

# Mali™ GPU Offline Shader Compiler

Version: 2.2

## User Guide



# Mali GPU Offline Shader Compiler

## User Guide

Copyright © 2009 ARM. All rights reserved.

### Release Information

The following changes have been made to this book.

#### Change history

Date	Issue	Confidentiality	Change
14 October 2009	A	Non-Confidential	First release for v2.2

### Proprietary Notice

Words and logos marked with ® or ™ are registered trademarks or trademarks of ARM in the EU and other countries, 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 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.

Where the term ARM is used it means “ARM or any of its subsidiaries as appropriate”.

### Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by ARM and the party that ARM delivered this document to.

Unrestricted Access is an ARM internal classification.

### Product Status

The information in this document is final, that is for a developed product.

### Web Address

<http://www.arm.com>

# Contents

## Mali GPU Offline Shader Compiler User Guide

	<b>Preface</b>	
	About this book .....	x
	Feedback .....	xiii
<b>Chapter 1</b>	<b>Introduction</b>	
	1.1 About the Mali GPU Offline Shader Compiler .....	1-2
<b>Chapter 2</b>	<b>Installation</b>	
	2.1 Installing the Offline Shader Compiler on Microsoft Windows .....	2-2
	2.2 Installing the Offline Shader Compiler on Linux .....	2-3
<b>Chapter 3</b>	<b>The Offline Shader Compiler</b>	
	3.1 General procedure for compiling files .....	3-2
	3.2 Testing the shader .....	3-4
	<b>Glossary</b>	



# List of Tables

## **Mali GPU Offline Shader Compiler User Guide**

Change history ..... ii



# List of Figures

## **Mali GPU Offline Shader Compiler User Guide**



# Preface

This preface introduces the *Mali GPU Offline Shader Compiler User Guide*. It contains the following sections:

- *About this book* on page x
- *Feedback* on page xiii.

## About this book

This is the *Mali GPU Offline Shader Compiler User Guide*. It provides guidelines for using the Mali Developer Tools to assist in the development of applications for standalone 2D and 3D graphics applications. This book is part of a suite belonging to the Mali Developer Tools.

## Intended audience

This guide is written for system integrators and software developers who are writing OpenGL ES or OpenVG applications on a desktop workstation, using the Windows XP or Linux operating system, and want to write applications for systems including a Mali GPU.

## Using this book

This book is organized into the following chapters:

### **Chapter 1** *Introduction*

Read this for an introduction to the Offline Shader Compiler.

### **Chapter 2** *Installation*

Read this to install the software.

### **Chapter 3** *The Offline Shader Compiler*

Read this for information how to use the Offline Shader Compiler.

**Glossary** Read this for definitions of terms used in this book.

## Typographical Conventions

The typographical conventions are:

<i>italic</i>	Highlights important notes, introduces special terminology, denotes internal cross-references, and citations.
<b>bold</b>	Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.
monospace	Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.

<code>monospace</code>	Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
<code>monospace italic</code>	Denotes arguments to monospace text where the argument is to be replaced by a specific value.
<code>monospace bold</code>	Denotes language keywords when used outside example code.
<code>&lt; and &gt;</code>	Enclose replaceable terms for assembler syntax where they appear in code or code fragments. For example: MRC p15, 0 <Rd.>, <Can>, <Ca1m>, <Opcode_2>

## Additional reading

This section lists publications by ARM and by third parties.

See Inefficient, <http://infocenter.arm.com>, for access to ARM documentation.

### ARM publications

This guide contains information that is specific to the Mali Developer Tools. See the following documents for other relevant information:

- *Mali GPU Development Tools Technical Overview* (ARM DUI 0501)
- *Mali GPU Performance Analysis Tool User Guide* (ARM DUI 0502)
- *Mali GPU Texture Compression Tool User Guide* (ARM DUI 0503)
- *Mali GPU Shader Development Studio User Guide* (ARM DUI 0504)
- *Mali GPU Demo Engine User Guide* (ARM DUI 0505)
- *OpenGL ES 1.1 Emulator User Guide* (ARM DUI 0506)
- *Mali GPU Mali Binary Asset Exporter User Guide* (ARM DUI 0507)
- *Mali GPU Shader Library User Guide* (ARM DUI 0510)
- *OpenGL ES 2.0 Emulator User Guide* (ARM DUI 0511).

### Other publications

This section lists relevant documents published by third parties:

- *OpenGL ES 1.1 Specification* at <http://www.khronos.org>.
- *OpenGL ES 2.0 Specification* at <http://www.khronos.org>.
- *OpenGL ES Shading Language Specification* at <http://www.khronos.org>.
- *OpenVG 1.1 Specification* at <http://www.khronos.org>.

- *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 2* (5th Edition, 2005), Addison-Wesley Professional. ISBN 0-321-33573-2.
- *OpenGL Shading Language* (2nd Edition, 2006), Addison-Wesley Professional. ISBN 0-321-33489-2.

## Feedback

ARM welcomes feedback on this product and its documentation.

### Feedback on this product

If you have any comments or suggestions about this product then contact [malidevelopers@arm.com](mailto:malidevelopers@arm.com) and give:

- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

### Feedback on content

If you have comments on content then send an e-mail to [errata@arm.com](mailto:errata@arm.com). Give:

- the title
- the number, ARM DUI 0513A
- the page numbers to which your comments apply
- a concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.



# Chapter 1

## Introduction

This chapter describes how to use the Mali GPU Offline Shader Compiler. It contains the following section:

It contains the following sections:

- *About the Mali GPU Offline Shader Compiler* on page 1-2

## 1.1 About the Mali GPU Offline Shader Compiler

The Mali GPU Offline Shader Compiler is a command line tool that translates vertex shaders and fragment shaders written in the *OpenGL ES Shading Language* (ESSL) into binary vertex shaders and binary fragment shaders that you can link and run on the Mali GPU.

The resulting binary file must be moved to the target platform, read in by the application and given as parameter in a call to the appropriate OpenGL ES 2.0 function.

Using the Mali GPU Offline Shader Compiler is optional. You can use the compiler to provide binary shaders and therefore reduce shader load-time on the GPU. However, using compiled shaders also reduces portability, because a particular binary shader can only be used on one type of GPU.

Alternatively, the On-target Shader Compiler enables the OpenGL ES 2.0 drivers to compile OpenGL ESSL shader code dynamically when running applications on a target platform.

Using the Mali GPU Offline Shader Compiler, you can ship 3D content with pre-compiled binary shaders instead of the OpenGL ESSL source code that is required when you use the On-target Shader Compiler.

You can use the Mali GPU Offline Shader Compiler to:

- Pre-compile shaders into binary code that you can distribute with your application. The binary code is then passed to the OpenGL ES 2.0 API as an alternative to shader source code.
- Assist software development, by checking that shaders compile properly without having to pass them through an OpenGL ES 2.0 application.
- Optimize your shaders, by collecting feedback about the number of cycles each execution of the shader takes when you run it on the GPU.

The Mali GPU Offline Shader Compiler runs on Windows XP, Windows Vista, and Red Hat Enterprise Linux, Release 4.

# Chapter 2

## Installation

This chapter provides information about installing the Offline Shader Compiler Tool. These procedure is described in:

- *Installing the Offline Shader Compiler on Microsoft Windows* on page 2-2
- *Installing the Offline Shader Compiler on Linux* on page 2-3.

## 2.1 Installing the Offline Shader Compiler on Microsoft Windows

This section describes how to install the Offline Shader Compiler on Microsoft Windows. It contains the following sections:

- *Installation requirements for the Offline Shader Compiler on Microsoft Windows*
- *Installation procedure for the Offline Shader Compiler on Microsoft Windows.*

———— **Note** —————

The Offline Shader Compiler has been tested successfully on a 32-bit computer.

---

### 2.1.1 Installation requirements for the Offline Shader Compiler on Microsoft Windows

To install the Offline Shader Compiler on Microsoft Windows, you require:

- Microsoft Windows XP Professional, Version 2002, service pack2
- 1MB is required for the shaders.

### 2.1.2 Installation procedure for the Offline Shader Compiler on Microsoft Windows

The procedure to install the Offline Shader Compiler on Microsoft Windows is:

1. Locate the Mali Developer Download Center website at:  
<http://www.malideveloper.com>
2. Select the Offline Shader Compiler package to download.
3. Run the file `Mali_GPU_Offline_Shader_Compiler_WinXP_vm.n.exe`  
where:  
*m* identifies the major version  
*n* identifies the minor version.
4. Select the required installation options and then click **Finish** to complete the installation.

By default, the Offline Shader Compiler is installed in:

`C:\Program Files\ARM\Mali Developer Tools\Mali GPU Offline Shader Compiler vm.n`

## 2.2 Installing the Offline Shader Compiler on Linux

This section describes how to install the Offline Shader Compiler on Linux. It contains the following sections:

- *Installation requirements for the Offline Shader Compiler on Linux*
- *Installation procedure for the Offline Shader Compiler on Linux*

### ———— Note —————

The Offline Shader Compiler has been tested successfully on a 32-bit computer.

### 2.2.1 Installation requirements for the Offline Shader Compiler on Linux

To install the Offline Shader Compiler on Linux, you require:

- Red Hat Enterprise Linux, Release 4
- GNU tar version 1.13 or higher
- 1MB is required for the shaders.

### 2.2.2 Installation procedure for the Offline Shader Compiler on Linux

To install the Offline Shader Compiler on Linux:

1. Locate the Mali Developer Download Center website at:  
<http://www.malideveloper.com>
2. Download the following package:  
Mali\_GPU\_Offline\_Shader\_Compiler\_RHEL4\_vm.n.tar.gz  
where:  
**m** identifies the major version  
**n** identifies the minor version.
3. To decompress the file:
  - open a command terminal and navigate to the directory where you have downloaded the package
  - type the following command:  
tar -zxvf Mali\_GPU\_Offline\_Shader\_Compiler\_RHEL4\_vm.n.tar.gz

By default, the Offline Shader Compiler is installed in:

ARM/Mali\_Developer\_Tools/Mali\_GPU\_Offline\_Shader\_Compiler\_vm.n



# Chapter 3

## The Offline Shader Compiler

This chapter describes the use of the Offline Shader Compiler. It contains the following section:

- *General procedure for compiling files* on page 3-2
- *Testing the shader* on page 3-4.

## 3.1 General procedure for compiling files

This section describes the general steps to use when compiling files using the Offline Shader Compiler. It also describes the command line syntax and how to specify options. It contains the following sections:

The general procedure for using the compiler is as follows:

1. Edit your shader file with a text editor program.
2. Compile the shader program using the `malisc` application, with suitable options. For Windows, the `malisc` executable file is located in:

`C:\Program Files\ARM\Mali Developmeer Tools\Mali GPU Offline Shader Compiler vm.n`

Compile the shader file using the following command:

```
malisc [options] [-o outfile] source-files
```

For information on the options that are available, see *Offline Shader Compiler options*.

Source files of the same shader type are concatenated in the order you specify, and are treated as a single shader file.

If one or more source files have the extension `.vert`, the shader is compiled as a vertex shader. If no source files have the `.vert` or `.frag` extension, use the `--vert` and `--frag` options to specify the kind of shader to use.

———— **Note** ————

You cannot compile vertex shaders and fragment shaders in the same compiler run.

If you specify no options, and no filename, the compiler returns help information. If you specify only a filename, then provided it has the `.frag` or `.vert` suffix, the file is compiled and saved as `output.binshader` in the current directory.

### 3.1.1 Offline Shader Compiler options

`-DNAME[=VALUE]`

Predefine `NAME` as an OpenGL ESSL macro, with definition `VALUE`. You can specify multiple macro definitions on the command line. Each macro affects all files specified. See the *OpenGL ES Shading Language Specification* for more information about ESSL macros.

`--vert` Process shader as a vertex shader.

- If one or more source files have the extension `.vert`, the shader is compiled as a vertex shader. If no source files have the `.vert` or `.frag` extension, use the `--vert` and `--frag` options to specify the kind of shader to use.
- `--frag` Process shader as a fragment shader.
- If one or more source files have the extension `.frag`, the shader is compiled as a fragment shader. If no source files have the `.vert` or `.frag` extension, use the `--vert` and `--frag` options to specify the kind of shader to use.
- `-v, --verbose` Print verbose information about the compiled shader.
- The verbose output from the compiler includes information about the number of GPU cycles the shader takes to execute. Use this information to assess the performance implications of changes to the shader source code.
- Example output when using the `--verbose` option to compile a fragment shader is as follows:
- ```
Number of instruction words emitted: 17
Number of cycles for shortest code path: 17
Number of cycles for longest code path: 17
Note: Cycle counts do not include possible stalls caused by cache misses.
```
- `-o outfile` Write output to `outfile`. Default output is: `output.binshader`.
- `--core=core` Target a specific graphics core.
- Supported cores are Mali-200 and Mali-400.
- `-r rXpY, --revision=rXpY`
- Target a specific hardware revision, release X, patch Y. Revisions `r0p0`, `r0p1`, `r0p2`, `r0p3`, `r0p4` and `r0p5` are currently supported. The default revision is `r0p5`.
- `--testchip` Same as `--revision=r0p1`.

## 3.2 Testing the shader

After compiling the shader, test the shader as follows:

1. Load the compiled shader into an OpenGL ES 2.0 application using the `glShaderBinary` call.
2. Draw geometry using the shader.
3. Examine the visual output. If unsatisfactory, alter the uniforms used to control the shader, or repeat the steps in *General procedure for compiling files* on page 3-2, using a different shader.

---

**Note**

A shader pair consisting of a vertex shader and a fragment shader, when successfully compiled using the Offline Shader Compiler, cannot fail linking because of resource constraints. Linking only fails if the uniform or varying variables of the shaders are incompatible.

---

See Mali GPU Shader Library for examples of vertex shaders and fragment shaders included in the Mali Shader Library.

# Glossary

This glossary describes some of the terms used in ARM manuals. Where terms can have several meanings, the meaning presented here is intended.

- API** *See* Application Programming Interface.
- Application Programming Interface (API)**  
A specification for a set of procedures, functions, data structures, and constants that are used to interface two or more software components together. For example, an API between an operating system and the application programs that use it might specify exactly how to read data from a file.
- ESSL** *See* OpenGL ES Shading Language.
- Fragment shader** A program running on the pixel processor that calculates the color and other characteristics of each fragment.
- GPU** *See* Graphics Processor Unit.
- Graphics application** A custom program that runs on the platform CPU and uses the Mali GPU to display graphics.
- Graphics Processor Unit (GPU)**  
A hardware accelerator for graphics systems using OpenGL ES and OpenVG. The hardware accelerator comprises of an optional geometry processor and a pixel processor together with memory management. Mali programmable GPUs, such as the

Mali-200 and Mali-400 MP GPUs, consist of a geometry processor and at least one pixel processor. Mali fixed-function GPUs, such as the Mali-55 GPU consist of a pixel processor only.

**Instrumented drivers**

Alternative graphics drivers that are used with the Mali GPU. The Instrumented drivers include additional functionality such as error logging and recording performance data files for use by the Performance Analysis Tool.

*See also* Performance Analysis Tool.

**Mali**

A name given to graphics software and hardware products from ARM that aid 2D and 3D acceleration.

**Mali Developer Tools**

A set of development programs that enables software developers to create graphic applications.

**Offline Shader Compiler**

A command line tool that translates vertex shaders and fragment shaders written in the ESSL into binary vertex shaders and binary fragment shaders that you can link and run on the Mali GPU.

**OpenGL ES Shading Language (ESSL)**

A programming language used to create custom shader programs that can be used within a programmable pipeline, on graphics hardware. You can also use pre-defined library shaders, written in ESSL.

**Performance Analysis Tool**

A fully-customizable GUI tool that displays and analyzes performance data files produced by the Instrumented drivers, together with framebuffer information.

*See also* Instrumented drivers, Performance data file.

**Performance data file**

Files that contain a description of the performance counters, together with the performance counter data in the form of a series of values and images. Performance data files are saved in .ds2 format and can be loaded directly into the Performance Analysis Tool.

**Pixel**

A pixel is a discrete element that forms part of an image on a display. The word pixel is derived from the term Picture Element.

**Shader**

A program, usually an application program, running on the GPU, that calculates some aspect of the graphical output. See fragment shader and vertex shader.

**Shader Library**

A set of shader examples, and other information, designed to assist with developing shader programs for the Mali GPU. The Shader Library is a component of the Mali Developer Tools.

**Vertex shader**

A program running on the geometry processor, that calculates the position and other characteristics, such as color and texture coordinates, for each vertex.

