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

Volume 5: Workarounds

July 2022, Revision 1.0

Table of Contents

Notices and Disclaimers

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Code names are used by Intel to identify products, technologies, or services that are in development and not publicly available. These are not "commercial" names and not intended to function as trademarks

Customer is responsible for safety of the overall system, including compliance with applicable safety-related requirements or standards.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document, with the sole exceptions that a) you may publish an unmodified copy and b) code included in this document is licensed subject to Zero-Clause BSD open source license (0BSD). You may create software implementations based on this document and in compliance with the foregoing that are intended to execute on the Intel product(s) referenced in this document. No rights are granted to create modifications or derivatives of this document.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

Workarounds

impact	title	bspec_wa_details	sku_impact
	Audio 8K1port - For certain VDSC bpp settings, hblank asserts before hblank_early, leading to a bad audio state	WA details can be found at: "Audio Hblank early enable sequence"	sku: ALL; Stepping_impacted: a0; stepping_fixed: ; wa_status: driver_permanent_wa
data_corruption	Underrun when FBC is compressing with odd plane size and first segment is only 3 lines	FBC causes screen corruption when plane size is odd for vertical and horizontal. Set 0x43224 bit 14 to 1 before enabling FBC. It is okay to leave it set when FBC is disabled.	sku: ALL; stepping_impacted: a0; stepping_fixed: ; wa_status: driver_permanent_wa
data_corruption	Corruption with FBC and plane enable/disable	Corruption with FBC around plane 1A enabling. In the Frame Buffer Compression programming sequence "Display Plane Enabling with FBC" add a wait for vblank between plane enabling step 1 and FBC enabling step 2.	sku: ALL; stepping_impacted: a0; stepping_fixed: ; wa_status: driver_permanent_wa
	DARBFunit early clock gating leading to underrun	Disable clock gating for DARBFunit. Set register offset 0x46530 bit 27 (DARBF Gating Dis) to 1 before first enabling display planes or cursors and keep set. No need to clear after disabling planes	sku: ALL; stepping_impacted: a0; stepping_fixed: ; wa_status: driver_permanent_wa
	BW Buddy CTL Register has incorrect default value for TLB Request timeout	Program BW_BUDDY_CTL0 and BW_BUDDY_CTL1 "TLB Request Timer" field to 8h.	sku: ALL; stepping_impacted: a0; stepping_fixed: ; wa_status: driver_permanent_wa
	PCH display clock remains active when it shouldn't; impact to power and sleep state residency	Display driver should set and clear register offset 0xC2000 bit #7 as last step in programming south display registers in preparation for entering S0ix state.	sku: ALL; stepping_impacted: a0; stepping_fixed: ; wa_status: driver_permanent_wa
	dupunit not generating line_pop indication for plane with minimum size	plane horizontal minimum size in PLANE_SIZE register need to be increased according to the following: 8bpp: 18 16bpp: 10 32bpp,yuv212,yuv216: 6 64bpp: 4 NV12: 20 P010,P012,P016: 12	sku: ALL; stepping_impacted: a0; stepping_fixed: ; wa_status: driver_permanent_wa
	PLANE_CC_VAL not getting updated immediately on async flip	Display async flips will not update the clear color value at the right point. Limit use of display clear color to sync flips.	sku: ALL; stepping_impacted: a0; stepping_fixed: ; wa_status: driver_permanent_wa
	Remove PM Req with unblock/memup + fill support -- SAGV enhancement not working as expected	SAGV fill timeout. Set 0x46434 bits 24 ,25, 26, and 27 to 1 at display initialization.	sku: ALL; stepping_impacted: a0; stepping_fixed: ; wa_status: driver_permanent_wa
	Underrun can occur in certain cases when FBC is enabled	For non-modulo 4 plane size(including plane size + yoffset), disable FBC when scanline is Vactive -10	sku: ALL; stepping_impacted: a0; stepping_fixed: ; wa_status: driver_permanent_wa

impact	title	bspec_wa_details	sku_impact
data_corruption	Display underrun can occur on cursor plane if WM0 is used without WM1	Cursor watermark 1 (WM1) value is being used when WM1 is disabled, causing underrun in some cases. WA: When cursor WM1 is disabled, copy contents of CUR_WM_0[30:0] (exclude the enable bit) into CUR_WM_1[30:0].	sku: ALL; stepping_impacted: a0; stepping_fixed: ; wa_status: driver_permanent_wa
other	PCH display HPD IRQ is not detected with default filter value	Program 0xC7204 (PP_CONTROL) bit #0 to '1' to enable workaround and clear to disable it. Driver shall enable this WA when external display is connected and remove WA when display is unplugged or before going into sleep to allow CS entry. Driver shall not enable WA when eDP is connected.	sku: ALL; stepping_impacted: a0; stepping_fixed: ; wa_status: driver_permanent_wa
data_corruption	BS-BS Jitter in DP output with MST + DSC + FEC enabled	Issue: Higher than expected line to line jitter of the horizontal blank start when using DP multi-stream with FEC (mainly used for DSC). WA: Adjust the transcoder data M and N values to fully utilize the VC payload by setting data M = VC payload size and data N = 64. This over allocates data within the payload, causing line pixel data to finish consistently early.	sku: ALL; stepping_impacted: a0; stepping_fixed: ; wa_status: driver_permanent_wa
data_corruption	DARBF read pointer does not clear upon DC5 entry causing spurious requests to be sent out during DC5 exit	Screen glitches with DC5 or DC6 enabled and potentially with planes enabled and then all disabled. Software needs to write 101038[1] = 0 before first enabling any planes or cursor. This will keep the IOSF ISM clkreq for primary clock asserted while memory is up.	sku: ALL; stepping_impacted: b0; stepping_fixed: ; wa_status: driver_permanent_wa
hang	Panel Flicker after press F11 or Alt+Tab switch tasks under system.	Corruption seen when FBC is first enabled. After setting the FBC enable, wait for the next start of vblank, then write the plane 1A surface address register.	sku: ALL; stepping_impacted: a0; stepping_fixed: ; wa_status: driver_permanent_wa
other	Underrun observed on PSR exit in HRR timings with small vblank	High refresh rate panels with small vblank size (either because of the panel vblank size or the internal delayed vblank) must have some watermark levels disabled to avoid underrun when the memory latency longer than vblank, as explained in the display programming section on High Refresh Rate Support. PSR1, PSR2, and Panel Replay have an optimization to override the enabled watermark level to the maximum, so that override must be disabled for these small vblanks. Refer to the display programming section on High Refresh Rate Support for the instructions on how to disable the optimization.	sku: ALL; stepping_impacted: a0; stepping_fixed: ; wa_status: driver_permanent_wa
data_corruption	Screen flicker with FBC and package C states enabled	Set 0x43224 bit13 before enabling FBC. Leave it set while FBC is enabled. There is no need to clear it afterwards.	sku: ALL; stepping_impacted: a0; stepping_fixed: ; wa_status: driver_permanent_wa

impact	title	bspec_wa_details	sku_impact
other	Underflow and screen shift when PSR2 enable/disable during update frames	PSR2 defeatured. Do not enable it.	sku: ALL; stepping_impacted: a0; stepping_fixed: ; wa_status: driver_permanent_wa
	Remove CCS override and credit counter reset fix from DARBA	Media compression (AKA End2End compression) must be disabled for displayable media surfaces. Do not enable Media Decompression in the display plane control registers.	sku: ALL; stepping_impacted: a0; stepping_fixed: c0; wa_status: driver_permanent_wa
	Gfx IOMMU should not report ECS caps bit since extended context support is deprecated from VTd spec; Additionally, Gfx IOMMU reports a number of other ECAP bits which, per spec, should only be reported if SMTS is supported & Gfx IOMMU doesn't support SMTS	To override Gfx IOMMU ECAP reporting to match the Vtd Spec (as of June 2019), BIOS must program the IOMMU_DEFEATURE_CAPECAPDIS register as follows: read the current value of 0x101048, "OR" that value with 0x13FC000 and write the result back into 0x101048 IOMMU_DEFEATURE_CAPECAPDIS (0x101048) is part of BIOSEARLY policy group. So, it can be accessed by BIOS. This will override the following caps bits to be conformant to the spec. ECS, DIS - deprecated MTS, NEST, PRS, SRS, EAFS, PASID - may only be set if SMTS is set, SMTS is not enabled on this product ERS - may only be set if PASID is enabled (see above) Note that of the override bits may not be strictly needed because the existing ECAP default value already reports some of these caps bits as 0... the implementation listed here is a simplification to avoid having to have a bunch of conditionals (e.g. check ECAP bit and then set defeature bit accordingly).	sku: ALL; stepping_impacted: a0; stepping_fixed: ; wa_status: driver_permanent_wa
performance	Edp panel will flicker when system idle at desktop with specific background picture	WA: The driver needs to program the FBC_STRIDE (0x43228) and enable the override stride once. The override stride should be programmed with : Compressed buffer seg stride (in CLs) = ceiling[(at least plane width in pixels * 4 * 4) / (64 * compression limit factor)] + 1 If the CFB size computed by: CFB size (in bytes) = Compressed buffer seg stride * Ceiling(MIN(FBC compressed vertical limit/4, plane vertical source size/4)) * 64, will not fit into the memory allocated to FBC, then driver will need to use a more aggressive compression limit factor.	sku: ALL; stepping_impacted: a0; stepping_fixed: ; wa_status: driver_permanent_wa
hang	Blank screen seen with 4 MST Displays	Issue: Blank screen with 4 MST Displays WA:	sku: ALL; stepping_impacted: a0; stepping_fixed: ; wa_status: driver_permanent_wa

<table>
<tr><th>impact</th><th>title</th><th>bspec_wa_details</th><th>sku_impact</th></tr>
<tr><td></td><td></td><td>
If MST primary is being enabled, clear DP VC Payload Bit before start of MST enable sequence and set is as part of regular MST enable sequence.

If MST secondary is being added to the MST primary transcoder, keep the VC Payload allocate bit of secondary stream set throughout the MST secondary enable sequence.

Clear DP VC Payload Bit only for MST/DP2.0 case before Wait for ACT Sent Status Handshake during Disable Sequence

Keep DP VC Payload Bit ON as part of HDMI/DVI Enable/Disable Sequence.

When enabling non-MST cases (eDP/DP-SST/HDMI/DVI), MstTransportSelect in TRANS_DDI_FUNC_CTL must be programmed to match the assigned pipe.

DP_VC_PAYLOAD bit programming during SST/MST Enable sequences
<table>
<tr><th>Scenario</th><th>Old</th><th>New</th></tr>
<tr><td>SST Enable</td><td>set</td><td>set</td></tr>
<tr><td>SST Disable</td><td>untouch</td><td>untouch</td></tr>
<tr><td>MST Enable [primary]</td><td>untouch</td><td>Clear (before start of the MST enable sequence)
Set (Set as part of regular MST enable sequence)</td></tr>
<tr><td>MST Disable [primary]</td><td>clear</td><td>clear</td></tr>
<tr><td>MST Enable [secondary]</td><td>untouch</td><td>untouch</td></tr>
<tr><td>MST Disable [secondary]</td><td>clear</td><td>clear</td></tr>
<tr><td>HDMI/DVI Enable</td><td>set</td><td>set</td></tr>
<tr><td>HDMI/DVI Disable</td><td>untouch</td><td>untouch</td></tr>
</table>
</td><td></td></tr>
</table>