Intel® Graphics for Linux* - Programmer's Reference Manuals

Contents

Programmer's Reference Manuals

1

The Programmer's Reference Manuals (PRM) describe the architectural behavior and programming environment of the chipset and graphics devices. The Graphics Controller (GC) contains an extensive set of registers and instructions for configuration, 2D, 3D, and video systems. The PRM describes the register, instruction, and memory interfaces and the device behaviors, as controlled and observed through those interfaces. The PRM also describes the registers and instructions and provides detailed bit/field descriptions. This information is critical to the development and maintenance of graphics drivers from Intel for this hardware.

Hardware Specifications

Intel® Iris® XE and Intel® UHD Graphics Open Source Programmer's Reference Manual for the 2020-2021 11th Generation Intel® Xeon® , Intel® Core™, Intel® Celeron®, Intel® Pentium® Gold Processors (Formerly Tiger Lake)

Volume 1: Preface

Volume 2a: Command Reference: Instructions

Volume 2b: Command Reference: Enumerations

Volume 2c: Command Reference: Registers Part 1 - Registers A through L

Volume 2c: Command Reference: Registers Part 2 - Registers M through Z

Volume 2d: Command Reference: Structures

Volume 3: GPU Overview

Volume 4: Configurations

Volume 5: Memory Data Formats

Volume 6: Memory Views

Volume 7: Memory Cache

Volume 8: Command Stream Programming

Volume 9: Render Engine

Volume 10: Copy Engine

Volume 11: Media Engine

Volume 12: Display Engine

Volume 13: General Assets

Volume 14: Workarounds

Intel® UHD Graphics Open Source Programmer's Reference Manual for the 2020 Intel® Core™ Processors with Intel® Hybrid Technology (Formerly Lakefield)

Volume 1: Preface

Volume 2a: Command Reference: Instructions

Volume 2b: Command Reference: Enumerations

Intel® Iris™ Xᵉ MAX Graphics Open Source Programmer's Reference Manual for the 2020 Intel® Arc™ GPU (Formerly DG1)

Intel® UHD Graphics Open Source Programmer's Reference Manual for the 2019 - 2020 Intel® Core™ Processors, Intel® Pentium® Gold Processors, and Intel® Celeron® Processors (Formerly Comet Lake)

Volume 1: Configurations

Intel® UHD Graphics Open Source Programmer's Reference Manual for the 2018 - 2019 Intel® Core™ Processors, Intel® Pentium® Gold Processors, and Intel® Celeron® Processors (Formerly Whiskey Lake)

Volume 1: Configurations

Intel® UHD Graphics Open Source Programmer's Referenece Manual for the 2018 - 2019 Intel® Core™ Processors and Intel® Pentium® Gold Processor Series (Formerly Amber Lake)

Volume 1: Configurations

Intel® Iris™ Plus Graphics and Intel® UHD Graphics Open Source Programmer's Reference Manual for the 2017 - 2019 Intel® Core™ Processors, Intel® Pentium® Gold Processors, Intel® Celeron® Processors, and Xeon® Processors (Formerly Coffee Lake)

Volume 1: Configurations

Intel® Iris™ Plus Graphics and Intel® UHD Graphics Open Source Programmer's Reference Manual for the 2019 10TH Generation Intel Core™ Processors (Formerly Ice Lake)

Volume 1: Preface

Volume 2a - Command Reference: Instructions (Command Opcodes)

Volume 2b: Command Reference: Enumerations

Volume 2c: Command Reference: Registers Part 1 – Registers A through L

Volume 2c: Command Reference: Registers, Part 2 – Registers M through Z

Volume 2d: Command Reference: Structures

Volume 3: GPU Overview

Volume 4: Configurations

Volume 5: Memory Data Formats

Volume 6: Memory Views

Volume 7: Memory Cache

Volume 8: Command Stream Programming

Volume 9: Render Engine

Volume 10: Copy Engine

Volume 11: Media Engines

Volume 12: Display Engine

Volume 13: Software/Hardware System Interface

Volume 14: Workarounds

Intel® HD Graphics Open Source Programmer's Reference Manual for the 2016 Intel Atom® Processors, Intel® Celeron® Processors, and Intel® Pentium® Processors (Formerly Apollo Lake (Broxton))

Volume 1: Preface

Volume 2a: Command Reference: Instructions (Command Opcodes)

Volume 2b: Command Reference: Registers

Volume 2c: Command Reference: Structures

Volume 3: Configurations

Volume 4: Memory Views

Volume 5: Command Stream Programming

Volume 6: 3D-Media-GPGPU

Volume 7: Display

Volume 8: Workarounds

Intel® HD Graphics, Intel® Iris™ Graphics, and Intel® Iris™ Pro Graphics Programmer's Reference Manual for the 2015-2016 Intel® Core™ Processor's, Intel® Celeron® Processors and Intel® Pentium® Processor's (Formerly Skylake)

Volume 1: Preface

Volume 2a: Command Reference-Instructions (Command Opcodes)

Volume 2a: Command Reference-Instructions (HuC)

Volume 2b: Command Reference-Enumerations

Volume 2c: Command Reference-Registers Part 1 - Registers A through L

Volume 2c: Command Reference-Registers Part 2 - Registers M through Z

Volume 2d: Command Reference-Structures

Volume 3: GPU Overview

Volume 4: Configurations

Volume 5: Memory Views

Volume 6: Command Stream Programming

Volume 7: 3D-Media-GPGPU

Volume 8: Media VDBOX

Volume 9: Media VEBOX

Volume 10: HEVC

Volume 11: Blitter

Volume 12: Display

Volume 13: MMIO

Volume 14: Observability

Volume 15: SFC

Volume 16: Workarounds

Intel® Graphics Programmer's Reference Manual for the 2014-2015 Intel Atom® Processors, Intel® Celeron® Processors, and Intel® Pentium® Processors (Formerly Cherry Trail/Braswell) (Cherryview/Braswell)

Volume 1: Preface

Volume 2a: Command Reference-Instructions

Volume 2b: Command Reference-Enumerations

Volume 2c: Command Reference-Registers

Volume 2d: Command Reference-Structures

Volume 2e: Command Reference-Additional Information

Volume 3: GPU Overview

Volume 4: Configurations

Volume 5: Memory Views

Volume 6: Command Stream Programming

Volume 7: 3D-Media-GPGPU

Volume 8: Media VDBOX

Volume 9: Media VEBOX

Volume 10: HEVC

Volume 11: Blitter

Volume 12: Display

Volume 13: MMIO

Volume 14: Observability

Volume 15: Graphics PCI Registers

Volume 16: Workarounds

Intel® HD Graphics and Intel® Iris™ Graphics Open Source Programmer's Reference Manual for the 2014-2015 Intel® Core™ Processors, Intel® Celeron® Processors, and Intel® Pentium® Processors (Formerly Broadwell)

Volume 1: Preface

Volume 2a: Command Reference-Instructions

Volume 2b: Command Reference-Enumerations

Volume 2c: Command Reference-Registers

Volume 2d: Command Reference-Structures

Volume 3: GPU Overview

Volume 4: Configurations

Volume 5: Memory Views

Volume 6: Command Stream Programming

Volume 7: 3D-Media-GPGPU

Volume 8: Media VDBOX

Volume 9: Media VEBOX

Volume 10: Blitter

Volume 11: Display

Volume 12: PCIE Configuration Registers

Volume 13: MMIO

Volume 14: Observability

Volume 15: Workarounds

Intel® Graphics Open Source Programmer's Reference Manual for the 2014 Intel Atom® Processors, Intel® Celeron® Processors, and Intel® Pentium® Processors, (Formerly Bay Trail Valleyview)

Volume 1: Introduction

Volume 2, Part 1: Command Reference-Enumerations

Volume 2, Part 2: Command Reference-Instructions

Volume 2, Part 3: Command Reference-Registers

Volume 2, Part 4: Command Reference-Structures

Volume 3: GPU Overview

Volume 4: Configurations

Volume 5: Memory Views

Volume 6: Command Stream Programming

Volume 7: 3D-Media-GPGGU

Volume 8: Media VDBOX

Volume 9: Blitter

Volume 10: Display

Volume 11: Graphics Interface

Intel® HD Graphics Open Source Programmer's Reference Manual for 2013 Intel® Core™ Processor Family Based on Platform Code Named "Haswell", Including, Intel® Iris™ Graphics, and Intel® Iris™ Pro Graphics

Volume 1 Preface and Introduction

Volume 2a Command Reference - Enumerations

Volume 2b Command Reference - Instructions (Command Opcodes)

Volume 2c Command Reference - Registers

Volume 2d Command Reference - Structures

Volume 3 GPU Overview

Volume 4 Configurations

Volume 5 Memory Views

Volume 6: Command Stream Programming

Volume 7: 3D Media GPGU

Volume 8 Media VDBOX

Volume 9 Media VEBOX

Volume 10 Blitter

Volume 11a Display

Volume 11b Display Watermark Guide

Volume 12 PCIE Configuration Registers

Observability Performance Counters

Intel® HD Graphics Open Source Programmer's Reference Manual for 2012 Intel® Core™ Processor Family

Volume 1 Part 1: Graphics Core

Volume 1 Part 2: Graphics Core - MMIO, Media Registers & Programming Environment

Volume 1 Part 3: Graphics Core - Memory Interface and Commands for the Render Engine

Volume 1 Part 4: Graphics Core - Blitter Engine

Volume 1 Part 5: Graphics Core - Video Codec Engine Command Streamer

Volume 1 Part 6: GT Interface Register

Volume 1 Part 7: L3$/URB

Volume 2 Part 1: 3D/Media - 3D Pipeline

Volume 2 Part 2: Media and General Purpose Pipeline

Volume 2 Part 3: Multi-Format Transcoder - MFX

Volume 3 Part 1: VGA and Extended VGA Registers

Volume 3 Part 2: PCI Registers

Volume 3 Part 3: North Display Engine

Volume 3 Part 4: South Display Engine

Volume 4 Part 1: Subsystem and Cores - Shared Functions

Volume 4 Part 2: Subsystem and Cores - Message Gateway, URB, Video Motion Estimation, Pixel Interpolator

Volume 4 Part 3: Execution Unit ISA

Intel® HD Graphics Open Source Programmer's Reference Manual for the 2011 Intel® Core™ Processor Family

Volume 1 Part 1: Graphics Core

Volume 1 Part 2: Graphics Core - MMIO, Media Registers & Programming Environment

Volume 1 Part 3: Graphics Core - Memory Interface and Commands for the Render Engine

Volume 1 Part 4: Graphics Core - Video Codec Engine

Volume 1 Part 5: Graphics Core - Blitter Engine

Volume 2 Part 1: 3D/Media - 3D Pipeline

Volume 2 Part 2: 3D/Media - Media

Volume 3 Part 1: Display Registers - VGA Registers

Volume 3 Part 2: Display Registers - CPU Registers

Volume 3 Part 3: PCH Display Registers

Supplement to Volume 3 Part 3: PCH Display Registers

Volume 4 Part 1: Subsystem and Cores - Shared Functions

Volume 4 Part 2: Subsystem and Cores - Message Gateway, URB, Video Motion, and IS

Intel® HD Graphics Open Source Programmer's Reference Manual for 2010 Intel® Core™ Processor Family

Volume 1 Part 1: Graphics Core

Volume 1 Part 2: Graphics Core - MMIO, Media Registers & Programming Environment

Volume 1 Part 3: Graphics Core - Memory Interface and Commands Render Engine

Volume 1 Part 4: Graphics Core - Video Codec Engine

Volume 1 Part 5: Graphics Core - Blitter Engine

Volume 2 Part 1: 3D/Media - 3D Pipeline

Volume 2 Part 2: 3D/Media - Media

Volume 3 Part 1: Display Registers - VGA Registers

Volume 3 Part 2: Display Registers - CPU Registers

Volume 3 Part 3: PCH Display Registers

Volume 4 Part 1: Subsystem and Cores - Shared Functions

Volume 4 Part 2: Subsystem and Cores - Message Gateway, URB, Video Motion, and IS

Intel® G45 Express Chipset [Graphics and Memory Controller HUB-GMCH] Programmer's Reference Manual

G45: Volume 1a Graphics Core

G45: Volume Two: 3D/Media

G45: Volume Three: Display Register

G45: Volume Four: Subsystem and Cores

Intel® 965 Express Chipset Family and Intel® G35 Express Chipset Graphics Controller Programmer's Reference Manual

Volume One: Graphics Core

Volume Two: 3D/Media

Volume Three: Display Registers

Volume Four: Subsystem and Cores

2023 Intel® Processors - Alchemist/Arctic Sound-M Platform

Intel® Arc™ A-Series Graphics and Intel Data Center GPU Flex Series. Open-Source Programmer's Reference Manual. For the discrete GPUs code named "Alchemist" and "Arctic Sound-M".

Attachment	Size
Volume 1: Preface	803 KB
Volume 2a: Command Reference: Instructions	14.2 MB
Volume 2b: Command Reference: Enumerations	1.1 MB
Volume 2c: Command Reference: Registers Part 1	12.7 MB
Volume 2c: Command Reference: Registers Part 2	18.1 MB
Volume 2d: Command Reference: Structures	11 MB
Volume 3: GPU Overview	204 KB
Volume 4: Configurations	452 KB
Volume 5: Memory Data Formats	1.5 MB
Volume 6: Memory Views	928 KB
Volume 7: Memory Cache	197 KB
Volume 8: Command Stream Programming	1 MB
Volume 9: Render Engine	9.4 MB
Volume 10: Copy Engine	826 KB

Attachment	Size
Volume 11: Media Engines	4.1 MB
Volume 12: Display Engines	6.6 MB
Volume 13: SW/HW System Interface	2.2 MB
Volume 14: Workarounds	317 KB

2021 Intel® Processors (Formerly Rocket Lake)

Intel® UHD Graphics Open Source Programmer's Reference Manual for the 2021 11th Generation Intel Core™ Processors, Intel Xeon® Processors, and Intel 500 Series Chipsets based on the "Rocket Lake" Platform

Attachment	Size
Volume 1: Preface	854 KB
Volume 2: Command Reference: Registers	4.7 MB
Volume 3: Configurations	272 KB
Volume 4: Display Engine	6.1 MB
Volume 5: Workarounds	257 KB

2020-2021 Intel® Processors (Formerly Tiger Lake)

Intel® Iris® Xᵉ and Intel® UHD Graphics Open Source Programmer's Reference Manual for the 2020-2021 11th Generation Intel® Xeon® , Intel® Core™ Processors, Intel® Celeron® Processors, Intel® Pentium® Gold Processors (Formerly Tiger Lake)

Attachment	Size
Volume 1: Preface	811.79 KB
Volume 2a: Command Reference: Instructions	13.68 MB
Volume 2b: Command Reference: Enumerations	654.24 KB
Volume 2c: Command Reference: Registers Part 1 - Registers A through L	11.47 MB
Volume 2c: Command Reference: Registers Part 2 - Registers M through Z	13.54 MB
Volume 2d: Command Reference: Structures	10.17 MB
Volume 3: GPU Overview	222.42 KB
Volume 4: Configurations	332.55 KB
Volume 5: Memory Data Formats	1.82 MB
Volume 6: Memory Views	845.51 KB
Volume 7: Memory Cache	242.48 KB
Volume 8: Command Stream Programming	948.03 KB
Volume 9: Render Engine	9.35 MB
Volume 10: Copy Engine	850.71 KB
Volume 11: Media Engine	4.85 MB
Volume 12: Display Engine	5.85 MB
Volume 13: General Assets	955.65 KB
Volume 14: Workarounds	716.08 KB

2020 Intel® Processors with Intel® Hybrid Technology (Formerly Lakefield)

Intel® UHD Graphics Open Source Programmer's Reference Manual for the 2020 Intel® Core™ Processors with Intel Hybrid Technology (formerly Lakefield)

Attachment	Size
Volume 1: Preface	825.55 KB

Attachment	Size
Volume 2a: Command Reference: Instructions	10.54 MB
Volume 2b: Command Reference: Enumerations	554.19 KB
Volume 2c: Command Reference: Registers Part 1 - Registers A through L	8.7 MB
Volume 2c: Command Reference: Registers Part 2 - Registers M through Z	12.5 MB
Volume 2d: Command Reference: Structures	10.38 MB
Volume 3: GPU Overview	207.86 KB
Volume 4: Configurations	212.6 KB
Volume 5: Memory Data Formats	2.51 MB
Volume 6: Memory Views	1.02 MB
Volume 7: Memory Cache	282.9 KB
Volume 8: Command Stream Programming	992.79 KB
Volume 9: Render Engine	9.5 MB
Volume 10: Copy Engine	718.24 KB
Volume 11: Media Engine	5.58 MB
Volume 12: Display Engine	4.65 MB
Volume 13: SW/HW System Interface	905.17 KB
Volume 14: Workarounds	328.96 KB

2020 Intel® Iris® Xe MAX GPU (formerly DG1)

Intel® Iris® Xe Graphics Open Source Programmer's Reference Manual for the 2020 Discrete GPU (Formerly DG1)

Attachment	Size
Volume 1: Preface	851.78 KB
Volume 2a: Command Reference: Instructions	18.11 MB
Volume 2b: Command Refernce: Enumerations	820.37 KB
Volume 2c: Command Reference: Registers Part 1 - Registers A through L	15.64 MB
Volume 2c: Command Reference: Registers Part 2 - Registers M through Z	20.25 MB
Volume 2d: Command Reference: Structures	14.38 MB
Volume 3: GPU Overview	225.49 KB
Volume 4: Configurations	237.07 KB
Volume 5: Memory Data Formats	2.39 MB
Volume 6: Memory Views	1.09 MB
Volume 7: Memory Cache	294.74 KB
Volume 8: Command Stream Programming	1.13 MB
Volume 9: Render Engine	11.82 MB
Volume 10: Copy Engine	956.08 KB
Volume 11: Media Engine	4.6 MB
Volume 12: Display Engine	5.73 MB
Volume 13: General Assets	1.86 MB
Volume 14: Workarounds	461.32 KB

2019 Intel® Processors (Formerly Ice Lake)

Intel® Iris® Plus Graphics and Intel® UHD Graphics Open Source Programmer's Reference Manual for the 2019 10th Generation Intel® Core™ Processors (Formerly Ice Lake)

Attachments	Size
Volume 1: Preface	855.3 KB
Volume 2a - Command Reference: Instructions (Command Opcodes)	15.06 MB
Volume 2b: Command Reference: Enumerations	716.62 KB
Volume 2c: Command Reference: Registers Part 1 - Registers A through L	4.2 MB
Volume 2c: Command Reference: Registers Part 2 - Registers M through Z	5.84 MB

Attachments	Size
Volume 2d: Command Reference: Structures	14.57 MB
Volume 3: GPU Overview	230.91 KB
Volume 4: Configurations	393.56 KB
Volume 5: Memory Data Formats	3.07 MB
Volume 6: Memory Views	2.67 MB
Volume 7: Memory Cache	317.1 KB
Volume 8: Command Stream Programming	1.14 MB
Volume 9: Render Engine	13.5 MB
Volume 10: Copy Engine	761.91 KB
Volume 11: Media Engines	5.68 MB
Volume 12: Display Engine	5.75 MB
Volume 13: SW/HW System Interface	769.27 KB
Volume 14: Workarounds	415.17 KB

2018-2019 Intel® Processors (Formerly Whiskey Lake)

Intel® UHD Graphics Open Source Programmer's Reference Manual for the 2018 - 2019 Intel® Core™ Processors, Intel® Pentium® Gold Processors, and Intel® Celeron® Processors (formerly Whiskey Lake)

Attachment	Size
Volume 1: Configurations	283.8 KB

2017-2019 Intel® Processors (Formerly Amber Lake)

Intel® UHD Graphics Open Source Programmer's Reference Manual for the 2018 - 2019 Intel® Core™ Processors, and Intel® Pentium® Gold Processor Series (Formerly Amber Lake)

Attachment	Size
Volume 1: Configurations	313.58 KB

2017-2019 Intel® Processors (Formerly Coffee Lake)

Intel® Iris® Plus Graphics and Intel® UHD Graphics Open Source Programmer's Reference Manual for the 2017-2019 Intel® Core™ Processors, Intel® Pentium® Gold Processors, Intel® Celeron® Processors, and Intel® Xeon® Processors (Formerly Coffee Lake)

Attachment	Size
Volume 1: Configurations	439.37 KB

2019-2020 Intel® Processors (Formerly Comet Lake)

Intel® Iris® Plus Graphics and Intel® UHD Graphics Open Source Programmer's Reference Manual for the 2019 - 2020 Intel® Core™ Processors, Intel® Pentium® Gold Processors, Intel® Celeron® Processors, and Xeon® Processors (Formerly Comet Lake)

Attachment	Size
Volume 1: Configurations	342.62 KB

2016 Intel® Processors (Formerly Kaby Lake)

Intel® HD Grahics and Intel® Iris® Plus Graphics Open Source Programmer's Reference Manual (PRM) for the 2016 - 2017 Intel® Core™ Proccesors, Intel® Celeron® Processors, and Intel® Pentium® Processors (Formerly Kaby Lake)

Attachment	Size
Volume 1: Preface	1.16 MB
Volume 2a: Command Reference: Instructions (Command Opcodes)	20.04 MB
Volume 2b: Command Reference: Enumerations	1.07 MB
Volume 2c: Command Reference: Registers, Part 1 - Registers A through L	15.23 MB
Volume 2c: Command Reference: Registers, Part 2 - Registers M through Z	16.93 MB
Volume 2d: Command Reference: Structures	9.81 MB
Volume 3: GPU Overview	1.12 MB
Volume 4: Configurations	630.24 KB
Volume 5: Memory Views	6.26 MB
Volume 6: Command Stream Programming	1.24 MB
Volume 7: 3D-Media-GPGPU	14.59 MB
Volumes 8: Media VDBOX	2.49 MB
Volume 9: Media VEBOX	661.31 KB
Volume 10: HEVC	6.73 MB
Volume 11: Blitter	1014.77 KB
Volume 12: Display	2.8 MB
Volume 13: MMIO	381.59 KB
Volume 14: Observability	712.44 KB
Volume 15: SFC	513.93 KB
Volume 16: Workarounds	631.01 KB

2016 Intel® Processors (Formerly Apollo Lake and Broxton)

Intel® HD Graphics Open Source Programmer's Reference Manual for the 2016 Intel Atom® Processors, Intel® Celeron® Processors, and Intel® Pentium® Processors (Formerly Apollo Lake and Broxton)

Attachment	Size
Volume 1: Preface	2.46 MB
Volume 2a: Command Reference: Instructions (Command Opcodes)	561.49 KB
Volume 2b: Command Reference: Registers	721.15 KB
Volume 2c: Command Reference: Structures	435.25 KB
Volume 3: Configurations	1.2 MB
Volume 4: Memory Views	1.94 MB
Volume 5: Command Stream Programming	478.25 KB
Volume 6: 3D-Media-GPGPU	5.71 MB
Volume 7: Display	16.93 MB
Volume 8: Workarounds	1.3 MB

2015-2016 Intel® Processors (Formerly Skylake)

Intel® HD Graphics, Intel® Iris® Graphics, and Intel® Iris® Pro Graphics Programmer's Reference Manual for 2015-2016 Intel® Core™ Processors, Intel® Celeron® Processors and Intel® Pentium® Processors (Formerly Skylake)

Attachment	Size
Volume 1: Preface	1.18 MB

Attachment	Size
Volume 2a: Command Reference: Instructions (Command Opcodes)	19.23 MB
Volume 2a: Command Reference: Instructions (HuC)	793.66 KB
Volume 2b: Command Reference: Enumerations	1.06 MB
Volume 2c: Command Reference: Registers Part 1 - Registers A through L	14.62 MB
Volume 2c: Command Reference: Registers Part 2 - Registers M through Z	16.71 MB
Volume 2d: Command Reference: Structures	9.36 MB
Volume 3: GPU Overview	1.14 MB
Volume 4: Configurations	806.05 KB
Volume 5: Memory Views	6.11 MB
Volume 6: Command Stream Programming	1.25 MB
Volume 7: 3D-Media-GPGPU	14.53 MB
Volume 8: Media VDBOX	2.75 MB
Volume 9: Media VEBOX	699.01 KB
Volume 10: HEVC	3.22 MB
Volume 11: Blitter	1.09 MB
Volume 12: Display	2.88 MB
Volume 13: MMIO	412.06 KB
Volume 14: Observability	718.89 KB
Volume 15: SFC	526.23 KB
Volume 16: Work-arounds	732.24 KB

2014 Intel® Processors (Formerly Bay Trail)

Intel® Graphics Open Source Programmer's Reference Manual for the 2014 Intel Atom® Processors, Intel® Celeron® Processors, and Intel® Pentium® Processors, (Formerly Bay Trail and Valleyview)

Attachment	Size
Volume1: Introduction	669.82 KB
Volume 2a: Command Reference: Enumerations	441.94 KB
Volume 2b: Command Reference: Instructions	6.51 MB
Volume 2c: Command Reference: Registers	2.96 MB
Volume 2d: Command Reference: Structures	2.6 MB
Volume 3: GPU Overview	310.3 KB
Volume 4: Configurations	144.69 KB
Volume 5: Memory Views	1.29 MB
Volume 6: Command Stream Programming	369.56 KB
Volume 7: 3D - Media - GPGPU	7.11 MB
Volume 8: Media VDBOX	798.59 KB
Volume 9: Blitter	477.13 KB
Volume 10: Display	2.3 MB
Volume 11: Graphics Interface	810.64 KB

2014-2015 Intel® Processors (Formerly Cherry Trail and Braswell)

Intel® Graphics Programmer's Reference Manual for the 2014-2015 Intel Atom® Processors, Intel® Celeron® Processors, and Intel® Pentium® Processors (Formerly Cherry Trail, Cherryview, and Braswell)

Attachment	Size
Volume 1: Preface	574.33 KB
Volume 2a: Command Reference: Instructions	18.45 MB

Attachment	Size
Volume 2b: Command Reference: Enumerations	1.11 MB
Volume 2c: Command Reference: Registers	17.94 MB
Volume 2d: Command Reference: Structures	7.93 MB
Volume 2e: Command Reference: Additional Information	1.25 MB
Volume 3: GPU Overview	898.28 KB
Volume 4: Configurations	706.95 KB
Volume 5: Memory Views	3.47 MB
Volume 6: Command Stream Programming	1.24 MB
Volume 7: 3D-Media-GPGPU	13.06 MB
Volume 8: Media VDBOX	2.14 MB
Volume 9: Media VEBOX	815.74 KB
Volume 10: HEVC	881.04 KB
Volume 11: Blitter	1.14 MB
Volume 12: Display	1.38 MB
Volume 13: MMIO	437.56 KB
Volume 14: Observability	660.3 KB
Volume 15: Graphics PCI Registers	547.33 KB
Volume 16: Workarounds	577.23 KB

2014-2015 Intel® Processors (Formerly Broadwell)

Intel® HD Graphics and Intel® Iris® Graphics Open Source Programmer's Reference Manual for the 2014-2015 Intel® Core™ Processors, Intel® Celeron® Processors, and Intel® Pentium® Processors (Formerly Broadwell)

Attachment	Size
Volume 1: Preface	1.18 MB
Volume 2a: Command Reference: Instructions	17.26 MB
Volume 2b: Command Reference: Enumerations	510.73 KB
Volume 2c: Command Reference: Registers	22.41 MB
Volume 2d: Command Reference: Structures	8.32 MB
Volume 3: GPU Overview	902.9 KB
Volume 4: Configurations	606.38 KB
Volume 5: Memory Views	3.34 MB
Volume 6: Command Stream Programming	1.2 MB
Volume 7: 3D-Media-GPGPU	12.42 MB
Volume 8: Media VDBOX	1.98 MB
Volume 9: Media VEBOX	357.16 KB
Volume 10: Blitter	1.12 MB
Volume 11: Display	2.84 MB
Volume 12: PCIE Configuration Registers	889.22 KB
Volume 13: MMIO	451.5 KB
Volume 14: Observability	813.03 KB
Volume 15: Workarounds	759.25 KB

2013 Intel® Core™ Processor Family (Formerly Haswell)

Intel® HD Graphics, Intel® Iris® Graphics, and Intel® Iris® Pro Graphics Open Source Programmer's Reference Manual for 2013 Intel® Core™ Processor Family (Formerly Haswell)

Attachment	Size
Volume 1: Preface and Introduction	856.89 KB
Volume 2a: Command Reference: Enumerations	547.95 KB
Volume 2b: Command Reference: Instructions (Command Opcodes)	9.56 MB

Attachment	Size
Volume 2c: Command Reference: Registers	7.68 MB
Volume 2d: Command Reference: Structures	4.11 MB
Volume 3: GPU Overview	319.42 KB
Volume 4: Configurations	224.54 KB
Volume 5: Memory Views	1.46 MB
Volume 6: Command Stream Programming	714.64 KB
Volume 7: 3D Media GPGU	9.5 MB
Volume 8: Media VDBOX	1.48 MB
Volume 9: Media VEBOX	462.83 KB
Volume 10: Blitter	440.38 KB
Volume 11a: Display	1.83 MB
Volume 11b: Display Watermark Guide	152.56 KB
Volume 12: PCIE Configuration Registers	961.15 KB
Observability Performance Counters	1.07 MB

2012 Intel® Core™ Processor Family (Formerly Ivy Bridge)

Intel® HD Graphics Open Source Programmer's Reference Manual for 2012 Intel® Core™ Processor Family

Attachment	Size
IVB - Volume 1a: Graphics Core	2.64 MB
IVB - Volume 1b: Graphics Core - MMIO, Media Registers & Programming Environment	1.3 MB
IVB - Volume 1c: Graphics Core - Memory Interface and Commands for the Render Engine	2.37 MB
IVB - Volume 1d: Graphics Core - Blitter Engine	2.3 MB
IVB - Volume 1e: Graphics Core - Video Codec Engine Command Streamer	1.09 MB
IVB - Volume 1f: GT Interface Register	966.91 KB
IVB - Volume 1g: L3$/URB	956.39 KB
IVB - Volume 2a: 3D/Media - 3D Pipeline	5.99 MB
IVB - Volume 2b: Media and General Purpose Pipeline	1.96 MB
IVB - Volume 2c: Multi-Format Transcoder - MFX	2.13 MB
IVB - Volume 3a: VGA and Extended VGA Registers	724.42 KB
IVB - Volume 3b: PCI Registers	612.04 KB
IVB - Volume 3c: North Display Engine	1.85 MB
IVB - Volume 3d: South Display Engine	1.57 MB
IVB - Volume 4a: Subsystem and Cores - Shared Functions	4.15 MB
IVB - Volume 4b: Subsystem and Cores - Message Gateway, URB, Video Motion Estimation, Pixel Interpolator	1.5 MB
IVB - Volume 4c: Execution Unit ISA	2.94 MB

2011 Intel® Core™ Processor Family (Formerly Sandy Bridge)

Intel® HD Graphics Open Source Programmer's Reference Manual for 2011 Intel® Core™ Processor Family

Attachment	Size
SNB - Volume 1 Part 1: Graphics Core	-
SNB - Volume 1 Part 2: Graphics Core - MMIO, Media Registers & Programming Environment	-

Attachment	Size
SNB - Volume 1 Part 3: Graphics Core - Memory Interface and Commands for the Render Engine	-
SNB - Volume 1 Part 4: Graphics Core - Video Codec Engine	-
SNB - Volume 1 Part 5: Graphics Core - Blitter Engine	629.13 KB
SNB - Volume 2 Part 1: 3D/Media - 3D Pipeline	1.31 MB
SNB - Volume 2 Part 2: 3D/Media - Media	579.91 KB
SNB - Volume 3 Part 1: Display Registers - VGA Registers	229.83 KB
SNB - Volume 3 Part 2: Display Registers - CPU Registers	754.46 KB
SNB - Volume 3 Part 3: PCH Display Registers	416.55 KB
Supplement 1 to SNB - Volume 3 Part 3: PCH Display Registers	
SNB - Volume 4 Part 1: Subsystem and Cores - Shared Functions	1.27 MB
SNB - Volume 4 Part 2: Subsystem and Cores - Message Gateway, URB, Video Motion, and IS	1006.25 KB

2010 Intel® Core™ Processor Family (Formerly Iron Lake)

Intel® HD Graphics Open Source Programmer's Reference Manual for 2010 Intel® Core™ Processor Family

Attachment	Size
ILK - Volume 1 Part 1: Graphics Core	1.16 MB
ILK - Volume 1 Part 2: Graphics Core - MMIO, Media Registers & Programming Environment	906.22 KB
ILK - Volume 1 Part 3: Graphics Core - Memory Interface and Commands Render Engine	1.21 MB
ILK - Volume 1 Part 4: Graphics Core - Video Codec Engine	423.87 KB
ILK - Volume 1 Part 5: Graphics Core - Blitter Engine	1.4 MB
ILK - Volume 2 Part 1: 3D/Media - 3D Pipeline	3.9 MB
ILK - Volume 2 Part 2: 3D/Media - Media	2.3 MB
ILK - Volume 3 Part 1: Display Registers - VGA Registers	778.15 KB
ILK - Volume 3 Part 2: Display Registers - CPU Registers	2.02 MB
ILK - Volume 3 Part 3: PCH Display Registers	1.56 MB
ILK - Volume 4 Part 1: Subsystem and Cores - Shared Functions	2.3 MB
ILK - Volume 4 Part 2: Subsystem and Cores - Message Gateway, URB, Video Motion, and IS	1.32 MB

Code Documentation

Documentation pages below are auto generated from code and updated regularly.

Linux Kernel GPU Driver Developer's Guide

This documentation is generated from DRM and i915 Linux kernel drivers.

https://01.org/linuxgraphics/gfx-docs/drm/gpu/index.html

Mesa's Source Code Documentation

https://docs.mesa3d.org/

IGT GPU Tools Reference Manual

This documentation is generated from intel-gpu-tools.

https://drm.pages.freedesktop.org/igt-gpu-tools/

Linux DRM Maintainer's Guide

This documentation is for maintainer-tools.

https://drm.pages.freedesktop.org/maintainer-tools/index.html

GUC to KMD API

https://01.org/linuxgraphics/code-documentation/guc-kmd-api

Source Code Repositories

Kernel Driver Development

The latest code from Intel is maintained at the public GIT Repository:

```
git://anongit.freedesktop.org/drm-intel
```

LIBDRM

The LIBDRM modules are available from the public DRM git repository:

```
https://gitlab.freedesktop.org/mesa/drm
```

X.ORG 2D Driver

The Intel driver for X.org is available from the public X.org git repository:

```
git://anongit.freedesktop.org/xorg/driver/xf86-video-intel
```

The release tarball can be downloaded from http://xorg.freedesktop.org/archive/individual/driver/

Open Source Intel® Graphics Driver (Formerly the Mesa Project)

The source code is available from the Mesa releases page

Mesa development tree:

```
https://gitlab.freedesktop.org/mesa/mesa
```

Intel® GPU Tools

Intel® GPU tools are available from the repository:

```
https://gitlab.freedesktop.org/drm/igt-gpu-tools
```

Build Guide

Introduction

This page is for experienced Linux users. If you are not familiar with building software components using autotools, we recommend that you use a pre-compiled and tested version available on your current distro.

Building Kernel

Please refer to https://kernelnewbies.org/KernelBuildfor detailed instruction on kernel building.

The only exception is that you should use drm-tip repository. The following is a short example to get you started:

```
git clone git://anongit.freedesktop.org/drm-tip

cd drm-tip

cp /boot/config-`uname -r`* .config
# or, depending on your distribution
gunzip -c /proc/config.gz > .config

make olddefconfig

make

sudo make modules_install

sudo make install
```

Building User Space Components

Please refer to the README files available inside each of the user space components in order to get the most updated building instructions for that code.

GuC KMD API

```
/*****************************************************************************************
** Copyright © 2016 Intel® Corporation
**
** Permission is hereby granted, free of charge, to any person obtaining a
** copy of this software and associated documentation files (the "Software"),
** to deal in the Software without restriction, including without limitation
** the rights to use, copy, modify, merge, publish, distribute, sublicense,
** and/or sell copies of the Software, and to permit persons to whom the
** Software is furnished to do so, subject to the following conditions:
**
** The above copyright notice and this permission notice (including the next
** paragraph) shall be included in all copies or substantial portions of the
** Software.
**
** THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
** IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
** FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.  IN NO EVENT SHALL
** THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
** LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
** FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS
** IN THE SOFTWARE.
**
*****************************************************************************************/

/*****************************************************************************************
** File Name    : UkGuckmdInterface.h
**
** Description  : Definitions for GuC to KMD interface
**
** Environment  : GuC
**
** Notes:
```

```
**
*****************************************************************************************/

#ifndef UK_GUC_KMD_INTEFACE_H
#define UK_GUC_KMD_INTEFACE_H
#if defined (_MSC_VER)
    #pragma pack(push, 1)
#else
    #pragma pack(1)
#endif
#include "types.h"

typedef ULONG64 ADDRESS64;
typedef ULONG   ADDRESS32;

//*****************************************************************************
// MACRO: KM_BIT_RANGE
// PURPOSE: Calculates the number of bits between the startbit and the endbit
// and count is inclusive of both bits. The bits are 0 based.
//*****************************************************************************
#ifndef KM_BIT_RANGE
#define KM_BIT_RANGE(endbit, startbit)     ((endbit)-(startbit)+1)

//*****************************************************************************
// MACRO: KM_BIT
// PURPOSE: Used for clarity when defining bit structs
//*****************************************************************************
#define KM_BIT(bit)     (1)

#endif

//*********************************************************************************************************
// Struct:  Render Power Clock State register definition
// PURPOSE: KMD/GuC uses this definition to program Render Power Clock State register in ring
context
//          at Exec List Submit Port time.
//*********************************************************************************************************
typedef struct _KM_UK_RENDER_PWR_CLK_STATE
{
    union
    {
        struct
        {
            // 0010: 2 EUs
            // 0100: 4 EUs
            // 0110: 6 EUs
            // 1000 :8 EUs
            DWORD EUmin                 : KM_BIT_RANGE(3,0); // Minimum number of EUs to power
(per subslice if multiple subslices enabled)
            DWORD EUmax                 : KM_BIT_RANGE(7,4); // Maximum number of EUs to power
(per subslice if multiple subslices enabled).
            //  To specify an exact number of subslices, set EUmax equal to EUmin

            // Number of subslices to power:
            // 001: 1 subslice
            // 010: 2 subslices
            // 011: 3 subslices
            // BXT only! This MBZ for SKL!
```

```
            DWORD SScount               : KM_BIT_RANGE(10,8); // sub-slice count

            // BXT only! This MBZ for SKL!
            DWORD SScountEn             : KM_BIT_RANGE(11,11); // Enable Subslice Count Request
(1 = Use SliceCount from this register)

            DWORD                       : KM_BIT_RANGE(14,12);

            // SKL only! This  MBZ for BXT!
            DWORD SliceCount            : KM_BIT_RANGE(17,15);
            // SKL only! This  MBZ for BXT!
            DWORD SCountEn              : KM_BIT_RANGE(18,18);

            DWORD                       : KM_BIT_RANGE(30,19);

            // Main trigger: Power Clock State Enable
            //  0: No specific power state set, no message/wait with PMunit
            //  1: CSunit sends the contents of this register to PMunit each time it is written,
Send contents of this register to PMunit, wait for Ack.
            DWORD PowerClkStateEn       : KM_BIT_RANGE(31,31);
        };

        DWORD Data;
    };
}KM_UK_RENDER_PWR_CLK_STATE;

//*****************************************************************************
// Enum:  IGFX_ENGINE
// PURPOSE: Engine IDs available in the GFX HW
//*****************************************************************************
typedef enum
{
    // The Enum values are important as some macros and loops depend on them being sequential
    IGFX_RENDER_ENGINE       = 0,
    IGFX_VIDEO_ENGINE        = 1,
    IGFX_BLITTER_ENGINE      = 2,
    IGFX_VIDEOENHANCE_ENGINE = 3,
    IGFX_VIDEO_ENGINE2       = 4,
    IGFX_ABSOLUTE_MAX_ENGINES,
    IGFX_ENGINE_INVALID = IGFX_ABSOLUTE_MAX_ENGINES
}IGFX_ENGINE;


//*****************************************************************************
// Enum:  UK_ENGINES_USED
// PURPOSE: A bit flag for each engines used. Used for making some memory optimizations.
//*****************************************************************************
typedef struct UK_ENGINES_USED_REC
{
  union
  {
    struct
    {
     // This sequence of engines (i.e bit positions) must match the corresponding IGFX_ENGINE
enum value,
     UCHAR      RenderEngine      : KM_BIT       (   0 );
     UCHAR      VideoEngine       : KM_BIT       (   1 );
     UCHAR      BltEngine         : KM_BIT       (   2 );
```

```
     UCHAR      VEEngine          : KM_BIT       (   3 );
     UCHAR      VideoEngine2      : KM_BIT       (   4 );
     UCHAR                        : KM_BIT_RANGE (7, 5 );
    };
    UCHAR Value;
  };
} IGFX_ENGINES_USED;

//*****************************************************************************
// Enum:  UK_QUEUE_STATUS
// PURPOSE:  to indicate the status of a Micro Kernel's Work/Submit Queue.
//  TBD: Define what state means what...
//*****************************************************************************
typedef enum
{
    UK_QSTATUS_ACTIVE           = 1,    //Work queue will be serviced if doorbell is owned by app
    UK_QSTATUS_SUSPENDED        ,       //Work queue will not be serviced.
    UK_QSTATUS_CMD_ERROR        ,       //Error in work queue
    UK_QUEUE_STATUS_ENGINE_ID_NOT_USED,
    UK_QSTATUS_SUSPENDED_FROM_ENGINE_RESET, //Suspended due to engine reset. App must consider
all work before WQ tail as in error and resubmit necessary work.
    UK_QSTATUS_INVALID_STATUS
}UK_QUEUE_STATUS;


//*****************************************************************************
// Enum:  UK_CONTEXT_PRIORITY
// PURPOSE:  To indicate the priority of UK context.
//*****************************************************************************
typedef enum
{
    UK_CONTEXT_PRIORITY_KMD_HIGH     = 0,    // KMD priority is only for SKL+
    UK_CONTEXT_PRIORITY_HIGH         = 1,
    UK_CONTEXT_PRIORITY_KMD_NORMAL   = 2,    // KMD priority is only for SKL+
    UK_CONTEXT_PRIORITY_NORMAL       = 3,
    UK_CONTEXT_PRIORITY_ABSOLUTE_MAX_COUNT ,
    UK_CONTEXT_PRIORITY_INVALID            = UK_CONTEXT_PRIORITY_ABSOLUTE_MAX_COUNT
}UK_CONTEXT_PRIORITY;

#define UK_BDW_MAX_PRIORITIES (2)
#define UK_SKL_MAX_PRIORITIES (4)

//*****************************************************************************
// Enum:  UK_ENGINE_EXECLIST_CONTEXT
// PURPOSE:  State of the context - this is per engine.
//*****************************************************************************
typedef struct UK_ENGINE_CONTEXT_STATE_REC
{
    union
    {
        struct
        {
            UCHAR    Submitted     : KM_BIT      (      0);
            UCHAR                  : KM_BIT_RANGE(  7,  1);
        };
        UCHAR SubmitValue;
    };
    union
```

```
    {
        struct
        {
            UCHAR    WaitForDisplayEvent    : KM_BIT      (      0);
            UCHAR    WaitForSemaphore       : KM_BIT      (      1);
            UCHAR    WaitForFaultFulfill    : KM_BIT      (      2);
            UCHAR    CATError               : KM_BIT      (      3);
            UCHAR    ReEnqueueToSubmitQueue : KM_BIT      (      4);
            UCHAR                           : KM_BIT_RANGE(  7,  5);
        };
        UCHAR WaitValue;
    };

} UK_ENGINE_CONTEXT_STATE;

//*****************************************************************************
// Enum:  UK_SCHED_INSTRUCTION_TYPE
// PURPOSE:  These are instructions to the scheduler
//*****************************************************************************
typedef enum
{
    SCHED_INSTRUCTION_BATCH_BUFFER_START = 0x1,
    SCHED_INSTRUCTION_GUC_CMD_PSEUDO     = 0x2,
    SCHED_INSTRUCTION_GUC_CMD_KMD        = 0x3,  // This work item is issued by KMD (Windows
mode)
    SCHED_INSTRUCTION_GUC_CMD_NOOP       = 0x4,  // Treat data following this WI header as no op
(i.e. skip it). uKernel will not read the data.
    SCHED_INSTRUCTION_INVALID                ,  // This must be the last-1 entry
    SCHED_INSTRUCTION_MAX                = 0xF,  // Currently this is the max value
}UK_SCHED_INSTRUCTION_TYPE;

//*****************************************************************************
// Enum:  UK_SCHED_WORK_QUEUE_WORKLOAD_TYPE
// PURPOSE:  Workload type information.
//*****************************************************************************
typedef enum
{
    SCHED_WORK_QUEUE_GENERAL_WORKLOAD   = 0,
    SCHED_WORK_QUEUE_GPGPU_WORKLOAD,
    SCHED_WORK_QUEUE_TOUCH_WORKLOAD,

    SCHED_WORK_QUEUE_MAX_WORKLOADS
}UK_SCHED_WORK_QUEUE_WORKLOAD_TYPE;

//*****************************************************************************
// Enum:  UK_SCHED_WORK_QUEUE_ITEM_HEADER
// PURPOSE: Header for every work item put in the work queue
//*****************************************************************************
typedef struct UK_SCHED_WORK_QUEUE_ITEM_HEADER_REC
{
    union
    {
      struct
       {
        ULONG  WorkInstructionType        : KM_BIT_RANGE( 7,  0); //UK_SCHED_INSTRUCTION_TYPE
        ULONG                             : KM_BIT_RANGE( 9,  8); //MBZ
        ULONG  TargetEngine               : KM_BIT_RANGE(13, 10); //IGFX_ENGINE
        ULONG                             : KM_BIT_RANGE(15, 14); //MBZ
```

```
        ULONG  CommandLengthDwords        : KM_BIT_RANGE(26, 16); //Length in number of Dwords
following this header
        ULONG  DoNotInsertWCFlushWait     : KM_BIT_RANGE(27, 27);
        ULONG  IsPresentWorkload          : KM_BIT_RANGE(28, 28);
        ULONG  WorkloadType               : KM_BIT_RANGE(31, 29); // Enum
UK_SCHED_WORK_QUEUE_WORKLOAD_TYPE
        //ULONG                             : KM_BIT_RANGE(31, 30); //Ignored on production
build. On debug, UMD can use to put tracking number (16 max)
       };
      ULONG Value;
    };

}UK_SCHED_WORK_QUEUE_ITEM_HEADER;

//INFO: Work Queue Item
//
//              Work Queue Item
// V--------------------------------------V
// ****************************************
// *                    *                 *
// * WQ Item Header     *     Commands    *
// ****************************************
// Commands are variable length so not included in any structure.


#define UK_MAX_WORK_QUEUE_ITEMS (128)
//Max size of WI (including header). This is used for security checks.
//If a single WI size increases more than this size, we need to evaluate use cases to increase
this define
#define UK_MAX_WI_SIZE_IN_DW (16)
//*****************************************************************************
// Struct:  UK_KM_APP_DOORBELL_INFO
// PURPOSE: Doorbell cacheline description
//*****************************************************************************
typedef struct UK_KM_APP_DOORBELL_INFO_REC
{
        struct
        {
            ULONG      DoorBellStatus;    // 0 / 1
            ULONG      Cookie;            // Only KMD/uKernel should write 0 here. App should
rollover to 1.
            ULONG      Reserved[14];      // Rest of the cache line.
        };

}UK_KM_APP_DOORBELL_INFO;

//*****************************************************************************
// Struct:  UK_DOORBELL_QW
// PURPOSE: Doorbell QW description
//*****************************************************************************
typedef struct UK_DOORBELL_QW_REC
{
    union
    {
        struct
        {
            ULONG      DoorBellStatus;    // 0 / 1
            ULONG      Cookie;            // Only KMD/uKernel should write 0 here. App should
```

```
rollover to 1.
        };
        ULONG64 QuadPart;
    };

}UK_DOORBELL_QW;
#define KM_DOORBELL_ENABLED (1)
#define KM_DOORBELL_DISABLED (0)


//*****************************************************************************
// Struct:  SCHED_CONTEXT_ENGINE_PRESENCE
// PURPOSE: Per Context Engine Status
//*****************************************************************************
typedef struct SCHED_CONTEXT_ENGINE_PRESENCE_STRUCT
{
   union
   {
       struct
       {
           ULONG        IsContextInRcsSubmitQueue       : KM_BIT      (   0);
           ULONG        IsContextInVcsSubmitQueue       : KM_BIT      (   1);
           ULONG        IsContextInBcsSubmitQueue       : KM_BIT      (   2);
           ULONG        IsContextInVecsSubmitQueue      : KM_BIT      (   3);
           ULONG        IsContextInVcs2SubmitQueue      : KM_BIT      (   4);
           ULONG                                        : KM_BIT_RANGE(30,5);
           ULONG                                        : KM_BIT      (  31);   //Used to be
IsSubmitQueueElementValid
       };
       ULONG    SubmitQueuePresenceValue;
   };

}SCHED_CONTEXT_ENGINE_PRESENCE;


//*****************************************************************************
// Struct:  UK_SCHED_PROCESS_DESCRIPTOR
// PURPOSE: A shared structure between app and uKernel communication.
//*****************************************************************************
typedef struct UK_SCHED_PROCESS_DESCRIPTOR_REC
{
   ULONG              ContextId;
   // Trigger / doorbell address for App - Reserved in uK space
   ADDRESS64          pDoorbell;                          // For Read-Write access by App. Ptr to
UK_KM_APP_DOORBELL_INFO
   ULONG              HeadOffset;                         // Byte Offset - App must not write
here.
   ULONG              TailOffset;                         // Byte Offset - uKernel will not write
here.
   ULONG              ErrorOffsetByte;
   ADDRESS64          WQVBaseAddress;                     // pVirt in app - for use only by
application
   ULONG              WQSizeBytes;
   UK_QUEUE_STATUS    WQStatus;                           // Read by App. Written by uKernel/KMD.
   SCHED_CONTEXT_ENGINE_PRESENCE  ContextEnginePresence; // Read by App. Written by uKernel. A
snapshot of context descriptor's copy. Updates only at Context Complete and WI Process complete.
   UK_CONTEXT_PRIORITY PriorityAssigned;                  // Read only by app
   ULONG               Reserved[4];                       // Reserved
```

```
   //Tracking variables for debug
   ULONG              WorkItemTrackingEnabled;          // Status set by KMD, Read only for
App. For speed reasons, tracking is enabled for all contexts or none.
   ULONG              AppTotalNumberOfWorkItemsAdded;    // App must set to 0 upon return from
OpenGPU, before using it. Written only by Application - Total number of Work Items added to WQ
so far (all Engines, monotonic increase)
   ULONG              AppReserved2[3];

   //uKernel side tracking for debug
   ULONG              TotalWorkItemsParsedByGuC;         // Written by uKernel at the time of
parsing and successfull removed from WQ (implies ring tail was updated).
   ULONG              TotalWorkItemsCollapsedByGuC;      // Written by uKernel if a WI was
collapsed if next WI is the same LRCA (optimization applies only to Secure/KMD contexts)
   ULONG              TotalWorkItemsCancelled;           // Written by uKernel if a WI was
cancelled due to preempt.

   ULONG              IsContextInEngineReset;            // Tells if the context is affected in
Engine Reset. UMD needs to clear it after taking appropriate Action(TBD).
   ULONG              EngineResetSampledWQTail;          // WQ Sampled tail at Engine Reset
Time. Valid only if IsContextInEngineReset = TRUE
   ULONG              EngineResetSampledWQTailValid;     // Valid from Engine reset until all
the affected Work Items are processed.
   ULONG              GuCReserved3[15];                  // Reserved

}UK_SCHED_PROCESS_DESCRIPTOR;

#define KM_MAX_PDP_REGISTERS    (4)
#define KM_PDP_REGISTER_DWORDS  (2)

#define MAX_GUC_GPU_CONTEXTS (1024)
#define MAX_GUC_BLOCKS_OF_16_GPU_CONTEXTS (64)
#define KM_GUC_INVALID_CONTEXT_ID (MAX_GUC_GPU_CONTEXTS)

//*****************************************************************************************
// Struct:  KM_UK_POWER_GATING_TRANSPORT_STATE
// PURPOSE: Transport current programmed and requested power gating states in shared context
//*****************************************************************************************
typedef struct KM_UK_POWER_GATING_TRANSPORT_STATE_REC
{
    KM_UK_RENDER_PWR_CLK_STATE CurrentState;
    KM_UK_RENDER_PWR_CLK_STATE RequestedState;

}KM_UK_POWER_GATING_TRANSPORT_STATE;

//*************************************************************************
// Enum:  KM_PAGE_FAULT_MODE
// PURPOSE: Page fault mode. Must match "Context Descriptor Format" in BSpec
//*************************************************************************
typedef enum KM_PAGE_FAULT_MODE_ENUM
{
    KM_PAGE_FAULT_MODE_FAULT_HANG     = 0x0,
    KM_PAGE_FAULT_MODE_FAULT_WAIT     = 0x1,  //aka Fault and Halt
    KM_PAGE_FAULT_MODE_FAULT_STREAM   = 0x2,
    KM_PAGE_FAULT_MODE_FAULT_CONTINUE = 0x3,  //SKL+

}KM_PAGE_FAULT_MODE;
```

```
//*****************************************************************************
// Enum:  KM_CONTEXT_ADDRESSING_MODE
// PURPOSE: Describes Addressing Mode & Legacy Context. Must match "Context Descriptor Format"
in BSpec
//*****************************************************************************
typedef enum KM_CONTEXT_ADDRESSING_MODE_ENUM
{
    KM_CONTEXT_ADDRESSING_MODE_INVALID               = 0x0,
    KM_CONTEXT_ADDRESSING_MODE_LEGACY_NO_64BIT_VA    = 0x1,  //Legacy & DOESN'T support any SVM
features. (32bit PPGGTT)
    KM_CONTEXT_ADDRESSING_MODE_ADVANCED              = 0x2,  //Advanced context mode and
supports SVM features
    KM_CONTEXT_ADDRESSING_MODE_LEGACY_WITH_64BIT_VA  = 0x3,  //Legacy & DOESN'T support any SVM
features. (64bit PPGTT)

}KM_CONTEXT_ADDRESSING_MODE;

//*****************************************************************************
// Struct: UK_KM_ADDRESS
// PURPOSE: Used to hold ptr to CPU virtual address and Ptr to Ukernel address
//*****************************************************************************
typedef struct UK_KM_ADDRESS_REC
{
    ADDRESS64  pCpuAddress; //Cpu / big core address (virtual)
    ADDRESS32  pUkAddress;  //uKernel address (gfx)
}UK_KM_ADDRESS;

//*****************************************************************************
// Struct:  UK_CONTEXT_ID_MAP
// PURPOSE: To represent the context ID structure and execlist/submit queues
//*****************************************************************************
typedef struct UK_CONTEXT_ID_MAP_REC
{
    union
    {
        struct
        {
            ULONG    ContextIndex          : KM_BIT_RANGE(  19,  0);  // NOTE: This can be index
in the app context pool in direct submission case or LRCA itself in proxy submission case
            ULONG    SubmissionByProxy     : KM_BIT_RANGE(  20, 20);  // If KMD or other context
submitted this context. This means, ContextID is LRCA[31:20]
            ULONG    Reserved              : KM_BIT_RANGE(  22, 21);  // Required by HW
            ULONG    SWCounter             : KM_BIT_RANGE(  28, 23);  // Used for tracking IOMMU
group resubmits (or if submit by proxy is true, lower 6 bits QWIndex).
            ULONG    EngineId              : KM_BIT_RANGE(  31, 29);
        };
        ULONG                        ContextIdDword;
    };

}UK_CONTEXT_ID_MAP;
#define CONTEXT_ID_MAP_SWCOUNTER_NUM_BITS (6)

//*****************************************************************************
// Struct:  SCHED_CONTEXT_DESCRIPTOR_LD_REC
// PURPOSE: The default setting for Context Descriptor Format to be set in UK_ELEMENT_DESC
//*****************************************************************************
typedef struct SCHED_ELEMENT_DESC_REC
{
```

```
   union
    {
        struct
        {
            ULONG    Valid                  : KM_BIT      (      0);
            ULONG    ForcePDRestore         : KM_BIT      (      1);  //Force Page Directory
            ULONG    ForceRestore           : KM_BIT      (      2);
            ULONG    CtxtAddrMode           : KM_BIT_RANGE(  4,  3);  //Addressing Mode & Legacy
Context (KM_CONTEXT_ADDRESSING_MODE)
            ULONG    L3LLCCoherencySupport  : KM_BIT      (      5);
            ULONG    FaultSupportType       : KM_BIT_RANGE(  7,  6);  //Type: KM_PAGE_FAULT_MODE
            ULONG    Privilege              : KM_BIT      (      8);
            ULONG                           : KM_BIT_RANGE( 11,  9);
            ULONG    LRCA                   : KM_BIT_RANGE( 31,  12);
        };
        ULONG Value;
    };

}SCHED_CONTEXT_DESCRIPTOR_LD;

//*****************************************************************************
// Struct:  UK_ELEMENT_DESC
// PURPOSE: The Execution list's element descriptor. Also in Submit Queue
//*****************************************************************************
typedef struct UK_ELEMENT_DESC_REC
{
    union
    {
        SCHED_CONTEXT_DESCRIPTOR_LD ContextDesc;
    };

    union
    {
        struct
        {
            UK_CONTEXT_ID_MAP    ContextId;
        };

        ULONG    HighDW;
    };

}UK_ELEMENT_DESC;

// Bit-flags used for storing information about each saved register
typedef union KM_MMIO_REGISTER_FLAGS
{
    ULONG FlagsValue;
    struct
    {
        ULONG PowerCycle            : KM_BIT_RANGE(0 ,  0);    //Restore at power cycle
        ULONG Masked                : KM_BIT_RANGE(1 ,  1);    //Register is masked
        ULONG EngineReset           : KM_BIT_RANGE(2 ,  2);    //Restore at Engine Reset
        ULONG SaveDefaultValue      : KM_BIT_RANGE(3 ,  3);    //Save default at init time
        ULONG SaveCurrentValue      : KM_BIT_RANGE(4 ,  4);    //Save current value just before
reset/sleep
        ULONG                       : KM_BIT_RANGE(31,  5);    //Reserved
    };
}KM_MMIO_REGISTER_FLAGS;
```

```
#define KM_REGSET_FLAG_FLAGS_NONE                0x00000000
#define KM_REGSET_FLAG_POWERCYCLE                0x00000001 //(__BIT(0))
#define KM_REGSET_FLAG_MASKED                    0x00000002 //(__BIT(1))
#define KM_REGSET_FLAG_ENGINERESET               0x00000004 //(__BIT(2))
#define KM_REGSET_FLAG_SAVE_DEFAULT_VALUE        0x00000008 //(__BIT(3))
#define KM_REGSET_FLAG_SAVE_CURRENT_VALUE        0x00000010 //(__BIT(4))

// Note: this value will need to be increased if any register set exceeds it
#define KM_REGSET_MAX_REGISTERS_PER_SET (25)

#define KM_MMIO_WHITE_LIST_MAX_OFFSETS 12
typedef struct KM_MMIO_WHITE_LIST_REC
{
    ULONG                   MmioStart;
    ULONG                   Offsets[KM_MMIO_WHITE_LIST_MAX_OFFSETS];
    ULONG                   Count;
} KM_MMIO_WHITE_LIST;

// Structure used for storing register offset, current value, default value
// and flags
typedef struct KM_MMIO_REGISTER_REC
{
    ULONG                    Offset;       // Register offset
    ULONG                    Value;        // Current value
    KM_MMIO_REGISTER_FLAGS   Flags;
} KM_MMIO_REGISTER;

// Structure describing register set
typedef struct KM_MMIO_REGISTER_SET_REC
{
    KM_MMIO_REGISTER           Registers[KM_REGSET_MAX_REGISTERS_PER_SET];
    BOOL                       RegisterValuesValid;
    ULONG                      NumberOfRegisters;
} KM_MMIO_REGISTER_SET;

// Structure describing collection of register sets
typedef struct KM_MMIO_REGISTER_STATE_REC
{
    KM_MMIO_REGISTER_SET    GlobalRegisters;                                    // Registers
that are not specific to a node
    KM_MMIO_REGISTER_SET    NodeRegisters[IGFX_ABSOLUTE_MAX_ENGINES];           // Node
specific registers
    KM_MMIO_WHITE_LIST      MMIOWhiteList[IGFX_ABSOLUTE_MAX_ENGINES];           // List of MMIO
registers that are set as non privileged
} KM_MMIO_REGISTER_STATE;

#define UK_MAX_GUC_S3_SAVE_SPACE_PAGES (10)
//*****************************************************************************
// Struct:  KM_GUC_ADDITIONAL_DATA_STRUCTS
// PURPOSE: Pointers to other data structures used for various operations
//*****************************************************************************
typedef struct KM_GUC_ADDITIONAL_DATA_STRUCTS_REC
{
    ADDRESS32               GfxAddressMMIORegisterState;                        //Gfx ptr to
KM_MMIO_REGISTER_STATE. Size is IGFX_ABSOLUTE_MAX_ENGINES (Gfx address must be persistent for
life of GuC)
    ADDRESS32               GfxPtrToGuCSStateSaveBuffer;                        //Max of
```

```
UK_MAX_GUC_S3_SAVE_SPACE_PAGES (Persistent)
    ADDRESS32               GfxGoldenContextLRCA;                                   //LRCA
addresses of golden contexts; (Persistent)
    ADDRESS32               GfxSchedulerPolicies;                                   //GFX addresses
of SCHED_SCHEDULING_POLICIES; (non-Persistent, may be released after initial load), NULL or
valid=0 flag value will cause default policies to be loaded.
    DWORD                   Reserved0[2];
    DWORD                   Reserved1;
    DWORD                   GoldenContextEngStateSizeInBytes[IGFX_ABSOLUTE_MAX_ENGINES];
    DWORD                   Reserved2[2];
    DWORD                   Reserved3[2];
}KM_GUC_ADDITIONAL_DATA_STRUCTS;

#define KM_KBYTES_TO_DWORD_COUNT(a)     ((a * 1024) / sizeof(DWORD))

#define KM_SCHED_MAX_SUBMIT_DATA_ENTRIES (64)
#define KM_SCHED_MAX_PREMPT_REPORT_ENTRIES (16)

//*****************************************************************************
// Struct:  KM_SCHED_SUBMIT_DATA
// PURPOSE: Currently Reserved. (Future: used for extended KMD - GuC Communication)
//*****************************************************************************
typedef struct KM_SCHED_SUBMIT_DATA_REC
{
   ULONG    Reserved[KM_SCHED_MAX_SUBMIT_DATA_ENTRIES];
   ULONG    MaxSubmitDataEntriesCount;        // MBZ - For future use.
   ULONG    SubmitDataFreeIndex;              // MBZ
}KM_SCHED_SUBMIT_DATA;

//*****************************************************************************
// Enum:  KM_SCHED_CONTEXT_AFFECTED_TYPE
// PURPOSE: To indicate how a context was affected when involved in preempt/reset
//*****************************************************************************
typedef enum KM_SCHED_CONTEXT_AFFECTED_TYPE_ENUM
{
    KM_SCHED_CONTEXT_AFFECTED_NONE    = 0x0,
    KM_SCHED_CONTEXT_AFFECTED_IN_HW_EXECUTION     = 0x1,
    KM_SCHED_CONTEXT_AFFECTED_IN_WORK_QUEUE       = 0x2,
    KM_SCHED_CONTEXT_AFFECTED_IN_SUBMIT_QUEUE     = 0x3,
}KM_SCHED_CONTEXT_AFFECTED_TYPE;

//*****************************************************************************
// Enum:  KM_SCHED_AC_REPORT_STATUS
// PURPOSE: To indicate report status of the context when involved in preempt/reset
//*****************************************************************************
typedef enum KM_SCHED_AC_REPORT_STATUS_ENUM
{
    KM_SCHED_AC_REPORT_STATUS_UNKNOWN       = 0x0, //Unknown / un-acked.
    KM_SCHED_AC_REPORT_STATUS_REQUEST_ACKED = 0x1, // Got the request to preempt
    KM_SCHED_AC_REPORT_STATUS_ERROR         = 0x2,
    KM_SCHED_AC_REPORT_STATUS_COMPLETE      = 0x4  //Request to preempt was completed by CSB  =
context complete for preempt context

}KM_SCHED_AC_REPORT_STATUS;
//KM_SCHED_AC_REPORT_STATUS_REACHED_CS    = 0x3, //[Currently not implemented due to perf]
Request to preempt was acked by CS HW (idle -> active/switch)

// Types of Reset supported by GuC
```

```
typedef enum UK_KM_RESET_TYPE_ENUM
{
    UK_KM_RESET_TYPE_MEDIA_RESET,
    UK_KM_RESET_TYPE_WINDOWS_ENGINE_RESET,
    UK_KM_RESET_TYPE_PRIVATE_ENGINE_RESET, // TODO: find out what is needed for this?? To
support this we may need to send UK_KM_RESET_TYPE from KMD
    UK_KM_RESET_TYPE_GUC_INTERNAL_ENGINE_RESET

}UK_KM_RESET_TYPE;

//*****************************************************************************
// Struct:  KM_SCHED_AFFECTED_CONTEXT_REPORT
// PURPOSE: Used by GuC to report back on which contexts were affected by
//          preempt request by KMD
//*****************************************************************************
typedef struct KM_SCHED_AFFECTED_CONTEXT_REPORT_REC
{
    KM_SCHED_AC_REPORT_STATUS       ReportReturnStatus;  // KMD must request report in Options
(see host->GuC interface doc)
                                                         // GuC sets this if a report is
requested and was able to satisfy the request
    ULONG64
ArrAffectedPtrExeclistContext[KM_SCHED_MAX_PREMPT_REPORT_ENTRIES];  // Kernel mode virtual
address (pExeclistContext, from Backpointer in LRCA)
    KM_SCHED_CONTEXT_AFFECTED_TYPE
ArrAffectedType[KM_SCHED_MAX_PREMPT_REPORT_ENTRIES];                // How the context is
affected.
    ULONG
ArrAffectedFenceId[KM_SCHED_MAX_PREMPT_REPORT_ENTRIES];             // This is the submit fence
ID as seen by GuC for debug. Should not use for reporting to OS.
    ULONG                           AffectExeclistCount;
    ULONG                           GuCInternal0;      // KMD should not write to this value as
long as preemption is in progress with GuC
    ULONG                           GuCInternal1;      // KMD should not write to this value as
long as preemption is in progress with GuC

}KM_SCHED_AFFECTED_CONTEXT_REPORT;

#define KM_UK_MAX_PREEMPTION_DEBUG_DWORDS 32

//*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
// typedef:
//        SLPM_PER_CTX_STATE_INFO
//
// Description:
//      This structure provides per-context storage for SLPM related information.
//
// *** IMPORTANT *** Keep in sync with slpmosinterface.h and UkGucKmdInterface.h
//-----------------------------------------------------------------------------*/
typedef struct KM_SLPM_PER_CTX_STATE_INFO_REC
{
    ULONG       EvaluationId;       // The evaluation id that this context last partipated in. 0
== This is a new context.
    ULONG       GtFreqRequest;      // The GT frequency request for this context. Opaque value
of type SLPM_GT_FREQ_REQUEST.
    ULONG       GtFreqRequestId;    // The GT frequency request ID used by SLPM to decide if a
new frequency change needs to be applied. 0 = There is not GT frequency request.
    ULONG       GtConfig;           // The recommended GT_CONFIG.
```

```
    ULONG       GtFreqLRI;          // Encoded GtFreqRequest ready for sending to pcode via LRI.
    UCHAR       AppMinGtFreqBins;   // Minimum GT frequency requested by application/KMD. Value
is in bins (1 = 50MHz). Default = 0x0.
    UCHAR       AppMaxGtFreqBins;   // Maximum GT frequency requested by application/KMD. Value
is in bins (1 = 50MHz). Default = 0xFF.
    UCHAR       GtHwMaxFreq;        // The maximum GT frequency for the recommended GT_CONFIG.

    // Last member of this structure.
    UCHAR       ChangeId;           // KMD/APP increments this value when a change is made to
the data above. SLPM uses this to know if to reread the data in this structure
                                    // or if it's safe to use it's internally cached values.
                                    //     - 0x00 : means that this context is new.
                                    //     - 0xFF : forces SLPM to read the structure all the
time.
}KM_SLPM_PER_CTX_STATE_INFO;

//*****************************************************************************
// Struct:  KM_SCHED_SHARED_CONTEXT_DATA_LOCK
// PURPOSE: Lock of KM_SCHED_SHARED_CONTEXT_DATA so that clients can access
//          this data even after the scheduler has submitted for execution.
//*****************************************************************************
typedef enum KM_SCHED_SHARED_CONTEXT_DATA_CLIENT_ENUM
{
    KM_SCHED_SHARED_CONTEXT_DATA_CLIENT_SLPM = 0,

    KM_SCHED_SHARED_CONTEXT_DATA_CLIENT_MAX
}KM_SCHED_SHARED_CONTEXT_DATA_CLIENT;
typedef struct KM_SCHED_SHARED_CONTEXT_DATA_LOCK_DATA
{
    union
    {
        ULONG Value;
        struct
        {
            UCHAR           Client[KM_SCHED_SHARED_CONTEXT_DATA_CLIENT_MAX];    // The value
will be 1 or 0.
                                                                                // UCHAR is used
rather than a bitfield
                                                                                // so that there
is no need to read
                                                                                // the lock
value into the GUC.
        };
    };
}KM_SCHED_SHARED_CONTEXT_DATA_LOCK;

//*****************************************************************************
// Struct:  KM_SCHED_SHARED_CONTEXT_DATA
// PURPOSE: Used for extended KMD - GuC Communication.
//          This is part of HW context allocation
//*****************************************************************************
typedef struct KM_SCHED_SHARED_CONTEXT_DATA_REC
{
    //Context specific address of dma buffer private data for tracking mid batch preemption
    ULONG       AddOfLastPreemptedPrivateDataLowPart;     // CPU address of private data for
currently executing preemtable workload. Used to save BB preempt data.
    ULONG       AddOfLastPreemptedPrivateDataHighPart;    // If these are non-zero at Preempt
Completion time, then this BB was preempted mid-batch or mid-thread
```

```
    ULONG       AddOfLastPreemptedPrivateDataHighPartTmp; // Used for
WaPipeControlUpperDwordCorruption
    ULONG       :32; // Qword Padding

    ULONG       IsMappedToProxyContext;
    ULONG       ProxyContextID;

    ULONG       EngineResetContextId;

    ULONG       MediaResetCount;

    // Reserved Space
    ULONG     Reserved[8];

    ULONG     UkLastContextSwitchReason;
    ULONG     WasReset;
    ULONG     LRCAGfxAddress;          // Gfx Address of LRCA used as back pointer
    ULONG64   pExeclistContext;        // Back pointer to ExecList Context (KM_EXECLIST_CONTEXT)
    KM_SCHED_SUBMIT_DATA               SchedSubmitData;        // MBZ
    KM_SCHED_AFFECTED_CONTEXT_REPORT   PreemptedContextReport[IGFX_ABSOLUTE_MAX_ENGINES]; // MBZ
prior to calling to GuC for this report TODO: Make this allocation per engine's shared area
    KM_SCHED_AFFECTED_CONTEXT_REPORT   EngResetContextReport[IGFX_ABSOLUTE_MAX_ENGINES];
    KM_UK_POWER_GATING_TRANSPORT_STATE PowerGatingStates; // SSEU payload to be transported to
GuC
    ULONG
ReservedPreemptDebugData[KM_UK_MAX_PREEMPTION_DEBUG_DWORDS];               // The structure is
reserved for Intel use
    ULONG                              KmIdleContextId;
    BOOL                               IsIdleContextValid;

    KM_SCHED_SHARED_CONTEXT_DATA_LOCK  Lock;    // This is used to keep track of when clients
need to prevent
                                               // this data from being destroyed. When not
zero, KMD cannot
                                               // garabage clean this memory from GTT or system
memory.

    // Fields for SLPM
    KM_SLPM_PER_CTX_STATE_INFO          SlpmPerContextState;

}KM_SCHED_SHARED_CONTEXT_DATA;

typedef enum UK_SLEEP_STATE_ENTER_STATUS_ENUM
{
    UK_SLEEP_STATE_ENTER_SUCCESS,
    UK_SLEEP_STATE_ENTER_PREEMPT_TO_IDLE_FAILED,
    UK_SLEEP_STATE_ENTER_ENG_RESET_FAILED,
}UK_SLEEP_STATE_ENTER_STATUS;

//*****************************************************************************
// Struct:  KM_SCHED_WORK_QUEUE_KMD_ELEMENT_INFO
// PURPOSE: Work item for submitting KMD workloads into work queue for GuC.
//*****************************************************************************
typedef struct KM_SCHED_WORK_QUEUE_KMD_ELEMENT_INFO_REC
{
    SCHED_CONTEXT_DESCRIPTOR_LD              ElementLowDW;             // 1 DW - Execlist
context descriptor's lower DW.
```

```
    union
    {
        struct
        {
            ULONG                               : KM_BIT_RANGE(  19,  0);
            ULONG    RingTailQWIndex            : KM_BIT_RANGE(  30, 20);  // QW Aligned, TailValue
<= 2048 (addresses 4 pages max)
            ULONG                               : KM_BIT_RANGE(  31, 31);
        };
        ULONG Value;
    } ElementHighDW;
} KM_SCHED_WORK_QUEUE_KMD_ELEMENT_INFO;

//KMD work Item looks like this:
//          KMD Work Queue Item ( KM_SCHED_WORK_QUEUE_ITEM)
// V-------------------------------------------------------------------------------V
// *********************************************************************************
// |                    |          KmdSubmitInfo (2 DWs)             |             |
// | WQ Item Header     | TailValue  |    LRCA    |  ContextDescLD   |   RsvdDW    |
// *********************************************************************************

//*********************************************************************************
// Struct:  KM_SCHED_WORK_QUEUE_ITEM
// PURPOSE: Work item for submitting KMD workloads into work queue for GuC.
//*********************************************************************************
typedef struct KM_SCHED_WORK_QUEUE_ITEM_REC
{
   UK_SCHED_WORK_QUEUE_ITEM_HEADER      Header;                 // 1 DW
   KM_SCHED_WORK_QUEUE_KMD_ELEMENT_INFO KmdSubmitElementInfo;   // 2 DW - Contains Ring Tail and
LRCA, ContextDescLD
   ULONG                                FenceId;                // 1 DW Fence ID from OS (used
for debug tracing). Bit 31, says its preempt or not
} KM_SCHED_WORK_QUEUE_ITEM;



// If we don't have  KM_MAX_ELEMENTS_PER_EXEC_LIST defined we are
// in non-windows gfx driver environment  (like fulsim/GuC code itself...)
// So, define these stuctures.
#ifndef KM_MAX_ELEMENTS_PER_EXEC_LIST
#ifndef __IGFXFMID_H__

//*********************************************************************************
// Enum:  GTTYPE
// PURPOSE: To GT type of the device
//*********************************************************************************
typedef enum __GTTYPE
{
    GTTYPE_GT1 = 0x0,
    GTTYPE_GT2,
    GTTYPE_GT2_FUSED_TO_GT1,
    GTTYPE_GT2_FUSED_TO_GT1_6, //IVB
    GTTYPE_GTL, // HSW
    GTTYPE_GTM, // HSW
    GTTYPE_GTH, // HSW
    GTTYPE_GT3, // BDW
    GTTYPE_GT4, // BDW
    GTTYPE_GT0, // BDW
```

```
    GTTYPE_GTA, // BXT
    GTTYPE_GTC, // BXT
    GTTYPE_UNDEFINED,//Always at the end.
}GTTYPE;

//*****************************************************************************
// Enum:  GFXCORE_FAMILY
// PURPOSE: Identify the family
//*****************************************************************************
typedef enum
{
    IGFX_GEN8_CORE       = 11,  //Gen8 Family
    IGFX_GEN9_CORE       = 12,  //Gen9 Family
    IGFX_GENNEXT_CORE    = 13,  //GenNext Family
    GFXCORE_FAMILY_FORCE_ULONG = 0x7fffffff
} GFXCORE_FAMILY;

//*****************************************************************************
// Enum:  PRODUCT_FAMILY
// PURPOSE: Identify the product
//*****************************************************************************

typedef enum
{
    IGFX_UNKNOWN        =  0,
    IGFX_BROADWELL      = 16,
    IGFX_CHERRYVIEW     = 17,
    IGFX_SKYLAKE        = 18,
    IGFX_WILLOWVIEW     = 19,
    IGFX_BROXTON        = 20,

    IGFX_GENNEXT               = 0x7ffffffe,
    PRODUCT_FAMILY_FORCE_ULONG = 0x7fffffff
} PRODUCT_FAMILY;

//*****************************************************************************
// Enum:  SKU_STEPPING_ID
// PURPOSE: Stepping ID for use for use with GuC
//*****************************************************************************
typedef enum SKU_STEPPING_ID_ENUM
{
    SKU_STEPPING_ID_A0 = 0x0,
    // Add Ax steppings here

    SKU_STEPPING_ID_B0 = 0x10,
    // Add Bx steppings here

    SKU_STEPPING_ID_C0 = 0x20,
    // Add Cx stepping here

    SKU_STEPPING_ID_D0 = 0x30,
    // Add Dx stepping here

    SKU_STEPPING_ID_E0 = 0x40
    // Add Ex stepping here

}SKU_STEPPING_ID;
```

```
#endif // ifndef __IGFXFMID_H__

/*****************************
KM_EXECLIST_RING_CONTEXT_REGISTER - Must Match BSpec
*****************************/
#define KM_MAX_ELEMENTS_PER_EXEC_LIST  (2)

typedef struct
{
    ULONG   Offset;
    ULONG   Data;
} KM_EXECLIST_RING_CONTEXT_REGISTER;

/*****************************
KM_EXECLIST_RING_CONTEXT - Must Match BSpec
*****************************/

typedef struct KM_EXECLIST_RING_CONTEXT_REC
{
    ULONG   Noop;
    ULONG   RingInfoLriHeader;

    KM_EXECLIST_RING_CONTEXT_REGISTER    ContextControl;
    KM_EXECLIST_RING_CONTEXT_REGISTER    RingHead;
    KM_EXECLIST_RING_CONTEXT_REGISTER    RingTail;
    KM_EXECLIST_RING_CONTEXT_REGISTER    RingBufferStart;
    KM_EXECLIST_RING_CONTEXT_REGISTER    RingBufferControl;
    KM_EXECLIST_RING_CONTEXT_REGISTER    BatchBufferCurrentHead_Udw;
    KM_EXECLIST_RING_CONTEXT_REGISTER    BatchBufferCurrentHead;
    KM_EXECLIST_RING_CONTEXT_REGISTER    BatchBufferState;

    KM_EXECLIST_RING_CONTEXT_REGISTER    SecondBatchBufferAddr_Udw;
    KM_EXECLIST_RING_CONTEXT_REGISTER    SecondBatchBufferAddr;
    KM_EXECLIST_RING_CONTEXT_REGISTER    SecondBatchBufferState;

    KM_EXECLIST_RING_CONTEXT_REGISTER    PostRestoreBb;
    KM_EXECLIST_RING_CONTEXT_REGISTER    IndirectContext;
    KM_EXECLIST_RING_CONTEXT_REGISTER    IndirectContextOffset;

    union
    {
        KM_EXECLIST_RING_CONTEXT_REGISTER    RingModeCCID;          //Only KBL-H
        ULONG   Padding1[2];    //2 Noops
    };

    ULONG   Padding2; // 1 Noop

    ULONG   PdpInfoLriHeader;

    KM_EXECLIST_RING_CONTEXT_REGISTER    ContextTimestamp;

    KM_EXECLIST_RING_CONTEXT_REGISTER    Pdp3_UDW;
    KM_EXECLIST_RING_CONTEXT_REGISTER    Pdp3_LDW;

    KM_EXECLIST_RING_CONTEXT_REGISTER    Pdp2_UDW;
    KM_EXECLIST_RING_CONTEXT_REGISTER    Pdp2_LDW;

    KM_EXECLIST_RING_CONTEXT_REGISTER    Pdp1_UDW;
```

```
    KM_EXECLIST_RING_CONTEXT_REGISTER    Pdp1_LDW;

    KM_EXECLIST_RING_CONTEXT_REGISTER    Pdp0_UDW;
    KM_EXECLIST_RING_CONTEXT_REGISTER    Pdp0_LDW;

    ULONG   Padding3[13];

    ULONG                                PcRegisterLriHeader;
    KM_EXECLIST_RING_CONTEXT_REGISTER    RcsPwrClkState;

    ULONG                                GpGpuCsrBaseAddress[3];

    ULONG   Padding4[9];
} KM_EXECLIST_HW_RING_CONTEXT;


//*****************************************************************************
// Enum:  KM_RING_STATUS
// PURPOSE: To describe status of current ring
//*****************************************************************************

typedef enum KM_RING_STATUS_REC
{
    RING_NOT_PRESENT,
    RING_NOT_ALLOCATED,
    RING_NOT_INITALIZED,
    RING_AVAILABLE,
    RING_CURRENTLY_USED,
    RING_STOPPED,
    RING_STATUS_UNKOWN,
} KM_RING_STATUS;

#endif     // end: #ifndef KM_MAX_ELEMENTS_PER_EXEC_LIST -- above is for non-KMD environments.

//*****************************************************************************
// Struct:  UK_EXECLIST_RING_BUFFER
// PURPOSE: To describe status and access information of current ring buffer for a given EL
context
//*****************************************************************************
typedef struct
{
    KM_RING_STATUS              RingStatus;               // Status
    ADDRESS32                   pExeclistRingContext;     // uKernel Pointer to
pExeclistRingContext

    ADDRESS32                   pRingBegin;               // uKernel address for RingBegin
    ADDRESS32                   pRingEnd;                 // uKernel final byte address that is
valid for this ring
    ADDRESS32                   pNextFreeLocation;        // uKernel address for next location
in ring
                                                          //
    ULONG                       CurrentTailPointerValue;  // Last value written by software for
tracking (just in case HW corrupts the tail in its context)

} UK_EXECLIST_RING_BUFFER;

typedef struct KM_PP_HWSP_REC
{
```

```
    union {
        struct {
            DWORD       Reserved0[4];   // 4 Reserved DWORDS at the beginning
            DWORD       RingHeadPointer;
            DWORD       Reserved1[11];
            DWORD       CumulativeContextRunTime;
            DWORD       SubmitInfo;
            ULONG64     PreemptRequestTimestamp;
            ULONG64     ContextRestoreCompleteTimestamp;
            ULONG64     ContextSaveFinishedTimestamp;
        };

        DWORD       PpHwSp[KM_KBYTES_TO_DWORD_COUNT(4)];
    };
} KM_PP_HWSP;

#define KM_EXECLIST_CONTEXT_PPHWSP_SIZE         PAGE_SIZE
#define KM_EXECLIST_SHARED_DATA_SIZE            PAGE_SIZE

// Macros to get the offset of elements in the execution list context layout
#define KM_EXECLIST_CONTEXT_GET_HHWSP_OFFSET                    ( KM_EXECLIST_SHARED_DATA_SIZE )
#define KM_EXECLIST_CONTEXT_GET_RING_CONTEXT_OFFSET             ( KM_EXECLIST_SHARED_DATA_SIZE +
KM_EXECLIST_CONTEXT_PPHWSP_SIZE )
#define KM_EXECLIST_CONTEXT_GET_ENGINE_STATE_OFFSET
( KM_EXECLIST_CONTEXT_GET_RING_CONTEXT_OFFSET + sizeof(KM_EXECLIST_HW_RING_CONTEXT) )

// Offsets in the BSPEC are from the beginning of the LRCA so they don't include the PPHWSP
// Engine state doesnt' include the Ring Contxt though, so we will take a context DWORD offset
// and remove the Ring Context from it via macro

#define KM_CONTEXT_DWORD_OFFSET_TO_ENGINE_STATE_DWORD_OFFSET(Offset)                \
    (Offset - (sizeof(KM_EXECLIST_HW_RING_CONTEXT) / sizeof(DWORD) ))

// Offsets in the BSPEC are from the beginning of the ring context so they don't include the
PPHWSP
// Engine state doesnt' include the Ring Contxt though, so we will take a context DWORD offset
// and remove the Ring Context from it via macro
#define KM_CONTEXT_DWORD_OFFSET_TO_ENGINE_STATE_DWORD_OFFSET(Offset)                \
    (Offset - (sizeof(KM_EXECLIST_HW_RING_CONTEXT) / sizeof(DWORD) ))

//*****************************************************************************
// Struct:  UK_EXECLIST_CONTEXT
// PURPOSE: The entire execlist context including software and HW information
//*****************************************************************************
typedef struct UK_EXECLIST_CONTEXT_REC
{
   UK_ELEMENT_DESC            ExecElement;
   UK_EXECLIST_RING_BUFFER    RingBufferObj;
   UK_ENGINE_CONTEXT_STATE    State;          // State holds same info as SwitchReason (removed)
   SHORT                      PageFaultCount; // Number of pagefaults outstanding (possible
negative numbers)
   SHORT                      EngineSubmitQueueCount;
} UK_EXECLIST_CONTEXT;


//*****************************************************************************
// Struct:  UK_KM_COMMON_CONTEXT_AREA_DESCRIPTOR
// PURPOSE: A common area for communication from uKernel OS to Scheduler.
```

```
//*****************************************************************************
typedef struct UK_KM_COMMON_CONTEXT_AREA_DESCRIPTOR_REC
{

    ULONG a;

}UK_KM_COMMON_CONTEXT_AREA_DESCRIPTOR;

//*****************************************************************************
// Struct:  UK_KM_CONTEXT_DESCRIPTOR
// PURPOSE: Context descriptor for communicating between uKernel and KM Driver
//*****************************************************************************
typedef struct UK_KM_CONTEXT_DESCRIPTOR_REC
{
   UK_KM_COMMON_CONTEXT_AREA_DESCRIPTOR UkSchedCommonArea;
   ULONG                ContextID;
   ULONG                PASID;
   IGFX_ENGINES_USED    EnginesUsed;
   UK_KM_ADDRESS        DoorbellTriggerAddress;   // The doorbell page's trigger cacheline (Ptr
to UK_KM_APP_DOORBELL_INFO)
   ADDRESS64            DoorbellTriggerAddressGPA;
   USHORT               DoorbellID;               // Do not modify manually once allocated

   UK_EXECLIST_CONTEXT  ExecListContext[IGFX_ABSOLUTE_MAX_ENGINES];

   union
   {
    struct
    {
     UCHAR      IsContextActive         : KM_BIT      (    0); // Is this context actively
assigned to an app
     UCHAR      IsPendingDoorbell       : KM_BIT      (    1);
     UCHAR      IsKMDCreatedContext     : KM_BIT      (    2); // KMD Special WorkItems can be
processed.
     UCHAR      IsKMDPreemptContext     : KM_BIT      (    3); // KMD Context used for
preemption. This flag is used for tracking if preemption workload is complete.
     UCHAR      IsContextEngReset       : KM_BIT      (    4); // Context was part of engine
reset. KMD must take appropriate action (this context will not be resubmitted until this bit is
cleared)
     UCHAR      WqProcessingLocked      : KM_BIT      (    5); // Set it = 1 to prevent other
code paths to do work queue processing as we use sampled values for WQ processing. Allowing
multiple code paths to do WQ processing will cause same workload to execute multiple times.
     UCHAR      IsDoorbellForPCHUse     : KM_BIT      (    6); // If set to 1 at acquire
doorbell time, this doorbell address will be programmed in appropriate XTM register so that PCH
can ring the doorbell
     UCHAR      IsContextTerminated     : KM_BIT      (    7); // If set 1, the context is
terminated by GuC, All the pending work is dropped, its doorbell is evicted and eventually this
context will be removed.
    };
    UCHAR      BoolValues;
   };

   UK_CONTEXT_PRIORITY  Priority;

   ULONG                WQSampledTailOffset;       // Sampled and set during doorbell ISR handler
   ULONG                TotalSubmitQueueEnqueues;  // Global (across all submit queues) pending
enqueues.
```

```
   ADDRESS32            pProcessDescriptor;        // pProcessDescriptor is ptr to
(UK_SCHED_PROCESS_DESCRIPTOR *)
   ADDRESS32            pWorkQueueAddress;
   ULONG                WorkQueueSizeBytes;        // WQ address & size is here because we do
not trust
                                                   // addresses in UK_SCHED_PROCESS_DESCRIPTOR

   //With the new Submit Queue Implementation also we do not do duplicate entries in the submit
queue
   //for a context if it is already present in the submit queue.
   //The flag below are set if a context is present in the submit queue of the engine
   SCHED_CONTEXT_ENGINE_PRESENCE EnginePresence;   // Note that a duplicate copy is in
UK_SCHED_PROCESS_DESCRIPTOR is used for letting app know the status.
                                                   // It is duplicated since app is not trusted
that it will not write to that location and
                                                   // Enginepresence is used in scheduling
decisions.
   IGFX_ENGINES_USED            EngineSchedSuspended;      // EngineSchedSuspended is used in
scheduling decisions. If the bit set, its schedule is suspended for corresponding engine.

   UCHAR                        Reserved0[3];

   ULONG64              Reserved1[1];              // For GuC Internal use. MBZ at init time.

}UK_KM_CONTEXT_DESCRIPTOR, ContextDescriptor;      //* ContextDescriptor this is for Fulsim */

//*****************************************************************************
// Struct:  KM_GUC_CONTEXT_INFO
// PURPOSE: Information about the actual contexts stored in the pool
//*****************************************************************************
typedef struct KM_GUC_POOLED_CONTEXT_INFO_REC
{
   UK_KM_CONTEXT_DESCRIPTOR  Context;               // UK_KM_CONTEXT_DESCRIPTORs
   ULONG64                   AssignedGuCGPUDesc;    // CPU back pointer to the GPU descriptor
(KM_GUC_GPU_DESC) to which this context is assigned to.
}KM_GUC_POOLED_CONTEXT_INFO;


//*****************************************************************************
// Struct:  KM_GUC_CONTEXT_POOL
// PURPOSE: Descriptor record for the pool of UK_KM_CONTEXT_DESCRIPTOR s
//*****************************************************************************
typedef struct KM_GUC_CONTEXT_POOL_DESC_REC
{

   ULONG64  GMMContextPoolHandle;    //Memory Manager's handle to the pool of
KM_GUC_POOLED_CONTEXT_INFO
   ULONG    PoolGFXAddressBegin;     //GFX Address of the pool of KM_GUC_POOLED_CONTEXT_INFO
   ULONG64  PoolCPUAddressBegin;     //CPU address of the pool of KM_GUC_POOLED_CONTEXT_INFO
   ULONG    AllocatedContextCount;   //Total number of Contexts in pool
   ULONG    AssignedContextCount;    //Total number of context assigned to some app (contexts in
use count)
   ULONG    NextFreeContextIndex;    //Next free index in the pool

}KM_GUC_CONTEXT_POOL_DESC;


//*****************************************************************************
```

```
// Struct:  KM_GUC_LOG_INIT_PARAMS
// PURPOSE: Loging Parameters sent via sched_control_data.
//  Maintained as separate structure to allow debug tools to access logs without
//  contacting guc (for when Guc is stuck in ISR)
//*****************************************************************************
typedef struct KM_GUC_LOG_INIT_PARAMS_REC
{

    union
    {
        struct
        {
            ULONG    IsLogBufferValid             : KM_BIT_RANGE(  0,   0); //Is logbuffer valid
            ULONG    NotifyOnLogHalfFull          : KM_BIT_RANGE(  1,   1); //Raises a GuC to
Host interrupt when log buffer is half full.
            ULONG                                 : KM_BIT_RANGE(  2,   2);
            ULONG    AllocatedCountUnits          : KM_BIT_RANGE(  3,   3); // 0 = Log buffer
allocation in Pages, 1 = Log buffer allocation in Megabytes
            ULONG    CrashDumpLogAllocatedCount   : KM_BIT_RANGE(  5,   4); //No. of units
allocated -1 for Crash Dump Log Buffer(MAX 4 Units)
            ULONG    DpcLogAllocatedCount         : KM_BIT_RANGE(  8,   6); //No. of units
allocated -1 for DPC Log Buffer(MAX 8 Units)
            ULONG    IsrLogAllocatedCount         : KM_BIT_RANGE( 11,   9); //No. of units
allocated -1 for ISR Log Buffer(MAX 8 Units)
            ULONG    LogBufferGFXAddress          : KM_BIT_RANGE( 31,  12); //Page aligned
address for log buffer
        };
        ULONG LogDwordValue;
    };

}KM_GUC_LOG_INIT_PARAMS; //Size must not exceed 1 DWORD


//*****************************************************************************
// Struct:  SCHED_CONTROL_DATA_REC
// PURPOSE: Holds the init values of various parameters used by the uKernel
//*****************************************************************************
typedef struct SCHED_CONTROL_DATA_REC
{
    //Dword 0
    union
    {
        struct
        {
            ULONG    NumContextsInPool16Blocks     : KM_BIT_RANGE( 11,   0);  // Num of Contexts
In Pool in blocks of 16, NumContextsInPool16Block = 1 if 16 contexts. 64 if 1024 contexts
allocated.
            ULONG    ContextPoolGFXAddressBegin    : KM_BIT_RANGE( 31,  12);  //Aligned bits
[31:12] of the GFX address where the pool begins
        };
    };

    //Dword 1
    union
    {
            ULONG   ARATHighRegValue     : KM_BIT_RANGE( 31,   0); // The value to be progammed
into ARAT (timer) High register, upon uKernel load
    };
```

```
    //Dword 2
    union
    {
            ULONG   ARATLowRegValue      : KM_BIT_RANGE( 31,   0); // The value to be programmed
into ARAT Low register,  upon uKernel load. 0-> selects a default value by ukernel
    };

    //Dword 3
    //Device details on which the uKernel is loaded.
    //Specified by the driver to reduce code in uKernel.
    union
    {
        struct
        {
            ULONG  GfxGtType             : KM_BIT_RANGE( 6,  0);  //Enum: GTTYPE
            ULONG  GfxCoreFamily         : KM_BIT_RANGE(12,  7);  //Enum: GFXCORE_FAMILY
            ULONG  usRevId               : KM_BIT_RANGE(28, 13);  //GFX rev ID. Set 0 unless
there is a stepping specific WA to be done in uKernel
            ULONG                        : KM_BIT_RANGE(31, 29);
        };
        ULONG DeviceDetailsDword;
    };

    //Dword 4
    // Log buffer setup
    KM_GUC_LOG_INIT_PARAMS  LogInitParams;

    //Dword 5
    //Page fault control
    union
    {
        struct
        {
            ULONG MinPageFaultsBeforeSwitch       : KM_BIT_RANGE (10, 0);  //Value should be
(num faults - 1.) 0 -> 1 fault, 4 -> 5 faults
            ULONG                                 : KM_BIT_RANGE (12,11);
            ULONG MinFaultsServicedBeforeRetry    : KM_BIT_RANGE (23,13);  //Value should be
(num faults - 1.) 0 -> 1 fault, 4 -> 5 faults
            ULONG                                 : KM_BIT_RANGE (25,24);
            ULONG                                 : KM_BIT_RANGE (31,29);
        };
        ULONG PageFaultControlDword;
    };

    //Dword 6
    //GuC uKernel Specific workarounds that needs to be applied. WA must have a HSD/traceable
reference!
    union
    {

        struct
        {
            ULONG     WaPreSiDummyGuC2HostOnCSInterrupt     : KM_BIT      (    0);
            ULONG     WaWorkaround1                         : KM_BIT      (    1);
            ULONG     WaDisableDefaultSchedulerPolicy       : KM_BIT      (    2);
            ULONG     WauKernelLoadedByDriver               : KM_BIT      (    3);
            ULONG     WaDisableDummyAllEngineFaultFix       : KM_BIT      (    4);
            ULONG     WaDisableLiteRestoreOptimization      : KM_BIT      (    5);
```

```
            ULONG     WaEnableMultipleAffectCountWrites     : KM_BIT      (    6);
            ULONG     WaReadBackAffectedCountAfterWrite     : KM_BIT      (    7);
            ULONG     WaUseGuc2HostFencingForSQFlush        : KM_BIT      (    8);
            ULONG     WaReadBackAffectedInfoAfterWrite      : KM_BIT      (    9);
            ULONG     WaEnableGoMsgToGAMDuringCPD           : KM_BIT      (    10);
            ULONG     WaDisable2ElemSubmission              : KM_BIT      (    11);
            ULONG     WaDisableDOPRenderClkGatingAtSubmit   : KM_BIT      (    12);
            ULONG     WaDummyWriteBeforeFenceCycle          : KM_BIT      (    13);
            ULONG     WaDisableSRAMRestoreDisable           : KM_BIT      (    14);
            ULONG     WaEnableGoMsgAckDuringCPD             : KM_BIT      (    15);
            ULONG                                           : KM_BIT_RANGE(31, 16);
        };
        ULONG WorkaroundDW;
    };

    //Dword 7
    //GuC uKernel Specific features
    union
    {
        struct
        {
            ULONG     FtrGuCEnableVCS2Engine         : KM_BIT      (    0);  // For GT3 configs.
            ULONG     FtrKmdGuCSubmissions           : KM_BIT      (    1);
            ULONG     FtrFeature2                    : KM_BIT      (    2);
            ULONG     FtrGucPowerGating              : KM_BIT      (    3);  // For power gating
            ULONG     FtrDisableGuCScheduler         : KM_BIT      (    4);  // Prevent GuC
Scheduler from initializing
            ULONG     FtrEnablePreemptionDataLogging : KM_BIT      (    5);  // Enable tracking
and logging preemption data
            ULONG     FtrEnableGuCPAVPControl        : KM_BIT      (    6);  // Enable GuC PAVP
Control
            ULONG     FtrEnableGuCSlpm               : KM_BIT      (    7);  // Enable SLPM in
GuC
            ULONG     FtrEnableEngineResetOnPreemptFailure      : KM_BIT      (    8);  //
Enable GuC Internal Reset feature.
            ULONG     FtrNotApplicable                          : KM_BIT      (    9);
            ULONG     FtrEnableAsyncFWInEngSchedule             : KM_BIT      (    10); // For
disabling tight poll on engine wake during schedule
            ULONG                                               : KM_BIT_RANGE(31, 11);
        };
        ULONG FeatureDword;
    };

    //Dword 8
    // Logging Verbosity Select and other tracking data control

    union
        {
            struct
            {
                ULONG     LoggingVerbosity               : KM_BIT_RANGE(3,    0);  // defines
the level of logging in uKernel. 0 -> logging only important events 3 -> logging at every step
of execution
                ULONG     LogOutputSelection             : KM_BIT_RANGE(5,    4);  //  For log
output to NPK/LogBuffer or both
                ULONG     LoggingDisabled                : KM_BIT_RANGE(6,    6);  // Flag to
indicate logging is enabled or not. Keeping the bit clear as logging enabled for backward
comaptibility with Ukernel.
```

```
                ULONG     ProfileEnabled                : KM_BIT_RANGE(7,    7);  // Flag to
indicate profile logs are enabled.
                ULONG     EnableWorkQueueItemTracking    : KM_BIT_RANGE(8,    8);  // This flag
enables counting work items seen by each context. This mainly for debug.
                ULONG     AdditionalDataStructsEnabled   : KM_BIT_RANGE(9,    9);  // Note:
Additional data structs is *not* optional for most POR use cases of GuC.
                ULONG     DefaultLoggingDisabled         : KM_BIT_RANGE(10, 10); //Enable
Logging Critical Messages always
                ULONG     GfxAddressAdditionalDataStructs: KM_BIT_RANGE(31, 11);  // Gfx Ptr to
KM_GUC_ADDITIONAL_DATA_STRUCTS

            };
        };

    //Dword 9
    union
    {
        struct
        {
            ULONG    GfxAddressKmSharedData;
        };
    };

}SCHED_CONTROL_DATA;

//Enum to determine what mode the scheduler is in.
typedef enum UK_SCHEDULER_MODE_REC
{
    UK_SCHEDULER_MODE_NORMAL, //Standard scheduling mode no stalling
    UK_SCHEDULER_MODE_STALL_PER_PRESENT, //Stalls scheduling on all engines WI's whose header
has IsPresentWorkload is to be scheduled
    UK_SCHEDULER_MODE_STALL_IMMEDIATE, //Stalls scheduling on all engines
    UK_SCHEDULER_MODE_STALL_IMMEDIATE_PREEMPT_TO_IDLE //Stalls scheduling on all engines and
preempts any executing workloads
}UK_SCHEDULER_MODE;

typedef struct SCHED_SCHEDULING_POLICY_FLAGS_REC
{
    union
    {
        struct
        {
            ULONG    ResetEngineUponPreemptFailure : KM_BIT_RANGE(0, 0);    // Should we reset
engine when preemption failed within its time quantum
            ULONG    PreemptToIdleOnQuantumExpiry  : KM_BIT_RANGE(1, 1);    // Should we preempt
to idle unconditionaly for the execution quantum expiry.
            ULONG                                  : KM_BIT_RANGE(31, 2);
        };
        ULONG PolicyDword;
    };

}SCHED_SCHEDULING_POLICY_FLAGS;


typedef struct SCHED_SCHEDULING_POLICY_REC
{
    ULONG                        ExecutionQuantum;                 // Time for one workload to
execute. (micro seconds)
```

```
    ULONG                           RSVD1;                                  // MaxQuantumCredits Usage
TBD: No. of schedule quantums before this queue has to give up even if it has work.
    ULONG                           WaitForPreemptionCompletionTime;    // Time to wait for a
preemption request to completed before issuing a reset. (micro seconds).
    ULONG                           QuantumUponFirstFaultTime;          // How much time to allow
to run after the first fault is observed. Then preempt afterwards. (micro seconds)
    SCHED_SCHEDULING_POLICY_FLAGS PolicyFlags;
    ULONG Rsvd;
    ULONG                           Rsvd2;
}SCHED_SCHEDULING_POLICY;


typedef struct SCHED_SCHEDULING_POLICIES_REC
{
    SCHED_SCHEDULING_POLICY PerSubmitQueuePolicy[UK_CONTEXT_PRIORITY_ABSOLUTE_MAX_COUNT]
[IGFX_ABSOLUTE_MAX_ENGINES];
    ULONG                   DPCPromoteTime; //In MicroSec. How much time to allow before DPC
processing is called back via interrupt (to prevent DPC queue drain starving). Typically 1000s
of micro seconds (example only, not granularity).
    ULONG                   IsValid;        //Must be set to take these new values.
    ULONG                   MaxNumWorkItemsPerDpcCall; // Number of WIs to process per call to
process single. Process Single could have a large Max Tail value which may keep CS idle. Process
MaxNumWorkItemsPerDpcCall WIs and try fast schedule.
    ULONG                   Rsvd[19];             //Other global policies that may be used in
future
}SCHED_SCHEDULING_POLICIES;




// Datastructures to keep the status of Logging Buffer.

typedef enum UK_LOG_BUFFER_TYPE_ENUM
{
    UK_ISR_LOG_BUFFER,
    UK_DPC_LOG_BUFFER,
    UK_CRASH_DUMP_LOG_BUFFER,
    UK_MAX_LOG_BUFFER,
}UK_LOG_BUFFER_TYPE;

typedef enum UK_LOG_VERBOSITY_ENUM
{
    UK_LOG_VERBOSITY_LOW     = 0x00,
    UK_LOG_VERBOSITY_MED     = 0x01,
    UK_LOG_VERBOSITY_HIGH    = 0x02,
    UK_LOG_VERBOSITY_ULTRA   = 0x03,
    UK_LOG_VERBOSITY_MAX
}UK_LOG_VERBOSITY;

//This enum controls the type of logging output. Can be changed dynamically using Host2GuC
interface.
typedef enum LOGOUTPUT_SELECTION_REC
{
    LOGOUTPUT_LOGBUFFER_ONLY    = 0x0,
    LOGOUTPUT_NPK_ONLY          = 0x1,
    LOGOUTPUT_LOGBUFFER_AND_NPK = 0x2,
    LOGOUTPUT_MAX
```

```
}LOGOUTPUT_SELECTION;

#define UK_KM_LOG_BUFFER_KMD_VERSION 1

// Filled by KMD except Version and Marker are initialized by uKernel
typedef struct UK_KM_SCHED_STATISTICS_REC
{
    ULONG               Marker[2];                  // Marks the beginning of statistics buffer
    ULONG               Version;
    ULONG               Reserved[5];

    ULONG               GlobalDoorbellRingCount;
    ULONG               GlobalPreemptRequestCount;

}UK_KM_SCHED_STATISTICS;

// Filled by KMD except Version and Marker are initialized by uKernel
typedef struct UK_KM_LOG_BUFFER_STATE_REC
{
    ULONG               Marker[2];                  // Marks the beginning of Buffer Flush(set
by uKernel at Log Buffer Init)
    ULONG               LogBufRdPtr;                // This is the Last Byte Offset Location
that was read by KMD. KMD will write to this and uKernel will read this.
    ULONG               LogBufWrPtr;                // This is the Byte Offset Location that
will be written by uKernel.
    ULONG               LogBufSize;                 // Log Buffer size (set by KMD)
    ULONG               SampledLogBufWrptr;         // This is written by ukernel when it sees
the log buffer becoming half full and KMD writes this value in the Log file to avoid stale data.
    union
    {
        struct
        {
            ULONG    LogBufFlushToFile : KM_BIT_RANGE  (0,0);   //  uKernel sets this when log
buffer is half full or when a forced flush has been requested through host2guc. uKernel will
send GUC2HOST only if this bit is cleared. This is to avoid unnecessary interrupts from GuC.
            ULONG    BufferFullCount   : KM_BIT_RANGE  (4,1);   //  uKernel increments this when
log buffer overflows
            ULONG                      : KM_BIT_RANGE  (31,5);
        };
        ULONG    LogBufFlags;
    };

    ULONG               Version;
}UK_KM_LOG_BUFFER_STATE;

//*****************************************************************************
// Struct:  SCHED_PREEMPT_OPTIONS
// PURPOSE: Options for Engine Preempt Host 2 GuC Actions
//*****************************************************************************
typedef struct SCHED_PREEMPT_OPTIONS_REC
{
    //Dword 0
    union
    {
        struct
        {
            ULONG    AttemptScheduleBeforeReturn   : KM_BIT      (       0);  // After adding
the Preempt Context ID to submit queue, attempt to schedule it to Runlist submit port before
```

```
returning from the call.
            ULONG                                   : KM_BIT      (       1);  // MBZ
            ULONG    PreemptTargetWorkQueueItems    : KM_BIT      (       2);  // For the target
context ID, drop the work items in WQ that are targeted to this engine.
            ULONG    DropSubmitQueueItemsOnPreempt : KM_BIT      (       3);  // For the target
priority, if flag is set, this flag drops the submit queue. Otherwise, reenqueues them back into
submit queue
            ULONG    ReportAffectedContexts         : KM_BIT      (       4);  // Applicable only
of following KMD submissions. Report back the ptr to Execlist contexts in an array
            ULONG                                   : KM_BIT_RANGE(31    ,5);
        };
        ULONG   OptionsDW;
    };
}SCHED_PREEMPT_OPTIONS;

typedef struct SCHED_ENG_RESET_OPTIONS_REC
{
    //Dword 0
    union
    {
        struct
        {
            ULONG    Reserved                       : KM_BIT      (       0);  // After adding
the Preempt Context ID to submit queue, attempt to schedule it to Runlist submit port before
returning from the call.
            ULONG                                   : KM_BIT      (       1);  // MBZ
            ULONG    Reserved1                      : KM_BIT      (       2);  // For the target
context ID, drop the work items in WQ that are targeted to this engine.
            ULONG    reserved2                      : KM_BIT      (       3);  // For the target
priority, if flag is set, this flag drops the submit queue. Otherwise, reenqueues them back into
submit queue
            ULONG    ReportAffectedContexts         : KM_BIT      (       4);  // Applicable only
of following KMD submissions. Report back the ptr to Execlist contexts in an array
            ULONG                                   : KM_BIT_RANGE(31    ,5);
        };
        ULONG   OptionsDW;
    };
}SCHED_ENG_RESET_OPTIONS;

/* This Status will be programmed in C1BC - SOFT_SCRATCH_15_REG */
typedef struct UK_GUC_TO_HOST_MESSAGE_RECORD
{
   union
    {
        struct
        {
            ULONG UkInitDone                        : KM_BIT_RANGE(0,  0 );
            ULONG CrashDumpPosted                   : KM_BIT_RANGE(1,  1 );
            ULONG IommuPageFaulted                  : KM_BIT_RANGE(2,  2 );
            ULONG FlushLogBufferToFile              : KM_BIT_RANGE(3,  3 );
            ULONG PreemptRequestOldPreemptPending   : KM_BIT_RANGE(4,  4 );  // Preempt request
was unable to start because an old preempt was already pending. Wait for some time.
            ULONG PreemptRequestTargetContextBad    : KM_BIT_RANGE(5,  5 );  // Input Target
context ID was incorrect / null.
            ULONG PreemptRequestInfoTargetHasWork   : KM_BIT_RANGE(6,  6 );  // Information flag
that indicates either submit queue has pending work. This is to help debug.
            ULONG SleepEntryInProgress              : KM_BIT_RANGE(7,  7 );  // s3/s4 entry in
progress.
```

```
            ULONG GuCInDebugHalt                    : KM_BIT_RANGE(8,  8 );  // Debug infinite
loop is hit inserted in ukernel and is hit. This will halt kmd to carry on debug.
            ULONG GuCReportEngineResetContextId     : KM_BIT_RANGE(9,  9 );
            ULONG HuCAuthenticationFailed           : KM_BIT_RANGE(10, 10);
            ULONG Reserved1                         : KM_BIT_RANGE(15, 11);
            ULONG GpaToHpaXlationError              : KM_BIT_RANGE(16, 16);
            ULONG DoorbellIDAllocationError         : KM_BIT_RANGE(17, 17);
            ULONG DoorbellIDAllocationInvalidCtxID  : KM_BIT_RANGE(18, 18);
            ULONG SetBackLightDutyCycle             : KM_BIT_RANGE(19, 19);  // Revisit later
and remove if not needed by dpst/turbo.
            ULONG ForceWakeTimedOut                 : KM_BIT_RANGE(20, 20);
            ULONG ForceWakeTimeOutCounter           : KM_BIT_RANGE(22, 21);  // Max 4, just as
an indication that time out happened multiple times
            ULONG PreemptRequestProcessDpcError     : KM_BIT_RANGE(23, 23);
            ULONG IommuCatPageFaulted               : KM_BIT_RANGE(24, 24);
            ULONG Reserved3                         : KM_BIT_RANGE(28, 25);
            ULONG DoorbellIDReleaseError            : KM_BIT_RANGE(29, 29);
            ULONG UkException                       : KM_BIT_RANGE(30, 30);
            ULONG BootromHalt                       : KM_BIT_RANGE(31, 31);
        };
        ULONG Dw;
    };

}UK_GUC_TO_HOST_MESSAGE;

// Note trigger is a write only register - it has no backing storage.
#define KM_GEN8_REG_GUC_HOST2GUC_INTERRUPT (0xC4C8)
#define KM_GEN8_RING_HOST2GUC_INTERRUPT (0x1)
#define KM_GEN8_REG_GUC_HOST2GUC_IIR  (0xC590)

#define KM_GUC_SLEEP_STATE_ENTER_STATUS (0xc1b8)

/* This Action will be programmed in C180 - SOFT_SCRATCH_O_REG */
typedef enum
{
    HOST2GUC_ACTION_DEFAULT                             = 0x0,
    HOST2GUC_ACTION_REQUEST_INIT_DONE_INTERRUPT         = 0x1,
    HOST2GUC_ACTION_REQUEST_PREEMPTION                  = 0x2,
    HOST2GUC_ACTION_REQUEST_ENGINE_RESET                = 0x3,
    HOST2GUC_ACTION_PAUSE_SCHEDULING                    = 0x4,  //Only for Dom0 type KMD
    HOST2GUC_ACTION_RESUME_SCHEDULING                   = 0x5,
    HOST2GUC_ACTION_SAMPLE_FORCEWAKE_FEATURE_REGISTER   = 0x6,  //SKL+ in case OS changes this
register to enable/disable it after GuC init, GuC should resample it
    HOST2GUC_ACTION_ALLOCATE_DOORBELL                   = 0x10,
    HOST2GUC_ACTION_DEALLOCATE_DOORBELL                 = 0x20,
    HOST2GUC_ACTION_LOG_BUFFER_FILE_FLUSH_COMPLETE      = 0x30,
    HOST2GUC_ACTION_CACHE_CRASH_DUMP                    = 0x200,       // Cache address from
host to send crash dump
    HOST2GUC_ACTION_DEBUG_RING_DB                       = 0x300,       // For debugging, ring a
doorbell via host 2 guc interrupt
    HOST2GUC_ACTION_PERFORM_GLOBAL_DEBUG_ACTIONS        = 0x301,
    HOST2GUC_ACTION_FORCE_LOGBUFFERFLUSH                = 0x302,       // Force a flush of the
log buffer in its current state by sending guc2host
    HOST2GUC_ACTION_UK_LOG_VERBOSITY_LOGOUTPUT_SELECT   = 0x400,       // Select Logging
Verbosity (may be raised by kmdebugtools) OR logging output selection
    HOST2GUC_ACTION_RESET_ENGINE                        = 0x500,       // Host OS calling for
reset of a specific command streamer engine
    HOST2GUC_ACTION_ENTER_S_STATE                       = 0x501,       // S3/S4 state
```

```
    HOST2GUC_ACTION_EXIT_S_STATE                        = 0x502,       // S3/S4 state
    HOST2GUC_ACTION_PREPARE_FOR_TDR                     = 0x503,       // State before. This is
to ensure GuC does not touch TDR registers.
    HOST2GUC_ACTION_SET_SCHEDULING_MODE                 = 0x504,       // Set scheduling mode.
Currently used by DCC/DFPS
    HOST2GUC_ACTION_PM_TEST_BASE                        = 0x1000,
    HOST2GUC_ACTION_PM_TEST_END                         = 0x2000,
    // Action for PC: 0x3000-0x3FFF
    HOST2GUC_ACTION_PC_TURBO_REQUEST                    = 0x3001,
    HOST2GUC_ACTION_PC_DPST_REQUEST                     = 0x3002,
    HOST2GUC_ACTION_PC_SLPM_REQUEST                     = 0x3003,
    // End action for PC
    HOST2GUC_ACTION_AUTHENTICATE_HUC                    = 0x4000,
    HOST2GUC_ACTION_MAX                                 = 0xF0000000
}HOST2GUC_ACTION;

#define __MAKE_HOST2GUC_STATUS(a) (HOST2GUC_ACTION_MAX+a)
typedef enum
{
    HOST2GUC_STATUS_SUCCESS                     = __MAKE_HOST2GUC_STATUS(0x0),
    HOST2GUC_STATUS_ALLOCATE_DOORBELL_FAIL      = __MAKE_HOST2GUC_STATUS(0x10),
    HOST2GUC_STATUS_DEALLOCATE_DOORBELL_FAIL    = __MAKE_HOST2GUC_STATUS(0x20),
    HOST2GUC_STATUS_FULSIM_TEST_FAIL            = __MAKE_HOST2GUC_STATUS(0x80),
    HOST2GUC_STATUS_LOG_ALLOCATION_TOO_SMALL    = __MAKE_HOST2GUC_STATUS(0x80),     // Not
enough space allocated in host for entire log
    HOST2GUC_STATUS_LOG_HOST_ADDRESS_NOT_VALID  = __MAKE_HOST2GUC_STATUS(0x80),     // Location
not in 48 bits, SRAM specified as type etc
    HOST2GUC_STATUS_LOG_DMA_FAILED              = __MAKE_HOST2GUC_STATUS(0x80),     // DMA
transfer failed. Upper word will have error code from HW
    HOST2GUC_STATUS_UNKNOWN_ACTION              = __MAKE_HOST2GUC_STATUS(0x100),
    HOST2GUC_STATUS_GENERIC_FAIL                = __MAKE_HOST2GUC_STATUS(0x0000F000)
}HOST2GUC_STATUS;

#define KM_MAX_NUM_INPUT_DWORDS (14)
#define KM_MAX_NUM_OUTPUT_DWORDS (16)
//*****************************************************************************
// Struct:  KM_GUC_HOST_TO_GUC_REQUEST
// PURPOSE: Used to pass info for doing a Host 2 Guc call within Driver
//*****************************************************************************
typedef struct KM_GUC_HOST_TO_GUC_REQUEST_REC
{
    ULONG           Rsvd;
    ULONG64         HwDeviceHandle;        //In: HW_DEVICE_EXTENSION as ULONG64
    ULONG          *pInput;                //In: Ptr to array of dwords for input
    ULONG           NumOfInputDwords;      //In: No. of Dwords in array
    ULONG           WriteDisableMask;      //In: Bitwise mask to disable write to registers for
each Input dword. when BIT(x) is set, Input[x] won't be written.
    ULONG          *pOutput;               //In/Out: Out of read registers after successful
host2guc.  Also space must be allocated by caller for this array for the number of Dwords.
    ULONG           NumOfOutputDwords;     //In: No. of Dwords to read
    LONG            ReturnStatus;          //Out: NTSTATUS of H2G request TODO: Remove this
windows specific status.
} KM_GUC_HOST_TO_GUC_REQUEST;


//*****************************************************************************
//
//                  Power Managment Module: SLPC
```

```
//
//*****************************************************************************

#ifndef _SLPC_KMD_INTERFACE_H_
#define _SLPC_KMD_INTERFACE_H_

/*************************************************************
OVERVIEW
========
This is the glue between the SLPM core layer and the HAL layer
that is provided by the OS under which the core is running (GuC,
KMD).

It defines the functions that need to be provided by the HAL
layer to the SLPM core layer and visa versa.

It also defines all structures that are shared between the
SLPM core layer and the layer above the HAL (the Host).
*************************************************************/

#ifdef __cplusplus
extern "C" {
#endif
    //A way to force compile check in GCC/ICC.
    // It creates an unique typedef.
#ifndef CASSERT
#define _impl_PASTE(a,b) a##b
#define _impl_CASSERT_LINE(predicate, line, file)  typedef char
_impl_PASTE(assertion_failed_##file##_,line)[2*!!(predicate)-1];
#define CASSERT(predicate, file) _impl_CASSERT_LINE(predicate,__LINE__, file)
#endif

#define SLPM_BIT_RANGE(endbit, startbit)     ((endbit)-(startbit)+1)
#define SLPM_BIT(bit)                        (1)

    //------------------------------------------------------------
    //STATUS CODES
    //------------------------------------------------------------
    typedef enum _SLPM_STATUS
    {
        SLPM_STATUS_OK                              = 0,
        SLPM_STATUS_ERROR                           = 1,
        SLPM_STATUS_ILLEGAL_COMMAND                 = 2,
        SLPM_STATUS_INVALID_ARGS                    = 3,
        SLPM_STATUS_INVALID_PARAMS                  = 4,
        SLPM_STATUS_INVALID_DATA                    = 5,
        SLPM_STATUS_OUT_OF_RANGE                    = 6,
        SLPM_STATUS_NOT_SUPPORTED                   = 7,
        SLPM_STATUS_NOT_IMPLEMENTED                 = 8,
        SLPM_STATUS_NO_DATA                         = 9,
        SLPM_STATUS_EVENT_NOT_REGISTERED            = 10,
        SLPM_STATUS_REGISTER_LOCKED                 = 11,
        SLPM_STATUS_TEMPORARILY_UNAVAILABLE         = 12,
        SLPM_STATUS_VALUE_ALREADY_SET               = 13,
        SLPM_STATUS_VALUE_ALREADY_UNSET             = 14,
        SLPM_STATUS_VALUE_NOT_CHANGED               = 15,
        SLPM_STATUS_MEMIO_ERROR                     = 16,
        SLPM_STATUS_EVENT_QUEUED_REQ_DPC            = 17,
```

```
        SLPM_STATUS_EVENT_QUEUED_NOREQ_DPC          = 18,
        SLPM_STATUS_NO_EVENT_QUEUED                 = 19,
        SLPM_STATUS_OUT_OF_SPACE                    = 20,
        SLPM_STATUS_TIMEOUT                         = 21,
        SLPM_STATUS_NO_LOCK                         = 22
    } SLPM_STATUS;
    //-----------------------------------------------------------------
    //END OF STATUS CODES
    //-----------------------------------------------------------------

    //-----------------------------------------------------------------
    //ALL EVENTS THAT THE HAL WILL SEND TO SLPM CORE.
    //WHEN PASSING EVENTS TO THE ISR HANDLING INTERFACE (SLPM_PFN_EVENT_ISR)
    //15 DWORD PAYLOAD CAN BE PASSED IN. FOR EACH EVENT BELOW, ARGUMENTS ARE
    //DEFINED IN COMMENTS.
    //-----------------------------------------------------------------
    typedef enum _SLPM_EVENT_ID
    {
        SLPM_KMD_EVENT_RESET                        = 0,        // Args[0] = lsb of
SLPM_KMD_SHARED_DATA, Args[1] = msb of SLPM_KMD_SHARED_DATA, [OPTIONAL] Args[2] = Bitfield of
SLPM_PROFILE_RESET_FLAGS
        SLPM_KMD_EVENT_SHUTDOWN                     = 1,        // Args[0] = lsb of
SLPM_KMD_SHARED_DATA, Args[1] = msb of SLPM_KMD_SHARED_DATA, [OPTIONAL] Args[2] = Bitfield of
SLPM_PROFILE_RESET_FLAGS
        SLPM_KMD_EVENT_PLATFORM_INFO_CHANGE         = 2,        // Args[0] = lsb of
SLPM_KMD_SHARED_DATA, Args[1] = msb of SLPM_KMD_SHARED_DATA
        SLPM_KMD_EVENT_DISPLAY_MODE_CHANGE          = 3,        // Args[0] =
SLPM_KMD_DISPLAY_MODE_EVENT_PARAM.GlobalData, Args[1..5] =
SLPM_KMD_DISPLAY_MODE_EVENT_PARAM.PerPipeData[1..5].Data,
        SLPM_KMD_EVENT_FLIP_COMPLETE                = 4,        // Args[0] =
KM_FPS_TRACKING_FLIP_INFO
        SLPM_KMD_EVENT_QUERY_TASK_STATE             = 5,
        SLPM_KMD_EVENT_PARAMETER_SET                = 6,        // Args[0] = SLPM_KMD_PARAM_ID,
Args[1] = Value
        SLPM_KMD_EVENT_PARAMETER_UNSET              = 7,        // Args[0] = SLPM_KMD_PARAM_ID
        SLPM_END_KMD_EVENTS
    } SLPM_EVENT_ID;
    //-----------------------------------------------------------------
    //END OF EVENTS
    //-----------------------------------------------------------------

    //-----------------------------------------------------------------
    //ALL PARAMS - this is used to start parameter values in an array
    //             and to use a bitfield at the start of the array to
    //             flag the parameters that are set.
    //-----------------------------------------------------------------
    typedef enum _SLPM_KMD_PARAM_ID
    {
        // Please explicitly define values to reduce chance of mismatch
        SLPM_KMD_PARAM_TASK_ENABLE_GTPERF                           = 0,
        SLPM_KMD_PARAM_TASK_DISABLE_GTPERF                          = 1,
        SLPM_KMD_PARAM_TASK_ENABLE_BALANCER                         = 2,
        SLPM_KMD_PARAM_TASK_DISABLE_BALANCER                        = 3,
        SLPM_KMD_PARAM_TASK_ENABLE_DCC                              = 4,
        SLPM_KMD_PARAM_TASK_DISABLE_DCC                             = 5,
        SLPM_KMD_PARAM_GLOBAL_MIN_GT_UNSLICE_FREQ_MHZ               = 6,    // Special values
defined in enum SLPM_GT_FREQ_MHZ
        SLPM_KMD_PARAM_GLOBAL_MAX_GT_UNSLICE_FREQ_MHZ               = 7,    // Special values
```

```
defined in enum SLPM_GT_FREQ_MHZ
        SLPM_KMD_PARAM_GLOBAL_MIN_GT_SLICE_FREQ_MHZ                = 8,    // Special values
defined in enum SLPM_GT_FREQ_MHZ
        SLPM_KMD_PARAM_GLOBAL_MAX_GT_SLICE_FREQ_MHZ                = 9,    // Special values
defined in enum SLPM_GT_FREQ_MHZ
        SLPM_KMD_PARAM_GTPERF_THRESHOLD_MAX_FPS                    = 10,
        SLPM_KMD_PARAM_GLOBAL_DISABLE_GT_FREQ_MANAGEMENT           = 11,   // This does not
need force enable. Freq range will take care of it when this is FALSE
        SLPM_KMD_PARAM_GTPERF_ENABLE_FRAMERATE_STALLING            = 12,
        SLPM_KMD_PARAM_GLOBAL_DISABLE_RC6_MODE_CHANGE              = 13,
        SLPM_KMD_PARAM_GLOBAL_OC_UNSLICE_FREQ_MHZ                  = 14,
        SLPM_KMD_PARAM_GLOBAL_OC_SLICE_FREQ_MHZ                    = 15,
        SLPM_KMD_PARAM_GLOBAL_ENABLE_IA_GT_BALANCING               = 16,
        SLPM_KMD_PARAM_GLOBAL_ENABLE_ADAPTIVE_BURST_TURBO          = 17,
        SLPM_KMD_PARAM_GLOBAL_ENABLE_EVAL_MODE                     = 18,
        SLPM_KMD_PARAM_GLOBAL_ENABLE_BALANCER_IN_NON_GAMING_MODE   = 19,  // Gaming mode refers
to when display flip is exclusively controlled by game context using command streamer flips

        // Also add new ones at the end. 32 KMD parameters are reserved.
        SLPM_END_KMD_PARAMS,
        SLPM_KMD_PARAMS_MAX = 32,
    } SLPM_KMD_PARAM_ID;
    CASSERT(SLPM_END_KMD_PARAMS <= SLPM_KMD_PARAMS_MAX, UkGucKmdInterface_h);

    //-----------------------------------------------------------------
    //STRUCTURE THAT WILL BE PASSED TO SLPM CORE BOOTSTRAP. IT
    //CONTAINS THE HAL FUNCTION POINTERS THAT SLPM CORE CAN USE,
    //CONTEXT DATA THAT NEEDS TO BE PASSED TO THESE FUNCTIONS AND
    //SHARED DATA STRUCTURES.
    //-----------------------------------------------------------------

    typedef enum _SLPM_PLATFORM_SKU
    {
        SLPM_PLATFORM_SKU_UNDEFINED     = 0,
        SLPM_PLATFORM_SKU_ULX           = 1,
        SLPM_PLATFORM_SKU_ULT           = 2,
        SLPM_PLATFORM_SKU_T             = 3,
        SLPM_PLATFORM_SKU_MOBL          = 4,
        SLPM_PLATFORM_SKU_DT            = 5,

        SLPM_PLATFORM_SKU_UNKNOWN,
        SLPM_PLATFORM_SKU_ALL = 7
    } SLPM_PLATFORM_SKU;
    CASSERT(SLPM_PLATFORM_SKU_UNKNOWN <= SLPM_PLATFORM_SKU_ALL, UkGucKmdInterface_h);

    typedef enum _SLPM_HW_POWERSRC
    {
        SLPM_POWERSRC_UNDEFINED         = 0,
        SLPM_POWERSRC_AC                = 1,
        SLPM_POWERSRC_DC                = 2,

        SLPM_POWERSRC_UNKNOWN,
        SLPM_POWERSRC_ALL = 3
    } SLPM_HW_POWERSRC;
    CASSERT(SLPM_POWERSRC_UNKNOWN <= SLPM_POWERSRC_ALL, UkGucKmdInterface_h);

    typedef enum _SLPM_HW_POWERPLAN
    {
```

```
    SLPM_POWERPLAN_UNDEFINED        = 0,
    SLPM_POWERPLAN_BATTERY_SAVER    = 1,
    SLPM_POWERPLAN_BALANCED         = 2,
    SLPM_POWERPLAN_PERFORMANCE      = 3,

    SLPM_POWERPLAN_UNKNOWN,
    SLPM_POWERPLAN_ALL = 7
} SLPM_HW_POWERPLAN;
CASSERT(SLPM_POWERPLAN_UNKNOWN <= SLPM_POWERPLAN_ALL, UkGucKmdInterface_h);

//-----------------------------------------------------------------
//END OF SHARED HOST-SLPM STRUCTURES
//-----------------------------------------------------------------

//-----------------------------------------------------------------
//STRUCTURES THAT ARE SHARED BETWEEN THE HOST AND SLPM CORE.
//THESE STRUCTURES ENABLE THE HOST TO COMMUNICATE DATA
//EFFICIENTLY TO THE SLPM.
//-----------------------------------------------------------------
typedef struct _SLPM_KMD_PLATFORM_INFO
{
    union
    {
        ULONG                   Bitfield1;       // SKU info
        struct
        {
            ULONG               PlatformSku      : SLPM_BIT_RANGE(7, 0);
            ULONG               FusedSliceCnt    : SLPM_BIT_RANGE(15, 8);
            ULONG               Reserved         : SLPM_BIT_RANGE(23, 16);
            ULONG               PowerPlan        : SLPM_BIT_RANGE(29, 24);
            ULONG               PowerSrc         : SLPM_BIT_RANGE(31, 30);
        };
    };
    union
    {
        ULONG                   Bitfield2;       // IA capability info
        struct
        {
            ULONG               MaxP0FreqBins    : SLPM_BIT_RANGE(7, 0);
            ULONG               P1FreqBins       : SLPM_BIT_RANGE(15, 8);
            ULONG               PeFreqBins       : SLPM_BIT_RANGE(23, 16);
            ULONG               PnFreqBins       : SLPM_BIT_RANGE(31, 24);
        };
    };
    ULONG                       Reserved1;
    ULONG                       Reserved2;
} SLPM_KMD_PLATFORM_INFO;

typedef enum _SLPM_GLOBAL_STATE
{
    SLPM_GLOBAL_STATE_NOT_RUNNING       = 0,
    SLPM_GLOBAL_STATE_INITIALIZING      = 1,
    SLPM_GLOBAL_STATE_RESETING          = 2,
    SLPM_GLOBAL_STATE_RUNNING           = 3,
    SLPM_GLOBAL_STATE_SHUTTING_DOWN     = 4,
    SLPM_GLOBAL_STATE_ERROR             = 5
} SLPM_GLOBAL_STATE;
```

```
    typedef struct _SLPM_TASK_STATE_DATA
    {
        union
        {
            ULONG Bitfield1;
            struct
            {
                // GT Perf data
                ULONG GtPerfTaskActive              : SLPM_BIT(0);
                ULONG GtPerfIsStallPossible         : SLPM_BIT(1);
                ULONG GtPerfInGamingMode            : SLPM_BIT(2);
                ULONG GtPerfTargetFps               : SLPM_BIT_RANGE(10, 3);    //8bits to
support upto 255FPS

                // DCC data
                ULONG DccTaskActive                 : SLPM_BIT(11);
                ULONG DccInDccMode                  : SLPM_BIT(12);
                ULONG DccInDctMode                  : SLPM_BIT(13);

                // Freq Switch data
                ULONG FreqSwitchActive              : SLPM_BIT(14);

                // IBC/PG1 data
                ULONG IbcEnabled                    : SLPM_BIT(15);
                ULONG IbcActive                     : SLPM_BIT(16);
                ULONG PG1Enabled                    : SLPM_BIT(17);
                ULONG PG1Active                     : SLPM_BIT(18);
            };
        };
        union
        {
            ULONG Bitfield2;
            struct
            {
                ULONG FreqRangeUnsliceMax           : SLPM_BIT_RANGE(7, 0);
                ULONG FreqRangeUnsliceMin           : SLPM_BIT_RANGE(15, 8);
                ULONG FreqRangeSliceMax             : SLPM_BIT_RANGE(23, 16);
                ULONG FreqRangeSliceMin             : SLPM_BIT_RANGE(31, 24);
            };
        };
    }SLPM_TASK_STATE_DATA;

#define SLPM_KMD_MAX_OVERRIDE_PARAMETERS 192
#define SLPM_UNSET_PARAM_SETARRAY(setarray, id)      {setarray[(id >> 5)] &= (~(1 << (id %
32)));}
#define SLPM_SET_PARAM_SETARRAY(setarray, id)        {setarray[(id >> 5)] |= (1 << (id % 32));}
#define SLPM_UNSET_PARAM(setarray, valarray, id)     {setarray[(id >> 5)] &= (~(1 << (id %
32)));  valarray[id] = 0;}
#define SLPM_SET_PARAM(setarray, valarray, id, val)  {setarray[(id >> 5)] |= (1 << (id %
32));     valarray[id] = val;}
#define SLPM_GET_PARAM(setarray, valarray, id)       valarray[id]
#define SLPM_ISSET_PARAM(setarray, valarray, id)     (setarray[(id >> 5)] & (1 << (id % 32)))
    CASSERT((SLPM_KMD_MAX_OVERRIDE_PARAMETERS & 0x1F) == 0, UkGucKmdInterface_h);

    typedef struct _SLPM_KMD_SHARED_DATA
    {
        ULONG                               Reserved;
        ULONG                               KmdSharedDataSize;            // KMD must fill this
```

```
in with the total size in bytes of this buffer.
        SLPM_GLOBAL_STATE                   KmdGlobalState;
        SLPM_KMD_PLATFORM_INFO              KmdPlatformInfo;
        SLPM_TASK_STATE_DATA                KmdTaskStateData;

        // KMD can override some of the internal parameters that are defined above in enum
SLPM_KMD_PARAM_ID.
        // There is an array for storing the override value and a bitfield array for specifying
which
        // elements of the array have been set. Use the above macros SLPM_UNSET_PARAM and
SLPM_SET_PARAM to
        // update these arrays.
        ULONG
KmdOverrideParametersSetBits[(SLPM_KMD_MAX_OVERRIDE_PARAMETERS + 31) / 32];
        ULONG
KmdOverrideParametersValues[SLPM_KMD_MAX_OVERRIDE_PARAMETERS];
    } SLPM_KMD_SHARED_DATA;
    //---------------------------------------------------------------
    //END OF SHARED HOST-SLPM STRUCTURES
    //---------------------------------------------------------------

    //---------------------------------------------------------------
    //STRUCTURES FOR DISPLAY MODE EVENT
    //---------------------------------------------------------------
#ifndef MAX_INTEL_PIPES
#define MAX_INTEL_PIPES 3
#endif
#ifndef MAX_PLANE_PER_PIPE
#define MAX_PLANE_PER_PIPE  4
#endif
#ifndef MAX_NUM_OF_PIPE
#define MAX_NUM_OF_PIPE     (MAX_INTEL_PIPES + 1)
#endif

    typedef enum SLPM_PLANE_ENUM
    {
        SLPM_PLANE_1 = 0,
        SLPM_PLANE_2 = 1,
        SLPM_PLANE_3 = 2,
        SLPM_PLANE_4 = 3
    }SLPM_PLANE;

    // Keep this structure in sync with KM_FPS_TRACKING_FLIP_INFO.
    typedef struct SLPM_KMD_FPS_TRACKING_FLIP_INFO_REC
    {
        union
        {
            ULONG Data;
            struct
            {
                ULONG BitwiseTargetPipe     : MAX_NUM_OF_PIPE;
                ULONG PipeAPlane            : MAX_PLANE_PER_PIPE;
                ULONG PipeBPlane            : MAX_PLANE_PER_PIPE;
                ULONG PipeCPlane            : MAX_PLANE_PER_PIPE;
                ULONG PipeTPVPlane          : MAX_PLANE_PER_PIPE;    //20bits
                ULONG VbiVsyncOn            : 1;
                ULONG MpoEnabled            : 1;
                ULONG FlipType              : 2; //MMIO or DMA
```

```
                //reserved, let complier pack
            };
        };
    } SLPM_KMD_FPS_TRACKING_FLIP_INFO;

    typedef struct KMD_DISPLAY_PIPE_BASIC_INFO_REC
    {
        union
        {
            ULONG   Data;
            struct
            {
                ULONG IsWiDi                : 1;
                ULONG RefreshRate           : 7;
                ULONG VSyncFTUsec           : 24;
            };
        };
    }KMD_DISPLAY_PIPE_BASIC_INFO;

    typedef struct SLPM_KMD_DISPLAY_MODE_EVENT_PARAM_REC
    {
        struct
        {
            KMD_DISPLAY_PIPE_BASIC_INFO PerPipeInfo[MAX_NUM_OF_PIPE];

            union
            {
                ULONG GlobalData;
                struct
                {
                    ULONG BitwiseActivePipes    : MAX_NUM_OF_PIPE;  // Bitwise
                    ULONG FullscreenPipes       : MAX_NUM_OF_PIPE;  // Bitwise - Fullscreen is
defined as a pipe whose
                    // display surfaces are not owned by the OS composer.
                    ULONG VbiVsyncOnPipes       : MAX_NUM_OF_PIPE;  // Bitwise - Pipes that have
VBI enabled.
                    ULONG NumOfActivePipes      : 2;
                };
            };
        };
    } SLPM_KMD_DISPLAY_MODE_EVENT_PARAM;
    //-----------------------------------------------------------------
    //END OF STRUCTURES FOR DISPLAY MODE EVENT
    //-----------------------------------------------------------------

    //-----------------------------------------------------------------
    //RESET/SHUTDOWN EVENT FLAGS
    //-----------------------------------------------------------------
    typedef enum _SLPM_KMD_RESET_FLAGS
    {
        SLPM_KMD_RESET_FLAG_TDR_OCCURRED = (1 << 0)
    } SLPM_KMD_RESET_FLAGS;
    //-----------------------------------------------------------------
    //END OF RESET/SHUTDOWN EVENT FLAGS
    //-----------------------------------------------------------------

    //-----------------------------------------------------------------
    //STRUCTURE USED TO PASS IN EVENT INPUT DATA
```

```
    //-----------------------------------------------------------------
#define SLPM_MAX_EVENT_INPUT_ARGS  7
#define SLPM_MAX_EVENT_OUTPUT_ARGS 1


    CASSERT(SLPM_MAX_EVENT_INPUT_ARGS <= 14, UkGucKmdInterface_h); // Host2GuC gives us 15
scratch registers (the first we will use for the event ID).
    CASSERT(SLPM_MAX_EVENT_INPUT_ARGS >= (MAX_NUM_OF_PIPE + 1), UkGucKmdInterface_h); //
SLPM_KMD_EVENT_DISPLAY_MODE_CHANGE requires the most number of arguments
    CASSERT(SLPM_MAX_EVENT_OUTPUT_ARGS >= 1, UkGucKmdInterface_h);  // Host2GuC number of Output
Args should be at least 1
    CASSERT(SLPM_MAX_EVENT_OUTPUT_ARGS <= 14, UkGucKmdInterface_h); // Host2GuC gives us 15
scratch registers (the first we will use for the event ID).

    typedef union _SLPM_EVENT_INPUT_HEADER
    {
        ULONG                       HeaderData;
        struct
        {
            ULONG                   NumArgs         : SLPM_BIT_RANGE(7, 0);
            ULONG                   EventId         : SLPM_BIT_RANGE(15, 8);
        };
    } SLPM_EVENT_INPUT_HEADER;

    typedef struct _SLPM_EVENT_INPUT
    {
        ULONG                       MessageId; // This is reserved for use when sending requests
to GuC.
        SLPM_EVENT_INPUT_HEADER     Header;
        ULONG                       Args[SLPM_MAX_EVENT_INPUT_ARGS];
    } SLPM_EVENT_INPUT;

    typedef union _SLPM_EVENT_OUTPUT_HEADER
    {
        ULONG                      HeaderData;
        struct
        {
            ULONG                  NumArgs          : SLPM_BIT_RANGE(7, 0);
            ULONG                  EventId          : SLPM_BIT_RANGE(15, 8);
            ULONG                  Status           : SLPM_BIT_RANGE(31, 16);
        };
    } SLPM_EVENT_OUTPUT_HEADER;

    typedef struct _SLPM_EVENT_OUTPUT
    {
        ULONG                       Error;  // This contains the result of the ISR function.
        SLPM_EVENT_OUTPUT_HEADER    Header;
        ULONG                       Args[SLPM_MAX_EVENT_OUTPUT_ARGS];
    } SLPM_EVENT_OUTPUT;
    //-----------------------------------------------------------------
    //EVENT INPUT DATA
    //-----------------------------------------------------------------

#ifdef __cplusplus
}
#endif

#endif  //End of '#define _SLPC_KMD_INTERFACE_H_'
```

```
#if defined (_MSC_VER)
    #pragma pack(pop)
#else
    #pragma pack()
#endif

#endif /*_UK_GUCKMDINTEFACE_H_*/
```

Bugs and Debugging

This section aims to give information about bugs (overall view of bug process, how to report bugs, tips, collecting useful information), as well as some hints for debugging linked to Intel® Graphics for Linux environment.

How to Report Bugs

How to File a Bug Report

1 - Pick a Culprit Component

When you encounter a bug, it can be on Kernel or on User Space, make your best guess and follow the instructions below.

Please **be specific**:

- file **separate** bugs for **each distinct** case of failure; if you find several issues in one use case, please split them into several bugs.
- report the defect by reporting **root cause** (*fault, which can be seen in logs, for example*) rather than just the symptom (*such as a test or application is failing*).

1.1 - DRM Kernel

Follow the bug filing instructions from the project's wiki.

1.2 - 3D MESA

Mesa's issue tracker can be found on Freedesktop.org's gitlab.

Include the following details:

- A clear subject. Starting with [[Driver] [Chipset] [Subsystem/Feature]] to highlight if needed, such as [IRIS ICL TransformFeedback].
- System environment:
  - Cchipset: (such as G965)
  - System architecture: ("uname -m")
  - Mmesa/libdrm version: (If you built it yourself, provide the git commit ID or stable release version. If it's shipped with distribution, provide libdrm version with "pkg-config --modversion libdrm" and mesa version with glxinfo output.)
  - Kernel version: ("uname -r")
  - Linux distribution:
  - Machine or mobo model:
- Steps to reproduce. Probability if not 100% reproducible.
- Attachment:

- dmesg output (mandatory boot option "drm.debug=0x06")
- screenshot or photo (optional, a picture is worth a thousand words)
- for GPU hang, get the last batch buffer (see the guide).
- for 3D issues, run with LIBGL_DEBUG=verbose in the environment.

2 - Assistance to Help Debugging

- Provide gdb backtrace. See the guide for how to use gdb for an X server crash.
- For regression, try using git-bisect to locate the commit which caused the regression.
- Compare with other products or configurations, if possible.
- For VT switch problems, use intel_reg_dumper tool (see the guide) to dump the register info: 1) prior to VT switch 2) after VT switch.

3 - Tips

- Specific: Each bug report is for only one issue/problem. If you found several issues in one test, please separate them into several bugs.
- Close the bug if the originally reported issue is resolved. For new issues, open new bugs, instead of continuing the discussion in the old bug.
- Attach a log, explicitly choosing "plain text (text/plain)" content type. Don't use zipped files (except on error state for gpu hang).
- After answering developer's NEEDINFO, make sure 'NEEDINFO' is cleared in the keywords field and reassign it back to the developer owner.
- Attach the monitor directly to the machine, rather than through a KVM or other devices.
- Using warm reboot to try the new code is not completely clean for the gfx environment. We recommend doing a power cycle instead of warm reboot.

4 - Appendix: Recommended General Bug Report Template

Bug description:

System environment:

-- chipset:

-- system architecture: XX-bit

-- xf86-video-intel:

-- xserver:

-- mesa:

-- libdrm:

-- kernel:

-- Linux distribution:

-- Machine or mobo model:

-- Display connector:

Reproducing steps:

Additional info:

How to get Kernel Backtrace

Since kernel version 3.7 with the new modeset code we have a lot more self-checks in the code. If one of these checks fail it will result in a backtrace in dmesg. The good thing is that even really small issues in the code can be caught much earlier (often before it negatively affects the end user experience), the downside is that we'll hit many more call trace issues in testing. To make handling these issues more efficient below are a few BKMs. The most important step is to figure out which kind of call trace (or backtrace) it is:

1- Bug Backtrace

Those come in two forms:

- One line BUG_ON warning like "`BUG:at drivers/gpu/drm/i915/intel_display.c:7867intel_modeset_check_state+0x33c/0x5b9[i915]()`" with source file and line + the hex offset of the crashing function. When reporting such a bug please put that exact line into the subject for the initial report (developers might change it later on to something more descriptive).
- BUG that does not have a function name in the first line, like "`BUG:unable to handle kernel NULL pointer dereference atvirtualaddress00000008`". To make the bug headline more descriptive please also add the name of the crashing function, it usually looks like "`EIPisat acpi_ns_internalize_name+0xd/0x83`" or "`IP:EIPisat acpi_ns_internalize_name+0xd/0x83 `".

BUGs will stop the current thread, so usually there's not a following call trace. In any case, all subsequent call traces are most likely issues that occur right after the main bug and those traces can be safely ignored.

2- Warning Backtrace

These start with something like "`WARNING: at drivers/gpu/drm/i915/intel_display.c:7867 intel_modeset_check_state+0x33c/0x5b9 [i915]()`".

For the normal WARNING, it is best to just report the very first line with source file, line number and function. But there are a few subcases of WARNINGs which need special care.

2.A- Warning With Additional Information Line

Some WARNING backtraces also dump an additional info line like:

```
[ 34.104233] [ cut here ]
[ 34.104252] WARNING: at drivers/gpu/drm/i915/intel_display.c:7839 intel_modeset_check_state
+0x33c/0x5b9 [i915]()
[ 34.104253] Hardware name: 2012 Client Platform
[ 34.104254] encoder's hw state doesn't match sw tracking (expected 1, found 0)
[ 34.104274] Modules linked in: ipv6 dm_mod acpi_cpufreq coretemp kvm_intel kvm microcode pcspkr
i2c_i801 snd_hda_codec_realtek iTCO_wdt iTCO_vendor_support snd_hda_intel firewire_ohci lpc_ich
snd_hda_codec firewire_core mfd_core snd_hwdep crc_itu_t snd_pcm snd_page_alloc shpchp snd_timer
snd soundcore i915 video button drm_kms_helper drm mperf freq_table
```

Here is the third line (between "Hardware name:" and "Modules linked in:" lines) is the important one. So please use it when reporting such a call trace like "`WARNING: encoder's hw state doesn't match sw tracking(expected1,found0)intel_modeset_check_state+0x33c/0x5b9`". Adding the function name is useful to see the context of the warning at a quick glance. Note that some BUG backtraces also have this info line, they should be handled similarly.

2.B- Warning From The Modeset Pipe Config Checker

In kernel 3.10 we've added even more detailed modeset check code but in that code any mismatch will result in the same "`pipe state doesn't match!`" WARNING. Right before that line there should be an *ERROR* which has more details. Example:

```
[ 6.769026] [drm:intel_pipe_config_compare] *ERROR* mismatch in adjusted_mode.flags (expected 1,
found 0)
[ 6.769026] [ cut here ]
[ 6.769049] WARNING: at drivers/gpu/drm/i915/intel_display.c:8186 intel_modeset_check_state
+0x904/0x94b [i915]()
[ 6.769049] pipe state doesn't match!
[ 6.769055] Modules linked in: i915(+) drm_kms_helper drm video button dm_mirror dm_region_hash
dm_log dm_mod ipv6
[ 6.769058] CPU: 0 PID: 1427 Comm: systemdudevd Not tainted 3.10.0rc2_drmintelnext▶
queued_92d446_20130603_+ #3325
[ 6.769059] Hardware name: Apple Inc. MacBookPro10,2/MacAFD8A9D944EA4843, BIOS
MBP102.88Z.0106.B03.1211161133 11/16/2012
[ 6.769062] ffffffff816de142 0000000000000000 ffffffff8102c67d ffff880262bb93a0
[ 6.769064] ffff880262bb93a8 0000000000000001 ffff880264797000 ffff880262af0800
[ 6.769065] ffff8802647976c8 ffff880262670000 ffffffff8102c72d ffffffffa0134caf
[ 6.769066] Call Trace:
[ 6.769072] [<ffffffff816de142>] ? dump_stack+0xd/0x17
[ 6.769075] [<ffffffff8102c67d>] ? warn_slowpath_common+0x5f/0x77
[ 6.769077] [<ffffffff8102c72d>] ? warn_slowpath_fmt+0x45/0x4a
[ 6.769098] [<ffffffffa00fbd74>] ? intel_modeset_check_state+0x904/0x94b [i915]
[ 6.769116] [<ffffffffa00fbe2b>] ? intel_set_mode+0x1d/0x27 [i915]
...
```

When filing such a backtrace please use the following template for the headline

"`WARNING: pipe state mismatch:<reason>fromthe*ERROR*line`"

So for the example

"`WARNING: pipe state mismatch:mismatchinadjusted_mode.flags(expected1,found0)`"

The above calltrace introduction is only for when the calltrace is the main issue indicator, if something else goes wrong (machine hardhangs or an automated test fails) that's more important and should be the headline (with a "with calltrace" note added). When verifying and there are still calltraces in dmesg it's important to verify whether it's still the same one. If the info line (or the preceding *ERROR*) have changed, it's a different one (so should be filed as a new bug). However, if the info line/*ERROR* (if available) and the function name are the same and only the line changed, it means that it is the same calltrace issue.

Developers will make sure that for calltraces without further information (i.e. no debug info line) the function name uniquely identifies the backtrace for a WARNING (BUGs might be different) by placing at most one WARNING without debug info per function.

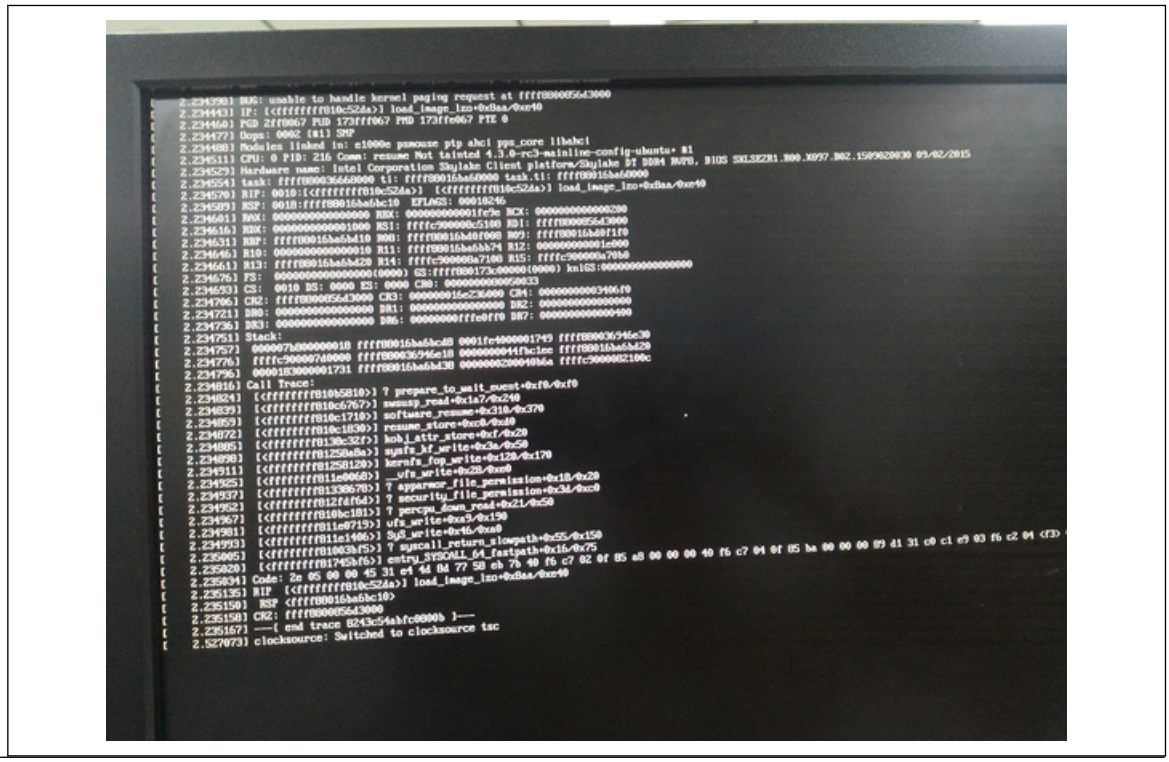
> **Important** When pasting a call trace into a bug report always include the important information right before the calltrace, too. There's a special `[cut here]` for that purpose.

3- Pasting Kernel Backtrace

For kernel bugs with a backtrace in dmesg it's often useful to paste the backtrace into a comment in bugzilla for a quick overview. There's a lot of other important information before the section starting with "Calltrace" though (in most cases this stuff is more important than the call trace itself). Unfortunately there's often no clear start marker for the interesting stuff, so I think as a rule of thumb everything 12 seconds before the calltrace should be included. Also, as a general rule only the very first backtrace is important, all the later ones are usually just follow-up issues.

If the machine hangs and you can't get at the backtrace with logs or netconsole it will (hopefully) get printed onto the screen. Please take a picture with a camera. To make reading such an oops or ***panic*** easy please make sure the picture is sharp all across the screen and that the picture is taken parallel to the screen.

Display picture showing trace following a kernel panic



Unfortunately it often scrolls off the screen, especially if multiple backtraces show up. If this is the case please boot with `pause_on_oops=60` so that the kernel waits 1 minute after the first oops before printing the next one (or generally continuing). Sometimes even just one backtrace is too large, setting `boot_delay=100` might help in this case, it should slow down the output (so that you can take multiple pictures and stitch it together).

Learn More About GFX Bug Handling

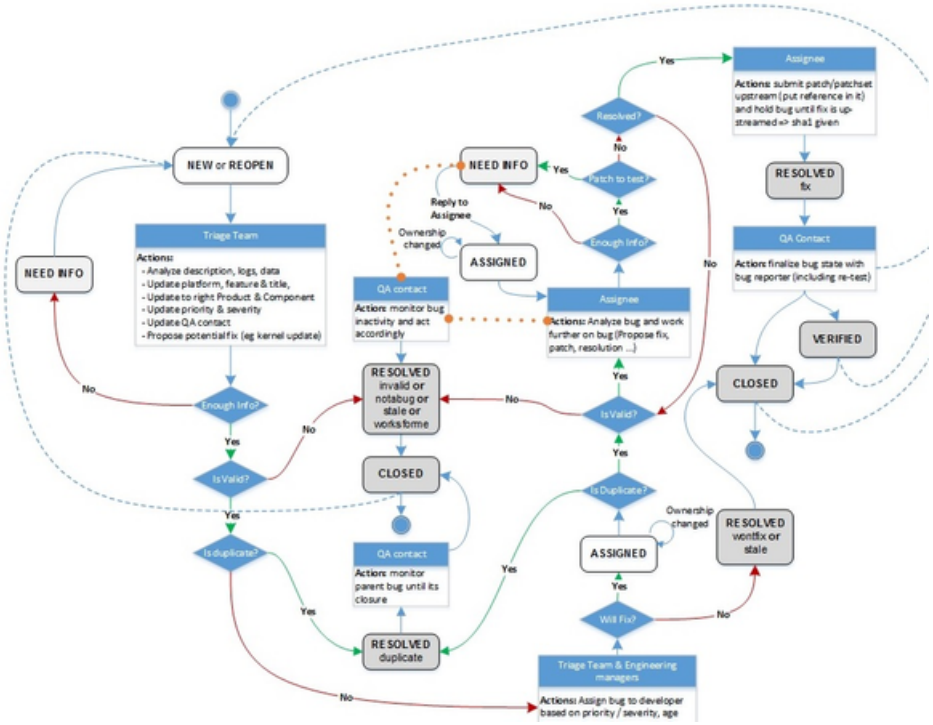
How Bugs are Handled

Our main interface to deal with defects in the kernel graphics driver is Bugzilla at Freedesktop.org. We cover multiple products and components depending on where the issue is located. Our development and QA teams try to respond and handle reported issues as quickly as possible. We constantly assess if we have enough information to proceed on analysis and investigation, and might contact the bug reporter to get further details or qualify a fix if we don't have the right equipment or configuration linked to the issue.

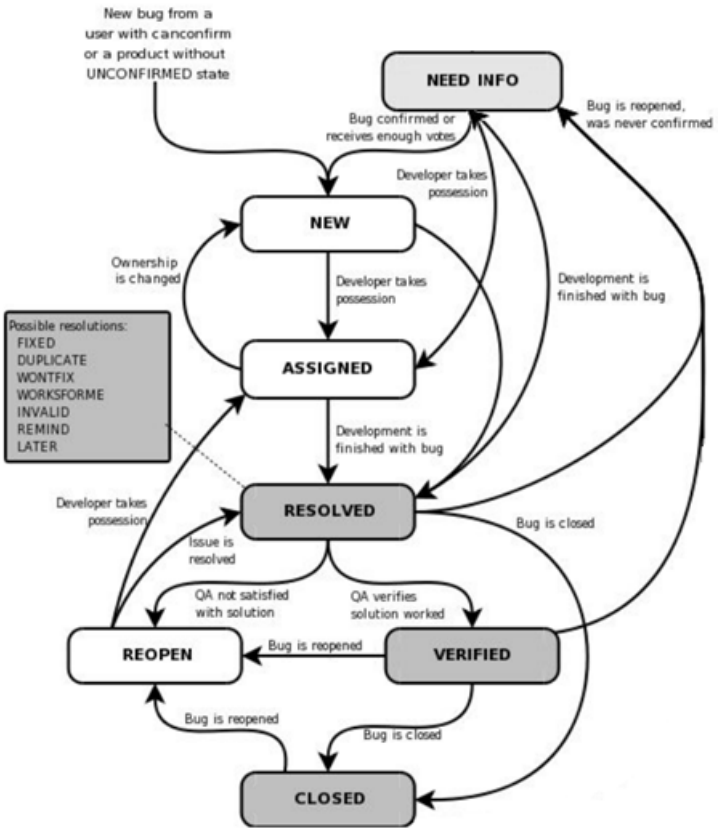
You might notice that our stacks are evolving every day to add new features, new platform support, new fixes, and new performance improvements. Therefore, if we negligibly forget a bug, or a bug has no activity for several months, we will ping the bug reporter to either confirm that the issue is fixed with latest stack, or to ask for more information. In cases where we don't get any response from the reporter for one week, we might assume that the bug is not reproducible and close the bug. The bug reporter can reopen the bug if they still see it in the new stack and can provide some fresh data that can help us to solve the issue.

You can find more details about the current bug life cycle, freedesktop.org (FDO), the Bugzilla bug state workflow, and what different bug priority levels mean below.

Bug Life Cycle



FDO Bugzilla Bug State Workflow



When a bug is reported, it comes with some default values, and its status is NEW. Note that successful bisecting should be done before setting Status = ASSIGNED and Assignee = as it is assumed that the revert or fix will be done by this person with existing knowledge. When a bug's status is moved to ASSIGNED, that means that a developer is assigned to that bug. As soon as development is done on the bug (including any upstream fixes), the status moves to RESOLVED. Then QA and / or the bug reporter can confirm that the solution resolves the issue by appropriately updating the bug status.

Developers or QA might ask for additional information. In that case, we change the bug status to NEEDINFO. Once you provide the requested information, you should change the status of the bug to its previous value.

Bug Priority

Usually, priority is changed and updated by QA, but severity can be modified by anybody. The high-level descriptions below provide some information on what we mean and how we address issues based on priority. This information is intended to be an overall indication.

P1≈ Highest or (High + blocker): Resolution of the defect takes precedence over other defects and most other development activities. It is applied "most of the time" to Critical severity defects blocking development, internal validation, or external customer validation.

For example: System hang, crash, general protection fault, severe security vulnerability, loss or data corruption, certification failure, factory lines down -> app failure impacting overall system behavior, no workaround currently available.

P2≈ High or (Medium + major + frequency > 50%): defects are intended to be resolved within one of the next two external releases. P2 is applied "most of the time" to High severity defects applying to functionality, features, and use cases currently being integrated and targeted for the next milestone release. Should not typically be shipped to customers, but can be documented if it is.

For example: Defect causes severe degradation to an intended feature, functionality, or use case. Recovery may require a soft reboot. The issue can be documented, a workaround might be possible without significantly impacting users, development or testing can be continued.

P3≈ Medium: defects must have a planned timeframe for resolution. P3 is applied "most of the time" to Medium severity defects applying to functionality, features, and use cases currently being integrated and targeted for the next milestone release.

For example: Product fails to meet its specifications or requirements for non-key features. Workaround exists and is easily implemented. Development and testing can be continued.

P4≈ Low or Lowest: Defect here may or may not have plans to be resolved.

For example: Cosmetics defects. Can be shipped to customers without resolution

Priority: Default is P3 or medium. Regression bug is a P2 or high, and a blocker bug becomes a P1 or highest priority.

Severity: Default is normal. If bug can impact many platforms, it can be up to major. Hang bugs are up to critical, and blocker bug can up to blocker.

Project: Intel® Graphics for Linux*

Tips that may Help to Solve your Issue in Less Timed

Here are some tips that certainly can foster issue resolution:

- Check if the issue you are facing is already reported thanks to the search function of Bugzilla. If so, please report your issue there with details, otherwise create a new issue,
- Verify that your issue exists with the latest Intel® Graphics Stack Recipe (ie DRM kernel, libdrm, xf86-video-intel, mesa, libva vaapi - i915 driver, firmware) since it may be already fixed,
- Verify on http://ark.intel.com/ your CPU/GPU specification in order to confirm that the issue you are facing is in the boundary of these specifications.

- Try to reproduce issue using default options (no specific i915 parameter in command line),
- Be specific: Each bug report is for only one issue/problem. If you found several issues in one test, please split them into several bugs,
- On a case of IGT test failing, note that test tries to tell you why something is failing and therefore, file separate bugs for each distinct case of failure,
- Add as much accurate details as possible, including hardware and software configuration, logs, gdb trace, … and clear steps to allow us to easily reproduce the issue,
- Always add drm.debug=0xe to the kernel command line to get details in kernel log,
- When attaching the dmesg also make sure that it is complete and contains everything from the first boot messages. If there's too much in dmesg so that the kernel dmesg buffer overflows and overwrites early boot messages, you can extend the dmesg buffer by adding log_buf_len=4M on the kernel cmdline. Increase the size even more if that's not enough. For kernel bugs always attach dmesg,
- Mark old attachments as obsolete (this is only required if there are already quite a few attachments on the bug) to be able to keep a quick overview,
- In a case of a regression, **always** add **Good commit id** (where regression was not occurring) and **Bad commit id** (where regression is occurring ; please note that Bad commit id must be more recent than Good commit id), and if you can, try to use git-bisect and provide bisect information such as commit that causes regression, Note that successful bisecting should be done before setting Status = ASSIGNED and Assignee = <Submitter/Author of Offending Commit> as it is assumed that reverting or fixing will be done by this person with existing knowledge.
- If you can quickly fix the bug, just do it and submit a patch :^)

How to Get the GPU Error State

For all GPU-hang bugs, the most useful information is on error state.

After a hang is detected all related and helpful information gets recorded in a error state that can be grabbed as instructed below.

It must be done after the hang, but before rebooting the machine.

Recent Kernel Version

On recent kernel you can grab this information from your sysfs:

```
$ cat /sys/class/drm/card0/error | gzip > error.gz
```

Please note, that depending on your environment, it can be a different card number. But, it is probably 0.

Old Kernel

On old kernel but not older than 2.6.34 you need to get it from debugfs:

`/sys/kernel/debug/dri/0/i915_error_state [2]`

Anything from that file, other than "no error state collected", is very interesting and very helpful for bug reports.

[1] If you don't have a /sys/kernel/debug/dri directory, then make sure that debugfs is mounted like this:

```
sudo mount -t debugfs debugfs /sys/kernel/debug
```

[2] If you have n gpus, the directory could be also be:

`/sys/kernel/debug/dri/<n>`

Compress Instead of Crop

All information on this file is useful, so please never crop the file when attaching it to a bug report, Compress it instead.

How to Dump Video BIOS (VBIOS)

These commands can be used to dump VBIOS into vbios.dump, so you can provide this info to help debugging.

- `echo 1 > /sys/devices/pci0000:00/0000:00:02.0/rom`
- `cat /sys/devices/pci0000:00/0000:00:02.0/rom > vbios.dump`
- `echo 0 > /sys/devices/pci0000:00/0000:00:02.0/rom`

How to Debug Suspend-Resume Issues

The purpose of going through the steps below is to help narrow down the root cause (BIOS? Kernel? Graphics driver?) of the suspend/resume failure and provide more info for debugging.

Prerequisite: No processes using /proc/acpi/event (which might be used by, for example, gnome-power-manager, to interfere with some ACPI related actions, like pressing the power button).

1. Get the process id by "lsof /proc/acpi/event".
2. Kill these processes.

NOTE

1. Perform suspend/resume test under both text console mode (init level 3) and X mode (init level 5) to see if it's X specific. If they are with the same failure, let's focus on text console mode and just provide logs under that condition.
2. Provide the following lofgs before and after suspend/resume:
   - dmesg output
   - intel_reg_dumper output (using reg_dump tool in xf86-video-intel source package)
   - intel_gpu_dump output (see this guide)
3. To narrow down the issue, try removing modules before you suspend. This can help determine which module is responsible for a suspend or resume failure. For example, if you remove the b43 module and then suspend/resume works, there's likely a problem with either b43 or the wifi stack.
4. If you remove i915 module, and suspend/resume still fails, it's likely not our drm graphics driver bug. Please file the bug at bugzilla.kernel.org,

   select the component "Power Management - Hibernation/Suspend", and CC rafael.j.wysocki@intel.com.

S3 Suspend (Suspend to RAM):

A. Check if the Resume Hang is Caused by a Graphics Driver:

1. Run: dmesg > dmesg_before; echo mem > /sys/power/state; dmesg > dmesg_after
2. Press the power button to see if the machine can be resumed.
3. If it can't be resumed, reboot the system then check if the file "dmesg_after" generated.
4. If dmesg_after exists, it means that the system can be resumed from BIOS, so it's likely to be a graphics issue. Otherwise, it's not related to graphics and don't report it to freedesktop.org bugzilla.

B. Suspend/Resume by Skipping BIOS:

1. Enable `CONFIG_PM_DEBUG` in kernel configuration, so that we can use /sys/power/pm_test.
2. Run: echo core/processors/devices > /sys/power/pm_test
3. Run: dmesg > dmesg_before; echo mem > /sys/power/state; dmesg > dmesg_after
4. Wait for about five seconds and see whether it can be resumed correctly.

5. If it can't be resumed correctly, it may be graphics driver bug. This information is helpful to narrow down the issue.

S4 Hibernation (Suspend to Disk):

If resume fails at kernel booting stage, try adding boot option "resume=/dev/XXXX" (XXXX means the swap partition).

Check if hibernation under console mode has the same failure as in X mode.

1. Boot the system into init level 3 (text mode).
2. Run: echo disk > /sys/power/state
3. Press the power button to see if the system can be resumed correctly.
4. If it can be resumed normally, the issue may be related to the graphics driver.
5. If it can't be resumed normally, the issue may be related to the Linux kernel.

Using Intel® Reg-Dumper

Intel® Register Dump Tool Guide

The intel_reg tool is useful to get the values of Intel registers. The dump option can get a detailed and parseable format, which is very helpful to the developers when investigating issues. To use it, follow these steps:

- Download the intel-gpu-tools package from the git repository at https://cgit.freedesktop.org/xorg/app/intel-gpu-tools/, or install the intel-gpu-toolspackage from your distribution package management application.
- If you are building the tool from the source, install it by following the standard procedure:

```
# ./autogen.sh
# configure
# make
# make install
```

- Run the tool to get the output:

```
$ tools/intel_reg dump --all > intel_reg_dump.txt
```

- Attach the generated `intel_reg_dump.txtfile` to the bug report.

Intel® G45 Express Chipset

Intel® G45 Express Chipset [Graphics and Memory Controller Hub-Gmch] Programmer's Reference Manual

Attachment	Size
G45 - Volume 1: Graphics Core	1.96 MB
G45 - Volume 2: 3D/Media	2.63 MB
G45 - Volume 3 Register_0_0	1.96 MB
G45 - Volume 4: Subsystem and Cores	2.55 MB

Intel® 965 Express Chipset Family and Intel® G35 Express Chipset Graphics Controller

Intel® 965 Express Chipset Family and Intel® G35 Express Chipset Graphics Controller PRM

Attachment	Size

965/G35 - Volume One: Graphics Core	5.47 MB
965/G35 - Volume Two: 3D/Media	2.24 MB
965/G35 - Volume Three: Display Registers	3.2 MB
965/G35 - Volume Four: Subsystem and Cores	5.95 MB

Intel® Integrated Graphics Device – OpRegion Specification

Attachment	Size
Opregion Specification	1.37 MB

Archived Documentation

3 - Pipes

3-pipes is a feature that allows users to have three Monitors plugged in. It is present at 3rd Generation Intel® Core™ processors with Intel® HD Graphics (code named IvyBridge) and 4th Generation Intel® Core™ processors with Intel® HD Graphics (code named Haswell).

For other platforms only Dual outputs are supported.

Ivybridge Limitations

In order to get three screen outputs at Ivy Bridge you should use two display ports plus any display with some limitations on modes supported.

An easy way to verify what modes are supported is using this script.

Haswell Limitations

Haswell 3-pipes is less restrictive than Ivy Bridge. You can have three screens with

* 2 Display Ports + any display

* 1 Display Port and 2 HDMI or DVI

* 1 VGA and 2 HDMI or DVI

and no restrictions on mode combination.

How to Set Up Dual Head for Intel® Graphics with RandR 1.2

1. Introduction

This guide is targeted toward people who want to use extended desktop mode on two outputs. Clone mode should work out-of-the-box with a normal configuration.

With RandR 1.2, you can set up dual head and add/remove the monitor dynamically (that is, on-the-fly, without restarting X).

To use RandR 1.2, you might need to know three key concepts: (Virtual) Screen, Crtc, and Output. A display is in one (Virtual) screen, which may consist of more than one crtc. Every crtc will occupy one rectangular region in the (Virtual) screen and may have different modes. A crtc can connect to more than one output and determine the output's property. Output is a port to monitor.

Intel® GFX device supports two pipes, which each correspond to a crtc. So there can be two crtcs per Intel® GFX device.

Prerequisite: To use randr1.2, you must use xserver 1.3 or above. And for Intel® gfx driver, you must use xf86-video-intel 2.0.0 or above.

> **NOTE** After the driver was moved to Kernel Mode Setting, the name of the output changes (such as from LVDS to LVDS1).

2. Two Methods of Setting

We can set up dual head: -- dynamically by using xran

How to Set Up Dual Head for Intel® Graphics with RandR 1.2

dr tool or -- statically by setting in xorg.conf.

xrandr tool (an app component in Xorg) is a command line interface to RandR extension, and can be used to set outputs for a screen dynamically, without any specific setting in xorg.conf. Refer to the xrandr manual for details.

Also, you can set up dual head statically in xorg.conf. With RandR1.2 enabled drivers, monitors may be tied to specific outputs of the video card. Refer to the xorg.conf manual for details.

Below, we use examples to show how to set up dual head both ways.

2.1. Dynamically Set Up with Xrandr

Xrandr can be used to change outputs' mode, rotation direction, position, etc. In this guide, we introduce only options related to the dual head setting.

You can see the outputs' status with option '-q'. Here's an example:

xrandr -q

Screen 0: minimum 320 x 200, current 2048 x 768, maximum 2048 x 2048

VGA connected 1024x768+0+0 (normal left inverted right x axis y axis) 304mm x 228mm

1024x768 60.0*+ 75.1 60.0*

800x600 75.0 60.3

640x480 75.0 60.0

720x400 70.1

LVDS connected 1024x768+1024+0 (normal left inverted right x axis y axis) 304mm x 228mm

1024x768 60.0*+ 50.0

800x600 60.3

640x480 60.0 59.9

In the example above, two outputs (VGA and LVDS) are connected. Both have a resolution of 1024x768@60. Output VGA's viewport is at (0, 0) and output LVDS's viewport is at (1024, 0). That is, the LVDS's viewport is located to the right of VGA's viewport. The size of the area occupied by both outputs is 2048x768. The virtual screen size (maximum framebuffer) is 2048x2048, which is statically set in xorg.conf (see the next section). You can decrease the maximum with the '--fb' option, but you cannot increase it. You need make sure your screen size falls into the maximum framebuffer, otherwise you should increase the value in xorg.conf.

There are five xrandr options that can be used to set the dual head:

```
--pos <x>x<y>

     --left-of <output>

     --right-of <output>

     --above <output>

     --below <output>
```

The last four will set the output's relative position to another output, for example:

xrandr --output VGA --left-of LVDS

Option '-pos' is more flexible which can place output to anywhere, for example:

xrandr --output VGA --pos 200x200

xrandr --output LVDS --pos 400x500

2.2. Statically Set Up in xorg.comf

RandR1.2 configuration in xorg.conf is set by monitor. So you need write a 'Monitor' section for each output and specify these monitors in the 'Device' section.

Below is a example snippet in xorg.conf.

```
Section "Device"

        Identifier      "Intel 945G "

        Driver         "intel"



        # Using the name of the output defined by the video driver plus the identifier of a

        #     monitor section, one associates a monitor section with an output by adding an

        #     option to the Device section in the following format:

        #     Option "Monitor-outputname" "monitor ID"

        Option          "monitor-VGA" "foo"

        Option          "monitor-LVDS" "bar"

        #Option         "monitor-TMDS-1" "dvi"

EndSection
```

```
Section "Monitor"

        Identifier      "foo"



        # specifies a mode to be marked as the preferred initial mode of the monitor

        # Option "PreferredMode"  "800x600"
```

```
        # This optional entry specifies the position of the monitor within the X screen.
        #Option        "Position" "1024 0"
        #This optional entry specifies that the monitor should be ignored
        #     entirely, and not reported through RandR.  This is useful if the
        #     hardware reports  the  presence  of  outputs  that do not exist.
        #Option "Ignore"  "true"
EndSection
Section "Monitor"
        Identifier      "bar"

        #Options LeftOf, RightOf, Above, Below specify monitors' relative position
        Option "LeftOf"  "foo"
        # This optional entry specifies  whether  the  monitor  should  be
        #     turned  on  at  startup.  By default, the server will attempt to
        #     enable all connected monitors.
        #Option "Enable"  "true"
        #This optional entry specifies the initial rotation of the given monitor.
        #     Valid values for rotation are "normal", "left", "right", and "inverted".
        # Option "Rotate"  "left"
EndSection
Section "Screen"
        Identifier      "Default Screen"
        Device        "Intel Corporation 945G Integrated Graphics Controller"
        Monitor       "foo"
        DefaultDepth  24
        SubSection "Display"
                Depth          24
                Modes         "1280x1024"  "1024x768"   "640x480"
```

```
        EndSubSection

EndSection
```

3. Reference

xorg.conf manual

xrandr manual