
Observability Performance Counters for Intel® Core™ Processor Family

NOTICE:
This document contains information on products in the design phase of development, and Intel® reserves the right to add or remove product features at any time, with or without changes to this open source documentation.
Creative Commons License

You are free to Share — to copy, distribute, display, and perform the work

Under the following conditions:

Attribution. You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).

No Derivative Works. You may not alter, transform, or build upon this work.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

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.

The products described in this document 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.

Implementations of the I2C bus/protocol may require licenses from various entities, including Philips Electronics N.V. and North American Philips Corporation.

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2013, Intel Corporation. All rights reserved.
# Observability

## Table of Contents

- Observability Overview ................................................................. 4
  - Device Tags and Definitions .......................................................... 4
- Trace ........................................................................................................ 5
- Performance Visibility ........................................................................... 6
  - Motivation For Hardware-Assisted Performance Visibility .................... 6
- Performance Event Counting ................................................................. 6
- HW Support .......................................................................................... 6
- Performance Counter Reporting ........................................................... 6
- Performance Counter Report Formats .................................................... 7
- Aggregating Counters ........................................................................... 12
- Aggregating Counters ........................................................................... 13
- MI_REPORT_PERF_COUNT ..................................................................... 16
- Performance Statistics Registers ........................................................... 18
Observability Overview

As GFX-enabled systems and usage models have grown in complexity over time, a number of HW features have been added specifically for the purpose of providing more insight into HW behavior while running a commercially available operating system. This chapter documents these features with pointers to relevant sections in other chapters. Supported observability features include:

- Performance counters
- Various other debug registers

**NOTE:** This document is intended to be used as a companion document to describe the ability to monitor performance for the various Intel graphics open source programmer’s reference manuals. Please review those documents to understand the terms, functionality and details for a specific Intel graphics device.

Device Tags and Definitions

The following table lists device “tags” (abbreviations) used in various parts of this document as aliases for the device names. Note that stepping information is sometimes appended to the device tag, e.g., DevSNB:E.

Information without any device tagging is applicable to all devices.

<table>
<thead>
<tr>
<th>Device Tag</th>
<th>Program Name</th>
<th>Graphics Architecture</th>
<th>SKU</th>
<th>Product Name / Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNB</td>
<td>SandyBridge</td>
<td>Gen6</td>
<td>GT1</td>
<td>SandyBridge GT1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GT2</td>
<td>SandyBridge GT2</td>
</tr>
<tr>
<td>IVB</td>
<td>IvyBridge</td>
<td>Gen7</td>
<td>GT1</td>
<td>IvyBridge GT1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GT2</td>
<td>IvyBridge GT2</td>
</tr>
<tr>
<td>VLV</td>
<td>Valley View</td>
<td>Gen7LP</td>
<td>GT</td>
<td>Low-power variation of IVB. VLV, VLV1, VLV1, and VLV:X0 are synonymous terms.</td>
</tr>
<tr>
<td>HSW</td>
<td>Haswell</td>
<td>Gen7.5</td>
<td>GT1</td>
<td>Haswell GT1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GT2</td>
<td>Haswell GT2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GT3</td>
<td>Haswell GT3</td>
</tr>
</tbody>
</table>
Trace

This section contains the following contents:

- Performance Visibility
Performance Visibility

Motivation For Hardware-Assisted Performance Visibility

As the focus on GFX performance and programmability has increased over time, the need for hardware (HW) support to rapidly identify bottlenecks in HW and efficiently tune the work sent to same has become correspondingly important. This part of the PRM describes the HW support for Performance Visibility.

Performance Event Counting

An earlier generation introduced dedicated GFX performance counters to address key issues associated with existing chipset CHAPs counters (lack of synchronization with GFX rendering work and low sampling frequency achievable when sampling via CPU MMIO read). The dedicated counter values are written to memory whenever an MI_REPORT_PERF_COUNT command is placed in the ring buffer.

While this approach eliminated much of the error associated with the previous approaches, it is still limited to sampling the counters only at the boundaries between ring commands. This inherently limited the ability of performance analysis tools to drill down into a primitive, which can contain thousands of triangles and require several hundreds of milliseconds to render.

DevIVB enhanced the aggregating counters to support the additional thread types generated by more advanced graphics APIs that support advanced features such as hull and domain shaders. The high rate at which interesting internal events can occur motivated adding an interrupt-generation capability so that HW could notify SW when the data buffer was approaching full.

DevHSW enhances support for high reporting frequencies by increasing the report buffer size in order to allow SW sufficient time between performance monitoring interrupts, enabling single run histogramming support for events like pixels per polygon. Issues with DevSNB support drove enhancements to enable performance monitoring with RC6 enabled, different report buffer ring wrap behavior, and MMIO visibility into performance counters.

HW Support

This section contains various reporting counters and registers for hardware support for Performance Visibility.

Performance Counter Reporting

When the MI_REPORT_PERF_COUNT command is received, a snapshot of the performance counter values is written to memory. The format used by HW for such reports is selected using the Counter Select field within the OACONTROL register. The organization and number of report formats vary per project and are detailed in the following section. In the following layouts, the RPT_ID is always stored in the lowest addressed DWORD.

[DevSNB]: Under conditions with high memory traffic, the values of the counters may not all be sampled at the same point in time.

[DevIVB]: In order to ensure coherent sampling of the counters, the counters are frozen and will not advance while sampling and reporting to memory is in progress. This may result in small counting errors in internally triggered reporting modes.
When an over flow condition occurs and the buffers need to be reset, or when software wants to change the OABUFFER to point to a new area in memory, Programming of the performance ring must follow the sequence below.

- Clear OA enable bit by writing 0x2360[0] = 0
- Write OASTATUS2
- Write OABUFFER
- Write OASTATUS1
- Set OA enable bit by writing 0x2360[0] = 1

When software wants to reinitialize the OA buffer Tail pointer should follow the below sequence.

- Clear OA enable bit by writing 0x2360[0] = 0
- Ensure Render command streamer doesn’t execute MI_REPORT_PERF_COUNT command until OA is enabled.
  - This can be achieved by ensuring render engine is IDLE and blocked from executing any new MI_REPORT_PERF_COUTN commands OR
  - Program the OA buffer programming through a series of MI_LOAD_REGISTER_IMM commands through render engine command buffer.
- Program OA Tail pointer, if required initialize OA head pointer.
- Set OA enable bit by writing 0x2360[0] = 1

[DevSNB] [Dev IVB] [DevVLV] [DevHSW] [all steppings] [all skus]: OA unit Is using render clock for its functionality. When trunk level clock gating takes place, OA clock would be gated, unable to count the events from non-render clock domain. Render clock gating must be disabled when OA is enabled to count the events from non-render domain. Unit level clock gating for RCS should also be disabled.

Performance Counter Report Formats

[DevSNB]

Counter Select = 000

<table>
<thead>
<tr>
<th>A-Cntr 0</th>
<th>A-Cntr 1</th>
<th>A-Cntr 2</th>
<th>A-Cntr 3</th>
<th>A-Cntr 4</th>
<th>TIME_STAMP</th>
<th>RPT_ID</th>
</tr>
</thead>
</table>

Counter Select = 001

<table>
<thead>
<tr>
<th>A-Cntr 0</th>
<th>A-Cntr 1</th>
<th>A-Cntr 2</th>
<th>A-Cntr 3</th>
<th>A-Cntr 4</th>
<th>TIME_STAMP</th>
<th>RPT_ID</th>
</tr>
</thead>
</table>
### Counter Select = 010

<table>
<thead>
<tr>
<th></th>
<th>A-Cntr 0</th>
<th>A-Cntr 1</th>
<th>A-Cntr 2</th>
<th>A-Cntr 3</th>
<th>A-Cntr 4</th>
<th>TIME_STAMP</th>
<th>RPT_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>B-Cntr 3</td>
<td>B-Cntr 2</td>
<td>B-Cntr 1</td>
<td>B-Cntr 0</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

### Counter Select = 011

<table>
<thead>
<tr>
<th></th>
<th>A-Cntr 0</th>
<th>A-Cntr 1</th>
<th>A-Cntr 2</th>
<th>A-Cntr 3</th>
<th>A-Cntr 4</th>
<th>TIME_STAMP</th>
<th>RPT_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>B-Cntr 3</td>
<td>B-Cntr 2</td>
<td>B-Cntr 1</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

### [DevIVB] [DevVLV] [DevVLVT]

### Counter Select = 000

<table>
<thead>
<tr>
<th></th>
<th>A-Cntr 4</th>
<th>A-Cntr 3</th>
<th>A-Cntr 2</th>
<th>A-Cntr 1</th>
<th>A-Cntr 0</th>
<th>TIME_STAMP</th>
<th>RPT_ID</th>
</tr>
</thead>
</table>

### Counter Select = 001

<table>
<thead>
<tr>
<th></th>
<th>A-Cntr 4</th>
<th>A-Cntr 3</th>
<th>A-Cntr 2</th>
<th>A-Cntr 1</th>
<th>A-Cntr 0</th>
<th>TIME_STAMP</th>
<th>RPT_ID</th>
</tr>
</thead>
</table>
### Counter Select = 010

<table>
<thead>
<tr>
<th>A-CNTR 4</th>
<th>A-CNTR 3</th>
<th>A-CNTR 2</th>
<th>A-CNTR 1</th>
<th>A-CNTR 0</th>
<th>TIME_STAMP</th>
<th>RPT_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>B-CNTR 3</td>
<td>B-CNTR 2</td>
<td>B-CNTR 1</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

### Counter Select = 011

<table>
<thead>
<tr>
<th>A-CNTR 4</th>
<th>A-CNTR 3</th>
<th>A-CNTR 2</th>
<th>A-CNTR 1</th>
<th>A-CNTR 0</th>
<th>TIME_STAMP</th>
<th>RPT_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>B-CNTR 3</td>
<td>B-CNTR 2</td>
<td>B-CNTR 1</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

### Counter Select = 101

<table>
<thead>
<tr>
<th>A-CNTR 4</th>
<th>A-CNTR 3</th>
<th>A-CNTR 2</th>
<th>A-CNTR 1</th>
<th>A-CNTR 0</th>
<th>TIME_STAMP</th>
<th>RPT_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>B-CNTR 3</td>
<td>B-CNTR 2</td>
<td>B-CNTR 1</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
**Counter Select = 110**

<table>
<thead>
<tr>
<th>Reserved</th>
<th>Reserved</th>
<th>Reserved</th>
<th>Reserved</th>
<th>TIME_STAMP</th>
<th>RPT_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

**[DevHSW] Counters layout for various values of select from the register:**

**Counter Select = 000**

<table>
<thead>
<tr>
<th>A-Cntr 4</th>
<th>A-Cntr 3</th>
<th>A-Cntr 2</th>
<th>A-Cntr 1</th>
<th>A-Cntr 0</th>
<th>TIME_STAMP</th>
<th>RPT_ID</th>
</tr>
</thead>
</table>

**Counter Select = 001**

<table>
<thead>
<tr>
<th>A-Cntr 4</th>
<th>A-Cntr 3</th>
<th>A-Cntr 2</th>
<th>A-Cntr 1</th>
<th>A-Cntr 0</th>
<th>TIME_STAMP</th>
<th>RPT_ID</th>
</tr>
</thead>
</table>

**Counter Select = 010**

<table>
<thead>
<tr>
<th>A-Cntr 4</th>
<th>A-Cntr 3</th>
<th>A-Cntr 2</th>
<th>A-Cntr 1</th>
<th>A-Cntr 0</th>
<th>TIME_STAMP</th>
<th>RPT_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>B-Cntr 7</td>
<td>B-Cntr 6</td>
<td>B-Cntr 5</td>
<td>B-Cntr 4</td>
<td>B-Cntr 3</td>
<td>B-Cntr 2</td>
<td>B-Cntr 1</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

**Counter Select = 011**

<table>
<thead>
<tr>
<th>A-Cntr 4</th>
<th>A-Cntr 3</th>
<th>A-Cntr 2</th>
<th>A-Cntr 1</th>
<th>A-Cntr 0</th>
<th>TIME_STAMP</th>
<th>RPT_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>B-Cntr 7</td>
<td>B-Cntr 6</td>
<td>B-Cntr 5</td>
<td>B-Cntr 4</td>
<td>B-Cntr 3</td>
<td>B-Cntr 2</td>
<td>B-Cntr 1</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
**Counter Select = 100**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>INST ADD</th>
<th>TIME_STAMP</th>
<th>RPT_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>B-Cntr 3</td>
<td>B-Cntr 2</td>
<td>B-Cntr 1</td>
<td>B-Cntr 0</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

**Counter Select = 101**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>A-Cntr 4</th>
<th>A-Cntr 3</th>
<th>A-Cntr 2</th>
<th>A-Cntr 1</th>
<th>A-Cntr 0</th>
<th>TIME_STAMP</th>
<th>RPT_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>B-Cntr 7</td>
<td>B-Cntr 6</td>
<td>B-Cntr 5</td>
<td>B-Cntr 4</td>
<td>B-Cntr 3</td>
<td>B-Cntr 2</td>
<td>B-Cntr 1</td>
<td>B-Cntr 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Counter Select = 110**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>INST ADD</th>
<th>TIME_STAMP</th>
<th>RPT_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>B-Cntr 3</td>
<td>B-Cntr 2</td>
<td>B-Cntr 1</td>
<td>B-Cntr 0</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

**Counter Select = 111**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>INST ADD</th>
<th>TIME_STAMP</th>
<th>RPT_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>B-Cntr 7</td>
<td>B-Cntr 6</td>
<td>B-Cntr 5</td>
<td>B-Cntr 4</td>
<td>B-Cntr 3</td>
<td>B-Cntr 2</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
## Aggregating Counters

<table>
<thead>
<tr>
<th>Cntr #</th>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A0</td>
<td>Aggregated Core Array Active</td>
<td>The sum of all cycles on all cores spent actively executing instructions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This does not count the time taken to service Send instructions. This time is considered by shader active counters to give the result.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[DevSNB]: A23 and A24 may be incorrect when post shader Z and/or stencil tests are required.</td>
</tr>
<tr>
<td>A1</td>
<td>Aggregated Core Array Stalled</td>
<td>The sum of all cycles on all cores spent stalled. (at least one thread loaded but the entire core is stalled for any reason)</td>
</tr>
<tr>
<td>A2</td>
<td>Vertex Shader Active Time</td>
<td>Total time in clocks the vertex shader spent active on all cores.</td>
</tr>
<tr>
<td>A3</td>
<td>Vertex Shader Stall Time</td>
<td>Total time in clocks the vertex shader spent stalled on all cores. This metric must be bucketed by stall type (“other” is ok – but must have buckets for things that are architecturally interesting)</td>
</tr>
<tr>
<td></td>
<td>(Event not supported in Gen 6 and so this counter should not be used for any performance analysis)</td>
<td></td>
</tr>
<tr>
<td>A4</td>
<td>Vertex Shader Stall Time – Core Stall</td>
<td>Total time in clocks the vertex shader spent stalled on all cores – and the entire core was stalled as well. This metric must be bucketed by stall type (“other” is ok – but must have buckets for things that are architecturally interesting)</td>
</tr>
<tr>
<td>A5</td>
<td># VS threads loaded</td>
<td>Number of VS threads loaded at any given time in the EUs.</td>
</tr>
<tr>
<td>A6</td>
<td>Vertex Shader ready but not running Time</td>
<td>Total time in clocks the vertex shader spent ready to run but not running on all cores.</td>
</tr>
<tr>
<td>A7</td>
<td>Geometry Shader/GPGPU Active Time</td>
<td>Total time in clocks the geometry shader or GPGPU spent active on all cores.</td>
</tr>
<tr>
<td>A8</td>
<td>Geometry Shader/GPGPU Stall Time</td>
<td>Total time in clocks the geometry shader or GPGPU spent stalled on all cores. This metric must be bucketed by stall type (“other” is ok – but must have buckets for things that are architecturally interesting)</td>
</tr>
<tr>
<td></td>
<td>(Event not supported in Gen 6 and so this counter should not be used for any performance analysis)</td>
<td></td>
</tr>
<tr>
<td>A9</td>
<td>Geometry Shader/GPGPU Stall Time – Core Stall</td>
<td>Total time in clocks the geometry shader or GPGPU spent stalled on all cores – and the entire core was stalled as well. This metric must be bucketed by stall type (“other” is ok – but must have buckets for things that are architecturally interesting)</td>
</tr>
<tr>
<td>A10</td>
<td># GS/GPGPU threads loaded</td>
<td>Number of GS or GPGPU threads loaded at any given time in the EUs.</td>
</tr>
<tr>
<td>A11</td>
<td>Geometry Shader/GPGPU ready but not running Time</td>
<td>Total time in clocks the geometry shader or GPGPU spent ready to run but not running on all cores.</td>
</tr>
<tr>
<td>A12</td>
<td>Pixel Shader Active Time</td>
<td>Total time in clocks the pixel shader spent active on all cores.</td>
</tr>
<tr>
<td>A13</td>
<td>Pixel Shader Stall Time</td>
<td>Total time in clocks the Pixel shader spent stalled on all cores. This metric must be bucketed by stall type (“other” is ok – but must have buckets for things that are architecturally interesting)</td>
</tr>
<tr>
<td></td>
<td>(Event not supported in Gen 6 and so this counter should not be used for any performance analysis)</td>
<td></td>
</tr>
<tr>
<td>A14</td>
<td>Pixel Shader Stall Time – Core Stall</td>
<td>Total time in clocks the pixel shader spent stalled on all cores – and the entire core was stalled as well. This metric must be bucketed by stall type (“other” is ok – but must have buckets for things that are architecturally interesting)</td>
</tr>
<tr>
<td>Cntr #</td>
<td>Event</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>A15</td>
<td># PS threads loaded</td>
<td>Number of PS threads loaded at any given time in the EUs.</td>
</tr>
<tr>
<td>A16</td>
<td>Pixel Shader ready but not running Time</td>
<td>Total time in clocks the Pixel shader spent ready to run but not running on all cores.</td>
</tr>
<tr>
<td>A17</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>A18</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>A19</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>A20</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>A21</td>
<td>Pixel Kill Count</td>
<td>Number of pixels/samples killed in the pixel shader. (How about chroma key?) [DevSNB]: Count reported is 2X or 4X the actual count for non-dual source blending or dual source blending respectively.</td>
</tr>
<tr>
<td>A22</td>
<td>Alpha Test Pixels Failed</td>
<td>Number of pixels/samples that fail alpha-test. Alpha to coverage may have some challenges in per-pixel invocation.</td>
</tr>
<tr>
<td>A23</td>
<td>Post PS Stencil Pixels Failed</td>
<td>Number of pixels/samples fail stencil test in the backend.</td>
</tr>
<tr>
<td>A24</td>
<td>Post PS Z buffer Pixels Failed</td>
<td>Number of pixels/samples fail Z test in the backend.</td>
</tr>
<tr>
<td>A25</td>
<td>Pixels/samples Written in the frame buffer</td>
<td>MRT case will report multiple of those.</td>
</tr>
<tr>
<td>A26</td>
<td>GPU Busy</td>
<td>CSunit indicating that ring is idle.</td>
</tr>
<tr>
<td>A27</td>
<td>CL active and not stalled</td>
<td>Clipper Fixed Function is active but not stalled</td>
</tr>
<tr>
<td>A28</td>
<td>SF active and stalled</td>
<td>SF Fixed Function is active but not stalled</td>
</tr>
</tbody>
</table>

### Aggregating Counters

<table>
<thead>
<tr>
<th>Cntr #</th>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A0</td>
<td>Aggregated Core Array Active</td>
<td>The sum of all cycles on all cores spent actively executing instructions. This does not count the time taken to service Send instructions. This time is considered by shader active counters to give the result.</td>
</tr>
<tr>
<td>A1</td>
<td>Aggregated Core Array Stalled</td>
<td>The sum of all cycles on all cores where the EU is not idle and is not actively executing ISA instructions. Generally this means that all loaded threads on the EU are stalled on some data dependency, but this also includes the time during which the TS is loading the thread dispatch header into the EU prior to thread execution and no other thread is fully loaded.</td>
</tr>
<tr>
<td>A2</td>
<td>Vertex Shader Active Time</td>
<td>Total time in clocks the vertex shader spent active on all cores.</td>
</tr>
<tr>
<td>A3</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>A4</td>
<td>Vertex Shader Stall Time – Core Stall</td>
<td>Total time in clocks the vertex shader spent stalled on all cores – and the entire core was stalled as well.</td>
</tr>
<tr>
<td>A5</td>
<td># VS threads loaded</td>
<td>Number of VS threads loaded at any given time in the EUs.</td>
</tr>
<tr>
<td>A6</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>A7</td>
<td>Hull Shader Active Time</td>
<td>Total time in clocks the Hull shader spent active on all cores.</td>
</tr>
<tr>
<td>Cntr #</td>
<td>Event</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>A8</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>A9</td>
<td>Hull Shader Stall Time – Core Stall</td>
<td>Total time in clocks the Hull shader spent stalled on all cores – and the entire core was stalled as well.</td>
</tr>
<tr>
<td>A10</td>
<td># HS threads loaded</td>
<td>Number of HS threads loaded at any given time in the EUs.</td>
</tr>
<tr>
<td>A11</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>A12</td>
<td>Domain Shader Active Time</td>
<td>Total time in clocks the Domain shader spent active on all cores.</td>
</tr>
<tr>
<td>A13</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>A14</td>
<td>Domain Shader Stall Time – Core Stall</td>
<td>Total time in clocks the domain shader spent stalled on all cores – and the entire core was stalled as well.</td>
</tr>
<tr>
<td>A15</td>
<td># DS threads loaded</td>
<td>Number of DS threads loaded at any given time in the EUs.</td>
</tr>
<tr>
<td>A16</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>A17</td>
<td>Compute Shader Active Time</td>
<td>Total time in clocks the compute shader spent active on all cores.</td>
</tr>
<tr>
<td>A18</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>A19</td>
<td>Compute Shader Stall Time – Core Stall</td>
<td>Total time in clocks the compute shader spent stalled on all cores – and the entire core was stalled as well.</td>
</tr>
<tr>
<td>A20</td>
<td># CS threads loaded</td>
<td>Number of CS threads loaded at any given time in the EUs.</td>
</tr>
<tr>
<td>A21</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>A22</td>
<td>Geometry Shader Active Time</td>
<td>Total time in clocks the geometry shader spent active on all cores.</td>
</tr>
<tr>
<td>A23</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>A24</td>
<td>Geometry Shader Stall Time – Core Stall</td>
<td>Total time in clocks the geometry shader spent stalled on all cores – and the entire core was stalled as well.</td>
</tr>
<tr>
<td>A25</td>
<td># GS threads loaded</td>
<td>Number of GS threads loaded at any given time in the EUs.</td>
</tr>
<tr>
<td>A26</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>A27</td>
<td>Pixel Shader Active Time</td>
<td>Total time in clocks the pixel shader spent active on all cores.</td>
</tr>
<tr>
<td>A28</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>A29</td>
<td>Pixel Shader Stall Time – Core Stall</td>
<td>Total time in clocks the pixel shader spent stalled on all cores – and the entire core was stalled as well.</td>
</tr>
<tr>
<td>A30</td>
<td># PS threads loaded</td>
<td>Number of PS threads loaded at any given time in the EUs.</td>
</tr>
<tr>
<td>Cntr #</td>
<td>Event</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>A31</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>A32</td>
<td>HIZ Fast Z Test Pixels Passing</td>
<td>[DevIVB] Count of pixels that pass the fast check (8x8). This counter under-counts slightly; a B-counter can be defined to correct the count. [DevVLV, DevVLVT] Count of pixels that pass the fast check (8x8). [DevHSW] Count of pixels that pass HiZ (8x8).</td>
</tr>
<tr>
<td>A34</td>
<td>[DevIVB] Slow Ztest Pixels Passing [DevHSW] Reserved</td>
<td>[DevIVB] Count of pixels passing the slow check (2x2) [DevHSW] Reserved</td>
</tr>
<tr>
<td>A35</td>
<td>Slow Z Test Pixels Failing</td>
<td>Count of pixels that fail the slow check (2x2)</td>
</tr>
<tr>
<td>A36</td>
<td>Pixel Kill Count</td>
<td>Number of pixels/samples killed in the pixel shader. Erratum: [DevIVB, DevVLV, DevVLVT]: Count reported is 2X the actual count for or dual source render target messages i.e. when PS has two output colors.</td>
</tr>
<tr>
<td>A37</td>
<td>Alpha Test Pixels Failed</td>
<td>Number of pixels/samples that fail alpha-test. Alpha to coverage may have some challenges in per-pixel invocation.</td>
</tr>
<tr>
<td>A38</td>
<td>Post PS Stencil Pixels Failed</td>
<td>Number of pixels/samples failing stencil test after the pixel shader has executed.</td>
</tr>
<tr>
<td>A39</td>
<td>Post PS Z buffer Pixels Failed</td>
<td>Number of pixels/samples fail Z test after the pixel shader has executed.</td>
</tr>
<tr>
<td>A40</td>
<td>3D/GPGPU Render Target Writes</td>
<td>MRT scenarios will cause this counter to increment multiple times.</td>
</tr>
<tr>
<td>A41</td>
<td>Render Engine Busy</td>
<td>Render engine is not idle. GPU Busy aggregate counter doesn’t increment under the following conditions: 1. Context Switch in Progress. 2. GPU stalled on executing MI_WAIT_FOR_EVENT. 3. GPU stalled on execution MI_SEMAPHORE_MBOX. 4. RCS idle but other parts of GPU active (e.g. only media engines active)</td>
</tr>
<tr>
<td>A42</td>
<td>VS bottleneck</td>
<td>VSUnit is stalling VF (upstream unit) and starving HS (downstream unit)</td>
</tr>
<tr>
<td>A43</td>
<td>GS bottleneck</td>
<td>GSUnit is stalling DS (upstream unit) and starving SOL(downstream unit)</td>
</tr>
<tr>
<td>A44</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
The MI_REPORT_PERF_COUNT command causes the GFX hardware to write out a snap-shot of performance counters to the address specified in this command along with constant ID field supplied and the time-stamp counter. This write is required to be treated as a cacheable write irrespective of GTT entry memory type. This command is specific to the render engine.

### Programming Notes

This command is to be used for performance debug mode and can be inserted after events of interest (frequently before and after a 3DPRIMITIVE command). SW is entirely responsible for managing the ID field and addresses used by such a series of commands.

GTT_SELECT must not be set to 1 (i.e. GGTT) when MI_REPORT_PERF_COUNT command is programmed in a non-privileged batch buffer. Refer to the “User Mode Privileged commands” Table in MI_BATCH_BUFFER_START command section for more details.

Each batch buffer is explicitly tagged as privileged or non-privileged.

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
<th>Project</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:29</td>
<td>Command Type</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default Value:</td>
<td>0h MI_COMMAND</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Format:</td>
<td>OpCode</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>28:23</td>
<td>MI Command Opcode</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Default Value:</td>
<td>28h MI_</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Format:</td>
<td>OpCode</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>22:6</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Format:</td>
<td>MBZ</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>5:0</td>
<td>DWord Length</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Format:</td>
<td>=n</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total Length - 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>31:6</td>
<td>Memory Address</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Format:</td>
<td>GraphicsAddress[31:6]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This field specifies 64B aligned GFX MEM address where the chap counter values are reported.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>5</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Format:</td>
<td>MBZ</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>3:1</td>
<td>Core Mode Enable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Project:</td>
<td>DevHSW+</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Format:</td>
<td>U1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This bit is set then the address will be offset by the Core ID:If Core ID 0, then there is no offsetIf Core ID 1, then the Memory is offset by the size of the data(64b).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>GTT Select</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Format:</td>
<td>MBZ</td>
<td></td>
</tr>
<tr>
<td>Format:</td>
<td>U1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>----</td>
<td></td>
<td></td>
</tr>
<tr>
<td>This field when set (i.e. bit = 1) selects the GGTT for address translation. When this bit is 0 (default value), HW should use PGTT for address translation.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2</th>
<th>Project: DevIVB+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Report ID</td>
<td></td>
</tr>
<tr>
<td>Project:</td>
<td>DevIVB+</td>
</tr>
<tr>
<td>Format:</td>
<td>U32</td>
</tr>
<tr>
<td>This field specifies the ID provided by SW for a given report command. It can be tracked to use different flavors of these reports based on where in command-stream they are inserted. This field is reported only when Counter Select Field is 0.</td>
<td></td>
</tr>
</tbody>
</table>
Performance Statistics Registers

**OACONTROL - Observation Architecture Control**

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevSNB+</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>02360h</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevSNB, DevIVB, DevVLV, DevVLVT, DevHSW]</td>
</tr>
</tbody>
</table>

This register controls global OA functionality, report format, interrupt steering and context filtering.

<table>
<thead>
<tr>
<th>Workaround</th>
<th>Project</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workaround: If software intends to reset the OA buffer to start a new one, after clearing the Timer Enable bit, software must check to see if the head pointer in OASTATUS2 is greater than the tail pointer in OASTATUS1. If so software must program the head pointer to a value less than the current head pointer value. This must be done before the buffer becomes active again.</td>
<td>DevSNB:GT2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:12</td>
<td><strong>Select Context ID</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Project:</strong> DevSNB, DevIVB, DevVLV, DevVLVT, DevHSW</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Specifies the context ID of the one context that affects the performance counters. All other contexts are ignored.</td>
</tr>
<tr>
<td>11:6</td>
<td></td>
<td><strong>Timer Period</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Project:</strong> DevIVB, DevVLV, DevVLVT, DevHSW</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Format:</strong> Select</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Specifies the period of the timer strobe as a function of the minimum TIME_STAMP resolution. The period is determined by selecting a specified bit from the TIME_STAMP register as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>StrobePeriod = MinimumTimeStampPeriod * 2^{(TimerPeriod + 1)}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The exponent is defined by this field.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note: The TIME_STAMP is not reset at start time so the phase of the strobe is not synchronized with the enable of the OA unit. This could result in approximately a full StrobePeriod elapsing prior to the first trigger. Usage for this mechanism should be time based periodic triggering, typically.</td>
</tr>
</tbody>
</table>
## OACONTROL - Observation Architecture Control

### Timer Enable

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Project</th>
</tr>
</thead>
<tbody>
<tr>
<td>0h</td>
<td>Disable</td>
<td>[Default]</td>
</tr>
<tr>
<td></td>
<td>Counter does not get written out on regular interval</td>
<td></td>
</tr>
<tr>
<td>1h</td>
<td>Enable</td>
<td>Counter gets written out on regular intervals, defined by the Timer Period</td>
</tr>
</tbody>
</table>

This field enables the timer logic to output a periodic strobe, as defined by the Timer Period. When disabled the timer output is not asserted.

### Counter Select

<table>
<thead>
<tr>
<th>Project</th>
<th>Pre-DevHSW</th>
</tr>
</thead>
</table>

This field selects which performance counter report format to use, please refer to Performance Counter Report Formats section for more details on the structure of the format.

### Reserved

<table>
<thead>
<tr>
<th>Project</th>
<th>DevHSW</th>
</tr>
</thead>
</table>

### Specific Context Enable

<table>
<thead>
<tr>
<th>Format</th>
<th>MBZ</th>
</tr>
</thead>
</table>

Enables counters to work on a context specific workload. The context is given by bits 31:12. OA unit level clock gating must be ENABLED when using specific ContextID feature.

### Performance Counter Enable

<table>
<thead>
<tr>
<th>Project</th>
<th>All</th>
</tr>
</thead>
</table>

Global performance counter enable. If clear, no counting will occur. MI_REPORT_PERF_COUNT is undefined when clear.

**Programming Notes**

When this bit is set, in order to have coherent counts, RC6 power state and render trunk clock gating must be disabled. This can be achieved by programming MMIO registers as 0xA094=0x0 and 0xA090[31]=1.

**Workaround**

Workaround: EU clock gating must be disabled when this bit is set.
## OSTATUS1 - Observation Architecture Status Register 1

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevSNB,DevIVB,DevHSW,DevVLV</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>02364h</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>DevSNB,DevIVB,DevHSW,DevVLV</td>
</tr>
</tbody>
</table>

This register is used to program the OA unit.

### DWord 0: Tail Pointer

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:6</td>
<td>Virtual address of the internal trigger based buffer and it is updated for every 64B cacheline write to memory when reporting via internal trigger. This pointer will not be updated for MI_REPORT_PERF_COUNT command based writes.</td>
</tr>
</tbody>
</table>

#### Programming Notes

When OA is enabled, this address must be programmed by SW to the base address of the internal trigger base mechanism. SW must ensure that Tail pointer and the Head Pointer (in OASTATUS2) do not have different values while programming.

### DWord 5: Inter Trigger Report Buffer Size

<table>
<thead>
<tr>
<th>Value</th>
<th>Name</th>
<th>Description</th>
<th>Project</th>
</tr>
</thead>
<tbody>
<tr>
<td>0h</td>
<td>All context considered [Default]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0b</td>
<td>128KB</td>
<td>DevHSW</td>
<td></td>
</tr>
<tr>
<td>1b</td>
<td>256KB</td>
<td>DevHSW</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>512KB</td>
<td>DevHSW</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1MB</td>
<td>DevHSW</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2MB</td>
<td>DevHSW</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4MB</td>
<td>DevHSW</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>8MB</td>
<td>DevHSW</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>16MB</td>
<td>DevHSW</td>
<td></td>
</tr>
<tr>
<td>0b</td>
<td>16KB</td>
<td>DevSNB, DevIVB</td>
<td></td>
</tr>
<tr>
<td>1b</td>
<td>32KB</td>
<td>DevSNB, DevIVB</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>48KB</td>
<td>DevSNB, DevIVB</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>64KB</td>
<td>DevSNB, DevIVB</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>80KB</td>
<td>DevSNB, DevIVB</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>96KB</td>
<td>DevSNB, DevIVB</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>112KB</td>
<td>DevSNB, DevIVB</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>128KB</td>
<td>DevSNB, DevIVB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Counter OverFlow Error</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Format:</strong></td>
<td><strong>Select</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td><strong>Project</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>This bit is set if any of the counters overflows. This bit can be reset by SW in B0.</td>
<td>DevSNB, DevIVB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Counter Overflow generation for B counters is incorrect. Counter overflow generation is getting generated on counter bit[31] transitioning form 0-&gt;1 OR 1-&gt;0. Counter overflow should be generated only when counter bit[31] transitions from 1-&gt;0, i.e when the counter wraps around max value. SW Should consider counter overflow as valid only when counter bit[31] is '0'. A and C counter overflow generation is correct.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Programming Notes</strong></td>
<td><strong>Project</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>This bit must be cleared after the ring is enabled and before OA is enabled.</td>
<td>DevSNB</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Buffer Overflow</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Default Value:</strong></td>
<td><strong>0h</strong></td>
</tr>
<tr>
<td>This bit is set when the Tail-pointer - Head pointer &gt; max internal trigger buffer size</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Report Lost Error</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Format:</strong></td>
<td><strong>Enable</strong></td>
</tr>
<tr>
<td>This bit is set if the Report Logic is requested to write out the counter values before the previous report request was completed. The report request is ignored and the counter continue to count. This bit can be reset by SW in B0.</td>
<td></td>
</tr>
<tr>
<td><strong>Programming Notes</strong></td>
<td><strong>Project</strong></td>
</tr>
<tr>
<td>Report Lost Error status is not functional and must not be looked at for any purposes.</td>
<td>VLV,VLVT</td>
</tr>
</tbody>
</table>
**OASTATUS2 - Observation Architecture Status Register 2**

Register Space: MMIO: 0/2/0  
Project: DevSNB, DevIVB, DevHSW  
Default Value: 0x00000000 [IVB, VLV, VLVT, SNB]  
0x00000001 [HSW]  
Access: R/W  
Size (in bits): 32  
Address: 02368h

This register is used to program the OA unit.

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
</table>
| 0     | 31:6| **Head Pointer**  
Virtual address of the internal trigger based buffer that is updated by software after consuming from the report buffer. This pointer must be updated by SW for internal trigger base buffer only. |
| 5     | Reserved | Format: MBZ |
| 4     | Tail Pointer Wrap Mask | Project: DevHSW |
| 3     | Tail Pointer Wrap Flag | Project: DevHSW  
Format: U1 |
| 2     | Head Pointer Wrap Mask | Project: DevHSW |
| 4:1   | Reserved | Project: Pre-DevHSW  
Format: MBZ |
| 1     | Head Pointer Wrap Flag | Project: DevHSW  
Format: U1 |

**Programming Notes**

- This bit should be set in order to program Tail Pointer Wrap Flag. This bit is for HW internal use. SW should always set this bit to 0.
- This bit is for HW internal use to context save/restore Tail Pointer Wrap Flag. SW should not program this bit. This bit gets programmed only when Tail Pointer Wrap Mask bit is set.
- This bit should be set in order to program Head Pointer Wrap Flag. This bit is for HW internal use. SW should always set this bit to 0.
### OASTATUS2 - Observation Architecture Status Register 2

<table>
<thead>
<tr>
<th>0</th>
<th>Memory select PPGTT/GGTT access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW</td>
</tr>
<tr>
<td>Access:</td>
<td>RO</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>PPGTT</td>
</tr>
<tr>
<td>1</td>
<td>GGTT [Default]</td>
</tr>
</tbody>
</table>

OABUFFER must always reside in GGTT memory. This bit must be set to ‘1’.

### OABUFFER - Observation Architecture Buffer

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevSNB+</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000 [SNB,IVB,VLV,VLVT,HSW]</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>023B0h</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevSNB]</td>
</tr>
<tr>
<td>Access:</td>
<td>WO</td>
</tr>
</tbody>
</table>

| Address: | 023B0h |
| Valid Projects: | [DevIVB, DevVLV, DevVLVT, DevHSW] |
| Access: | R/W |

This register is used to program the OA unit.

### Programming Notes

<table>
<thead>
<tr>
<th>Project</th>
</tr>
</thead>
<tbody>
<tr>
<td>DevSNB+</td>
</tr>
</tbody>
</table>

This MMIO must be set before the OASTATUS1 register and set after the OASTATUS2 register. This is to enable proper functionality of the overflow bit.

Report Buffer Offset Must be 512KB aligned.

Report Buffer Offset Must be 16MB aligned.

### DWord | Bit | Description
--- | --- | ---
0 | 31:6 | Report Buffer Offset
### OABUFFER - Observation Architecture Buffer

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Project</th>
<th>Format</th>
<th>Value</th>
<th>Name</th>
<th>Description</th>
<th>Project</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Format:</strong></td>
<td>GraphicsAddress[31:6]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>This field specifies 64B aligned GFX MEM address where the chap counter values are reported.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>5</strong> Reserved</td>
<td>Project: DevSNB+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Format: MBZ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>4</strong> OVERRUN STATUS</td>
<td>Default Value: 0h Enabled</td>
<td>DevHSW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project: DevHSW</td>
<td>Format: Enable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>This field indicates the status of overrun for debug purpose. This bit is read only and writing to this bit will have no effect. This bit will reflect the status of overrun irrespective of Overrun Mode enabled or disabled.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>4:3</strong> Reserved</td>
<td>Project: Pre-DevHSW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Format: MBZ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>3</strong> Disable Overrun Mode</td>
<td>Project: DevHSW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Format: Enable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>This field defines the mode of reporting for internal trigger/timer based reporting. When this bit is set, overrun does not lose reports but stops reporting. Based on the head and tail pointer, when HW detects room for the report, it would resume reporting to the buffer. This mode would not set the over-run bit in the register. When this mode bit is reset, buffer overrun can happen and lose the reports while setting the buffer over-run bit.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Value</strong></td>
<td>Name</td>
<td>Description</td>
<td>Project</td>
<td>Format</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0h</td>
<td>Disable [Default]</td>
<td>Counter gets written out on regular intervals, defined by the Timer Period</td>
<td>DevHSW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1h</td>
<td>Enable</td>
<td>Counter does not get written out on regular interval</td>
<td>DevHSW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2</strong> Reserved</td>
<td>Project: DevSNB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Format: MBZ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>1</strong> Counter Stop Resume Mechanism Enable</td>
<td>Project: DevIVB, DevHSW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>1:0</strong> Reserved</td>
<td>Project: DevSNB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Format: MBZ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>0</strong> Counter Stop-Resume Mechanism</td>
<td>Project: DevIVB, DevHSW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Format: Enable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Value</strong></td>
<td>Name</td>
<td>Description</td>
<td>Project</td>
<td>Format</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>[Default]</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>Resume counting for all counters</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### CEC0 - Customizable Event Creation 0-0

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevSNB+</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>02390h</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevSNB, DevIVB]</td>
</tr>
</tbody>
</table>

This register is used to define custom counter event 0, set this register to 0x00000003 and configure CEC0-1 properly in order to have counter B0 increment every GPU clock.

### CEC0 - Customizable Event Creation 0-1

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevSNB+</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>02394h</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevSNB, DevIVB, DevVLV, DevVLVT]</td>
</tr>
</tbody>
</table>

This register is used to define custom counter event 0, set it to 0x0000FFFF and configure CEC0-1 properly in order to have counter B0 increment every GPU clock.

### CEC1 - Customizable Event Creation 1-0

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevSNB+</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>02398h</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevSNB, DevIVB, DevVLV, DevVLVT]</td>
</tr>
</tbody>
</table>

This register is used to define custom counter event 1, set this register to 0x00000003 and configure CEC1-1 properly in order to have counter B1 increment every GPU clock.
### CEC1 - Customizable Event Creation 1-1

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevSNB+</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>0239Ch</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevSNB, DevVB]</td>
</tr>
<tr>
<td>Address:</td>
<td>0277Ch</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevHSW+]</td>
</tr>
</tbody>
</table>

This register is used to define custom counter event 1, set it to 0x0000FFFE and configure CEC1-0 properly in order to have counter B1 increment every GPU clock.

### CEC2 - Customizable Event Creation 2-0

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevSNB+</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>023A0h</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevSNB, DevVB]</td>
</tr>
<tr>
<td>Address:</td>
<td>02780h</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevHSW+]</td>
</tr>
</tbody>
</table>

This register is used to define custom counter event 2, set this register to 0x00000003 and configure CEC2-1 properly in order to have counter B2 increment every GPU clock.

### CEC2 - Customizable Event Creation 2-1

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevSNB+</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>023A4h</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevSNB, DevVB]</td>
</tr>
<tr>
<td>Address:</td>
<td>02784h</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevHSW+]</td>
</tr>
</tbody>
</table>

This register is used to define custom counter event 2, set it to 0x0000FFFE and configure CEC2-0 properly in order to have counter B2 increment every GPU clock.
## CEC3-0 - Customizable Event Creation 3-0

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevSNB+</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>023A8h</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevSNB, DevIVB]</td>
</tr>
</tbody>
</table>

This register is used to define custom counter event 3, set this register to 0x00000003 and configure CEC3-1 properly in order to have counter B3 increment every GPU clock.

## CEC3-1 - Customizable Event Creation 3-1

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevSNB+</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>023ACh</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevSNB, DevIVB]</td>
</tr>
</tbody>
</table>

This register is used to define custom counter event 3, set it to 0x00000003 and configure CEC3-1 properly in order to have counter B3 increment every GPU clock.

## CEC4-0 - Customizable Event Creation 4-0

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW+</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>02790h</td>
</tr>
</tbody>
</table>

This register is used to define custom counter event 4, set this register to 0x00000003 and configure CEC4-1 properly in order to have counter B4 increment every GPU clock.
<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW+</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>02794h</td>
</tr>
</tbody>
</table>

This register is used to define custom counter event 4, set it to 0x0000FFFE and configure CEC4-0 properly in order to have counter B4 increment every GPU clock.

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW+</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>02798h</td>
</tr>
</tbody>
</table>

This register is used to define custom counter event 5, set this register to 0x00000003 and configure CEC5-0 properly in order to have counter B0 increment every GPU clock.

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW+</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>0279Ch</td>
</tr>
</tbody>
</table>

This register is used to define custom counter event 5, set it to 0x0000FFFE and configure CEC5-1 properly in order to have counter B5 increment every GPU clock.

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW+</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>027A0h</td>
</tr>
</tbody>
</table>

This register is used to define custom counter event 6, set this register to 0x00000003 and configure CEC6-0 properly in order to have counter B6 increment every GPU clock.
<table>
<thead>
<tr>
<th>Register Space</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project</td>
<td>DevHSW+</td>
</tr>
<tr>
<td>Default Value</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits)</td>
<td>32</td>
</tr>
<tr>
<td>Address</td>
<td>027A4h</td>
</tr>
</tbody>
</table>

This register is used to define custom counter event 6, set it to 0x0000FFFFE and configure CEC6-0 properly in order to have counter B6 increment every GPU clock.

<table>
<thead>
<tr>
<th>Register Space</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project</td>
<td>DevHSW+</td>
</tr>
<tr>
<td>Default Value</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits)</td>
<td>32</td>
</tr>
<tr>
<td>Address</td>
<td>027A8h</td>
</tr>
</tbody>
</table>

This register is used to define custom counter event 7, set this register to 0x00000003 and configure CEC7-0 properly in order to have counter B7 increment every GPU clock.

<table>
<thead>
<tr>
<th>Register Space</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project</td>
<td>DevHSW+</td>
</tr>
<tr>
<td>Default Value</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits)</td>
<td>32</td>
</tr>
<tr>
<td>Address</td>
<td>027ACH</td>
</tr>
</tbody>
</table>

This register is used to define custom counter event 7, set it to 0x0000FFFFE and configure CEC7-1 properly in order to have counter B7 increment every GPU clock.
The following Performance Statistics registers must be part of the power context:

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>02800h</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevHSW]</td>
</tr>
</tbody>
</table>

This register reflects the count value of the OA Performance counter A0. DefaultValue="00000000h"

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td>Considerations</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no &quot;latch and hold&quot; mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>02804h</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevHSW]</td>
</tr>
</tbody>
</table>

This register reflects the count value of the OA Performance counter A1. DefaultValue="00000000h"

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td>Considerations</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no &quot;latch and hold&quot; mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>02808h</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevHSW]</td>
</tr>
</tbody>
</table>

This register reflects the count value of the OA Performance counter A2. DefaultValue="00000000h"
### OAPERF_A2 - Aggregate Perf Counter A2

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td><strong>Considerations</strong>&lt;br&gt;This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no &quot;latch and hold&quot; mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.</td>
</tr>
</tbody>
</table>

### OAPERF_A3 - Aggregate Perf Counter A3

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>0280Ch</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevHSW]</td>
</tr>
</tbody>
</table>

This register reflects the count value of the OA Performance counter A3. DefaultValue="00000000h"

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td><strong>Considerations</strong>&lt;br&gt;This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no &quot;latch and hold&quot; mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.</td>
</tr>
</tbody>
</table>

### OAPERF_A4 - Aggregate Perf Counter A4

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>HSW</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>02810h</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[HSW]</td>
</tr>
</tbody>
</table>

This register reflects the count value of the OA Performance counter A4. DefaultValue="00000000h"

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td><strong>Considerations</strong>&lt;br&gt;This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no &quot;latch and hold&quot; mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.</td>
</tr>
</tbody>
</table>
### OAPERF_A5 - Aggregate Perf Counter A5

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>02814h</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevHSW]</td>
</tr>
</tbody>
</table>

This register reflects the count value of the OA Performance counter A5. DefaultValue="00000000h"

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
</table>
| 0     | 31:0 | **Considerations**

This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

### OAPERF_A6 - Aggregate Perf Counter A6

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>HSW</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>02818h</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[HSW]</td>
</tr>
</tbody>
</table>

This register reflects the count value of the OA Performance counter A6. DefaultValue="00000000h"

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
</table>
| 0     | 31:0 | **Considerations**

This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.
### OAPERF_A7 - Aggregate Perf Counter A7

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Register Space</td>
<td>MMIO: 0/2/0</td>
</tr>
<tr>
<td>Project</td>
<td>DevHSW</td>
</tr>
<tr>
<td>Default Value</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits)</td>
<td>32</td>
</tr>
<tr>
<td>Address</td>
<td>0281Ch</td>
</tr>
<tr>
<td>Valid Projects</td>
<td>[DevHSW]</td>
</tr>
</tbody>
</table>

This register reflects the count value of the OA Performance counter A7. DefaultValue="00000000h"

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td><strong>Considerations</strong> This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no &quot;latch and hold&quot; mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.</td>
</tr>
</tbody>
</table>

### OAPERF_A8 - Aggregate Perf Counter A8

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Register Space</td>
<td>MMIO: 0/2/0</td>
</tr>
<tr>
<td>Project</td>
<td>DevHSW</td>
</tr>
<tr>
<td>Default Value</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits)</td>
<td>32</td>
</tr>
<tr>
<td>Address</td>
<td>02820h</td>
</tr>
<tr>
<td>Valid Projects</td>
<td>[DevHSW]</td>
</tr>
</tbody>
</table>

This register reflects the count value of the OA Performance counter A8. DefaultValue="00000000h"

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td><strong>Considerations</strong> This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no &quot;latch and hold&quot; mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.</td>
</tr>
</tbody>
</table>

### OAPERF_A9 - Aggregate Perf Counter A9

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Register Space</td>
<td>MMIO: 0/2/0</td>
</tr>
<tr>
<td>Project</td>
<td>DevHSW</td>
</tr>
<tr>
<td>Default Value</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits)</td>
<td>32</td>
</tr>
<tr>
<td>Address</td>
<td>02824h</td>
</tr>
<tr>
<td>Valid Projects</td>
<td>[DevHSW]</td>
</tr>
</tbody>
</table>

This register reflects the count value of the OA Performance counter A9. DefaultValue="00000000h"

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### OAPERF_A9 - Aggregate Perf Counter A9

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>Considerations: This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no &quot;latch and hold&quot; mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.</td>
</tr>
</tbody>
</table>

### OAPERF_A10 - Aggregate Perf Counter A10

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>02828h</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevHSW]</td>
</tr>
</tbody>
</table>

This register reflects the count value of the OA Performance counter A10. DefaultValue="00000000h"

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td>Considerations: This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no &quot;latch and hold&quot; mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.</td>
</tr>
</tbody>
</table>

### OAPERF_A11 - Aggregate Perf Counter A11

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>0282Ch</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevHSW]</td>
</tr>
</tbody>
</table>

This register reflects the count value of the OA Performance counter A11. DefaultValue="00000000h"

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td>Considerations: This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no &quot;latch and hold&quot; mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.</td>
</tr>
</tbody>
</table>
### OAPERF_A12 - Aggregate Perf Counter A12

Register Space: MMIO: 0/2/0  
Project: DevHSW  
Default Value: 0x00000000  
Access: R/W  
Size (in bits): 32  
Address: 02830h  
Valid Projects: [DevHSW]  

This register reflects the count value of the OA Performance counter A12. Default Value = "00000000h"

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
</table>
| 0     | 31:0 | **Considerations**  
This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads. |

### OAPERF_A13 - Aggregate Perf Counter A13

Register Space: MMIO: 0/2/0  
Project: DevHSW  
Default Value: 0x00000000  
Access: R/W  
Size (in bits): 32  
Address: 02834h  
Valid Projects: [DevHSW]  

This register reflects the count value of the OA Performance counter A13. Default Value = "00000000h"

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
</table>
| 0     | 31:0 | **Considerations**  
This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads. |

### OAPERF_A14 - Aggregate Perf Counter A14

Register Space: MMIO: 0/2/0  
Project: DevHSW  
Default Value: 0x00000000  
Access: R/W  
Size (in bits): 32  
Address: 02838h  
Valid Projects: [DevHSW]  

This register reflects the count value of the OA Performance counter A14. Default Value = "00000000h"

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
</table>
| 0     | 31:0 | **Considerations**  
This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads. |
### OAPERF_A14 - Aggregate Perf Counter A14

This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

### OAPERF_A15 - Aggregate Perf Counter A15

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>0283Ch</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevHSW]</td>
</tr>
</tbody>
</table>

This register reflects the count value of the OA Performance counter A15. DefaultValue="00000000h"

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Considerations</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td>This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no &quot;latch and hold&quot; mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.</td>
</tr>
</tbody>
</table>

### OAPERF_A16 - Aggregate Perf Counter A16

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>02840h</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevHSW]</td>
</tr>
</tbody>
</table>

This register reflects the count value of the OA Performance counter A16. DefaultValue="00000000h"

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Considerations</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td>This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no &quot;latch and hold&quot; mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.</td>
</tr>
</tbody>
</table>
### OAPERF_A17 - Aggregate Perf Counter A17

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
</tbody>
</table>

**Address:** 02844h  
**Valid Projects:** [DevHSW]

This register reflects the count value of the OA Performance counter A17. **DefaultValue=“00000000h”**

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
</table>
| 0     | 31:0| **Considerations**  
This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

### OAPERF_A18 - Aggregate Perf Counter A18

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
</tbody>
</table>

**Address:** 02848h  
**Valid Projects:** [DevHSW]

This register reflects the count value of the OA Performance counter A9. **DefaultValue=“00000000h”**

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
</table>
| 0     | 31:0| **Considerations**  
This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.
### OAPERF_A19 - Aggregate Perf Counter A19

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>HSW</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>0284Ch</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[HSW]</td>
</tr>
</tbody>
</table>

This register reflects the count value of the OA Performance counter A19. DefaultValue="00000000h".

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td><strong>Considerations</strong></td>
</tr>
</tbody>
</table>

This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

### OAPERF_A20 - Aggregate Perf Counter A20

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>HSW</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>02850h</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[HSW]</td>
</tr>
</tbody>
</table>

This register reflects the count value of the OA Performance counter A20. DefaultValue="00000000h".

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td><strong>Considerations</strong></td>
</tr>
</tbody>
</table>

This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.
### OAPERF_A21 - Aggregate Perf Counter A21

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>02854h</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevHSW]</td>
</tr>
</tbody>
</table>

This register reflects the count value of the OA Performance counter A21. DefaultValue="00000000h"

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
</table>
| 0     | 31:0| **Considerations**  
This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.|

### OAPERF_A22 - Aggregate Perf Counter A22

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>02858h</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevHSW]</td>
</tr>
</tbody>
</table>

This register reflects the count value of the OA Performance counter A22. DefaultValue="00000000h"

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
</table>
| 0     | 31:0| **Considerations**  
This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.|
### OAPERF_A23 - Aggregate Perf Counter A23

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>0285Ch</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevHSW]</td>
</tr>
</tbody>
</table>

This register reflects the count value of the OA Performance counter A23. DefaultValue="00000000h"

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td><strong>Considerations</strong></td>
</tr>
</tbody>
</table>

This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

### OAPERF_A24 - Aggregate Perf Counter A24

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>02860h</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevHSW]</td>
</tr>
</tbody>
</table>

This register reflects the count value of the OA Performance counter A24. DefaultValue="00000000h"

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td><strong>Considerations</strong></td>
</tr>
</tbody>
</table>

This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

### OAPERF_A25 - Aggregate Perf Counter A25

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>02864h</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevHSW]</td>
</tr>
</tbody>
</table>

This register reflects the count value of the OA Performance counter A25. DefaultValue="00000000h"

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td><strong>Considerations</strong></td>
</tr>
</tbody>
</table>

### OAPERF_A25 - Aggregate Perf Counter A25

This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

### OAPERF_A26 - Aggregate Perf Counter A26

<table>
<thead>
<tr>
<th>Register Space</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project</td>
<td>DevHSW</td>
</tr>
<tr>
<td>Default Value</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits)</td>
<td>32</td>
</tr>
<tr>
<td>Address</td>
<td>02868h</td>
</tr>
<tr>
<td>Valid Projects</td>
<td>[DevHSW]</td>
</tr>
</tbody>
</table>

This register reflects the count value of the OA Performance counter A26. DefaultValue="00000000h"

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td><strong>Considerations</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no &quot;latch and hold&quot; mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.</td>
</tr>
</tbody>
</table>

### OAPERF_A27 - Aggregate Perf Counter A27

<table>
<thead>
<tr>
<th>Register Space</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project</td>
<td>DevHSW</td>
</tr>
<tr>
<td>Default Value</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits)</td>
<td>32</td>
</tr>
<tr>
<td>Address</td>
<td>0286Ch</td>
</tr>
<tr>
<td>Valid Projects</td>
<td>[DevHSW]</td>
</tr>
</tbody>
</table>

This register reflects the count value of the OA Performance counter A27. DefaultValue="00000000h"

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td><strong>Considerations</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no &quot;latch and hold&quot; mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.</td>
</tr>
</tbody>
</table>

### OAPERF_A28 - Aggregate Perf Counter A28

<table>
<thead>
<tr>
<th>Register Space</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project</td>
<td>DevHSW</td>
</tr>
<tr>
<td>Default Value</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits)</td>
<td>32</td>
</tr>
<tr>
<td>Address</td>
<td>02870h</td>
</tr>
</tbody>
</table>
## OAPERF_A28 - Aggregate Perf Counter A28

<table>
<thead>
<tr>
<th>Valid Projects:</th>
<th>[DevHSW]</th>
</tr>
</thead>
<tbody>
<tr>
<td>This register reflects the count value of the OA Performance counter A28. DefaultValue=&quot;00000000h&quot;</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td><strong>Considerations</strong>&lt;br&gt;This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no &quot;latch and hold&quot; mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.</td>
</tr>
</tbody>
</table>

## OAPERF_A29 - Aggregate Perf Counter A29

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>02874h</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevHSW]</td>
</tr>
</tbody>
</table>

This register reflects the count value of the OA Performance counter A29. DefaultValue="00000000h"

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td><strong>Considerations</strong>&lt;br&gt;This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no &quot;latch and hold&quot; mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.</td>
</tr>
</tbody>
</table>

## OAPERF_A30 - Aggregate Perf Counter A30

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>02878h</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevHSW]</td>
</tr>
</tbody>
</table>

This register reflects the count value of the OA Performance counter A30. DefaultValue="00000000h"

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td><strong>Considerations</strong>&lt;br&gt;This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no &quot;latch and hold&quot; mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.</td>
</tr>
</tbody>
</table>
## OAPERF_A31 - Aggregate_Perf.Counter_A31

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>0278Ch</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevHSW]</td>
</tr>
</tbody>
</table>

This register reflects the count value of the OA Performance counter A31.

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td><strong>Considerations</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Format: U32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no &quot;latch and hold&quot; mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.</td>
</tr>
</tbody>
</table>

## OAPERF_A32 - Aggregate_Perf.Counter_A32

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>02880h</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevHSW]</td>
</tr>
</tbody>
</table>

This register reflects the count value of the OA Performance counter A32.

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td><strong>Considerations</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Format: U32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no &quot;latch and hold&quot; mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.</td>
</tr>
</tbody>
</table>

## OAPERF_A33 - Aggregate_Perf.Counter_A33

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW+</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>02884h</td>
</tr>
</tbody>
</table>
### OAPERF_A33 - Aggregate_Perf_Counter_A33

This register reflects the count value of the OA Performance counter A33

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td>Considerations</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Format:</th>
<th>U32</th>
</tr>
</thead>
<tbody>
<tr>
<td>This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no &quot;latch and hold&quot; mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.</td>
<td></td>
</tr>
</tbody>
</table>

### OAPERF_A34 - Aggregate_Perf_Counter_A34

This register reflects the count value of the OA Performance counter A34

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW+</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x000000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>02888h</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevHSW]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td>Considerations</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Format:</th>
<th>U32</th>
</tr>
</thead>
<tbody>
<tr>
<td>This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no &quot;latch and hold&quot; mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.</td>
<td></td>
</tr>
</tbody>
</table>

### OAPERF_A35 - Aggregate_Perf_Counter_A35

This register reflects the count value of the OA Performance counter A35

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW+</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x000000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>02888Ch</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevHSW]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td>Considerations</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Format:</th>
<th>U32</th>
</tr>
</thead>
<tbody>
<tr>
<td>This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no &quot;latch and hold&quot; mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.</td>
<td></td>
</tr>
</tbody>
</table>
### OAPERF_A36 - Aggregate_Perf_Counter_A36

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>02890h</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevHSW]</td>
</tr>
</tbody>
</table>

This register reflects the count value of the OA Performance counter A36

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td>Considerations</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Format:</th>
<th>U32</th>
</tr>
</thead>
</table>

This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

### OAPERF_A37 - Aggregate_Perf_Counter_A37

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>02894h</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevHSW]</td>
</tr>
</tbody>
</table>

This register reflects the count value of the OA Performance counter A37

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td>Considerations</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Format:</th>
<th>U32</th>
</tr>
</thead>
</table>

This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.
### OAPERF_A38 - Aggregate_Perf.Counter_A38

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>02898h</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevHSW]</td>
</tr>
</tbody>
</table>

This register reflects the count value of the OA Performance counter A38

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td><strong>Considerations</strong></td>
</tr>
</tbody>
</table>

| Format: | U32 |

This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

### OAPERF_A39 - Aggregate_Perf.Counter_A39

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>0289Ch</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevHSW]</td>
</tr>
</tbody>
</table>

This register reflects the count value of the OA Performance counter A39

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td><strong>Considerations</strong></td>
</tr>
</tbody>
</table>

| Format: | U32 |

This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.
**OAPERF_A40 - Aggregate_Perf_Counter_A40**

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>028A0h</td>
</tr>
<tr>
<td>Valid Projects :</td>
<td>[DevHSW]</td>
</tr>
</tbody>
</table>

This register reflects the count value of the OA Performance counter A40

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td><strong>Considerations</strong></td>
</tr>
</tbody>
</table>

```
Format: U32
This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.
```

**OAPERF_A41 - Aggregate_Perf_Counter_A41**

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>028A4h</td>
</tr>
<tr>
<td>Valid Projects :</td>
<td>[DevHSW]</td>
</tr>
</tbody>
</table>

This register reflects the count value of the OA Performance counter A41

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td><strong>Considerations</strong></td>
</tr>
</tbody>
</table>

```
Format: U32
This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.
```
### OAPERF_A42 - Aggregate_Perf.Counter_A42

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>028A8h</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevHSW]</td>
</tr>
</tbody>
</table>

This register reflects the count value of the OA Performance counter A42

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td><strong>Considerations</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Format:</th>
<th>U32</th>
</tr>
</thead>
<tbody>
<tr>
<td>This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no &quot;latch and hold&quot; mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.</td>
<td></td>
</tr>
</tbody>
</table>

### OAPERF_A43 - Aggregate.Perf.Counter_A43

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>028ACh</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevHSW]</td>
</tr>
</tbody>
</table>

This register reflects the count value of the OA Performance counter A43

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td><strong>Considerations</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Format:</th>
<th>U32</th>
</tr>
</thead>
<tbody>
<tr>
<td>This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no &quot;latch and hold&quot; mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.</td>
<td></td>
</tr>
</tbody>
</table>
**OAPERF_A44 - Aggregate_Perf.Counter_A44**

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>028B0h</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevHSW]</td>
</tr>
</tbody>
</table>

This register reflects the count value of the OA Performance counter A44.

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td><strong>Considerations</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Format:</strong> U32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no &quot;latch and hold&quot; mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.</td>
</tr>
</tbody>
</table>
### OAPERF_B0 - Boolean_Counter_B0

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW+</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>028B4h</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevHSW]</td>
</tr>
</tbody>
</table>

This register enables the current live value of performance counter B0 to be read. Since what conditions cause B0 to increment are defined by the programming of CEC0-0/CEC0-1, please refer to the description of these registers and the values they hold to determine what event is reported via this register.

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td><strong>Considerations</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Format: U32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no &quot;latch and hold&quot; mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.</td>
</tr>
</tbody>
</table>

### OAPERF_B1 - Boolean_Counter_B1

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW+</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>028B8h</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevHSW]</td>
</tr>
</tbody>
</table>

This register enables the current live value of performance counter B1 to be read. Since what conditions cause B1 to increment are defined by the programming of CEC1-0/CEC1-1, please refer to the description of these registers and the values they hold to determine what event is reported via this register.

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td><strong>Considerations</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Format: U32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no &quot;latch and hold&quot; mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.</td>
</tr>
</tbody>
</table>
This register enables the current live value of performance counter B2 to be read. Since what conditions cause B2 to increment are defined by the programming of CEC0-0/CEC0-1, please refer to the description of these registers and the values they hold to determine what event is reported via this register.

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td><strong>Considerations</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Format:</th>
<th>U32</th>
</tr>
</thead>
</table>

This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

This register enables the current live value of performance counter B3 to be read. Since what conditions cause B3 to increment are defined by the programming of CEC0-0/CEC0-1, please refer to the description of these registers and the values they hold to determine what event is reported via this register.

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td><strong>Considerations</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Format:</th>
<th>U32</th>
</tr>
</thead>
</table>

This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.
OAPERF_B4 - Boolean_Counter_B4

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW+</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>028C4h</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevHSW]</td>
</tr>
</tbody>
</table>

This register enables the current live value of performance counter B4 to be read. Since what conditions cause B4 to increment are defined by the programming of CEC0-0/CEC0-1, please refer to the description of these registers and the values they hold to determine what event is reported via this register.

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td><strong>Considerations</strong></td>
</tr>
</tbody>
</table>

Format: U32

This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

OAPERF_B5 - Boolean_Counter_B5

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW+</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
<tr>
<td>Address:</td>
<td>028C8h</td>
</tr>
<tr>
<td>Valid Projects:</td>
<td>[DevHSW]</td>
</tr>
</tbody>
</table>

This register enables the current live value of performance counter B5 to be read. Since what conditions cause B5 to increment are defined by the programming of CEC0-0/CEC0-1, please refer to the description of these registers and the values they hold to determine what event is reported via this register.

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td><strong>Considerations</strong></td>
</tr>
</tbody>
</table>

Format: U32

This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.
**OAPERF_B6 - Boolean_Counter_B6**

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW+</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
</tbody>
</table>

**Address:** 028CCh  
**Valid Projects:** [DevHSW]

This register enables the current live value of performance counter B6 to be read. Since what conditions cause B6 to increment are defined by the programming of CEC0-0/CEC0-1, please refer to the description of these registers and the values they hold to determine what event is reported via this register.

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td><strong>Considerations</strong></td>
</tr>
</tbody>
</table>

*Format:* U32  

This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.

**OAPERF_B7 - Boolean_Counter_B7**

<table>
<thead>
<tr>
<th>Register Space:</th>
<th>MMIO: 0/2/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project:</td>
<td>DevHSW+</td>
</tr>
<tr>
<td>Default Value:</td>
<td>0x00000000</td>
</tr>
<tr>
<td>Access:</td>
<td>R/W</td>
</tr>
<tr>
<td>Size (in bits):</td>
<td>32</td>
</tr>
</tbody>
</table>

**Address:** 028D0h  
**Valid Projects:** [DevHSW]

This register enables the current live value of performance counter B7 to be read. Since what conditions cause B7 to increment are defined by the programming of CEC0-0/CEC0-1, please refer to the description of these registers and the values they hold to determine what event is reported via this register.

<table>
<thead>
<tr>
<th>DWord</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31:0</td>
<td><strong>Considerations</strong></td>
</tr>
</tbody>
</table>

*Format:* U32  

This 32-bit field returns bits 31:0 of the live performance counter value when read. Note that there is no "latch and hold" mechanism for performance counters when they are accessed through MMIO, so the value returned from this register may be different on back-to-back reads.