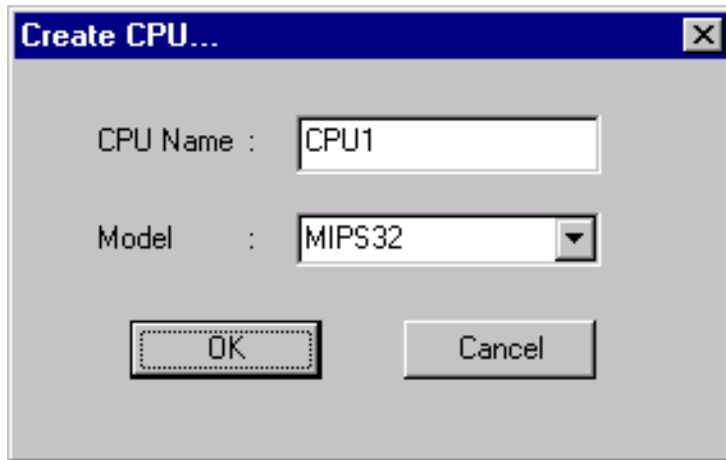


Figure 1.4 CreateCPU

Specify the following details in the dialog box and click ‘OK’.

CPU Name : Name reference of the CPU.

Model : Processor model to be used.

Set the name of the CPU ‘CPU1’ to be used for debugging by invoking the menu item “System->SetCPU->CPU1”.

**Step4:** Create a Memory Block for the System.

Invoke “SysConfig->CreateMemoryBlock”

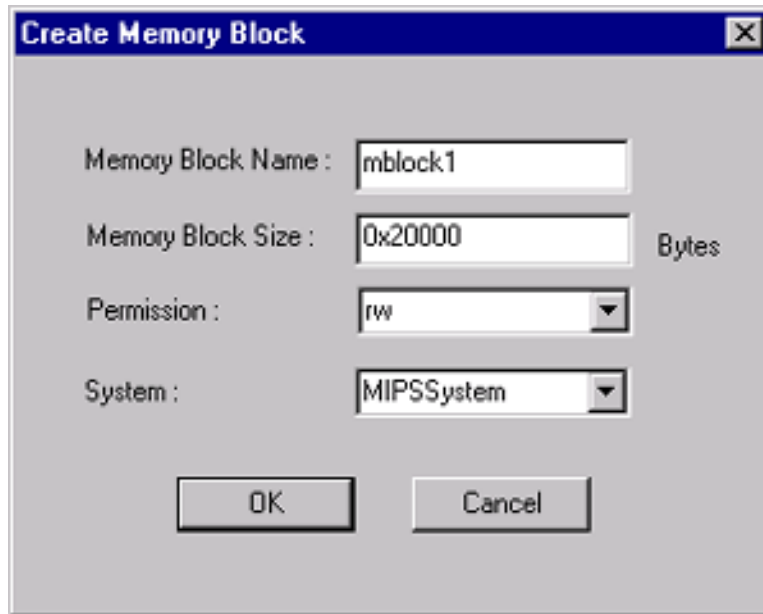


Figure 1.5 Create Memory Block

Specify the following details in the memory block window.

- Memory Block Name - Name of the memory block
- Memory Block Size - Size of the memory block
- Permissions - Permissions for the memory block.
- System - Name of the System.

**Step5:** Create or map memory region for the CPU.

Click “SysConfig->CreateMemoryRegion” .

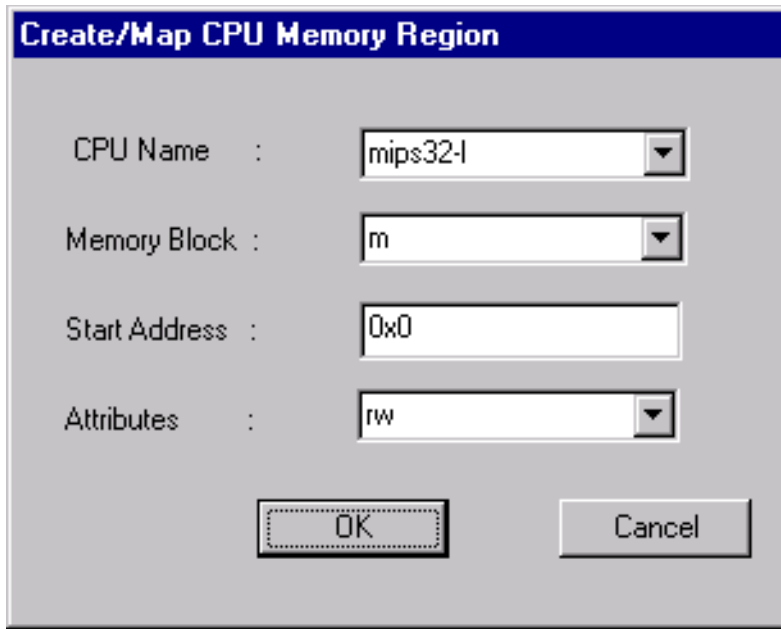


Figure 1.6 CreateMemoryRegion

Specify the following details in the dialog box.

- CPU Name - Name reference of the CPU created in Step 3
- Memory Block - Name of the memory block created in Step 4
- Start Address - Start Address of the application to be debugged
- Attributes - Specify the attributes for the memory region

**Step6:** Load the application by clicking “Debug->LoadFileToTarget”.

**Step7:** Invoke “Debug->IncludeCommands” and include the command file in the file dialog box.

Similarly multiple Systems can be created and multiple CPUs can be created for each system and debugging can be performed.

## 1.4 Debugging using Simserver

### 1.4.1 Overview of Debugging Process

Here is the complete overview of the steps to be followed for debugging an application using SANKHYA Debugger.

Step-1. Build the application.

Step-2. Create command files.

Step-3. Build the simserver client

Step-4. Debug the application using simserver client

The following section explain each step specified in the overview in detail for a simple 'debug\_cpu' sample that debugs the swap application using MIPS simulator server.

### 1.4.2 A SimServer Client sample

Simulator server samples for the target 'tgt' (arm/mips) are available under %SD\_HOME%\samples\simserver\{tgt}. Simulator server is basically a wrapper server for the Sankhya simulator libraries build using SANKHYA Varadhi, an ORB(Object Request Broker). Any CORBA compliant client can use the interface methods of simulator server to perform various simulator actions.

SANKHYA Varadhi, an ORB (Object Request Broker) is used for building the simserver client. Please refer "Sankhya Varadhi User Guide and Reference Manual (<http://sankhya.com/info/products/varadhi/docs.html>) for more information on SANKHYA Varadhi.

**Step-1:** Build the application.

The ‘**swap**’ sample explained below uses the application swap.x generated by the STC compiler. Please refer the Compiler manual for more information on building the sample using STC Compiler tools for MIPS.

**Step-2.** Create command files as below.

Contents of swap.cmd:

```
set r29 0x410000
set r31 0xffffffff
memset 0x412000 10
memset 0x412004 20
set r4 0x412000
set r5 0x412004
```

Contents of run.cmd:

```
run 0xffffffff
```

**Step-3:** Build the Simserver Client using the following steps.

**Step-3a.** Set the Microsoft VC++ environment.

```
>vcvars32.bat
```

**Step-3b.** Set the Varadhi host development environment.

```
>%VARADHI%\varadhi.cmd
```

where,

%VARADHI% - Varadhi root installation directory.

**Step-3c.** Set the SD host development environment

```
>%SD_HOME%\sd.cmd
```

where,

%SD\_HOME% - SANKHYA Debugger root installation directory.

**Step-3d.** Create and set the Varadhi platform environment as explained below. Here `simserver.cfg` (simserver platform configuration file) contains the corbaloc format Object URL.

```
>cd %SD_HOME%\samples\simserver
```

```
>vconf defaults se win32 simserver.cfg
```

```
>varadhi\platform.bat
```

**Step-3e.** Move to the sample directory and build the simserver client

```
>cd %SD_HOME%\samples\simserver\mips\debug_cpu
```

```
>nmake /f makefile.nt
```

Note:

For more information on the client sample, please refer `client.cc` in the sample directory.

**Step-4.** Invoke the Simulator Server and simserver client for MIPS32 as below.

```
>start simservermips
```

```
>client 127.0.0.1:2000
```

On invoking the client, the output will be displayed as follows.

```
The System "System1" is created...
System1 is connected...
CPU1 is created in System1 ...
Memory Block (m) of size 0x20000 is created in System1 ...
Memory Region for CPU1 is configured with start address 0x400030 ...
CPU1 is set as current CPU of the System System1...
Contents of 0x412000, 0x412004 before swap : 0000000a;00000014
Loading swap.x to CPU1...
Running swap application...
Contents of 0x412000, 0x412004 after swap : 00000014;0000000a
System1 is disconnected...
```

**Note:**

Please refer samples directory for information on how to run a sample using Simulator server for ARM.

---

# REFERENCE MANUAL

**Sankhya Technologies Private Limited**

---

---

## *Part 2 - Reference Manual*

---

### **2.1 SANKHYA Debugger**

#### **2.1.1 Introduction**

Sankhya Debugger (SD) is a system level debugger for debugging embedded software applications running on simulated and real target systems. SD can be used to simulate, configure and debug the target systems in an user friendly environment.

#### **2.1.2 Synopsis**

For MIPS,

```
> sdmips [-simport <port no>] [-V] [-h]
```

For ARM,

```
> sdarm [-simport <port no>] [-V] [-h]
```

The following are the options and its functionality supported by Sankhya Debugger.

- V - displays version information
- h - displays help
- simport <port no> - specifies the port no at which the simserver will be started.  
By default, sdmips starts the simserver in the port '2000'.

## 2.1.3 GUI Features

### 2.1.3.1 Main Window

The SD main window supports menu, menu items and hotkeys to provide a user friendly environment for performing system simulation/configuration and debugging. The SD main window is split into two panes. The upper pane is the Code Window and the lower pane is the Command Window.

SD window menus provides menu items which supports various debugger operation. Code window consists of separate tabs for viewing source code and assembly level instructions of the loaded application. Please refer section 1.2.1 for the screenshot of Sankhya debugger for MIPS.

#### 2.1.3.1.1 Connection Management

‘**Connect**’ Menu provides menu items to create, connect and disconnect a system.

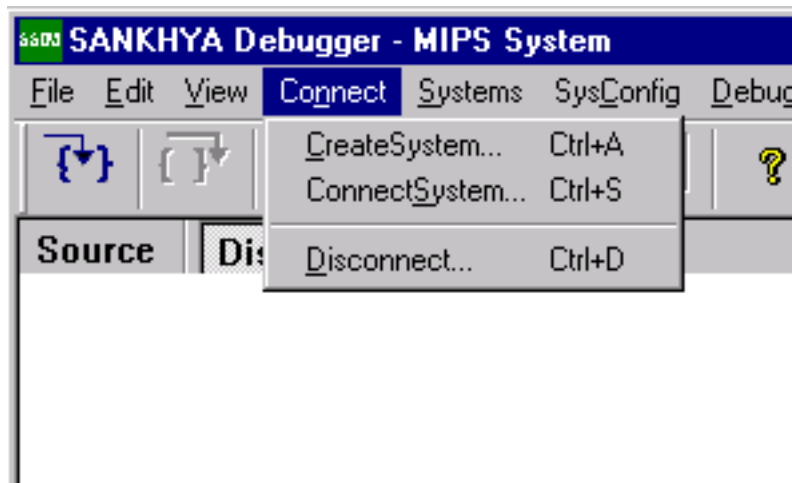


Figure 2.1 Connect Menu

The menu item “**CreateSystem**” can be used to create a system for debugging. This can be done using the hotkey “Ctrl+A”.

A dialog box titled “SystemDialog” will pop up as shown below.

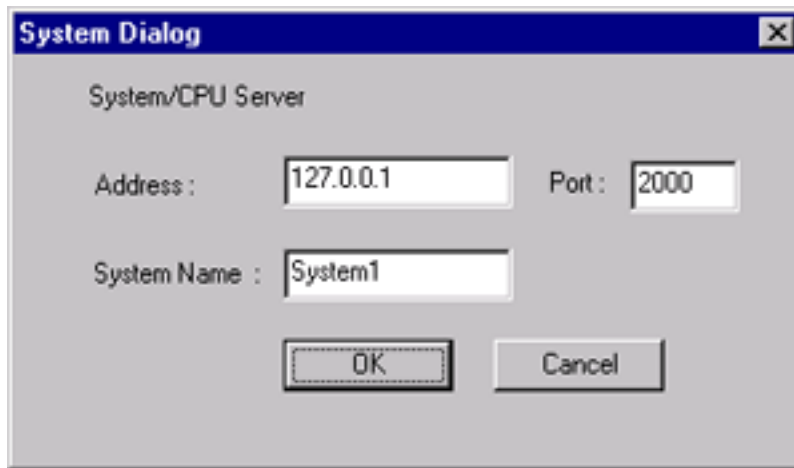


Figure 2.2 CreateSystem

Specify the following details in the dialog box and click ‘OK’.

Address : IP address of the host in which the simulator server or BDM Server is running. Default value is ‘127.0.0.1’

Port : Port in which the Server is listening. Default value is ‘2000’.

System Name : Name of the system. Default value is ‘System1’.

A system with the specified name will be created with the specified values.

The menu item “**ConnectSystem**” allows the user to connect to the created system. “Connect System” dialog box will be opened by using the hotkey “Ctrl+S” or by clicking “Connect->ConnectSystem” as shown below.

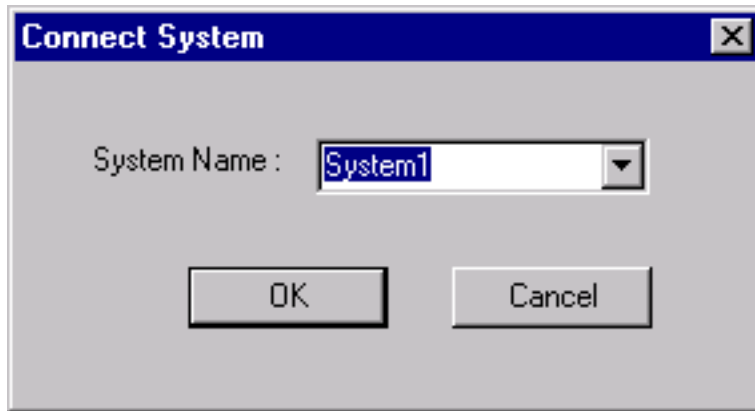


Figure 2.3 ConnectSystem

Choose the system to be connected and click ‘OK’. By default the combo box “System Name” will display the system created recently.

Now the ‘**Systems**’ menu in the Main Window will display the connected system “System1”.

“**DisConnect**” menu item allows the user to disconnect a connected system. Invoke “Connect->Disconnect” or use the hotkey “Ctrl+D”. A dialog box titled “Disconnect System” will pop up as shown in Figure 2.4 .

By default the system specified in the configuration file ( ‘MIPSSystem’ and ‘ARMSystem’) will be displayed in the combo box .

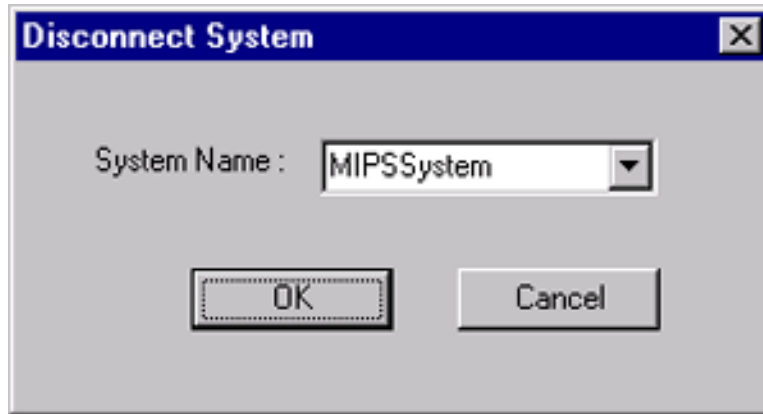


Figure 2.4 Disconnect System

The configuration files ‘mips\_sys.ini’ and ‘arm\_sys.ini’ available in %SD\_HOME%\etc are the files basically used by SANKHYA Debugger for configuring the systems in the simulator server.

The above configuration files supports the following actions.

- Creating a system in the simulator.
- Connecting to the system.
- Configuring the connected system.

#### 2.1.3.1.2 System Management

‘**Systems**’ Menu will list the systems connected to the debugger. By checking the toggle menu item, user can choose the current system if multiple systems are connected.

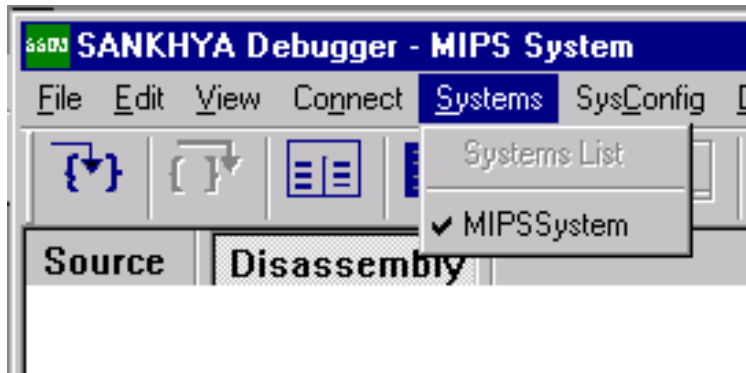


Figure 2.5 Systems Menu

### 2.1.3.1.3 System Configuration

'SysConfig' Menu provides menu items for configuring the system.

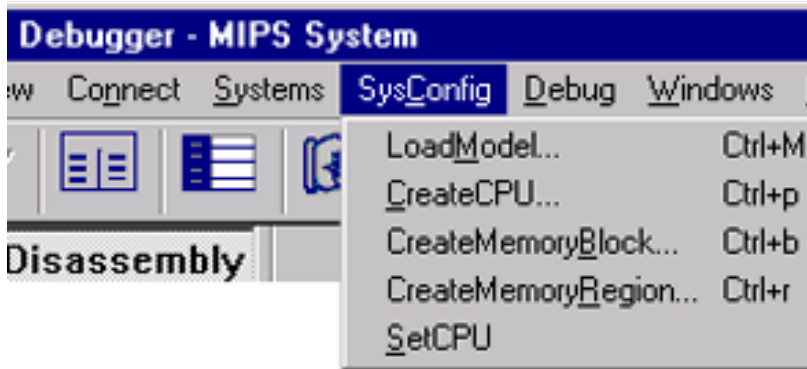


Figure 2.6 SysConfig Menu

The following explains the functionality of each menu item in detail.

“**CreateCPU**” allows the user to create a CPU for the connected system.

Invoke “SysConfig->CreateCPU” or use the hotkey “Ctrl+P”. A dialog box titled “Create CPU” will pop up as shown below.

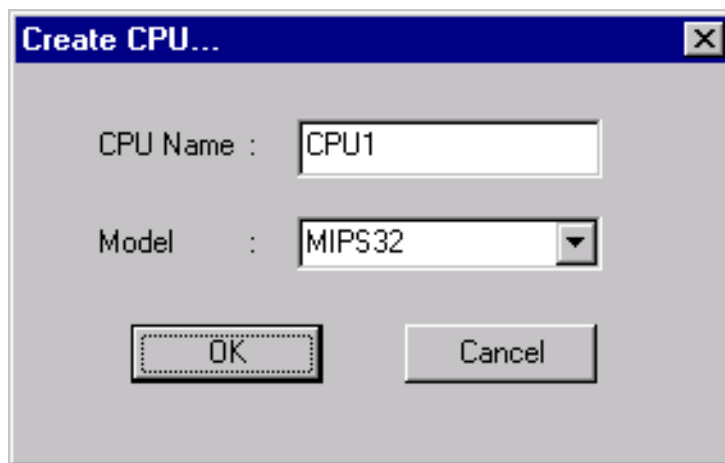


Figure 2.7 CreateCPU

Specify the following details in the dialog box and click ‘OK’.

CPU Name : Name reference of the CPU. The default value is ‘CPU1’

Model : Processor model to be used. The default value is “ MIPS32”  
and “ARMv4” for MIPS and ARM targets respectively.

“**CreateMemoryBlock**” allows the user to create a memory block and associate with the system. Invoke “SysConfig->CreateMemoryBlock” or use the hotkey “Ctrl+B”.

A dialog box titled “Create Memory Block” will pop up as shown below.

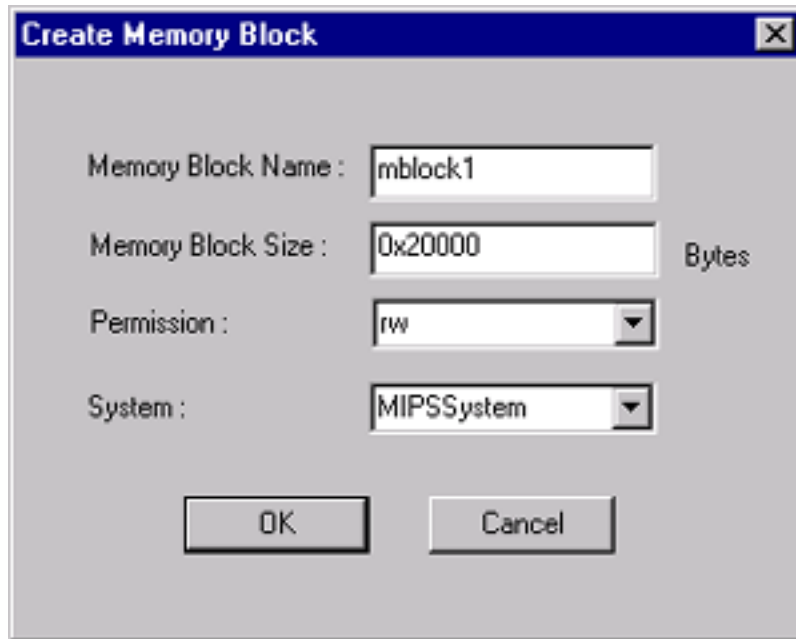


Figure 2.8 CreateMemoryBlock

Specify the following details in the dialog box and click ‘OK’.

- Memory Block Name - Name of the memory block. The default value is ‘mblock1’.
- Memory Block Size - Size of the memory block. The default value is ‘0x2000’.
- Permissions - Permissions for the memory block. Default value: “rw”.
- System - Name of the System.

The default value(MIPSystem, ARMSystem) for the combo box “System” is the one specified in the configuration file mips\_sys.ini and arm\_sys.ini in %SD\_HOME%\etc directory.

“**CreateMemoryRegion**” allows the user to create a memory region for the CPU. Click on “SysConfig->CreateMemoryRegion” or use the hotkey “Ctrl+R”. A dialog box titled ‘Create/Map CPU Memory Region’ will pop up as in figure 1.2.

Specify the following values in the ‘Create/Map CPU Memory Region’ dialog box and click ‘OK’.

CPUName - Select the cpu name reference from the combo box.

MemoryBlock - Select the memory block from the combo box.

StartAddress - Specify the starting address of CPU’s memory region.

Attributes - Select “rw” from the combo box. The attribute “rw” specifies that the memory region will be created with read write permissions.

A memory region will be created with the specified starting address.

“**SetCPU**” allows the user to select the current CPU for debugging the application.

#### **2.1.3.1.4 Debug**

‘**Debug**’ menu provides menu items for loading an application to the target and debugging the application.

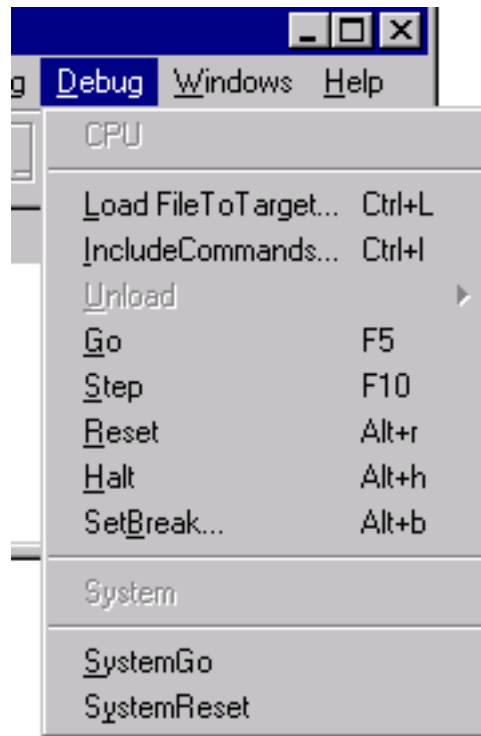


Figure 2.9 Debug Menu

“**LoadFileToTarget**” allows the user to load the application to the target.

Invoke “Debug->LoadFileToTarget” or use the hotkey “Ctrl+L”. A file dialog box titled “LoadFileToTarget...” will pop up as shown below.

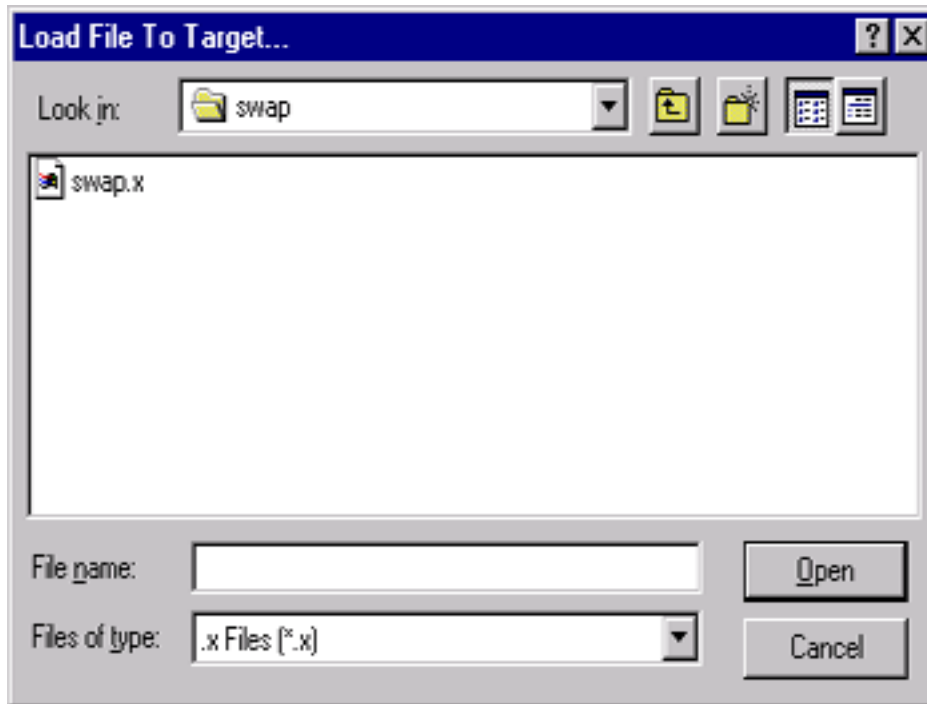


Figure 2.10 Load File To Target

Application can be loaded by specifying the name in the text field “File name”. Once an application is loaded, assembly level instructions will be displayed in the main window.

“**IncludeCommands**” menu item allows the user to include the command file. Debugging commands can be given through the command file.

Invoke “Debug->IncludeCommands” or use the hotkey “Ctrl+I” to open the file dialog box titled “IncludeCommand File...”.

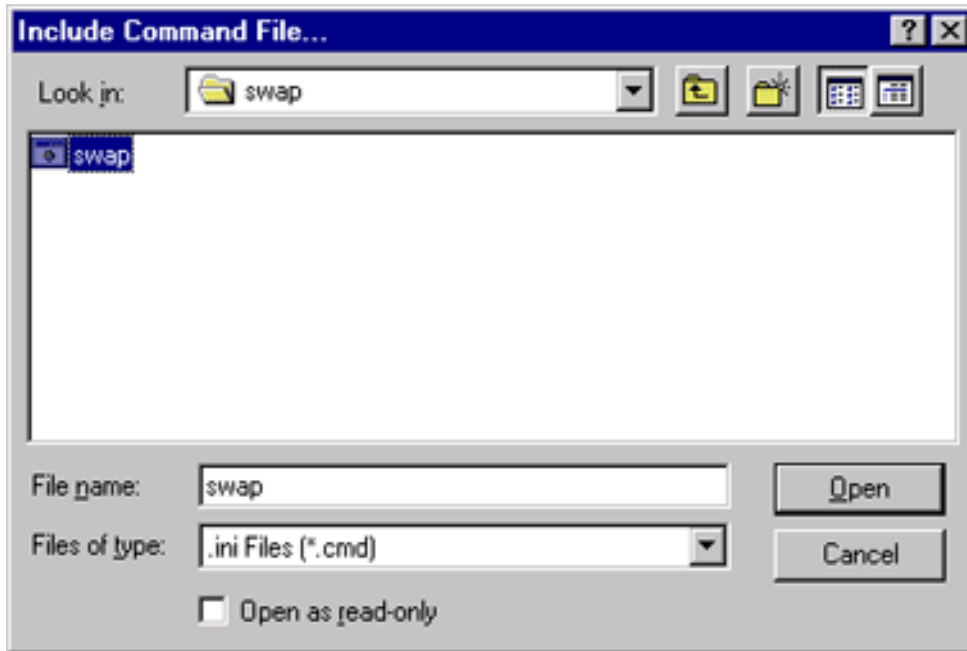


Figure 2.11 Include Command File

“Go” will start the execution and it will stop until it reaches a break-point or the end of the execution.

“Step” will perform single-step execution of the instruction.

### 2.1.3.1.5 Breakpoint Management

‘Debug’ menu provides the ‘SetBreak’ menu item to set a breakpoint at a specified address. Invoke “Debug->SetBreak” or use the hotkey “Alt+B”. A dialog box titled “Set Break” will pop up as shown below.

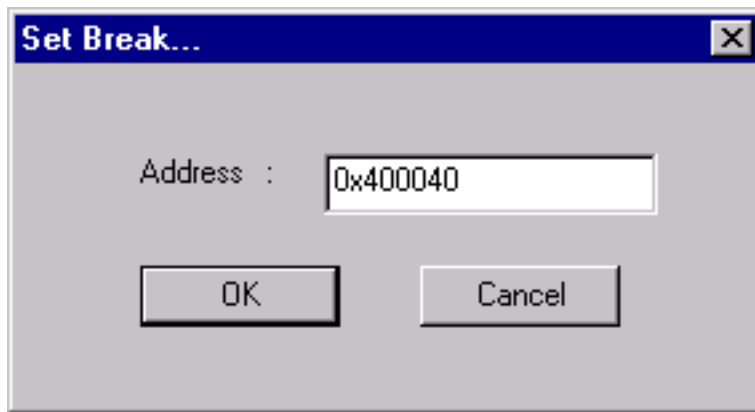


Figure 2.12 Set break dialog box

Specify the address at which breakpoint should be set and click 'OK'.

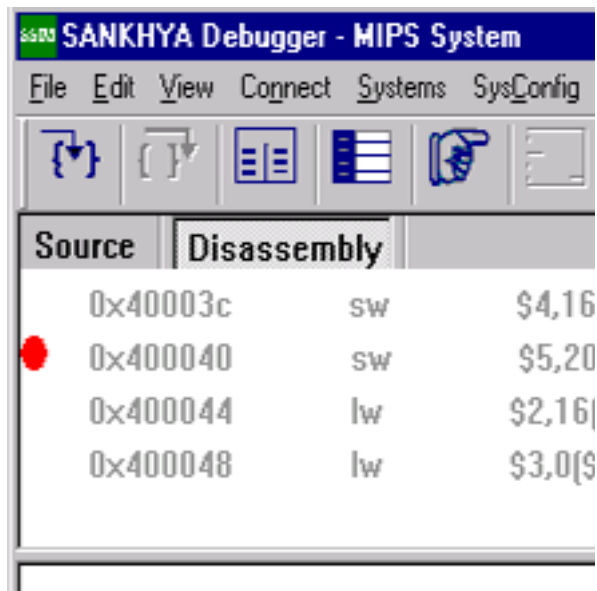


Figure 2.13 Window with breakpoint

The breakpoint will be set in the specified address and reflected in the code window highlighted by the red spot as shown in figure 2.13.

Breakpoints can be added from the debugger main window using mouse right click event. On right clicking at the specified address, a pop up menu will be displayed as shown below.

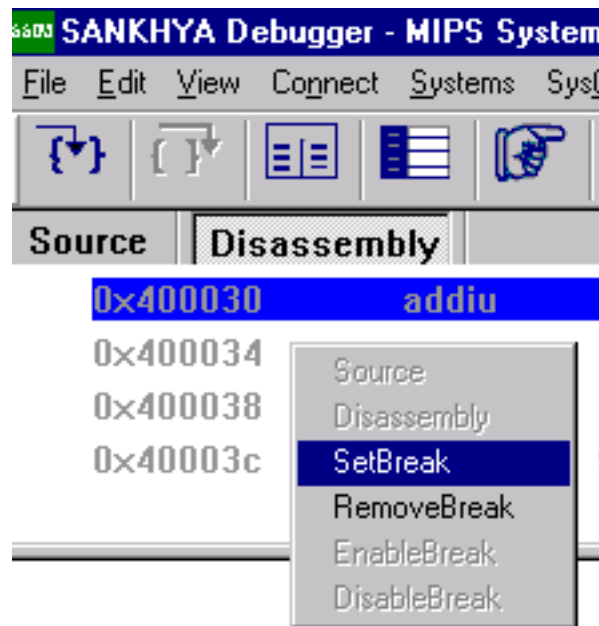


Figure 2.14 SetBreak with mouse right click

Removal of breakpoints can be done from the main window by right clicking at the address at which breakpoint is set.

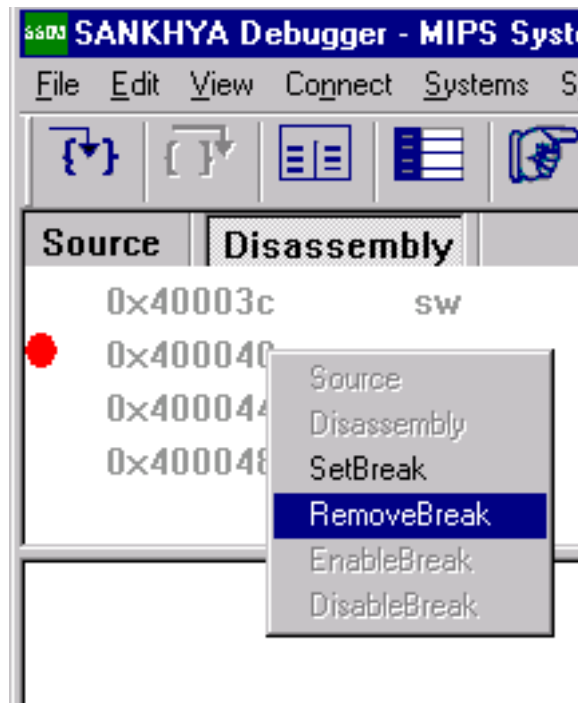


Figure 2.15 RemoveBreak with mouse right click

Breakpoint Management can be done using breakpoint manager window.

### 2.1.3.1.6 Windows Management

'**Windows**' Menu provides list of menu items of debugger windows. Selecting each of this menu item will open the corresponding window.

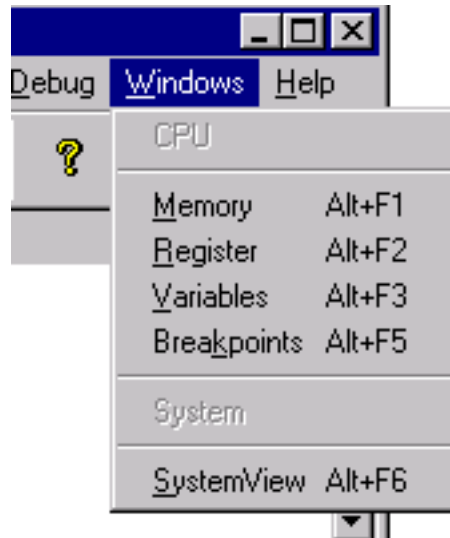


Figure 2.16 Windows Menu

## 1. Register Window

Register window is used to view and update the values of processor registers. Register window provides display mode option to display the values in hexadecimal and decimal format. Register window title will display the system and CPU name used for debugging.

Register Window can be opened by Clicking “Windows->Register”. Register Window for MIPS displays the 31 general purpose registers, Program counter, HI and LO registers and pseudo registers.

Here is the screenshot of Register Window for MIPS.

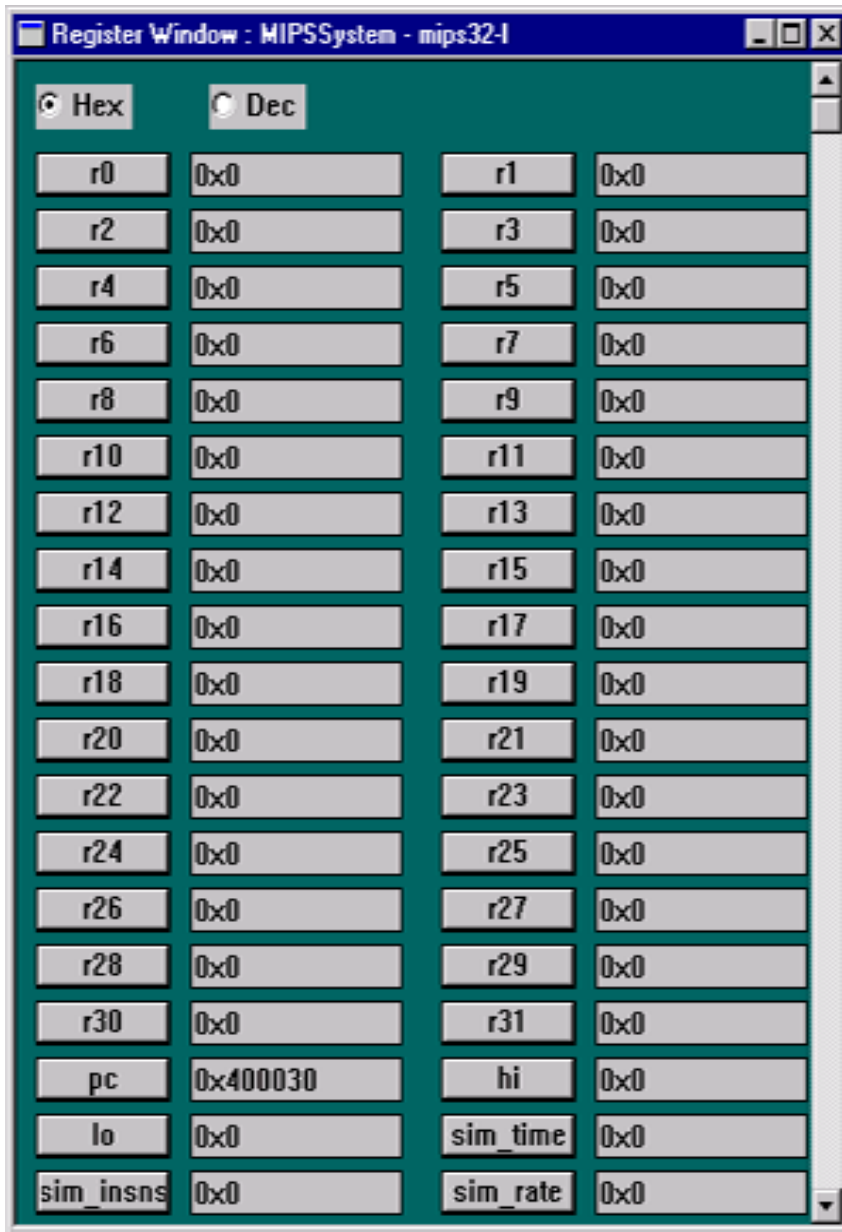


Figure 2.17 Register Window for MIPS

Here is the screen shot of register window for ARM.

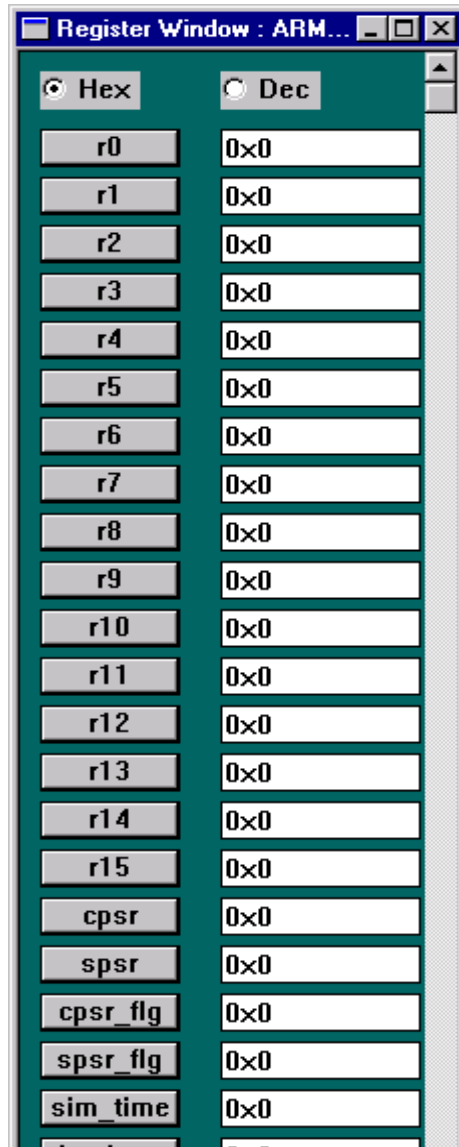


Figure 2.18 Register Window for ARM

Pseudo Registers are the registers which are supported by Sankhya Debugger in order to measure the performance of the simulated targets. These are not the actual processor registers.

Currently the following pseudo registers are supported by Sankhya Debugger :

- `ssim_insns` - No of instructions executed
- `ssim_time` - Execution time (in micro seconds)
- `ssim_rate` - Execution Rate(Instructions executed per second)

Values can be set for a register using the following steps.

**Step1:**

Click the register button in the register window. To set the value for r0, click the ‘r0’ button. A dialog box titled “Edit Register” will pop up as shown below.

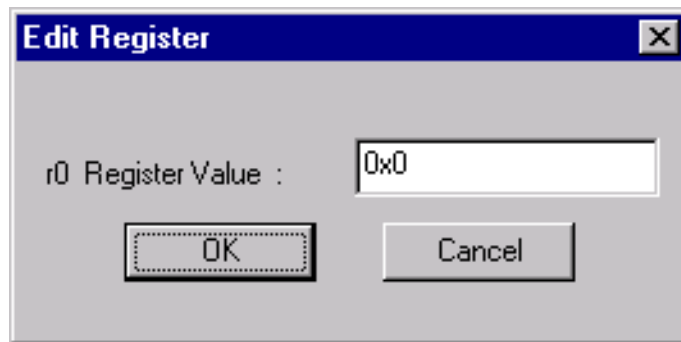


Figure 2.19 Edit Register

**Step2:**

Specify the value to be set in the text field and click ‘OK’. Now the specified value will be displayed in the corresponding text field of the register window.

## 2. Memory Window

Memory window displays the memory values starting at the address specified by ‘Start:’. Memory Window can be opened by clicking “Windows->Memory”. Memory window title will display the System and CPU name used for debugging.

Here is the screenshot of Memory Window for MIPS.

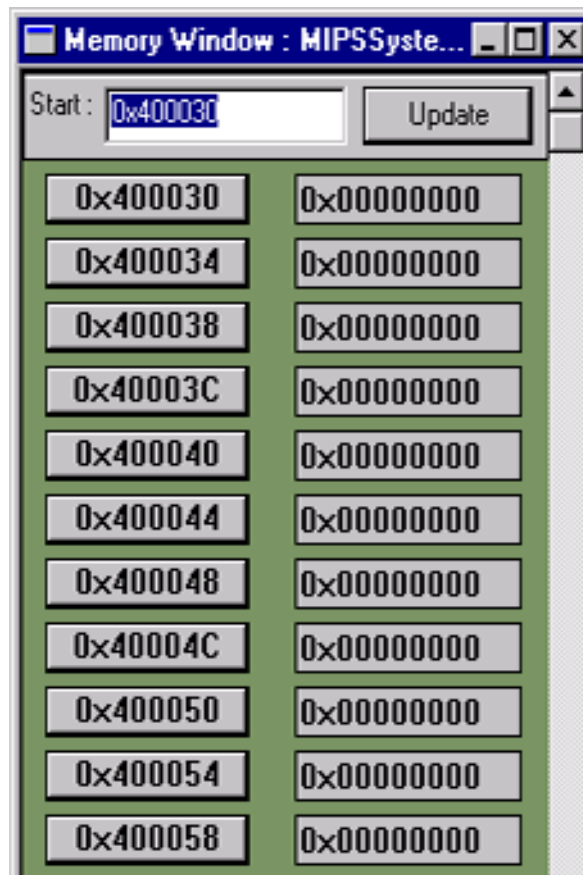


Figure 2.20 Memory Window for MIPS

Values can be set at a memory address using the following steps.

**Step1:**

Click the button with the memory address in the memory window.

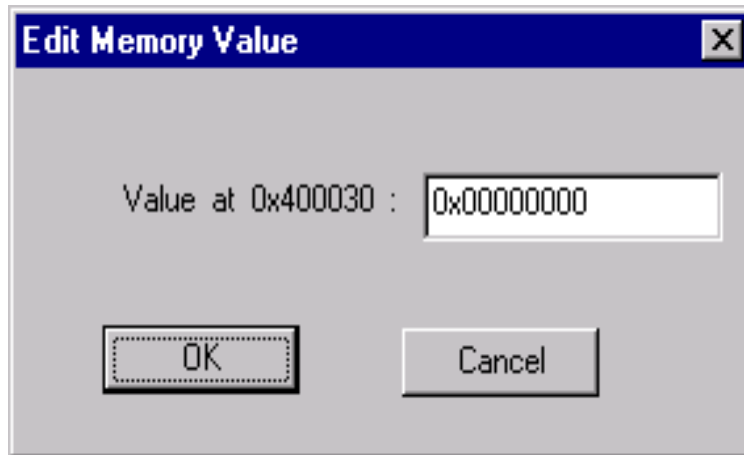


Figure 2.21 Edit Memory Window

A dialog box titled “Edit Memory Value” will pop up as shown above.

**Step2:**

Specify the value to be set in the text field and click ‘OK’.

Now the specified value will be displayed in the corresponding text field of the memory window.

### 3. Breakpoint Manager Window

Breakpoint Manager window displays the list of breakpoints that are set for the current CPU. Adding, Removing, Enabling and Disabling a breakpoint can be done through this window.

Breakpoint Manager Window can be opened by clicking “Windows->Breakpoints”.

Here is the screenshot of Breakpoint Manager Window.

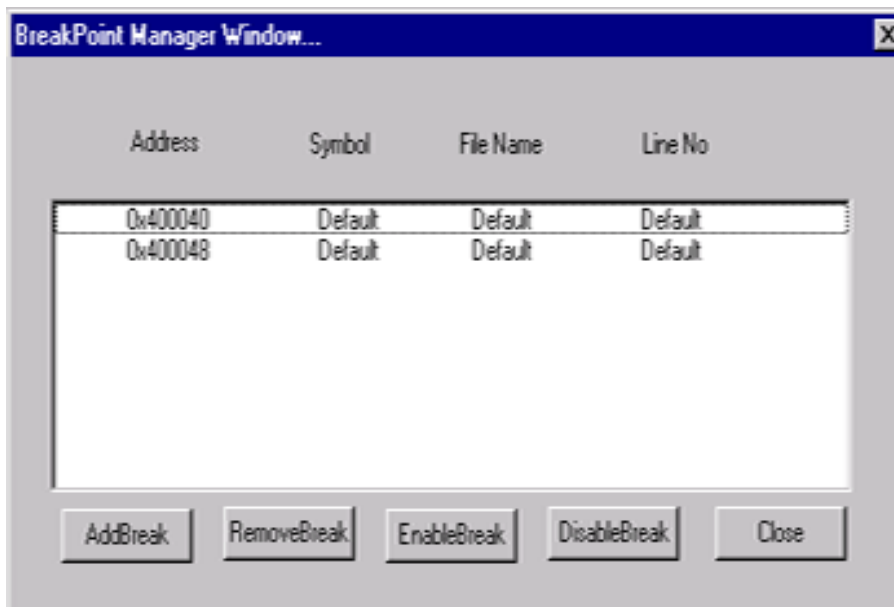


Figure 2.22 BreakPoint Manager Window

The following table lists the menu item and its corresponding hotkey.

<b>Menu Item</b>	<b>Hot Key</b>
CreateSystem	Ctrl+A
ConnectSystem	Ctrl+S
Disconnect	Ctrl+D
CreateCPU...	Ctrl+P
CreateMemoryBlock...	Ctrl+B
CreateMemoryRegion...	Ctrl+R
LoadFileToTarget...	Ctrl+L
IncludeCommands...	Ctrl+I
Go	F5
Step	F10
Reset	Alt+R
Halt	Alt+H
SetBreak	Alt+B
Memory Window	Alt+F1
Register Window	Alt+F2
Variables Window	Alt+F3
Breakpoints Manager Window	Alt+F5

## 2.2 SANKHYA Debugger Commands

SANKHYA Debugger accepts the following debugging commands in the command file. A command beginning with a “;” is treated as a comment by the debugger. The following lists the debugging commands that can be specified in the command file.

`memset <addr> <value>`

- to write given value into given address

**Example:** `memset 0x400030 0x27de0010`

`set <regname> <val>`

- to set register value

**Example:** `set r4 0x10`

`step [<count>]`

- performs single step through each machine instruction by default.

- performs multiline step according to the count option

**Example:** `step 8`

`run [<address>]`

- to execute the program from PC until exception or breakpoint is reached

- to execute the program from PC until the address specified as option is reached

**Example:** `run 0xffffffff`

`break <address>`

- to set the breakpoint at the specified address.

**Example:** `break 0xfec`

## *Appendix*

### **Additional Readings and Further Recommendations**

1. MIPS32 4K™ Processor Core Family Software User's Manual.
2. ARM Architecture Reference Manual.

## *Index*

### B

Background Debug Mode 2

Breakpoint Manager 39

### C

Connection Management 19

### D

debugging 2

### M

Memory Block 25

Memory window 37

### P

Pseudo Registers 36

### R

Register window 33

### S

Sankhya Debugger 1

SD Code window 19

sdarm 4

sdmips 4

Simulator server 10

System Configuration 23

System Level debugging 2

System Management 22

## For More Information

SD Download	<a href="http://www.sankhya.com/info/products/tools/download.html">http://www.sankhya.com/info/products/tools/download.html</a>
SD Documentation	<a href="http://www.sankhya.com/info/products/tools/docs.html">http://www.sankhya.com/info/products/tools/docs.html</a>
SD Sales & Support	<a href="mailto:sales@sankhya.com">sales@sankhya.com</a>

---

## SANKHYA™

**Sankhya Technologies Private Limited**  
**#13/2, "JayaShree", Third Floor, First Street, Jayalakshmipuram,**  
**Nungambakkam,**  
**Chennai 600 034, INDIA**  
**Tel: +91 44 2822 7358**  
**Fax: +91 44 2822 7357**

**Sankhya Technologies India Operations Private Limited**  
**#30-15-58, "Silver Willow", Third Floor,**  
**Dabagardens,**  
**Visakhapatnam 530 020, INDIA**  
**Tel: +91 891 554 2666**  
**Fax: +91 891 554 2665**  
**Email: [sales@sankhya.com](mailto:sales@sankhya.com)**  
**<http://www.sankhya.com>**

---

SANKHYA, SANKHYA TECHNOLOGIES, SANKHYA Debugger, SANKHYA Tools Collection, Dynamically Targetable Tools Framework, SANKHYA Varadhi, SANKHYA Software are Trademarks, Service Marks or Registered Trademarks of Sankhya Technologies Private Limited. All other brands and names are the property of their respective owners.