3D Graphics Accelerator

This chapter is a basic overview and describes the integration of the 3D graphics processing unit (GPU) subsystem in the device.

NOTE: The GPU subsystem is an instantiation by Texas Instruments of the POWERVR™ SGX544 core from Imagination Technologies Ltd.

NOTE: This chapter describes a module (subsystem) in the superset device. The availability is device part number dependent. Refer to device Data Manual, for more information.

Topic **Page**

12.1 GPU Overview

The 3D graphics processing unit (GPU) accelerates 2-dimensional (2D) and 3-dimensional (3D) graphics and compute applications. It is based on the POWERVR® SGX544-MP2 core from Imagination Technologies. The SGX544-MP2 core is a multicore (dual-core) evolution of the POWERVR SGX544 GPU.

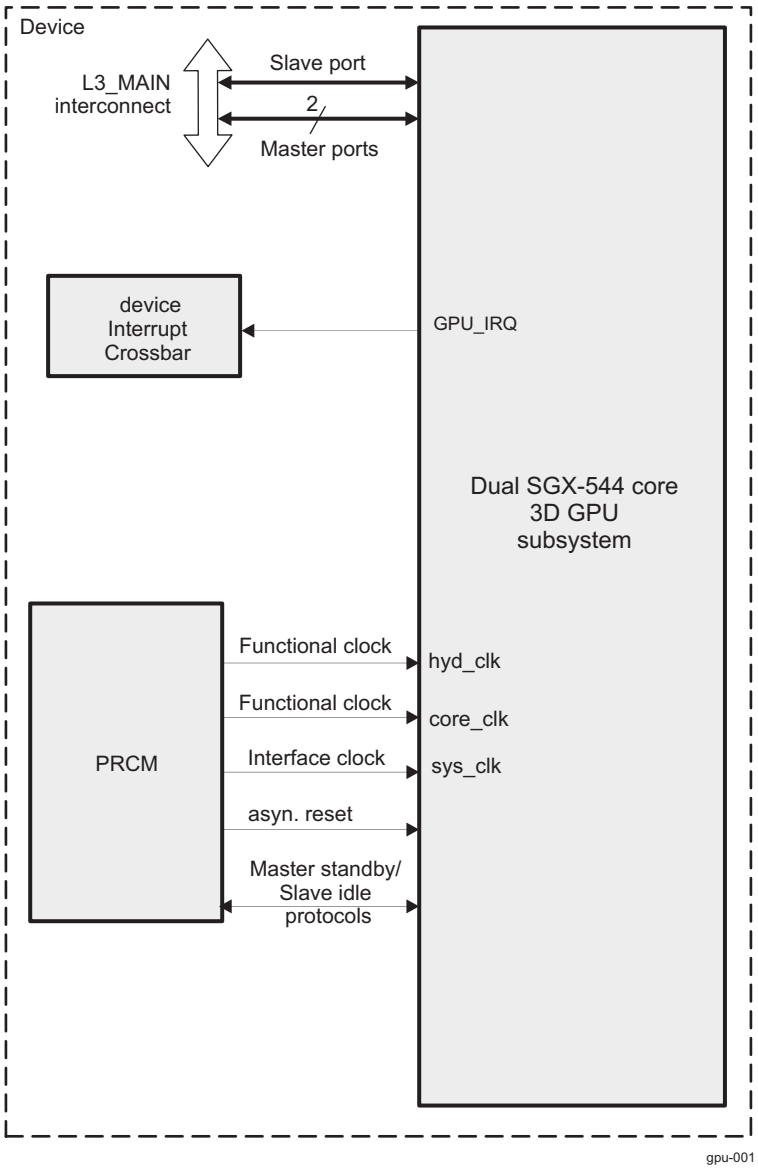
SGX is a new generation of programmable POWERVR graphics and video IP cores. The POWERVR SGX is a scalable architecture which efficiently processes a number of differing multimedia data types concurrently :

- Pixel Data
- Vertex Data
- General Purpose Processing

The dual core GPU splits geometry and pixel rendering among the cores to improve performance proportional to the number of cores.

Figure 12-1 shows the device GPU subsystem.

Figure 12-1. GPU Overview



12.1.1 GPU Features Overview

- API support for industry standards:
  - OpenGL® - ES 1.1 and 2.0
- Multicore GPU architecture:
  - 2 × SGX544 cores
  - Shared system level cache of 128 KiB (64 KiB per SGX-544 core)
- Tile-based deferred rendering architecture:
  - Reduces external bandwidth to SDRAM
- Universal Scalable Shader Engine (USSE™):
  - Multithreaded engine incorporating vertex and pixel shader functionality
  - Automatic load balancing of vertex and pixel processing tasks
- Present and texture load accelerator (PTLA):
  - Enables to move, rotate, twiddle, and scale texture surfaces
  - Supports RGB, ARGB, YUV4:2:2, and YUV4:2:0 surface formats
  - Supports bilinear upscale
  - Supports source color key
- Fully virtualized memory addressing for operating system (OS) in a unified memory architecture:
  - Memory management unit (MMU)
  - Up to 4-GiB virtual address space

The 3D-GPU subsystem generates a single (aggregate) interrupt connected to the device Interrupt Crossbar. This allows for this interrupt to be programmatically mapped to multiple device host interrupt controllers (see Section 12.2).

12.1.2 Graphics Feature Overview

- Texture support:
  - Cube map
  - Projected textures
  - Non-square textures
- Texture formats:
  - RGBA 8888, 565, 1555, and 1565
  - Monochromatic 8, 16, 16f, 32f, and 32int
  - Dual channel, 8:8, 16:16, and 16f:16f
  - Compressed textures:
    - PVRTC-i 2 bpp
    - PVRTC-i 4 bpp
    - PVRTC-ii 2 bpp
    - PVRTC-ii 4 bpp
    - ETC1
    - DXT 1-5 and BC 4-5
  - Programmable support for YUV formats:
    - Programmable matrix in hardware, coefficients on 12 bits
    - YUV4:2:2, YUV4:2:0, two planes (NV12 or NV21); YUV4:2:0, three planes
- Resolution support:
  - Frame buffer maximum = 4096 × 4096
  - Texture maximum size = 4096 × 4096
- Texture filtering:

  - Bilinear, trilinear
  - Independent minimum and mag control
- Anti-aliasing:
  - 4× multisampling
  - Programmable sample positions

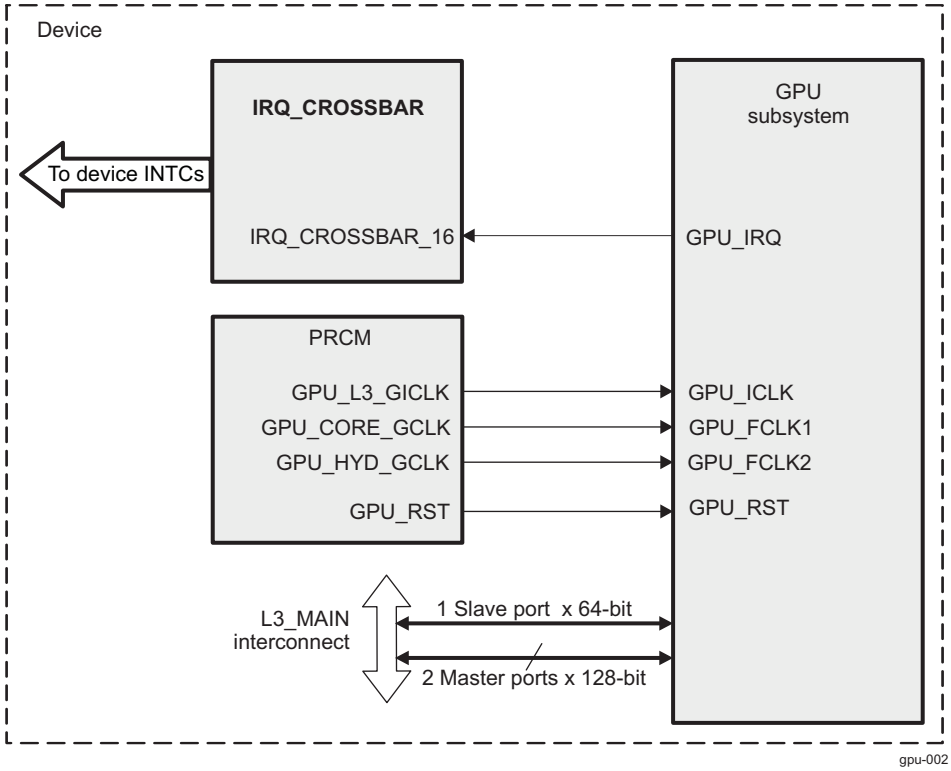
NOTE: TI provides the DXT1-5 and BC4-5 texture compression technology for use only with a Microsoft Windows operating system. A separate license is required for the use of this technology (also referred to as S3 texture compression technology) with any other operating system.

12.2 GPU Integration

This section describes the integration of the module in the device, including information about clocks, resets, and hardware requests.

Figure 12-2 shows the GPU integration.

Figure 12-2. GPU Integration



The GPU subsystem is connected to the level 3 (L3_MAIN) interconnect by two 128-bit master and a 64-bit slave interfaces.

Table 12-1 through Table 12-3 summarize the integration of the module in the device.

Table 12-1. GPU Integration Attributes

Module Instance	Attributes	
	Power Domain	Interconnect
GPU	PD_GPU	L3_MAIN

Table 12-2. GPU Clocks and Resets

Clocks				
Module Instance	**Destination Signal Name**	**Source Signal Name**	**Source**	**Description**
GPU	GPU_ICLK	GPU_L3_GICLK	PRCM	GPU interface clock
GPU	GPU_FCLK1	GPU_CORE_GCLK	PRCM	GPU functional clock of the internal graphic cores
GPU	GPU_FCLK2	GPU_HYD_GCLK	PRCM	GPU functional clock of the internal L2-cache controller and memories
Resets				
Module Instance	**Destination Signal Name**	**Source Signal Name**	**Source**	**Description**
GPU	GPU_RST	GPU_RST	PRCM	GPU non-retention reset signal

Table 12-3. GPU Hardware Requests

Interrupt Requests				
Module Instance	**Source Signal Name**	**IRQ_CROSSBAR Input**	**Default Mapping**	**Description**
GPU	GPU_IRQ	IRQ_CROSSBAR_16	MPU_IRQ_21 DSP1_IRQ_47 DSP2_IRQ_47 PRUSS1_IRQ_47 PRUSS2_IRQ_47	GPU interrupt request mapped to the device Interrupt Crossbar

NOTE: The "**Default Mapping**" column in Table 12-3, *GPU Hardware Requests* shows the default mapping of module IRQ source signals. These IRQ source signals can also be mapped to other lines of each device Interrupt controller through the IRQ_CROSSBAR module.
For more information about the IRQ_CROSSBAR module, see Section 18.4.6.4, *IRQ_CROSSBAR Module Functional Description*, in Chapter 18, *Control Module*.
For more information about the device interrupt controllers, see Chapter 17, *Interrupt Controllers*.

NOTE: No DMA and no wake-up requests are generated by the GPU subsystem.

12.3 GPU Functional Description

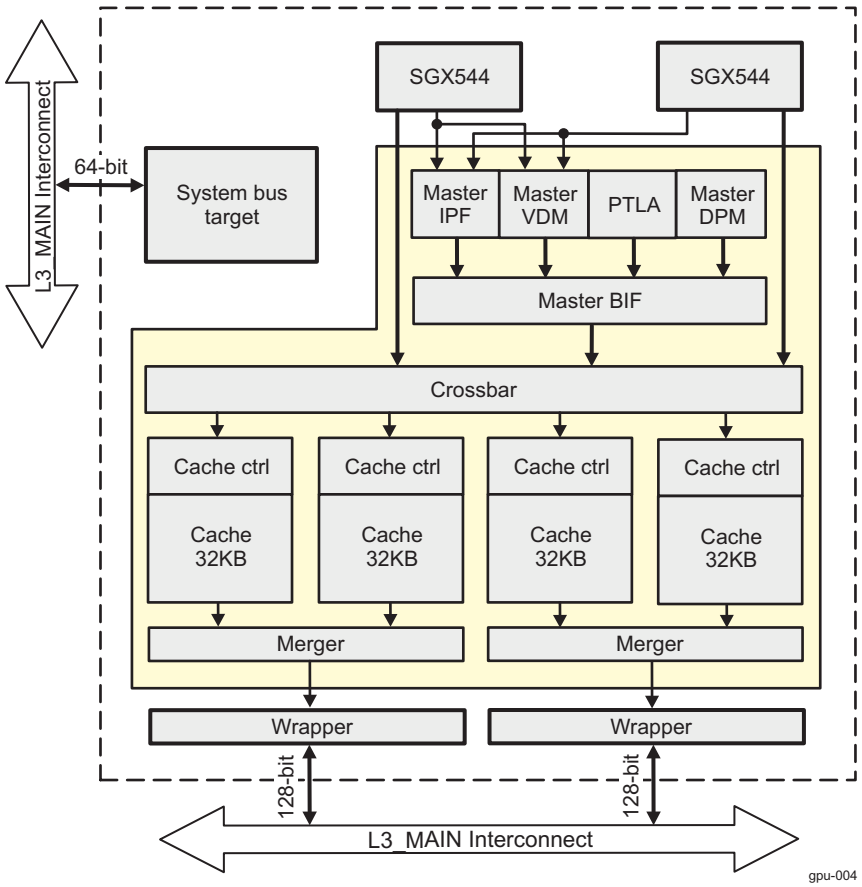
12.3.1 GPU Block Diagram

The GPU subsystem is based on the POWERVR SGX544-MP2 core. Multicore GPU can split geometry and pixel rendering among the cores to improve performance proportional to the number of cores. The graphics software engineer programming the device GPU does not have to deal with the multiple cores, this is done by the graphics drivers. The multiple cores are completely hidden behind the standard high-level graphics APIs that are supported by the graphics core. The GPU architecture comprises the following elements:

- SGX544 cores
- PTLA
- Cross bar
- System-level cache (SLC)

Figure 12-3 shows the GPU top-level block diagram.

Figure 12-3. GPU Block Diagram



The SGX544-MP2 has 2 × SGX544 cores. Graphics rendering is automatically load-balanced between the 2 × SGX544 cores. It is possible to disable one or two of the cores if required. The glue logic is used to enable the multicore architecture. The crossbar enables any SGX544 core to access the SLC. The SLC is a 128-KiB unified multibanked cache with four banks of 32 KiB. The cache line size is 64 bytes. The SGX544 core accesses are interleaved in the different banks.

12.3.2 GPU Clock Configuration

The GPU subsystem operates from three clocks: an interface clock (GPU_ICLK) and two functional clocks (GPU_FCLK1 and GPU_FCLK2). The power, reset, and clock management (PRCM) module generates and distributes the clocks inside the device.

- The GPU_ICLK manages the data transfer on the L3_MAIN master and slave ports.

  The GPU_ICLK frequency is selected based on the L3_MAIN interconnect clock frequency of the whole device. For more information about the interface clock, see Section 3.9.12, *CD_GPU Clock Domain*, in the Chapter 3, *Power, Reset, and Clock Management*.

  When no longer required by the GPU subsystem, GPU_ICLK can be disabled by software at the PRCM level. For more information, see Section 3.10.10, *PD_GPU Description*, in the Chapter 3, *Power, Reset, and Clock Management*.

  NOTE: GPU_ICLK is cut only if the GPU is ready to go into IDLE state.

- GPU_FCLK1 and GPU_FCLK2 are the functional clocks and are used inside the GPU subsystem to generate clock signals to multiple GPU clock domains. The GPU_FCLK1 input supplies clock to the internal graphics cores and the GPU_FCLK2 input supplies clock to the shared-cache memories and controllers.

  Using the clock source selection and the digital phase-locked loop (DPLL) settings, GPU_FCLK1 and GPU_FCLK2 frequencies can be adjusted.

  The GPU_FCLK1 and GPU_FCLK2 clocks are provided by the peripheral DPLL and the core DPLL, as described in Section 3.9.12, *CD_GPU Clock Domain* in the Chapter 3,*Power, Reset, and Clock Management*. Selection is made at the PRCM level.

  When no longer needed by the GPU subsystem, GPU_FCLK1 and GPU_FCLK2 can be cut by software at the PRCM level if the module is ready to enter IDLE state. For more information, see Section 3.10.10, *PD_GPU Description* in the Chapter 3, *Power, Reset, and Clock Management*.

12.3.3 GPU Software Reset

The GPU subsystem has its own reset domain. Global reset of the GPU is performed by activating the GPU_RST signal in the GPU_RST domain.

NOTE: The APIs delivered with the GPU provide a software reset functionally equivalent to a hardware reset.

12.3.4 GPU Power Management

The GPU subsystem has its own power domain (GPU power domain - PD_GPU).

The GPU handles automatic clock gating performed on the multiple internal module clock domains.

In addition, software-controlled clock gating is managed inside the GPU and handled by the related API delivered with the module.

For additional information about the GPU power domain, see Section 3.10.10, *PD_GPU Description* in the Chapter 3, *Power, Reset, and Clock Management*.

12.3.5 GPU Thermal Management

There is a dedicated thermal sensor to monitor operating temperature of the GPU subsystem in the chip. The GPU temperature processing logic is located within the device CTRL_MODULE_CORE and is capable to generate a thermal alert interrupt which can be mapped to all (host) interrupt controllers within the device, via the device IRQ_CROSSBAR. The thermal logic can also generate a thermal shut-down warm reset event to the device PRCM. For further details on the GPU thermal management operation features and register settings, refer to the Section 18.4.6.2, *Thermal Management Related Registers* of the Chapter 18, *Control Module*.

12.3.6 GPU Interrupt Requests

The GPU subsystem can generate one interrupt signal - GPU_IRQ mapped to the IRQ_CROSSBAR_16 input of the device interrupt crossbar.

For more details on programmable configuration of the GPU_IRQ mapping to the different device host interrupt controllers, refer to the Chapter 18, *Control Module* and the Chapter 17, *Interrupt Controllers*.

12.4 GPU Register Manual

CAUTION

All GPU registers are limited to 32-bit data accesses; 8- and 16-bit accesses are not allowed because they can corrupt register content.

12.4.1 GPU Instance Summary

Table 12-4. GPU Instance Summary

Module Name	Base Address	Size
GPU_WRAPPER	0x5600 FE00	512 bytes

The GPU domain's base address is at 0x5600 0000. GPU address space is 32MiB wide.

12.4.2 GPU Registers

12.4.2.1 GPU_WRAPPER Register Summary

Table 12-5. GPU_WRAPPER Registers Mapping Summary

Register Name	Type	Register Width (Bits)	Address Offset	Physical Address
REVISION	R	32	0x0000 0000	0x5600 FE00
HWINFO	R	32	0x0000 0004	0x5600 FE04
SYSCONFIG	RW	32	0x0000 0010	0x5600 FE10
IRQSTATUS_RAW_0	RW	32	0x0000 0024	0x5600 FE24
IRQSTATUS_RAW_1	RW	32	0x0000 0028	0x5600 FE28
IRQSTATUS_RAW_2	RW	32	0x0000 002C	0x5600 FE2C
IRQSTATUS_0	RW	32	0x0000 0030	0x5600 FE30
IRQSTATUS_1	RW	32	0x0000 0034	0x5600 FE34
IRQSTATUS_2	RW	32	0x0000 0038	0x5600 FE38
IRQENABLE_SET_0	RW	32	0x0000 003C	0x5600 FE3C
IRQENABLE_SET_1	RW	32	0x0000 0040	0x5600 FE40
IRQENABLE_SET_2	RW	32	0x0000 0044	0x5600 FE44
IRQENABLE_CLR_0	RW	32	0x0000 0048	0x5600 FE48
IRQENABLE_CLR_1	RW	32	0x0000 004C	0x5600 FE4C
IRQENABLE_CLR_2	RW	32	0x0000 0050	0x5600 FE50
PAGE_CONFIG	RW	32	0x0000 0100	0x5600 FF00
INTERRUPT_EVENT	RW	32	0x0000 0104	0x5600 FF04
DEBUG_CONFIG	RW	32	0x0000 0108	0x5600 FF08
DEBUG_STATUS_0	R	32	0x0000 010C	0x5600 FF0C
DEBUG_STATUS_1	R	32	0x0000 0110	0x5600 FF10

12.4.2.2 GPU_WRAPPER Register Description

Table 12-6. REVISION

Address Offset	0x0000 0000		
Physical Address	0x5600 FE00	**Instance**	GPU_WRAPPER
Description	Revision register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REVISIONID																															

Bits	Field Name	Description	Type	Reset
31:0	REVISIONID	Revision value	R	See (1)

(1) TI internal data

Table 12-7. Register Call Summary for Register REVISION

GPU Register Manual
- GPU Register Summary: [0]

Table 12-8. HWINFO

Address Offset	0x0000 0004		
Physical Address	0x5600 FE04	**Instance**	GPU_WRAPPER
Description	Hardware implementation information		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																													MEM_BUS_WIDTH	SYS_BUS_WIDTH	

Bits	Field Name	Description	Type	Reset
31:3	RESERVED		R	0x0000 0000
2	MEM_BUS_WIDTH	Memory bus width Read 0x0: 64 bits Read 0x1: 128 bits	R	1
1:0	SYS_BUS_WIDTH	System bus width Read 0x0: 32 bits Read 0x1: 64 bits Read 0x2: 128 bits Read 0x3: Reserved	R	0x1

Table 12-9. Register Call Summary for Register HWINFO

GPU Register Manual
- GPU Register Summary: [0]

Table 12-10. SYSCONFIG

Address Offset	0x0000 0010		
Physical Address	0x5600 FE10	**Instance**	GPU_WRAPPER
Description	System configuration register		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																								STANDBY_MODE		IDLE_MODE		RESERVED			

Bits	Field Name	Description	Type	Reset
31:6	RESERVED		R	0x000 0000
5:4	STANDBY_MODE	Clock standby mode: 0x0: Force-standby 0x1: No-standby 0x2: Smart-standby 0x3: Reserved	RW	0x2
3:2	IDLE_MODE	Clock idle mode: 0x0: Force-standby 0x1: No-standby 0x2: Smart-standby 0x3: Reserved	RW	0x2
1:0	RESERVED		R	0x0

Table 12-11. Register Call Summary for Register SYSCONFIG

GPU Register Manual

- GPU Register Summary: [0]

Table 12-12. IRQSTATUS_RAW_0

Address Offset	0x0000 0024		
Physical Address	0x5600 FE24	**Instance**	GPU_WRAPPER
Description	Raw IRQ 0 status		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																															INIT_MINTERRUPT_RAW

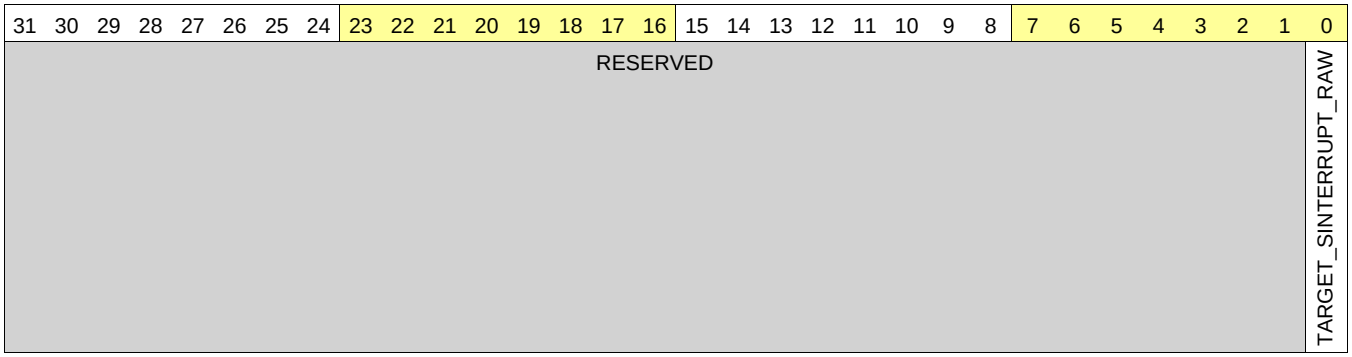
Bits	Field Name	Description	Type	Reset
31:1	RESERVED		R	0x0000 0000
0	INIT_MINTERRUPT_RAW	Interrupt 0 raw event: Write 0x0: No action Write 0x1: Set event (used for debug) Read 0x0: No event pending Read 0x1: Event pending	RW	0

Table 12-13. Register Call Summary for Register IRQSTATUS_RAW_0

GPU Register Manual
- GPU Register Summary: [0]

Table 12-14. IRQSTATUS_RAW_1

Address Offset	0x0000 0028		
Physical Address	0x5600 FE28	**Instance**	GPU_WRAPPER
Description	Raw IRQ 1 status. Slave port interrupt.		
Type	RW		



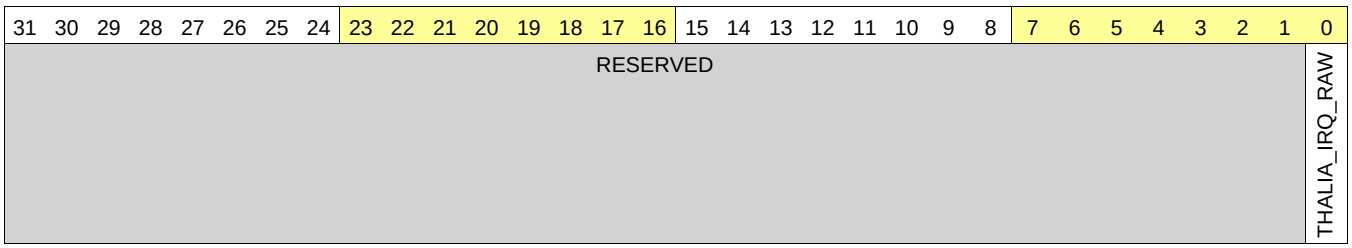
Bits	Field Name	Description	Type	Reset
31:1	RESERVED		R	0x0000 0000
0	TARGET_SINTERRUPT_RAW	Interrupt 1 raw event: Write 0x0: No action Write 0x1: Set event (used for debug) Read 0x0: No event pending Read 0x1: Event pending	RW	0

Table 12-15. Register Call Summary for Register IRQSTATUS_RAW_1

GPU Register Manual
- GPU Register Summary: [0]

Table 12-16. IRQSTATUS_RAW_2

Address Offset	0x0000 002C		
Physical Address	0x5600 FE2C	**Instance**	GPU_WRAPPER
Description	Raw IRQ 2 status. Core interrupt.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:1	RESERVED		R	0x0000 0000
0	THALIA_IRQ_RAW	Interrupt 0 raw event: Write 0x0: No action Write 0x1: Set event (used for debug) Read 0x0: No event pending Read 0x1: Event pending	RW	0

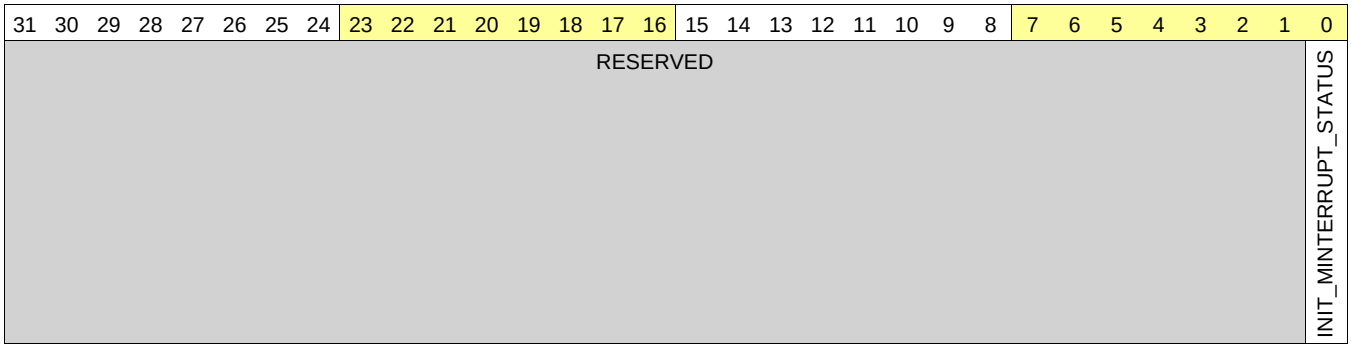
Table 12-17. Register Call Summary for Register IRQSTATUS_RAW_2

GPU Register Manual

- GPU Register Summary: [0]

Table 12-18. IRQSTATUS_0

Address Offset	0x0000 0030		
Physical Address	0x5600 FE30	**Instance**	GPU_WRAPPER
Description	Interrupt 0 status event. Master port interrupt.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:1	RESERVED		R	0x0000 0000
0	INIT_MINTERRUPT_STATUS	Interrupt 0 raw event: Write 0x0: No action Write 0x1: Clear event Read 0x0: No event pending Read 0x1: Event pending and interrupt enabled	RW	0

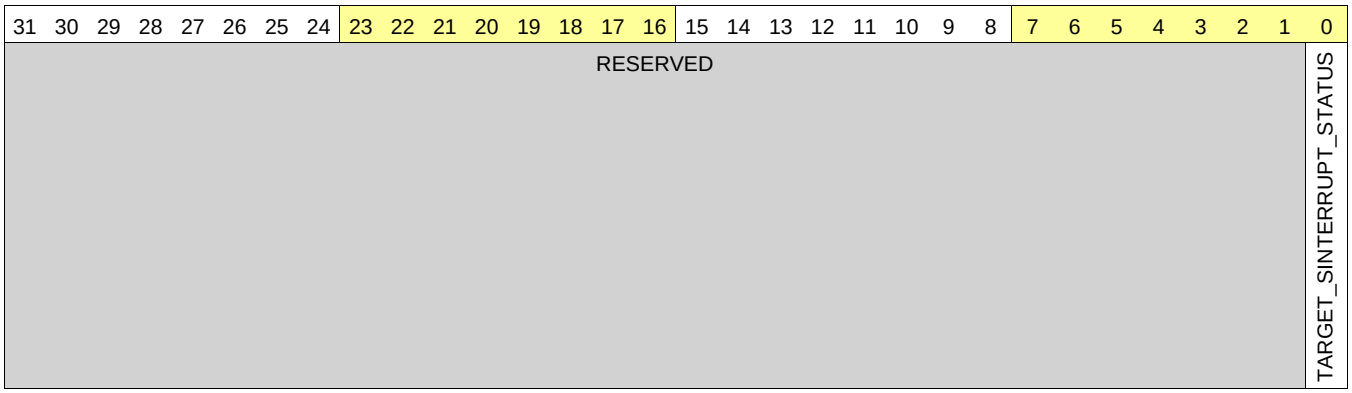
Table 12-19. Register Call Summary for Register IRQSTATUS_0

GPU Register Manual

- GPU Register Summary: [0]

Table 12-20. IRQSTATUS_1

Address Offset	0x0000 0034		
Physical Address	0x5600 FE34	**Instance**	GPU_WRAPPER
Description	Interrupt 1 - slave port status event		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:1	RESERVED		R	0x0000 0000
0	TARGET_SINTERRUPT_STATUS	Interrupt 0 raw event: Write 0x0: No action Write 0x1: Clear event Read 0x0: No event pending Read 0x1: Event pending and interrupt enabled	RW	0

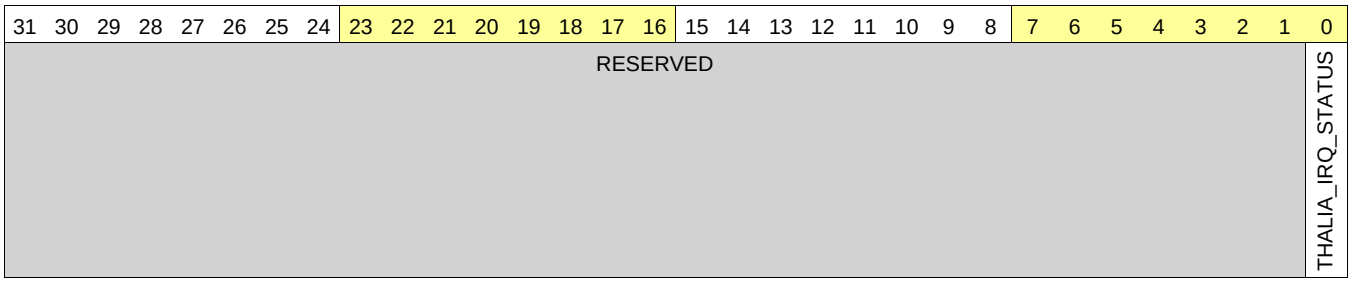
Table 12-21. Register Call Summary for Register IRQSTATUS_1

GPU Register Manual

- GPU Register Summary: [0]

Table 12-22. IRQSTATUS_2

Address Offset	0x0000 0038		
Physical Address	0x5600 FE38	**Instance**	GPU_WRAPPER
Description	Interrupt 2 - Core status event		
Type	RW		



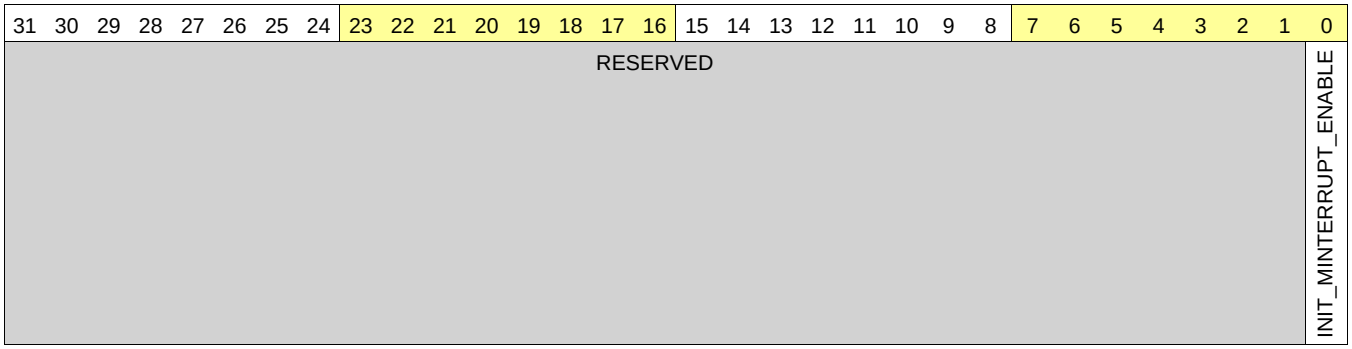
Bits	Field Name	Description	Type	Reset
31:1	RESERVED		R	0x0000 0000
0	THALIA_IRQ_STATUS	Interrupt 0 raw event: Write 0x0: No action Write 0x1: Clear event Read 0x0: No event pending Read 0x1: Event pending and interrupt enabled	RW	0

Table 12-23. Register Call Summary for Register IRQSTATUS_2

GPU Register Manual
- GPU Register Summary: [0]

Table 12-24. IRQENABLE_SET_0

Address Offset	0x0000 003C		
Physical Address	0x5600 FE3C	**Instance**	GPU_WRAPPER
Description	Enable Interrupt 0 - Master port.		
Type	RW		



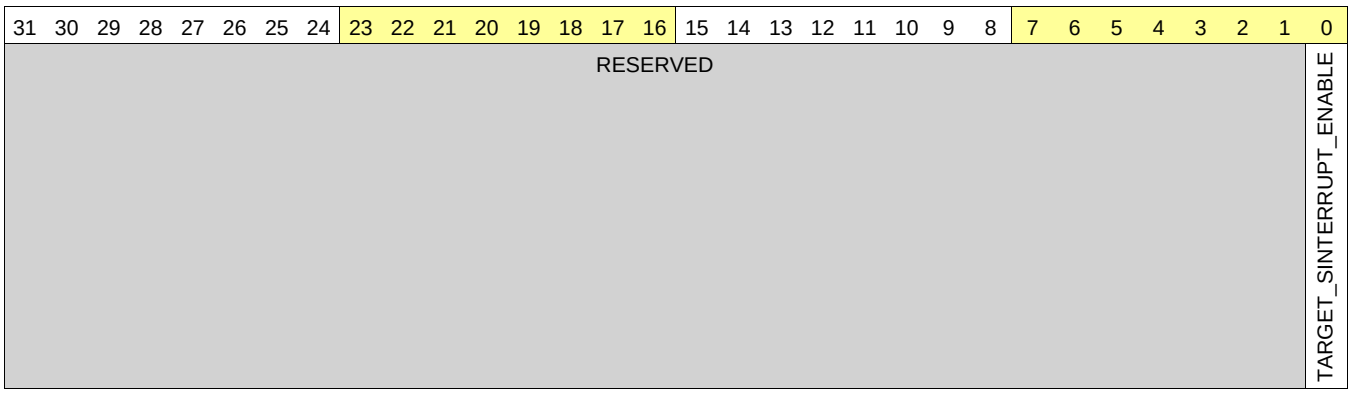
Bits	Field Name	Description	Type	Reset
31:1	RESERVED		R	0x0000 0000
0	INIT_MINTERRUPT_ENABLE	To enable interrupt: Write 0x0: No action Write 0x1: Enable interrupt Read 0x0: Interrupt is disabled Read 0x1: Interrupt is enabled	RW	0

Table 12-25. Register Call Summary for Register IRQENABLE_SET_0

GPU Register Manual
- GPU Register Summary: [0]

Table 12-26. IRQENABLE_SET_1

Address Offset	0x0000 0040		
Physical Address	0x5600 FE40	**Instance**	GPU_WRAPPER
Description	Enable Interrupt 1. Core interrupt.		
Type	RW		



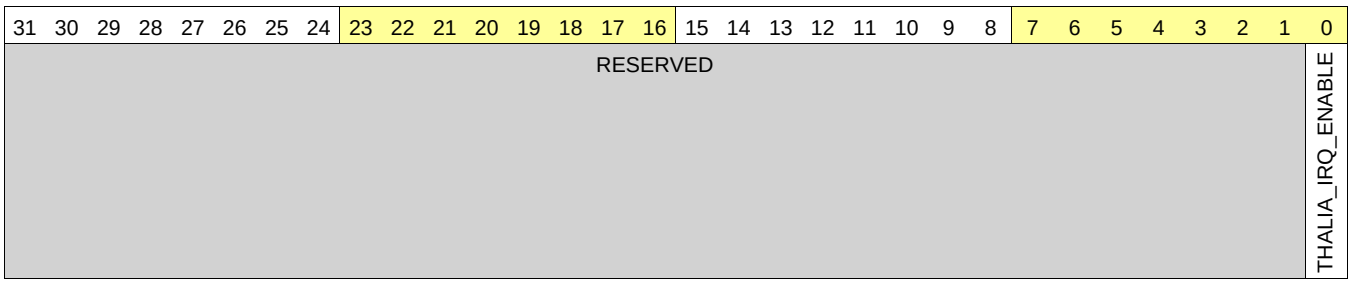
Bits	Field Name	Description	Type	Reset
31:1	RESERVED		R	0x0000 0000
0	TARGET_SINTERRUPT_ENABLE	To enable interrupt: Write 0x0: No action Write 0x1: Enable interrupt Read 0x0: Interrupt is disabled Read 0x1: Interrupt is enabled	RW	0

Table 12-27. Register Call Summary for Register IRQENABLE_SET_1

GPU Register Manual
- GPU Register Summary: [0]

Table 12-28. IRQENABLE_SET_2

Address Offset	0x0000 0044		
Physical Address	0x5600 FE44	**Instance**	GPU_WRAPPER
Description	Enable Interrupt 2. Core interrupt.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:1	RESERVED		R	0x0000 0000
0	THALIA_IRQ_ENABLE	To enable interrupt: Write 0x0: No action Write 0x1: Enable interrupt Read 0x0: Interrupt is disabled Read 0x1: Interrupt is enabled	RW	0

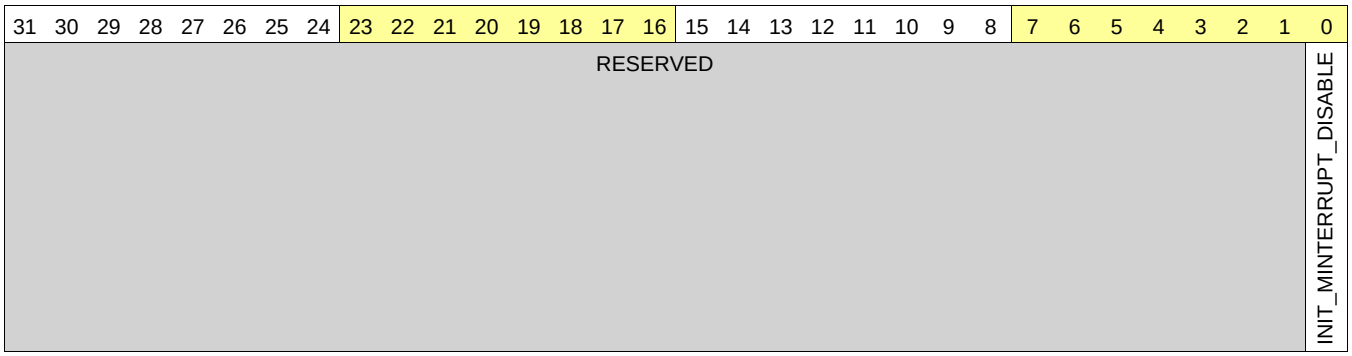
Table 12-29. Register Call Summary for Register IRQENABLE_SET_2

GPU Register Manual

- GPU Register Summary: [0]

Table 12-30. IRQENABLE_CLR_0

Address Offset	0x0000 0048		
Physical Address	0x5600 FE48	**Instance**	GPU_WRAPPER
Description	Disable Interrupt 0 - Master port.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:1	RESERVED		R	0x0000 0000
0	INIT_MINTERRUPT_DISABLE	To disable interrupt: Write 0x0: No action Write 0x1: Disable interrupt Read 0x0: Interrupt is disabled Read 0x1: Interrupt is enabled	RW	0

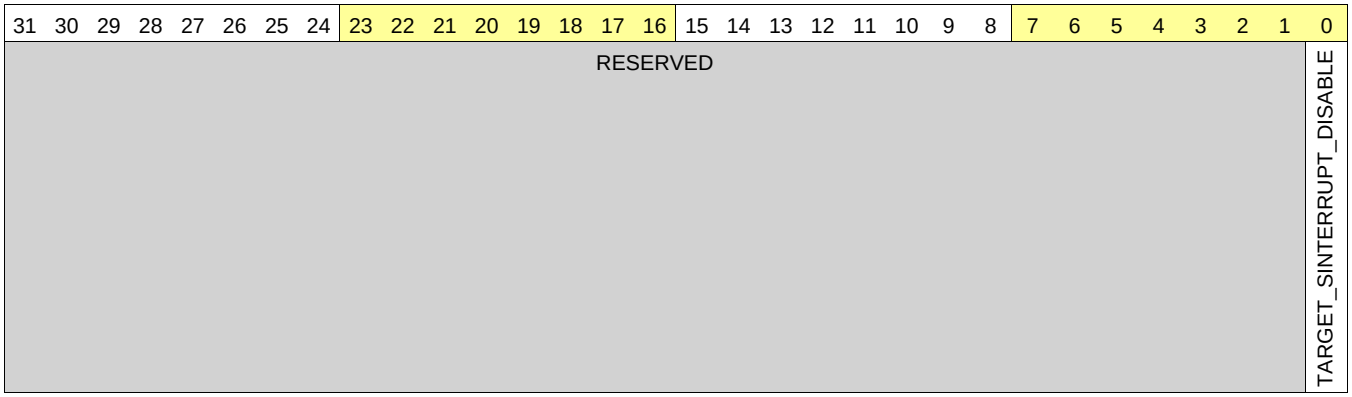
Table 12-31. Register Call Summary for Register IRQENABLE_CLR_0

GPU Register Manual

- GPU Register Summary: [0]

Table 12-32. IRQENABLE_CLR_1

Address Offset	0x0000 004C		
Physical Address	0x5600 FE4C	**Instance**	GPU_WRAPPER
Description	Disable Interrupt 2 - Core interrupt.		
Type	RW		



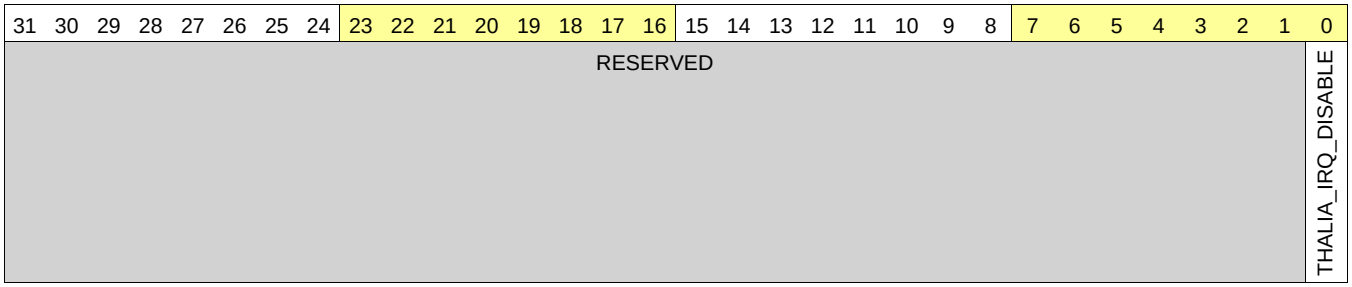
Bits	Field Name	Description	Type	Reset
31:1	RESERVED		R	0x0000 0000
0	TARGET_SINTERRUPT_DISABLE	To disable interrupt: Write 0x0: No action Write 0x1: Disable interrupt Read 0x0: Interrupt is disabled Read 0x1: Interrupt is enabled	RW	0

Table 12-33. Register Call Summary for Register IRQENABLE_CLR_1

GPU Register Manual
- GPU Register Summary: [0]

Table 12-34. IRQENABLE_CLR_2

Address Offset	0x0000 0050		
Physical Address	0x5600 FE50	**Instance**	GPU_WRAPPER
Description	Disable Interrupt 2 - Core interrupt.		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:1	RESERVED		R	0x0000 0000
0	THALIA_IRQ_DISABLE	To disable interrupt: Write 0x0: No action Write 0x1: Disable interrupt Read 0x0: Interrupt is disabled Read 0x1: Interrupt is enabled	RW	0

Table 12-35. Register Call Summary for Register IRQENABLE_CLR_2

GPU Register Manual
- GPU Register Summary: [0]

Table 12-36. PAGE_CONFIG

Address Offset	0x0000 0100		
Physical Address	0x5600 FF00	**Instance**	GPU_WRAPPER
Description	Configure memory pages.		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
THALIA_INT_BYPASS	RESERVED																										OCP_PAGE_SIZE		MEM_PAGE_CHECK_EN	MEM_PAGE_SIZE	

Bits	Field Name	Description	Type	Reset
31	THALIA_INT_BYPASS	Bypass OCP IPG interrupt logic 0x0: Do not bypass 0x1 Bypass core interrupt to I/O pin; that is, disregard the interrupt enable setting in the IPG register.	RW	0
30:5	RESERVED		R	0x000 0000
4:3	OCP_PAGE_SIZE	Defines the page size on OCP memory interface: 0x0: 4 KiB 0x1: 2 KiB 0x2: 1 KiB 0x3: 512B	RW	0x2
2	MEM_PAGE_CHECK_EN	To enable page boundary checking: 0x0: Disabled 0x1: Enabled	RW	1
1:0	MEM_PAGE_SIZE	Defines the page size on internal memory interface: 0x0: 4 KiB 0x1: 2 KiB 0x2: 1 KiB 0x3: 512B	RW	0x0

Table 12-37. Register Call Summary for Register PAGE_CONFIG

GPU Register Manual

- GPU Register Summary: [0]

Table 12-38. INTERRUPT_EVENT

Address Offset	0x0000 0104		
Physical Address	0x5600 FF04	**Instance**	GPU_WRAPPER
Description	Interrupt events		
Type	RW		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED													TARGET_INVALID_OCP_CMD	TARGET_CMD_FIFO_FULL	TARGET_RESP_FIFO_FULL	RESERVED		INT_MEM_REQ_FIFO_OVERRUN_1	INIT_READ_TAG_FIFO_OVERRUN_1	INIT_PAGE_CROSS_ERROR_1	INIT_RESP_ERROR_1	INIT_RESP_UNUSED_TAG_1	INIT_RESP_UNEXPECTED_1	RESERVED		INIT_MEM_REQ_FIFO_OVERRUN_0	INIT_READ_TAG_FIFO_OVERRUN_0	INIT_PAGE_CROSS_ERROR_0	INIT_RESP_ERROR_0	INIT_RESP_UNUSED_TAG_0	INIT_RESP_UNEXPECTED_0

Bits	Field Name	Description	Type	Reset
31:19	RESERVED		R	0x0000
18	TARGET_INVALID_OCP_CMD	Invalid command from OCP: Write 0x0: Clear the event Write 0x1: Set the event and interrupt if enabled (debug only) Read 0x0: No event pending Read 0x1: Event pending	RW	0
17	TARGET_CMD_FIFO_FULL	Command FIFO full: Write 0x0: Clear the event. Write 0x1: Set the event and interrupt if enabled (debug only). Read 0x0: No event pending Read 0x1: Event pending	RW	0
16	TARGET_RESP_FIFO_FULL	Response FIFO full: Write 0x0: Clear the event. Write 0x1: Set the event and interrupt if enabled (debug only). Read 0x0: No event pending Read 0x1: Event pending	RW	0
15:14	RESERVED		R	0x0
13	INT_MEM_REQ_FIFO_OVERRUN_1	Memory request FIFO overrun: Write 0x0: Clear the event. Write 0x1: Set the event and interrupt if enabled (debug only). Read 0x0: No event pending Read 0x1: Event pending	RW	0
12	INIT_READ_TAG_FIFO_OVERRUN_1	Read tag FIFO overrun: Write 0x0: Clear the event. Write 0x1: Set the event and interrupt if enabled (debug only). Read 0x0: No event pending Read 0x1: Event pending	RW	0
11	INIT_PAGE_CROSS_ERROR_1	Memory page had been crossed during a burst: Write 0x0: Clear the event. Write 0x1: Set the event and interrupt if enabled (debug only). Read 0x0: No event pending Read 0x1: Event pending	RW	0

Bits	Field Name	Description	Type	Reset
10	INIT_RESP_ERROR_1	Receiving error response: Write 0x0: Clear the event. Write 0x1: Set the event and interrupt if enabled (debug only). Read 0x0: No event pending Read 0x1: Event pending	RW	0
9	INIT_RESP_UNUSED_TAG_1	Receiving response on an unused OCP TAG: Write 0x0: Clear the event Write 0x1: Set the event and interrupt if enabled (debug only) Read 0x0: No event pending Read 0x1: Event pending	RW	0
8	INIT_RESP_UNEXPECTED_1	Receiving response when not expected: Write 0x0: Clear the event. Write 0x1: Set the event and interrupt if enabled (debug only). Read 0x0: No event pending Read 0x1: Event pending	RW	0
7:6	RESERVED		R	0x0
5	INIT_MEM_REQ_FIFO_OVERRUN_0	Memory request FIFO overrun; Write 0x0: Clear the event. Write 0x1: Set the event and interrupt if enabled (debug only). Read 0x0: No event pending Read 0x1: Event pending	RW	0
4	INIT_READ_TAG_FIFO_OVERRUN_0	Read tag FIFO overrun: Write 0x0: Clear the event. Write 0x1: Set the event and interrupt if enabled (debug only). Read 0x0: No event pending Read 0x1: Event pending	RW	0
3	INIT_PAGE_CROSS_ERROR_0	Memory page had been crossed during a burst. Write 0x0: Clear the event. Write 0x1: Set the event and interrupt if enabled (debug only). Read 0x0: No event pending Read 0x1: Event pending	RW	0
2	INIT_RESP_ERROR_0	Receiving error response: Write 0x0: Clear the event. Write 0x1: Set the event and interrupt if enabled (debug only). Read 0x0: No event pending Read 0x1: Event pending	RW	0
1	INIT_RESP_UNUSED_TAG_0	Receiving response on an unused OCP TAG: Write 0x0: Clear the event. Write 0x1: Set the event and interrupt if enabled (debug only). Read 0x0: No event pending Read 0x1: Event pending	RW	0
0	INIT_RESP_UNEXPECTED_0	Receiving response when not expected: Write 0x0: Clear the event. Write 0x1: Set the event and interrupt if enabled (debug only). Read 0x0: No event pending Read 0x1: Event pending	RW	0

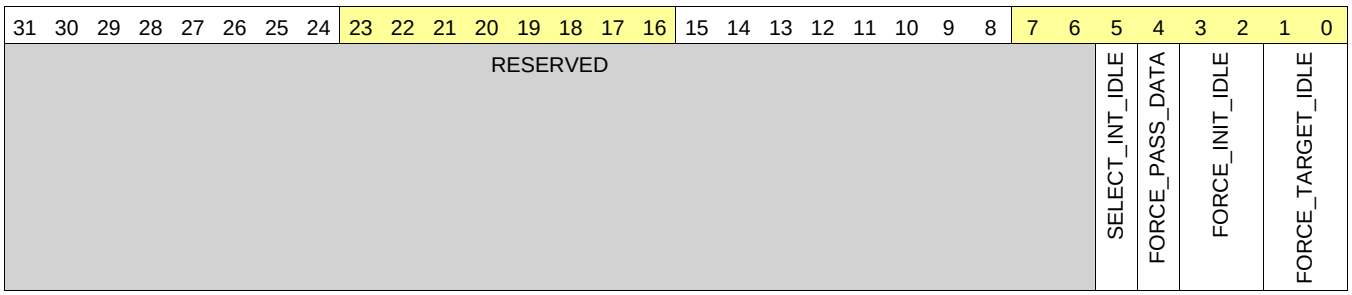
Table 12-39. Register Call Summary for Register INTERRUPT_EVENT

GPU Register Manual

- GPU Register Summary: [0]

Table 12-40. DEBUG_CONFIG

Address Offset	0x0000 0108		
Physical Address	0x5600 FF08	**Instance**	GPU_WRAPPER
Description	Configuration of debug modes		
Type	RW		



Bits	Field Name	Description	Type	Reset
31:6	RESERVED		R	0x000 0000
5	SELECT_INT_IDLE	To select which idle the disconnect protocol should act on: 0x0: Whole SGX idle 0x1: OCP initiator idle	RW	0
4	FORCE_PASS_DATA	Forces the initiator to pass data independent of disconnect protocol: 0x0: Do not force, normal operation 0x1: Never fence request to OCP	RW	0
3:2	FORCE_INIT_IDLE	Forces initiator idle: 0x0, 0x3: Do not force, normal operation 0x1: Always idle 0x2: Never idle	RW	0x0
1:0	FORCE_TARGET_IDLE	Forces target idle: 0x0, 0x3: Do not force, normal operation 0x1: Always idle 0x2: Never idle	RW	0x0

Table 12-41. Register Call Summary for Register DEBUG_CONFIG

GPU Register Manual

- GPU Register Summary: [0]

Table 12-42. DEBUG_STATUS_0

Address Offset	0x0000 010C		
Physical Address	0x5600 FF0C	**Instance**	GPU_WRAPPER
Description	Port0 debug status register		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMD_DEBUG_STATE	CMD_RESP_DEBUG_STATE	TARGET_IDLE	RESP_FIFO_FULL	CMD_FIFO_FULL	RESP_ERROR	WHICH_TARGET_REGISTER					TARGET_CMD_OUT			INIT_MSTANDBY	INIT_MWAIT	INIT_MDISCREQ	INIT_MDISCACK			INIT_SCONNECT_2	INIT_SCONNECT_1	INIT_SCONNECT_0	INIT_MCONNECT	TARGET_SIDLEACK		TARGET_SDISCACK		TARGET_SIDLEREQ	TARGET_SCONNECT	TARGET_MCONNECT	

Bits	Field Name	Description	Type	Reset
31	CMD_DEBUG_STATE	Target command state-machine: 0x0: IDLE 0x1: Accept command	R	0
30	CMD_RESP_DEBUG_STATE	Target response state-machine: 0x0: Send accept 0x1: Wait accept	R	0
29	TARGET_IDLE	Target idle	R	0
28	RESP_FIFO_FULL	Target response FIFO full	R	0
27	CMD_FIFO_FULL	Target command FIFO full	R	0
26	RESP_ERROR	Respond to OCP with error, which could be caused by either address misalignment or invalid byte enable.	R	0
25:21	WHICH_TARGET_REGISTER	Indicates which OCP target registers to read	R	0x00
20:18	TARGET_CMD_OUT	Command received from OCP: 0x0: CMD_WRSYS 0x1: CMD_RDSYS 0x2: CMD_WR_ERROR 0x3: CMD_RD_ERROR 0x4: CMD_CHK_WRADDR_PAGE (not used) 0x5: CMD_CHK_RDADDR_PAGE (not used) 0x6: CMD_TARGET_REG_WRITE 0x7: CMD_TARGET_REG_READ	R	0x0
17	INIT_MSTANDBY	Status of init_MStandby signal	R	0
16	INIT_MWAIT	Status of init_MWait signal	R	0
15	INIT_MDISCREQ	Request to disconnect from OCP interface	R	0
14:13	INIT_MDISCACK	Disconnect status of the OCP interface: 0x0: FUNCT 0x1: TRANS 0x2: Reserved 0x3: IDLE	R	0x0
12	INIT_SCONNECT_2	Defines whether to wait in M_WAIT state for MConnect FSM: 0x0: Skip M_WAIT state 0x1: Wait in M_WAIT state	R	0
11	INIT_SCONNECT_1	Defines the busy-ness state of the slave: 0x0: Slave is drained 0x1: Slave is loaded	R	0
10	INIT_SCONNECT_0	Disconnect from slave: 0x0: Disconnect request from slave 0x1: Connect request from slave	R	0

Bits	Field Name	Description	Type	Reset
9:8	INIT_MCONNECT	Initiator MConnect state: 0x0: M_OFF 0x1: M_WAIT 0x2: M_DISC 0x3: M_CON	R	0x0
7:6	TARGET_SIDLEACK	Acknowledge the SIdleAck state-machine: 0x0: FUNCT 0x1: SLEEP TRANS 0x2: Reserved 0x3: IDLE	R	0x0
5:4	TARGET_SDISCACK	Acknowledge the SDiscAck state-machine: 0x0: FUNCT 0x1: TRANS 0x2: Reserved 0x3: IDLE	R	0x0
3	TARGET_SIDLEREQ	Request the target to go idle: 0 Do not go idle, or go active 1 Go idle	R	0
2	TARGET_SCONNECT	Target SConnect bit 0 state: 0x0: Disconnect interface 0x1: Connect OCP interface	R	0
1:0	TARGET_MCONNECT	Target MConnect state: 0x0: M_OFF 0x1: M_WAIT 0x2: M_DISC 0x3: M_CON	R	0x0

Table 12-43. Register Call Summary for Register DEBUG_STATUS_0

GPU Register Manual

- GPU Register Summary: [0]

Table 12-44. DEBUG_STATUS_1

Address Offset	0x0000 0110		
Physical Address	0x5600 FF10	**Instance**	GPU_WRAPPER
Description	Port1 debug status register		
Type	R		

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMD_DEBUG_STATE	CMD_RESP_DEBUG_STATE	TARGET_IDLE	RESP_FIFO_FULL	CMD_FIFO_FULL	RESP_ERROR	WHICH_TARGET_REGISTER					TARGET_CMD_OUT			INIT_MSTANDBY	INIT_MWAIT	INIT_MDISCREQ	INIT_MDISCACK	INIT_SCONNECT_2	INIT_SCONNECT_1	INIT_SCONNECT_0	INIT_MCONNECT			TARGET_SIDLEACK		TARGET_SDISCACK		TARGET_SIDLEREQ	TARGET_SCONNECT	TARGET_MCONNECT	

Bits	Field Name	Description	Type	Reset
31	CMD_DEBUG_STATE	Target command state-machine: 0x0: IDLE 0x1: Accept command	R	0
30	CMD_RESP_DEBUG_STATE	Target response state-machine: 0x0: Send accept 0x1: Wait accept	R	0
29	TARGET_IDLE	Target idle	R	0

Bits	Field Name	Description	Type	Reset
28	RESP_FIFO_FULL	Target response FIFO full	R	0
27	CMD_FIFO_FULL	Target command FIFO full	R	0
26	RESP_ERROR	Respond to OCP with error, which could be caused by either address misalignment or invalid byte enable.	R	0
25:21	WHICH_TARGET_REGISTER	Indicates which OCP target registers to read	R	0x00
20:18	TARGET_CMD_OUT	Command received from OCP: 0x0: CMD_WRSYS 0x1: CMD_RDSYS 0x2: CMD_WR_ERROR 0x3: CMD_RD_ERROR 0x4: CMD_CHK_WRADDR_PAGE (not used) 0x5: CMD_CHK_RDADDR_PAGE (not used) 0x6: CMD_TARGET_REG_WRITE 0x7: CMD_TARGET_REG_READ	R	0x0
17	INIT_MSTANDBY	Status of init_MStandby signal	R	0
16	INIT_MWAIT	Status of init_MWait signal	R	0
15	INIT_MDISCREQ	Request to disconnect from OCP interface	R	0
14:13	INIT_MDISCACK	Disconnect status of the OCP interface: 0x0: FUNCT 0x1: SLEEP TRANS 0x2: Reserved 0x3: IDLE	R	0x0
12	INIT_SCONNECT_2	Defines whether to wait in M_WAIT state for MConnect FSM: 0x0: Skip M_WAIT state. 0x1: Wait in M_WAIT state.	R	0
11	INIT_SCONNECT_1	Defines the busy-ness state of the slave: 0x0: Slave is drained. 0x1: Slave is loaded.	R	0
10	INIT_SCONNECT_0	Disconnect from slave: 0x0: Disconnect request from slave 0x1: Connect request from slave	R	0
9:8	INIT_MCONNECT	Initiator MConnect state: 0x0: M_OFF 0x1: M_WAIT 0x2: M_DISC 0x3: M_CON	R	0x0
7:6	TARGET_SIDLEACK	Acknowledge the SIdleAck state-machine: 0x0: FUNCT 0x1: SLEEP TRANS 0x2: Reserved 0x3: IDLE	R	0x0
5:4	TARGET_SDISCACK	Acknowledge the SDiscAck state-machine: 0x0: FUNCT 0x1: TRANS 0x2: Reserved 0x3: IDLE	R	0x0
3	TARGET_SIDLEREQ	Request the target to go idle: 0x0: Do not go idle, or go active 0x1: Go idle	R	0
2	TARGET_SCONNECT	Target SConnect bit 0 state: 0x0: Disconnect interface 0x1: Connect OCP interface	R	0
1:0	TARGET_MCONNECT	Target MConnect state: 0x0: M_OFF 0x1: M_WAIT 0x2: M_DISC 0x3: M_CON	R	0x0

Table 12-45. Register Call Summary for Register DEBUG_STATUS_1

GPU Register Manual

- GPU Register Summary: [0]