PowerPC

CPU Awareness and True-Time Simulation

Revised 12-Feb-2004

metrowerks

How to Contact Metrowerks

Corporate Headquarters	Metrowerks Corporation 7700 West Parmer Lane Austin, TX 78729 U.S.A.
World Wide Web	`http://www.metrowerks.com`
Sales	Voice: 800-377-5416 Fax: 512-996-4910 Email: sales@metrowerks.com
Technical Support	Voice: 800-377-5416 Email: support@metrowerks.com

Table of Contents

1

Introduction

This manual explains the Metrowerks target software simulator.

Read the Release Notes

Before you use your CodeWarrior™ IDE simulator, you should read its product release notes. The release notes include important last-minute information about new features, problem workarounds, or incompatibilities that may not be included in this manual.

Simulator Target Component

This section helps you start using the Simulator Target Component.

Introduction

Simulator software simulates a target system. The simulator consists of a CPU simulator, a memory simulator, and several simulated I/O devices. The simulator lets you set the simulated environment: memory, I/O-device placement, code, and so forth.

The simulator includes a universal timing facility that components can use to simulate realistic timing conditions. This facility lets components take control after a certain number of clock cycles or processor instructions.

You load the simulator driver as part of loading the simulator target component.

Simulator Setup

This section explains how to load the Simulator target.

Default Target Setup

As with any other target, you can load the simulator target component from the CodeWarrior IDE *Target* menu. Alternatively, you can use the `PROJECT.INI` file (Listing 1.1 on page 6) to set the simulator target component as the default target.

Listing 1.1 Example of PROJECT.INI File

```
[HI-WAVE]
Window0=Source      0   0  60  30
Window1=Assembly   60   0  40  30
Window2=Procedur    0  30  60  25
Window3=Register   60  30  40  30
Window4=Memory     60  60  40  40
Window5=Data        0  55  60  23
Window6=Data        0  78  60  22
Target=Sim
```

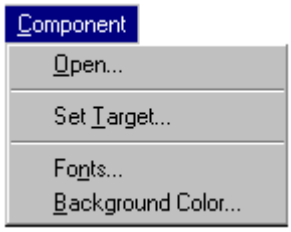
NOTE The HI-WAVE User's Guide has additional information about the `PROJECT.INI` file.

Loading the Simulator Target

The `PROJECT.INI` file line **`Target=Sim`** sets the target to be the simulator target component.

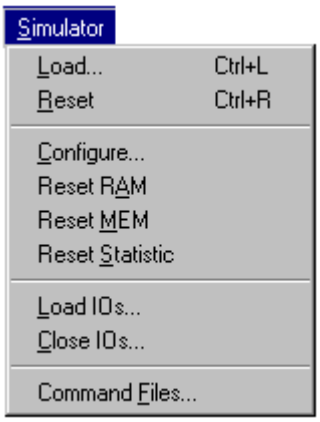
If the `PROJECT.INI` file does not set a target, or if it sets a different target, you can use the main menu to select the simulator. Select **Component > Set Target...** , as Figure 1.1 depicts. Choose **Simulator** from the list of possible targets.

Figure 1.1 The Component Menu



After loading, the **Simulator** (Figure 1.2 on page 7) replaces the **Target** menu.

Figure 1.2 The Simulator Menu



The HI-WAVE Status Bar for the Simulator

Once you have loaded the Simulator Target Component, the HI-WAVE status bar (Figure 1.3) shows status and other information. As well as execution status, it includes a context-sensitive menu help line, and target- specific information like the number of CPU cycles (64 bits) since the application started.

Figure 1.3 The Debugger Status Bar

Simulator Target Component Features

This section explains the major features of the Simulator Target Component.

Introduction

The memory configuration facility is an integral part of HI-WAVE's advanced target configuration possibilities. The memory is divided into blocks. Amemory manager handles the list of memory blocks. The memory configuration facility offers you some degree of automation, but does not restrict the flexibility of manual adjustment. The memory configuration facility lets you specify types and properties of memory blocks, such as RAM, ROM, and so forth.

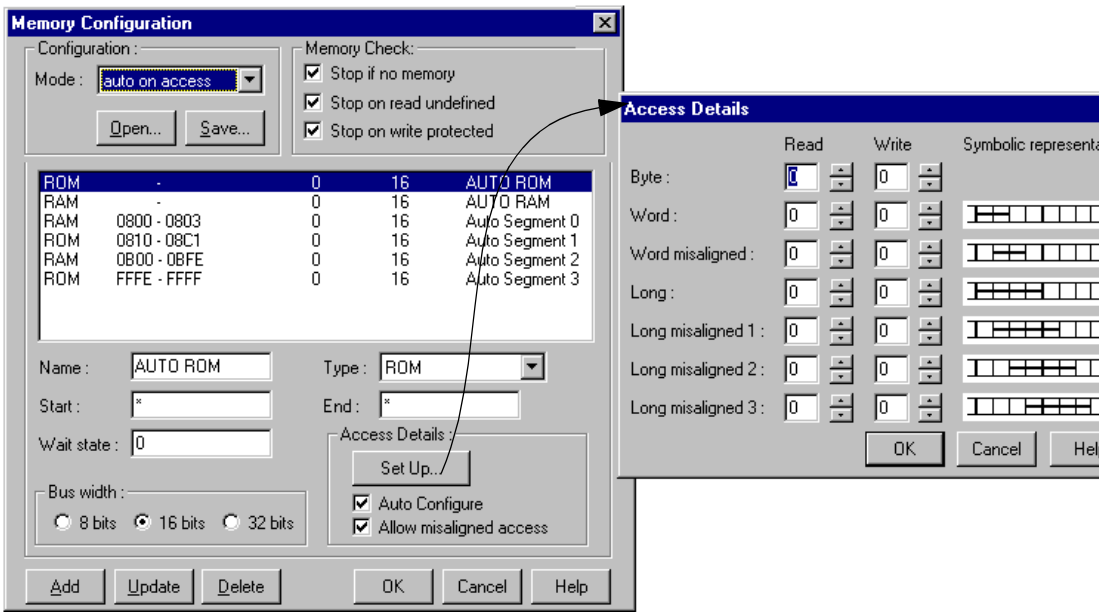
The memory configuration facility uses a binary file format to read and set the target configuration. The extension for binary files is `.mem`; the default memory file is `default.mem`. (The subsection "Format of the Default Memory Configuration File" includes Listing 1.2 on page 14, the EBNF-syntax definition of the file format.)

Memory Configuration Dialog Box Features

The memory configuration dialog box (Figure 1.4 on page 8) lets you perform these memory-block operations interactively:

- Select the configuration mode for simulation
- Define a memory block name
- Define how the simulator verifies the memory
- Set the type of the memory: RAM, ROM, FLASH, EEPROM or I/O
- Define start and end addresses
- Define the wait state (the time for each read or write access)
- Set the width of the bus that accesses the memory
- Set access details like:
  - *auto configure*: automatically computing read and write access
  - *misaligned access*: allowing misaligned access on words and longs
- Open and save memory configuration
- Add, delete, or update memory blocks

Figure 1.4 The Memory Configuration Dialog Box



Memory Configuration Modes

Use the **Memory Configuration** dialog box to select the memory configuration mode: **auto configuration on access**, **auto configuration on load,** or **user defined**. Depending on your settings, the the simulator target component initializes target memory as Table 1.1 explains.

Table 1.1 Memory Configuration Modes

Mode	Description
Auto Configuration on Access (Standard Configuration)	Defines target memory as RAM of unlimited size. The *Mode* combo box displays *auto on access*.
Auto Configuration on Load (default)	Defines target memory as RAM and ROM, according to the code and data area defined in a loaded absolute file. Defines code segments as ROM. Defines data segments as RAM. (Memory outside these segments is *not implemented*; access to not-implemented locations result in error messages.) The *Mode* combo box displays *auto on load*.
Manual Configuration: (User Defined)	Defines target memory as RAM, ROM, non-volatile RAM, ... , depending on your configuration. You construct this definition interactively with the Memory Configuration dialog box, or read it in from a file. The *Mode* combo box displays *user defined*.

Memory Configuration Settings

Depending on the configuration mode, the Memory Configuration dialog box lets you redefine memory settings within certain limits. You always must set I/O devices manually.

Standard Configuration: Auto on Access: The Memory Configuration dialog box contains a single RAM entry with unspecified (*) starting and ending addresses. You cannot modify these addresses. You can adjust wait states, and other such settings, only for the whole RAM block.

Auto Configuration on Load: Initially, the dialog box lists a single RAM and a single ROM block, with unspecified (*) starting and ending addresses. You can adjust wait states, and other such settings, separately for RAM and ROM blocks.

For the ELF/DWARF Object file format, the Memory Configuration dialog box lists separate RAM and ROM blocks for each data and code segment in the absolute file, once an application has been loaded. The segmane addresses and lengths determine the starting and ending addresses of each block; you cannot modify these addresses.

Initial attributes of each code and data block come from the corresponding initial RAM and ROM blocks; you can modify these attributes independently.

Manual Configuration: The Memory Configuration dialog box lists an entry for each memory block. You can modify such entries without restriction.

NOTE	To simulate an absolute file generated in HIWARE object file format, you must open the Memory Configuration dialog box, set the "**auto on load**" mode, then add a new RAM segment. The start and end addresses of this segment must match the associated `.prm` file. Once you close the dialog box, you can load your application and start a simulation.

Open Memory Block

Click the **Open** button to load a memory blocks file. The **Open Memory blocks** standard dialog box appears. Select a memory map file, then click the **OK** button. The dialog box closes, and the system loads the memory blocks file.

The *Mode* combo box changes to indicate the mode contained in the memory map file.

The list box lists the memory blocks loaded from the file, selecting the first memory block. Appropriate data appears in the fields **Name**, **Type**, **Start**, **End**, **Wait state**, **Bus width** and **Access Details**.

Save Memory Block

Click the **Save** button to store the current memory blocks configuration. The **Save Memory blocks** standard dialog box appears. Enter a file name, then click the **OK** button. The dialog box closes, and the system stores the memory block configuration into the file.

Memory Check Options

The Memory Check group box consists of three checkboxes, all checked when you bring up the Memory Configuration dialog box:

- Stop if no memory — Check this box to have the simulator stop upon an access to non-existent memory. (If you do not want the simulator to stop, clear this checkbox.)
- Stop on read undefined — Check this box to have the simulator stop upon a read of undefined memory. (If you do not want the simulator to stop, clear this checkbox.)

- Stop on write protected — Check this box to have the simulator stop upon a write to read-only (write-protected) memory. (If you do not want the simulator to stop, clear this checkbox.)

Memory Configuration Module Startup

Memory configuration is a *dynamically loaded* facility. That is, the new entry **Configure**... appears in the *Simulator* menu upon loading of the target (the Simulator dll). Selecting **Configure**... opens the Memory Configuration dialog box, so that you can configure memory.

Memory Block Setting

You must set memory blocks within the available memory; each block must cover a certain range. The *start address* and *end address* define each memory block.

Memory Block Properties

Table 1.2 lists the properties you may specify for a memory block:

Table 1.2 The Memory Block Properties

Item	Description
name	Name of the memory block.
type	RAM, ROM, FLASH, EEPROM or I/O
start	Start address of the memory block
end	End address of the memory block
wait state	Time used for reading or writing a specific number of bytes
bus width	Width of the bus that accesses the memory
read access	Table that defines read-access details on Byte, Word, Word misaligned, Long, and Long misaligned
write access	Table that defines write-access details on Byte, Word, Word misaligned, Long, and Long misaligned
auto configure	Flag that directs automatic computation of read and write accesses

Table 1.2 The Memory Block Properties

Item	Description
allow misaligned access	Flag that allows Word misaligned and Long misaligned
block type	USER_DEF (block you define), AUTO_GEN (block automatically generated), AUTO_MEM (master block for standard configuration), AUTO_RAM (RAM master block for auto configuration), or AUTO_ROM (ROM master block for auto configuration)

Memory Configuration Command Buttons

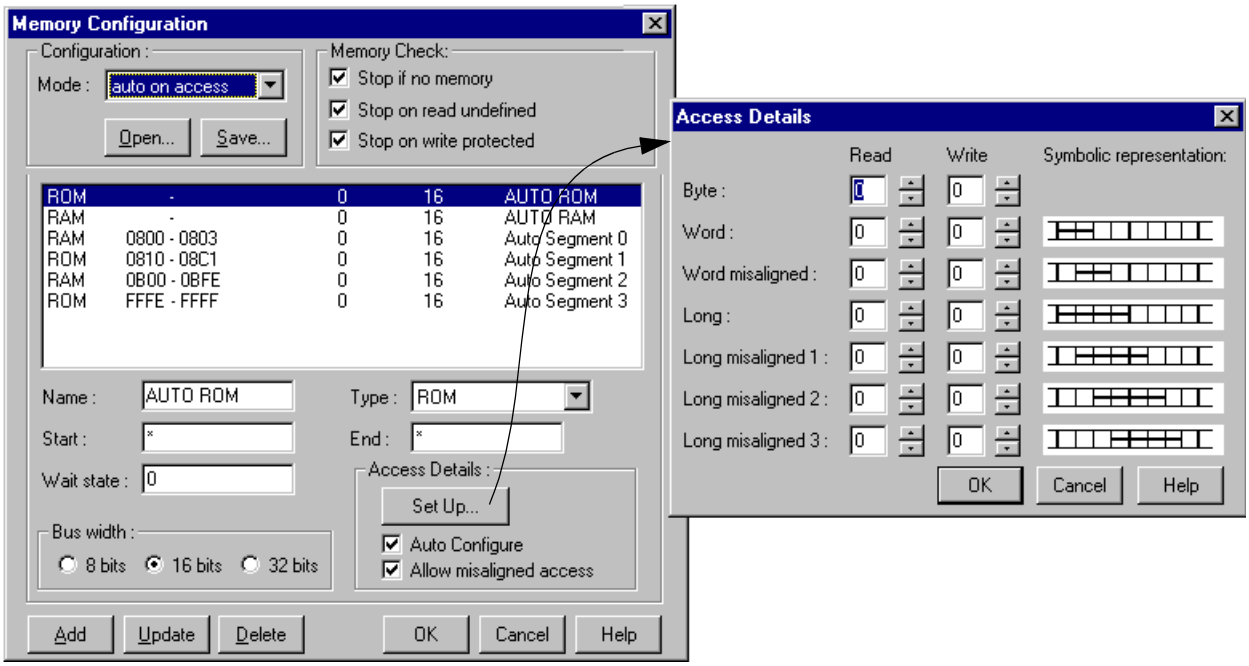
The command buttons of this dialog box are:

- **Add** — Fills a new memory block according to the current data of the **Name**, **Type**, **Start**, **End**, **Bus width**, and **Access Details** controls. This new memory block appears at the end of the list box. If there are any errors in this new block (such as an improper field value), the system generates a message box that informs you of the problem.
- **Update** — Updates the current memory block according to the current data of the **Name**, **Type**, **Start**, **End**, **Bus width**, and **Access Details** controls.
- **Delete** — Removes the currently selected memory block from the list box. The list box contents adjust, to reflect this deletion.
- **OK** — Closes the dialog box and validates the list of modified memory blocks. The parent class can access this list, updating its own list.
- **Cancel** — Closes the dialog box, canceling your modifications.
- **Help** — Opens the dialog-box help file.

Access Details Dialog Box Features

Figure 1.5 shows the **Access Details** dialog box, which lets you change read and write access values for seven types.

Figure 1.5 Access Dialog Window



Follow this guidance to use the **Access Details** dialog box::

- To modify the value of each read or write type, change the value of the associated spin box.
- The lowest possible value is -1.
- The highest possible value is 100.
- To store changes into the currently selected memory block, click the **OK** button. The **Access Details** dialog box disappears, and the system clears the **Auto Configure** checkbox.
- To abandon your changes, click the **Cancel** button. The **Access Details** dialog box disappears; the system discards your changes.
- To bring up appropriate help information, click the **Help** button.

Output

You can save the current memory configuration into the file you defined at the outset.

Listing 1.2 shows the format of the Default Memory Configuration File, in EBNF notation.

Listing 1.2 Format: Default Memory Configuration File.

```
memConfFile = head mode numberBlocks data
head = number
mode = STD_MODE | AUTO_MODE | MAN_MODE
numberBlocks = number
data = {memoryBlock}
memoryBlock = name type start end waitState busWidth accessRead
accessWrite autoConfigure allowMisalignedAccess blockType
name = string
type = string
start = number
end = number
waitState = number
busWidth = number
accessRead = array of number
accessWrite = array of number
autoConfigure = boolean
allowMisalignedAccess = boolean
blockType = USER_DEF | AUTO_GEN | AUTO_MEM | AUTO_RAM | AUTO_ROM
```

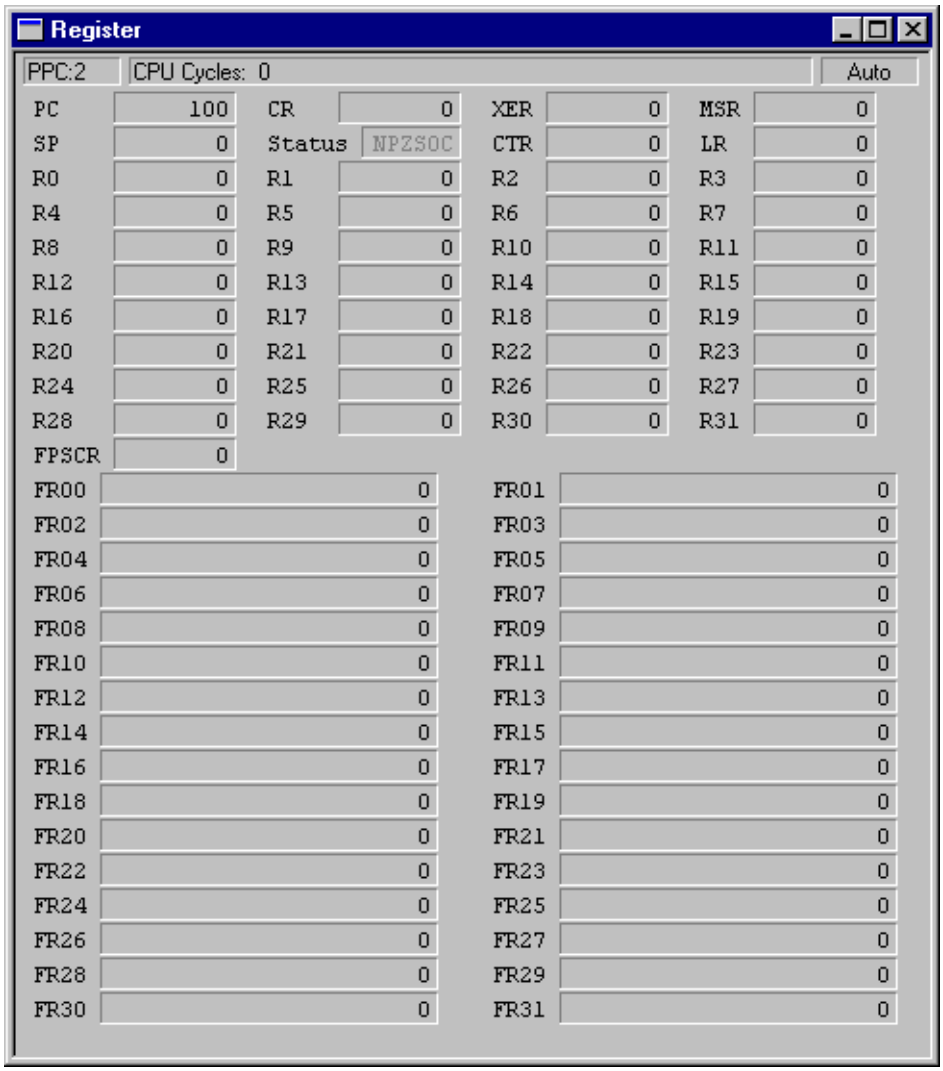
The following documents are available from Motorola:

PowerPC Simulator Specifics

PowerPC Registers

The **Register Components** window (Figure 1.6) displays the PowerPC processor register. Values can be in any of five formats: hexadecimal, binary, octal, decimal, or unsigned decimal.

Figure 1.6 The Power-PC Register Components Window



- PC: Program counter register value
- CR: Condition register value
- XER: Carry, overflow, or other condition indicator value
- MSR: Machine state register values
- SP: Special purpose register value
- Status
- CTR: Count register values
- LR: Link register values
- R0 to R31: General purpose register values
- FPSCR: Floating-point status and control register value
- FR00 to FR31: Floating-point register values

NOTE For more information, see the Motorola PowerPC reference manual.

Displaying Special Registers

To visualize registers that the debugger **Register** component does not display, use either of two methods:

- Displaying Registers with the RD Command.
- Displaying Registers with the Visualization Tool Component.

Displaying Registers with the RD Command.

Open the debugger **Command Line** component and use the **RD** command:

Example:

```
in>RD spr44
in>SPR44=0x0
```

NOTE For more information about the RD command, please see to the True Time Simulator and real Time Debugger manual.

Displaying Registers with the Visualization Tool Component.

Open the debugger **Visualization Tool**, then:

1. Create a new instrument (such as, value as text)
2. Set kind of port to Register
3. Set port to display to your register (e.g. spr44)

NOTE For more information about the Visualization Tool, please see to the Visualization Tool manual.

Index