Rewind User's Guide

Literature Number: SPRU713B
August 2005

Printed on Recycled Paper

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments
Post Office Box 655303 Dallas, Texas 75265

Preface

Read This First

About This Manual

This user guide describes a special feature supported by C6000 and C5500 simulators in the Code Composer Studio (CCStudio) environment called Rewind. Using this feature, you can view the past history of application being executed on the simulator, reducing the time to debug the application.

Related Documentation From Texas Instruments

TMS320C55x Instruction Set Simulator Technical Reference (SPRU599) describes the different C5500 simulator configurations that are available with Code Composer Studio. It includes details on simulator types, hardware modelled, supported features, and cycle accuracy.

TMS320C6000 Instruction Set Simulator Technical Reference (SPRU600) describes the different C6000 simulator configurations that are available with Code Composer Studio. It includes details on simulator types, hardware modelled, supported features, and cycle accuracy.

To obtain a copy of any of these documents, call the Texas Instruments Literature Response Center at (800) 477–8924. When ordering, please identify the book by its title and literature number.

Trademarks

Contents

Figures

Tables

Chapter 1

Introduction

The C6000 and C5500 simulators under the CCStudio environment support a new feature called Rewind. Using this feature you can view the past history of an application being executed on the simulator. This reduces the time required to debug an application.

Topic **Page**

1.1 The Rewind Feature

The process of debugging any software requires pinpointing the exact point of execution where the software starts behaving in some unintended way. The process of finding that exact point may take a few iterations to guess the correct failure point. The searching process may run as follows:

- Set a breakpoint at a possible point of error
- Run the application on the simulator up to this breakpoint
- Check all your visible resources (register and memory contents)
- If the resources are as expected, then you must choose another possible point of error ahead of the current point, or another possible point of error between the last guess-point and this point
- Repeat these steps until you pinpoint the error location

If you find the resources are as expected, and you suspect another possible point of error ahead, then you repeat the process. However, a problem arises if the next guess-point is behind the current point. As there is no way to run the application backwards, you would have to set a breakpoint at the new guess-point and run the application from the start. This process can be time-consuming and tiresome if you must test a sequence of guess-points that fall behind the current point.

With the Rewind feature, you do not have to restart your application on a guess point which falls behind. You can view the state of the resources at a point behind the current point just as you would view the resources at a guess point ahead.

1.2 C5500 CPU Simulator Support for Rewind Feature

The Rewind feature is currently available in the CCStudio environment with the following simulator configurations.

- C55x Rev 2.x CPU Functional Simulator
- C55x Rev 3.0 CPU Functional Simulator
- C55x Rev 2.x CPU Cycle Accurate Simulator
- C55x Rev 3.0 CPU Cycle Accurate Simulator

1.3 C6000 Simulator Support for Rewind Feature

The Rewind feature is currently available in the CCStudio environment with the following simulator configurations.

- C62xx CPU Cycle Accurate Simulator
- C64xx CPU Cycle Accurate Simulator
- C67xx CPU Cycle Accurate Simulator
- C6412 Device Functional Simulator
- C6416 Device Functional Simulator
- DM642 Device Functional Simulator
- C6713 Device Functional Simulator

Chapter 2

Rewind Feature in Application Software Debugging

This chapter discusses how the Rewind feature can be used to debug embedded application software easily.

Topic **Page**

2.1 Using the Rewind Feature

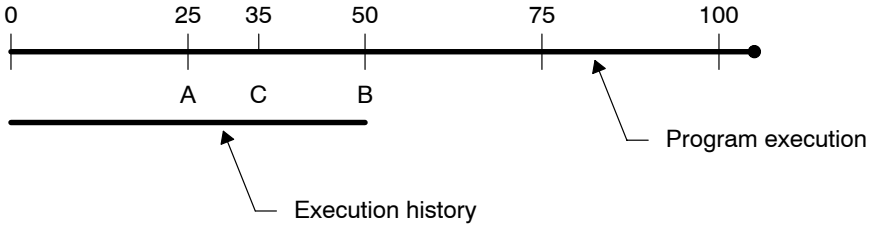
A simulator with the Rewind feature enables you to easily narrow down a suspected problem area. Rewind keeps a history of executions made by normal step and run commands. It also provides commands that allow you to run back or step back to view the previous program state, using information that has been stored during the normal run or step. The following example clarifies how to use these commands.

For example, Figure 2–1 illustrates a linear program without branches that has an address range between 0–100. You set breakpoints at addresses 25, 50, and 75 to find a problem. You issue a run command at address 25 (point A). The error is not visible at point A, however, it is visible at the next breakpoint at address 50 (point B).

Using the Rewind tool, you can step back (or undo) what has occurred. Thus, you can step back by one address until the problem occurs. Alternatively, you can set a breakpoint at address 35 and run back to that breakpoint (point C).

In this example, you still do not see the bug, so you step forward. The code is not re-executed from address 35–50 but is replayed based on the information that was stored during the original run. This allows you to determine exactly what has happened to the software system based on the input data that was used during the original run. Therefore, once the execution is captured, you can go back and forth in the history log as many times as needed and the inputs and outputs remain the same.

Figure 2–1. Example Program



2.2 Simulator Modes While Using Rewind

A simulator with the Rewind feature has two modes:

- **Real mode**. When you first execute forward a part of your application on the simulator, it is in real mode. You can modify resources in this mode. During real mode execution, the simulator generates a trace of the simulator state, which can be played back during replay mode.
- **Replay mode**. Any subsequent backwards or forwards execution of that part of the application is in replay mode. You can only view in this mode. You cannot modify the resources.

The mode change occurs as follows:

1) Initially, the simulator starts in real mode.
2) The simulator remains in real mode until you perform a backward advance. For instance, the time is t_{last} when the first backward advance from real mode is made. On backward advance, the simulator mode changes to replay.
3) The simulator remains in replay mode for all forward and backward advances within $t_{last.}$ Any forward advance from t_{last} changes the mode to real.
4) This process repeats.

A backward step or run can be made until either of the following happens:

- Previous trace information is not available, as it has already reached the beginning of execution history.
- Previous trace information is not available, as it has been overwritten by other execution history in the circular buffer (if the circular buffer is used for storing the history information).

In replay mode, you can observe the simulator state (see section 2.3). This state at time point *t* in replay mode shall be identical to that at time point *t* in real mode.

2.3 Simulator State in Replay Mode

The state of the simulator is affected by the replay mode as follows:

- **Memory**
  - All program and data memory values are shown correctly.
  - For all unmodeled peripherals, registers are modeled as flat I/O memory. Only this part of the simulator state is reproduced correctly in replay mode.
  - Other peripheral registers observed in replay mode may not show correct values; that is, the same value the register had in real mode execution.
- **Register**
  - CPU register values as identified by the CPU register window are correct.
- **Cycle count**
  - Clock value can be seen in replay mode when the profiling is set up for the following events.
    - C6000 CPU Cycle Accurate simulators: cycle.CPU
    - C5500 CPU Functional simulators: CPU.execute_packet
    - C5500 CPU Cycle Accurate simulators: cycle.CPU

Note:

Other simulator events are not updated in replay mode.

Please refer to the main online help in Code Composer Studio for information on profiling for different events.

The simulator internally manages the execution history. The simulator can be configured to keep the execution history in a host file or in the host memory. It is preferable to keep the execution history in a host file, providing there is enough free disk space to store execution history. For big applications, disk file size might be a constraint. To alleviate the problem, the simulator can be configured to keep the history in host memory in a circular buffer fashion. In that case, only the last few executed instructions can be replayed. Section 3.1 describes the configuration of the simulator for these options.

Chapter 3

Using the Rewind Feature with Simulators

This chapter provides details on working with the Rewind feature of the simulator.

Topic **Page**

3.1 Configuring the Simulators for Rewind

The Rewind feature is configurable from Code Composer Studio Setup for all supported simulators. This feature is disabled (OFF) by default. To enable Rewind, right-click on the processor for a simulator in the CCStudio Setup program and choose Properties from the context menu. In the Processor Properties dialog box, choose ON in the drop-down menu for the Rewind option. Once you enable Rewind, CCStudio Setup provides an additional option to use the host memory or host file for storing the Rewind traces. Use the option On Disk to use the host file, or the option In Memory (default) to use the host memory.

3.2 Debugging an Application With Rewind

When Rewind is enabled from Code Composer Studio Setup, the Code Composer Studio IDE starts up with the following new buttons. These buttons will not be visible if you disable the Rewind feature from Code Composer Studio Setup.

Table 3–1. Rewind Related Buttons and Their Functions

Button	Function
	Source single-step backwards
	Assembly single step backwards
	Run backwards up to the cursor
	Run backwards until break point or start point

The rewind functions are analogous to their forward counterparts. By using these buttons you can view the past state of the program. Executing the commands Reset, Reload, or Restart program will reset the trace generated. Execution cannot go backwards beyond these commands.

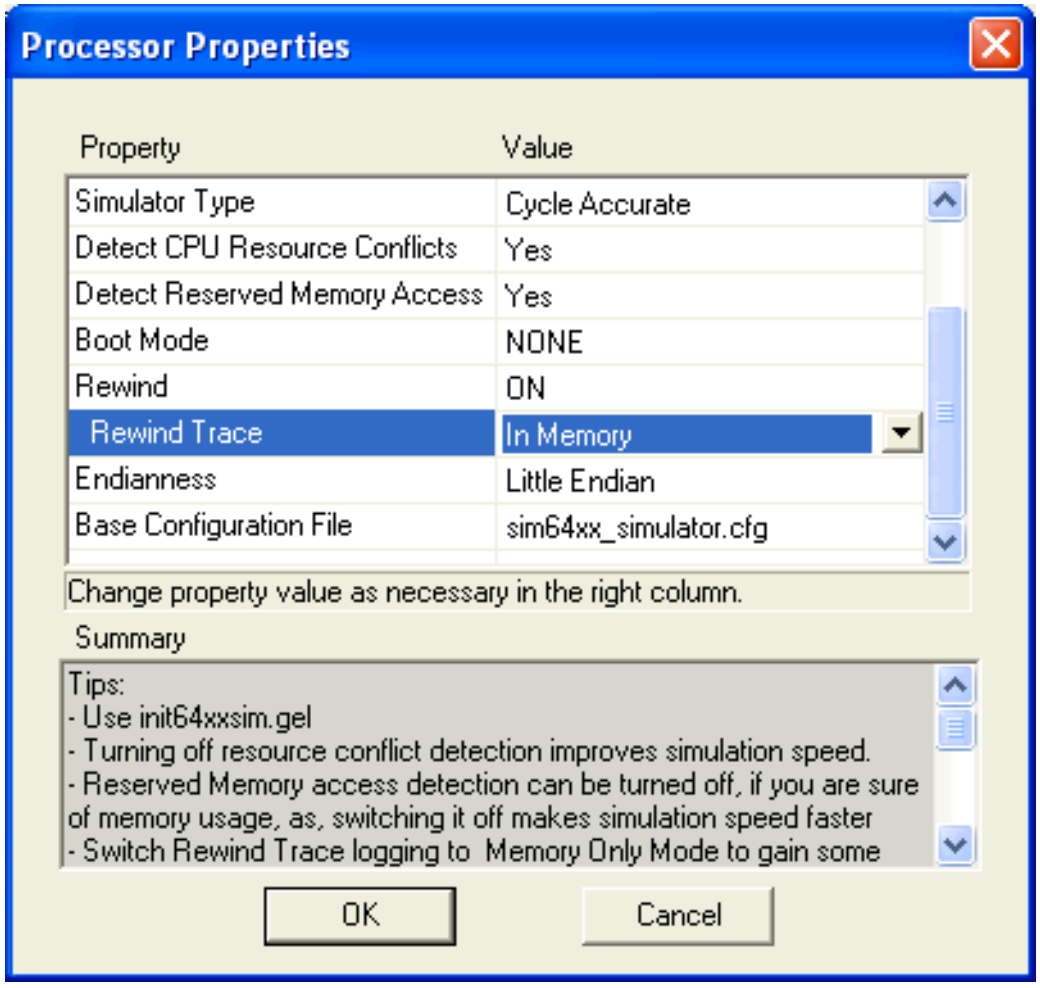
The icon  on the Code Composer Studio status bar indicates the replay mode of simulator.

3.3 Customizing Rewind

The Rewind feature reproduces past values of the registers and memory in replay mode by using the execution history (trace) generated during real execution. On average, the simulator generates four bytes of trace data per cycle. By default, the simulator keeps the trace in the host memory. The simulator can also be customized to keep this trace data in some temporary host file instead. Keeping the trace in host memory eliminates the problem of disk space constraints for larger applications, but it can only replay the last few instructions that were executed.

To enable Rewind, right-click on the processor for a simulator and choose Properties from the context menu. In the Properties dialog box, choose ON in the drop-down menu for the Rewind option. Once you enable Rewind, Code Composer Studio Setup provides an additional option to use the host memory or a host file for storing the Rewind traces. Use the option On Disk to use the host file, and the option In Memory (default) to use the host memory. See Figure 3–1.

Figure 3−1. Enabling Rewind



There are some advanced options for Rewind which are not configurable via Code Composer Studio Setup. Table 3−2 provides the details and values for these options.

Table 3–2. Advanced Rewind Configuration Options

Option	Value	Value in the Base Configuration File	Description
MAX_BUF_SIZE	32 bit unsigned integer >= 0x100000 (1MB)	0x800000 //8MB	For memory trace, it specifies the size of the memory used to keep the trace. For file trace, it specifies the size of the temporary buffer used for efficient file operations.
MAX_FILE_SIZE	32 bit unsigned integer >= 0x800000 (8MB)	0x40000000 //1GB	For file trace, it specifies the maximum size of the execution history in bytes. If the size of the trace reaches this value, the execution history is reset. This option is not used for memory trace.
TMP_DIR	Directory path	c:	For file trace, it specifies the directory where the simulator creates the file. This option is not used for memory trace.

Note: If the path specified as TMP_DIR in the configuration file is not writeable or there is not enough space in the disk (i.e. available space on the disk is less than the size specified through MAX_FILE_SIZE), the simulator will disable the Rewind feature immediately. In that case, the behavior will be as if ENABLE_REWIND is specified as OFF. In the current version, the simulator will not generate a error message.

To customize the simulator, modify the Base Configuration File as follows:

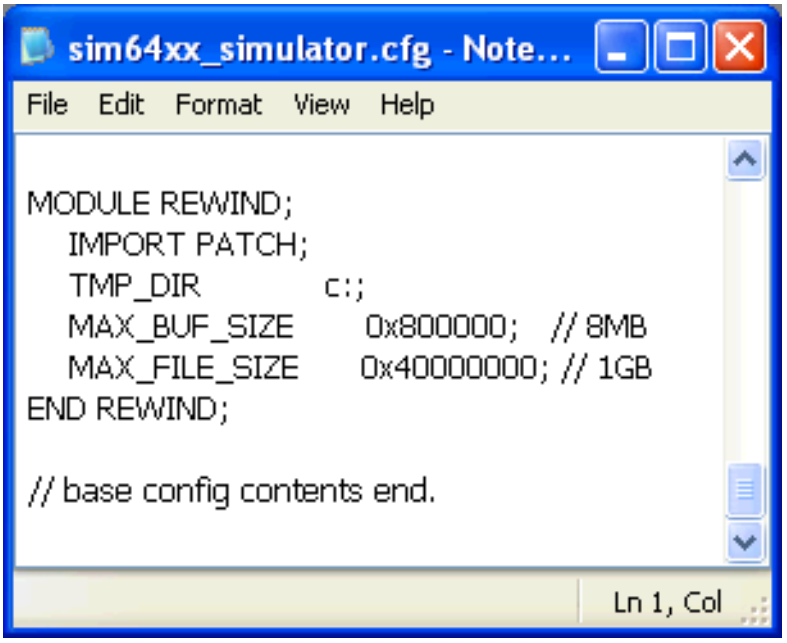
1) Go to the Code Composer Studio installation directory. For example, C:\CCStudio.

2) Go to the drivers directory inside it (e.g., C:\CCStudio\drivers).

3) Open the appropriate base configuration file (.cfg) in a text editor. (e.g., sim64xx_simulator.cfg)

4) Look for the module named REWIND.

Figure 3–2. Rewind Configuration



5) Edit this module to customize Rewind. It is advised to keep a copy of this file before making any modifications.

6) Save the changes and quit the editor.

Appendix A

Guidelines and Limitations

- On reaching the end of the execution history, the simulator stops, but it does not show any error messages. Error messages are also not shown if the trace file cannot be written due to invalid path, disk is full, or any other reason. The error messages will be shown in later versions.
- In replay mode, doing a restart will cause the program counter to rewind momentarily, but will continue from the last point on the next advance. The workaround for this is to first reset the simulator and then restart.
  - After resetting the simulator and the cycle event (see section 2.3 for picking the event usable in review mode), you will see that profile clock counter runs correctly in the forward direction. When you switch to replay mode, the counter shows an old count. The relative count remains correct afterwards. This problem does not occur if you use the Auto Reset option for the profile clock.
  - If the counter has to show negative values (for instance, during back stepping with the profile clock set to the Auto Reset), it will show the corresponding unsigned 64-bit integer. For example, -1 will show as 4294967295.
- Some of the peripheral registers will not show correct values in the replay mode.
- C55x CPU Cycle Accurate simulators support the emulation mode of execution which flushes pipeline on halt. The corresponding effects on program counter value (e.g. PC skid) are described in SPRU599. These effects will be visible even in the replay mode of the simulator. Thus, you may see PC skids in replay mode if the simulator is run in emulation mode.
- For functions that do not have enough debug information (for example, rts lib functions) in the normal forward execution, performing a Source Single-Step at that function call actually results in Step Over. However, in the rewind mode, Source Single Back-Step at that function call results in assembly back step in and not a Source Step Over.
- Boot loading: If you try to back-step through the code which is copied during boot-loading, the disassembly window shows only NOPs.