

Integrator™ Model

Version 1.0

User Guide

ARM

Integrator Model

User Guide

Copyright © 2001. All rights reserved.

Release Information

Change history

Date	Issue	Change
7 December 2001	A-01	Frist release

Proprietary Notice

Words and logos marked with ® or ™ are registered trademarks or trademarks owned by ARM Limited, except as otherwise stated below in this proprietary notice. Other brands and names mentioned herein may be the trademarks of their respective owners.

Neither the whole nor any part of the information contained in, or the product described in, this document may be adapted or reproduced in any material form except with the prior written permission of the copyright holder.

The product described in this document is subject to continuous developments and improvements. All particulars of the product and its use contained in this document are given by ARM in good faith. However, all warranties implied or expressed, including but not limited to implied warranties of merchantability, or fitness for purpose, are excluded.

This document is intended only to assist the reader in the use of the product. ARM Limited shall not be liable for any loss or damage arising from the use of any information in this document, or any error or omission in such information, or any incorrect use of the product.

Confidentiality Status

This document is Open Access. This document has no restriction on distribution.

Product Status

The information in this document is final (information on a developed product).

Web Address

<http://www.arm.com>

Contents

Integrator Model User Guide

	Preface	
	About this document	vi
	Feedback	viii
Chapter 1	Introduction	
	1.1 About Integrator Model	1-2
Chapter 2	Loading and Running the Integrator Model	
	2.1 Setting up the environment	2-2
	2.2 Choosing a core module	2-3
	2.3 Basic configuration	2-5
	2.4 Integrator Model and other ARMulator models	2-6
	2.5 Bootmonitor	2-8
Chapter 3	Differences between Integrator Model and Integrator Hardware	
	3.1 Motherboard and core module features	3-2
Chapter 4	Integrator Model Usage and Configuration	
	4.1 Host and debug setup	4-2
	4.2 Core module simulation	4-7
	4.3 Simulation of motherboard peripherals	4-9
	4.4 Simulation of additional peripherals	4-13

4.5 Performance suggestions 4-15

Glossary

Preface

This preface introduces the Integrator Model and its reference documentation. It contains the following sections:

- *About this document* on page vi
- *Feedback* on page viii.

About this document

This document is the user guide for Integrator Model Version 1.0. It contains the following:

- Introductory material and simple examples. These illustrate basic principles of using and configuring Integrator Model, and are targeted at new users.
- Reference and general user information to help you configure Integrator Model to run your software images optimally.

For more information about Integrator, see the *Integrator/AP User Guide* (ARM DUI 0098A).

Intended audience

This document has been written for engineers who need to use Integrator Model as part of the development process. It assumes that you are an experienced software developer and that you are familiar with the *ARM Developer Suite* (ADS 1.2).

Using this manual

This document is organized into the following chapters:

Chapter 1 *Introduction*

Read this chapter for an overview of the main features of Integrator Model.

Chapter 2 *Loading and Running the Integrator Model*

Read this chapter for basic information to help you start using Integrator Model.

Chapter 3 *Differences between Integrator Model and Integrator Hardware*

This chapter provides information on differences between Integrator Model and the Integrator hardware.

Chapter 4 *Integrator Model Usage and Configuration*

Read this chapter to learn how to configure and use Integrator Model.

Typographical conventions

The following typographical conventions are used in this book:

<i>italic</i>	Highlights important notes, introduces special terminology, denotes internal cross-references, and citations.
---------------	---

bold	Highlights interface elements, such as menu names. Denotes ARM processor signal names. Also used for terms in descriptive lists, where appropriate.
<code>monospace</code>	Denotes text that can be entered at the keyboard, such as commands, file and program names, and source code.
<u>monospace</u>	Denotes a permitted abbreviation for a command or option. The underlined text can be entered instead of the full command or option name.
<i>monospace italic</i>	Denotes arguments to commands and functions where the argument is to be replaced by a specific value.
monospace bold	Denotes language keywords when used outside example code.

Further reading

This section lists publications by ARM Limited.

ARM periodically provides updates and corrections to its documentation. See <http://www.arm.com> for current errata sheets and addenda.

See also the ARM Frequently Asked Questions list at: <http://www.arm.com/DevSupp/Sales+Support/faq.html>

ARM publications

This document contains information that is specific to Integrator Model. See the following documents for other relevant information:

- *Integrator/AP User Guide* (ARM DDI 0098A).
- *ADS Debug Target Guide* (ARM DUI0058D).

Feedback

ARM Limited welcomes feedback both on Integrator Model, and on the documentation.

Feedback on Integrator Model

If you have any comments or suggestions about this product, please contact your supplier giving:

- the product name
- a concise explanation of your comments.

Feedback on this document

If you have any comments on this document, please send email to errata@arm.com giving:

- the document title
- the document number
- the page number(s) to which your comments refer
- a concise explanation of your comments.

General suggestions for additions and improvements are also welcome.

Chapter 1

Introduction

This chapter provides an overview of Integrator Model Version 1.0. It contains the following section:

- *About Integrator Model* on page 1-2.

1.1 About Integrator Model

Integrator Model is a software simulation of the Integrator/AP development platform. The Integrator/AP development platform is designed for the development of applications and hardware with ARM processors.

Integrator Model is implemented as an ARMulator memory model. It inserts itself into the memory chain above the flat memory system. It is always located lower in the chain than any memory model that inserts itself below the core or cache. For more information about ARMulator memory models, see *ADS Debug Target Guide*.

For more information about Integrator hardware, see the *Integrator User Guide*.

1.1.1 System requirements

Integrator Model is an ARMulator plug-in. To run Integrator Model, you must have:

ADS 1.2 ARMulator is a component of ARM Developer Suite version 1.2.

Linux, Solaris, or Windows

Integrator Model runs on the following operating systems:

- RedHat Linux 6.2 and above
- Solaris 2.5.1 and above
- Windows NT4 and above.

Chapter 2

Loading and Running the Integrator Model

This chapter describes how to load Integrator Model. It includes the following sections:

- *Setting up the environment* on page 2-2
- *Choosing a core module* on page 2-3
- *Basic configuration* on page 2-5
- *Integrator Model and other ARMulator models* on page 2-6
- *Bootmonitor* on page 2-8.

2.1 Setting up the environment

Before loading Integrator Model, you must ensure that the following environment variables are extended as appropriate:

ARMDLL	A list of paths for ARMulator to search for dynamically loaded plugins, in this case <code>integrator.dll</code> (on Windows) or <code>integrator.so</code> (on Linux or Solaris). You must extend it to include the directory containing <code>integrator.dll</code> or <code>integrator.so</code> .
ARMCONF	A list of paths for ARMulator to search for <code>.ami</code> and <code>.dsc</code> configuration files. You must extend it to include the directories containing <code>integrator.ami</code> and <code>integrator.dsc</code> .
PATH	A standard environment variable. You must extend it to include the directory containing the <code>wxintegrator</code> executable file. If you are using Windows, you must also extend it to include the directory containing the <code>wx22_7.dll</code> file.
LD_LIBRARY_PATH	<p>A UNIX environment variable. You must extend it to include the directory containing the <code>libwx_gtk.so</code> file.</p> <p>The <code>libwx_gtk.so</code> library requires <code>glib</code> and <code>gtk</code>. <code>glib</code> and <code>gtk</code> contain:</p> <ul style="list-style-type: none">• <code>libgtk.so</code>• <code>libgdk.so</code>• <code>libglib.so</code>• <code>libgmodule.so</code>• <code>libgthread.so</code> <p>Supported Linux systems include <code>glib</code> and <code>gtk</code>. You can also download them from http://www.gtk.org/download/.</p>

2.2 Choosing a core module

When you specify the processor, or CPU, to the debugger, you are actually specifying the whole combination of processor and a set of peripherals.

You must choose the core module you want to simulate. Predefined Integrator Model processors are prefixed CM. They consist of the named processor with its associated core module, and optionally an Integrator/AP motherboard. Available core modules are:

- CM7TDMI
- CM720T
- CM740T
- CM920T
- CM940T
- CM966ES
- CM1020T.

If you use one of these predefined modules, you do not need to remove incompatible peripherals (as in *Integrator Model and other ARMulator models* on page 2-6), or set the core module type (as in *INTE_CMO_TYPE* on page 4-8), because this is configured automatically.

Alternatively, you can add an instantiation of Integrator Model to the set of peripherals that are automatically loaded by the standard ARM processor types, ARM7TDMI for example. In this case, you must remove incompatible peripherals (see *Integrator Model and other ARMulator models* on page 2-6) and ensure that the Integrator Model core type matches the CPU model in use (see *INTE_CMO_TYPE* on page 4-8).

2.2.1 Running from the command line, using armsd

Use the `-cpu` argument when running `armsd`, for example:

```
armsd -cpu CM920T
```

2.2.2 Running from a graphical debugger such as AXD

For AXD, do the following:

1. Select **Options Menu** → **Configure Target**.
2. choose `armulate.dll`.
3. click in the **Configure Debugger** dialogue box.
4. select the **Processor: Variant** pop-up menu.
5. choose the appropriate processor type. For a predefined Integrator Model core module, this is one of the options with a CM prefix.

For other graphical debuggers, follow a corresponding procedure as far as possible.

2.2.3 Startup

When Integrator Model is loaded, it appears in the ARMulator startup banner:

```
ARMulator ADS1.2 [Build 805]  
CM920T, 16KB I-cache, 16KB D-cache, (Physical memory, BIU), Littleendian, Debug  
Comms Channel, 4GB, Integrator, Semihosting  
Loading Integrator rom and flash images.....
```

Integrator Model takes some time to load ROM and Flash images into memory. This is indicated by a row of dots after the startup banner.

2.3 Basic configuration

The following files must be placed on the path pointed to by the environment variable `ARMCONF`:

- `integrator.dsc`
- `integrator.ami`

`integrator.dsc` sets up various constant and default values. You must not edit `integrator.dsc`.

`integrator.ami` is the configuration file for Integrator Model. You can edit `integrator.ami` in an ordinary text editor.

2.3.1 `integrator.ami`

In `integrator.ami`, configuration options take the form:

`tag_name=value`

For details of tags and appropriate values, see Chapter 4 *Integrator Model Usage and Configuration*.

`integrator.ami` can contain comments. A semicolon marks the beginning of a comment. The comment ends at the next line break. Do not place a comment on the same line as a configuration option.

2.4 Integrator Model and other ARMulator models

If you specify a predefined Integrator Model core module and CPU type when you start ARMulator, the Integrator Model is loaded with its standard peripherals. Other peripherals and models are not included.

You can use the Integrator Model memory model with other ARM processor models. If you do this, you must be aware of some compatibility issues. These are described in this section.

If you use an Integrator Model CPU type (a CPU with a CM- prefix), ARMulator disables unrequired models. You can instantiate other models as necessary by including a definition in the Integrator_Peripherals block in the integrator.ami file.

For more information about ARMulator configuration, see *ADS Debug Target Guide*.

2.4.1 Pagetables

The default pagetables model writes into areas of memory that do not exist as standard RAM on Integrator Model.

———— **Note** ————

Do not attempt to use the pagetables model, in its default configuration, with Integrator Model.

—————

2.4.2 Semihosting

Semihosting provides code running on an ARM target use of facilities on a host computer that is running an ARM debugger. Examples of such facilities include the keyboard input, screen output, and disk I/O.

You can run Integrator Model with or without semihosting.

- For some software, for example testsuite, you must have semihosting enabled.
- For some software, you must not have semihosting enabled.
- For some software, it does not matter whether you have semihosting enabled or not. For example, a program might output text using semihosting if semihosting is available, but be able to use UART0 otherwise.

2.4.3 Interrupt controller, intc

Integrator Model contains its own implementation of `intc`, the interrupt controller. The default `intc` must be disabled if you use the Integrator Model memory model outside the predefined `CMxxx` processor definitions.

Other models can access the Integrator Model interrupt controller using the usual `ARMulif_GetInterruptController` interface, see *ADS Debug Target Guide*. They must only use interrupt signals 9 to 12. These are allocated for use by logic modules.

2.5 Bootmonitor

When you first start Integrator Model, you must usually run Bootmonitor manually. Bootmonitor sets up a memory state. Many program images rely on this memory state.

To run Bootmonitor do the following:

1. Ensure that switches S1-1 and S1-4 are both **SET** in the Integrator DIL S1 window.
2. In your chosen debugger, ensure that PC is set to zero.
3. Start executing.

When Bootmonitor starts, it displays a startup banner:

```
ARM bootPROM [Version 1.2] Rebuilt on Aug  3 2000 at 11:59:34
Running on a Integrator Evaluation Board
Board Revision V1.0, ARM7TDMI Processor
Memory Size is 128MBytes, Flash Size is 32MBytes
Copyright (c) ARM Limited 1999 - 2000. All rights reserved.
Board designed by ARM Limited
Hardware support provided at http://www.arm.com/
For help on the available commands type ? or h
```

This is followed by the prompt:

```
boot Monitor>
```

If S1-4 is **CLEAR**, Integrator Model continues to boot from the default image in the Flash RAM, instead of displaying the prompt.

Chapter 3

Differences between Integrator Model and Integrator Hardware

Most operations work identically on Integrator hardware and on Intergrator Model. This chapter describes the exceptions. It contains the following section:

- *Motherboard and core module features* on page 3-2.

3.1 Motherboard and core module features

This section describes the differences between the Integrator Model and the Integrator hardware.

3.1.1 Local RAM, ROM, and peripherals

The ROM area of memory is a flash chip in hardware. In Integrator model it is implemented as read-only memory for normal accesses. The debugger can write to it.

3.1.2 PCI space

You can configure Integrator Model to include a PCI video card in one of the slots of the PCI bus. The card is a 2D simple framebuffer model based on a subset of the registers of the Imagination Neon250 graphics chipset.

This release of Integrator Model does not support any other PCI cards.

See *INTE_DISPLAY* on page 4-14 for more information.

3.1.3 Core module alias memory

Integrator hardware supports up to four core modules. Integrator Model only supports one core module.

In Integrator hardware, the area of address space from 0x90000000 to 0xBFFFFFFF holds the local RAM for each module. This area of address space is not implemented in Integrator Model.

Read and write accesses to this area of memory are undefined. They cause a message to be sent to the console if warnings are enabled, see *INTE_WARNINGS* on page 4-2.

3.1.4 Logic modules

Integrator Model does not explicitly support logic modules. However, using ADS 1.2 you can load any ARMulator peripheral model into the address range 0xC0000000 to 0xFFFFFFFF. Accesses to these addresses are forwarded to the appropriate peripheral.

3.1.5 PS/2 keyboard

Integrator Model has a keyboard model attached to KMI port 0. The keyboard model only supports PS/2 scancode sets 1 and 2 codes.

To send host keyboard activity to the keyboard model, ensure that the Integrator Model PS/2 window has the input focus.

3.1.6 PS/2 mouse

Integrator Model has a mouse model attached to KMI port 1.

To move the simulated mouse:

1. Click.
2. Hold.
3. Drag the pointer within the upper rectangle of the Integrator Model PS/2 window.

To click the simulated mouse buttons, left-click in one of the lower rectangles of the Integrator Model PS/2 window. To toggle holding down the simulated mouse buttons, right-click in one of the lower rectangles of the Integrator Model PS/2 window.

Chapter 4

Integrator Model Usage and Configuration

This chapter describes the methods that you can use to debug your Integrator Model models. It contains the following sections:

- *Host and debug setup* on page 4-2
- *Core module simulation* on page 4-7
- *Simulation of motherboard peripherals* on page 4-9
- *Simulation of additional peripherals* on page 4-13
- *Performance suggestions* on page 4-15.

4.1 Host and debug setup

This section describes tags in an instantiation of the Integrator model. These tags are usually in the `integrator.ami` file.

4.1.1 INTE_FLASHIMAGE

Use the `INTE_FLASHIMAGE` tag to specify the file or pathname for the flash RAM image. The flash RAM image is a gzipped file containing the image for locations `0x24000000` to `0x27FFFFFF`.

The flash image is automatically saved on exit if any changes have been made to it during an execution run. The previous version of the file is renamed for backup by appending a tilde (~) to the filename.

4.1.2 INTE_ROMIMAGE

Use the `INTE_ROMIMAGE` tag to specify the file or pathname for the ROM image. The ROM image is a plain binary file containing the image for locations `0x20000000` to `0x2003FFFF`.

4.1.3 INTE_WARNINGS

Set `INTE_WARNINGS` to `True` to make Integrator Model display a warning to the console when an unsupported event occurs. Unsupported events include:

- attempts to access certain addresses, such as:
 - in peripheral areas, addresses that do not correspond to a register
 - in peripheral areas, reading from a write-only register
 - in peripheral areas, writing to a read-only register
 - in the PCI address range, accesses to an address not covered by V360 bridge windows
- attempts to write to nonwritable areas of memory, such as ROM or Flash
- unexpected or unrecognized commands to peripherals, such as the PS/2 mouse or PS/2 keyboard
- an error during an attempt to write a file.

4.1.4 INTE_BREAKONWARNINGS

Set `INTE_BREAKONWARNINGS` to `True` if you want control to return to the debugger when a warning occurs during execution, see *INTE_WARNINGS* on page 4-2 for a list of possible warnings.

4.1.5 INTE_DEBUGFILE

Use the INTE_DEBUGFILE tag to specify the file or pathname for the Integrator Model output log file. The file contains details of Integrator Model register reads and writes, other events, and their effects.

4.1.6 INTE_DEBUGLEVEL

Use the INTE_DEBUGLEVEL tag to specify the amount of information you want to appear in the output log file. Larger numbers give more verbose output. This can impair the speed performance of Integrator Model.

INTE_DEBUGLEVEL must be given one of the following values:

- 0** Nothing is included in the log.
- 2** Startup configuration information and major changes are included in the log.
- 4** In addition, writes to registers are logged.
- 6** In addition, raised interrupts are logged.
- 8** In addition, reads from important registers are logged.
- 10** All useful information is logged.

You can use intermediate levels to log more or less important information within these categories.

You can override the default debug level set by the INTE_DEBUGLEVEL tag for some specific areas within Integrator Model. You can insert any of the following tags into `integrator.ami`, giving them any of the INTE_DEBUGLEVEL values:

INTE_DEBUGLEVEL_CM

Controls log entries relating to the core module.

INTE_DEBUGLEVEL_UART

Controls log entries relating to the UART input and output.

INTE_DEBUGLEVEL_FLASH

Controls log entries for Flash RAM and ROM mode changes and writes.

INTE_DEBUGLEVEL_TIMER

Controls log entries for on-board timers and the real-time clock.

INTE_DEBUGLEVEL_PCI

Controls log entries for the PCI bridge control chip and the PCI bus.

INTE_DEBUGLEVEL_PCICard

Controls log entries for a simulated card on the PCI bus.

INTE_DEBUGLEVEL_GUI

Controls log entries for the graphical output display windows, or host windowing system.

INTE_DEBUGLEVEL_MISC

Controls log entries for other peripherals or events.

For example, you might want to reduce the amount of information logged about the UART port. This is likely to be large, because each character of text output on the UART involves at least one register write.

Example

```
INTE_DEBUGLEVEL=6
INTE_DEBUGLEVEL_UART=3
```

4.1.7 INTE_LEDWINDOW

Use the INTE_LEDWINDOW tag to specify the size and location of the window showing the state of the Integrator motherboard LEDs, the core module LED, and the motherboard alphanumeric display. The tag definition must be a space-separated list of four numbers:

- the x-coordinate of the window
- the y-coordinate of the window
- the width of the window
- the height of the window.

Example

```
INTE_LEDWINDOW=50 50 478 83
```

4.1.8 INTE_ALPHA_OFF_RGB

Use the INTE_ALPHA_OFF_RGB tag to specify the colour of the Alphanumeric display segments when they are not lit. The tag definition must be a space-separated list of three numbers, the red, green and blue components of the required colour, each in the range 0-255.

Example

```
INTE_ALPHA_OFF_RGB=120 100 100
```

4.1.9 INTE_ALPHA_ON_RGB

Use the INTE_ALPHA_ON_RGB tag to specify the colour of the Alphanumeric display segments when they are lit. The tag definition must be a space-separated list of three numbers, the red, green, and blue components of the required colour, each in the range 0-255.

Example

```
INTE_ALPHA_ON_RGB=255 160 140
```

4.1.10 INTE_DILWINDOW

Use the INTE_DILWINDOW tag to specify the size and location of the window showing the state of the four DIL switches on the Integrator motherboard. You can also control these switches from this window. The tag definition must be a space-separated list of four numbers:

- the x-coordinate of the window
- the y-coordinate of the window
- the width of the window
- the height of the window.

Example

```
INTE_DILWINDOW=528 50 170 83
```

4.1.11 INTE_PS2WINDOW

Use the INTE_PS2WINDOW tag to specify the size and location of the window you can use to capture PS/2 keyboard and mouse inputs. The tag definition must be a space-separated list of four numbers:

- the x-coordinate of the window
- the y-coordinate of the window
- the width of the window
- the height of the window.

Example

```
INTE_PS2WINDOW=698 50 208 178
```

4.1.12 INTE_GPIOWINDOW

Use the INTE_GPIOWINDOW tag to specify the size and location of the general purpose I/O bus window.

This window displays the state of the Integrator LEDs that show the state of any output pins, and has switches that control the state of any input pins. The direction of each pin is controlled from Integrator software by writing to the GPIO_DIRN register at 0x1B000008.

The tag definition must be a space-separated list of four numbers:

- the x-coordinate of the window
- the y-coordinate of the window
- the width of the window
- the height of the window.

Example

```
INTE_GPIOWINDOW=50 132 648 96
```

4.1.13 INTE_VGAWINDOW

Use the INTE_VGAWINDOW tag to specify the size and location of the window that displays the contents of the 2D framebuffer VGA card, if present. The tag definition must be a space-separated list of four numbers:

- the x-coordinate of the window
- the y-coordinate of the window
- the width of the window
- the height of the window.

Example

```
INTE_VGAWINDOW=50 230 640 480
```

4.1.14 INTE_CLOSEALL

Set the INTE_CLOSEALL tag to True if you want all the Integrator Model windows to close when you hit the close button in any of the Integrator Model windows.

4.2 Core module simulation

This section lists the tags in `integrator.amt` that you can edit to configure Integrator Model for the particular Integrator/AP board you want to simulate.

It contains the following subsections:

- *INTE_MOBOARD* on page 4-7
- *INTE_CMO_TYPE* on page 4-8
- *INTE_CMO_SDRAMSIZE* on page 4-8
- *INTE_SYSCLK* on page 4-8.

4.2.1 INTE_MOBOARD

Use the `INTE_MOBOARD` tag to specify whether the core module is attached to an Integrator/AP motherboard. You can use any of the following values:

- | | |
|----------|--|
| 0 | The core module is running in standalone mode. It is not attached to any motherboard. |
| 1 | The core module is attached to an early motherboard, revision A build 55. The value returned by register <code>SC_ID</code> is <code>0x41001550</code> . There is no <code>EBI_LOCK</code> register. |
| 2 | The core module is attached to a later motherboard, revision B build 23. The value returned by register <code>SC_ID</code> is <code>0x41012231</code> . There is an <code>EBI_LOCK</code> register. This is the default. |

Example

```
INTE_MOBOARD=2
```

4.2.2 INTE_CMO_TYPE

The INTE_CMO_TYPE tag is defined in the core module processor definitions in `integrator.dsc`. You only need to use it if you are instantiating the Integrator Model without using the predefined core modules (see *Choosing a core module* on page 2-3).

Use the INTE_CMO_TYPE to specify which model of Core Model you want to use. It controls the value returned by CM_ID and CM_PROC, and enables additional registers in later core module versions.

The value is a numeric value that should match the processor in use. For example:

700	represents an ARM7TDMI
720	represents an ARM720T
940	represents an ARM940T.

4.2.3 INTE_CM0_SDRAMSIZE

Use the INTE_CM0_SDRAMSIZE tag to specify the size, in megabytes, of the SDRAM DIMM attached to the first core module. The maximum size is 256MB. The default value is 64MB.

Example

```
INTE_CM0_SDRAMSIZE=128
```

4.2.4 INTE_SYCLK

Use the INTE_SYCLK tag to specify the speed of the Integrator motherboard system bus. This affects the rate of Timer0, and the initial value of the SC_OSC register. The default is 24MHz.

You can set the assumed processor clock speed using the `-clock` argument on the `armsd` command line, or by placing a CPUSPEED tag in a `.ami` file, see *ADS Debug Target Guide*.

You can also set speeds from within the model by writing to the SC_OSC and CM_OSC registers.

Example

```
INTE_SYCLK=24MHz
```

4.3 Simulation of motherboard peripherals

This section lists the tags in `integrator.amf` that you can edit to configure Integrator Model for the particular motherboard peripherals you want to simulate.

It contains the following subsections:

- *INTE_DIL_SL1* on page 4-9
- *INTE_UART* on page 4-10
- *INTE_UART_TTYFEATURES* on page 4-11
- *INTE_UART_NOTELNET* on page 4-11
- *INTE_UNTIMED_UARTS* on page 4-11
- *INTE_UART_PERIOD* on page 4-11
- *INTE_TIMERMULTIPLY* on page 4-12.

4.3.1 INTE_DIL_SL1

Use the `INTE_DIL_SL1` tag to set up the initial state of the four DIL switches on the motherboard. The value is an integer that corresponds to a four-bit binary number representing the state of the four switches. This is the same as the way the value is returned from the `LED_SWITCH` register.

For example, `INTE_DIL_SL1=9` sets switches S1-1 and S1-4 to True. This makes Integrator Model boot from ROM instead of from flash, and makes `Bootmonitor` stop at the prompt.

4.3.2 INTE_UART

There are four INTE_UART tags:

- INTE_UART0in
- INTE_UART0out
- INTE_UART1in
- INTE_UART1out.

Use these tags to specify the targets to which UART inputs and outputs are directed.

On UNIX, the target can be, for example, a named pipe, or a tty.

On Windows and UNIX, the target can be a filepath or a socket.

Sockets

On Windows and UNIX, the target can be a socket, for example:

`INTE_UART0in=SOCKET:31337`

where 31337 is an arbitrary port number. This opens a socket on the local machine with this port number.

———— **Note** ————

If you attach a UART to a socket, the input and output are both connected to the socket. The value given to INTE_UART0out, or INTE_UART1out as appropriate, is ignored.

Integrator Model also launches a telnet process and connects this to the socket. If you want to attach a different process to the socket you can prevent Integrator Model from launching telnet.

See *INTE_UART_TTYFEATURES* on page 4-11 for details of how to make the target behave like a terminal.

———— **Warning** ————

Integrator Model blocks at start up if there is no connection to the socket.

Do not attach both UART inputs to the same target. If you do this, each input reads characters at arbitrary intervals.

4.3.3 INTE_UART_TTYFEATURES

Set `INTE_UART_TTYFEATURES=True` if you want a target connected to a UART to behave like a terminal. If you do this, the target:

- echos settings
- waits for a newline character before forwarding any characters to the UART.

Do not set this tag if you are opening the default terminal `/dev/tty`.

4.3.4 INTE_UART_NOTELNET

Set `INTE_NOTELNET=True` if you want to prevent Integrator Model connecting a telnet process to a socket. See *INTE_UART* on page 4-10.

4.3.5 INTE_UNTIMED_UARTS

Set `INTE_UNTIMED_UARTS=True` if you want any data sent to the UART outputs to be dealt with immediately, with no delay.

This gives faster simulation, but results in inaccurate simulation. It is set to `False` by default.

4.3.6 INTE_UART_PERIOD

Use the `INTE_UART_PERIOD` tag to specify the period, in core clock cycles, between polling for consecutive characters on the UART inputs, and between outputting consecutive characters from the UART outputs.

For example, for a processor running at 20MHz, a 56 kbit/s modem is equivalent to a `INTE_UART_PERIOD` tag value of about 3000.

Note

This is ignored for outputs if `INTE_UNTIMED_UARTS` is set to `True`. See *INTE_UNTIMED_UARTS* on page 4-11.

A shorter period gives an apparently more responsive system, but can increase the simulation overhead. The UART passes the test suite with a value of about 500 or more. With values from 500 down to 300, the UART has a diminishing probability of passing the test suite. This depends on exactly when, within the period, the test suite happens to send and expects to receive data.

4.3.7 INTE_TIMERMULTIPLY

Use the INTE_TIMER_MULTIPLY tag to set all motherboard timers to run at a multiple of their normal frequency. For example, to halve the period (double the frequency) of all the timers, you can set:

INTE_TIMER_MULTIPLY=2

4.4 Simulation of additional peripherals

This section lists the tags in `integrator.ami` that you can edit to configure Integrator Model for the particular external peripherals you want to simulate.

It contains the following subsections:

- `INTE_MOUSE_PERIOD` on page 4-13
- `INTE_PCI_VGACARD` on page 4-13
- `INTE_DISPLAY` on page 4-14
- `INTE_PCI_ETHERNETCARD` on page 4-14.

4.4.1 INTE_MOUSE_PERIOD

Use the `INTE_MOUSE_PERIOD` tag to specify the period, in clock cycles, between consecutive data reports from the PS/2 mouse.

Do not attempt to configure realistic periods in clock cycles. For example, for a 20MHz processor, a 40Hz signal has a period of 500 000 clock cycles. This is very slow in simulation. See also *Performance suggestions* on page 4-15.

Integrator Model ignores PS/2 commands to change the report rate.

4.4.2 INTE_PCI_VGACARD

If you want to attach a video card to the PCI bus, set the `INTE_PCI_VGACARD` tag to 0, 1, or 2. Use values 0, 1, or 2 to specify which physical slot the card is simulated in. The card is a 2D framebuffer card based on a subset of the registers of the Imagination Neon 250 chipset.

If you do not want a video card attached to the PCI bus, comment this line out, by placing a semicolon at the beginning of the line. This is the default.

Examples

```
INTE_PCI_VGACARD=2 ; Video card in slot 2.  
; INTE_PCI_VGACARD=1 ; No video card (commented out)
```

4.4.3 INTE_DISPLAY

There are three INTE_DISPLAY tags:

INTE_DISPLAYX Locks the video card framebuffer into a particular x resolution.

INTE_DISPLAYY Locks the video card framebuffer into a particular y resolution.

INTE_DISPLAY Locks the video card framebuffer into a particular bitdepth.

This is only necessary if your simulated software uses a video driver that attempts to change the framebuffer resolution in a way that Integrator Model does not support, for example by using registers that are not supported. If this occurs, contact ARM with full details of the video driver.

Examples

```
INTE_DISPLAYX=1024  
INTE_DISPLAYY=768  
INTE_DISPLAYBITS=24
```

4.4.4 INTE_PCI_ETHERNETCARD

Use the INTE_PCI_ETHERNETCARD tag if the operating system you are using requires an ethernet card during its configuration phase. Use values 0, 1, or 2 to specify which physical slot the card is simulated in.

This is not a complete implementation of an ethernet card. The only function it has is to identify itself as an ethernet device for an operating system to detect during configuration.

Do not use this tag unless the card is required for your operating system to boot. By default this tag is not present.

Examples

```
INTE_PCI_ETHERNETCARD=1 ; Ethernet card in slot 2 (non-functional).  
; INTE_PCI_ETHERNETCARD=0 ; No ethernet card (commented out)
```

4.5 Performance suggestions

Use the following tags with care:

- *INTE_UART_PERIOD* on page 4-11
- *INTE_TIMERMULTIPLY* on page 4-12
- *INTE_MOUSE_PERIOD* on page 4-13.

Each of them:

- can sometimes increase the apparent running speed of your image
- can sometimes decrease the apparent running speed of your image.

If the image spends a large proportion of its time waiting for a real-time delay to occur, the image appears to run more quickly. This happens when UART and mouse periods are decreased, or timer multiply is increased.

If the image spends a large proportion of its time servicing interrupts, the image appears to run more slowly. This happens when UART and mouse periods are increased, or timer multiply is decreased.

Switching on untimed UARTS can increase speed by a very large factor if the operating system outputs heavily to a UART, see *INTE_UNTIMED_UARTS* on page 4-11.

Note

Each of these features can cause errors if you make assumptions about the timings associated with the peripherals. If unexplained errors occur, try resetting these tags to their normal values.

Reducing the verbosity of the debug log can increase execution speed, see *INTE_DEBUGLEVEL* on page 4-3.

Glossary

This glossary contains terms that are relevant to Integrator Model.

ADS *See* ARM Developer Suite.

ARM Developer Suite A suite of applications, together with supporting documentation and examples, that enable you to write and debug applications for the ARM family of RISC processors.

ARM eXtended Debugger The ARM eXtended Debugger (AXD) is the latest debugger software from ARM that enables you to make use of a debug agent in order to examine and control the execution of software running on a debug target. AXD is supplied in both Windows and UNIX versions.

armsd The ARM Symbolic Debugger (armsd) is an interactive source-level debugger providing high-level debugging support for languages such as C, and low-level support for assembly language. It is a command-line debugger that runs on all supported platforms.

AXD *See* ARM eXtended Debugger.

Bus A bounded array of wires that can be treated as a unit. The array has both upper and lower bounds, although both bounds must be either positive or zero.

Debugger	An application that monitors and controls the execution of a second application. Usually used to find errors in the application program flow.
Host	A computer which provides data and other services to another computer.
Image	<p>An executable file which has been loaded onto a processor for execution.</p> <p>A binary execution file loaded onto a processor and given a thread of execution. An image can have multiple threads. An image is related to the processor on which its default thread runs.</p>
Interrupt	A change in the normal processing sequence of an application caused by, for example, an external signal.
Revision	A unique version of a model.
Semihosting	A mechanism whereby the target communicates I/O requests made in the application code to the host system, rather attempting to support the I/O itself.
Target	<p>The simulated processor on which the target application is running.</p> <p>The fundamental object in any debugging session. The basis of the debugging system. The environment in which the target software will run.</p>

Index

The items in this index are listed in alphabetical order, with symbols and numerics appearing at the end. The references given are to page numbers.

A

ARMCONF 2-2, 2-5
ARMDLL 2-2

B

Bootmonitor 2-8, 4-9

C

CM_ID 4-8
CM_OSC 4-8
CM_PROC 4-8
Core module 2-3, 4-7
CPUSPEED 4-8

D

Debug log file 4-3, 4-15

DIL switches 4-5, 4-9
DIMM 4-8
Disk 2-6
Display 4-14

E

Environment variables 2-2
Errors 4-15
Ethernet 4-14

F

Flash 3-2, 4-2

I

intc 2-7
integrator.ami 2-5, 4-2
integrator.dsc 2-5

Interrupt controller 2-7
INTE_ALPHA_OFF_RGB 4-4
INTE_ALPHA_ON_RGB 4-5
INTE_BREAKONWARNINGS 4-2
INTE_CLOSEALL 4-6
INTE_CMO_TYPE 4-8
INTE_CMO_SDRAMSIZE 4-8
INTE_DEBUGFILE 4-3
INTE_DEBUGLEVEL 4-3
INTE_DILWINDOW 4-5
INTE_DIL_SL1 4-9
INTE_DISPLAY 4-14
INTE_FLASHIMAGE 4-2
INTE_GPIOWINDOW 4-6
INTE_LEDWINDOW 4-4
INTE_MOBOARD 4-7
INTE_MOUSE_PERIOD 4-13
INTE_NOTELNET 4-11
INTE_PCI_ETHERNETCARD 4-14
INTE_PCI_VGACARD 4-13
INTE_PS2WINDOW 4-5
INTE_ROMIMAGE 4-2
INTE_SYSCLOCK 4-8

INTE_TIMER_MULTIPLY 4-12
INTE_UART 4-10
INTE_UART_PERIOD 4-11
INTE_UART_TTYFEATURES 4-11
INTE_UNTIMED_UARTS 4-11
INTE_VGAWINDOW 4-6
INTE_WARNINGS 4-2
I/O bus window 4-6

K

Keyboard 2-6, 3-2
Keyboard window 4-5

L

LEDs 4-6
LED_SWITCH 4-9
Local RAM 3-2
Log file 4-3, 4-15
Logic modules 3-2

M

Memory model 1-2
Memory state 2-8
Motherboard 4-7
Mouse 3-3, 4-13
Mouse window 4-5

O

Output log file 4-3, 4-15

P

Pagatables 2-6
PCI bus 4-13
PCI space 3-2
Performance 4-15
Peripherals 4-9, 4-13
Pipes 4-10
Ports 4-10
PS/2 keyboard 3-2

PS/2 mouse 3-3, 4-13

R

RAM 3-2, 4-8
ROM 3-2

S

Screen 2-6
SC_OSC 4-8
SDRAM 4-8
Semihosting 2-6
Sockets 4-10
Switches 2-8, 4-5, 4-9

T

Telnet 4-11
Terminal 4-11
TTY 4-10, 4-11

U

UARTs 4-10

V

Verbosity 4-3, 4-15
VGA card 4-6, 4-13
Video card 3-2, 4-6, 4-13
Video driver 4-14