

Mali™ Offline Shader Compiler

Version: 3.0

User Guide



Mali Offline Shader Compiler

User Guide

Copyright © 2009-2012 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
31 January 2012	B	Non-Confidential	Updated for v3.0

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.

Product Status

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

Web Address

<http://www.arm.com>

Contents

Mali Offline Shader Compiler User Guide

	Preface	
	About this book	v
	Feedback	vii
Chapter 1	Introduction	
	1.1 About the Mali Offline Shader Compiler	1-2
Chapter 2	Installation	
	2.1 Installing the Offline Shader Compiler on Microsoft Windows	2-2
	2.2 Installing the Offline Shader Compiler on Mac OS X	2-3
	2.3 Installing the Offline Shader Compiler on Linux	2-4
Chapter 3	The Offline Shader Compiler	
	3.1 General procedure for compiling files	3-2
	3.2 Testing the shader	3-4

Preface

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

- *About this book* on page v
- *Feedback* on page vii.

About this book

This is the *Mali 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 applications on a desktop workstation, using the Windows XP, Mac OS X, or Linux operating systems, 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.

Typographical Conventions

The typographical conventions are:

<i>italic</i>	Highlights important notes, introduces special terminology, denotes internal cross-references, and citations.
bold	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.
<u>monospace</u>	Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
<i>monospace italic</i>	Denotes arguments to monospace text where the argument is to be replaced by a specific value.
monospace bold	Denotes language keywords when used outside example code.

Additional reading

This section lists publications by ARM and by third parties.

See Infocenter, <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 Performance Analysis Tool User Guide* (ARM DUI 0502)

- *Mali 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)
- *Mali GPU Mali Binary Asset Exporter User Guide* (ARM DUI 0507)
- *Mali GPU Shader Library User Guide* (ARM DUI 0510)
- *OpenGL ES 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>.
- *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 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. Give:

- the title
- the number, ARM DUI 0513B
- 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 Offline Shader Compiler. It contains the following section:

- [About the Mali Offline Shader Compiler on page 1-2](#)

1.1 About the Mali Offline Shader Compiler

The Mali Offline Shader Compiler is a command line tool that compiles 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.

You can use the Mali 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. Binary shaders reduce the time it takes to load a shader load time.
- 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.
- Ship 3D content with pre-compiled binary shaders instead of the OpenGL ESSL source code that is required if you use the On-target Shader Compiler.

Note

- Using the Mali Offline Shader Compiler is optional.
 - Because a particular binary shader can only be used on one type of GPU, using compiled shaders reduces portability.
 - If you require portability, you can use the On-target Shader Compiler which enables the OpenGL ES 2.0 drivers to compile OpenGL ESSL shader code dynamically when application runs on a target platform.
-

Chapter 2

Installation

This chapter provides information about installing the Offline Shader Compiler Tool. It contains the following sections:

- *Installing the Offline Shader Compiler on Microsoft Windows* on page 2-2
- *Installing the Offline Shader Compiler on Mac OS X* on page 2-3
- *Installing the Offline Shader Compiler on Linux* on page 2-4.

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, service pack 3 (or above).

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 web site at:
<http://www.malideveloper.com>
2. Select the Offline Shader Compiler package to download.
3. Run the file `Mali_Offline_Shader_Compiler_vm.n.o.p_Win32.msi`
where:
m identifies the major version
n, o, and p
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 Offline Shader Compiler vm.n.o`

2.2 Installing the Offline Shader Compiler on Mac OS X

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

- [Installation requirements for the Offline Shader Compiler on Mac OS X](#)
- [Installation procedure for the Offline Shader Compiler on Mac OS X](#)

———— 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 Mac OS X

To install the Offline Shader Compiler on Mac OS X, you require Mac OS X 10.6 (or above).

2.2.2 Installation procedure for the Offline Shader Compiler on Mac OS X

To install the Offline Shader Compiler on Mac OS X:

1. Locate the Mali Developer Download Center web site at:
<http://www.malideveloper.com>
2. Download the following package:
Mali_Offline_Shader_Compiler_vm.n.o.p_MacOSX.tar.gz
where:
m identifies the major version
n, o, and p
identifies the minor version.
3. 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_Offline_Shader_Compiler_vm.n.o.p_MacOSX.tar.gz

The following directory is created which contains the Offline Shader Compiler:

Mali_Offline_Shader_Compiler_vm.n.o

2.3 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.

2.3.1 Installation requirements for the Offline Shader Compiler on Linux

To install the Offline Shader Compiler on Linux, you require Ubuntu Linux 10.04 (or above).

2.3.2 Installation procedure for the Offline Shader Compiler on Linux

To install the Offline Shader Compiler on Linux:

1. Locate the Mali Developer Download Center web site at:
<http://www.malideveloper.com>
2. Download the following package:
`Mali_Offline_Shader_Compiler_vm.n.o.p_Linux.tar.gz`
where:
m identifies the major version
n, o, and p identifies the minor version.
3. 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_Offline_Shader_Compiler_vm.n.o.p_Linux.tar.gz`

The following directory is created which contains the Offline Shader Compiler:

`Mali_Offline_Shader_Compiler_vm.n.o`

Chapter 3

The Offline Shader Compiler

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

- *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 to compile files with the Offline Shader Compiler. It also describes the command line syntax and how to specify options.

The general procedure for using the compiler is as follows:

1. Edit your shader file with a text editor program.
2. Use the following command to compile the shader file:

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

For Windows, the malisc executable file is located in:

```
C:\Program Files\ARM\Mali Development Tools\Mali Offline Shader Compiler vm.n.o
```

For options, see [Offline Shader Compiler options](#).

————— Note —————

- You cannot compile vertex shaders and fragment shaders in the same compiler run.
- Source files of the same shader type are concatenated in the order you specify, and are treated as a single shader file.
- 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.
- 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.
- If you specify no options, and no filename, the compiler returns help information.

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 none of source files have a .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 3-1 on page 3-3](#) shows an example of the output resulting from the --verbose option to compile a fragment shader:

Example 3-1 Example of using the --verbose option

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*.
`--testchip` Same as `--core=Mali-200 --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 use a different shader and repeat the steps in [General procedure for compiling files on page 3-2](#).

———— **Note** —————

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

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