Arm® Cortex®-A65 Core

Copyright © 2018, 2019 Arm Limited or its affiliates. All rights reserved.

Release Information

<table>
<thead>
<tr>
<th>Issue</th>
<th>Date</th>
<th>Confidentiality</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000-02</td>
<td>31 January 2018</td>
<td>Confidential</td>
<td>First release for r0p0</td>
</tr>
<tr>
<td>0100-00</td>
<td>06 July 2018</td>
<td>Confidential</td>
<td>First release for r1p0</td>
</tr>
<tr>
<td>0101-00</td>
<td>21 February 2019</td>
<td>Non-Confidential</td>
<td>First release for r1p1</td>
</tr>
</tbody>
</table>

Non-Confidential Proprietary Notice

This document is protected by copyright and other related rights and the practice or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm. No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether implementations infringe any third party patents.

THIS DOCUMENT IS PROVIDED “AS IS”. ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, third party patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word “partner” in reference to Arm’s customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

If any of the provisions contained in these terms conflict with any of the provisions of any click through or signed written agreement covering this document with Arm, then the click through or signed written agreement prevails over and supersedes the conflicting provisions of these terms. This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. Please follow Arm’s trademark usage guidelines at http://www.arm.com/company/policies/trademarks.

Copyright © 2018, 2019 Arm Limited (or its affiliates). All rights reserved.

110 Fulbourn Road, Cambridge, England CB1 9NJ.
LES-PRE-20349
Confidentiality Status
This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

Product Status
The information in this document is Final, that is for a developed product.

Web Address
http://www.arm.com
Contents

Arm® Cortex®-A65 Core Technical Reference Manual

Preface
About this book ................................................................. 16
Feedback ........................................................................... 20

Part A  Functional description

Chapter A1  Introduction
A1.1 About the core ......................................................... A1-24
A1.2 Features ........................................................................ A1-25
A1.3 Implementation options ........................................... A1-26
A1.4 Supported standards and specifications ..................... A1-27
A1.5 Test features ............................................................... A1-28
A1.6 Design tasks ............................................................. A1-29
A1.7 Product revisions ....................................................... A1-30

Chapter A2  Technical overview
A2.1 Components ............................................................ A2-32
A2.2 Interfaces ................................................................. A2-36
A2.3 About system control ............................................... A2-37
A2.4 About the Generic Timer .......................................... A2-38

Chapter A3  Clocks, resets, and input synchronization
A3.1 About clocks, resets, and input synchronization .......... A3-40
A3.2 Asynchronous interface ........................................... A3-41
## Part B  Register descriptions

### Chapter B1  AArch64 system registers

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>B1.1</td>
<td>AArch64 registers</td>
<td>B1-116</td>
</tr>
<tr>
<td>B1.2</td>
<td>AArch64 architectural system register summary</td>
<td>B1-117</td>
</tr>
<tr>
<td>B1.3</td>
<td>AArch64 IMPLEMENTATION DEFINED register summary</td>
<td>B1-123</td>
</tr>
<tr>
<td>B1.4</td>
<td>AArch64 registers by functional group</td>
<td>B1-124</td>
</tr>
<tr>
<td>B1.5</td>
<td>ACTLR_EL1, Auxiliary Control Register, EL1</td>
<td>B1-130</td>
</tr>
<tr>
<td>B1.6</td>
<td>ACTLR_EL2, Auxiliary Control Register, EL2</td>
<td>B1-131</td>
</tr>
<tr>
<td>B1.7</td>
<td>ACTLR_EL3, Auxiliary Control Register, EL3</td>
<td>B1-133</td>
</tr>
<tr>
<td>B1.8</td>
<td>AFSR0_EL1, Auxiliary Fault Status Register 0, EL1</td>
<td>B1-135</td>
</tr>
<tr>
<td>B1.9</td>
<td>AFSR0_EL2, Auxiliary Fault Status Register 0, EL2</td>
<td>B1-136</td>
</tr>
<tr>
<td>B1.10</td>
<td>AFSR0_EL3, Auxiliary Fault Status Register 0, EL3</td>
<td>B1-137</td>
</tr>
<tr>
<td>B1.11</td>
<td>AFSR1_EL1, Auxiliary Fault Status Register 1, EL1</td>
<td>B1-138</td>
</tr>
<tr>
<td>B1.12</td>
<td>AFSR1_EL2, Auxiliary Fault Status Register 1, EL2</td>
<td>B1-139</td>
</tr>
<tr>
<td>B1.13</td>
<td>AFSR1_EL3, Auxiliary Fault Status Register 1, EL3</td>
<td>B1-140</td>
</tr>
<tr>
<td>B1.14</td>
<td>AIDR_EL1, Auxiliary ID Register, EL1</td>
<td>B1-141</td>
</tr>
<tr>
<td>B1.15</td>
<td>AMAIR_EL1, Auxiliary Memory Attribute Indirection Register, EL1</td>
<td>B1-142</td>
</tr>
<tr>
<td>B1.16</td>
<td>AMAIR_EL2, Auxiliary Memory Attribute Indirection Register, EL2</td>
<td>B1-143</td>
</tr>
<tr>
<td>B1.17</td>
<td>AMAIR_EL3, Auxiliary Memory Attribute Indirection Register, EL3</td>
<td>B1-144</td>
</tr>
<tr>
<td>B1.18</td>
<td>CCSIDR_EL1, Cache Size ID Register, EL1</td>
<td>B1-145</td>
</tr>
<tr>
<td>B1.19</td>
<td>CUIDR_EL1, Cache Level ID Register, EL1</td>
<td>B1-147</td>
</tr>
<tr>
<td>B1.20</td>
<td>CPACR_EL1, Architectural Feature Access Control Register, EL1</td>
<td>B1-149</td>
</tr>
<tr>
<td>B1.21</td>
<td>CPTR_EL2, Architectural Feature Trap Register, EL2</td>
<td>B1-150</td>
</tr>
<tr>
<td>B1.22</td>
<td>CPTR_EL3, Architectural Feature Trap Register, EL3</td>
<td>B1-153</td>
</tr>
<tr>
<td>B1.23</td>
<td>CPUACTLR_EL1, CPU Auxiliary Control Register, EL1</td>
<td>B1-154</td>
</tr>
<tr>
<td>B1.24</td>
<td>CPUCFR_EL1, CPU Configuration Register, EL1</td>
<td>B1-156</td>
</tr>
<tr>
<td>B1.25</td>
<td>CPUECTLR_EL1, CPU Extended Control Register, EL1</td>
<td>B1-158</td>
</tr>
<tr>
<td>B1.26</td>
<td>CPUPCR_EL3, CPU Private Control Register, EL3</td>
<td>B1-162</td>
</tr>
<tr>
<td>B1.27</td>
<td>CPUPMR_EL3, CPU Private Mask Register, EL3</td>
<td>B1-164</td>
</tr>
<tr>
<td>B1.28</td>
<td>CPUPOR_EL3, CPU Private Operation Register, EL3</td>
<td>B1-166</td>
</tr>
<tr>
<td>B1.29</td>
<td>CPUPselR_EL3, CPU Private Selection Register, EL3</td>
<td>B1-168</td>
</tr>
<tr>
<td>B1.30</td>
<td>CPUPWRCtLR_EL1, Power Control Register, EL1</td>
<td>B1-170</td>
</tr>
<tr>
<td>B1.31</td>
<td>CSSELR_EL1, Cache Size Selection Register, EL1</td>
<td>B1-173</td>
</tr>
<tr>
<td>B1.32</td>
<td>CTR_EL0, Cache Type Register, EL0</td>
<td>B1-174</td>
</tr>
<tr>
<td>B1.33</td>
<td>DCZID_EL0, Data Cache Zero ID Register, EL0</td>
<td>B1-176</td>
</tr>
<tr>
<td>B1.34</td>
<td>DISR_EL1, Deferred Interrupt Status Register, EL1</td>
<td>B1-177</td>
</tr>
<tr>
<td>B1.35</td>
<td>ERRIDR_EL1, Error ID Register, EL1</td>
<td>B1-179</td>
</tr>
<tr>
<td>B1.36</td>
<td>ERRselR_EL1, Error Record Select Register, EL1</td>
<td>B1-180</td>
</tr>
<tr>
<td>B1.37</td>
<td>ERXctlR_EL1, Selected Error Record Control Register, EL1</td>
<td>B1-181</td>
</tr>
<tr>
<td>B1.38</td>
<td>ERXFr_EL1, Selected Error Record Feature Register, EL1</td>
<td>B1-182</td>
</tr>
<tr>
<td>B1.39</td>
<td>ERXMISC0_EL1, Selected Error Record Miscellaneous Register 0, EL1</td>
<td>B1-183</td>
</tr>
<tr>
<td>B1.40</td>
<td>ERXPFGCDN_EL1, Selected Error Pseudo Fault Generation Count Down Register, EL1</td>
<td>B1-184</td>
</tr>
<tr>
<td>B1.41</td>
<td>ERXPFGCTL_EL1, Selected Error Pseudo Fault Generation Control Register, EL1</td>
<td>B1-186</td>
</tr>
<tr>
<td>B1.42</td>
<td>ERXPFGF_EL1, Selected Pseudo Fault Generation Feature Register, EL1</td>
<td>B1-188</td>
</tr>
<tr>
<td>B1.43</td>
<td>ERXstatus_EL1, Selected Error Record Primary Status Register, EL1</td>
<td>B1-189</td>
</tr>
<tr>
<td>B1.44</td>
<td>ESR_EL1, Exception Syndrome Register, EL1</td>
<td>B1-190</td>
</tr>
<tr>
<td>B1.45</td>
<td>ESR_EL2, Exception Syndrome Register, EL2 ......................................................... B1-191</td>
<td></td>
</tr>
<tr>
<td>B1.46</td>
<td>ESR_EL3, Exception Syndrome Register, EL3 .......................................................... B1-192</td>
<td></td>
</tr>
<tr>
<td>B1.47</td>
<td>HACR_EL2, Hyp Auxiliary Configuration Register, EL2 ........................................... B1-193</td>
<td></td>
</tr>
<tr>
<td>B1.48</td>
<td>HCR_EL2, Hypervisor Configuration Register, EL2 .................................................. B1-194</td>
<td></td>
</tr>
<tr>
<td>B1.49</td>
<td>ID_AA64AFR0_EL1, AArch64 Auxiliary Feature Register 0 .......................................... B1-196</td>
<td></td>
</tr>
<tr>
<td>B1.50</td>
<td>ID_AA64AFR1_EL1, AArch64 Auxiliary Feature Register 1 .......................................... B1-197</td>
<td></td>
</tr>
<tr>
<td>B1.51</td>
<td>ID_AA64DFR0_EL1, AArch64 Debug Feature Register 0, EL1 .................................... B1-198</td>
<td></td>
</tr>
<tr>
<td>B1.52</td>
<td>ID_AA64DFR1_EL1, AArch64 Debug Feature Register 1, EL1 ..................................... B1-200</td>
<td></td>
</tr>
<tr>
<td>B1.53</td>
<td>ID_AA64ISAR0_EL1, AArch64 Instruction Set Attribute Register 0, EL1 ..................... B1-201</td>
<td></td>
</tr>
<tr>
<td>B1.54</td>
<td>ID_AA64ISAR1_EL1, AArch64 Instruction Set Attribute Register 1, EL1 ..................... B1-203</td>
<td></td>
</tr>
<tr>
<td>B1.55</td>
<td>ID_AA64MMFR0_EL1, AArch64 Memory Model Feature Register 0, EL1 ....................... B1-204</td>
<td></td>
</tr>
<tr>
<td>B1.56</td>
<td>ID_AA64MMFR1_EL1, AArch64 Memory Model Feature Register 1, EL1 ....................... B1-206</td>
<td></td>
</tr>
<tr>
<td>B1.57</td>
<td>ID_AA64MMFR2_EL1, AArch64 Memory Model Feature Register 2, EL1 ....................... B1-208</td>
<td></td>
</tr>
<tr>
<td>B1.58</td>
<td>ID_AA64PFRO_EL1, AArch64 Processor Feature Register 0, EL1 ................................ B1-209</td>
<td></td>
</tr>
<tr>
<td>B1.59</td>
<td>ID_AA64PFRI_EL1, AArch64 Processor Feature Register 1, EL1 ................................ B1-211</td>
<td></td>
</tr>
<tr>
<td>B1.60</td>
<td>LORC_EL1, LORegion Control Register, EL1 ............................................................. B1-212</td>
<td></td>
</tr>
<tr>
<td>B1.61</td>
<td>LORID_EL1, LORegion ID Register, EL1 .................................................................... B1-213</td>
<td></td>
</tr>
<tr>
<td>B1.62</td>
<td>LORN_EL1, LORegion Number Register, EL1 ............................................................ B1-214</td>
<td></td>
</tr>
<tr>
<td>B1.63</td>
<td>MDCR_EL3, Monitor Debug Configuration Register, EL3 ........................................... B1-215</td>
<td></td>
</tr>
<tr>
<td>B1.64</td>
<td>MIDR_EL1, Main ID Register, EL1 ............................................................................ B1-217</td>
<td></td>
</tr>
<tr>
<td>B1.65</td>
<td>MPIDR_EL1, Multiprocessor Affinity Register, EL1 .................................................... B1-218</td>
<td></td>
</tr>
<tr>
<td>B1.66</td>
<td>PAR_EL1, Physical Address Register, EL1 ................................................................. B1-220</td>
<td></td>
</tr>
<tr>
<td>B1.67</td>
<td>REVIDR_EL1, Revision ID Register, EL1 ................................................................. B1-221</td>
<td></td>
</tr>
<tr>
<td>B1.68</td>
<td>RMR_EL3, Reset Management Register .................................................................. B1-222</td>
<td></td>
</tr>
<tr>
<td>B1.69</td>
<td>RVBAR_EL3, Reset Vector Base Address Register, EL3 ............................................ B1-223</td>
<td></td>
</tr>
<tr>
<td>B1.70</td>
<td>SCTLR_EL1, System Control Register, EL1 ............................................................... B1-224</td>
<td></td>
</tr>
<tr>
<td>B1.71</td>
<td>SCTLR_EL2, System Control Register, EL2 ............................................................... B1-226</td>
<td></td>
</tr>
<tr>
<td>B1.72</td>
<td>SCTLR_EL3, System Control Register, EL3 ............................................................... B1-227</td>
<td></td>
</tr>
<tr>
<td>B1.73</td>
<td>TCR_EL1, Translation Control Register, EL1 ............................................................. B1-229</td>
<td></td>
</tr>
<tr>
<td>B1.74</td>
<td>TCR_EL2, Translation Control Register, EL2 ............................................................. B1-230</td>
<td></td>
</tr>
<tr>
<td>B1.75</td>
<td>TCR_EL3, Translation Control Register, EL3 ............................................................. B1-231</td>
<td></td>
</tr>
<tr>
<td>B1.76</td>
<td>TTBR0_EL1, Translation Table Base Register 0, EL1 ................................................. B1-233</td>
<td></td>
</tr>
<tr>
<td>B1.77</td>
<td>TTBR0_EL2, Translation Table Base Register 0, EL2 ................................................. B1-234</td>
<td></td>
</tr>
<tr>
<td>B1.78</td>
<td>TTBR0_EL3, Translation Table Base Register 0, EL3 ................................................. B1-235</td>
<td></td>
</tr>
<tr>
<td>B1.79</td>
<td>TTBR1_EL1, Translation Table Base Register 1, EL1 ............................................... B1-236</td>
<td></td>
</tr>
<tr>
<td>B1.80</td>
<td>TTBR1_EL2, Translation Table Base Register 1, EL2 ............................................... B1-237</td>
<td></td>
</tr>
<tr>
<td>B1.81</td>
<td>VDISR_EL2, Virtual Deferred Interrupt Status Register, EL2 ................................ B1-238</td>
<td></td>
</tr>
<tr>
<td>B1.82</td>
<td>VSESR_EL2, Virtual SError Exception Syndrome Register ....................................... B1-239</td>
<td></td>
</tr>
<tr>
<td>B1.83</td>
<td>VTCR_EL2, Virtualization Translation Control Register, EL2 ................................... B1-240</td>
<td></td>
</tr>
<tr>
<td>B1.84</td>
<td>VTTBR_EL2, Virtualization Translation Table Base Register, EL2 ............................ B1-241</td>
<td></td>
</tr>
</tbody>
</table>

**Chapter B2**

**Error system registers**

| B2.1 | Error system register summary ........................................................................ B2-244 |
| B2.2 | ERR0CTRL, Error Record Control Register .......................................................... B2-245 |
| B2.3 | ERR0FR, Error Record Feature Register ................................................................ B2-247 |
| B2.4 | ERR0MISC0, Error Record Miscellaneous Register 0 ........................................... B2-249 |
| B2.5 | ERR0PGFCDN, Error Pseudo Fault Generation Count Down Register ................... B2-251 |
| B2.6 | ERR0PGFCTL, Error Pseudo Fault Generation Control Register ......................... B2-252 |
| B2.7 | ERR0PGGF, Error Pseudo Fault Generation Feature Register .............................. B2-254 |
| B2.8 | ERR0STATUS, Error Record Primary Status Register ............................................ B2-256 |
Chapter B3  
**GIC registers**

B3.1 CPU interface registers .......................................................... B3-261  
B3.2 AArch64 physical GIC CPU interface system register summary ....... B3-262  
B3.3 ICC_AP0R0_EL1, Interrupt Controller Active Priorities Group 0 Register 0, EL1 ...

B3.4 ICC_AP1R0_EL1, Interrupt Controller Active Priorities Group 1 Register 0 EL1 B3-264  
B3.5 ICC_BPR0_EL1, Interrupt Controller Binary Point Register 0, EL1 ........ B3-265  
B3.6 ICC_BPR1_EL1, Interrupt Controller Binary Point Register 1, EL1 ........ B3-266  
B3.7 ICC_CTLR_EL1, Interrupt Controller Control Register, EL1 ............ B3-267  
B3.8 ICC_CTLR_EL3, Interrupt Controller Control Register, EL3 ............ B3-268  
B3.9 ICC_SRE_EL1, Interrupt Controller System Register Enable Register, EL1 .... B3-270

B3.10 ICC_SRE_EL2, Interrupt Controller System Register Enable register, EL2 ... B3-272  
B3.11 ICC_SRE_EL3, Interrupt Controller System Register Enable register, EL3 ... B3-274  
B3.12 AArch64 virtual GIC CPU interface register summary ................. B3-276  
B3.13 ICC_AP0R0_EL1, Interrupt Controller Virtual Active Priorities Group 0 Register 0, EL1 ...

B3.14 ICC_AP1R0_EL1, Interrupt Controller Virtual Active Priorities Group 1 Register 0, EL1 ...

B3.15 ICC_BPR0_EL1, Interrupt Controller Virtual Binary Point Register 0, EL1 .... B3-279  
B3.16 ICC_BPR1_EL1, Interrupt Controller Virtual Binary Point Register 1, EL1 .... B3-280  
B3.17 ICC_CTLR_EL1, Interrupt Controller Virtual Control Register, EL1 ........ B3-281  
B3.18 AArch64 virtual interface control system register summary .......... B3-283  
B3.19 ICH_AP0R0_EL2, Interrupt Controller Hyp Active Priorities Group 0 Register 0, EL2 ...

B3.20 ICH_AP1R0_EL2, Interrupt Controller Hyp Active Priorities Group 1 Register 0, EL2 ...

B3.21 ICH_HCR_EL2, Interrupt Controller Hyp Control Register, EL2 .......... B3-286  
B3.22 ICH_VMCR_EL2, Interrupt Controller Virtual Machine Control Register, EL2 ... B3-289  
B3.23 ICH_VTR_EL2, Interrupt Controller VGIC Type Register, EL2 .......... B3-291

Chapter B4  
**Advanced SIMD and floating-point registers**

B4.1 AArch64 register summary ..................................................... B4-294  
B4.2 FPCR, Floating-point Control Register ....................................... B4-295  
B4.3 FPSR, Floating-point Status Register .......................................... B4-297

Part C  
**Debug descriptions**

Chapter C1  
**Debug**

C1.1 About debug methods .............................................................. C1-302  
C1.2 Debug functional description ................................................... C1-303  
C1.3 Debug register interfaces ....................................................... C1-305  
C1.4 Debug events ........................................................................ C1-307  
C1.5 External debug interface .......................................................... C1-308

Chapter C2  
**Performance Monitor Unit**

C2.1 About the PMU ................................................................. C2-311  
C2.2 PMU functional description ..................................................... C2-312  
C2.3 External register access permissions to the PMU registers ............. C2-313  
C2.4 PMU events ........................................................................ C2-317  
C2.5 PMU interrupts ..................................................................... C2-321
Chapter C3  Embedded Trace Macrocell
C3.1  About the ETM ................................................................. C3-332
C3.2  ETM trace unit generation options and resources ......................... C3-333
C3.3  ETM trace unit functional description ........................................ C3-335
C3.4  Resetting the ETM ............................................................. C3-336
C3.5  Programming and reading ETM trace unit registers ........................ C3-337
C3.6  ETM trace unit register interfaces .......................................... C3-338
C3.7  Interaction with the PMU and Debug ....................................... C3-339

Part D  Debug registers

Chapter D1  AArch64 debug registers
D1.1  AArch64 debug register summary ........................................ C1-344
D1.2  DBGBCRn_EL1, Debug Breakpoint Control Registers, EL1 ............. C1-346
D1.3  DBGCLAIMSET_EL1, Debug Claim Tag Set Register, EL1 ............... C1-349
D1.4  DBGWCRn_EL1, Debug Watchpoint Control Registers, EL1 .......... C1-350
D1.5  MDSCR_EL1, Monitor Debug System Control Register, EL1 ........... C1-352

Chapter D2  Memory-mapped debug registers
D2.1  Memory-mapped debug register summary .................................. C2-356
D2.2  EDCIDR0, External Debug Component Identification Register 0 ....... C2-360
D2.3  EDCIDR1, External Debug Component Identification Register 1 ....... C2-361
D2.4  EDCIDR2, External Debug Component Identification Register 2 ....... C2-362
D2.5  EDCIDR3, External Debug Component Identification Register 3 ....... C2-363
D2.6  EDDEVID, External Debug Device ID Register 0 .......................... C2-364
D2.7  EDDEVID1, External Debug Device ID Register 1 ......................... C2-365
D2.8  EDDFR, External Debug Feature Register ...................................... C2-366
D2.9  EDITCTRL, External Debug Integration Mode Control Register ........ C2-368
D2.10 EDPFBR, External Debug Processor Feature Register .................... C2-369
D2.11 EDPIDR0, External Debug Peripheral Identification Register 0 ........ C2-371
D2.12 EDPIDR1, External Debug Peripheral Identification Register 1 ........ C2-372
D2.13 EDPIDR2, External Debug Peripheral Identification Register 2 ........ C2-373
D2.14 EDPIDR3, External Debug Peripheral Identification Register 3 ........ C2-374
D2.15 EDPIDR4, External Debug Peripheral Identification Register 4 ........ C2-375
D2.16 EDPIDRn, External Debug Peripheral Identification Registers 5-7 ...... C2-376
D2.17 EDRCR, External Debug Reserve Control Register ....................... C2-377

Chapter D3  AArch64 PMU registers
D3.1  AArch64 PMU register summary ........................................... C3-380
D3.2  PMCEID0_EL0, Performance Monitors Common Event Identification Register 0, EL0 .. C3-382
D3.3  PMCEID1_EL0, Performance Monitors Common Event Identification Register 1, EL0 .. C3-386
D3.4  PMCR_EL0, Performance Monitors Control Register, EL0 ............... C3-389

Chapter D4  Memory-mapped PMU registers
D4.1  Memory-mapped PMU register summary ..................................... C4-392
D4.2  PMCFGR, Performance Monitors Configuration Register ................. C4-396
Chapter D5  PMU snapshot registers

D5.1 PMU snapshot register summary ................................................................. D5-408
D5.2 PMPCSSR, Snapshot Program Counter Sample Register .......................... D5-409
D5.3 PMCIDSSR, Snapshot CONTEXTIDR_EL1 Sample Register .................... D5-410
D5.4 PMCID2SSR, Snapshot CONTEXTIDR_EL2 Sample Register .................. D5-411
D5.5 PMSSSR, PMU Snapshot Status Register .................................................. D5-412
D5.6 PMOVSSR, PMU Overflow Status Snapshot Register ............................... D5-413
D5.7 PMCCNTSR, PMU Cycle Counter Snapshot Register ................................. D5-414
D5.8 PMEVCNTSRn, PMU Cycle Counter Snapshot Registers 0-5 ....................... D5-415
D5.9 PMSSCR, PMU Snapshot Capture Register ............................................... D5-416

Chapter D6  ETM registers

D6.1 ETM register summary ............................................................................. D6-419
D6.2 TRCACATRn, Address Comparator Access Type Registers 0-7 ..................... D6-423
D6.3 TRCACVNr, Address Comparator Value Registers 0-7 ............................... D6-425
D6.4 TRCAUTHSTATUS, Authentication Status Register ............................... D6-426
D6.5 TRCAUXCTLR, Auxiliary Control Register .............................................. D6-427
D6.6 TRCBBCTLR, Branch Broadcast Control Register .................................... D6-429
D6.7 TRCCRCTLR, Cycle Count Control Register ............................................ D6-430
D6.8 TRCICDCTLRO, Context ID Comparator Control Register 0 ..................... D6-431
D6.9 TRCICDVR0, Context ID Comparator Value Register 0 ............................ D6-432
D6.10 TRCICDR0, ETM Component Identification Register 0 .......................... D6-433
D6.11 TRCICDR1, ETM Component Identification Register 1 .......................... D6-434
D6.12 TRCICDR2, ETM Component Identification Register 2 .......................... D6-435
D6.13 TRCICDR3, ETM Component Identification Register 3 .......................... D6-436
D6.14 TRCCLAIMCLR, Claim Tag Clear Register ............................................ D6-437
D6.15 TRCCLAIMSET, Claim Tag Set Register ............................................... D6-438
D6.16 TRCCNTCTLR0, Counter Control Register 0 ......................................... D6-439
D6.17 TRCCNTCTLR1, Counter Control Register 1 ......................................... D6-441
D6.18 TRCCRNTRLVRn, Counter Reload Value Registers 0-1 .............................. D6-443
D6.19 TRCCNTVRn, Counter Value Registers 0-1 ............................................ D6-444
D6.20 TRCCONFGR, Trace Configuration Register .......................................... D6-445
D6.21 TRCDEVAFF0, Device Affinity Register 0 ............................................. D6-448
D6.22 TRCDEVAFF1, Device Affinity Register 1 ............................................. D6-449
D6.23 TRCDEVARCH, Device Architecture Register ....................................... D6-450
D6.24 TRCDEVID, Device ID Register ............................................................ D6-451
D6.25 TRCDEVTYPE, Device Type Register .................................................... D6-452
D6.26 TRCEVENTCTL0R, Event Control 0 Register ........................................ D6-453
D6.27 TRCEVENTCTL1R, Event Control 1 Register .......................................... D6-455
| D6.28  | TRCEXINTSELR, External Input Select Register .................................................. D6-456 |
| D6.29  | TRCIDR0, ID Register 0 .................................................................................. D6-457 |
| D6.30  | TRCIDR1, ID Register 1 .................................................................................. D6-459 |
| D6.31  | TRCIDR2, ID Register 2 .................................................................................. D6-460 |
| D6.32  | TRCIDR3, ID Register 3 .................................................................................. D6-462 |
| D6.33  | TRCIDR4, ID Register 4 .................................................................................. D6-464 |
| D6.34  | TRCIDR5, ID Register 5 .................................................................................. D6-466 |
| D6.35  | TRCIDR8, ID Register 8 .................................................................................. D6-468 |
| D6.36  | TRCIDR9, ID Register 9 .................................................................................. D6-469 |
| D6.37  | TRCIDR10, ID Register 10 .............................................................................. D6-470 |
| D6.38  | TRCIDR11, ID Register 11 .............................................................................. D6-471 |
| D6.39  | TRCIDR12, ID Register 12 .............................................................................. D6-472 |
| D6.40  | TRCIDR13, ID Register 13 .............................................................................. D6-473 |
| D6.41  | TRCIMSPEC0, IMPLEMENTATION SPECIFIC Register 0 .......................................... D6-474 |
| D6.42  | TRCITATBIDR, Integration ATB Identification Register ........................................ D6-475 |
| D6.43  | TRCITCTRL, Integration Mode Control Register .................................................. D6-476 |
| D6.44  | TRCITATBINR, Integration Instruction ATB In Register ........................................ D6-477 |
| D6.45  | TRCITATBOUTR, Integration Instruction ATB Out Register ................................... D6-478 |
| D6.46  | TRCITIDATAR, Integration Instruction ATB Data Register .................................... D6-479 |
| D6.47  | TRCLAR, Software Lock Access Register ................................................................ D6-480 |
| D6.48  | TRCLSR, Software Lock Status Register ................................................................ D6-481 |
| D6.49  | TRCCNTVRn, Counter Value Registers 0-1 .......................................................... D6-482 |
| D6.50  | TRCOSLAR, OS Lock Access Register .................................................................. D6-483 |
| D6.51  | TRCOSLSR, OS Lock Status Register ................................................................... D6-484 |
| D6.52  | TRCPDCR, Power Down Control Register ................................................................ D6-485 |
| D6.53  | TRCPDSR, Power Down Status Register ................................................................ D6-486 |
| D6.54  | TRCPIDR0, ETM Peripheral Identification Register 0 .......................................... D6-487 |
| D6.55  | TRCPIDR1, ETM Peripheral Identification Register 1 .......................................... D6-488 |
| D6.56  | TRCPIDR2, ETM Peripheral Identification Register 2 .......................................... D6-489 |
| D6.57  | TRCPIDR3, ETM Peripheral Identification Register 3 .......................................... D6-490 |
| D6.58  | TRCPIDR4, ETM Peripheral Identification Register 4 .......................................... D6-491 |
| D6.59  | TRCPIDRn, ETM Peripheral Identification Registers 5-7 ..................................... D6-492 |
| D6.60  | TRCPRGCCTL, Programming Control Register .................................................... D6-493 |
| D6.61  | TRCRSCTLn, Resource Selection Control Registers 2-16 ..................................... D6-494 |
| D6.62  | TRCSEQEVRn, Sequencer State Transition Control Registers 0-2 ......................... D6-495 |
| D6.63  | TRCSEQEVST, Sequencer Reset Control Register .................................................. D6-497 |
| D6.64  | TRCSEQST, Sequencer State Register .................................................................. D6-498 |
| D6.65  | TRCSSCCSR0, Single-Shot Comparator Control Register 0 .................................. D6-499 |
| D6.66  | TRCSSCCSR0, Single-Shot Comparator Status Register 0 .................................. D6-500 |
| D6.67  | TRCSTALLCTL, Stall Control Register .................................................................. D6-501 |
| D6.68  | TRCSTATR, Status Register ................................................................................ D6-502 |
| D6.69  | TRCSYNCCR, Synchronization Period Register .................................................... D6-503 |
| D6.70  | TRCTRACEIDR, Trace ID Register ...................................................................... D6-504 |
| D6.71  | TRCTSCTL, Global Timestamp Control Register .................................................. D6-505 |
| D6.72  | TRCVICTLR, ViewInst Main Control Register ..................................................... D6-506 |
| D6.73  | TRCVIECTL, ViewInst Include-Exclude Control Register .................................... D6-508 |
| D6.74  | TRCVISSCTL, ViewInst Start-Stop Control Register .......................................... D6-509 |
| D6.75  | TRCVMIDCCTL0, Virtual context identifier Comparator Control Register 0 ........ D6-510 |
| D6.76  | TRCVMIDCVR0, VMD Comparator Value Register 0 ........................................... D6-511 |
This preface introduces the *Arm® Cortex®-A65 Core Technical Reference Manual*.

It contains the following:

- *About this book on page 16.*
- *Feedback on page 20.*
About this book

This Technical Reference Manual is for the Cortex®-A65 core. It provides reference documentation and contains programming details for registers. It also describes the memory system, the caches, the interrupts, and the debug features.

Product revision status

The rmpn identifier indicates the revision status of the product described in this book, for example, r1p2, where:

- rm Identifies the major revision of the product, for example, r1.
- pn Identifies the minor revision or modification status of the product, for example, p2.

Intended audience

This manual is for system designers, system integrators, and programmers who are designing or programming a System-on-Chip (SoC) that uses an Arm core.

Using this book

This book is organized into the following chapters:

Part A Functional description

This part describes the main functionality of the Cortex-A65 core.

Chapter A1 Introduction

This chapter provides an overview of the Cortex-A65 core and its features.

Chapter A2 Technical overview

This chapter describes the structure of the Cortex-A65 core.

Chapter A3 Clocks, resets, and input synchronization

This chapter describes the clocks, resets, and input synchronization of the Cortex-A65 core.

Chapter A4 Power management

This chapter describes the power domains and the power modes in the Cortex-A65 core.

Chapter A5 Memory Management Unit

This chapter describes the Memory Management Unit (MMU) of the Cortex-A65 core.

Chapter A6 Level 1 memory system

This chapter describes the L1 instruction cache and data cache that make up the L1 memory system.

Chapter A7 Level 2 memory system

This chapter describes the L2 memory system.

Chapter A8 Reliability, Availability, and Serviceability (RAS)

This chapter describes the RAS features implemented in the Cortex-A65 core.

Chapter A9 Generic Interrupt Controller CPU interface

This chapter describes the Cortex-A65 core implementation of the Arm Generic Interrupt Controller (GIC) CPU interface.

Chapter A10 Advanced SIMD and floating-point support

This chapter describes the Advanced SIMD and floating-point features and registers in the Cortex-A65 core. The unit in charge of handling the Advanced SIMD and floating-point features is also referred to as data engine in this manual.

Part B Register descriptions

This part describes the system registers of the Cortex-A65 core.
Chapter B1 AArch64 system registers
This chapter describes the system registers in the AArch64 state.

Chapter B2 Error system registers
This chapter describes the error registers accessed by the AArch64 error registers.

Chapter B3 GIC registers
This chapter describes the GIC registers.

Chapter B4 Advanced SIMD and floating-point registers
This chapter describes the Advanced SIMD and floating-point registers.

Part C Debug descriptions
This part describes the debug functionality of the Cortex-A65 core.

Chapter C1 Debug
This chapter describes the debug features of the core.

Chapter C2 Performance Monitor Unit
This chapter describes the Performance Monitor Unit (PMU).

Chapter C3 Embedded Trace Macrocell
This chapter describes the Embedded Trace Macrocell (ETM) for the Cortex-A65 core.

Part D Debug registers
This part describes the debug registers of the Cortex-A65 core.

Chapter D1 AArch64 debug registers
This chapter describes the debug registers in the AArch64 Execution state and shows examples of how to use them.

Chapter D2 Memory-mapped debug registers
This chapter describes the memory-mapped debug registers and shows examples of how to use them.

Chapter D3 AArch64 PMU registers
This chapter describes the AArch64 PMU registers and shows examples of how to use them.

Chapter D4 Memory-mapped PMU registers
This chapter describes the memory-mapped PMU registers and shows examples of how to use them.

Chapter D5 PMU snapshot registers
PMU snapshot registers are an IMPLEMENTATION DEFINED extension to an Armv8-A compliant PMU to support an external core monitor that connects to a system profiler.

Chapter D6 ETM registers
This chapter describes the ETM registers.

Part E Appendices
This part describes the appendices of the Cortex-A65 core.

Appendix A Revisions
This appendix describes the technical changes between released issues of this book.

Glossary
The Arm® Glossary is a list of terms used in Arm documentation, together with definitions for those terms. The Arm Glossary does not contain terms that are industry standard unless the Arm meaning differs from the generally accepted meaning.

See the Arm® Glossary for more information.
**Typographic conventions**

*italic*  
Introduces special terminology, denotes cross-references, and citations.

*bold*  
Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.

*monospace*  
Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.

*monospace*  
Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.

*monospace italic*  
Denotes arguments to monospace text where the argument is to be replaced by a specific value.

*monospace bold*  
Denotes language keywords when used outside example code.

<and>  
Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example:  

```
MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2>
```

**Small capitals**  
Used in body text for a few terms that have specific technical meanings, that are defined in the *Arm® Glossary*. For example, *IMPLEMENTATION DEFINED*, *IMPLEMENTATION SPECIFIC*, *UNKNOWN*, and *UNPREDICTABLE*.

**Timing diagrams**  
The following figure explains the components used in timing diagrams. Variations, when they occur, have clear labels. You must not assume any timing information that is not explicit in the diagrams.

Shaded bus and signal areas are undefined, so the bus or signal can assume any value within the shaded area at that time. The actual level is unimportant and does not affect normal operation.

![Timing diagrams](image)

**Figure 1  Key to timing diagram conventions**

**Signals**  
The signal conventions are:
Signal level
The level of an asserted signal depends on whether the signal is active-HIGH or active-LOW. Asserted means:
• HIGH for active-HIGH signals.
• LOW for active-LOW signals.

Lowercase n
At the start or end of a signal name denotes an active-LOW signal.

Additional reading
This book contains information that is specific to this product. See the following documents for other relevant information.

Arm publications
• Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile (DDI 0487).
• Arm® DynamIQ™ Shared Unit Technical Reference Manual (100453).
• AMBA® AXI and ACE Protocol Specification AXI3, AXI4, AXI5, ACE and ACE5 (IHI 0022).
• Arm® AMBA® 5 CHI Architecture Specification (IHI 0050).
• Arm® CoreSight™ Architecture Specification v3.0 (IHI 0029).
• Arm® Debug Interface Architecture Specification, ADlv5.0 to ADlv5.2 (IHI 0031).
• AMBA® 4 ATB Protocol Specification (IHI 0032).
• Arm® Generic Interrupt Controller Architecture Specification (IHI 0069).
• Arm® Embedded Trace Macrocell Architecture Specification ETMv4 (IHI 0064).
• AMBA® Low Power Interface Specification Arm® Q-Channel and P-Channel Interfaces (IHI 0068).
• Arm® Reliability, Availability, and Serviceability (RAS) Specification, Armv8, for the Armv8-A architecture profile (DDI 0587A).

The following confidential books are only available to licensees:
• Arm® Cortex®-A65 Core Configuration and Sign-off Guide (100440).
• Arm® Cortex®-A65 Core Integration Manual (100251).
• Arm® DynamIQ™ Shared Unit Configuration and Sign-off Guide (100454).
• Arm® DynamIQ™ Shared Unit Integration Manual (100455).

Other publications

——— Note ———-
Arm floating-point terminology is largely based on the earlier ANSI/IEEE Std 754-1985 issue of the standard. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile for more information.
Feedback

Feedback on this product

If you have any comments or suggestions about this product, contact your supplier and give:
- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

Feedback on content

If you have comments on content then send an e-mail to errata@arm.com. Give:
- The number 100439_0101_00_en.
- If applicable, the page number(s) to which your comments refer.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.

Note

Arm tests the PDF only in Adobe Acrobat and Acrobat Reader, and cannot guarantee the quality of the represented document when used with any other PDF reader.
Part A
Functional description
Chapter A1
Introduction

This chapter provides an overview of the Cortex-A65 core and its features.

It contains the following sections:

• **A1.1 About the core** on page A1-24.
• **A1.2 Features** on page A1-25.
• **A1.4 Supported standards and specifications** on page A1-27.
• **A1.5 Test features** on page A1-28.
• **A1.6 Design tasks** on page A1-29.
• **A1.7 Product revisions** on page A1-30.
A1.1 About the core

The Cortex-A65 core is a power-efficient, mid-range, throughput computing oriented, Simultaneously MultiThreaded (SMT) core that implements the AArch64 execution state of the Armv8-A architecture.

The Cortex-A65 core supports:
- The Armv8.2-A extension.
- The RAS extension.
- The Load acquire (LDAPR) instructions introduced in the Armv8.3-A extension.
- The Dot Product support instructions introduced in the Armv8.4-A extension.
- The PSTATE SSBS (Speculative Store Bypass Safe) bit that supports software mitigation for Spectre Variant 4 introduced in the Armv8.5-A extension.

The Cortex-A65 core does not implement the AArch32 execution state of the Armv8-A architecture.

As an SMT core, the Cortex-A65 core supports two execution threads on each core. Each thread is a separate architectural Processing Element (PE), and so has a complete copy of the architectural state.

The simultaneous nature of the multithreading means that the Cortex-A65 core is able to issue and execute instructions from both threads simultaneously in the same cycle. Software views a thread on the Cortex-A65 core in the same way as it would view a core in a traditional, single-threaded, multi-core processor. Thus, existing software for both single core and multi-core can run unmodified on the Cortex-A65 core.

In an Arm DynamIQ Shared Unit (DSU) cluster, each Cortex-A65 core has a separate Level 1 (L1) instruction cache, a separate L1 data cache, and an optional private per-core Level 2 (L2) cache. For more information, see the Arm® DynamIQ™ Shared Unit Technical Reference Manual.

The following figure shows an example of a configuration with four Cortex-A65 cores.

![Figure A1-1 Example Cortex-A65 core configuration](image-url)
A1.2 Features

The Cortex-A65 core includes the following features:

Core features

- Full implementation of the Armv8.2-A A64 instruction set.
- AArch64 execution state at all Exception levels (EL0 to EL3).
- Superscalar, variable-length, out-of-order pipeline.
- Simultaneous multithreading, with two execution threads on each core.
- Memory Management Unit (MMU).
- Support for Arm TrustZone® technology.
- 44-bit physical address (PA). Optional execution unit that implements the Advanced SIMD and floating-point architecture support.
- Optional Cryptographic Extension. This architectural extension is only available if the Advanced SIMD and floating-point execution unit is present.
- Generic Interrupt Controller (GIC) CPU interface to connect to an external distributor.
- Generic Timer interface supporting 64-bit count input from an external system counter.

Cache features

- Separate L1 data and instruction caches.
- Optional unified private L2 cache.
- L1 and L2 cache protection in the form of Error Correcting Code (ECC) or parity cache protection on all RAM instances.

Debug features

- Reliability, Availability, and Serviceability (RAS) Extension.
- Armv8.2-A debug logic.
- Performance Monitor Unit (PMU).
- Embedded Trace Macrocell (ETM) that supports instruction trace only.
A1.3 Implementation options

The Cortex-A65 core is highly configurable.

Build-time configuration options make it possible to meet functional requirements with the smallest possible area and power. In a configuration with more than one core, all cores have the same build-time configuration except for the L2 cache inclusion and size.

The following table lists the implementation options for a core.

---

**Note**

The features implemented are shared resources for both threads. The selected cache options are dynamically shared per usage.

---

<table>
<thead>
<tr>
<th>Feature</th>
<th>Range of options</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1 instruction cache size</td>
<td>• 32KB</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>• 64KB</td>
<td></td>
</tr>
<tr>
<td>L1 data cache size</td>
<td>• 32KB</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>• 64KB</td>
<td></td>
</tr>
<tr>
<td>L2 cache</td>
<td>• Included</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>• Not included</td>
<td></td>
</tr>
<tr>
<td>L2 cache size</td>
<td>• 64KB</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>• 128KB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 256KB</td>
<td></td>
</tr>
<tr>
<td>Advanced SIMD and floating-point support</td>
<td>• Included</td>
<td>There is no option to implement floating-point without Advanced SIMD.</td>
</tr>
<tr>
<td></td>
<td>• Not included</td>
<td></td>
</tr>
<tr>
<td>Cryptographic Extension</td>
<td>• Included</td>
<td>Requires Advanced SIMD and floating-point support.</td>
</tr>
<tr>
<td></td>
<td>• Not included</td>
<td></td>
</tr>
<tr>
<td>CoreSight Embedded Logic Analyzer (ELA)</td>
<td>Optional support</td>
<td>Support for integrating CoreSight ELA-500. CoreSight is a range of debug components available as a separately licensable product.</td>
</tr>
<tr>
<td>Dot Product support instructions</td>
<td>• Included</td>
<td>Support for the Armv8.4-A SDOT and UDOT instructions. Requires Advanced SIMD and floating-point support.</td>
</tr>
</tbody>
</table>
A1.4 Supported standards and specifications

The Cortex-A65 core implements the Armv8-A architecture and some architecture extensions. It also supports various interconnect, interrupt, timer, debug, and trace architectures.

<table>
<thead>
<tr>
<th>Architecture specification or standard</th>
<th>Version</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arm architecture</td>
<td>Armv8-A</td>
<td>• AArch64 execution state only</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• All Exception levels, EL0-EL3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A64 instruction set</td>
</tr>
<tr>
<td>Arm architecture extensions</td>
<td>Armv8.1-A</td>
<td>• You cannot implement floating-point without Advanced SIMD.</td>
</tr>
<tr>
<td></td>
<td>extensions</td>
<td>• You cannot implement the Cryptographic Extension without the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Advanced SIMD and floating-point support</td>
</tr>
<tr>
<td></td>
<td>Armv8.2-A</td>
<td>• The Cortex-A65 core implements the LDAPR instructions introduced</td>
</tr>
<tr>
<td></td>
<td>extensions</td>
<td>in the Armv8.3-A extensions.</td>
</tr>
<tr>
<td></td>
<td>Advanced SIMD</td>
<td>• From the Armv8.4-A extensions, the Cortex-A65 core only implements</td>
</tr>
<tr>
<td></td>
<td>and floating-point support</td>
<td>the UD0T and SD0T instructions.</td>
</tr>
<tr>
<td></td>
<td>Cryptographic</td>
<td>• The Cortex-A65 core implements the PSTATE SSBS (Speculative</td>
</tr>
<tr>
<td></td>
<td>Extension</td>
<td>Store Bypass Safe) bit that supports software mitigation for Spectre</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Variant 4 introduced in the Armv8.5-A extension.</td>
</tr>
<tr>
<td></td>
<td>Armv8.3-A</td>
<td>• The Cortex-A65 core implements the PSTATE SSBS (Speculative Store</td>
</tr>
<tr>
<td></td>
<td>LDAPR</td>
<td>Bypass Safe) bit that supports software mitigation for Spectre Variant 4 introduced in the Armv8.5-A extension.</td>
</tr>
<tr>
<td></td>
<td>instructions</td>
<td>• Instruction trace only. Data trace is not supported.</td>
</tr>
<tr>
<td></td>
<td>Armv8.4-A</td>
<td>• System register access to the ETM is not supported.</td>
</tr>
<tr>
<td></td>
<td>Dot Product</td>
<td></td>
</tr>
<tr>
<td></td>
<td>support</td>
<td></td>
</tr>
<tr>
<td></td>
<td>instructions</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Armv8.5-A</td>
<td></td>
</tr>
<tr>
<td></td>
<td>extensions</td>
<td></td>
</tr>
<tr>
<td>Generic Interrupt Controller</td>
<td>GICv4</td>
<td>Backwards compatible with GICv3</td>
</tr>
<tr>
<td>PMU</td>
<td>PMUv3</td>
<td>-</td>
</tr>
<tr>
<td>Debug</td>
<td>Armv8-A</td>
<td>With support for the debug features added by the Armv8.2-A extensions</td>
</tr>
<tr>
<td>CoreSight</td>
<td>CoreSightv3</td>
<td>-</td>
</tr>
<tr>
<td>Embedded Trace Macrocell</td>
<td>ETMv4.2</td>
<td></td>
</tr>
</tbody>
</table>

See Additional reading on page 19 for a list of architectural references.
A1.5 Test features

The Cortex-A65 core provides test signals that enable the use of both Automatic Test Pattern Generation (ATPG) and Memory Built-In Self Test (MBIST) to test the core logic and memory arrays.

For more information, see the Arm® Cortex®-A65 Core Integration Manual.
A1.6 Design tasks

The Cortex-A65 core is delivered as a synthesizable Register Transfer Level (RTL) description in Verilog HDL. Before you can use the Cortex-A65 core, you must implement it, integrate it, and program it.

A different party can perform each of the following tasks. Each task can include implementation and integration choices that affect the behavior and features of the core.

Implementation
The implementer configures and synthesizes the RTL to produce a hard macrocell. This task includes integrating RAMs into the design.

Integration
The integrator connects the macrocell into a SoC. This task includes connecting it to a memory system and peripherals.

Programming
In the final task, the system programmer develops the software to configure and initialize the core and tests the application software.

The operation of the final device depends on the following:

Build configuration
The implementer chooses the options that affect how the RTL source files are pre-processed. These options usually include or exclude logic that affects one or more of the area, maximum frequency, and features of the resulting macrocell.

Configuration inputs
The integrator configures some features of the core by tying inputs to specific values. These configuration settings affect the start-up behavior before any software configuration is made. They can also limit the options available to the software.

Software configuration
The programmer configures the core by programming particular values into registers. The configuration choices affect the behavior of the core.
A1.7 Product revisions

This section indicates the first release and, in subsequent releases, describes the differences in functionality between product revisions.

- **r0p0** First release.
- **r1p0** There are minor changes that affect the programmers model. For more information, see A.1 Revisions on page Appx-A-516.
- **r1p1** No functional changes to core for this revision. For more information, see A.1 Revisions on page Appx-A-516.
This chapter describes the structure of the Cortex-A65 core.

It contains the following sections:

- A2.1 Components on page A2-32.
- A2.2 Interfaces on page A2-36.
- A2.3 About system control on page A2-37.
- A2.4 About the Generic Timer on page A2-38.
A2.1 Components

The cluster consists of:

- One to eight cores.
- The DynamIQ Shared Unit (DSU), which connects the cores to an external memory system.

For more information, see the Arm® DynamIQ™ Shared Unit Technical Reference Manual.

The following figure includes a top-level functional diagram of a core.

Figure A2-1  Core block diagram

* Optional

Note

There are multiple asynchronous bridges between the Cortex-A65 core and the DSU. Only the coherent interface between the Cortex-A65 core and the DSU can be configured to run synchronously, however it does not affect the other interfaces such as debug, trace, and GIC which are always asynchronous. For
more information on how to set the coherent interface to run either synchronously or asynchronously, see Configuration Guidelines in the Arm® DynamIQ™ Shared Unit Configuration and Sign-off Guide.

---

**Instruction Fetch Unit (IFU)**

The IFU fetches instructions from the instruction cache or from external memory and predicts the outcome of branches in the instruction stream. It passes the instructions to the Data Processing Unit (DPU) for processing.

**Data Processing Unit (DPU)**

The DPU decodes and executes instructions. It executes instructions that require data transfer to or from the memory system by interfacing to the Data Cache Unit (DCU). The DPU includes the PMU, the Advanced SIMD and floating-point support, and the Cryptographic Extension.

**PMU**

The PMU provides six performance monitors that can be configured to gather statistics on the operation of each core and the memory system. The information can be used for debug and code profiling.

**Advanced SIMD and floating-point support**

Advanced SIMD is a media and signal processing architecture that adds instructions primarily for audio, video, 3D graphics, image and speech processing. The floating-point architecture provides support for single-precision and double-precision floating-point operations.

All scalar floating-point instructions are available in the A64 instruction set.

The A64 instruction set offers additional Advanced SIMD instructions, including double-precision floating-point vector operations.

---

**Note**

The Advanced SIMD architecture, its associated implementations, and supporting software, are also referred to as NEON™ technology.

---

**Cryptographic Extension**

The optional Cortex-A65 core Cryptographic Extension supports the Armv8-A Cryptographic Extension. It is a configuration option that can be set when configuring and integrating the core into a system and applies to all cores. The Cryptographic Extension adds new instructions to Advanced SIMD that accelerate:

- Advanced Encryption Standard (AES) encryption and decryption.
- The Secure Hash Algorithm (SHA) functions SHA-1, SHA-224, and SHA-256.
- Finite field arithmetic used in algorithms such as Galois/Counter Mode and Elliptic Curve Cryptography.

**Memory Management Unit (MMU)**

The MMU provides fine-grained memory system control through a set of virtual-to-physical address mappings and memory attributes that are held in translation tables. These are saved into the Translation Lookaside Buffer (TLB) when an address is translated. The TLB entries include global and Address Space Identifiers (ASIDs) to prevent context switch TLB flushes. They also include Virtual Machine Identifiers (VMIDs) to prevent TLB flushes on virtual machine switches by the hypervisor.

**L1 TLBs**

The first level of caching for the translation table information is an L1 TLB. It is implemented on both of the instruction and data sides. All TLB-related maintenance operations result in flushing both the instruction and data L1 TLBs.
L2 TLB
A unified L2 TLB handles the misses from the L1 TLBs.
In implementations with core cache protection, parity bits protect the TLB RAMs by enabling the detection of any single-bit error. If an error is detected, the entry is invalidated and fetched again.

L1 memory system
The L1 memory system includes the Data Cache Unit DCU, the Store Buffer (STB), and the Bus Interface Unit (BIU).

DCU
The DCU manages all load and store operations.
The L1 data cache RAMs are protected using Error Correction Codes (ECC). The ECC scheme is Single Error Correct Double Error Detect (SECDED). The DCU includes a combined local and global exclusive monitor that is used by Load-Exclusive and Store-Exclusive instructions.

STB
The STB holds store operations when they have left the load/store pipeline in the DCU and have been committed by the DPU. The STB can request access to the L1 data cache, initiate linefills, or write to L2 and L3 memory systems.
The STB is also used to queue maintenance operations before they are broadcast to other cores in the cluster.

BIU
The BIU contains the interface to the L2 memory system and buffers to decouple the interface from the L1 data cache and STB.

L2 memory system
The L2 memory system contains the L2 cache. The L2 cache is optional and private to each core. The L2 cache is 4-way set associative, supports 64-byte cache lines, and has a configurable cache RAM size between 64KB and 256KB. The L2 memory system is connected to the DynamIQ Shared Unit through an optional asynchronous bridge.

GIC CPU interface
The GIC CPU interface, when integrated with an external distributor component, is a resource for supporting and managing interrupts in a cluster system.

DynamIQ™ Shared Unit
The DynamIQ Shared Unit (DSU) contains the L3 cache and logic required to maintain coherence between the cores in the cluster. For more information, see the Arm® DynamIQ™ Shared Unit Technical Reference Manual.

Debug and trace components
The Cortex-A65 core supports a range of debug, test, and trace options including:
• Six performance event counters, provided by the PMU, and one cycle counter.
• Six hardware breakpoints, and four watchpoints.
• Per-core instruction trace only ETM.
• Per-core support for an ELA-500.
• AMBA 4 APB interfaces between the cluster and the DebugBlock.
Details of the core-specific debug elements can be found in this document. For information on the cluster debug and trace components supported by the Cortex-A65 core, see the Arm® DynamIQ™ Shared Unit Technical Reference Manual.
Related references

Chapter A5 Memory Management Unit on page A5-63
Chapter A6 Level 1 memory system on page A6-73
Chapter A7 Level 2 memory system on page A7-89
Chapter A9 Generic Interrupt Controller CPU interface on page A9-103
Chapter C1 Debug on page C1-301
Chapter C2 Performance Monitor Unit on page C2-309
Chapter C3 Embedded Trace Macrocell on page C3-331
A2.2 Interfaces

The Cortex-A65 core has several interfaces to connect it to a SoC. The DSU manages all interfaces.

For information on the interfaces, see the Arm® DynamIQ™ Shared Unit Technical Reference Manual.
A2.3 About system control

The system registers control and provide status information for the functions that the core implements.

The main functions of the system registers are:
- Overall system control and configuration.
- MMU configuration and management.
- Cache configuration and management.
- System performance monitoring.
- GiC configuration and management.

Some of the system registers are accessible through the external debug interface.
A2.4 About the Generic Timer

The Generic Timer can schedule events and trigger interrupts that are based on an incrementing counter value. It generates timer events as active-LOW interrupt outputs and event streams.

The Cortex-A65 core provides a set of timer registers for each thread. The timers are:

- An EL1 Non-secure physical timer.
- An EL2 Hypervisor physical timer.
- An EL3 Secure physical timer.
- A virtual timer.
- A Hypervisor virtual timer.

The Cortex-A65 core does not include the system counter. This resides in the SoC. The system counter value is distributed to the core with a 64-bit bus.

For more information on the Generic Timer, see the Arm® DynamIQ™ Shared Unit Technical Reference Manual and the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.
Chapter A3
Clocks, resets, and input synchronization

This chapter describes the clocks, resets, and input synchronization of the Cortex-A65 core.

It contains the following sections:

• A3.1 About clocks, resets, and input synchronization on page A3-40.
• A3.2 Asynchronous interface on page A3-41.
A3.1 About clocks, resets, and input synchronization

The Cortex-A65 core supports hierarchical clock gating.

The Cortex-A65 core contains several interfaces that connect to other components in the system. These interfaces can be in the same clock domain or in other clock domains.

For information about clocks, resets, and input synchronization, see the Arm® DynamIQ™ Shared Unit Technical Reference Manual.
A3.2 Asynchronous interface

Your implementation can include an optional asynchronous interface between the core and the DSU top level.

See the Arm® DynamIQ™ Shared Unit Technical Reference Manual for more information.
A3 Clocks, resets, and input synchronization
A3.2 Asynchronous interface
Chapter A4
Power management

This chapter describes the power domains and the power modes in the Cortex-A65 core.

It contains the following sections:
- A4.1 About power management on page A4-44.
- A4.2 Voltage domains on page A4-45.
- A4.3 Power domains on page A4-46.
- A4.5 Power control on page A4-50.
- A4.6 Core power modes on page A4-51.
- A4.7 Thread power modes on page A4-57.
- A4.8 Relationship between power modes and power domains on page A4-59.
- A4.9 Power down sequence on page A4-60.
- A4.10 Debug over powerdown on page A4-61.
A4.1 About power management

The Cortex-A65 core provides mechanisms to control both dynamic and static power dissipation.

The dynamic power management includes the following features:

- Architectural clock gating.
- Per-core Dynamic Voltage and Frequency Scaling (DVFS).

The static power management includes the following features:

- Dynamic retention.
- Powerdown.

Related references

A4.3 Power domains on page A4-46
A4.5 Power control on page A4-50
A4.2 Voltage domains

The Cortex-A65 core supports a VCPU voltage domain and a VSYS voltage domain.

The following figure shows the VCPU and VSYS voltage domains in each Cortex-A65 core and in the DSU. The example shows a configuration with four Cortex-A65 cores.

Asynchronous bridge logic exists between the voltage domains. The Cortex-A65 core logic and core clock domain of the asynchronous bridge are in the VCPU voltage domain. The DSU clock domain of the asynchronous bridge is in the VSYS voltage domain.

--- Note ---

You can tie VCPU and VSYS to the same supply if one of the following conditions is met:
- The core is configured to run synchronously with the DSU sharing the same clock. The core can still be powered down independently if it is in its own power domain with proper isolation.
- The core is not required to support DVFS.
A4.3 Power domains

The Cortex-A65 core supports multiple power domains.

The following figure shows the power domains in the Cortex-A65 core. The colored boxes indicate the PDADVSIMD, PDCPU, and PDSYS power domains, with respective voltage domains shown in dotted lines.

![Cortex-A65 core power domains](image)

In a DSU cluster, there might be multiple Cortex-A65 cores. The following figure shows the power domains for four Cortex-A65 cores in a DSU cluster. Everything in the same color is part of the same power domain. The number of power domains increases based on the number of cores present. This example only shows the power domains that are associated with the Cortex-A65 cores. For information
on the other power domains required for a DSU cluster, see the Power management chapter of the Arm®

![DSU Cluster Diagram]

**Note**

You do not need to use the full flexibility that the Cortex-A65 clock, voltage, and power domains
provide.

The Advanced SIMD and floating-point block in each core is also part of the power domain for that core.
However, to support independent retention control, each Advanced SIMD and floating-point block also
has its own power domain for isolation from the surrounding domain.

The following table shows the power domains that the Cortex-A65 core supports.

<table>
<thead>
<tr>
<th>Power domain</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDCPU&lt;n&gt;</td>
<td>This domain contains all helios_cpu logic and cpu clock domain logic of the asynchronous bridge. It also includes the optional Advanced SIMD and floating-point block, the L1 and L2 TLBs, L1 and L2 core RAMs, and debug registers that are associated with the core. &lt;n&gt; where n is the core number in the range 0-7. The number represents core 0, core 1, core 2, to core 7. If a core is not present, the corresponding power domain is not present.</td>
</tr>
<tr>
<td>PDADVSIMD&lt;n&gt;</td>
<td>This is an optional power domain for Advanced SIMD and floating-point block to implement dynamic retention. &lt;n&gt; where n is the core number in the range 0-7. The number represents core 0, core 1, core 2, to core 7. If a core is not present, the corresponding power domain is not present.</td>
</tr>
<tr>
<td>PDSYS</td>
<td>This domain contains the cluster clock domain logic of the asynchronous bridge.</td>
</tr>
</tbody>
</table>

Clamping cells between power domains are inferred rather than instantiated in the RTL.
A4.4 Architectural clock gating modes

When the core is in standby mode, it is architecturally clock gated at the top of the clock tree.

*Wait for Interrupt* (WFI) and *Wait for Event* (WFE) are features of the Armv8-A architecture that puts the core in a low-power standby mode by architecturally disabling the clock at the top of the clock tree.

If both threads on the core are in WFI or WFE mode, then all registers in the core do not receive a clock signal.

Each thread has a power mode, which is either the run, standby, or shutdown thread power mode. For more information, see A4.7 Thread power modes on page A4-57.

If one of the threads on the core executes a WFI or WFE instruction, and there is no pending wake-up event for that thread, the following occurs:

- All preceding instructions for the thread are completed and retired.
- All following instructions for the thread are flushed.
- The thread moves into standby mode.

The core begins to quiesce when either of the following occurs:

- Both of the threads are in standby mode.
- One thread is in standby mode, and the other thread is in deactivated mode.

The core is put into the lowest power state legally possible:

- If both threads are in WFI or WFE, then the core is put into WFI or WFE respectively.
- If one thread is in WFE and the other is in WFI, then core is put into WFE.
- If one thread is in shutdown mode, and the other thread is in WFI or WFE, then the core is put into WFI or WFE respectively.

This section contains the following subsections:

- **A4.4.1 Core Wait for Interrupt** on page A4-48.
- **A4.4.2 Core Wait for Event** on page A4-49.

### A4.4.1 Core Wait for Interrupt

WFI puts the core in a low-power state by disabling most of the clocks in the core, while keeping the core powered up.

There is a small dynamic power overhead from the logic that is required to wake up the core from WFI low-power state. Other than this, the power that is drawn is reduced to static leakage current only.

When a thread on the core executes the WFI instruction, the thread waits for all instructions in the core to retire before it enters standby mode. When both threads are in standby mode, or one is in standby mode and the other is off, the core will begin to quiesce. This process ensures that all explicit memory accesses that occurred before the WFI instruction have retired in program order. In addition, this process ensures that store instructions have updated the cache or have been issued to the L3 memory system.

While the core is in WFI low-power state, the clocks in the core are temporarily enabled without causing the core to exit WFI low-power state when any of the following events are detected:

- An L3 snoop request that must be serviced by the core data caches.
- A cache or TLB maintenance operation that must be serviced by the core L1 instruction cache, data cache, TLB, or L2 cache.
- An APB access to the debug or trace registers residing in the core power domain.
- A GIC CPU access through the AXI4 stream channel.

Exit from WFI low-power state occurs when one of the following occurs:

- The core detects one of the WFI wake-up events.
- The core detects a reset.
For more information, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

### A4.4.2 Core Wait for Event

WFE is a feature of the Armv8-A architecture. It uses a locking mechanism based on events, to put the core in a low-power state by disabling most of the clocks in the core, while keeping the core powered up.

There is a small dynamic power overhead from the logic that is required to wake up the core from WFE low-power state. Other than this, the power that is drawn is reduced to static leakage current only.

A thread on the core enters into WFE low-power state by executing the `WFE` instruction. When the `WFE` instruction executes, the thread waits for all instructions in the core to complete before it enters the standby mode.

If the event register is set, execution of WFE does not cause entry into standby mode, but clears the event register. When both threads are in standby mode, or one is in standby mode and the other is off, the core will begin to transition to WFE low-power state.

While the core is in WFE low-power state, the clocks in the core are temporarily enabled without causing the core to exit WFE low-power state when any of the following events are detected:

- An external snoop request that must be serviced by the core data caches.
- A cache or TLB maintenance operation that must be serviced by the core L1 instruction cache, data cache, TLB, or L2 cache.
- An APB access to the debug or trace registers residing in the core power domain.
- A GIC CPU access through the AXI4 stream channel.

Exit from WFE low-power state occurs when one of the following occurs:

- The core detects one of the WFE wake-up events.
- The `EVENTI` input signal is asserted.
- The core detects a reset.

For more information, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

*Related references*

* A4.9 Power down sequence on page A4-60
A4.5  Power control

All power mode transitions are performed at the request of the power controller, using a P-Channel interface to communicate with the Cortex-A65 core.

There is one P-Channel per core, plus one P-Channel for the cluster. The Cortex-A65 core provides the current requirements on the PACTIVE signals, so that the power controller can make decisions and request any change with PREQ and PSTATE. The Cortex-A65 core then performs any actions necessary to reach the requested power mode, such as gating clocks, flushing caches, or disabling coherency, before accepting the request.

If the request is not valid, either because of an incorrect transition or because the status has changed so that state is no longer appropriate, then the request is denied. The power mode of each core can be independent of other cores in the cluster, however the cluster power mode is linked to the mode of the cores.
A4.6 Core power modes

The Cortex-A65 core supports the following core power modes for each core domain P-Channel:

- A4.6.1 On on page A4-53.
- A4.6.2 Off on page A4-53.
- A4.6.3 Off (emulated) on page A4-53.
- A4.6.4 SIMD dynamic retention on page A4-53.
- A4.6.5 Core dynamic retention on page A4-54.
- A4.6.6 Debug recovery on page A4-54.
- A4.6.7 Encoding for core power modes on page A4-55.

There are legal transitions between each core power mode. The threads can only run in certain thread power modes (run, standby, or deactivated mode), depending on which core power modes the threads are in. For more information on the corresponding thread power modes for each core power mode and power domain, see A4.8 Relationship between power modes and power domains on page A4-59.

The following figure shows the supported legal transitions between the core power modes, and the corresponding thread power modes that the threads should be in.
This section contains the following subsections:

- **A4.6.1 On** on page A4-53.
- **A4.6.2 Off** on page A4-53.
- **A4.6.3 Off (emulated)** on page A4-53.
- **A4.6.4 SIMD dynamic retention** on page A4-53.
- **A4.6.5 Core dynamic retention** on page A4-54.
- **A4.6.6 Debug recovery** on page A4-54.

For information on core power mode encodings, see **A4.6.7 Encoding for core power modes** on page A4-55.
A4.6.1 On

In this mode, the core is on and fully operational. Each thread can be in run, standby, or deactivated mode.

The core can be initialized into the On mode with the required threads enabled. If the core does not use its P-Channel, you can tie the core in the On mode by tying PREQ LOW.

When a transition to the On mode completes, all caches are accessible and coherent. Other than the normal architectural steps to enable caches, no additional software configuration is required.

When the core domain P-Channel is initialized into the On mode, either as a shortcut for entering that mode or as a tie-off for an unused P-Channel, it is an assumed transition from the Off mode. This includes an invalidation of any cache RAM within the core domain.

A4.6.2 Off

The Cortex-A65 core supports a full shutdown mode where power can be removed completely and no state is retained.

The shutdown can be for either the whole cluster or just for an individual core, which allows other cores in the cluster to continue operating.

In this mode, both threads are deactivated, and core logic and RAMs are off. The domain is inoperable and all core state is lost. The L1 and L2 caches are disabled, flushed, and the core is removed from coherency automatically on transition to Off mode.

A Cold reset can reset the core in this mode.

The core P-Channel can be initialized into this mode.

An attempted debug access when the core domain is off returns an error response on the internal debug interface indicating the core is not available.

A4.6.3 Off (emulated)

In this mode, both threads are deactivated, and Core domain logic and RAMs are kept on. However, a core Warm reset can be asserted externally to emulate a power off scenario while keeping core debug state and allowing debug access.

All debug registers must retain their mode and be accessible from the external debug interface. All other functional interfaces behave as if the core were Off.

A4.6.4 SIMD dynamic retention

In this mode, the Advanced SIMD and floating-point logic is in retention (inoperable but with state retained) and the remainder of the core logic is operational.

This means that if an Advanced SIMD and floating-point instruction in either thread is executed while in this mode, it is stalled until the core enters the On mode.

When the Advanced SIMD and floating-point logic is in retention, the clock to the logic is automatically gated outside of the retained domain.

The SIMD dynamic retention is controlled by the CPUPWRCTLR.SIMD_RET_CTRL bit. Both threads must satisfy the requested idle conditions before the Advanced SIMD and floating-point logic is placed into retention.

Related references

B1.30 CPUPWRCTLR_EL1, Power Control Register, EL1 on page B1-170
A4.6.5 Core dynamic retention

In this mode, all core logic and RAMs are in retention and the core domain is inoperable. The core can enter this power mode when both threads are in WFI mode, both threads are in WFE mode, or when one thread is in WFI mode and the other thread is in WFE mode.

The core dynamic retention can be enabled and disabled separately for WFI and WFE by software running on each thread in the core. Separate timeout values can be programmed for entry into this mode from WFI and WFE mode:

- Use the CPUPWRCTLWFI_RET_CTRL register bits to program timeout values for entry into core dynamic retention mode from WFI mode.
- Use the CPUPWRCTLWFE_RET_CTRL register bits to program timeout values for entry into core dynamic retention mode from WFE mode.

When in dynamic retention and the core is synchronous to the cluster, the clock to the core is automatically gated outside of the domain. However, if the core is running asynchronous to the cluster, the system integrator must gate the clock externally during core dynamic retention. For more information, see the Arm® DynamIQ™ Shared Unit Configuration and Sign-off Guide.

The outputs of the domain must be isolated to prevent buffers without power from propagating unknown values to any operational parts of the system.

When the core is in dynamic retention there is support for Snoop, GIC, and debug access, so the core appears as if it were in WFI or WFE mode. When such an incoming access occurs, it stalls and the On PACTIVE bit is set HIGH. The incoming access proceeds when the domain is returned to On using P-Channel.

When the incoming access completes, and if the core has not exited WFI or WFE mode, then the On PACTIVE bit is set LOW after the programmed retention timeout. The power controller can then request to reenter the core dynamic retention mode.

Related references

B1.30 CPUPWRCTRLW_EL1, Power Control Register, EL1 on page B1-170

A4.6.6 Debug recovery

Debug recovery can be used to assist debug of external watchdog-triggered reset events.

It allows contents of the core L1 data and L2 caches that were present before the reset to be observable after the reset. The contents of the caches are retained and are not altered on the transition back to the On mode.

By default, the core invalidates its caches when transitioning from Off to On mode. If P-Channel is initialized to debug recovery, and the core is cycled through Cold or Warm reset along with system resets, then the cache invalidation is disabled. The cache contents are preserved when the core is transitioned to the On mode.

Debug recovery also supports preserving Reliability, Availability, and Serviceability (RAS) state, in addition to the cache contents. In this case, a transition to debug recovery is made from any of the current states. Once in debug recovery mode, a cluster-wide Warm reset must be applied externally. The RAS and cache state are preserved when the core is transitioned to the On mode.

——— Caution ————

Debug recovery is strictly for debug purposes. It must not be used for functional purposes, as correct operation of the caches is not guaranteed when entering this mode.

- This mode can occur at any time with no guarantee of the state of the core. A P-Channel request of this type is accepted immediately, therefore its effects on the core, cluster, or the wider system are unpredictable, and a wider system reset might be required. In particular, if there were outstanding
memory system transactions at the time of the reset, then these might complete after the reset when the core is not expecting them and cause a system deadlock.

- If the system sends a snoop to the cluster during this mode, then depending on the cluster state, the snoop might get a response and disturb the contents of the caches, or it might not get a response and cause a system deadlock.

### A4.6.7 Encoding for core power modes

The following table shows the encodings for the supported modes for each core domain P-Channel.

<table>
<thead>
<tr>
<th>Core power mode name</th>
<th>Short Name</th>
<th>PACTIVE Bit Number</th>
<th>PSTATE[5:4] value</th>
<th>PSTATE[3:0] value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debug recovery</td>
<td>DBG_RECOV</td>
<td>-</td>
<td>0b00</td>
<td>0b1010</td>
<td>Threads must be deactivated, and PSTATE[5:4] must be 0b00. Logic is off (or in reset), RAM state is retained and not invalidated when transition to On mode.</td>
</tr>
<tr>
<td>On</td>
<td>ON</td>
<td>8</td>
<td>0b01</td>
<td>0b1000</td>
<td>All powerup, each thread can be in Run, Standby, or Deactivated mode. PSTATE[5:4]: 0b00 Both threads are deactivated. 0b01 Thread 0 is activated. 0b10 Thread 1 is activated. 0b11 Both threads are activated.</td>
</tr>
<tr>
<td>SIMD dynamic retention</td>
<td>SIMD_DYN_RET</td>
<td>7</td>
<td>0b01</td>
<td>0b0111</td>
<td>SIMD logic is in retention and inoperable. All other logic is on and operational. PSTATE[5:4]: 0b01 Thread 0 is activated. 0b10 Thread 1 is activated. 0b11 Both threads are activated.</td>
</tr>
<tr>
<td>Core dynamic retention</td>
<td>CORE_DYN_RET</td>
<td>5</td>
<td>0b01</td>
<td>0b0101</td>
<td>Logic and RAM state are inoperable but retained. PSTATE[5:4]: 0b01 Thread 0 is activated. 0b10 Thread 1 is activated. 0b11 Both threads are activated.</td>
</tr>
<tr>
<td>Core power mode name</td>
<td>Short Name</td>
<td>PACTIVE Bit Number</td>
<td>PSTATE[5:4] value</td>
<td>PSTATE[3:0] value</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>------------</td>
<td>--------------------</td>
<td>------------------</td>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Off (emulated)</td>
<td>EMU_OFF</td>
<td>1</td>
<td>0b00</td>
<td>0b0001</td>
<td>Threads must be deactivated, and PSTATE[5:4] must be 0b00. On with Warm reset asserted, debug state is retained and accessible.</td>
</tr>
<tr>
<td>Off</td>
<td>OFF</td>
<td>0 (implicit)</td>
<td>0b00</td>
<td>0b0000</td>
<td>Threads must be deactivated, and PSTATE[5:4] must be 0b00. All powerdown.</td>
</tr>
</tbody>
</table>
A4.7 Thread power modes

Each thread on a Cortex-A65 core has a power mode. The combination of these modes drive the overall core power mode.

This section contains the following subsections:

- A4.7.1 Run mode on page A4-57.
- A4.7.2 Standby mode on page A4-57.
- A4.7.3 Deactivated mode on page A4-57.

A4.7.1 Run mode

Run mode is the normal mode of thread operations.

In run mode, the thread is active and executes instructions.

A4.7.2 Standby mode

A thread enters standby mode through the execution of a \texttt{WFI} or \texttt{WFE} instruction.

In standby mode, the thread and its architectural state are active. However, the instruction execution is suspended.

A wake-up event causes the thread to reenter run mode.

A4.7.3 Deactivated mode

The Cortex-A65 core uses its P-Channel to activate or deactivate threads.

If a thread powers down by setting the CPUPWRCTRLR.CORE_PWRDN_EN bit HIGH and executing WFI, then the power controller can deactivate the thread.

The power controller uses \texttt{PSTATE[5:4]} bits to activate or deactivate both or any of the two threads in the same P-Channel request.

\texttt{PACTIVE[17:16]} bits indicate to the power controller whether the threads are required to be active. If a thread executes the deactivated sequence, then the corresponding \texttt{PACTIVE} bit for that thread is set LOW. The power controller then sets the matching \texttt{PSTATE} bit LOW. This arrangement allows a system power controller to treat each thread as a separate core and manage powerup or powerdown requests for each thread.

If one of the threads executes a deactivated sequence, the core enters a single-threaded linked mode. In this mode, the core can allocate all resources to the remaining thread, which achieves higher performance.

In deactivated mode, the thread is inactive, and all architectural states are \texttt{UNKNOWN}.

The following table shows the corresponding \texttt{PSTATE} and \texttt{PACTIVE} bits for respective threads.

<table>
<thead>
<tr>
<th>\texttt{PSTATE[5]}</th>
<th>\texttt{PSTATE[4]}</th>
<th>\texttt{PACTIVE[17]}</th>
<th>\texttt{PACTIVE[16]}</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0b0</td>
<td>0b0</td>
<td>LOW</td>
<td>LOW</td>
<td>Thread 0 and 1 are deactivated.</td>
</tr>
<tr>
<td>0b0</td>
<td>0b1</td>
<td>LOW</td>
<td>HIGH</td>
<td>Thread 0 is activated. The core enters a single-threaded linked mode.</td>
</tr>
<tr>
<td>0b1</td>
<td>0b0</td>
<td>HIGH</td>
<td>LOW</td>
<td>Thread 1 is activated. The core enters a single-threaded linked mode.</td>
</tr>
<tr>
<td>0b1</td>
<td>0b1</td>
<td>HIGH</td>
<td>HIGH</td>
<td>Thread 0 and 1 are activated.</td>
</tr>
</tbody>
</table>
Threads are active only when the core power modes are in any of the following states:

- On.
- SIMD dynamic retention.

A thread can be activated by a P-Channel request with the corresponding PSTATE[5:4] bits set HIGH while the core power mode is On.

A thread can be deactivated by a P-Channel request with the corresponding PSTATE[5:4] bits set LOW while the core power mode is On.
A4.8 Relationship between power modes and power domains

The following table describes the core power modes, and the corresponding thread power modes and power domain states for individual cores.

The power domains can be controlled independently to give different combinations when powered-up and powered-down. However, only some powered-up and powered-down domain combinations are valid and supported.

—— Caution ——
States that are not shown in the following table are unsupported and must not occur.

Table A4-4 Supported combinations of core power mode, thread power mode, and power domain states

<table>
<thead>
<tr>
<th>Core power mode</th>
<th>Thread power mode</th>
<th>Power domain</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>PDCPU</td>
<td>PDADVSIMD</td>
</tr>
<tr>
<td>Debug recovery</td>
<td>Deactivated for both</td>
<td>On</td>
<td>On</td>
</tr>
<tr>
<td>On</td>
<td>threads</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIMD dynamic</td>
<td>Run, standby, or</td>
<td>On</td>
<td>Ret</td>
</tr>
<tr>
<td>retention</td>
<td>deactivated</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Core dynamic</td>
<td>Deactivated for both</td>
<td>Ret</td>
<td>Ret</td>
</tr>
<tr>
<td>retention</td>
<td>threads</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Off (emulated)</td>
<td>Deactivated for both</td>
<td>On</td>
<td>On</td>
</tr>
<tr>
<td>Off</td>
<td>threads</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Deviating from the legal power modes can lead to unpredictable results. You must comply with the dynamic power management and powerup and powerdown sequences described in the following sections.
A4.9 Power down sequence

The Cortex-A65 core uses the following power down sequence.

To power down a core, perform the following programming sequence on each thread:

1. Save all architectural state.
2. To disable or reroute interrupts away from this thread, configure the GIC distributor.
3. To indicate to the power controller that a powerdown is requested, set the CPUPWRCTRLR.CORE_PWRDN_EN bit to 1.
4. Execute an Instruction Synchronization Barrier (ISB) instruction.
5. Execute a WFI instruction.

After a WFI executes, hardware, under the direction of the power controller, perform all cache disables, cache flushing (L1 and L2), and removal of the core from coherency.

--- Note ---

When the CPUPWRCTRLR.CORE_PWRDN_EN bit is set and WFI instruction is executed, the thread can be deactivated under the direction of power controller. This masks all interrupts and wake-up events in the core for that thread. The power controller can activate the thread via a P-Channel thread activation request.

---

If one of the threads executes a power down sequence, the core enters a single-threaded mode. In this mode, the core can allocate all resources to the remaining thread which achieves higher performance.

Related references

B1.30 CPUPWRCTRLR_EL1, Power Control Register, EL1 on page B1-170
### A4.10 Debug over powerdown

The Cortex-A65 core supports debug over powerdown, which allows a debugger to retain its connection with the core even when powered down. This enables debug to continue through powerdown scenarios, rather than having to re-establish a connection each time the core is powered up.

The debug over powerdown logic is part of the DebugBlock, which is external to the cluster, and must remain powered on during the debug over powerdown process.

See the *Arm® DynamIQ™ Shared Unit Technical Reference Manual*. 
Chapter A5  
Memory Management Unit

This chapter describes the Memory Management Unit (MMU) of the Cortex-A65 core.

It contains the following sections:
- A5.1 About the MMU on page A5-64.
- A5.2 TLB organization on page A5-66.
- A5.3 TLB match process on page A5-67.
- A5.4 Translation table walks on page A5-68.
- A5.5 MMU memory accesses on page A5-69.
- A5.6 Responses on page A5-70.
A5.1 About the MMU

The Memory Management Unit (MMU) is responsible for translating addresses of code and data Virtual Addresses (VA) to Physical Addresses (PAs) in the real system. The MMU also controls memory access permissions, memory ordering, and cache policies for each region of memory.

A5.1.1 Main functions

The three main functions of the MMU are to:

- Control the translation table walk hardware that accesses translation tables in main memory.
- Translate Virtual Addresses (VAs) to Physical Addresses (PAs).
- Provide fine-grained memory system control through a set of virtual-to-physical address mappings and memory attributes that are held in translation tables.

Each stage of address translation uses a set of address translations and associated memory properties that are held in memory mapped tables called translation tables. Translation table entries can be cached into a Translation Lookaside Buffer (TLB).

The following table describes the components included in the MMU.

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instruction L1 TLB</td>
<td>20 entries, fully associative</td>
</tr>
<tr>
<td>Data L1 TLB</td>
<td>32 entries, fully associative</td>
</tr>
<tr>
<td>L2 TLB</td>
<td>1024 entries, 4-way set associative</td>
</tr>
<tr>
<td>Walk cache RAM</td>
<td>64 entries, 4-way set associative</td>
</tr>
<tr>
<td>IPA cache RAM</td>
<td>64 entries, 4-way set associative</td>
</tr>
</tbody>
</table>

L2 TLB entries contain global and Address Space Identifiers (ASID) to prevent context switch TLB flushes.

The TLB entries contain a Virtual Machine Identifier (VMID) to prevent context switch TLB flushes on virtual machine switches by the hypervisor.

The Cortex-A65 core supports a 44-bit physical address range, which allows 16TB of physical memory to be addressed.

A5.1.2 AArch64 MMU behavior

The Cortex-A65 core is an Armv8-A compliant core that supports execution in AArch64 state only.

The following table shows the behavior in AArch64 state.

<table>
<thead>
<tr>
<th>AArch64</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address translation system</td>
<td>The Armv8-A address translation system resembles an extension to the Long descriptor format address translation system to support the expanded virtual and physical address space.</td>
</tr>
<tr>
<td>Translation granule</td>
<td>4KB, 16KB, or 64KB</td>
</tr>
<tr>
<td>ASID size</td>
<td>8 or 16 bits, depending on the value of TCR_ELx.AS.</td>
</tr>
<tr>
<td>VMID size</td>
<td>8 or 16 bits, depending on the value of VTCR_EL2.VS.</td>
</tr>
<tr>
<td>PA size</td>
<td>Maximum 44 bits.</td>
</tr>
</tbody>
</table>
The Cortex-A65 core also supports the *Virtualization Host Extension* (VHE) including ASID space for EL2. When VHE is implemented and enabled, EL2 has the same behavior as EL1.

See the *Arm® Architecture Reference Manual Arm®v8, for Arm®v8-A architecture profile* for more information on concatenated translation tables and for address translation formats.
A5.2 TLB organization

The Translation Lookaside Buffer (TLB) is a cache of recently executed page translations within the MMU. The Cortex-A65 core implements a two-level TLB structure. The L2 TLB stores all page sizes and is responsible for breaking these down into smaller pages when required for the data-side or instruction-side L1 TLB.

TLB lockdown is not supported.

After reset, an Invalidate All operation is executed and all entries in the TLB are invalidated.

A5.2.1 L1 TLB

The first level of caching for the translation table information is an L1 TLB, implemented on each of the instruction and data sides.

The Cortex-A65 L1 instruction TLB supports 4KB, 16KB, 64KB, and 2MB pages.

The Cortex-A65 L1 data TLB supports 4KB, 16KB, and 64KB pages.

Any other page sizes are fractured after the L2 TLB and the appropriate page size sent to the L1 TLB.

All TLB maintenance operations affect both the L1 instruction and data TLBs and cause them to be invalidated.

A5.2.2 L2 TLB

A unified L2 TLB handles any misses from the L1 instruction and data TLBs.

• A 4-way, set-associative, 1024-entry cache.
• Supports all Virtual Memory System Architecture (VMSA) block sizes as described in the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile. See VMSAv8-64 in the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile for more information.

Accesses to the L2 TLB take a variable number of cycles, based on:

• Competing requests from the L1 TLBs.
• TLB maintenance operations in flight.
• Different page size mappings in use.

A5.2.3 IPA cache RAM

The Intermediate Physical Address (IPA) cache RAM holds mappings between the IPAs and Physical Addresses (PAs).

Only Non-secure EL1 and EL0 stage 2 translations use the IPA cache. When a stage 2 translation completes, the cache is updated. The IPA cache is checked whenever a stage 2 translation is required.

Like the L2 TLB, the IPA cache RAM can hold entries for different sizes.

A5.2.4 Walk cache RAM

The walk cache RAM holds the result of a stage 1 translation up to, but not including, the last level.
A5.3 TLB match process

The Armv8-A architecture provides support for multiple maps from the VA space that are translated differently.

TLB entries store the context information that is required to facilitate a match and avoid the need for a TLB flush on a context or virtual machine switch.

Each TLB entry contains a:
- VA.
- PA.
- Set of memory properties that include type and access permissions.

Each entry is either associated with a particular Address Space Identifier (ASID) or is global. In addition, each TLB entry contains a field to store the Virtual Machine Identifier (VMID) in the entry applicable to accesses from Non-secure EL0 and EL1 Exception levels.

Each entry is associated with a particular translation regime.
- EL3 in Secure state.
- EL2 (or EL0 in VHE mode) in Non-secure state.
- EL1 or EL0 in Secure state.
- EL1 or EL0 in Non-secure state.

A TLB match entry occurs when the following conditions are met:
- Its VA, moderated by the page size such as the VA bits[48:N], where N is \( \log_2 \) of the block size for that translation that is stored in the TLB entry, matches the requested address.
- Entry translation regime matches the current translation regime.
- The ASID matches the current ASID held in the CONTEXTIDR, TTBR0, or TTBR1 register associated with the target translation regime, or the entry is marked global.
- The VMID matches the current VMID held in the VTTBR_EL2 register.
- The ASID and VMID matches are ignored when ASID and VMID are not relevant.

ASID is relevant when the translation regime is:
- EL2 in Non-secure state with HCR_EL2.E2H and HCR_EL2.TGE set to 1.
- EL1 or EL0 in Secure state.
- EL1 or EL0 in Non-secure state.

VMID is relevant for EL1 or EL0 in Non-secure state when HCR_EL2.E2H and HCR_EL2.TGE are not both set.
A5.4 Translation table walks

When the Cortex-A65 core generates a memory access, the MMU:

1. Performs a lookup for the requested VA and current translation regime in the relevant instruction or data L1 TLB.
2. If there is a miss in the relevant L1 TLB, the MMU performs a lookup for the requested VA, current ASID, current VMID, and translation regime in the L2 TLB.
3. If there is a miss in the L2 TLB, the MMU performs a hardware translation table walk.

In the case of an L2 TLB miss, the hardware does a translation table walk as long as the MMU is enabled, and the translation using the base register has not been disabled.

If the translation table walk is disabled for a particular base register, the core returns a Translation Fault.

If the TLB finds a matching entry, it uses the information in the entry as follows.

The access permission bits and the domain determine if the access is permitted. If the matching entry does not pass the permission checks, the MMU signals a Permission fault. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile for details of Permission faults, including:

- A description of the various faults.
- The fault codes.
- Information regarding the registers where the fault codes are set.

Page granule

When executing at a particular Exception level, you can configure the hardware translation table walk to use either the 4KB, 16KB, or 64KB translation granule. Program the Translation Granule bit, TG0, in the appropriate translation control register:

- TCR_EL1.
- TCR_EL2.
- TCR_EL3.
- VTCR_EL2.

For TCR_EL1, you can program the Translation Granule bits TG0 and TG1 to configure the translation granule respectively for TTBR0_EL1 and TTBR1_EL1, or TCR_EL2 when VHE is enabled.
A5.5 MMU memory accesses

During a translation table walk, the MMU generates accesses. This section describes the specific behaviors of the core for MMU memory accesses.

A5.5.1 Configuring MMU accesses

By programming the IRGN and ORGN bits, you can configure the MMU in the appropriate TCR_ELx register to perform translation table walks in cacheable or Non-cacheable regions:

If the encoding of both the ORGN and IRGN bits is Write-Back, the data cache lookup is performed and data is read from the data cache. External memory is accessed, if the ORGN and IRGN bit contain different attributes, or if the encoding of the ORGN and IRGN bits is Write-Through or Non-cacheable.

A5.5.2 Hardware management of the Access flag and dirty state

The Cortex-A65 core includes the option to perform hardware updates to the translation tables.

These features are enabled in registers TCR_ELx and VTCR_EL2. To support the hardware management of dirty state, the Dirty Bit Modifier (DBM) field is added to the translation table descriptors as part of Armv8.1-A architecture.

The core supports hardware updates to the Access flag and to dirty state only when the translation tables are held in Inner Write-Back, Outer Write-Back Normal memory regions.

If software requests a hardware update in a region that is not Inner Write-Back or Outer Write-Back Normal memory, then the core returns an abort with the following encoding:

• ESR.ELx.DFSC = 0b110001 for Data Aborts.
• ESR.ELx.IFSC = 0b110001 for Instruction Aborts.

The following table shows the conditions that can cause hardware updates to the Access flag or dirty state.

<table>
<thead>
<tr>
<th>Conditions</th>
<th>DBM</th>
<th>Access permission</th>
<th>Situation</th>
<th>Hardware update in the translation table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instruction to the memory location</td>
<td>DBM bit is 0b1</td>
<td>Stage 1 AP[2] bit is 0b1</td>
<td>The Store-Exclusive fails because the exclusive monitor is not in the exclusive state.</td>
<td>AP[2] bit is updated.</td>
</tr>
<tr>
<td></td>
<td>DBM bit is 0b1</td>
<td>Stage 2 S2AP[1] bit is 0b0</td>
<td></td>
<td>S2AP[1] bit is updated.</td>
</tr>
<tr>
<td>Store-Exclusive</td>
<td>DBM bit is 0b1</td>
<td>Stage 1 AP[2] bit is 0b1</td>
<td></td>
<td>AP[2] bit is updated.</td>
</tr>
</tbody>
</table>
|                         | DBM bit is 0b1 | Stage 2 S2AP[1] bit is 0b0 | The memory location generates any of the following:  
• A synchronous external abort on a write for a store to a memory location.  
• A watchpoint on a write. | S2AP[1] bit is updated. |
| Store                   | DBM bit is 0b1 | Stage 1 AP[2] bit is 0b1 | The compare fails and the location is not updated. | AP[2] bit is updated.          |
|                         | DBM bit is 0b1 | Stage 2 S2AP[1] bit is 0b0 | | S2AP[1] bit is updated. |

For more information about hardware updates of the Access flag and dirty state, see the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.
A5.6 Responses

Certain faults and aborts can cause an exception to be taken because of a memory access.

A5.6.1 MMU responses

When one of the following translations is completed, the MMU generates a response to the requester:

- An L1 TLB hit.
- An L2 TLB hit.
- A translation table walk.

The response from the MMU contains the following information:

- The PA corresponding to the translation.
- A set of permissions.
- Secure or Non-secure.
- All the information required to report aborts. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile for more details.

A5.6.2 MMU aborts

The MMU can detect faults that are related to address translation and can cause exceptions to be taken to the processing element. Faults can include address size, translation, access flags, and permissions.

See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile for more information about aborts.

A5.6.3 External aborts

External aborts are aborts that occur in the memory system rather than aborts that the MMU detects. Normally, external memory aborts are rare. External aborts are caused by errors flagged by the external memory interfaces or are generated because of an uncorrected ECC error in the L1 data cache or L2 cache arrays.

When an external abort to the external interface occurs on an access for a translation table walk access, the MMU returns a synchronous external abort. For a Load Multiple or a Store Multiple operation, the address captured in the fault register is that of the address that generated the synchronous external abort.

A5.6.4 Mis-programming contiguous hints

A programmer might mis-program the translation tables so that:

- The block size being used to translate the address is larger than the size of the input address.
- The address range translated by a set of blocks marked as contiguous, by use of the contiguous bit, is larger than the size of the input address.

If there is this kind of mis-programming, the Cortex-A65 core does not generate a translation fault.

A5.6.5 Conflict aborts

If a conflict abort is detected in the L2 TLB, the MMU chooses one valid translation and does not generate a conflict abort.

See also the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile for more information.

A5.6.6 Memory Behavior

The Cortex-A65 core supports all the Armv8-A memory types.

However, the following behaviors are simplified and so for best performance their use is not recommended:
**Write-Through**

Memory that is marked as Write-Through cannot be cached on the data-side and does not make coherency requests. On the instruction-side, areas that are marked as Write-Through and Write-Back can be cached in the L1 instruction cache. However, only areas marked as Write-Back can be cached in the L2 cache or the L3 cache.

**Mixed inner and outer cacheability**

Memory that is not marked as inner and outer Write-Back cannot be cached on the data-side and does not make coherency requests. This applies to the memory type only, and not to the allocation hints. All caches within the cluster are treated as being part of the inner cacheability domain.

For more information on supported memory behaviors, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

### A5.6.7 Support for Arm®v8-A device memory types

The Armv8-A architecture includes memory types that replace the Armv7 Device and Strongly-ordered memory types. These device memory types have the following three attributes:

- **G – Gathering**
  
  The capability to gather and merge requests together into a single transaction.

- **R – Reordering**
  
  The capability to reorder transactions.

- **E – Early Write Acknowledgement**
  
  The capability to accept early acknowledge of transactions from the interconnect.

The legal combinations are described in the following table:

<table>
<thead>
<tr>
<th>Memory type</th>
<th>Cortex-A65 support</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRE</td>
<td>Yes</td>
<td>Similar to Normal non-cacheable, but does not permit speculative accesses.</td>
</tr>
<tr>
<td>nGRE</td>
<td>Yes</td>
<td>Transactions might be reordered within the L3 memory system, or in the system interconnect.</td>
</tr>
<tr>
<td>nGnRE</td>
<td>Yes</td>
<td>Corresponds to Device in Armv7.</td>
</tr>
<tr>
<td>nGnRnE</td>
<td>Yes</td>
<td>Corresponds to Strongly Ordered in Armv7. Treated the same as nGnRE inside the Cortex-A65 core, but reported differently on the bus interface.</td>
</tr>
</tbody>
</table>

For more information, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*. 
Chapter A6
Level 1 memory system

This chapter describes the L1 instruction cache and data cache that make up the L1 memory system.

It contains the following sections:
• A6.1 About the L1 memory system on page A6-74.
• A6.2 Cache behavior on page A6-75.
• A6.3 L1 instruction memory system on page A6-78.
• A6.4 L1 data memory system on page A6-79.
• A6.5 Data prefetching on page A6-82.
• A6.6 Direct access to internal memory on page A6-83.
A6.1 About the L1 memory system

The L1 memory system enhances the performance and power efficiency in the Cortex-A65 core. It consists of separate instruction and data caches. You can configure instruction and data caches independently during implementation to sizes of 32KB or 64KB.

L1 instruction-side memory system
The L1 instruction-side memory system provides an instruction stream to the DPU. Its key features are:
- 64-byte instruction side cache line length.
- 4-way set associative L1 instruction cache.
- 128-bit read interface to the L2 memory system.

The Cortex-A65 core uses extensive branch prediction to improve Instructions Per Clock (IPC) and power efficiency.

L1 data-side memory system
The L1 data-side memory system responds to load and store requests from the DPU. It also responds to SCU snoop requests from other cores, or external masters. Its key features are:
- 64-byte data side cache line length.
- 4-way set associative L1 data cache.
- Read buffer that services both the Data Cache Unit (DCU), and the Instruction Fetch Unit (IFU).
- 128-bit read path from the data L1 memory system to the datapath.
- 512-bit write path from the datapath to the L1 memory system.
- Merging store buffer capability which writes to all types of memory (device, normal cacheable and normal non-cacheable).
- Data side prefetch engine that detects patterns of strides with multiple streams are allowed in parallel, capable of detecting both constant and patterns of strides.
A6.2 Cache behavior

On a cache miss, the cache performs a critical word-first fill.

This word-first fill is an implementation-specific feature of the instruction and data caches.

A6.2.1 Instruction cache disabled behavior

If the instruction cache is disabled, all instruction fetches to cacheable memory are treated as if they were non-cacheable.

This means that instruction fetches might not be coherent with caches in other cores, and software must take account of this.

Lines may still be allocated into the instruction cache even if the memory is marked non-cacheable or the instruction cache is disabled. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile for more information.

A6.2.2 Instruction cache speculative memory accesses

Instruction fetches are speculative, as there can be several unresolved branches in the pipeline. There is no execution guarantee.

A branch instruction or exception in the code stream can cause a pipeline flush, discarding the currently fetched instructions. On instruction fetch accesses, pages with Device memory type attributes are treated as Non-Cacheable Normal Memory.

Device memory pages must be marked with the translation table descriptor attribute bit Execute Never (XN). The device and code address spaces must be separated in the physical memory map. This separation prevents speculative fetches to read-sensitive devices when address translation is disabled.

If the instruction cache is enabled, and if the instruction fetches miss in the L1 instruction cache, they can still look up in the L1 data caches. However, a new line is not allocated in the data cache unless the data cache is enabled.

See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile for more information.

A6.2.3 Data cache disabled behavior

If the SCTLR.C bit is set to 0, load and store instructions do not access any of the L1 data, L2, or DSU L3 caches.

The SCTLR.C bit controls whether accesses from the core can look up and allocate into the data cache and unified L2 or L3 caches. Data cache maintenance operations execute normally, regardless of how the SCTLR.C bit is set.

If the SCTLR.C bit is set to 0, then the following apply:

• Instruction fetches cannot allocate in the L2 or L3 caches.
• All load and store instructions to cacheable memory are treated as if they were non-cacheable.
  Therefore, they are not coherent with the caches in this core or the caches in other cores, and software must take this into account.

The L2 and L1 data caches cannot be disabled independently.

For more information, see Table B1-5 Other system control registers on page B1-125.

A6.2.4 Data cache maintenance considerations

DC IVAC instructions in AArch64 perform an invalidate of the target address.

If the data is dirty, a clean is performed before the invalidate.

DC ISW and DC CSW instructions in AArch64 perform both a clean and invalidate of the target set/way. The values of HCR.SWIO and HCR_EL2.SWIO have no effect.
A6.2.5 Data cache coherency

The Cortex-A65 core uses the MESI protocol to maintain data coherency between multiple cores.

MESI describes the state that a shareable line in a L1 data cache can be in:

- **M** Modified/UniqueDirty (UD). The line is in only this cache and is dirty.
- **E** Exclusive/UniqueClean (UC). The line is in only this cache and is clean.
- **S** Shared/SharedClean (SC). The line is possibly in more than one cache and is clean.
- **I** Invalid/Invalid (I). The line is not in this cache.

The DCU stores the MESI state of the cache line in the tag and dirty RAMs.

---

**Note**

The names UniqueDirty, SharedDirty, UniqueClean, SharedClean, and Invalid are the AMBA names for the cache states. The Cortex-A65 core does not use the SharedDirty AMBA state.

---

A6.2.6 Write Streaming Mode

A cache line is allocated to the L1 on either a read miss or a write miss.

However, there are some situations where allocating on writes is not required. For example, when executing the C standard library `memset()` function to clear a large block of memory to a known value. Writes of large blocks of data can pollute the cache with unnecessary data. It can also waste power and performance if a linefill must be performed only to discard the linefill data because the entire line was subsequently written by the `memset()`.

To counter this, the BIU includes logic to detect when the core has written a full cache line before the linefill completes. If this situation is detected on a configurable number of consecutive linefills, then it switches into write streaming mode. This is sometimes referred to as read allocate mode.

When in write streaming mode, loads will behave as normal, and can still cause linefills, and writes will still lookup in the cache, but if they miss then they will write out to L2 (or possibly L3) rather than starting a linefill.

---

**Note**

More than the specified number of linefills might be observed on the ACE or CHI master interface, before the BIU detects that three full cache lines have been written and switches to write streaming mode.

---

The BIU continues in write streaming mode until it detects either a cacheable write burst that is not a full cache line, or there is a load from the same line as is currently being written to L2.

When a core has dropped into write streaming mode, the BIU continues to monitor the bus traffic and will signal to the L2 for it to go into write streaming mode when a further number of full cache line writes are seen.

The Cortex-A65 core is a multithreaded core. Entering write streaming mode is managed independently by each of the two threads.

**AArch64 state**

- `CPUECTL_EL1.L1WSCTL` configures the L1 write streaming mode threshold.
- `CPUECTL_EL1.L2WSCTL` configures the L2 write streaming mode threshold.
- `CPUECTL_EL1.L3WSCTL` configures the L3 write streaming mode threshold.

For more information, see [B1.25 CPUECTL_EL1, CPU Extended Control Register, EL1 on page B1-158](#).
A6.2.7 Data cache invalidate on reset

The Armv8-A architecture does not support an operation to invalidate the entire data cache.

The Cortex-A65 core automatically invalidates caches on reset unless suppressed with the debug recovery P-channel state. It is therefore not necessary for software to invalidate the caches on startup.
A6.3 L1 instruction memory system

The L1 instruction side memory system provides an instruction stream to the Data Processing Unit (DPU).

To increase overall performance and to reduce power consumption, it uses:
- Dynamic branch prediction.
- Instruction caching.

A6.3.1 Program flow prediction

The Cortex-A65 core contains program flow prediction hardware, also known as branch prediction. Branch prediction increases overall performance and reduces power consumption. With program flow prediction disabled, all taken branches incur a penalty that is associated with flushing the pipeline.

To avoid this penalty, the branch prediction hardware predicts if a conditional or unconditional branch is to be taken. For conditional branches, the hardware predicts if the branch is to be taken. It also predicts the address that the branch goes to, known as the branch target address. For unconditional branches, only the target is predicted.

The hardware contains the following functionality:
- A BTAC holding the branch target address of previously taken branches.
- Dynamic branch predictor history.
- The return stack, a stack of nested subroutine return addresses.
- A static branch predictor.
- An indirect branch predictor.

Predicted and non-predicted instructions

Unless otherwise specified, the following list applies to A64 instructions. As a rule the flow prediction hardware predicts all branch instructions regardless of the addressing mode, and includes:
- Conditional branches.
- Unconditional branches.
- Indirect branches that are associated with procedure call and return instructions.

The following branch instructions are not predicted:
- Exception return instructions.

Return stack

The return stack stores the address and instruction set state.

This address is equal to the link register value stored in X30.

In AArch64 state, the RET instruction causes a return stack pop.

As exception return instructions can change core privilege mode and security state, they are not predicted. These include:
- ERET
A6.4 L1 data memory system

The L1 data cache is organized as a Virtually Indexed Physically Tagged (VIPT) cache, with alias avoidance logic so that it appears to software as if it were physically indexed.

The Armv8-A architecture does not support an operation to invalidate the entire data cache. If software requires this function, it must be constructed by iterating over the cache geometry and executing a series of individual invalidate by set/way instructions.

A6.4.1 Memory system implementation

This section describes the implementation of the L1 memory system.

Limited Order Regions

The Cortex-A65 core supports four limited order ranges that include the entire memory space.

Atomic instructions

The Cortex-A65 core supports the atomic instructions added in the Armv8.1-A architecture.

Atomic instructions to cachable memory can be performed as either near atomics or far atomics, depending on where the cache line containing the data resides. If the instruction hits in the L1 data cache in a unique state then it will be performed as a near atomic in the L1 memory system. If the atomic operation misses in the L1 cache, or the line is shared with another core then the atomic is sent as a far atomic out to the L3 cache. If the operation misses everywhere within the cluster, and the master interface is configured as CHI, and the interconnect supports far atomics, then the atomic will be passed on to the interconnect to perform the operation. If the operation hits anywhere inside the cluster, or the interconnect does not support atomics, then the L3 memory system will perform the atomic operation and allocate the line into the L3 cache if it is not already there.

The Cortex-A65 core supports atomics to device or non-cacheable memory, however this relies on the interconnect also supporting atomics. If such an atomic instruction is executed when the interconnect does not support them, it will result in a synchronous Data Abort (for load atomics) or an asynchronous Data Abort (for store atomics). The behavior of the atomic instructions can be modified by the CPUECTLR register settings.

For more information on the CPUECTLR register, see B1.25 CPUECTLR_EL1, CPU Extended Control Register, EL1 on page B1-158.

LDAPR instructions

The core supports Load acquire instructions adhering to the RCpc consistency semantic introduced in the Armv8.3-A extensions. This is reflected in register ID_AA64ISAR1_EL1 where bits[23:20] are set to 0b0001 to indicate that the core supports LDAPRB, LDAPRH, and LDAPR instructions implemented in AArch64.

For more information on the ID_AA64ISAR1_EL1 register, see B1.53 ID_AA64ISAR0_EL1, AArch64 Instruction Set Attribute Register 0, EL1 on page B1-201.

Transient memory region

The core has a specific behavior for memory regions that are marked as Write-Back cacheable and transient, as defined in the Armv8-A architecture.

For any load that is targeted at a memory region that is marked as transient, the following occurs:

- If the memory access misses in the L1 data cache, the returned cache line is allocated in the L1 data cache but is marked as transient.
- On eviction, if the line is clean and marked as transient, it is not allocated into the L2 cache but is marked as invalid.
For streams of contiguous stores that are targeted at a memory region that is marked as transient, the following occurs:

- A line may or may not be allocated if the stores coalesce a full cache line and miss the L1 data cache, the cache line is streamed to the external memory system without being allocated into the L1 data cache, the L2 cache, or the L3 cache.
- A line may or may not be allocated if the stores do not cover the entire cache line and miss the L1 data cache, the modified cache line will not allocate into the L1 data cache but will allocate and write into the L2 cache, marking it as transient.

**Non-temporal loads**

Non-temporal loads indicate to the caches that the data is likely to be used for only short periods. For example, when streaming single-use read data that is then discarded. In addition to non-temporal loads, there are also prefetch-memory (PRFM) hint instructions with the STRM qualifier.

Non-temporal loads cause allocation into the L1 data cache, with the same performance as normal loads. However, when a later linefill is allocated into the cache, the cacheline marked as non-temporal has higher priority to be replaced. To prevent pollution of the L2 cache, a non-temporal line that is evicted from L1, is not allocated to L2 as would happen for a normal line.

--- Note ---

The line is only marked as non-temporal in the cache if the core has the line in a unique state. If shared with other cores, the line is treated normally.

---

Non-temporal stores are treated as if write streaming mode was active. They are not allocated into any cache in the cluster unless they hit in the cache.

**A6.4.2 Internal exclusive monitor**

The Cortex-A65 core L1 memory system has internal exclusive monitors, one for each thread.

The exclusive monitor for each thread is a 2-state, open and exclusive, state machine that manages Load-Exclusive or Store-Exclusive accesses and Clear-Exclusive (CLREX) instructions. You can use these instructions to construct semaphores, ensuring synchronization between different processes running on the core, and also between different cores that are using the same coherent memory locations for the semaphore. A Load-Exclusive instruction tags a small block of memory for exclusive access. CTR.ERG defines the size of the tagged block as 16 words, one cache line.

--- Note ---

A load/store exclusive instruction is any instruction that has a mnemonic starting with LDX, LDAX, STX, or STLX.

---

If a Load-Exclusive instruction is performed to non-cacheable or device memory, and is to a region of memory in the SoC that does not support exclusive accesses, it causes a Data Abort exception with a Data Fault Status Code of either:

- 0b110101, when using the long descriptor format.
- 0b10101, when using the short descriptor format.

See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information about these instructions.

**Treatment of intervening STR operations**

Where there is an intervening store operation between an exclusive load and an exclusive store from the same thread, the intermediate store does not produce any direct effect on the internal exclusive monitor.

After the exclusive load, the local monitor is in the Exclusive Access state. It remains in the Exclusive Access state after the store, and then returns to the Open Access state only after an exclusive store, a CLREX instruction, or an exception return.
However, if the exclusive code sequence accessed address is in cacheable memory, any eviction of the cache line containing that address clears the monitor. Arm recommends that no load or store instructions are placed between the exclusive load and the exclusive store, because these additional instructions can cause a cache eviction. Any data cache maintenance instruction can also clear the exclusive monitor.

——— Note ———
A store from the opposite thread in the same core will clear the internal exclusive monitor.

A6.4.3 Exclusive monitor

In the exclusive state machine, the IMPLEMENTATION DEFINED transitions are as follows:

- If the monitor is in the exclusive state, and a store exclusive is performed to a different address, then the store exclusive fails and does not update memory.
- If the monitor is in the exclusive state, and a store exclusive is performed on the same thread to a different address, then the store exclusive will:
  - Fail.
  - Clear the local exclusive monitor.
  - Not update the memory.
  - Not clear the global monitor.
  - Not send a wakeup event.

In this case, clearing the global monitor and sending the wakeup event is IMPLEMENTATION DEFINED.
- If a normal store is performed to a different address, it does not affect the exclusive monitor.
- If a normal store is performed from a different thread to the same address it clears the exclusive monitor. If the store is from the same thread then it does not clear the monitor.
A6.5 Data prefetching

The following section describes the software and hardware data prefetching behavior of the Cortex-A65 core.

**Hardware data prefetcher**

The Cortex-A65 core has a data prefetch mechanism that looks for cache line fetches with regular patterns. If the data prefetcher detects a pattern, then it signals to the memory system that memory accesses from a specified address are likely to occur soon. The memory system responds by starting new linefills to fetch the predicted addresses ahead of the demand loads.

The Cortex-A65 core can track multiple streams in parallel.

Prefetch streams end when either:

- The pattern is broken.
- A DSB is executed.
- A WFI or WFE is executed.
- A data cache maintenance operation is executed.

For read streams, the prefetcher is based on the virtual addresses. A given stream is allowed to prefetch addresses through multiple pages as long as they are cacheable and with read permissions. If the new page is still cacheable and has read permission, it can cross page boundaries. Write streams are based on physical addresses and so cannot cross page boundaries. However, if full cache line writes are performed then the prefetcher does not activate and write streaming mode is used instead.

For some types of pattern, when the prefetcher is confident in the stream, it can start progressively increasing the prefetch distance ahead of the current accesses. These accesses start to allocate to the L3 cache rather than L1. Allocating to the L3 cache allows better utilization of the larger resources available at L3. Also, utilizing the L3 cache reduces the amount of pollution of the L1 cache if the stream ends or is incorrectly predicted. If the prefetching to L3 was accurate, the line will be removed from L3 and allocated to L1 when the stream reaches that address.

The CPUECTLR register allows you to:

- Deactivate the prefetcher.
- Alter the number of outstanding requests that the prefetcher can make.

**Preload instructions**

The Cortex-A65 core supports PRFM instructions. If PRFM miss and are to a cacheable address, then these instructions perform a lookup in the cache and start a linefill. A prefetch request can be sent to L2 or L3 to start a linefill, and then the instruction can retire without any data being returned to L1.

PLI, PLIL1KEEP, and PLIL1STRM are implemented as a prefetch to L2.

Use the PRFM instruction for data prefetching where short sequences or irregular pattern fetches are required. For more information about prefetch memory and preloading caches, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

**Data Cache Zero**

The Data Cache Zero by Virtual Address (DC ZVA) instruction enables a block of 64-bytes in memory, which is aligned to 64-bytes in size, to be set to 0. The DCZID_EL0 register passes this value.

The DC ZVA instruction allocates this value into the data cache using the same method as a normal store instruction.
A6.6 Direct access to internal memory

The Cortex-A65 core provides a mechanism to read the internal memory that is used by the L1 cache and TLB structures through implementation defined system registers. This functionality can be useful when investigating issues where the coherency between the data in the cache and data in system memory is broken.

When the core executes in AArch64 state, the appropriate memory block and location are selected using several write-only registers. The data is read from read-only registers as shown in the following table. These operations are available only in EL3. In all other modes, executing these instructions results in an Undefined Instruction exception.

<table>
<thead>
<tr>
<th>Register name</th>
<th>Function</th>
<th>Access</th>
<th>Operation</th>
<th>Rd Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDBGDR0_EL3</td>
<td>Data Register 0</td>
<td>Read-only</td>
<td>MRS &lt;Xd&gt;, S3_6_c15_c0_0</td>
<td>Data</td>
</tr>
<tr>
<td>CDBGDR1_EL3</td>
<td>Data Register 1</td>
<td>Read-only</td>
<td>MRS &lt;Xd&gt;, S3_6_c15_c0_1</td>
<td>Data</td>
</tr>
<tr>
<td>CDBGDR2_EL3</td>
<td>Data Register 2</td>
<td>Read-only</td>
<td>MRS &lt;Xd&gt;, S3_6_c15_c0_2</td>
<td>Data</td>
</tr>
<tr>
<td>CDBGDCT_EL3</td>
<td>Data Cache Tag Read Operation Register</td>
<td>Write-only</td>
<td>MSR S1_6_c15_c2_0, &lt;Xd&gt;</td>
<td>Set/Way</td>
</tr>
<tr>
<td>CDBGICT_EL3</td>
<td>Instruction Cache Tag Read Operation Register</td>
<td>Write-only</td>
<td>MSR S1_6_c15_c2_1, &lt;Xd&gt;</td>
<td>Set/Way</td>
</tr>
<tr>
<td>CDBGTT_EL3</td>
<td>TLB Tag Read Operation Register</td>
<td>Write-only</td>
<td>MSR S1_6_c15_c2_2, &lt;Xd&gt;</td>
<td>Index/Way</td>
</tr>
<tr>
<td>CDBGDCD_EL3</td>
<td>Data Cache Data Read Operation Register</td>
<td>Write-only</td>
<td>MSR S1_6_c15_c4_0, &lt;Xd&gt;</td>
<td>Set/Way/Offset</td>
</tr>
<tr>
<td>CDBGICD_EL3</td>
<td>Instruction Cache Data Read Operation Register</td>
<td>Write-only</td>
<td>MSR S1_6_c15_c4_1, &lt;Xd&gt;</td>
<td>Set/Way/Offset</td>
</tr>
<tr>
<td>CDBGTD_EL3</td>
<td>TLB Data Read Operation Register</td>
<td>Write-only</td>
<td>MSR S1_6_c15_c4_2, &lt;Xd&gt;</td>
<td>Index/Way</td>
</tr>
</tbody>
</table>

A6.6.1 Encoding for tag and data in the L1 data cache

The Cortex-A65 L1 data cache is a 4-way set associative structure.

The size of the configured cache determines the number of sets in each way. The following table shows the encoding (set in Rd in the appropriate MCR instruction) used to locate the cache data entry for tag and data memory. It is similar for both the tag and data RAM access.

Data RAM access includes an extra field to locate the appropriate word in the cache line. The set-index range parameter (S) is:

- S=13 for a 32KB cache.
- S=14 for a 64KB cache.

<table>
<thead>
<tr>
<th>Bitfield of Rd</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[31:30]</td>
<td>Cache Way</td>
</tr>
<tr>
<td>[29:S]</td>
<td>Unused</td>
</tr>
<tr>
<td>[S-1:6]</td>
<td>Set index</td>
</tr>
<tr>
<td>[5:3]</td>
<td>Cache data element offset</td>
</tr>
<tr>
<td>[2:0]</td>
<td>Unused (Zero)</td>
</tr>
</tbody>
</table>

Tag information (MESI state, outer attributes, and valid) for the selected cache line, returns using Data Register 0 and Data Register 1.
Use the format that is shown in the following table.

### Table A6-3  Cortex-A65 L1 Data Cache Tag data format

<table>
<thead>
<tr>
<th>Bitfield of Data Register 0 and 1</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DR1[31:30]</td>
<td>MESI State (from tag RAM):</td>
</tr>
<tr>
<td></td>
<td>0b00 Invalid</td>
</tr>
<tr>
<td></td>
<td>0b01 Shared</td>
</tr>
<tr>
<td></td>
<td>0b10 Unique non-transient</td>
</tr>
<tr>
<td></td>
<td>0b11 Unique transient</td>
</tr>
<tr>
<td>DR1[29]</td>
<td>Non-secure state (NS) (from tag RAM)</td>
</tr>
<tr>
<td>DR1[0]</td>
<td>Unused (Zero)</td>
</tr>
<tr>
<td>DR0[31:5]</td>
<td>Unused (Zero)</td>
</tr>
<tr>
<td>DR0[4]</td>
<td>Dirty bit (from Dirty RAM)</td>
</tr>
<tr>
<td>DR0[3]</td>
<td>Shareability (from Dirty RAM)</td>
</tr>
<tr>
<td>DR0[2:1]</td>
<td>Age (from Dirty RAM)</td>
</tr>
<tr>
<td>DR0[0]</td>
<td>Outer Allocation Hint (from Dirty RAM)</td>
</tr>
</tbody>
</table>

The 64 bits of cache data returns in Data register 0 and Data register 1.

### A6.6.2 Encoding for tag and data in the L1 instruction cache

The L1 instruction cache is different from the L1 data cache. This is shown in the encodings and data format used in the cache debug operations that are used to access the tag and data memories.

The following table shows the encoding that is required to select a given cache line.

The set-index range parameter (S) is:

- **S=13**  For a 32KB cache.
- **S=14**  For a 64KB cache.

### Table A6-4  Cortex-A65 Instruction Cache Tag and Data location encoding

<table>
<thead>
<tr>
<th>Bitfield of Rd</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[31:30]</td>
<td>Cache Way</td>
</tr>
<tr>
<td>[29:S]</td>
<td>Unused</td>
</tr>
<tr>
<td>[S-1:6]</td>
<td>Set index</td>
</tr>
<tr>
<td>[5:2]</td>
<td>Cache data element offset (Data Register only)</td>
</tr>
<tr>
<td>[1:0]</td>
<td>Unused</td>
</tr>
</tbody>
</table>

The following table shows the tag, instruction, and valid data for the selected cache line using only Data Register.
The cache data RAMs store instructions in a pre-decoded format. The L1 Instruction Cache Data Read Operation returns two 20-bit entries from the cache in Data Register 0. Each corresponds to the 16-bit aligned offset in the cache line:

**Data Register 0[19:0]** Pre-decode data from cache offset.
**Data Register 1[19:0]** Pre-decode data from cache offset +2.

### A6.6.3 Encoding for the L2 TLB

The Cortex-A65 core L2 TLB is built from a 4-way set associative RAM-based structure and contains the data for the main TLB RAM, the Walk cache and IPA cache.

To read the individual entries into the data registers, software must write to the TLB Tag Read Operation Register and to the TLB Data Read Operation Register.

#### Table A6-6 Cortex-A65 TLB Data Read Operation Register location encoding

<table>
<thead>
<tr>
<th>Bitfield of Rd</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[31:30]</td>
<td>TLB Way</td>
</tr>
<tr>
<td>[29:9]</td>
<td>Unused</td>
</tr>
<tr>
<td>[8:0]</td>
<td>TLB index</td>
</tr>
</tbody>
</table>

The TLB index is used to select the index from the TLB, walk cache, or IPA cache.

#### Table A6-7 TLB index

<table>
<thead>
<tr>
<th>Bitfield of Rd</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000-0FF</td>
<td>Main TLB</td>
</tr>
<tr>
<td>0x100-10F</td>
<td>Walk cache</td>
</tr>
<tr>
<td>0x110-11F</td>
<td>IPA cache</td>
</tr>
</tbody>
</table>

The TLB uses an encoding for the descriptor that is returned using the following Data Registers:

**Data Register 0[31:0]** TLB Descriptor[31:0]
**Data Register 1[31:0]** TLB Descriptor[63:32]
**Data Register 2[31:0]** TLB Descriptor[88:64]
A6.6.4 Main TLB RAM descriptor fields

The Main TLB RAM is divided into two parts, where one part for storing the tag and the other for storing the data. The following tables list the descriptor fields.

**Table A6-8 TLB descriptor fields for Tag RAM**

<table>
<thead>
<tr>
<th>Field</th>
<th>Bits</th>
<th>Width</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid</td>
<td>[0]</td>
<td>1</td>
<td>Indicates that the entry is valid.</td>
</tr>
<tr>
<td>NS (walk)</td>
<td>[1]</td>
<td>1</td>
<td>The security state of core. Used to compare with the NS state for TLB lookup entry match.</td>
</tr>
<tr>
<td>ASID</td>
<td>[17:2]</td>
<td>16</td>
<td>Indicates the Address Space Identifier (ASID). This field will be 0 if ASID is not used.</td>
</tr>
<tr>
<td>VMID</td>
<td>[33:18]</td>
<td>16</td>
<td>Indicates the virtual machine identifier. This field will be 0 if VMID is not used.</td>
</tr>
</tbody>
</table>
| Size    | [36:34] | 3     | Indicates the combined page size of stage 1 and stage 2. The Domain[1] bit (bit[44]) is used together to encode the size information as a 3-bit field combination {Domain[1]:Size[2:1]}:
|         |      |       | 0b000 4KB  |
|         |      |       | 0b001 64KB |
|         |      |       | 0b010 2MB   |
|         |      |       | 0b011 512MB |
|         |      |       | 0b100 16KB  |
|         |      |       | 0b101 32MB  |
| nG      | [37] | 1     | Indicates the non-global bit.                                              |
| AP/HYP  | [40:38] | 3     | Indicates that stage 1 access permission is represented by the S2AP field for either:
|         |      |       | • EL2/EL3 translation regimes.                                            |
|         |      |       | • EL0 translation regimes, which are non-secure when the Virtual Host Extension (VHE) is configured. |
| S2AP    | [42:41] | 2     | When VHE is not configured, S2AP indicates the stage 2 permission for EL1/EL0. When VHE is configured, S2AP[0] indicates that the cached translation applies to EL3/EL2/EL0. Additionally, S2AP[1] represents stage 1 access permission. |
| Domain  | [46:43] | 4     | Indicates miscellaneous control information like Common not Private bit state, AP[1] access permission for EL0 in VHE or stage 1 Dirty Bit Modifier (DBM) bit when not in VHE mode, or last level of translation table walk as Level 3 or Level 2 for each translation granule size. |
| S1 Size | [49:47] | 3     | Indicates the page or block size of the stage 1 translation result.       |
| Address Sign bit | [50] | 1     | Indicates the VA sign bit, VA[48].                                        |
| VA      | [78:51] | 28    | Indicates the virtual address.                                            |
| DBM     | [79]  | 1     | Indicates the Dirty Bit Modifier (DBM) bit for stage 1 translation table walks when in VHE mode or stage 2 translation table walks when not in VHE mode. |
| Parity  | [81:80] | 2     | Indicates the parity bits.                                                |
### Table A6-9 TLB descriptor fields for Data RAM

<table>
<thead>
<tr>
<th>Field</th>
<th>Bits</th>
<th>Width</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XS1Usr</td>
<td>[0]</td>
<td>1</td>
<td>Executable in stage 1 user mode</td>
</tr>
<tr>
<td>XS1Non-Usr</td>
<td>[1]</td>
<td>1</td>
<td>Executable in stage 1 non-user mode</td>
</tr>
<tr>
<td>XS2Usr</td>
<td>[2]</td>
<td>1</td>
<td>Executable in stage 2 user mode</td>
</tr>
<tr>
<td>XS2Non-Usr</td>
<td>[3]</td>
<td>1</td>
<td>Executable in stage 2 non-user mode</td>
</tr>
<tr>
<td>Memory type and shareability</td>
<td>[11:4]</td>
<td>8</td>
<td>Defines the memory attribute</td>
</tr>
<tr>
<td>S2 Level</td>
<td>[13:12]</td>
<td>2</td>
<td>The stage 2 level that gave this translation</td>
</tr>
<tr>
<td>NS (descriptor)</td>
<td>[14]</td>
<td>1</td>
<td>The security state allocated to this memory region</td>
</tr>
<tr>
<td>PA</td>
<td>[46:15]</td>
<td>32</td>
<td>The physical address</td>
</tr>
<tr>
<td>Parity</td>
<td>[47]</td>
<td>1</td>
<td>Parity</td>
</tr>
</tbody>
</table>

### A6.6.5 Walk cache descriptor fields

The following table shows the walk cache descriptor data fields for Tag and Data RAMs.

#### Table A6-10 Walk cache descriptor fields for Tag RAM

<table>
<thead>
<tr>
<th>Field</th>
<th>Bit Position</th>
<th>Width</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid</td>
<td>[0]</td>
<td>1</td>
<td>Indicates that the entry is valid</td>
</tr>
<tr>
<td>NS (walk)</td>
<td>[1]</td>
<td>1</td>
<td>The Security state of the entry fetch</td>
</tr>
<tr>
<td>ASID</td>
<td>[17:2]</td>
<td>16</td>
<td>Address Space Identifier</td>
</tr>
<tr>
<td>VMID</td>
<td>[33:18]</td>
<td>16</td>
<td>Virtual Machine Identifier</td>
</tr>
<tr>
<td>HYP/EL2</td>
<td>[34]</td>
<td>1</td>
<td>Set if the entry was fetched in HYP, EL2, or Virtual Host Extension (VHE) mode.</td>
</tr>
<tr>
<td>EL3</td>
<td>[35]</td>
<td>1</td>
<td>Set if the entry was fetched in AArch64 EL3 mode</td>
</tr>
<tr>
<td>Arch</td>
<td>[38:36]</td>
<td>3</td>
<td>Used to determine how many and which bits of address are used for constructing the physical address of the pagewalk</td>
</tr>
<tr>
<td>Domain</td>
<td>[42:39]</td>
<td>4</td>
<td>Invalid</td>
</tr>
<tr>
<td>Thread ID</td>
<td>[43]</td>
<td>1</td>
<td>Thread ID for the thread that wrote the translation descriptor in Walk cache</td>
</tr>
<tr>
<td>CnP</td>
<td>[44]</td>
<td>1</td>
<td>Common not Private bit</td>
</tr>
<tr>
<td>Address Sign Bit</td>
<td>[45]</td>
<td>1</td>
<td>Address sign bit, VA[48]</td>
</tr>
<tr>
<td>VA</td>
<td>[69:46]</td>
<td>24</td>
<td>Virtual Address sign bit</td>
</tr>
<tr>
<td>S2AP</td>
<td>[71:70]</td>
<td>2</td>
<td>Stage 2 access permission</td>
</tr>
<tr>
<td>S2level</td>
<td>[73:72]</td>
<td>2</td>
<td>The stage 2 level which translates the IPA to PA for the page table entry</td>
</tr>
<tr>
<td>Parity</td>
<td>[81:80]</td>
<td>2</td>
<td>Parity</td>
</tr>
</tbody>
</table>

#### Table A6-11 Walk cache descriptor fields for Data RAM

<table>
<thead>
<tr>
<th>Field</th>
<th>Bit Position</th>
<th>Width</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>APTable</td>
<td>[1:0]</td>
<td>2</td>
<td>Combined ATable bits from stage 1 descriptors up to the last level</td>
</tr>
<tr>
<td>XNTable</td>
<td>[2]</td>
<td>1</td>
<td>Combined XNTable bits from stage 1 descriptors up to the last level</td>
</tr>
</tbody>
</table>
### Table A6-11 Walk cache descriptor fields for Data RAM (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Bit Position</th>
<th>Width</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PXNTable</td>
<td>[3]</td>
<td>1</td>
<td>Combined PXNTable bits from stage 1 descriptors up to the last level</td>
</tr>
<tr>
<td>NSTable</td>
<td>[4]</td>
<td>1</td>
<td>Combined NSTable bits from first and second-level stage 1 tables or NS descriptors (VMSA)</td>
</tr>
<tr>
<td>Attrs</td>
<td>[12:5]</td>
<td>8</td>
<td>Physical address attributes of the final level stage 1 table</td>
</tr>
<tr>
<td>PA</td>
<td>[46:13]</td>
<td>34</td>
<td>Physical address of the stage 1 last translation level page table entry</td>
</tr>
<tr>
<td>Parity</td>
<td>[47]</td>
<td>1</td>
<td>Parity</td>
</tr>
</tbody>
</table>

### A6.6.6 IPA cache descriptor fields

The IPA cache holds mappings from intermediate physical addresses (IPA) to physical addresses. It is only used for translations performed in non-secure EL0/1. It is updated whenever a stage 2 translation is completed, and checked whenever a stage 2 translation is required.

The following table shows the data and tag fields in the IPA cache descriptor.

#### Table A6-12 IPA cache descriptor fields for Tag RAM

<table>
<thead>
<tr>
<th>Fields</th>
<th>Bits</th>
<th>Width</th>
<th>Descriptor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid</td>
<td>[0]</td>
<td>1</td>
<td>Indicates that the entry is valid.</td>
</tr>
<tr>
<td>Entry granule</td>
<td>[2:1]</td>
<td>2</td>
<td>Indicates the entry granule size.</td>
</tr>
<tr>
<td>Thread ID</td>
<td>[3]</td>
<td>1</td>
<td>Indicates Thread ID.</td>
</tr>
<tr>
<td>Size</td>
<td>[8:5]</td>
<td>4</td>
<td>Indicates the S2 page size for this entry.</td>
</tr>
<tr>
<td>DBM</td>
<td>[9]</td>
<td>1</td>
<td>Indicates the DBM.</td>
</tr>
<tr>
<td>Unused</td>
<td>[17:10]</td>
<td>8</td>
<td>Must be set to 0.</td>
</tr>
<tr>
<td>VMID</td>
<td>[33:18]</td>
<td>16</td>
<td>Indicates the virtual machine identifier.</td>
</tr>
<tr>
<td>IPA</td>
<td>[61:34]</td>
<td>28</td>
<td>Unused lower bits, page size dependant, must be set to zero.</td>
</tr>
<tr>
<td>Unused</td>
<td>[79:62]</td>
<td>18</td>
<td>Must be set to 0.</td>
</tr>
<tr>
<td>Parity</td>
<td>[81:80]</td>
<td>2</td>
<td>Parity.</td>
</tr>
</tbody>
</table>

#### Table A6-13 IPA cache descriptor fields for Data RAM

<table>
<thead>
<tr>
<th>Fields</th>
<th>Bits</th>
<th>Width</th>
<th>Descriptor</th>
</tr>
</thead>
<tbody>
<tr>
<td>SH</td>
<td>[1:0]</td>
<td>2</td>
<td>Shareability.</td>
</tr>
<tr>
<td>S2AP</td>
<td>[3:2]</td>
<td>2</td>
<td>Stage 2 access permissions</td>
</tr>
<tr>
<td>XN</td>
<td>[5:4]</td>
<td>2</td>
<td>Controls EL1 and EL0 access permissions.</td>
</tr>
<tr>
<td>PA</td>
<td>[41:10]</td>
<td>32</td>
<td>Physical Address.</td>
</tr>
<tr>
<td>Unused</td>
<td>[46:42]</td>
<td>5</td>
<td>Must be set to 0.</td>
</tr>
<tr>
<td>Parity</td>
<td>[47]</td>
<td>1</td>
<td>Parity.</td>
</tr>
</tbody>
</table>
Chapter A7
Level 2 memory system

This chapter describes the L2 memory system.

It contains the following sections:

• *A7.1 About the L2 memory system* on page A7-90.
• *A7.2 Optional integrated L2 cache* on page A7-91.
• *A7.3 Support for memory types* on page A7-92.
A7.1 About the L2 memory system

The Cortex-A65 L2 memory system is required to interface the Cortex-A65 cores to the L3 memory system.

The L2 cache controller handles requests from the L1 instruction and data caches, and snoop requests from the L3 memory system. The L2 memory system forwards responses from the L3 system to the core, which can then take precise or imprecise aborts, depending on the type of transaction.

The L2 memory subsystem consists of:

- An optional 4-way, set-associative L2 cache with a configurable size of 64KB, 128KB, or 256KB. Cache lines have a fixed length of 64 bytes.
- ECC protection for tag, data, and L2 data buffer RAM structures.

The main features of the L2 memory system are:

- Strictly exclusive with L1 data cache.
- Pseudo-inclusive with L1 instruction cache.
- Private per-core unified L2 cache.
- 44-bit physical address space.
- Physically indexed, physically tagged.
A7.2 Optional integrated L2 cache

Data is allocated to the L2 cache only when evicted from the L1 memory system, not when first fetched from the system.

The exceptions to this rule are:

• If the Read-Allocate hint is set, cacheable reads from the TLB or instruction side are allocated in the L2 cache.
• If the Write-Allocate hint is set when the L1 enters write-streaming mode, cacheable writes are allocated in the L2 until the L2 streaming threshold is reached.
• L2 prefetches issued by the L1 through a PRFM instruction are allocated in the L2 regardless of the Read-Allocate hint.

When non-temporal data is evicted from the L1 memory system, the data is sent directly to L3 and is not allocated in L2.

L2 RAMs are invalidated automatically at reset unless the debug recovery P-Channel state is used.
A7.3 Support for memory types

The Cortex-A65 core simplifies the coherency logic by downgrading some memory types.

- Memory that is marked as both Inner Write-Back Cacheable and Outer Write-Back Cacheable is cached in the L1 data cache and the L2 cache.
- All other memory types are Non-cacheable.

The additional attribute hints are used as follows:

**Allocation hint**
Determines the rules of allocation of newly fetched lines in the system, see *A7.2 Optional integrated L2 cache* on page A7-91.

**Transient hint**
Allocating reads (from TLB or instruction side) and writes in write-streaming mode that have the transient bit set are allocated in L2 cache and marked as most likely to be evicted according to the L2 eviction policy.

Evictions from L1 cache marked as transient are not allocated in L2 cache.

The standard CHI attributes are passed to DSU with no modifications (except for translating architectural attributes to CHI attributes):
- Allocate hint.
- Cacheability (inner and outer are merged together, as the Cortex-A65 core only allocates both inner and outer cacheable memory).
- Shareability.
Chapter A8
Reliability, Availability, and Serviceability (RAS)

This chapter describes the RAS features implemented in the Cortex-A65 core.

It contains the following sections:

• **A8.1 Cache ECC and parity** on page A8-94.
• **A8.2 Cache protection behavior** on page A8-95.
• **A8.3 Uncorrected errors and data poisoning** on page A8-97.
• **A8.4 RAS error types** on page A8-98.
• **A8.5 Error synchronization barrier** on page A8-100.
• **A8.6 Error recording** on page A8-101.
• **A8.7 Error injection** on page A8-102.
A8.1 Cache ECC and parity

The Cortex-A65 core implements the RAS extension to the Armv8-A architecture which provides mechanisms for standardized reporting of the errors generated by cache protection mechanisms.

Core cache protection enables the Cortex-A65 core to detect and correct a 1-bit error in any RAM and detect 2-bit errors in some RAMs.

--- Note ---

For information about SCU-L3 cache protection, see the Arm® DynamIQ™ Shared Unit Technical Reference Manual.

The RAS extension improves the system by reducing unplanned outages:

• Transient errors can be detected and corrected before they cause application or system failure.
• Failing components can be identified and replaced.
• Failure can be predicted ahead of time to allow replacement during planned maintenance.

The severity of a failure can range from minor to catastrophic. In many systems, data or service loss is regarded as more of a minor failure than data corruption, as long as backup data is available.

The RAS extension focuses on errors that are produced from hardware faults, which fall into two main categories:

• Transient faults.
• Persistent faults.

The RAS extension describes data corruption faults, which mostly occur in memories and on data links. RAS concepts can also be used for the management of other types of physical faults found in systems, such as lock-step errors, thermal trip, and mechanical failure. The RAS extension provides a common programmers model and mechanisms for fault handling and error recovery.
A8.2 Cache protection behavior

The core protects against soft errors that result in a RAM bitcell temporarily holding the incorrect value. The Cortex-A65 core writes a new value to the RAM to correct the error. If the error is a hard error that is not corrected by writing to the RAM, for example a physical defect in the RAM, and if there are two or more than two such hard errors in the core, then the core might get into a livelock as it continually detects and then tries to correct the error.

Some RAMs have Single Error Detect (SED) capability, while others have Single Error Correct, Double Error Detect (SECDED) capability. The core can make progress and remain functionally correct when there is transient single bit error in any RAM. If there are multiple single bit errors in different RAMs, or within different protection granules within the same RAM, then the core also remains functionally correct. If there is a double bit error in a single RAM within the same protection granule, then the behavior depends on the RAM:

- For RAMs with SECDED capability listed in the following table, the error is detected and reported as described in error reporting. If the error is in a cache line containing dirty data, then that data might be lost, resulting in data corruption.
- For RAMs with only SED, a double bit error is not detected and therefore might cause data corruption.

If there are three or more bit errors, then depending on the RAM and the position of the errors within the RAM, the errors might be detected or might not be detected.

The Cortex-A65 cache protection support has a minimal performance impact when no errors are present. When an error is detected, the access that caused the error is stalled while the correction takes place. When the correction is complete, the access either continues with the corrected data, or is retried. If the access is retried, it either hits in the cache again with the corrected data, or misses in the cache and refetches the data from a lower level cache or from main memory. The behavior for each RAM is shown in the following table.

<table>
<thead>
<tr>
<th>RAM</th>
<th>Protection type</th>
<th>Protection granule</th>
<th>Correction behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1 instruction cache tag</td>
<td>Parity, SED</td>
<td>34 bits</td>
<td>Both lines in the cache set are invalidated, then the line requested is refetched from L2 or external memory.</td>
</tr>
<tr>
<td>L1 instruction cache data</td>
<td>Parity, SED</td>
<td>32 bits</td>
<td>Both lines in the cache set are invalidated, then the line requested is refetched from L2 or external memory.</td>
</tr>
<tr>
<td>L2 TLB tag</td>
<td>Parity, SED</td>
<td>40 bits</td>
<td>Entry invalidated, new pagewalk started to refetch it.</td>
</tr>
<tr>
<td>L2 TLB data</td>
<td>Parity, SED</td>
<td>47 bits</td>
<td>Entry invalidated, new pagewalk started to refetch it.</td>
</tr>
<tr>
<td>L1 data cache tag</td>
<td>ECC, SECDED</td>
<td>35 bits</td>
<td>Line cleaned and invalidated from L1. SCU duplicate tags are used to get the correct address. Line refetched from L2 or external memory, with single bit errors corrected as part of the eviction.</td>
</tr>
<tr>
<td>L1 data cache data</td>
<td>ECC, SECDED</td>
<td>32 bits</td>
<td>Line cleaned and invalidated from L1, with single bit errors corrected as part of the eviction. Line refetched from L2 or external memory.</td>
</tr>
<tr>
<td>L1 data cache dirty</td>
<td>ECC, SECDED</td>
<td>2 bits</td>
<td>Line cleaned and invalidated from L1, with single bit errors corrected as part of the eviction. Only the dirty bit is protected. The other bits are performance hints, therefore do not cause a functional failure if they are incorrect.</td>
</tr>
<tr>
<td>L2 cache tag</td>
<td>ECC, SECDED</td>
<td>30, 31, or 32 bits depending on the cache size.</td>
<td>Tag rewritten with correct value, access retried. If the error is uncorrectable then the tag is invalidated.</td>
</tr>
</tbody>
</table>
### Table A8-1 Cache protection behavior (continued)

<table>
<thead>
<tr>
<th>RAM</th>
<th>Protection type</th>
<th>Protection granule</th>
<th>Correction behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>L2 cache victim</td>
<td>None</td>
<td>-</td>
<td>The victim RAM is used only as a performance hint. It does not result in a functional failure if the contents are incorrect.</td>
</tr>
<tr>
<td>L2 cache data</td>
<td>ECC, SECDED</td>
<td>64 bits</td>
<td>Data is corrected inline, access might stall for an additional cycle or two while the correction takes place.</td>
</tr>
<tr>
<td>L2 data buffer</td>
<td>ECC, SECDED</td>
<td>72 bits</td>
<td>Data is corrected inline, access might stall for an additional cycle or two while the correction takes place.</td>
</tr>
<tr>
<td>Branch predictor</td>
<td>None</td>
<td>-</td>
<td>The branch predictor RAMs are used only as a performance hint. They do not result in a functional failure if the contents are incorrect.</td>
</tr>
</tbody>
</table>

**Persistent error handling**

A persistent, hard, or stuck-at RAM error is an error that cannot be corrected. Execution on the Cortex-A65 core with instruction, data, and translation table RAMs sourced from non-cacheable memory will not livelock due to any single persistent errors in the:

- L1 instruction cache.
- L1 data cache.
- Main TLB cache.
- L2 cache.
- L3 cache.

Cache coherency is not guaranteed past a detected persistent error.
A8.3 Uncorrected errors and data poisoning

When an error is detected, the correction mechanism is triggered. However, if the error is a 2-bit error in a RAM protected by ECC, then the error is not correctable.

The behavior on an uncorrected error depends on the type of RAM.

**Uncorrected error detected in a data RAM**

When an uncorrected error is detected in a data RAM:
- The chunk of data with the error is marked as poisoned. This poison information is then transferred with the data and stored in the cache if the data is allocated back into a cache. The poisoned data is stored per 64 bits of data, except in the L1 data cache where it is stored per 32 bits of data.
- If the interconnect supports poisoning, then the poison is passed along with the data when the line is evicted from the cluster. No abort is generated when a line is poisoned, as the abort can be deferred until the point when the poisoned data is consumed by a load or instruction fetch.

**Uncorrected error detected in a tag or dirty RAM**

When an uncorrected error is detected in a tag RAM or dirty RAM, either the address or coherency state of the line is not known anymore, and the data cannot be poisoned. In this case, the line is invalidated and an interrupt is generated to notify software that data has potentially been lost.
A8.4 RAS error types

For a standard error record, three error types can be recorded. When a processing element accesses memory or other state, errors might be detected in that memory or state, and corrected, deferred, or signaled to the processing element as a detected error. The component that detects an error is called a node.

Corrected error (CE) An error was detected and corrected. It no longer infects the state of the node and has not been silently propagated. The node continues to operate.

Deferred error (DE) An error was detected, was not corrected, and was deferred. The error is not silently propagated and may be latent in the system. The node continues to operate.

Uncorrected Error (UC) An error was detected and was not corrected or deferred. The error is latent in the system.

Note Uncorrected errors can have subtypes depending on whether the error was produced or consumed at the node.

Errors produced at the node

For uncorrected errors that are produced at the node, the subtypes, in increasing severity, are:

Latent The error has not been propagated. That is, the error was detected but not consumed, and was not recorded as a deferred error.

Signaled The error has not been silently propagated. The error has been or might have been consumed, and was not recorded as a deferred error.

Note The producer cannot know if a consumer has architecturally consumed the error. If it has definitely not been propagated to any consumer, and signaled otherwise, an error might be marked as Latent.

Unrecoverable (UEU) The error has not been silently propagated. The node cannot continue operating.

Uncontainable (UC) The error might have been silently propagated. If the error cannot be isolated, the system must be shut down to avoid catastrophic failure.

Errors consumed at the node

For uncorrected errors that are consumed at the node, the subtypes, in increasing severity, are:

Restartable (UEO) The error has not been silently propagated. The node halts operation because of consuming an error. The node does not rely on the corrupted data so can continue to operate without repairing the error.

Recoverable (UER) The error has not been silently propagated. The node halts operation. To continue, the node relies on consuming the corrupted data. If software can locate and repair the error, the halted operation can continue.

Unrecoverable (UEU) The error has not been silently propagated. The node halts operation and cannot resume from its halted state.

Uncontainable (UC) The error might have been silently propagated. If the error cannot be isolated, the system must be shut down to avoid catastrophic failure.
Error status priority

The highest priority recorded error type is recorded in the Selected Error Record Primary Status Register. For more information, see B2.8 ERR0STATUS, Error Record Primary Status Register on page B2-256.

Related references

B1.43 ERXSTATUS_EL1, Selected Error Record Primary Status Register, EL1 on page B1-189
A8.5 Error synchronization barrier

The Error Synchronization Barrier (ESB) instruction synchronizes unrecoverable errors.

The RAS extension adds the ESB instruction used to synchronize unrecoverable errors. Unrecoverable errors are containable errors consumed by the core and not silently propagated.

The ESB instruction allows efficient isolation of errors:

- The ESB instruction does not wait for completion of accesses that cannot generate an asynchronous external abort. For example, if all external aborts are handled synchronously or it is known that no such accesses are outstanding.
- The ESB instruction does not order accesses and does not guarantee a pipeline flush.

All unrecoverable errors must be synchronized by an ESB instruction. The ESB instruction guarantees the following:

- All unrecoverable errors that are generated before the ESB instruction have pended a System Error Interrupts (SEI) exception.
- If a physical SEI is pended by or was pending before the ESB instruction is executed:
  - If the physical SEI is unmasked at the current Exception level, then it is taken before completion of the ESB instruction.
  - If the physical SEI is masked at the current Exception level, the pending SEI is cleared, the SEI syndrome is recorded in DISR/DISR_EL1, and DISR/DISR_EL1.A is set to 1. This indicates that the SEI was generated before the ESB by instructions that occur in program order.

The ESB instruction also guarantees the following:

- SEIs generated before the ESB instruction are either taken before or at the ESB instruction, or are pended in DISR/DISR_EL1.
- SEIs generated after the ESB are not pended in DISR/DISR_EL1.

This includes unrecoverable errors that are generated by instructions, translation table walks, and instructions fetches on the same core.

Note

DISR_EL1 can only be accessed at EL1 or above. If EL2 is implemented and HCR_EL2.AMO is set to 1, then reads and writes of DISR_EL1 at Non-secure EL1 access VDISR_EL2.

See the following registers:

- B1.34 DISR_EL1, Deferred Interrupt Status Register, EL1 on page B1-177.
- B1.48 HCR_EL2, Hypervisor Configuration Register, EL2 on page B1-194.
- B1.81 VDISR_EL2, Virtual Deferred Interrupt Status Register, EL2 on page B1-238.
A8.6 Error recording

The component that detects an error is called a node. The Cortex-A65 core is a node that interacts with the DynamIQ Shared Unit node. There is one record per node for the errors detected.

For more information on error recording generated by cache protection, see the Arm® Reliability, Availability, and Serviceability (RAS) Specification, Armv8, for the Armv8-A architecture profile. The following points apply specifically to the Cortex-A65 core:

• In the Cortex-A65 core, any error that is detected is reported and recorded in the error record registers:
  — B1.36 ERRSELR_EL1, Error Record Select Register, EL1 on page B1-180
  — B1.37 ERXCTLR_EL1, Selected Error Record Control Register, EL1 on page B1-181
  — B1.38 ERXFR_EL1, Selected Error Record Feature Register, EL1 on page B1-182
  — B1.39 ERXMISC0_EL1, Selected Error Record Miscellaneous Register 0, EL1 on page B1-183
  — B1.40 ERXPFGCDN_EL1, Selected Error Pseudo Fault Generation Count Down Register, EL1 on page B1-184
  — B1.41 ERXPFGCTL_EL1, Selected Error Pseudo Fault Generation Control Register, EL1 on page B1-186
  — B1.42 ERXPFGF_EL1, Selected Pseudo Fault Generation Feature Register, EL1 on page B1-188
  — B1.43 ERXSTATUS_EL1, Selected Error Record Primary Status Register, EL1 on page B1-189
• There are two error records provided, which can be selected with the ERRSELR_EL1 register:
  — Record 0 is private to the core, and is updated on any error in the core RAMs including L1 caches, TLB, and L2 cache.
  — Record 1 records any error in the L3 and snoop filter RAMs and is shared between all cores in the cluster.
• The fault handling interrupt is generated on the nFAULTIRQ[0] pin for L3 and snoop filter errors, or on the nFAULTIRQ[n+1] pin for core n L1 and L2 errors.
A8.7 Error injection

To support testing of error handling software, the Cortex-A65 core can inject errors into the error detection logic.

The following table describes the possible types of errors that the core can encounter and therefore inject.

<table>
<thead>
<tr>
<th>Error type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corrected errors</td>
<td>A CE is generated for a single ECC error on L1 data cache access.</td>
</tr>
<tr>
<td>Deferred errors</td>
<td>A DE is generated for a double ECC error on eviction of a cache line from the L1 to the L2, or as a result of a snoop on the L1.</td>
</tr>
<tr>
<td>Uncontainable errors</td>
<td>A UC is generated for a double ECC error on the L1 TAG RAM following an eviction.</td>
</tr>
<tr>
<td>Latent error</td>
<td>A UEO is generated as a double ECC error on an L1 data read.</td>
</tr>
</tbody>
</table>

Error registers

The following table describes the registers that handle error injection in the Cortex-A65 core.

<table>
<thead>
<tr>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERR0PFGF_EL1</td>
<td>The ERR Pseudo Fault Generation Feature register defines which errors can be injected.</td>
</tr>
<tr>
<td>ERR0PFGCTL_EL1</td>
<td>The ERR Pseudo Fault Generation Control register controls the errors that are injected.</td>
</tr>
<tr>
<td>ERR0PFGCDN_EL1</td>
<td>The Selected Pseudo Fault Generation Count Down register controls the fault injection timing.</td>
</tr>
</tbody>
</table>

--- Note ---

This mechanism simulates the corruption of any RAM but the data is not actually corrupted.

See also:
- B2.7 ERR0PFGF, Error Pseudo Fault Generation Feature Register on page B2-254.
- B2.6 ERR0PFGCTL, Error Pseudo Fault Generation Control Register on page B2-252.
This chapter describes the Cortex-A65 core implementation of the Arm Generic Interrupt Controller (GIC) CPU interface.

It contains the following sections:

- A9.1 About the Generic Interrupt Controller CPU interface on page A9-104.
- A9.2 Bypassing the CPU interface on page A9-105.
A9.1 About the Generic Interrupt Controller CPU interface

The GIC CPU interface, when integrated with an external distributor component, is a resource for supporting and managing interrupts in a cluster system.

The GIC CPU interface hosts registers to mask, identify, and control states of interrupts forwarded to that core. There is a separate GIC CPU interface for each core in the system.

The Cortex-A65 core implements the GIC CPU interface as described in the Arm® Generic Interrupt Controller Architecture Specification. This interfaces with an external GICv3 or GICv4 interrupt distributor component within the system.

Note

This chapter describes only features that are specific to the Cortex-A65 core implementation. Additional information specific to the DSU can be found in Arm® DynamIQ™ Shared Unit Technical Reference Manual.

The GICv4 architecture supports:

- Two security states.
- Interrupt virtualization.
- Software-generated Interrupts (SGIs).
- Message Based Interrupts.
- System register access for the CPU interface.
- Interrupt masking and prioritization.
- Cluster environments, including systems that contain more than eight cores.
- Wake-up events in power management environments.

The GIC includes interrupt grouping functionality that supports:

- Configuring each interrupt to belong to an interrupt group.
- Signaling Group 1 interrupts to the target core using either the IRQ or the FIQ exception request.
- Signaling Group 0 interrupts to the target core using the FIQ exception request only.
- A unified scheme for handling the priority of Group 0 and Group 1 interrupts.

This chapter describes only features that are specific to the Cortex-A65 core implementation.
A9.2 Bypassing the CPU interface

The GIC CPU interface is always implemented within the Cortex-A65 core.

However, you can disable it if you assert the GICDISABLE signal HIGH at reset. If the GIC is enabled, the input pins nVIRQ and nVFIQ must be tied off to HIGH. This is because the internal GIC CPU interface generates the virtual interrupt signals to the cores. The nIRQ and nFIQ signals are controlled by software, therefore there is no requirement to tie them HIGH. If you disable the GIC CPU interface, the input pins nVIRQ and nVFIQ can be driven by an external GIC in the SoC.

If the Cortex-A65 core is not integrated with an external GICv3 or GICv4 distributor component in the system, then you can disable the GIC CPU interface by asserting the GICDISABLE signal HIGH at reset.

GIC system register access generates UNDEFINED instruction exceptions when the GICDISABLE signal is HIGH.
A9 Generic Interrupt Controller CPU interface

A9.2 Bypassing the CPU interface
This chapter describes the Advanced SIMD and floating-point features and registers in the Cortex-A65 core. The unit in charge of handling the Advanced SIMD and floating-point features is also referred to as data engine in this manual.

It contains the following sections:

- **A10.1 About the Advanced SIMD and floating-point support** on page A10-108.
- **A10.2 Accessing the feature identification registers** on page A10-109.
A10.1 About the Advanced SIMD and floating-point support

The Cortex-A65 core supports the Advanced SIMD and scalar floating-point instructions in the A64 instruction set.

The Cortex-A65 floating-point implementation:
• Does not generate floating-point exceptions.
• Implements all scalar operations in hardware with support for all combinations of:
  — Rounding modes.
  — Flush-to-zero.
  — Default Not a Number (NaN) modes.

The Armv8-A architecture does not define a separate version number for its Advanced SIMD and floating-point support in the AArch64 execution state because the instructions are always implicitly present.
### A10.2 Accessing the feature identification registers

Software can identify the Advanced SIMD and floating-point features using the feature identification registers in the AArch64 Execution state only.

You can access the feature identification registers in the AArch64 Execution state using the `MRS` instruction, for example:

```
MRS <Xt>, ID_AA64PFR0_EL1 ; Read ID_AA64PFR0_EL1 into Xt
```

#### Table A10-1 AArch64 Advanced SIMD and scalar floating-point feature identification registers

<table>
<thead>
<tr>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID_AA64ISAR0_EL1</td>
<td>See B1.53 ID_AA64ISAR0_EL1, AArch64 Instruction Set Attribute Register 0, EL1 on page B1-201.</td>
</tr>
<tr>
<td>ID_AA64PFR0_EL1</td>
<td>See B1.58 ID_AA64PFR0_EL1, AArch64 Processor Feature Register 0, EL1 on page B1-209.</td>
</tr>
</tbody>
</table>
A10 Advanced SIMD and floating-point support
A10.2 Accessing the feature identification registers
Part B
Register descriptions
Chapter B1
AArch64 system registers

This chapter describes the system registers in the AArch64 state.

It contains the following sections:

- B1.1 AArch64 registers on page B1-116.
- B1.2 AArch64 architectural system register summary on page B1-117.
- B1.3 AArch64 IMPLEMENTATION DEFINED register summary on page B1-123.
- B1.4 AArch64 registers by functional group on page B1-124.
- B1.5 ACTLR_EL1, Auxiliary Control Register, EL1 on page B1-130.
- B1.6 ACTLR_EL2, Auxiliary Control Register, EL2 on page B1-131.
- B1.7 ACTLR_EL3, Auxiliary Control Register, EL3 on page B1-133.
- B1.8 AFSR0_EL1, Auxiliary Fault Status Register 0, EL1 on page B1-135.
- B1.9 AFSR0_EL2, Auxiliary Fault Status Register 0, EL2 on page B1-136.
- B1.10 AFSR0_EL3, Auxiliary Fault Status Register 0, EL3 on page B1-137.
- B1.11 AFSR1_EL1, Auxiliary Fault Status Register 1, EL1 on page B1-138.
- B1.12 AFSR1_EL2, Auxiliary Fault Status Register 1, EL2 on page B1-139.
- B1.13 AFSR1_EL3, Auxiliary Fault Status Register 1, EL3 on page B1-140.
- B1.14 AIDR_EL1, Auxiliary ID Register, EL1 on page B1-141.
- B1.15 AMAIR_EL1, Auxiliary Memory Attribute Indirection Register, EL1 on page B1-142.
- B1.16 AMAIR_EL2, Auxiliary Memory Attribute Indirection Register, EL2 on page B1-143.
- B1.17 AMAIR_EL3, Auxiliary Memory Attribute Indirection Register, EL3 on page B1-144.
- B1.18 CCSIDR_EL1, Cache Size ID Register, EL1 on page B1-145.
- B1.19 CLIDR_EL1, Cache Level ID Register, EL1 on page B1-147.
- B1.20 CPACR_EL1, Architectural Feature Access Control Register, EL1 on page B1-149.
- B1.21 CPTR_EL2, Architectural Feature Trap Register, EL2 on page B1-150.
- B1.23 CPUACTLR_EL1, CPU Auxiliary Control Register, EL1 on page B1-154.
B.1.24 CPUCFR_EL1, CPU Configuration Register, EL1 on page B1-156.
B.1.25 CPUECTLR_EL1, CPU Extended Control Register, EL1 on page B1-158.
B.1.26 CPUPCR_EL3, CPU Private Control Register, EL3 on page B1-162.
B.1.27 CPUPMR_EL3, CPU Private Mask Register, EL3 on page B1-164.
B.1.28 CPUPOR_EL3, CPU Private Operation Register, EL3 on page B1-166.
B.1.29 CPUPSLR_EL3, CPU Private Selection Register, EL3 on page B1-168.
B.1.30 CPUPWRCTLR_EL1, Power Control Register, EL1 on page B1-170.
B.1.31 CSSEL_EL1, Cache Size Selection Register, EL1 on page B1-173.
B.1.32 CTR_EL0, Cache Type Register, EL0 on page B1-174.
B.1.33 DCZID_EL0, Data Cache Zero ID Register, EL0 on page B1-176.
B.1.34 DISR_EL1, Deferred Interrupt Status Register, EL1 on page B1-177.
B.1.35 ERRIDR_EL1, Error ID Register, EL1 on page B1-179.
B.1.36 ERRSELR_EL1, Error Record Select Register, EL1 on page B1-180.
B.1.37 ERXCTLR_EL1, Selected Error Record Control Register, EL1 on page B1-181.
B.1.38 ERXFR_EL1, Selected Error Record Feature Register, EL1 on page B1-182.
B.1.39 ERXMISC0_EL1, Selected Error Record Miscellaneous Register 0, EL1 on page B1-183.
B.1.40 ERXPFGCDN_EL1, Selected Error Pseudo Fault Generation Count Down Register, EL1 on page B1-184.
B.1.41 ERXPFGCTL_EL1, Selected Error Pseudo Fault Generation Control Register, EL1 on page B1-186.
B.1.42 ERXPFGF_EL1, Selected Pseudo Fault Generation Feature Register, EL1 on page B1-188.
B.1.43 ERXSTATUS_EL1, Selected Error Record Primary Status Register, EL1 on page B1-189.
B.1.44 ESR_EL1, Exception Syndrome Register, EL1 on page B1-190.
B.1.45 ESR_EL2, Exception Syndrome Register, EL2 on page B1-191.
B.1.46 ESR_EL3, Exception Syndrome Register, EL3 on page B1-192.
B.1.47 HACR_EL2, Hyp Auxiliary Configuration Register, EL2 on page B1-193.
B.1.48 HCR_EL2, Hypervisor Configuration Register, EL2 on page B1-194.
B.1.49 ID_AA64AFR0_EL1, AArch64 Auxiliary Feature Register 0 on page B1-196.
B.1.50 ID_AA64AFR1_EL1, AArch64 Auxiliary Feature Register 1 on page B1-197.
B.1.51 ID_AA64DFR0_EL1, AArch64 Debug Feature Register 0, EL1 on page B1-198.
B.1.52 ID_AA64DFR1_EL1, AArch64 Debug Feature Register 1, EL1 on page B1-200.
B.1.53 ID_AA64ISAR0_EL1, AArch64 Instruction Set Attribute Register 0, EL1 on page B1-201.
B.1.54 ID_AA64ISAR1_EL1, AArch64 Instruction Set Attribute Register 1, EL1 on page B1-203.
B.1.55 ID_AA64MMFR0_EL1, AArch64 Memory Model Feature Register 0, EL1 on page B1-204.
B.1.56 ID_AA64MMFR1_EL1, AArch64 Memory Model Feature Register 1, EL1 on page B1-206.
B.1.57 ID_AA64MMFR2_EL1, AArch64 Memory Model Feature Register 2, EL1 on page B1-208.
B.1.58 ID_AA64PFRO_EL1, AArch64 Processor Feature Register 0, EL1 on page B1-209.
B.1.59 ID_AA64PFR1_EL1, AArch64 Processor Feature Register 1, EL1 on page B1-211.
B.1.60 LORC_EL1, LORegion Control Register, EL1 on page B1-212.
B.1.61 LORID_EL1, LORegion ID Register, EL1 on page B1-213.
B.1.62 LORN_EL1, LORegion Number Register, EL1 on page B1-214.
B.1.63 MDCR_EL3, Monitor Debug Configuration Register, EL3 on page B1-215.
B.1.64 MIDR_EL1, Main ID Register, EL1 on page B1-217.
B.1.65 MPIDR_EL1, Multiprocessor Affinity Register, EL1 on page B1-218.
B.1.66 PAR_EL1, Physical Address Register, EL1 on page B1-220.
B.1.67 REVIDR_EL1, Revision ID Register, EL1 on page B1-221.
B.1.68 RMR_EL3, Reset Management Register on page B1-222.
B.1.69 RVBAR_EL3, Reset Vector Base Address Register, EL3 on page B1-223.
B.1.70 SCTLR_EL1, System Control Register, EL1 on page B1-224.
B.1.71 SCTLR_EL2, System Control Register, EL2 on page B1-226.
B.1.72 SCTLR_EL3, System Control Register, EL3 on page B1-227.
B.1.73 TCR_EL1, Translation Control Register, EL1 on page B1-229.
B.1.74 TCR_EL2, Translation Control Register, EL2 on page B1-230.
B.1.75 TCR_EL3, Translation Control Register, EL3 on page B1-231.
B.1.76 TTBR0_EL1, Translation Table Base Register 0, EL1 on page B1-233.
B.1.77 TTBR0_EL2, Translation Table Base Register 0, EL2 on page B1-234.
• B1.78 TTBR0_EL3, Translation Table Base Register 0, EL3 on page B1-235.
• B1.79 TTBR1_EL1, Translation Table Base Register 1, EL1 on page B1-236.
• B1.80 TTBR1_EL2, Translation Table Base Register 1, EL2 on page B1-237.
• B1.81 VDISR_EL2, Virtual Deferred Interrupt Status Register, EL2 on page B1-238.
• B1.82 VSESR_EL2, Virtual SError Exception Syndrome Register on page B1-239.
• B1.83 VTCR_EL2, Virtualization Translation Control Register, EL2 on page B1-240.
• B1.84 VTTBR_EL2, Virtualization Translation Table Base Register, EL2 on page B1-241.
B1.1 AArch64 registers

This chapter provides information about the AArch64 system registers with IMPLEMENTATION DEFINED bit fields and IMPLEMENTATION DEFINED registers associated with the core.

The chapter provides IMPLEMENTATION SPECIFIC information, for a complete description of the registers, see the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.

As a Simultaneously MultiThreaded (SMT) core, the Cortex-A65 core supports two execution threads on each core. Each thread is a separate architectural processing element (PE), and so has a complete copy of the architectural state. Therefore each thread accesses an identical set of the architectural registers.

The chapter is presented as follows:

AArch64 architectural system register summary

This section identifies the AArch64 architectural system registers implemented in the Cortex-A65 core that have IMPLEMENTATION DEFINED bit fields. The register descriptions for these registers only contain information about the IMPLEMENTATION DEFINED bits.

AArch64 IMPLEMENTATION DEFINED register summary

This section identifies the AArch64 architectural registers implemented in the Cortex-A65 core that are IMPLEMENTATION DEFINED.

AArch64 registers by functional group

This section groups the IMPLEMENTATION DEFINED registers and architectural system registers with IMPLEMENTATION DEFINED bit fields, as identified previously, by function. It also provides reset details for key register types.

Register descriptions

The remainder of the chapter provides register descriptions of the IMPLEMENTATION DEFINED registers and architectural system registers with IMPLEMENTATION DEFINED bit fields, as identified previously. These are listed in alphabetic order.
B1.2  AArch64 architectural system register summary

This section describes the AArch64 architectural system registers implemented in the Cortex-A65 core. The section contains two tables:

**Registers with implementation defined bit fields**

This table identifies the architecturally defined registers in Cortex-A65 that have implementation defined bit fields. The register descriptions for these registers only contain information about the implementation defined bits.

See Table B1-1  Registers with implementation defined bit fields on page B1-117.

**Other architecturally defined registers**

This table identifies the other architecturally defined registers that are implemented in the Cortex-A65 core. These registers are described in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

See Other architecturally defined registers on page B1-120.

<table>
<thead>
<tr>
<th>Name</th>
<th>Op0</th>
<th>CRn</th>
<th>Op1</th>
<th>CRm</th>
<th>Op2</th>
<th>Width</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTLR_EL1</td>
<td>3</td>
<td>c1</td>
<td>0</td>
<td>c0</td>
<td>1</td>
<td>64</td>
<td>B1.5 ACTLR_EL1, Auxiliary Control Register, EL1 on page B1-130</td>
</tr>
<tr>
<td>ACTLR_EL2</td>
<td>3</td>
<td>c1</td>
<td>4</td>
<td>c0</td>
<td>1</td>
<td>64</td>
<td>B1.6 ACTLR_EL2, Auxiliary Control Register, EL2 on page B1-131</td>
</tr>
<tr>
<td>ACTLR_EL3</td>
<td>3</td>
<td>c1</td>
<td>6</td>
<td>c0</td>
<td>1</td>
<td>64</td>
<td>B1.7 ACTLR_EL3, Auxiliary Control Register, EL3 on page B1-133</td>
</tr>
<tr>
<td>AFSR0_EL1</td>
<td>3</td>
<td>c5</td>
<td>0</td>
<td>c1</td>
<td>0</td>
<td>32</td>
<td>B1.8 AFSR0_EL1, Auxiliary Fault Status Register 0, EL1 on page B1-135</td>
</tr>
<tr>
<td>AFSR0_EL2</td>
<td>3</td>
<td>c5</td>
<td>4</td>
<td>c1</td>
<td>0</td>
<td>32</td>
<td>B1.9 AFSR0_EL2, Auxiliary Fault Status Register 0, EL2 on page B1-136</td>
</tr>
<tr>
<td>AFSR0_EL3</td>
<td>3</td>
<td>c5</td>
<td>6</td>
<td>c1</td>
<td>0</td>
<td>32</td>
<td>B1.10 AFSR0_EL3, Auxiliary Fault Status Register 0, EL3 on page B1-137</td>
</tr>
<tr>
<td>AFSR1_EL1</td>
<td>3</td>
<td>c5</td>
<td>0</td>
<td>c1</td>
<td>1</td>
<td>32</td>
<td>B1.11 AFSR1_EL1, Auxiliary Fault Status Register 1, EL1 on page B1-138</td>
</tr>
<tr>
<td>AFSR1_EL2</td>
<td>3</td>
<td>c5</td>
<td>4</td>
<td>c1</td>
<td>1</td>
<td>32</td>
<td>B1.12 AFSR1_EL2, Auxiliary Fault Status Register 1, EL2 on page B1-139</td>
</tr>
<tr>
<td>AFSR1_EL3</td>
<td>3</td>
<td>c5</td>
<td>6</td>
<td>c1</td>
<td>1</td>
<td>32</td>
<td>B1.13 AFSR1_EL3, Auxiliary Fault Status Register 1, EL3 on page B1-140</td>
</tr>
<tr>
<td>AIDR_EL1</td>
<td>3</td>
<td>c0</td>
<td>1</td>
<td>c0</td>
<td>7</td>
<td>32</td>
<td>B1.14 AIDR_EL1, Auxiliary ID Register, EL1 on page B1-141</td>
</tr>
<tr>
<td>AMAIR_EL1</td>
<td>3</td>
<td>c10</td>
<td>0</td>
<td>c3</td>
<td>0</td>
<td>64</td>
<td>B1.15 AMAIR_EL1, Auxiliary Memory Attribute Indirection Register, EL1 on page B1-142</td>
</tr>
<tr>
<td>AMAIR_EL2</td>
<td>3</td>
<td>c10</td>
<td>4</td>
<td>c3</td>
<td>0</td>
<td>64</td>
<td>B1.16 AMAIR_EL2, Auxiliary Memory Attribute Indirection Register, EL2 on page B1-143</td>
</tr>
<tr>
<td>AMAIR_EL3</td>
<td>3</td>
<td>c10</td>
<td>6</td>
<td>c3</td>
<td>0</td>
<td>64</td>
<td>B1.17 AMAIR_EL3, Auxiliary Memory Attribute Indirection Register, EL3 on page B1-144</td>
</tr>
<tr>
<td>CCSIDR_EL1</td>
<td>3</td>
<td>c0</td>
<td>1</td>
<td>c0</td>
<td>0</td>
<td>32</td>
<td>B1.18 CCSIDR_EL1, Cache Size ID Register, EL1 on page B1-145</td>
</tr>
<tr>
<td>Name</td>
<td>Op0</td>
<td>CRn</td>
<td>Op1</td>
<td>CRm</td>
<td>Op2</td>
<td>Width</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CLIDR_EL1</td>
<td>3</td>
<td>c0</td>
<td>1</td>
<td>c0</td>
<td>1</td>
<td>64</td>
<td>B1.19 CLIDR_EL1, Cache Level ID Register, EL1 on page B1-147</td>
</tr>
<tr>
<td>CPACR_EL1</td>
<td>3</td>
<td>c1</td>
<td>0</td>
<td>c0</td>
<td>2</td>
<td>32</td>
<td>B1.20 CPACR_EL1, Architectural Feature Access Control Register, EL1 on page B1-149</td>
</tr>
<tr>
<td>CPTR_EL2</td>
<td>3</td>
<td>c1</td>
<td>4</td>
<td>c1</td>
<td>2</td>
<td>32</td>
<td>B1.21 CPTR_EL2, Architectural Feature Trap Register, EL2 on page B1-150</td>
</tr>
<tr>
<td>CPTR_EL3</td>
<td>3</td>
<td>c1</td>
<td>6</td>
<td>c1</td>
<td>2</td>
<td>32</td>
<td>B1.22 CPTR_EL3, Architectural Feature Trap Register, EL3 on page B1-153</td>
</tr>
<tr>
<td>CSSELR_EL1</td>
<td>3</td>
<td>c0</td>
<td>2</td>
<td>c0</td>
<td>0</td>
<td>32</td>
<td>B1.31 CSSELR_EL1, Cache Size Selection Register, EL1 on page B1-173</td>
</tr>
<tr>
<td>CTR_EL0</td>
<td>3</td>
<td>c0</td>
<td>3</td>
<td>c0</td>
<td>1</td>
<td>32</td>
<td>B1.32 CTR_EL0, Cache Type Register, EL0 on page B1-174</td>
</tr>
<tr>
<td>DISR_EL1</td>
<td>3</td>
<td>c12</td>
<td>0</td>
<td>c1</td>
<td>1</td>
<td>64</td>
<td>B1.34 DISR_EL1, Deferred Interrupt Status Register, EL1 on page B1-177</td>
</tr>
<tr>
<td>ERRIDR_EL1</td>
<td>3</td>
<td>c5</td>
<td>0</td>
<td>c3</td>
<td>0</td>
<td>32</td>
<td>B1.35 ERRIDR_EL1, Error ID Register, EL1 on page B1-179</td>
</tr>
<tr>
<td>ERRSELR_EL1</td>
<td>3</td>
<td>c5</td>
<td>0</td>
<td>c3</td>
<td>1</td>
<td>32</td>
<td>B1.36 ERRSELR_EL1, Error Record Select Register, EL1 on page B1-180</td>
</tr>
<tr>
<td>ERXCTLR_EL1</td>
<td>3</td>
<td>c5</td>
<td>0</td>
<td>c4</td>
<td>1</td>
<td>64</td>
<td>B1.37 ERXCTLR_EL1, Selected Error Record Control Register, EL1 on page B1-181</td>
</tr>
<tr>
<td>ERXFR_EL1</td>
<td>3</td>
<td>c5</td>
<td>0</td>
<td>c4</td>
<td>0</td>
<td>64</td>
<td>B1.38 ERXFR_EL1, Selected Error Record Feature Register, EL1 on page B1-182</td>
</tr>
<tr>
<td>ERXMISC0_EL1</td>
<td>3</td>
<td>c5</td>
<td>0</td>
<td>c5</td>
<td>0</td>
<td>64</td>
<td>B1.39 ERXMISC0_EL1, Selected Error Record Miscellaneous Register 0, EL1 on page B1-183</td>
</tr>
<tr>
<td>ERXSTATUS_EL1</td>
<td>3</td>
<td>c5</td>
<td>0</td>
<td>c4</td>
<td>2</td>
<td>32</td>
<td>B1.43 ERXSTATUS_EL1, Selected Error Record Primary Status Register, EL1 on page B1-189</td>
</tr>
<tr>
<td>ESR_EL1</td>
<td>3</td>
<td>c5</td>
<td>0</td>
<td>c2</td>
<td>0</td>
<td>32</td>
<td>B1.44 ESR_EL1, Exception Syndrome Register, EL1 on page B1-190</td>
</tr>
<tr>
<td>ESR_EL2</td>
<td>3</td>
<td>c5</td>
<td>4</td>
<td>c2</td>
<td>0</td>
<td>32</td>
<td>B1.45 ESR_EL2, Exception Syndrome Register, EL2 on page B1-191</td>
</tr>
<tr>
<td>ESR_EL3</td>
<td>3</td>
<td>c5</td>
<td>6</td>
<td>c2</td>
<td>0</td>
<td>32</td>
<td>B1.46 ESR_EL3, Exception Syndrome Register, EL3 on page B1-192</td>
</tr>
<tr>
<td>HACR_EL2</td>
<td>3</td>
<td>c1</td>
<td>4</td>
<td>c1</td>
<td>7</td>
<td>32</td>
<td>B1.47 HACR_EL2, Hyp Auxiliary Configuration Register, EL2 on page B1-193</td>
</tr>
<tr>
<td>HCR_EL2</td>
<td>3</td>
<td>c1</td>
<td>4</td>
<td>c1</td>
<td>0</td>
<td>64</td>
<td>B1.48 HCR_EL2, Hypervisor Configuration Register, EL2 on page B1-194</td>
</tr>
<tr>
<td>ID_AA64DFR0_EL1</td>
<td>3</td>
<td>c0</td>
<td>0</td>
<td>c5</td>
<td>0</td>
<td>64</td>
<td>B1.51 ID_AA64DFR0_EL1, AArch64 Debug Feature Register 0, EL1 on page B1-198</td>
</tr>
<tr>
<td>ID_AA64ISAR0_EL1</td>
<td>3</td>
<td>c0</td>
<td>0</td>
<td>c6</td>
<td>0</td>
<td>64</td>
<td>B1.53 ID_AA64ISAR0_EL1, AArch64 Instruction Set Attribute Register 0, EL1 on page B1-201</td>
</tr>
<tr>
<td>ID_AA64ISAR1_EL1</td>
<td>3</td>
<td>c0</td>
<td>0</td>
<td>c6</td>
<td>1</td>
<td>64</td>
<td>B1.54 ID_AA64ISAR1_EL1, AArch64 Instruction Set Attribute Register 1, EL1 on page B1-203</td>
</tr>
</tbody>
</table>
Table B1-1  Registers with implementation defined bit fields (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Op0</th>
<th>CRn</th>
<th>Op1</th>
<th>CRm</th>
<th>Op2</th>
<th>Width</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID_AA64MMFR0_EL1</td>
<td>3</td>
<td>c0</td>
<td>0</td>
<td>c7</td>
<td>0</td>
<td>64</td>
<td>B1.55 ID_AA64MMFR0_EL1, AArch64 Memory Model Feature Register 0, EL1 on page B1-204</td>
</tr>
<tr>
<td>ID_AA64MMFR1_EL1</td>
<td>3</td>
<td>c0</td>
<td>0</td>
<td>c7</td>
<td>1</td>
<td>64</td>
<td>B1.56 ID_AA64MMFR1_EL1, AArch64 Memory Model Feature Register 1, EL1 on page B1-206</td>
</tr>
<tr>
<td>ID_AA64MMFR2_EL1</td>
<td>3</td>
<td>c0</td>
<td>0</td>
<td>c7</td>
<td>2</td>
<td>64</td>
<td>B1.57 ID_AA64MMFR2_EL1, AArch64 Memory Model Feature Register 2, EL1 on page B1-208</td>
</tr>
<tr>
<td>ID_AA64PFR0_EL1</td>
<td>3</td>
<td>c0</td>
<td>0</td>
<td>c4</td>
<td>0</td>
<td>64</td>
<td>B1.58 ID_AA64PFR0_EL1, AArch64 Processor Feature Register 0, EL1 on page B1-209</td>
</tr>
<tr>
<td>LORC_EL1</td>
<td>3</td>
<td>c10</td>
<td>0</td>
<td>c4</td>
<td>3</td>
<td>64</td>
<td>B1.60 LORC_EL1, LORegion Control Register, EL1 on page B1-212</td>
</tr>
<tr>
<td>LORID_EL1</td>
<td>3</td>
<td>c10</td>
<td>0</td>
<td>c4</td>
<td>7</td>
<td>64</td>
<td>B1.61 LORID_EL1, LORegion ID Register, EL1 on page B1-213</td>
</tr>
<tr>
<td>LORN_EL1</td>
<td>3</td>
<td>c10</td>
<td>0</td>
<td>c4</td>
<td>2</td>
<td>64</td>
<td>B1.62 LORN_EL1, LORegion Number Register, EL1 on page B1-214</td>
</tr>
<tr>
<td>MDCR_EL3</td>
<td>3</td>
<td>c1</td>
<td>6</td>
<td>c3</td>
<td>1</td>
<td>32</td>
<td>B1.63 MDCR_EL3, Monitor Debug Configuration Register, EL3 on page B1-215</td>
</tr>
<tr>
<td>MIDR_EL1</td>
<td>3</td>
<td>c0</td>
<td>0</td>
<td>c0</td>
<td>0</td>
<td>32</td>
<td>B1.64 MIDR_EL1, Main ID Register, EL1 on page B1-217</td>
</tr>
<tr>
<td>MPIDR_EL1</td>
<td>3</td>
<td>c0</td>
<td>0</td>
<td>c0</td>
<td>5</td>
<td>64</td>
<td>B1.65 MPIDR_EL1, Multiprocessor Affinity Register, EL1 on page B1-218</td>
</tr>
<tr>
<td>PAR_EL1</td>
<td>3</td>
<td>c7</td>
<td>0</td>
<td>c4</td>
<td>0</td>
<td>64</td>
<td>B1.66 PAR_EL1, Physical Address Register, EL1 on page B1-220</td>
</tr>
<tr>
<td>REVIDR_EL1</td>
<td>3</td>
<td>c0</td>
<td>0</td>
<td>c0</td>
<td>6</td>
<td>32</td>
<td>B1.67 REVIDR_EL1, Revision ID Register, EL1 on page B1-221</td>
</tr>
<tr>
<td>RVBAR_EL3</td>
<td>3</td>
<td>c12</td>
<td>6</td>
<td>c0</td>
<td>1</td>
<td>64</td>
<td>B1.69 RVBAR_EL3, Reset Vector Base Address Register, EL3 on page B1-223</td>
</tr>
<tr>
<td>SCTLR_EL1</td>
<td>3</td>
<td>c1</td>
<td>0</td>
<td>c0</td>
<td>0</td>
<td>32</td>
<td>B1.70 SCTLR_EL1, System Control Register, EL1 on page B1-224</td>
</tr>
<tr>
<td>SCTLR_EL2</td>
<td>3</td>
<td>c1</td>
<td>4</td>
<td>c0</td>
<td>0</td>
<td>32</td>
<td>B1.71 SCTLR_EL2, System Control Register, EL2 on page B1-226</td>
</tr>
<tr>
<td>SCTLR_EL12</td>
<td>3</td>
<td>c1</td>
<td>5</td>
<td>c0</td>
<td>0</td>
<td>32</td>
<td>B1.70 SCTLR_EL1, System Control Register, EL1 on page B1-224</td>
</tr>
<tr>
<td>SCTLR_EL3</td>
<td>3</td>
<td>c1</td>
<td>6</td>
<td>c0</td>
<td>0</td>
<td>32</td>
<td>B1.72 SCTLR_EL3, System Control Register, EL3 on page B1-227</td>
</tr>
<tr>
<td>TCR_EL1</td>
<td>3</td>
<td>c2</td>
<td>0</td>
<td>c0</td>
<td>2</td>
<td>64</td>
<td>B1.73 TCR_EL1, Translation Control Register, EL1 on page B1-229</td>
</tr>
<tr>
<td>TCR_EL2</td>
<td>3</td>
<td>c2</td>
<td>4</td>
<td>c0</td>
<td>2</td>
<td>64</td>
<td>B1.74 TCR_EL2, Translation Control Register, EL2 on page B1-230</td>
</tr>
<tr>
<td>TCR_EL3</td>
<td>3</td>
<td>c2</td>
<td>6</td>
<td>c0</td>
<td>2</td>
<td>64</td>
<td>B1.75 TCR_EL3, Translation Control Register, EL3 on page B1-231</td>
</tr>
<tr>
<td>TTBR0_EL1</td>
<td>3</td>
<td>c2</td>
<td>0</td>
<td>c0</td>
<td>0</td>
<td>64</td>
<td>B1.76 TTBR0_EL1, Translation Table Base Register 0, EL1 on page B1-233</td>
</tr>
<tr>
<td>TTBR0_EL2</td>
<td>3</td>
<td>c2</td>
<td>4</td>
<td>c0</td>
<td>0</td>
<td>64</td>
<td>B1.77 TTBR0_EL2, Translation Table Base Register 0, EL2 on page B1-234</td>
</tr>
</tbody>
</table>
### Table B1-1 Registers with implementation defined bit fields (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Op0</th>
<th>CRn</th>
<th>Op1</th>
<th>CRm</th>
<th>Op2</th>
<th>Width</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TTBR0_EL3</td>
<td>3</td>
<td>c2</td>
<td>6</td>
<td>c0</td>
<td>0</td>
<td>64</td>
<td>\textit{B1.78 TTBR0_EL3, Translation Table Base Register 0, EL3} on page B1-235</td>
</tr>
<tr>
<td>TTBR1_EL1</td>
<td>3</td>
<td>c2</td>
<td>0</td>
<td>c0</td>
<td>1</td>
<td>64</td>
<td>\textit{B1.79 TTBR1_EL1, Translation Table Base Register 1, EL1} on page B1-236</td>
</tr>
<tr>
<td>TTBR1_EL2</td>
<td>3</td>
<td>c2</td>
<td>4</td>
<td>c0</td>
<td>1</td>
<td>64</td>
<td>\textit{B1.80 TTBR1_EL2, Translation Table Base Register 1, EL2} on page B1-237</td>
</tr>
<tr>
<td>VDISR_EL2</td>
<td>3</td>
<td>c12</td>
<td>4</td>
<td>c1</td>
<td>1</td>
<td>64</td>
<td>\textit{B1.81 VDISR_EL2, Virtual Deferred Interrupt Status Register, EL2} on page B1-238</td>
</tr>
<tr>
<td>VSESR_EL2</td>
<td>3</td>
<td>c5</td>
<td>4</td>
<td>c2</td>
<td>3</td>
<td>64</td>
<td>\textit{B1.82 VSESR_EL2, Virtual SError Exception Syndrome Register} on page B1-239</td>
</tr>
<tr>
<td>VTCR_EL2</td>
<td>3</td>
<td>c2</td>
<td>4</td>
<td>c1</td>
<td>2</td>
<td>32</td>
<td>\textit{B1.83 VTCR_EL2, Virtualization Translation Control Register, EL2} on page B1-240</td>
</tr>
<tr>
<td>VTTBR_EL2</td>
<td>3</td>
<td>c2</td>
<td>4</td>
<td>c1</td>
<td>0</td>
<td>64</td>
<td>\textit{B1.84 VTTBR_EL2, Virtualization Translation Table Base Register, EL2} on page B1-241</td>
</tr>
</tbody>
</table>

### Table B1-2 Other architecturally defined registers

<table>
<thead>
<tr>
<th>Name</th>
<th>Op0</th>
<th>CRn</th>
<th>Op1</th>
<th>CRm</th>
<th>Op2</th>
<th>Width</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFSR0_EL12</td>
<td>3</td>
<td>c5</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>32</td>
<td>Auxiliary Fault Status Register 0</td>
</tr>
<tr>
<td>AFSR1_EL12</td>
<td>3</td>
<td>c5</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>32</td>
<td>Auxiliary Fault Status Register 1</td>
</tr>
<tr>
<td>AMAIR_EL12</td>
<td>3</td>
<td>c10</td>
<td>5</td>
<td>c3</td>
<td>0</td>
<td>64</td>
<td>Auxiliary Memory Attribute Indirection Register</td>
</tr>
<tr>
<td>CNTFRQ_EL0</td>
<td>3</td>
<td>c14</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>32</td>
<td>Counter-timer Frequency register</td>
</tr>
<tr>
<td>CNTHCTL_EL2</td>
<td>3</td>
<td>c14</td>
<td>4</td>
<td>c1</td>
<td>0</td>
<td>32</td>
<td>Counter-timer Hypervisor Control register</td>
</tr>
<tr>
<td>CNTHP_CTL_EL2</td>
<td>3</td>
<td>c14</td>
<td>4</td>
<td>c2</td>
<td>1</td>
<td>32</td>
<td>Counter-timer Hypervisor Physical Timer Control register</td>
</tr>
<tr>
<td>CNTHP_CVAL_EL2</td>
<td>3</td>
<td>c14</td>
<td>4</td>
<td>c2</td>
<td>2</td>
<td>64</td>
<td>Counter-timer Hyp Physical CompareValue register</td>
</tr>
<tr>
<td>CNTHP_TVAL_EL2</td>
<td>3</td>
<td>c14</td>
<td>4</td>
<td>c2</td>
<td>0</td>
<td>32</td>
<td>Counter-timer Hyp Physical Timer TimerValue register</td>
</tr>
<tr>
<td>CNTHV_CTL_EL2</td>
<td>3</td>
<td>c14</td>
<td>4</td>
<td>c3</td>
<td>1</td>
<td>32</td>
<td>Counter-timer Virtual Timer Control register</td>
</tr>
<tr>
<td>CNTHV_CVAL_EL2</td>
<td>3</td>
<td>c14</td>
<td>4</td>
<td>c3</td>
<td>2</td>
<td>64</td>
<td>Counter-timer Virtual Timer CompareValue register</td>
</tr>
<tr>
<td>CNTHV_TVAL_EL2</td>
<td>3</td>
<td>c14</td>
<td>4</td>
<td>c3</td>
<td>0</td>
<td>32</td>
<td>Counter-timer Virtual Timer TimerValue register</td>
</tr>
<tr>
<td>CNTKCTL_EL1</td>
<td>3</td>
<td>c14</td>
<td>0</td>
<td>c1</td>
<td>0</td>
<td>32</td>
<td>Counter-timer Kernel Control register</td>
</tr>
<tr>
<td>CNTKCTL_EL12</td>
<td>3</td>
<td>c14</td>
<td>5</td>
<td>c1</td>
<td>0</td>
<td>32</td>
<td>Counter-timer Kernel Control register</td>
</tr>
<tr>
<td>CNTP_CTL_EL0</td>
<td>3</td>
<td>c14</td>
<td>3</td>
<td>c2</td>
<td>1</td>
<td>32</td>
<td>Counter-timer Physical Timer Control register</td>
</tr>
<tr>
<td>CNTP_CTL_EL02</td>
<td>3</td>
<td>c14</td>
<td>5</td>
<td>c2</td>
<td>1</td>
<td>32</td>
<td>Counter-timer Physical Timer Control register</td>
</tr>
<tr>
<td>CNTP_CVAL_EL0</td>
<td>3</td>
<td>c14</td>
<td>3</td>
<td>c2</td>
<td>2</td>
<td>64</td>
<td>Counter-timer Physical Timer CompareValue register</td>
</tr>
<tr>
<td>CNTP_CVAL_EL02</td>
<td>3</td>
<td>c14</td>
<td>5</td>
<td>c2</td>
<td>2</td>
<td>64</td>
<td>Counter-timer Physical Timer CompareValue register</td>
</tr>
<tr>
<td>CNTP_TVAL_EL0</td>
<td>3</td>
<td>c14</td>
<td>3</td>
<td>c2</td>
<td>0</td>
<td>32</td>
<td>Counter-timer Physical Timer TimerValue register</td>
</tr>
<tr>
<td>CNTP_TVAL_EL02</td>
<td>3</td>
<td>c14</td>
<td>5</td>
<td>c2</td>
<td>0</td>
<td>32</td>
<td>Counter-timer Physical Timer TimerValue register</td>
</tr>
<tr>
<td>CNTPCT_EL0</td>
<td>3</td>
<td>c14</td>
<td>3</td>
<td>c0</td>
<td>1</td>
<td>64</td>
<td>Counter-timer Physical Count register</td>
</tr>
<tr>
<td>Name</td>
<td>Op0</td>
<td>CRn</td>
<td>Op1</td>
<td>CRm</td>
<td>Op2</td>
<td>Width</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-------</td>
<td>-------------------------------------------------</td>
</tr>
<tr>
<td>CNTPS_CTL_EL1</td>
<td>3</td>
<td>c14</td>
<td>7</td>
<td>c2</td>
<td>1</td>
<td>32</td>
<td>Counter-timer Physical Secure Timer Control register</td>
</tr>
<tr>
<td>CNTPS_CVAL_EL1</td>
<td>3</td>
<td>c14</td>
<td>7</td>
<td>c2</td>
<td>2</td>
<td>64</td>
<td>Counter-timer Physical Secure Timer CompareValue register</td>
</tr>
<tr>
<td>CNTPS_TVAL_EL1</td>
<td>3</td>
<td>c14</td>
<td>7</td>
<td>c2</td>
<td>0</td>
<td>32</td>
<td>Counter-timer Physical Secure Timer TimerValue register</td>
</tr>
<tr>
<td>CNTV_CTL_EL0</td>
<td>3</td>
<td>c14</td>
<td>3</td>
<td>c3</td>
<td>1</td>
<td>32</td>
<td>Counter-timer Virtual Timer Control register</td>
</tr>
<tr>
<td>CNTV_CTL_EL02</td>
<td>3</td>
<td>c14</td>
<td>5</td>
<td>c3</td>
<td>1</td>
<td>32</td>
<td>Counter-timer Virtual Timer Control register</td>
</tr>
<tr>
<td>CNTV_CVAL_EL0</td>
<td>3</td>
<td>c14</td>
<td>3</td>
<td>c3</td>
<td>2</td>
<td>64</td>
<td>Counter-timer Virtual Timer CompareValue register</td>
</tr>
<tr>
<td>CNTV_TVAL_EL0</td>
<td>3</td>
<td>c14</td>
<td>5</td>
<td>c3</td>
<td>2</td>
<td>64</td>
<td>Counter-timer Virtual Timer CompareValue register</td>
</tr>
<tr>
<td>CNTV_CVAL_EL02</td>
<td>3</td>
<td>c14</td>
<td>3</td>
<td>c3</td>
<td>0</td>
<td>32</td>
<td>Counter-timer Virtual Timer TimerValue register</td>
</tr>
<tr>
<td>CNTV_TVAL_EL02</td>
<td>3</td>
<td>c14</td>
<td>5</td>
<td>c3</td>
<td>0</td>
<td>32</td>
<td>Counter-timer Virtual Timer TimerValue register</td>
</tr>
<tr>
<td>CNTVCT_EL0</td>
<td>3</td>
<td>c14</td>
<td>3</td>
<td>c0</td>
<td>2</td>
<td>64</td>
<td>Counter-timer Virtual Count register</td>
</tr>
<tr>
<td>CNTVOFF_EL2</td>
<td>3</td>
<td>c14</td>
<td>4</td>
<td>c0</td>
<td>3</td>
<td>64</td>
<td>Counter-timer Virtual Offset register</td>
</tr>
<tr>
<td>CONTEXTIDR_EL1</td>
<td>3</td>
<td>c13</td>
<td>0</td>
<td>c0</td>
<td>1</td>
<td>32</td>
<td>Context ID Register (EL1)</td>
</tr>
<tr>
<td>CONTEXTIDR_EL12</td>
<td>3</td>
<td>c13</td>
<td>5</td>
<td>c0</td>
<td>1</td>
<td>32</td>
<td>Context ID Register (EL12)</td>
</tr>
<tr>
<td>CONTEXTIDR_EL2</td>
<td>3</td>
<td>c13</td>
<td>4</td>
<td>c0</td>
<td>1</td>
<td>32</td>
<td>Context ID Register (EL2)</td>
</tr>
<tr>
<td>CPACR_EL12</td>
<td>3</td>
<td>c1</td>
<td>5</td>
<td>c0</td>
<td>2</td>
<td>32</td>
<td>Architectural Feature Access Control Register</td>
</tr>
<tr>
<td>CPTER_EL3</td>
<td>3</td>
<td>c1</td>
<td>6</td>
<td>c1</td>
<td>2</td>
<td>32</td>
<td>Architectural Feature Trap Register (EL3)</td>
</tr>
<tr>
<td>DACR32_EL2</td>
<td>3</td>
<td>c3</td>
<td>4</td>
<td>c0</td>
<td>0</td>
<td>32</td>
<td>Domain Access Control Register</td>
</tr>
<tr>
<td>ESR_EL12</td>
<td>3</td>
<td>c5</td>
<td>5</td>
<td>c2</td>
<td>0</td>
<td>32</td>
<td>Exception Syndrome Register (EL12)</td>
</tr>
<tr>
<td>FAR_EL1</td>
<td>3</td>
<td>c6</td>
<td>0</td>
<td>c0</td>
<td>0</td>
<td>64</td>
<td>Fault Address Register (EL1)</td>
</tr>
<tr>
<td>FAR_EL12</td>
<td>3</td>
<td>c6</td>
<td>5</td>
<td>c0</td>
<td>0</td>
<td>64</td>
<td>Fault Address Register (EL12)</td>
</tr>
<tr>
<td>FAR_EL2</td>
<td>3</td>
<td>c6</td>
<td>4</td>
<td>c0</td>
<td>0</td>
<td>64</td>
<td>Fault Address Register (EL2)</td>
</tr>
<tr>
<td>FAR_EL3</td>
<td>3</td>
<td>c6</td>
<td>6</td>
<td>c0</td>
<td>0</td>
<td>64</td>
<td>Fault Address Register (EL3)</td>
</tr>
<tr>
<td>FPEXC32_EL2</td>
<td>3</td>
<td>c5</td>
<td>4</td>
<td>c3</td>
<td>0</td>
<td>32</td>
<td>Floating-point Exception Control register</td>
</tr>
<tr>
<td>HPFAR_EL2</td>
<td>3</td>
<td>c6</td>
<td>4</td>
<td>c0</td>
<td>4</td>
<td>64</td>
<td>Hypervisor IPA Fault Address Register</td>
</tr>
<tr>
<td>HSTR_EL2</td>
<td>3</td>
<td>c1</td>
<td>4</td>
<td>c1</td>
<td>3</td>
<td>32</td>
<td>Hypervisor System Trap Register</td>
</tr>
<tr>
<td>ID_AA64AFR0_EL1</td>
<td>3</td>
<td>c0</td>
<td>0</td>
<td>c5</td>
<td>4</td>
<td>64</td>
<td>AArch64 Auxiliary Feature Register 0</td>
</tr>
<tr>
<td>ID_AA64AFR1_EL1</td>
<td>3</td>
<td>c0</td>
<td>0</td>
<td>c5</td>
<td>5</td>
<td>64</td>
<td>AArch64 Auxiliary Feature Register 1</td>
</tr>
<tr>
<td>ID_AA64DFR1_EL1</td>
<td>3</td>
<td>c0</td>
<td>0</td>
<td>c5</td>
<td>1</td>
<td>64</td>
<td>AArch64 Debug Feature Register 1</td>
</tr>
<tr>
<td>ID_AA64PFR1_EL1</td>
<td>3</td>
<td>c0</td>
<td>0</td>
<td>c4</td>
<td>1</td>
<td>64</td>
<td>AArch64 Core Feature Register 1</td>
</tr>
<tr>
<td>ISR_EL1</td>
<td>3</td>
<td>c12</td>
<td>0</td>
<td>c1</td>
<td>0</td>
<td>32</td>
<td>Interrupt Status Register</td>
</tr>
<tr>
<td>LOREA_EL1</td>
<td>3</td>
<td>c10</td>
<td>0</td>
<td>c4</td>
<td>1</td>
<td>64</td>
<td>LORegion End Address Register</td>
</tr>
<tr>
<td>LORSA_EL1</td>
<td>3</td>
<td>c10</td>
<td>0</td>
<td>c4</td>
<td>0</td>
<td>64</td>
<td>LORegion Start Address Register</td>
</tr>
<tr>
<td>MAIR_EL1</td>
<td>3</td>
<td>c10</td>
<td>0</td>
<td>c2</td>
<td>0</td>
<td>64</td>
<td>Memory Attribute Indirection Register (EL1)</td>
</tr>
<tr>
<td>MAIR_EL12</td>
<td>3</td>
<td>c10</td>
<td>5</td>
<td>c2</td>
<td>0</td>
<td>64</td>
<td>Memory Attribute Indirection Register (EL12)</td>
</tr>
<tr>
<td>MAIR_EL2</td>
<td>3</td>
<td>c10</td>
<td>4</td>
<td>c2</td>
<td>0</td>
<td>64</td>
<td>Memory Attribute Indirection Register (EL2)</td>
</tr>
<tr>
<td>Name</td>
<td>Op0</td>
<td>CRn</td>
<td>Op1</td>
<td>CRm</td>
<td>Op2</td>
<td>Width</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>--------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>MAIR_EL3</td>
<td>3</td>
<td>c10</td>
<td>6</td>
<td>c2</td>
<td>0</td>
<td>64</td>
<td>Memory Attribute Indirection Register (EL3)</td>
</tr>
<tr>
<td>MDCR_EL2</td>
<td>3</td>
<td>c1</td>
<td>4</td>
<td>c1</td>
<td>1</td>
<td>32</td>
<td>Monitor Debug Configuration Register</td>
</tr>
<tr>
<td>MVFR0_EL1</td>
<td>3</td>
<td>c0</td>
<td>0</td>
<td>c3</td>
<td>0</td>
<td>32</td>
<td>AArch32 Media and VFP Feature Register 0</td>
</tr>
<tr>
<td>MVFR1_EL1</td>
<td>3</td>
<td>c0</td>
<td>0</td>
<td>c3</td>
<td>1</td>
<td>32</td>
<td>AArch32 Media and VFP Feature Register 1</td>
</tr>
<tr>
<td>MVFR2_EL1</td>
<td>3</td>
<td>c0</td>
<td>0</td>
<td>c3</td>
<td>2</td>
<td>32</td>
<td>AArch32 Media and VFP Feature Register 2</td>
</tr>
<tr>
<td>RMR_EL3</td>
<td>3</td>
<td>c12</td>
<td>6</td>
<td>c0</td>
<td>2</td>
<td>32</td>
<td>Reset Management Register</td>
</tr>
<tr>
<td>SCR_EL3</td>
<td>3</td>
<td>c1</td>
<td>6</td>
<td>c1</td>
<td>0</td>
<td>32</td>
<td>Secure Configuration Register</td>
</tr>
<tr>
<td>SDER32_EL3</td>
<td>3</td>
<td>c1</td>
<td>6</td>
<td>c1</td>
<td>1</td>
<td>32</td>
<td>AArch32 Secure Debug Enable Register</td>
</tr>
<tr>
<td>TCR_EL12</td>
<td>3</td>
<td>c2</td>
<td>5</td>
<td>c0</td>
<td>2</td>
<td>64</td>
<td>Translation Control Register (EL12)</td>
</tr>
<tr>
<td>TPIDR_EL0</td>
<td>3</td>
<td>c13</td>
<td>3</td>
<td>c0</td>
<td>2</td>
<td>64</td>
<td>EL0 Read/Write Software Thread ID Register</td>
</tr>
<tr>
<td>TPIDR_EL1</td>
<td>3</td>
<td>c13</td>
<td>0</td>
<td>c0</td>
<td>4</td>
<td>64</td>
<td>EL1 Software Thread ID Register</td>
</tr>
<tr>
<td>TPIDR_EL2</td>
<td>3</td>
<td>c13</td>
<td>4</td>
<td>c0</td>
<td>2</td>
<td>64</td>
<td>EL2 Software Thread ID Register</td>
</tr>
<tr>
<td>TPIDR_EL3</td>
<td>3</td>
<td>c13</td>
<td>6</td>
<td>c0</td>
<td>2</td>
<td>64</td>
<td>EL3 Software Thread ID Register</td>
</tr>
<tr>
<td>TPIDRRO_EL0</td>
<td>3</td>
<td>c13</td>
<td>3</td>
<td>c0</td>
<td>3</td>
<td>64</td>
<td>EL0 Read-Only Software Thread ID Register</td>
</tr>
<tr>
<td>TTBR0_EL12</td>
<td>3</td>
<td>c2</td>
<td>5</td>
<td>c0</td>
<td>0</td>
<td>64</td>
<td>Translation Table Base Register 0 (EL12)</td>
</tr>
<tr>
<td>TTBR1_EL12</td>
<td>3</td>
<td>c2</td>
<td>5</td>
<td>c0</td>
<td>1</td>
<td>64</td>
<td>Translation Table Base Register 1 (EL12)</td>
</tr>
<tr>
<td>VBAR_EL1</td>
<td>3</td>
<td>c12</td>
<td>0</td>
<td>c0</td>
<td>0</td>
<td>64</td>
<td>Vector Base Address Register (EL1)</td>
</tr>
<tr>
<td>VBAR_EL12</td>
<td>3</td>
<td>c12</td>
<td>5</td>
<td>c0</td>
<td>0</td>
<td>64</td>
<td>Vector Base Address Register (EL12)</td>
</tr>
<tr>
<td>VBAR_EL2</td>
<td>3</td>
<td>c12</td>
<td>4</td>
<td>c0</td>
<td>0</td>
<td>64</td>
<td>Vector Base Address Register (EL2)</td>
</tr>
<tr>
<td>VBAR_EL3</td>
<td>3</td>
<td>c12</td>
<td>6</td>
<td>c0</td>
<td>0</td>
<td>64</td>
<td>Vector Base Address Register (EL3)</td>
</tr>
<tr>
<td>VMPIDR_EL2</td>
<td>3</td>
<td>c0</td>
<td>4</td>
<td>c0</td>
<td>5</td>
<td>64</td>
<td>Virtualization Multiprocessor ID Register</td>
</tr>
<tr>
<td>VPIDR_EL2</td>
<td>3</td>
<td>c0</td>
<td>4</td>
<td>c0</td>
<td>0</td>
<td>32</td>
<td>Virtualization Core ID Register</td>
</tr>
</tbody>
</table>
B1.3  AArch64 IMPLEMENTATION DEFINED register summary

This section describes the AArch64 registers in the Cortex-A65 core that are IMPLEMENTATION DEFINED.

The following tables lists the AArch 64 IMPLEMENTATION DEFINED registers, sorted by opcode.

<table>
<thead>
<tr>
<th>Name</th>
<th>Copro</th>
<th>CRn</th>
<th>Op1</th>
<th>CRm</th>
<th>Op2</th>
<th>Width</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPUACTLR_EL1</td>
<td>3</td>
<td>c15</td>
<td>0</td>
<td>c1</td>
<td>0</td>
<td>64</td>
<td>B1.23 CPUACTLR_EL1, CPU Auxiliary Control Register, EL1 on page B1-154</td>
</tr>
<tr>
<td>CPUCFR_EL1</td>
<td>3</td>
<td>c15</td>
<td>0</td>
<td>c0</td>
<td>0</td>
<td>32</td>
<td>B1.24 CPUCFR_EL1, CPU Configuration Register, EL1 on page B1-156</td>
</tr>
<tr>
<td>CPUECTLR_EL1</td>
<td>3</td>
<td>c15</td>
<td>0</td>
<td>c1</td>
<td>4</td>
<td>64</td>
<td>B1.25 CPUECTLR_EL1, CPU Extended Control Register, EL1 on page B1-158</td>
</tr>
<tr>
<td>CPUPCR_EL3</td>
<td>3</td>
<td>15</td>
<td>6</td>
<td>c8</td>
<td>1</td>
<td>64</td>
<td>B1.26 CPUPCR_EL3, CPU Private Control Register, EL3 on page B1-162</td>
</tr>
<tr>
<td>CPUPMR_EL3</td>
<td>3</td>
<td>c15</td>
<td>6</td>
<td>c8</td>
<td>3</td>
<td>64</td>
<td>B1.27 CPUPMR_EL3, CPU Private Mask Register, EL3 on page B1-164</td>
</tr>
<tr>
<td>CPUPOR_EL3</td>
<td>3</td>
<td>c15</td>
<td>6</td>
<td>c8</td>
<td>2</td>
<td>64</td>
<td>B1.28 CPUPOR_EL3, CPU Private Operation Register, EL3 on page B1-166</td>
</tr>
<tr>
<td>CPUPSELR_EL3</td>
<td>3</td>
<td>c15</td>
<td>6</td>
<td>c8</td>
<td>0</td>
<td>32</td>
<td>B1.29 CPUPSELR_EL3, CPU Private Selection Register, EL3 on page B1-168</td>
</tr>
<tr>
<td>CPUPWRCRTL_EL1</td>
<td>3</td>
<td>c15</td>
<td>0</td>
<td>c2</td>
<td>7</td>
<td>32</td>
<td>B1.30 CPUPWRCRTL_EL1, Power Control Register, EL1 on page B1-170</td>
</tr>
<tr>
<td>ERXPFGCDN_EL1</td>
<td>3</td>
<td>c15</td>
<td>0</td>
<td>c2</td>
<td>2</td>
<td>32</td>
<td>B1.40 ERXPFGCDN_EL1, Selected Error Pseudo Fault Generation Count Down Register, EL1 on page B1-184</td>
</tr>
<tr>
<td>ERXPFGCTL_EL1</td>
<td>3</td>
<td>c15</td>
<td>0</td>
<td>c2</td>
<td>1</td>
<td>32</td>
<td>B1.41 ERXPFGCTL_EL1, Selected Error Pseudo Fault Generation Control Register, EL1 on page B1-186</td>
</tr>
<tr>
<td>ERXPFGF_EL1</td>
<td>3</td>
<td>c15</td>
<td>0</td>
<td>c2</td>
<td>0</td>
<td>32</td>
<td>B1.42 ERXPFGF_EL1, Selected Pseudo Fault Generation Feature Register, EL1 on page B1-188</td>
</tr>
</tbody>
</table>
B1.4 **AArch64 registers by functional group**

This section identifies the AArch64 registers by their functional groups and applies to the registers in the core that are *IMPLEMENTATION DEFINED* or have micro-architectural bit fields. Reset values are provided for these registers.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCSIDR__EL1</td>
<td>RO</td>
<td>-</td>
<td><strong>B1.18 CCSIDR_EL1, Cache Size ID Register, EL1</strong> on page B1-145</td>
</tr>
</tbody>
</table>
| CLIDR_EL1           | RO   | 0xC3000123 if L3 cache present.  
                     |                                 | 0x82000023 if no L3 cache.  
                     |                                 | **B1.19 CLIDR_EL1, Cache Level ID Register, EL1** on page B1-147             |
| CSSELR_EL1          | RW   | UNK                    | **B1.31 CSSELR_EL1, Cache Size Selection Register, EL1** on page B1-173      |
| CTR_EL0             | RO   | 0x84448004             | **B1.32 CTR_EL0, Cache Type Register, EL0** on page B1-174                   |
| DCZID_EL0           | RO   | 0x00000004             | **B1.33 DCZID_EL0, Data Cache Zero ID Register, EL0** on page B1-176         |
| ERRIDR_EL1          | RO   | -                      | **B1.35 ERRIDR_EL1, Error ID Register, EL1** on page B1-179                   |
| ID_AA64AFR0_EL1     | RO   | 0x00000000             | **B1.49 ID_AA64AFR0_EL1, AArch64 Auxiliary Feature Register 0** on page B1-196 |
| ID_AA64AFR1_EL1     | RO   | 0x00000000             | **B1.50 ID_AA64AFR1_EL1, AArch64 Auxiliary Feature Register 1** on page B1-197 |
| ID_AA64DFR0_EL1     | RO   | 0x0000000010305408     | **B1.51 ID_AA64DFR0_EL1, AArch64 Debug Feature Register 0, EL1** on page B1-198 |
| ID_AA64DFR1_EL1     | RO   | 0x00000000             | **B1.52 ID_AA64DFR1_EL1, AArch64 Debug Feature Register 1, EL1** on page B1-200 |
| ID_AA64ISAR0_EL1    | RO   | 0x00000000010211120 if the Cryptographic Extension is implemented.  
                     |                                 | 0x00000000010210000 if the Cryptographic Extension is not implemented.  
<pre><code>                 |                                 | **B1.53 ID_AA64ISAR0_EL1, AArch64 Instruction Set Attribute Register 0, EL1** on page B1-201 |
</code></pre>
<p>| ID_AA64ISAR1_EL1    | RO   | 0x000000000000000100001 | <strong>B1.54 ID_AA64ISAR1_EL1, AArch64 Instruction Set Attribute Register 1, EL1</strong> on page B1-203 |
| ID_AA64MMFR0_EL1    | RO   | 0x0000000001011124     | <strong>B1.55 ID_AA64MMFR0_EL1, AArch64 Memory Model Feature Register 0, EL1</strong> on page B1-204 |
| ID_AA64MMFR1_EL1    | RO   | 0x00000000010212122    | <strong>B1.56 ID_AA64MMFR1_EL1, AArch64 Memory Model Feature Register 1, EL1</strong> on page B1-206 |
| ID_AA64MMFR2_EL1    | RO   | 0x00000000000010111    | <strong>B1.57 ID_AA64MMFR2_EL1, AArch64 Memory Model Feature Register 2, EL1</strong> on page B1-208 |</p>
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID_AA64PFR0_EL1</td>
<td>RO</td>
<td>• 0x000000000010112222 if the GICv4 interface is disabled.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0x0000000011112222 if the GICv4 interface is enabled.</td>
<td></td>
</tr>
<tr>
<td>ID_AA64PFR1_EL1</td>
<td>RO</td>
<td>0x0000000000000010</td>
<td>B1.59 ID_AA64PFR1_EL1, AArch64 Processor Feature Register 1, EL1 on page B1-211</td>
</tr>
<tr>
<td>LORID_EL1</td>
<td>RO</td>
<td>0x00000000000000040004</td>
<td>B1.61 LORID_EL1, LORegion ID Register, EL1 on page B1-213</td>
</tr>
<tr>
<td>MIDR_EL1</td>
<td>RO</td>
<td>0x411FD061</td>
<td>B1.64 MIDR_EL1, Main ID Register, EL1 on page B1-217</td>
</tr>
<tr>
<td>MPIDR_EL1</td>
<td>RO</td>
<td>The reset value depends on CLUSTERIDAFF2[7:0] and CLUSTERIDAFF3[7:0]. See register description for details.</td>
<td>B1.65 MPIDR_EL1, Multiprocessor Affinity Register, EL1 on page B1-218</td>
</tr>
<tr>
<td>REVIDR_EL1</td>
<td>RO</td>
<td>0x00000000</td>
<td>B1.67 REVIDR_EL1, Revision ID Register, EL1 on page B1-221</td>
</tr>
<tr>
<td>VMPIDR_EL2</td>
<td>RW</td>
<td>The reset value is the value of MPIDR_EL1.</td>
<td>Virtualization Multiprocessor ID Register EL2</td>
</tr>
<tr>
<td>VPIDR_EL2</td>
<td>RW</td>
<td>The reset value is the value of MIDR_EL1.</td>
<td>Virtualization Core ID Register EL2</td>
</tr>
</tbody>
</table>

Table B1-5  Other system control registers

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTLR_EL1</td>
<td>RW</td>
<td>B1.5 ACTLR_EL1, Auxiliary Control Register, EL1 on page B1-130</td>
</tr>
<tr>
<td>ACTLR_EL2</td>
<td>RW</td>
<td>B1.6 ACTLR_EL2, Auxiliary Control Register, EL2 on page B1-131</td>
</tr>
<tr>
<td>ACTLR_EL3</td>
<td>RW</td>
<td>B1.7 ACTLR_EL3, Auxiliary Control Register, EL3 on page B1-133</td>
</tr>
<tr>
<td>CPACR_EL1</td>
<td>RW</td>
<td>B1.20 CPACR_EL1, Architectural Feature Access Control Register, EL1 on page B1-149</td>
</tr>
<tr>
<td>SCTLR_EL1</td>
<td>RW</td>
<td>B1.70 SCTLR_EL1, System Control Register, EL1 on page B1-224</td>
</tr>
<tr>
<td>SCTLR_EL2</td>
<td>RW</td>
<td>B1.71 SCTLR_EL2, System Control Register, EL2 on page B1-226</td>
</tr>
<tr>
<td>SCTLR_EL3</td>
<td>RW</td>
<td>B1.72 SCTLR_EL3, System Control Register, EL3 on page B1-227</td>
</tr>
<tr>
<td>SCTLR_EL12</td>
<td>RW</td>
<td>B1.70 SCTLR_EL1, System Control Register, EL1 on page B1-224</td>
</tr>
</tbody>
</table>
### Table B1-6 Reliability, Availability, Serviceability (RAS) registers

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISR_EL1</td>
<td>RW</td>
<td>B1.34 DISR_EL1, Deferred Interrupt Status Register, EL1 on page B1-177</td>
</tr>
<tr>
<td>ERRIDR_EL1</td>
<td>RW</td>
<td>B1.35 ERRIDR_EL1, Error ID Register, EL1 on page B1-179</td>
</tr>
<tr>
<td>ERRSELR_EL1</td>
<td>RW</td>
<td>B1.36 ERRSELR_EL1, Error Record Select Register, EL1 on page B1-180</td>
</tr>
<tr>
<td>ERXCTRLR_EL1</td>
<td>RW</td>
<td>B1.37 ERXCTRLR_EL1, Selected Error Record Control Register, EL1 on page B1-181</td>
</tr>
<tr>
<td>ERXFR_EL1</td>
<td>RO</td>
<td>B1.38 ERXFR_EL1, Selected Error Record Feature Register, EL1 on page B1-182</td>
</tr>
<tr>
<td>ERXMISC0_EL1</td>
<td>RW</td>
<td>B1.39 ERXMISC0_EL1, Selected Error Record Miscellaneous Register 0, EL1 on page B1-183</td>
</tr>
<tr>
<td>ERXSTATUS_EL1</td>
<td>RW</td>
<td>B1.43 ERXSTATUS_EL1, Selected Error Record Primary Status Register, EL1 on page B1-189</td>
</tr>
<tr>
<td>ERXPFGCDN_EL1</td>
<td>RW</td>
<td>B1.40 ERXPFGCDN_EL1, Selected Error Pseudo Fault Generation Count Down Register, EL1 on page B1-184</td>
</tr>
<tr>
<td>ERXPFCTLR_EL1</td>
<td>RW</td>
<td>B1.41 ERXPFCTLR_EL1, Selected Error Pseudo Fault Generation Control Register, EL1 on page B1-186</td>
</tr>
<tr>
<td>ERXPFGF_EL1</td>
<td>RO</td>
<td>B1.42 ERXPFGF_EL1, Selected Pseudo Fault Generation Feature Register, EL1 on page B1-188</td>
</tr>
<tr>
<td>HCR_EL2</td>
<td>RW</td>
<td>B1.48 HCR_EL2, Hypervisor Configuration Register, EL2 on page B1-194</td>
</tr>
<tr>
<td>VDISR_EL2</td>
<td>RW</td>
<td>B1.81 VDISR_EL2, Virtual Deferred Interrupt Status Register, EL2 on page B1-238</td>
</tr>
<tr>
<td>VSESIR_EL2</td>
<td>RW</td>
<td>B1.82 VSESIR_EL2, Virtual SError Exception Syndrome Register on page B1-239</td>
</tr>
</tbody>
</table>

### Table B1-7 Virtual Memory control registers

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMAIR_EL1</td>
<td>RW</td>
<td>B1.15 AMAIR_EL1, Auxiliary Memory Attribute Indirection Register, EL1 on page B1-142</td>
</tr>
<tr>
<td>AMAIR_EL2</td>
<td>RW</td>
<td>B1.16 AMAIR_EL2, Auxiliary Memory Attribute Indirection Register, EL2 on page B1-143</td>
</tr>
<tr>
<td>AMAIR_EL3</td>
<td>RW</td>
<td>B1.17 AMAIR_EL3, Auxiliary Memory Attribute Indirection Register, EL3 on page B1-144</td>
</tr>
<tr>
<td>LORC_EL1</td>
<td>RW</td>
<td>B1.60 LORC_EL1, LORegion Control Register, EL1 on page B1-212</td>
</tr>
<tr>
<td>LOREA_EL1</td>
<td>RW</td>
<td>LORegion End Address Register EL1</td>
</tr>
<tr>
<td>LORID_EL1</td>
<td>RO</td>
<td>B1.61 LORID_EL1, LORegion ID Register, EL1 on page B1-213</td>
</tr>
<tr>
<td>LORN_EL1</td>
<td>RW</td>
<td>B1.62 LORN_EL1, LORegion Number Register, EL1 on page B1-214</td>
</tr>
<tr>
<td>LORSA_EL1</td>
<td>RW</td>
<td>LORegion Start Address Register EL1</td>
</tr>
<tr>
<td>TCR_EL1</td>
<td>RW</td>
<td>B1.73 TCR_EL1, Translation Control Register, EL1 on page B1-229</td>
</tr>
<tr>
<td>TCR_EL2</td>
<td>RW</td>
<td>B1.74 TCR_EL2, Translation Control Register, EL2 on page B1-230</td>
</tr>
<tr>
<td>TCR_EL3</td>
<td>RW</td>
<td>B1.75 TCR_EL3, Translation Control Register, EL3 on page B1-231</td>
</tr>
</tbody>
</table>
Table B1-7  Virtual Memory control registers (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TTBR0_EL1</td>
<td>RW</td>
<td>B1.76 TTBR0_EL1, Translation Table Base Register 0, EL1 on page B1-233</td>
</tr>
<tr>
<td>TTBR0_EL2</td>
<td>RW</td>
<td>B1.77 TTBR0_EL2, Translation Table Base Register 0, EL2 on page B1-234</td>
</tr>
<tr>
<td>TTBR0_EL3</td>
<td>RW</td>
<td>B1.78 TTBR0_EL3, Translation Table Base Register 0, EL3 on page B1-235</td>
</tr>
<tr>
<td>TTBR1_EL1</td>
<td>RW</td>
<td>B1.79 TTBR1_EL1, Translation Table Base Register 1, EL1 on page B1-236</td>
</tr>
<tr>
<td>TTBR1_EL2</td>
<td>RW</td>
<td>B1.80 TTBR1_EL2, Translation Table Base Register 1, EL2 on page B1-237</td>
</tr>
<tr>
<td>VTTBR_EL2</td>
<td>RW</td>
<td>B1.84 VTTBR_EL2, Virtualization Translation Table Base Register, EL2 on page B1-241</td>
</tr>
</tbody>
</table>

Table B1-8  Virtualization registers

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTLR_EL2</td>
<td>RW</td>
<td>B1.6 ACTLR_EL2, Auxiliary Control Register, EL2 on page B1-131</td>
</tr>
<tr>
<td>AFSR0_EL2</td>
<td>RW</td>
<td>B1.9 AFSR0_EL2, Auxiliary Fault Status Register 0, EL2 on page B1-136</td>
</tr>
<tr>
<td>AFSR1_EL2</td>
<td>RW</td>
<td>B1.12 AFSR1_EL2, Auxiliary Fault Status Register 1, EL2 on page B1-139</td>
</tr>
<tr>
<td>AMAIR_EL2</td>
<td>RW</td>
<td>B1.16 AMAIR_EL2, Auxiliary Memory Attribute Indirection Register, EL2 on page B1-143</td>
</tr>
<tr>
<td>CPTR_EL2</td>
<td>RW</td>
<td>B1.21 CPTR_EL2, Architectural Feature Trap Register, EL2 on page B1-150</td>
</tr>
<tr>
<td>ESR_EL2</td>
<td>RW</td>
<td>B1.45 ESR_EL2, Exception Syndrome Register, EL2 on page B1-191</td>
</tr>
<tr>
<td>HACR_EL2</td>
<td>RW</td>
<td>B1.47 HACR_EL2, Hyp Auxiliary Configuration Register, EL2 on page B1-193</td>
</tr>
<tr>
<td>HCR_EL2</td>
<td>RW</td>
<td>B1.48 HCR_EL2, Hypervisor Configuration Register, EL2 on page B1-194</td>
</tr>
<tr>
<td>HPFAR_EL2</td>
<td>RW</td>
<td>Hypervisor IPA Fault Address Register EL2</td>
</tr>
<tr>
<td>TCR_EL2</td>
<td>RW</td>
<td>B1.74 TCR_EL2, Translation Control Register, EL2 on page B1-230</td>
</tr>
<tr>
<td>VMPIDR_EL2</td>
<td>RW</td>
<td>Virtualization Multiprocessor ID Register EL2</td>
</tr>
<tr>
<td>VPIDR_EL2</td>
<td>RW</td>
<td>Virtualization Core ID Register EL2</td>
</tr>
<tr>
<td>VSESR_EL2</td>
<td>RW</td>
<td>B1.82 VSESR_EL2, Virtual SError Exception Syndrome Register on page B1-239</td>
</tr>
<tr>
<td>VTCR_EL2</td>
<td>RW</td>
<td>B1.83 VTCR_EL2, Virtualization Translation Control Register, EL2 on page B1-240</td>
</tr>
<tr>
<td>VTTBR_EL2</td>
<td>RW</td>
<td>B1.84 VTTBR_EL2, Virtualization Translation Table Base Register, EL2 on page B1-241</td>
</tr>
</tbody>
</table>

Table B1-9  Exception and fault handling registers

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFSR0_EL1</td>
<td>RW</td>
<td>B1.8 AFSR0_EL1, Auxiliary Fault Status Register 0, EL1 on page B1-135</td>
</tr>
<tr>
<td>AFSR0_EL2</td>
<td>RW</td>
<td>B1.9 AFSR0_EL2, Auxiliary Fault Status Register 0, EL2 on page B1-136</td>
</tr>
</tbody>
</table>
## Table B1-9 Exception and fault handling registers (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFSR0_EL3</td>
<td>RW</td>
<td>B1.10 AFSR0_EL3, Auxiliary Fault Status Register 0, EL3 on page B1-137</td>
</tr>
<tr>
<td>AFSR1_EL1</td>
<td>RW</td>
<td>B1.11 AFSR1_EL1, Auxiliary Fault Status Register 1, EL1 on page B1-138</td>
</tr>
<tr>
<td>AFSR1_EL2</td>
<td>RW</td>
<td>B1.12 AFSR1_EL2, Auxiliary Fault Status Register 1, EL2 on page B1-139</td>
</tr>
<tr>
<td>AFSR1_EL3</td>
<td>RW</td>
<td>B1.13 AFSR1_EL3, Auxiliary Fault Status Register 1, EL3 on page B1-140</td>
</tr>
<tr>
<td>DISR_EL1</td>
<td>RW</td>
<td>B1.34 DISR_EL1, Deferred Interrupt Status Register, EL1 on page B1-177</td>
</tr>
<tr>
<td>ESR_EL1</td>
<td>RW</td>
<td>B1.44 ESR_EL1, Exception Syndrome Register, EL1 on page B1-190</td>
</tr>
<tr>
<td>ESR_EL2</td>
<td>RW</td>
<td>B1.45 ESR_EL2, Exception Syndrome Register, EL2 on page B1-191</td>
</tr>
<tr>
<td>ESR_EL3</td>
<td>RW</td>
<td>B1.46 ESR_EL3, Exception Syndrome Register, EL3 on page B1-192</td>
</tr>
<tr>
<td>HPFAR_EL2</td>
<td>RW</td>
<td>Hypervisor IPA Fault Address Register EL2</td>
</tr>
<tr>
<td>VDISR_EL2</td>
<td>RW</td>
<td>B1.81 VDISR_EL2, Virtual Deferred Interrupt Status Register, EL2 on page B1-238</td>
</tr>
<tr>
<td>VSESR_EL2</td>
<td>RW</td>
<td>B1.82 VSESR_EL2, Virtual SError Exception Syndrome Register on page B1-239</td>
</tr>
</tbody>
</table>

## Table B1-10 IMPLEMENTATION DEFINED registers

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPUACTLR_EL1</td>
<td>RW</td>
<td>B1.23 CPUACTLR_EL1, CPU Auxiliary Control Register, EL1 on page B1-154</td>
</tr>
<tr>
<td>CPUCFR_EL1</td>
<td>RO</td>
<td>B1.24 CPUCFR_EL1, CPU Configuration Register, EL1 on page B1-156</td>
</tr>
<tr>
<td>CPUECTLR_EL1</td>
<td>RW</td>
<td>B1.25 CPUECTLR_EL1, CPU Extended Control Register, EL1 on page B1-158</td>
</tr>
<tr>
<td>CPUPWRCTLR_EL1</td>
<td>RW</td>
<td>B1.30 CPUPWRCTLR_EL1, Power Control Register, EL1 on page B1-170</td>
</tr>
<tr>
<td>ERXPFGCDN_EL1</td>
<td>RW</td>
<td>B1.40 ERXPFGCDN_EL1, Selected Error Pseudo Fault Generation Count Down Register, EL1 on page B1-184</td>
</tr>
<tr>
<td>ERXPFGCTL_EL1</td>
<td>RW</td>
<td>B1.41 ERXPFGCTL_EL1, Selected Error Pseudo Fault Generation Control Register, EL1 on page B1-186</td>
</tr>
<tr>
<td>ERXPFGF_EL1</td>
<td>RW</td>
<td>B1.42 ERXPFGF_EL1, Selected Pseudo Fault Generation Feature Register, EL1 on page B1-188</td>
</tr>
</tbody>
</table>

## Table B1-11 Security

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTLR_EL3</td>
<td>RW</td>
<td>B1.7 ACTLR_EL3, Auxiliary Control Register, EL3 on page B1-133</td>
</tr>
<tr>
<td>AFSR0_EL3</td>
<td>RW</td>
<td>B1.10 AFSR0_EL3, Auxiliary Fault Status Register 0, EL3 on page B1-137</td>
</tr>
<tr>
<td>AFSR1_EL3</td>
<td>RW</td>
<td>B1.13 AFSR1_EL3, Auxiliary Fault Status Register 1, EL3 on page B1-140</td>
</tr>
<tr>
<td>AMAIR_EL3</td>
<td>RW</td>
<td>B1.17 AMAIR_EL3, Auxiliary Memory Attribute Indirection Register, EL3 on page B1-144</td>
</tr>
</tbody>
</table>
### Table B1-11 Security (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPTR_EL3</td>
<td>RW</td>
<td>B1.22 CPTR_EL3, Architectural Feature Trap Register, EL3 on page B1-153</td>
</tr>
<tr>
<td>MDCR_EL3</td>
<td>RW</td>
<td>B1.63 MDCR_EL3, Monitor Debug Configuration Register, EL3 on page B1-215</td>
</tr>
</tbody>
</table>

### Table B1-12 Reset management registers

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RMR_EL3</td>
<td>RW</td>
<td>B1.68 RMR_EL3, Reset Management Register on page B1-222</td>
</tr>
<tr>
<td>RVBAR_EL3</td>
<td>RW</td>
<td>B1.69 RVBAR_EL3, Reset Vector Base Address Register, EL3 on page B1-223</td>
</tr>
</tbody>
</table>

### Table B1-13 Address registers

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAR_EL1</td>
<td>RW</td>
<td>B1.66 PAR_EL1, Physical Address Register, EL1 on page B1-220</td>
</tr>
</tbody>
</table>
B1.5 ACTLR_EL1, Auxiliary Control Register, EL1

ACTLR_EL1 provides IMPLEMENTATION DEFINED configuration and control options for execution at EL1 and EL0.

Bit field descriptions
ACTLR_EL1 is a 64-bit register, and is part of:
- The Other system control registers functional group.
- The IMPLEMENTATION DEFINED functional group.

RES0, [63:0]
RES0 Reserved.

Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.

Accessibility

For a description of the prioritization of any generated exceptions, see Synchronous exception prioritization in the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile. Subject to these prioritization rules, this register provides the following access permissions.

Execution at EL3:
- Write access to CPUACTLR_EL1 always.
- Write access to CPUECTLR_EL1 always.

Execution at EL2:
- Write access to CPUACTLR_EL1 if ACTLR_EL3[0] = 1.
- Write access to CPUECTLR_EL1 if ACTLR_EL3[1] = 1.

Execution at EL1:
- Write access to CPUACTLR_EL1 if ACTLR_EL3[0] = 1 & (ACTLR_EL2[0] = 1 || SCR_EL3.NS = 0).

If write access is not possible the instruction traps to the lowest Exception level that denied access. For example, if the ACTLR_EL2 denies access to lower Exception levels then an attempt at EL1 traps to EL2.
**B1.6 ACTLR_EL2, Auxiliary Control Register, EL2**

The ACTLR_EL2 provides **IMPLEMENTATION DEFINED** configuration and control options for EL2.

**Bit field descriptions**

ACTLR_EL2 is a 64-bit register, and is part of:
- The Virtualization registers functional group.
- The Other system control registers functional group.
- The **IMPLEMENTATION DEFINED** functional group.

![Figure B1-2 ACTLR_EL2 bit assignments](image)

**RES0, [63:13]**  
RES0 Reserved.

**CLUSTERPMUEN, [12]**  
Performance Management Registers enable. The possible values are:
- 0  CLUSTERPM* registers are not write-accessible from a lower Exception level. This is the reset value.
- 1  CLUSTERPM* registers are write-accessible from EL1 Non-secure if they are write-accessible from EL2.

**SMEN, [11]**  
Scheme Management Registers enable. The possible values are:
- 0  Registers CLUSTERACPSID, CLUSTERSTASHSID, CLUSTERPARTCR, CLUSTERBUSQOS, and CLUSTERTHREADSIDOVR are not write-accessible from EL1 Non-secure. This is the reset value.
- 1  Registers CLUSTERACPSID, CLUSTERSTASHSID, CLUSTERPARTCR, CLUSTERBUSQOS, and CLUSTERTHREADSIDOVR are write-accessible from EL1 Non-secure if they are write-accessible from EL2.

**TSIDEN, [10]**  
Thread Scheme ID Register enable. The possible values are:
- 0  Register CLUSTERTHREADSID is not write-accessible from EL1 Non-secure. This is the reset value.
- 1  Register CLUSTERTHREADSID is write-accessible from EL1 Non-secure if they are write-accessible from EL2.

**RES0, [9:8]**
PWREN, [7]

Power Control Registers enable. The possible values are:

0  Registers CPUPWRCTRLR, CLUSTERPWRCTRLR, CLUSTERPWRDN,
    CLUSTERPWRSTAT, CLUSTERL3HIT and CLUSTERL3MISS are not write-
    accessible from EL1 Non-secure. This is the reset value.
1  Registers CPUPWRCTRLR, CLUSTERPWRCTRLR, CLUSTERPWRDN,
    CLUSTERPWRSTAT, CLUSTERL3HIT and CLUSTERL3MISS are write-accessible
    from EL1 Non-secure if they are write-accessible from EL2.

RES0, [6]

RES0  Reserved.

ERXPFGEN, [5]

Error Record Registers enable. The possible values are:

0  ERXPFG* are not write-accessible from EL1 Non-secure. This is the reset value.
1  ERXPFG* are write-accessible from EL1 Non-secure if they are write-accessible from
    EL2.

RES0, [4:2]

RES0  Reserved.

ECTLRREN, [1]

Extended Control Registers enable. The possible values are:

0  CPUECTLR and CLUSTERECTLR are not write-accessible from EL1 Non-secure.
    This is the reset value.
1  CPUECTLR and CLUSTERECTLR are write-accessible from EL1 Non-secure if they
    are write-accessible from EL2.

ACTLRREN, [0]

Auxiliary Control Registers enable. The possible values are:

0  CPUACTLR and CLUSTERACTLR are not write-accessible from EL1 Non-secure.
    This is the reset value.
1  CPUACTLR and CLUSTERACTLR are write-accessible from EL1 Non-secure if
    they are write-accessible from EL2.

Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the
B1.7 ACTLR_EL3, Auxiliary Control Register, EL3

The ACTLR_EL3 provides IMPLEMENTATION DEFINED configuration and control options for EL3.

Bit field descriptions
ACTLR_EL3 is a 64-bit register, and is part of:
- The Other system control registers functional group.
- The Security registers functional group.
- The IMPLEMENTATION DEFINED functional group.

RES0, [63:13]
RES0 Reserved.

CLUSTERPMUEN, [12]
Performance Management Registers enable. The possible values are:
- 0 CLUSTERPM* registers are not write-accessible from a lower Exception level. This is the reset value.
- 1 CLUSTERPM* registers are write-accessible from EL2 and EL1 Secure.

Scheme Management Registers enable. The possible values are:
- 0 Registers CLUSTERACPSID, CLUSTERSTASHSID, CLUSTERPARTCR, CLUSTERBUSQOS, and CLUSTERTHREADSIDOVR are not write-accessible from EL2 and EL1 Secure. This is the reset value.
- 1 Registers CLUSTERACPSID, CLUSTERSTASHSID, CLUSTERPARTCR, CLUSTERBUSQOS, and CLUSTERTHREADSIDOVR are write-accessible from EL2 and EL1 Secure.

TSIDEN, [10]
Thread Scheme ID Register enable. The possible values are:
- 0 Register CLUSTERTHREADSID is not write-accessible from EL2 and EL1 Secure. This is the reset value.
- 1 Register CLUSTERTHREADSID is write-accessible from EL2 and EL1 Secure.

RES0, [9:8]
RES0 Reserved.

PWREN, [7]
Power Control Registers enable. The possible values are:

0 Registers CPUPWRCTRL, CLUSTERPWRCTRL, CLUSTERPWRDN, CLUSTERPWRSTAT, CLUSTERL3HIT and CLUSTERL3MISS are not write-accessible from EL2 and EL1 Secure. This is the reset value.

1 Registers CPUPWRCTRL, CLUSTERPWRCTRL, CLUSTERPWRDN, CLUSTERPWRSTAT, CLUSTERL3HIT and CLUSTERL3MISS are write-accessible from EL2 and EL1 Secure.

RES0, [6]

RES0 Reserved.

ERXPFGEN, [5]

Error Record Registers enable. The possible values are:

0 ERXPFG* are not write-accessible from EL2 and EL1 Secure. This is the reset value.

1 ERXPFG* are write-accessible from EL2 and EL1 Secure.

RES0, [4:2]

RES0 Reserved.

ECTLRREN, [1]

Extended Control Registers enable. The possible values are:

0 CPUECTRL and CLUSTERECTLR are not write-accessible from EL2 and EL1 Secure. This is the reset value.

1 CPUECTRL and CLUSTERECTLR are write-accessible from EL2 and EL1 Secure.

ACTLRREN, [0]

Auxiliary Control Registers enable. The possible values are:

0 CPUACTLR and CLUSTERACTLR are not write-accessible from EL2 and EL1 Secure. This is the reset value.

1 CPUACTLR and CLUSTERACTLR are write-accessible from EL2 and EL1 Secure.

Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.
B1.8 AFSR0_EL1, Auxiliary Fault Status Register 0, EL1

AFSR0_EL1 provides additional IMPLEMENTATION DEFINED fault status information for exceptions that are taken to EL1. In the Cortex-A65 core, no additional information is provided for these exceptions. Therefore this register is not used.
### B1.9 AFSR0_EL2, Auxiliary Fault Status Register 0, EL2

AFSR0_EL2 provides additional implementation defined fault status information for exceptions that are taken to EL2.

**Bit field descriptions**

AFSR0_EL2 is a 32-bit register, and is part of:

- The Virtualization registers functional group.
- The Exception and fault handling registers functional group.
- The implementation defined functional group.

```plaintext
0 31
```

**RES0**

Reserved, RES0.

**Configurations**

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.*
B1.10 AFSR0_EL3, Auxiliary Fault Status Register 0, EL3

AFSR0_EL3 provides additional IMPLEMENTATION DEFINED fault status information for exceptions that are taken to EL3. In the Cortex-A65 core, no additional information is provided for these exceptions. Therefore this register is not used.
B1.11 AFSR1_EL1, Auxiliary Fault Status Register 1, EL1

AFSR1_EL1 provides additional IMPLEMENTATION DEFINED fault status information for exceptions that are taken to EL1. This register is not used in Cortex-A65.
B1.12 AFSR1_EL2, Auxiliary Fault Status Register 1, EL2

AFSR1_EL2 provides additional IMPLEMENTATION DEFINED fault status information for exceptions that are taken to EL2. This register is not used in the Cortex-A65 core.
B1.13 AFSR1_EL3, Auxiliary Fault Status Register 1, EL3

AFSR1_EL3 provides additional IMPLEMENTATION DEFINED fault status information for exceptions that are taken to EL3. This register is not used in the Cortex-A65 core.
B1.14 AIDR_EL1, Auxiliary ID Register, EL1

AIDR_EL1 provides IMPLEMENTATION DEFINED identification information. This register is not used in the A65 core.
B1.15 AMAIR_EL1, Auxiliary Memory Attribute Indirection Register, EL1

AMAIR_EL1 provides IMPLEMENTATION DEFINED memory attributes for the memory regions specified by MAIR_EL1. This register is not used in the Cortex-A65 core.
B1.16 AMAIR_EL2, Auxiliary Memory Attribute Indirection Register, EL2

AMAIR_EL2 provides IMPLEMENTATION DEFINED memory attributes for the memory regions specified by MAIR_EL2. This register is not used in the Cortex-A65 core.
B1.17 AMAIR_EL3, Auxiliary Memory Attribute Indirection Register, EL3

AMAIR_EL3 provides IMPLEMENTATION DEFINED memory attributes for the memory regions specified by MAIR_EL3. This register is not used in the Cortex-A65 core.
B1.18 CCSIDR_EL1, Cache Size ID Register, EL1

The CCSIDR_EL1 provides information about the architecture of the currently selected cache.

Bit field descriptions

CCSIDR_EL1 is a 32-bit register, and is part of the Identification registers functional group.

This register is Read Only.

WT, [31]
Indicates whether the selected cache level supports Write-Through:
0 Cache Write-Through is not supported at any level.
1 Write-Through is supported.

For more information about encoding, see CCSIDR_EL1 encodings on page B1-146.

WB, [30]
Indicates whether the selected cache level supports Write-Back. Permitted values are:
0 Write-Back is not supported.
1 Write-Back is supported.

For more information about encoding, see CCSIDR_EL1 encodings on page B1-146.

RA, [29]
Indicates whether the selected cache level supports read-allocation. Permitted values are:
0 Read-allocation is not supported.
1 Read-allocation is supported.

For more information about encoding, see CCSIDR_EL1 encodings on page B1-146.

WA, [28]
Indicates whether the selected cache level supports write-allocation. Permitted values are:
0 Write-allocation is not supported.
1 Write-allocation is supported.

For more information about encoding, see CCSIDR_EL1 encodings on page B1-146.

NumSets, [27:13]
(Number of sets in cache) - 1. Therefore, a value of 0 indicates one set in the cache. The number of sets does not have to be a power of 2.

For more information about encoding, see CCSIDR_EL1 encodings on page B1-146.
Associativity, [12:3]

(Associativity of cache) - 1. Therefore, a value of 0 indicates an associativity of 1. The
associativity does not have to be a power of 2.

For more information about encoding, see CCSIDR_EL1 encodings on page B1-146.

LineSize, [2:0]

(Log₂(Number of bytes in cache line)) - 4. For example:

Indicates the (log₂ (number of words in cache line)) - 2:

For a line length of 16 bytes: Log₂(16) = 4, LineSize entry = 0. This is the minimum
line length.

For a line length of 32 bytes: Log₂(32) = 5, LineSize entry = 1.

For more information about encoding, see CCSIDR_EL1 encodings on page B1-146.

Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the

CCSIDR_EL1 encodings

The following table shows the individual bit field and complete register encodings for the CCSIDR_EL1.

<table>
<thead>
<tr>
<th>CSSEL R</th>
<th>Cache</th>
<th>Size</th>
<th>Complete register encoding</th>
<th>Register bit field encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>WT</td>
</tr>
<tr>
<td>0b000</td>
<td>0b0 L1 Data cache</td>
<td>32KB</td>
<td>0x0700FE01A</td>
<td>0x0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>64KB</td>
<td>0x0701FE01A</td>
<td>0x0</td>
</tr>
<tr>
<td>0b000</td>
<td>0b1 L1 Instruction cache</td>
<td>32KB</td>
<td>0x2000FE01A</td>
<td>0x0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>64KB</td>
<td>0x201FE01A</td>
<td>0x0</td>
</tr>
<tr>
<td>0b001</td>
<td>0b0 L2 cache</td>
<td>64KB</td>
<td>0x0701FE01A</td>
<td>0x0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>128KB</td>
<td>0x0703FE01A</td>
<td>0x0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>256KB</td>
<td>0x0707FE01A</td>
<td>0x0</td>
</tr>
<tr>
<td>0b001</td>
<td>0b1 Reserved</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0b010</td>
<td>0b0 L3 cache</td>
<td>512KB</td>
<td>0x0703FE07A</td>
<td>0x0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1MB</td>
<td>0x0707FE07A</td>
<td>0x0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2MB</td>
<td>0x070FFE07A</td>
<td>0x0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4MB</td>
<td>0x071FFE07A</td>
<td>0x0</td>
</tr>
<tr>
<td>0b010</td>
<td>0b1 Reserved</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0b0101  - 0b1111</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Note

If no per-core L2 cache is present the core uses L3 cache as L2 (cluster L2 cache) and the L3 encodings
apply.
B1.19 CLIDR_EL1, Cache Level ID Register, EL1

The CLIDR_EL1 identifies the type of cache, or caches, implemented at each level, up to a maximum of seven levels.

It also identifies the Level of Coherency (LoC) and Level of Unification (LoU) for the cache hierarchy.

**Bit field descriptions**

CLIDR_EL1 is a 64-bit register, and is part of the Identification registers functional group.

This register is Read Only.

**RES0, [63:33]**

RES0 Reserved.

**ICB, [32:30]**

Inner cache boundary. This field indicates the boundary between the inner and the outer domain:

- **0b001** Neither per-core L2 or cluster cache are present.
- **0b010** Either per-core L2 or cluster L2 cache is present, but not both.
- **0b011** Both per-core L2 and cluster L3 caches are present.

**LoUU, [29:27]**

Indicates the Level of Unification Uniprocessor for the cache hierarchy:

- **0b000** No levels of cache need to cleaned or invalidated when cleaning or invalidating to the Point of Unification. This is the value if no caches are configured.

**LoC, [26:24]**

Indicates the Level of Coherency for the cache hierarchy:

- **0b001** Neither per-core L2 or cluster cache are present.
- **0b010** Either per-core L2 or cluster L2 cache is present, but not both.
- **0b011** Both per-core L2 and cluster L3 caches are present.

**LoUIS, [23:21]**

Indicates the Level of Unification Inner Shareable (LoUIS) for the cache hierarchy.

- **0b000** No cache level needs cleaning to Point of Unification.

**RES0, [20:9]**

No cache at levels L7 down to L4.

RES0 Reserved.

**Ctype3, [8:6]**
Indicates the type of cache if the core implements L3 cache. If present, unified instruction and data caches at Level 3:

- 0b100  Both per-core L2 and cluster L3 caches are present.
- 0b000  All other options.

If Ctype2 has a value of 0b000, then the value of Ctype3 must be ignored.

Ctype2, [5:3]
Indicates the type of unified instruction and data caches at Level 2:

- 0b100  Either per-core L2 or cluster L2 cache is present.
- 0b000  All other options.

Ctype1, [2:0]
Indicates the type of cache implemented at L1:

- 0b011  Separate instruction and data caches at L1.

Configurations
There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.
B1.20 CPACR_EL1, Architectural Feature Access Control Register, EL1

The CPACR_EL1 controls access to Advanced SIMD and floating-point functionality from EL0, EL1, and EL3.

**Bit field descriptions**

CPACR_EL1 is a 32-bit register, and is part of the Other system control registers functional group.

![Figure B1-7 CPACR_EL1 bit assignments](image)

**RES0, [31:22]**

RES0 is reserved.

**FPEN, [21:20]**

Traps instructions that access registers associated with Advanced SIMD and floating-point execution to trap to EL1 when executed from EL0 or EL1. The possible values are:

- **0b00**: Trap any instruction in EL0 or EL1 that uses registers associated with Advanced SIMD and floating-point execution. The reset value is 0b00.
- **0b01**: Trap any instruction in EL0 that uses registers associated with Advanced SIMD and floating-point execution. Instructions in EL1 are not trapped.
- **0b10**: Trap any instruction in EL0 or EL1 that uses registers associated with Advanced SIMD and floating-point execution.
- **0b11**: No instructions are trapped.

This field is RES0 if Advanced SIMD and floating-point are not implemented.

**RAZ/WI, [19:0]**

RAZ/WI is read as zero, write ignore.

**Configurations**

There are no configuration notes.

Bit fields and details not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.
B1.21 CPTR_EL2, Architectural Feature Trap Register, EL2

The CPTR_EL2 controls trapping to EL2 for accesses to CPACR, Trace functionality and registers associated with Advanced SIMD and floating-point execution. It also controls EL2 access to this functionality.

Bit field descriptions

CPTR_EL2 is a 32-bit register, and is part of the Virtualization registers functional group.

When HCR_EL2.E2H == 0:

TCPAC, [31]
Traps direct access to CPACR_EL1 from Non-secure EL1 to EL2. The possible values are:
0 Access to CPACR is not trapped. This is the reset value.
1 Non-secure EL1 accesses to CPACR_EL1 are trapped to EL2.

Note

CPACR_EL1 is not accessible at EL0.

RES0, [30:21]
RES0 Reserved.

TTA, [20]
Trap Trace Access. Traps Non-secure System register accesses to all implemented trace registers to EL2, from both Execution states.

0 No accesses are trapped. This is the reset value.
1 Any attempt at EL2, or Non-secure EL0 or EL1, to execute a System register access to an implemented trace register is trapped to EL2.

System register accesses to the trace registers can have side-effects. When a System register access is trapped, any side-effects that are normally associated with the access do not occur before the exception is taken.

If System register access to the trace functionality is not supported, this bit is RES0.

RES0, [19:14]
RES0 Reserved.

RES1, [13:12]
RES1 Reserved.
RES0, [11]

Reserved.

TFP, [10]

Traps instructions that access registers associated with Advanced SIMD and floating-point execution from a lower exception level to EL2, unless trapped to EL1. The possible values are:

0  Does not cause any instruction to be trapped. This is the reset value.
1  Causes any instructions that use the registers that are associated with Advanced SIMD or floating-point execution to be trapped. This is always the value if the Advanced SIMD and floating-point support is not implemented.

RES1, [9:0]

Reserved.

When HCR_EL2.E2H == 1:

TCPAC, [31]

When HCR_EL2.TGE is 0, traps Non-secure EL1 accesses to CPACR_EL1 to EL2, from both Execution states. The possible values are:

0  Access to CPACR_EL1 is not trapped. This is the reset value.
1  Non-secure EL1 accesses to CPACR_EL1 are trapped to EL2.

Note

CPACR_EL1 is not accessible at EL0.

RES0, [30:29]

Reserved.

TTA, [28]

Trap Trace Access. Traps Non-secure System register accesses to all implemented trace registers to EL2, from both Execution states.

0  No accesses are trapped. This is the reset value.
1  Any attempt at EL2, or Non-secure EL0 or EL1, to execute a System register access to an implemented trace register is trapped to EL2.

System register accesses to the trace registers can have side-effects. When a System register access is trapped, any side-effects that are normally associated with the access do not occur before the exception is taken.

If System register access to the trace functionality is not supported, this bit is RES0.
RES0, [27:22]
  RES0  Reserved.

FPEN, [21:20]
Traps EL2, Non-secure EL0 and, when HCR_EL2.TGE is 0, Non-secure EL1 accesses to the
Advanced SIMD and floating-point registers to EL2, from both Execution states. The possible
values are:
0b00  This control causes any instructions at Non-secure EL0, EL1, or EL2 that use the
registers associated with Advanced SIMD and floating-point execution to be trapped,
unless HCR_EL2.TGE is 0 and they are trapped by CPACR_EL1.FPEN,
CPACR_EL1.ZEN, or CPTR_EL2.ZEN. This is the reset value.
0b01  When HCR_EL2.TGE is 0, this control does not cause any instructions to be trapped.
When HCR_EL2.TGE is 1, this control causes instructions at Non-secure EL0 that use
the registers associated with Advanced SIMD and floating-point execution to be
trapped, unless they are trapped by CPTR_EL2.ZEN.
0b10  This control causes any instructions at Non-secure EL0, EL1, or EL2 that use the
registers associated with Advanced SIMD and floating-point execution to be trapped,
unless HCR_EL2.TGE is 0 and they are trapped by CPACR_EL1.FPEN,
CPACR_EL1.ZEN, or CPTR_EL2.ZEN.
0b11  No instructions are trapped.

RES0, [19:18]
  RES0  Reserved.

RES0, [17:16]
  RES0  Reserved.

RES0, [15:0]
  RES0  Reserved.

Configurations
RW fields in this register reset to UNKNOWN values.
Bit fields and details that are not provided in this description are architecturally defined. See the
B1.22 CPTR_EL3, Architectural Feature Trap Register, EL3

The CPTR_EL3 controls trapping to EL3 of access to CPACR_EL1, CPTR_EL2, trace functionality and registers associated with Advanced SIMD and floating-point execution.

It also controls EL3 access to trace functionality and registers associated with Advanced SIMD and floating-point execution.

Bit field descriptions

CPTR_EL3 is a 32-bit register, and is part of the Security registers functional group.

<table>
<thead>
<tr>
<th>31</th>
<th></th>
<th></th>
<th>11 10 9</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAZ/WI</td>
<td></td>
<td></td>
<td>RAZ/WI</td>
<td></td>
</tr>
</tbody>
</table>

TFP, [10]

This causes instructions that access the registers that are associated with Advanced SIMD or floating-point execution to trap to EL3 when executed from any Exception level, unless trapped to EL1 or EL2. The possible values are:

0 Does not cause any instruction to be trapped. This is the reset value.
1 Any attempt at any Exception level to execute an instruction that uses the registers that are associated with SVE, Advanced SIMD and floating-point is trapped to EL3, subject to the exception prioritization rules.

RAZ/WI, [9:0]

RAZ/WI Read-As-Zero, write ignore.

Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.
B1.23 CPUACTLR_EL1, CPU Auxiliary Control Register, EL1

The CPUACTLR_EL1 provides implementation defined configuration and control options for the core.

Bit field descriptions

CPUACTLR_EL1 is a 64-bit register, and is part of the implementation defined registers functional group.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>63</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Figure B1-11 CPUACTLR_EL1 bit assignments

Reserved, [63:0]

Reserved for Arm internal use.

Configurations

CPUACTLR_EL1 is common to the Secure and Non-secure states.

Usage constraints

Accessing the CPUACTLR_EL1

The CPU Auxiliary Control Register can be written only when the system is idle. Arm recommends that you write to this register after a powerup reset, before the MMU is enabled. While each thread appears to have its own CPUACTLR_EL1, the register is physically shared. Writes from one thread are visible to the other thread. Arm recommends that the register only be updated once software synchronization between the threads guarantees exclusive access to the register.

Setting many of these bits can cause significantly lower performance on your code. Therefore, Arm strongly recommends that you do not modify this register unless directed by Arm.

This register is accessible as follows:

This register can be read with the MRS instruction using the following syntax:

MRS <Xt>,<systemreg>

This register can be written with the MSR instruction using the following syntax:

MSR <systemreg>, <Xt>

This syntax is encoded with the following settings in the instruction encoding:

<table>
<thead>
<tr>
<th>&lt;systemreg&gt;</th>
<th>op0</th>
<th>op1</th>
<th>CRn</th>
<th>CRm</th>
<th>op2</th>
</tr>
</thead>
<tbody>
<tr>
<td>S3_0_C15_C1_0</td>
<td>11</td>
<td>000</td>
<td>1111</td>
<td>0001</td>
<td>000</td>
</tr>
</tbody>
</table>

Accessibility

This register is accessible in software as follows:
This register is write-accessible in EL1 on either of these conditions:
• ACTLR_EL3.CPUACTLR_EN == 1 && ACTLR_EL2.CPUACTLR_EN==1.
• ACTLR_EL3.CPUACTLR_EN==1 && SCR.NS==0.

This register is write-accessible in EL2 if ACTLR_EL3.CPUACTLR_EN==1.

If write access is not possible, then trap to the lowest Exception level that denied the access (EL2 or EL3).

'n/a Not accessible. The core cannot be executing at this Exception level, so this access is not possible.

**Traps and enables**

For a description of the prioritization of any generated exceptions, see *Synchronous exception prioritization* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.*
B1.24 CPUCFR_EL1, CPU Configuration Register, EL1

The CPUCFR_EL1 provides configuration information for the core.

Bit field descriptions

CPUCFR_EL1 is a 32-bit register, and is part of the IMPLEMENTATION DEFINED registers functional group. This register is Read Only, Write Ignore.

![Figure B1-12 CPUCFR_EL1 bit assignments](image)

RES0, [31:2]
Reserved, RES0.

ECC, [1:0]
Indicates whether ECC is present or not. The value is:
01 ECC is present.

Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.

Usage constraints

Accessing the CPUCFR_EL1

This register can be read with the MRS instruction using the following syntax:

```
MRS <Xt>,<systemreg>
```

To access the CPUCFR_EL1:

```
MRS <Xt>, CPUCFR_EL1 ; Read CPUCFR_EL1 into Xt
```

This syntax is encoded with the following settings in the instruction encoding:

<table>
<thead>
<tr>
<th>&lt;systemreg&gt;</th>
<th>op0</th>
<th>op1</th>
<th>CRn</th>
<th>CRm</th>
<th>op2</th>
</tr>
</thead>
<tbody>
<tr>
<td>S3_0_C15_C0_0</td>
<td>11</td>
<td>000</td>
<td>1111</td>
<td>0000</td>
<td>000</td>
</tr>
</tbody>
</table>

Accessibility

This register is accessible in software as follows:

<table>
<thead>
<tr>
<th>&lt;systemreg&gt;</th>
<th>Control</th>
<th>Accessibility</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>E2H</td>
<td>TGE</td>
</tr>
<tr>
<td>S3_0_C15_C0_0</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>S3_0_C15_C0_0</td>
<td>x</td>
<td>0</td>
</tr>
<tr>
<td>S3_0_C15_C0_0</td>
<td>x</td>
<td>1</td>
</tr>
</tbody>
</table>
'n/a' Not accessible. The PE cannot be executing at this Exception level, so this access is not possible.
B1.25  CPUECTLR_EL1, CPU Extended Control Register, EL1

The CPUECTLR_EL1 provides extra implementation defined configuration and control options for the core.

**Bit field descriptions**

CPUECTLR_EL1 is a 64-bit register, and is part of the 64-bit registers functional group.

![CPUECTLR_EL1 bit assignments](image)

**RES0, [63:40]**

Reserved.

**ATOMIC, [39:38]**

Controls behavior of both cacheable load atomic instructions (including SWP and CAS) and store atomic instructions that do not hit in the L1 cache in the unique state. The possible values are:

- **00**: Cacheable load atomic instructions (including SWP and CAS) will make up one fill request to perform as near atomics. They will otherwise perform as far atomics. Cacheable store atomic instructions are performed as near atomics if they hit the cache in a unique state. Otherwise, they are performed as far atomics. This is the reset value.
- **01**: All cacheable atomic instructions are forced to be executed as near atomics in the L1 cache.
- **10**: All cacheable atomic instructions are forced to be executed as far atomics in the L3 cache or beyond.
- **11**: Cacheable load atomic instructions (including SWP and CAS) are forced to be executed as near atomics. Store atomics are performed as near atomics if they hit in the cache in a unique state; otherwise, they are performed as far atomics.

**L2FLUSH, [37]**

Controls handling of clean lines during core powerdown sequences. The possible values are:

- **0**: During a core powerdown sequence, clean lines in the L1 and L2 caches are allocated into the L3 cache. This can improve performance if it is known that the data is likely to be used soon by another core. This is the reset value.
- **1**: During a core powerdown sequence, clean lines in the L1 and L2 caches are discarded and do not allocate into the L3 cache.

**RES0, [36:30]**

Reserved.

**L3WSCTL, [29:28]**
Write streaming no-L3-allocate threshold. The possible values are:

00  128th consecutive streaming cache line does not allocate in the L1, L2, or L3 cache.
01  1024th consecutive streaming cache line does not allocate in the L1, L2, or L3 cache. This is the reset value.
10  4096th consecutive streaming cache line does not allocate in the L1, L2, or L3 cache.
11  Disables streaming. All Write-Allocate lines allocate in the L1, L2, or L3 cache.

L2WSCTL, [27:26]
Write streaming no-L2-allocate threshold. The possible values are:

00  16th consecutive streaming cache line does not allocate in the L1 or L2 cache.
01  128th consecutive streaming cache line does not allocate in the L1 or L2 cache. This is the reset value.
10  512th consecutive streaming cache line does not allocate in the L1 or L2 cache.
11  Disables streaming. All Write-Allocate lines allocate in the L1 or L2 cache.

L1WSCTL, [25:24]
Write streaming no-L1-allocate threshold. The possible values are:

00  4th consecutive streaming cache line does not allocate in the L1 cache. This is the reset value.
01  64th consecutive streaming cache line does not allocate in the L1 cache.
10  128th consecutive streaming cache line does not allocate in the L1 cache.
11  Disables streaming. All Write-Allocate lines allocate in the L1 cache.

RES0, [23]
RES0 Reserved.

STBPFDIS, [22]
Disable Store Buffer STB accesses from being tracked and trained by the hardware prefcher. The possible values are:

0  STB accesses are tracked and trained by the hardware prefcher. Hardware prefetch requests may be generated as a result of STB generated line fills. This is the reset value.
1  STB accesses are not tracked and trained by the prefcher. No hardware prefetch requests will be generated as a result of STB generated line fills.

RES0, [21:16]
RES0 Reserved.

L1PCTL, [15:13]
L1 Data prefetch control. The value of the L1PCTL field determines the maximum number of outstanding data prefetches allowed in the L1 memory system (not counting the data prefetches generated by software load/PLD instructions).

000  Prefetch disabled.
001  1 outstanding prefetch allowed.
010  2 outstanding prefetches allowed.
011  3 outstanding prefetches allowed.
100  4 outstanding prefetches allowed.
101  5 outstanding prefetches allowed. This is the reset value.
110  6 outstanding prefetches allowed.
111  7 outstanding prefetches allowed.

L3PCTL, [12:10]
L3 Data prefetch control. The value of the L3PCTL field determines the approximate distance
between the L1 prefetcher and requests sent to the L3 memory system. Increasing this distance
may improve performance on systems with higher latency to main memory, but increasing it too
far can reduce performance.
000  Fetch 16 lines ahead.
001  Fetch 32 lines ahead.
010  Reserved.
011  Reserved.
100  Disable L3 prefetching.
101  Fetch 2 lines ahead.
110  Fetch 4 lines ahead.
111  Fetch 8 lines ahead. This is the reset value.

RES0, [9:1]
RES0  Reserved.

EXTLLC, [0]
0  Indicates that an external Last-level cache is present in the system, and that the
DataSource field on the master CHI interface indicates when data is returned from the
LLC. This is used to control how the LL_CACHE* PMU events count.

Configurations
This register has no configuration notes.

Usage constraints

Accessing the CPUECTRL_EL1
The CPUECTRL_EL1 can be written dynamically.
This register is accessible as follows:
This register can be read with the MRS instruction using the following syntax:

MRS <Xt>,<systemreg>

This register can be written with the MRS instruction using the following syntax:

MSR <systemreg>, <Xt>

This syntax is encoded with the following settings in the instruction encoding:

<table>
<thead>
<tr>
<th>&lt;systemreg&gt;</th>
<th>op0</th>
<th>op1</th>
<th>CRn</th>
<th>CRm</th>
<th>op2</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPUECTRL_EL1</td>
<td>11</td>
<td>000</td>
<td>1111</td>
<td>0001</td>
<td>100</td>
</tr>
</tbody>
</table>
Accessibility

This register is accessible in software as follows:

<table>
<thead>
<tr>
<th>&lt;systemreg&gt;</th>
<th>Control</th>
<th>Accessibility</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>E2H</td>
<td>TGE</td>
</tr>
<tr>
<td>CPUECTRL_EL1</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>CPUECTRL_EL1</td>
<td>x</td>
<td>0</td>
</tr>
<tr>
<td>CPUECTRL_EL1</td>
<td>x</td>
<td>1</td>
</tr>
</tbody>
</table>

'n/a' Not accessible. The PE cannot be executing at this Exception level, so this access is not possible.

Traps and enables

For a description of the prioritization of any generated exceptions, see Synchronous exception prioritization in the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.
B1.26 CPUPCR_EL3, CPU Private Control Register, EL3

The CPUPCR_EL3 provides implementation defined configuration and control options for the core.

**Bit field descriptions**

CPUPCR_EL3 is a 64-bit register, and is part of the implementation defined registers functional group.

<table>
<thead>
<tr>
<th>Bit Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>63</td>
<td>Reserved</td>
</tr>
<tr>
<td>62-0</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Reserved, [63:0]

Reserved for Arm internal use.

**Configurations**

CPUPCR_EL3 is only accessible in Secure state.

**Usage constraints**

**Accessing the CPUPCR_EL3**

The CPUPCR_EL3 can be written only when the system is idle. Arm recommends that you write to this register after a powerup reset, before the MMU is enabled.

Writing to this register might cause unpredictable behaviors. Therefore, Arm strongly recommends that you do not modify this register unless directed by Arm.

This register is accessible as follows:

This register can be read with the MRS instruction using the following syntax:

\[
\text{MRS } \langle \text{Xt} \rangle, \langle \text{systemreg} \rangle
\]

This register can be written with the MSR instruction using the following syntax:

\[
\text{MSR } \langle \text{systemreg} \rangle, \langle \text{Xt} \rangle
\]

This syntax is encoded with the following settings in the instruction encoding:

<table>
<thead>
<tr>
<th>&lt;systemreg&gt;</th>
<th>op0</th>
<th>op1</th>
<th>CRn</th>
<th>CRm</th>
<th>op2</th>
</tr>
</thead>
<tbody>
<tr>
<td>S3_6_C15_8_1</td>
<td>11</td>
<td>110</td>
<td>1111</td>
<td>1000</td>
<td>001</td>
</tr>
</tbody>
</table>

**Accessibility**

This register is accessible in software as follows:

<table>
<thead>
<tr>
<th>&lt;systemreg&gt;</th>
<th>Control</th>
<th>Accessibility</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>E2H</td>
<td>TGE</td>
</tr>
<tr>
<td>S3_6_C15_8_1</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>S3_6_C15_8_1</td>
<td>x</td>
<td>0</td>
</tr>
<tr>
<td>S3_6_C15_8_1</td>
<td>x</td>
<td>1</td>
</tr>
</tbody>
</table>

'n/a' Not accessible. The core cannot be executing at this Exception level, so this access is not possible.
Traps and enables

For a description of the prioritization of any generated exceptions, see *Synchronous exception prioritization* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.*
B1.27 CPUPMR_EL3, CPU Private Mask Register, EL3

The CPUPMR_EL3 provides IMPLEMENTATION DEFINED configuration and control options for the core.

Bit field descriptions

CPUPMR_EL3 is a 64-bit register, and is part of the IMPLEMENTATION DEFINED registers functional group.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>63</td>
<td>Reserved</td>
</tr>
<tr>
<td>31-0</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Reserved, [63:0]

Reserved for Arm internal use.

Configurations

CPUPMR_EL3 is only accessible in Secure state.

Usage constraints

Accessing the CPUPMR_EL3

The CPUPMR_EL3 can be written only when the system is idle. Arm recommends that you write to this register after a powerup reset, before the MMU is enabled.

Writing to this register might cause UNPREDICTABLE behaviors. Therefore, Arm strongly recommends that you do not modify this register unless directed by Arm.

This register is accessible as follows:

This register can be read with the MRS instruction using the following syntax:

```
MRS <Xt>,<systemreg>
```

This register can be written with the MSR instruction using the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax is encoded with the following settings in the instruction encoding:

<table>
<thead>
<tr>
<th>&lt;systemreg&gt;</th>
<th>op0</th>
<th>op1</th>
<th>CRn</th>
<th>CRm</th>
<th>op2</th>
</tr>
</thead>
<tbody>
<tr>
<td>S3_6.C15_8_3</td>
<td>11</td>
<td>110</td>
<td>111</td>
<td>1000</td>
<td>011</td>
</tr>
</tbody>
</table>

Accessibility

This register is accessible in software as follows:

<table>
<thead>
<tr>
<th>&lt;systemreg&gt;</th>
<th>Control</th>
<th>Accessibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>E2H</td>
<td>TGE</td>
<td>NS</td>
</tr>
<tr>
<td>S3_6.C15_8_3</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>S3_6.C15_8_3</td>
<td>x</td>
<td>0</td>
</tr>
<tr>
<td>S3_6.C15_8_3</td>
<td>x</td>
<td>1</td>
</tr>
</tbody>
</table>

'n/a' Not accessible. The core cannot be executing at this Exception level, so this access is not possible.
Traps and enables

For a description of the prioritization of any generated exceptions, see *Synchronous exception prioritization* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.*
B1.28  CPUPOR_EL3, CPU Private Operation Register, EL3

The CPUPOR_EL3 provides IMPLEMENTATION DEFINED configuration and control options for the core.

**Bit field descriptions**

CPUPOR_EL3 is a 64-bit register, and is part of the IMPLEMENTATION DEFINED registers functional group.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>63</td>
<td>Reserved</td>
</tr>
<tr>
<td>0</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

**Figure B1-16 CPUPOR_EL3 bit assignments**

Reserved, [63:0]

Reserved for Arm internal use.

**Configurations**

CPUPOR_EL3 is only accessible in Secure state.

**Usage constraints**

**Accessing the CPUPOR_EL3**

The CPUPOR_EL3 can be written only when the system is idle. Arm recommends that you write to this register after a powerup reset, before the MMU is enabled.

Writing to this register might cause UNPREDICTABLE behaviors. Therefore, Arm strongly recommends that you do not modify this register unless directed by Arm.

This register is accessible as follows:

This register can be read with the MRS instruction using the following syntax:

```
MRS <Xt>,<systemreg>
```

This register can be written with the MSR instruction using the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax is encoded with the following settings in the instruction encoding:

<table>
<thead>
<tr>
<th>&lt;systemreg&gt;</th>
<th>op0</th>
<th>op1</th>
<th>CRn</th>
<th>CRm</th>
<th>op2</th>
</tr>
</thead>
<tbody>
<tr>
<td>S3_6_C15_8_2</td>
<td>11</td>
<td>110</td>
<td>1111</td>
<td>1000</td>
<td>010</td>
</tr>
</tbody>
</table>

**Accessibility**

This register is accessible in software as follows:

<table>
<thead>
<tr>
<th>&lt;systemreg&gt;</th>
<th>Control</th>
<th>Accessibility</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>E2H</td>
<td>TGE</td>
</tr>
<tr>
<td>S3_6_C15_8_2</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>S3_6_C15_8_2</td>
<td>x</td>
<td>0</td>
</tr>
<tr>
<td>S3_6_C15_8_2</td>
<td>x</td>
<td>1</td>
</tr>
</tbody>
</table>

‘n/a’ Not accessible. The core cannot be executing at this Exception level, so this access is not possible.
Traps and enables

For a description of the prioritization of any generated exceptions, see *Synchronous exception prioritization* in the *Arm*® *Architecture Reference Manual* *Armv8*, for *Armv8-A architecture profile*.
B1.29  CPUPSERL_EL3, CPU Private Selection Register, EL3

The CPUPSERL_EL3 provides IMPLEMENTATION DEFINED configuration and control options for the core.

Bit field descriptions
CPUPSERL_EL3 is a 32-bit register, and is part of the IMPLEMENTATION DEFINED registers functional group.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Reserved, [31:0]  
Reserved for Arm internal use.

Configurations  
CPUPSERL_EL3 is only accessible in Secure state.

Usage constraints

Accessing the CPUPSERL_EL3
The CPUPSERL_EL3 can be written only when the system is idle. Arm recommends that you write to this register after a powerup reset, before the MMU is enabled.

Writing to this register might cause UNPREDICTABLE behaviors. Therefore, Arm strongly recommends that you do not modify this register unless directed by Arm.

This register is accessible as follows:

This register can be read with the MRS instruction using the following syntax:

MRS <Xt>,<systemreg>

This register can be written with the MSR instruction using the following syntax:

MSR <systemreg>, <Xt>

This syntax is encoded with the following settings in the instruction encoding:

<table>
<thead>
<tr>
<th>&lt;systemreg&gt;</th>
<th>op0</th>
<th>op1</th>
<th>CRn</th>
<th>CRm</th>
<th>op2</th>
</tr>
</thead>
<tbody>
<tr>
<td>S3.6.C15.8.0</td>
<td>11</td>
<td>110</td>
<td>1111</td>
<td>1000</td>
<td>000</td>
</tr>
</tbody>
</table>

Accessibility

This register is accessible in software as follows:

<table>
<thead>
<tr>
<th>&lt;systemreg&gt;</th>
<th>Control</th>
<th>Accessibility</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>E2H</td>
<td>TGE</td>
</tr>
<tr>
<td>S3.6.C15.8.0</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>S3.6.C15.8.0</td>
<td>x</td>
<td>0</td>
</tr>
<tr>
<td>S3.6.C15.8.0</td>
<td>x</td>
<td>1</td>
</tr>
</tbody>
</table>

'n/a' Not accessible. The core cannot be executing at this Exception level, so this access is not possible.
Traps and enables

For a description of the prioritization of any generated exceptions, see *Synchronous exception prioritization* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.*
B1.30 CPUPWRCTLR_EL1, Power Control Register, EL1

The CPUPWRCTLR_EL1 provides information about power control support for the core.

**Bit field descriptions**

CPUPWRCTLR_EL1 is a 32-bit register, and is part of the IMPLEMENTATION DEFINED registers functional group.

This register resets to value 0x00000000.

![Figure B1-18 CPUPWRCTLR_EL1 bit assignments](image)

**RES0, [31:13]**

RES0 Reserved.

**SIMD_RET_CTRL, [12:10]**

Advanced SIMD and floating-point retention control:

000 Disable the retention circuit. This is the default value, see Table B1-15 CPUPWRCTRL Retention Control Field on page B1-171 for more retention control options.

**WFE_RET_CTRL, [9:7]**

CPU WFE retention control:

000 Disable the retention circuit. This is the default value, see Table B1-15 CPUPWRCTRL Retention Control Field on page B1-171 for more retention control options.

**WFI_RET_CTRL, [6:4]**

CPU WFI retention control:

000 Disable the retention circuit. This is the default value, see Table B1-15 CPUPWRCTRL Retention Control Field on page B1-171 for more retention control options.

**RES0, [3:1]**

RES0 Reserved.

**CORE_PWRDN_EN, [0]**

Indicates to the power controller using PACTIVE if the core wants to power down when it enters WFI state.

0 No power down requested. This is the reset value.
1 A power down is requested.
Table B1-15 CPUPWRCTLR Retention Control Field

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Number of counter ticks</th>
<th>Minimum retention entry delay (System counter at 50MHz-10MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>Disable the retention circuit</td>
<td>Default Condition.</td>
</tr>
<tr>
<td>001</td>
<td>2</td>
<td>40ns-200ns</td>
</tr>
<tr>
<td>010</td>
<td>8</td>
<td>160ns-800ns</td>
</tr>
<tr>
<td>011</td>
<td>32</td>
<td>640ns – 3,200ns</td>
</tr>
<tr>
<td>100</td>
<td>64</td>
<td>1,280ns-6,400ns</td>
</tr>
<tr>
<td>101</td>
<td>128</td>
<td>2,560ns-12,800ns</td>
</tr>
<tr>
<td>110</td>
<td>256</td>
<td>5,120ns-25,600ns</td>
</tr>
<tr>
<td>111</td>
<td>512</td>
<td>10,240ns-51,200ns</td>
</tr>
</tbody>
</table>

Configurations

There are no configuration notes.

Usage constraints

Accessing the CPUPWRCTLR_EL1

This register can be read using MRS with the following syntax:

MRS <Xt>,<systemreg>

This register can be written using MSR with the following syntax:

MSR <systemreg>, <Xt>

This syntax is encoded with the following settings in the instruction encoding:

<table>
<thead>
<tr>
<th>&lt;systemreg&gt;</th>
<th>op0</th>
<th>op1</th>
<th>CRn</th>
<th>CRm</th>
<th>op2</th>
</tr>
</thead>
<tbody>
<tr>
<td>S3_0_C15_C2_7</td>
<td>11</td>
<td>000</td>
<td>1111</td>
<td>0010</td>
<td>11</td>
</tr>
</tbody>
</table>

Accessibility

This register is accessible in software as follows:

<table>
<thead>
<tr>
<th>&lt;systemreg&gt;</th>
<th>Control</th>
<th>Accessibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>E2H</td>
<td>TGE</td>
<td>NS</td>
</tr>
<tr>
<td>S3_0_C15_C2_7</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>S3_0_C15_C2_7</td>
<td>x</td>
<td>0</td>
</tr>
<tr>
<td>S3_0_C15_C2_7</td>
<td>x</td>
<td>1</td>
</tr>
</tbody>
</table>

'n/a' Not accessible. The PE cannot be executing at this Exception level, so this access is not possible.

\[a\] The number of system counter ticks required before the core signals retention readiness on PACTIVE to the power controller. The core does not accept a retention entry request until this time.
Traps and enables

For a description of the prioritization of any generated exceptions, see *Synchronous exception prioritization* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for exceptions taken to AArch64 state.

Write access to this register from EL1 or EL2 depends on the value of bit[7] of ACTLR_EL2 and ACTLR_EL3.
B1.31 CSSELR_EL1, Cache Size Selection Register, EL1

CSSELR_EL1 selects the current Cache Size ID Register (CCSIDR_EL1), by specifying:

- The required cache level.
- The cache type, either instruction or data cache.

For details of the CCSIDR_EL1, see B1.18 CCSIDR_EL1, Cache Size ID Register, EL1 on page B1-145.

**Bit field descriptions**

CSSELR_EL1 is a 32-bit register, and is part of the Identification registers functional group.

![Figure B1-19 CSSELR_EL1 bit assignments](image)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:4</td>
<td>Reserved.</td>
</tr>
<tr>
<td>Level, [3:1]</td>
<td>Cache level of required cache:</td>
</tr>
<tr>
<td>000</td>
<td>L1.</td>
</tr>
<tr>
<td>001</td>
<td>L2.</td>
</tr>
<tr>
<td>010</td>
<td>L3, if present.</td>
</tr>
</tbody>
</table>

The combination of Level=001 and InD=1 is reserved.

The combinations of Level and InD for 0100 to 1111 are reserved.

<table>
<thead>
<tr>
<th>InD, [0]</th>
<th>Instruction not Data bit:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Data or unified cache.</td>
</tr>
<tr>
<td>1</td>
<td>Instruction cache.</td>
</tr>
</tbody>
</table>

The combination of Level=001 and InD=1 is reserved.

The combinations of Level and InD for 0100 to 1111 are reserved.

**Configurations**

If a cache level is missing but CSSELR_EL1 selects this level, then a CCSIDR_EL1 read returns an **UNKNOWN** value.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.*
B1.32  CTR_EL0, Cache Type Register, EL0

The CTR_EL0 provides information about the architecture of the caches.

**Bit field descriptions**

CTR_EL0 is a 32-bit register, and is part of the Identification registers functional group. This register is Read Only.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>28</th>
<th>27</th>
<th>24</th>
<th>23</th>
<th>20</th>
<th>19</th>
<th>16</th>
<th>14</th>
<th>13</th>
<th>4</th>
<th>3</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWG</td>
<td>ERG</td>
<td>DminLine</td>
<td>L1Ip</td>
<td>IminLine</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**RES1, [31]**

RES1  Reserved.

**RES0, [30:28]**

RES0  Reserved.

**CWG, [27:24]**

Cache write-back granule. Log<sub>2</sub> of the number of words of the maximum size of memory that can be overwritten as a result of the eviction of a cache entry that has had a memory location in it modified:

0100  Cache write-back granule size is 16 words.

**ERG, [23:20]**

Exclusives Reservation Granule. Log<sub>2</sub> of the number of words of the maximum size of the reservation granule that has been implemented for the Load-Exclusive and Store-Exclusive instructions:

0100  Exclusive reservation granule size is 16 words.

**DminLine, [19:16]**

Log<sub>2</sub> of the number of words in the smallest cache line of all the data and unified caches that the core controls:

0100  Smallest data cache line size is 16 words.

**L1Ip, [15:14]**

Instruction cache policy. Indicates the indexing and tagging policy for the L1 Instruction cache:

10  *Virtually Indexed Physically Tagged* (VIPT).

**RES0, [13:4]**

RES0  Reserved.

**IminLine, [3:0]**
Log₂ of the number of words in the smallest cache line of all the instruction caches that the core controls.

0100  Smallest instruction cache line size is 16 words.

Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.
B1.33 DCZID_EL0, Data Cache Zero ID Register, EL0

The DCZID_EL0 indicates the block size written with byte values of zero by the DC ZVA (Data Cache Zero by Address) system instruction.

**Bit field descriptions**

DCZID_EL0 is a 32-bit register, and is part of the Identification registers functional group. This register is Read Only.

![DCZID_EL0 bit assignments](image)

**RES0, [31:5]**

Reserved.

**DZP, [4]**

Prohibit the DC ZVA instruction:

- 0: DC ZVA instruction permitted.
- 1: DC ZVA instruction is prohibited.

**BlockSize, [3:0]**

Log₂ of the block size in words:

- 0100: The block size is 16 words.

**Configurations**

There are no configuration notes.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.*
B1.34 DISR_EL1, Deferred Interrupt Status Register, EL1

The DISR_EL1 records the SError interrupts consumed by an ESB instruction.

**Bit field descriptions**

DISR_EL1 is a 64-bit register, and is part of the registers Reliability, Availability, Serviceability (RAS) functional group.

![Figure B1-22 DISR_EL1 bit assignments, DISR_EL1.IDS is 0](image)

**RES0, [63:32]**
Reserved, RES0.

**A, [31]**
Set to 1 when ESB defers an asynchronous SError interrupt. If the implementation does not include any synchronizable sources of SError interrupt, this bit is RES0.

**RES0, [30:25]**
Reserved, RES0.

**IDS, [24]**
Indicates the type of format the deferred SError interrupt uses. The value of this bit is:
- 0: Deferred error uses architecturally-defined format.

**RES0, [23:13]**
Reserved, RES0.

**AET, [12:10]**
Asynchronous Error Type. Describes the state of the core after taking an asynchronous Data Abort exception. The value or values are:
- 0b000: Uncontainable error (UC).
- 0b001: Unrecoverable error (UEU).
- 0b010: Restartable error (UEO).
- 0b011: Recoverable error (UER).
- 0b110: Corrected error (CE).

All other values are reserved. Reserved values might be defined in a future version of the architecture.

**Note**
- This field is only valid if IDS == 0b0 and DFSC == 0b10010.
- The recovery software must also examine any implemented fault records to determine the location and extent of the error.
EA, [9]
  Reserved, RES0.

RES0, [8:6]
  Reserved, RES0.

DFSC, [5:0]
  Data Fault Status Code. The possible values of this field are:
  
  010001  Asynchronous SError interrupt.

Configurations

There are no configuration notes.

Bit fields and details not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.
B1.35 ERRIDR_EL1, Error ID Register, EL1

The ERRIDR_EL1 defines the number of error record registers.

Bit field descriptions

ERRIDR_EL1 is a 32-bit register, and is part of the registers Reliability, Availability, Serviceability (RAS) functional group.

This register is Read Only.

RES0, [31:16]
RES0  Reserved.

NUM, [15:0]
Number of records that can be accessed through the Error Record system registers.
0x0002  Two records present, if L3 cache is present.

Configurations

There are no configuration notes.

Bit fields and details not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.
B1.36 ERRSELR_EL1, Error Record Select Register, EL1

The ERRSELR_EL1 selects which error record should be accessed through the Error Record system registers. This register is not reset on a warm reset.

**Bit field descriptions**

ERRSELR_EL1 is a 64-bit register, and is part of the **Reliability, Availability, Serviceability (RAS)** registers functional group.

![Figure B1-24 ERRSELR_EL1 bit assignments](image)

RES0, [63:1]

Reserved, RES0.

SEL, [0]

Selects which error record should be accessed.

0 Select record 0 containing errors from Level 1 and Level 2 RAMs located on the Cortex-A65 core.

1 Select record 1 containing errors from Level 3 RAMs located on the DSU.

**Configurations**

There are no configuration notes.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.*
B1.37 ERXCTRL_EL1, Selected Error Record Control Register, EL1

Register ERXCTRL_EL1 accesses the ERR<sub>n</sub>CTRL control register for the error record selected by ERRSEL_EL1.SEL.

If ERRSEL_EL1.SEL==0, then ERXCTRL_EL1 accesses the ERR0CTRL register of the core error record. See B2.2 ERR0CTRL, Error Record Control Register on page B2-245.

If ERRSEL_EL1.SEL==1, then ERXCTRL_EL1 accesses the ERR1CTRL register of the DSU error record. See the Arm® DynamIQ™ Shared Unit Technical Reference Manual.
B1.38 ERXFR_EL1, Selected Error Record Feature Register, EL1

Register ERXFR_EL1 accesses the ERR<n>FR feature register for the error record selected by ERRSEL_EL1.SEL.

If ERRSEL_EL1.SEL==0, then ERXFR_EL1 accesses the ERR0FR register of the core error record. See B2.3 ERR0FR, Error Record Feature Register on page B2-247.

If ERRSEL_EL1.SEL==1, then ERXFR_EL1 accesses the ERR1FR register of the DSU error record. See the Arm® DynamIQ™ Shared Unit Technical Reference Manual.
B1.39  ERXMISC0_EL1, Selected Error Record Miscellaneous Register 0, EL1

Register ERXMISC0_EL1 accesses the ERR<n>MISC0 register for the error record selected by
ERRSELR_EL1.SEL.

If ERRSELR_EL1.SEL==0, then ERXMISC0_EL1 accesses the ERR0MISC0 register of the core error
record. See B2.4 ERR0MISC0, Error Record Miscellaneous Register 0 on page B2-249.

If ERRSELR_EL1.SEL==1, then ERXMISC0_EL1 accesses the ERR1MISC0 register of the DSU error
record. See the Arm® DynamIQ™ Shared Unit Technical Reference Manual.
B1.40  ERXPFGCDN_EL1, Selected Error Pseudo Fault Generation Count Down Register, EL1

Register ERXPFGCDN_EL1 accesses the ERR<n>PFGCND register for the error record selected by ERRSELR_EL1.SEL.

If ERRSELR_EL1.SEL==0, then ERXPFGCDN_EL1 accesses the ERR0PFGCDN register of the core error record. See B2.5 ERR0PFGCDN, Error Pseudo Fault Generation Count Down Register on page B2-251.

If ERRSELR_EL1.SEL==1, then ERXPFGCDN_EL1 accesses the ERR1PFGCDNR register of the DSU error record. See the Arm® DynamIQ™ Shared Unit Technical Reference Manual.

Configurations

There are no configuration notes.

Accessing the ERXPFGCDN_EL1

This register can be read using MRS with the following syntax:

\[ \text{MRS} \text{ <Xt>,<systemreg> } \]

This register can be written using MSR with the following syntax:

\[ \text{MSR} \text{ <Xt>,<systemreg> } \]

This syntax is encoded with the following settings in the instruction encoding:

<table>
<thead>
<tr>
<th>&lt;systemreg&gt;</th>
<th>op0</th>
<th>op1</th>
<th>CRn</th>
<th>CRm</th>
<th>op2</th>
</tr>
</thead>
<tbody>
<tr>
<td>S3_0_C15_C2_2</td>
<td>11</td>
<td>000</td>
<td>1111</td>
<td>0010</td>
<td>010</td>
</tr>
</tbody>
</table>

Accessibility

This register is accessible in software as follows:

<table>
<thead>
<tr>
<th>&lt;syntax&gt;</th>
<th>Control</th>
<th>Accessibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>E2H</td>
<td>TGE</td>
<td>NS</td>
</tr>
<tr>
<td>S3_0_C15_C2_2</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>S3_0_C15_C2_2</td>
<td>x</td>
<td>0</td>
</tr>
<tr>
<td>S3_0_C15_C2_2</td>
<td>x</td>
<td>1</td>
</tr>
</tbody>
</table>

n/a Not accessible. Executing the PE at this Exception level is not permitted.
Traps and enables

For a description of the prioritization of any generated exceptions, see *Synchronous exception prioritization* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for exceptions taken to AArch64 state. Subject to these prioritization rules, the following traps and enables are applicable when accessing this register.

ERXPFGCDN_EL1 is accessible at EL3 and can be accessible at EL1 and EL2 depending on the value of bit[5] in ACTLR_EL2 and ACTLR_EL3. See *B1.6 ACTLR_EL2, Auxiliary Control Register, EL2* on page B1-131 and *B1.7 ACTLR_EL3, Auxiliary Control Register, EL3* on page B1-133.

ERXPFGCDN_EL1 is **undefined** at EL0.

If ERXPFGCDN_EL1 is accessible at EL1 and HCR_EL2.TERR == 1, then direct reads and writes of ERXPFGCDN_EL1 at Non-secure EL1 generate a Trap exception to EL2.

If ERXPFGCDN_EL1 is accessible at EL1 or EL2 and SCR_EL3.TERR == 1, then direct reads and writes of ERXPFGCDN_EL1 at EL1 or EL2 generate a Trap exception to EL3.
B1.41 ERXPFGCTL_EL1, Selected Error Pseudo Fault Generation Control Register, EL1

Register ERXPFGCTL_EL1 accesses the ERR<n>PFGCTL register for the error record selected by ERRSELR_EL1.SEL.

If ERRSELR_EL1.SEL==0, then ERXPFGCTL_EL1 accesses the ERR0PFGCTLR register of the core error record. See B2.6 ERR0PFGCTL, Error Pseudo Fault Generation Control Register on page B2-252.

If ERRSELR_EL1.SEL==1, then ERXPFGCTL_EL1 accesses the ERR1PFGCTLR register of the DSU error record. See the Arm® DynamIQ™ Shared Unit Technical Reference Manual.

Configurations

There are no configuration notes.

Accessing the ERXPFGCTL_EL1

This register can be read using MRS with the following syntax:

MRS <Xt>,<systemreg>

This register can be written using MSR with the following syntax:

MSR <Xt>,<systemreg>

This syntax is encoded with the following settings in the instruction encoding:

<table>
<thead>
<tr>
<th>&lt;systemreg&gt;</th>
<th>op0</th>
<th>op1</th>
<th>CRn</th>
<th>CRm</th>
<th>op2</th>
</tr>
</thead>
<tbody>
<tr>
<td>S3_0_C15_C2_1</td>
<td>11</td>
<td>000</td>
<td>1111</td>
<td>0010</td>
<td>001</td>
</tr>
</tbody>
</table>

Accessibility

This register is accessible in software as follows:

<table>
<thead>
<tr>
<th>&lt;syntax&gt;</th>
<th>Control</th>
<th>Accessibility</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>E2H TGE NS EL0 EL1 EL2 EL3</td>
<td></td>
</tr>
<tr>
<td>S3_0_C15_C2_1</td>
<td>x x 0 -</td>
<td>RW n/a RW</td>
</tr>
<tr>
<td>S3_0_C15_C2_1</td>
<td>x 0 1 -</td>
<td>RW RW RW</td>
</tr>
<tr>
<td>S3_0_C15_C2_1</td>
<td>x 1 1 -</td>
<td>n/a RW RW</td>
</tr>
</tbody>
</table>

'n/a' Not accessible. The PE cannot be executing at this Exception level, so this access is not possible.
Traps and enables

For a description of the prioritization of any generated exceptions, see Synchronous exception prioritization in the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile for exceptions taken to AArch64 state. Subject to these prioritization rules, the following traps and enables are applicable when accessing this register.

ERXPFGCTL_EL1 is accessible at EL3 and can be accessible at EL1 and EL2 depending on the value of bit[5] in ACTLR_EL2 and ACTLR_EL3. See B1.6 ACTLR_EL2, Auxiliary Control Register, EL2 on page B1-131 and B1.7 ACTLR_EL3, Auxiliary Control Register, EL3 on page B1-133.

ERXPFGCTL_EL1 is undefined at EL0.

If ERXPFGCTL_EL1 is accessible at EL1 and HCR_EL2.TERR == 1, then direct reads and writes of ERXPFGCTL_EL1 at Non-secure EL1 generate a Trap exception to EL2.

If ERXPFGCTL_EL1 is accessible at EL1 or EL2 and SCR_EL3.TERR == 1, then direct reads and writes of ERXPFGCTL_EL1 at EL1 or EL2 generate a Trap exception to EL3.
**B1.42 ERXPFGF_EL1, Selected Pseudo Fault Generation Feature Register, EL1**

Register ERXPFGF_EL1 accesses the ERR<n>PFGF register for the error record selected by ERRSELR_EL1.SEL.

If ERRSELR_EL1.SEL==0, then ERXPFGF_EL1 accesses the ERR0PFGF register of the core error record. See B2.7 ERR0PFGF, Error Pseudo Fault Generation Feature Register on page B2-254.

If ERRSELR_EL1.SEL==1, then ERXPFGF_EL1 accesses the ERR1PFGFR register of the DSU error record. See the Arm® DynamIQ™ Shared Unit Technical Reference Manual.

**Configurations**

This core has no configuration notes.

**Accessing the ERXPFGF_EL1**

This register can be read using MRS with the following syntax:

\[
\text{MRS } <\text{xt}>,<\text{sysreg}>
\]

This syntax is encoded with the following settings in the instruction encoding:

<table>
<thead>
<tr>
<th>&lt;sysreg&gt;</th>
<th>op0</th>
<th>op1</th>
<th>CRn</th>
<th>CRm</th>
<th>op2</th>
</tr>
</thead>
<tbody>
<tr>
<td>S3_0_C15_C2_0</td>
<td>11</td>
<td>000</td>
<td>1111</td>
<td>0010</td>
<td>000</td>
</tr>
</tbody>
</table>

**Accessibility**

This register is accessible in software as follows:

<table>
<thead>
<tr>
<th>&lt;syntax&gt;</th>
<th>Control</th>
<th>Accessibility</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>E2H</td>
<td>TGE</td>
</tr>
<tr>
<td>S3_0_C15_C2_0</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>S3_0_C15_C2_0</td>
<td>x</td>
<td>0</td>
</tr>
<tr>
<td>S3_0_C15_C2_0</td>
<td>x</td>
<td>1</td>
</tr>
</tbody>
</table>

'n/a' Not accessible. The PE cannot be executing at this Exception level, so this access is not possible.

**Traps and enables**

For a description of the prioritization of any generated exceptions, see Synchronous exception prioritization in the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile for exceptions taken to AArch64 state. Subject to these prioritization rules, the following traps and enables are applicable when accessing this register.

ERXPFGR_EL1 is accessible at EL3 and can be accessible at EL1 and EL2 depending on the value of bit[5] in ACTLR_EL2 and ACTLR_EL3. See B1.6 ACTLR_EL2, Auxiliary Control Register; EL2 on page B1-131 and B1.7 ACTLR_EL3, Auxiliary Control Register; EL3 on page B1-133.

ERXPFGR_EL1 is undefined at EL0.

If ERXPFGR_EL1 is accessible at EL1 and HCR_EL2.TERR == 1, then direct reads and writes of ERXPFGR_EL1 at Non-secure EL1 generate a Trap exception to EL2.

If ERXPFGR_EL1 is accessible at EL1 or EL2 and SCR_EL3.TERR == 1, then direct reads and writes of ERXPFGR_EL1 at EL1 or EL2 generate a Trap exception to EL3.
B1.43 ERXSTATUS_EL1, Selected Error Record Primary Status Register, EL1

Register ERXSTATUS_EL1 accesses the ERR<n>STATUS primary status register for the error record selected by ERRSELR_EL1.SEL.

If ERRSELR_EL1.SEL==0, then ERXSTATUS_EL1 accesses the ERR0STATUS register of the core error record. See B2.8 ERR0STATUS, Error Record Primary Status Register on page B2-256.

If ERRSELR_EL1.SEL==1, then ERXSTATUS_EL1 accesses the ERR1STATUS register of the DSU error record. See the Arm® DynamIQ™ Shared Unit Technical Reference Manual.
B1.44 ESR_EL1, Exception Syndrome Register, EL1

The ESR_EL1 holds syndrome information for an exception taken to EL1.

**Bit field descriptions**

ESR_EL1 is a 32-bit register, and is part of the Exception and fault handling registers functional group.

![Figure B1-25 ESR_EL1 bit assignments](image)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EC</td>
<td>Exception Class. Indicates the reason for the exception that this register holds information about.</td>
</tr>
</tbody>
</table>
| IL    | Instruction Length for synchronous exceptions. The possible values are:  
  0  16-bit.  
  1  32-bit.  
  This field is 1 for the SError interrupt, instruction aborts, misaligned PC, Stack pointer misalignment, Data Aborts for which the ISV bit is 0, exceptions caused by an illegal instruction set state, and exceptions using the 0x00 Exception Class. |
| ISS Valid, [24] | Syndrome valid. The possible values are:  
  0  ISS not valid, ISS is RES0.  
  1  ISS valid. |
| ISS, [23:0] | Syndrome information.  
  When the EC field is 0x2F, indicating an SError interrupt has occurred, the ISS field contents are IMPLEMENTATION DEFINED. |

**Configurations**

There are no configuration notes.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*. 
### B1.45 ESR_EL2, Exception Syndrome Register, EL2

The ESR_EL2 holds syndrome information for an exception taken to EL2.

**Bit field descriptions**

ESR_EL2 is a 32-bit register, and is part of:
- The Virtualization registers functional group.
- The Exception and fault handling registers functional group.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>EC</td>
</tr>
<tr>
<td>26</td>
<td>25 24 0</td>
</tr>
</tbody>
</table>

**Figure B1-26 ESR_EL2 bit assignments**

**EC, [31:26]**

Exception Class. Indicates the reason for the exception that this register holds information about. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.

**IL, [25]**

Instruction Length for synchronous exceptions. The possible values are:

- 0 16-bit.
- 1 32-bit.

See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.

**ISS, [24:0]**

Syndrome information. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.

When reporting a virtual SEI, bits[24:0] take the value of VSESRL_EL2[24:0].

When reporting a physical SEI, the following occurs:
- IDS==0 (architectural syndrome).
- AET always reports an uncontainable error (UC) with value 0b000 or an unrecoverable error (UEU) with value 0b01.
- EA is RES0.

When reporting a synchronous Data Abort, EA is RES0.

See *B1.82 VSESR_EL2, Virtual SError Exception Syndrome Register* on page B1-239.

**Configurations**

If EL2 is not implemented, this register is RES0 from EL3.

RW fields in this register reset to architecturally **UNKNOWN** values.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*. 
B1.46 ESR_EL3, Exception Syndrome Register, EL3

The ESR_EL3 holds syndrome information for an exception taken to EL3.

**Bit field descriptions**

ESR_EL3 is a 32-bit register, and is part of the Exception and fault handling registers functional group.

| 31 | 26 25 24 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|    | EC         | ISS Valid | IL |

**Figure B1-27 ESR_EL3 bit assignments**

EC, [31:26]
Exception Class. Indicates the reason for the exception that this register holds information about.

IL, [25]
Instruction Length for synchronous exceptions. The possible values are:

- 0: 16-bit.
- 1: 32-bit.

This field is 1 for the SError interrupt, instruction aborts, misaligned PC, Stack pointer misalignment, data aborts for which the ISV bit is 0, exceptions caused by an illegal instruction set state, and exceptions using the 0x0 Exception Class.

ISS Valid, [24]
Syndrome valid. The possible values are:

- 0: ISS not valid, ISS is RES0.
- 1: ISS valid.

ISS, [23:0]
Syndrome information.

When the EC field is 0x2F, indicating an SError interrupt has occurred, the ISS field contents are IMPLEMENTATION DEFINED.

**Configurations**

RW fields in this register reset to architecturally UNKNOWN values.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.*
B1.47 HACR_EL2, Hyp Auxiliary Configuration Register, EL2

HACR_EL2 controls trapping to EL2 of IMPLEMENTATION DEFINED aspects of Non-secure EL1 or EL0 operation. This register is not used in the Cortex-A65 core.
B1.48 HCR_EL2, Hypervisor Configuration Register, EL2

The HCR_EL2 provides configuration control for virtualization, including whether various Non-secure operations are trapped to EL2.

**Bit field descriptions**

HCR_EL2 is a 64-bit register, and is part of the Virtualization registers functional group.

![HCR_EL2 bit assignments](image_url)

RES0, [63:39]

RES0 Reserved.

MIOCNCE, [38]

Mismatched Inner/Outer Cacheable Non-Coherency Enable, for the Non-secure EL1 and EL0 translation regime.

RW, [31]

RES1 Reserved.
TGE, [27]
Traps general exceptions. If this bit is set, and SCR_EL3.NS is set, then:

- All exceptions that would be routed to EL1 are routed to EL2.
- The SCTLR_EL1.M bit is treated as 0 regardless of its actual state, other than for reading the bit.
- The HCR_EL2.FMO, IMO, and AMO bits are treated as 1 regardless of their actual state, other than for reading the bits.
- All virtual interrupts are disabled.
- Any implementation defined mechanisms for signaling virtual interrupts are disabled.
- An exception return to EL1 is treated as an illegal exception return.

HCR_EL2.TGE must not be cached in a TLB.

When the value of SCR_EL3.NS is 0 the core behaves as if this field is 0 for all purposes other than a direct read or write access of HCR_EL2.

TID3, [18]
Traps ID group 3 registers. The possible values are:

0    ID group 3 register accesses are not trapped.
1    Reads to ID group 3 registers executed from Non-secure EL1 are trapped to EL2.

See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile for the registers covered by this setting.

RES0, [15]
RES0    Reserved.

Configurations
If EL2 is not implemented, this register is RES0 from EL3

RW fields in this register reset to architecturally UNKNOWN values.

Bit fields and details not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.
B1.49 ID_AA64AFR0_EL1, AArch64 Auxiliary Feature Register 0

The core does not use this register, ID_AA64AFR0_EL1 is RES0.
B1.50 ID_AA64AFR1_EL1, AArch64 Auxiliary Feature Register 1

The core does not use this register, ID_AA64AFR0_EL1 is RES0.
B1.51  ID_AA64DFR0_EL1, AArch64 Debug Feature Register 0, EL1

Provides top-level information about the debug system in AArch64.

**Bit field descriptions**

ID-AA64DFR0_EL1 is a 64-bit register, and is part of the Identification registers functional group.

This register is Read Only.

![Figure B1-29 ID_AA64DFR0_EL1 bit assignments](image)

**RES0, [63:32]**
- **RES0** Reserved.

**CTX_CMPs, [31:28]**
- Number of breakpoints that are context-aware, minus 1. These are the highest numbered breakpoints:
  - 0x1 Two breakpoints are context-aware.

**RES0, [27:24]**
- **RES0** Reserved.

**WRPs, [23:20]**
- The number of watchpoints minus 1:
  - 0x3 Four watchpoints.

**RES0, [19:16]**
- **RES0** Reserved.

**BRPs, [15:12]**
- The number of breakpoints minus 1:
  - 0x5 Six breakpoints.

**PMUVer, [11:8]**
- Performance Monitors Extension version.
  - 0x4 Performance monitor system registers implemented, PMUv3.

**TraceVer, [7:4]**
- Trace extension:
  - 0x0 Trace system registers not implemented.

**DebugVer, [3:0]**
- Debug architecture version:
  - 0x8 Armv8-A debug architecture implemented.
Configurations

ID_AA64DFR0_EL1 is architecturally mapped to external register EDDFR.

Bit fields and details that are not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.
B1.52 ID_AA64DFR1_EL1, AArch64 Debug Feature Register 1, EL1

This register is reserved for future expansion of top level information about the debug system in AArch64 state.
**B1.53 ID_AA64ISAR0_EL1, AArch64 Instruction Set Attribute Register 0, EL1**

The ID_AA64ISAR0_EL1 provides information about the instructions implemented in AArch64 state, including the instructions that are provided by the Cryptographic Extension.

**Bit field descriptions**

ID_AA64ISAR0_EL1 is a 64-bit register, and is part of the Identification registers functional group. This register is Read Only.

The optional Cryptographic Extension is not included in the base product of the core. Arm requires licensees to have contractual rights to obtain the Cryptographic Extension.

<table>
<thead>
<tr>
<th>Bit Field</th>
<th>Description</th>
<th>Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>63-48</td>
<td>RES0</td>
<td>0x0</td>
<td>Reserved.</td>
</tr>
<tr>
<td>47-44</td>
<td>DP</td>
<td>0x0</td>
<td>No Dot Product support</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x1</td>
<td>UD0T, SD0T instructions are</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>implemented.</td>
</tr>
<tr>
<td>43-32</td>
<td>RES0</td>
<td>0x0</td>
<td>Reserved.</td>
</tr>
<tr>
<td>31-28</td>
<td>RDM</td>
<td>0x1</td>
<td>SQRDMLAH and SQRDMLSH</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>instructions implemented.</td>
</tr>
<tr>
<td>27-24</td>
<td>RES0</td>
<td>0x0</td>
<td>Reserved.</td>
</tr>
<tr>
<td>23-20</td>
<td>Atomic</td>
<td>0x2</td>
<td>LDADD, LDCLR, LDEOR, LDSET,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LDSMAX, LDSMIN, LDUMAX,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LDUMIN, CAS, CASP, and SWP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>instructions are implemented.</td>
</tr>
<tr>
<td>19-16</td>
<td>CRC32</td>
<td>0x1</td>
<td>CRC32 instructions are</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>implemented.</td>
</tr>
<tr>
<td>15-12</td>
<td>SHA2</td>
<td>0x2</td>
<td>SHA2 instructions are</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>implemented.</td>
</tr>
<tr>
<td>8-7</td>
<td>SHA1</td>
<td>0x0</td>
<td>SHA1 instructions are</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>implemented.</td>
</tr>
<tr>
<td>4-3</td>
<td>AES</td>
<td>0x0</td>
<td>AES instructions are</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>implemented.</td>
</tr>
</tbody>
</table>

**Figure B1-30 ID_AA64ISAR0_EL1 bit assignments**

RES0, [63:48]  
RES0 Reserved.

DP, [47:44]  
Indicates whether Dot Product support instructions are implemented.

0x0 No Dot Product support instructions are implemented.

0x1 UD0T, SD0T instructions are implemented.

RES0, [43:32]  
RES0 Reserved.

RDM, [31:28]  
Indicates whether SQRDMLAH and SQRDMLSH instructions in AArch64 are implemented.

0x1 SQRDMLAH and SQRDMLSH instructions implemented.

RES0, [27:24]  
RES0 Reserved.

Atomic, [23:20]  
Indicates whether Atomic instructions in AArch64 are implemented. The value is:

0x2 LDADD, LDCLR, LDEOR, LDSET, LDSMAX, LDSMIN, LDUMAX, LDUMIN, CAS, CASP, and SWP instructions are implemented.

CRC32, [19:16]  
Indicates whether CRC32 instructions are implemented. The value is:

0x1 CRC32 instructions are implemented.

SHA2, [15:12]  
Indicates whether SHA2 instructions are implemented.
Indicates whether SHA2 instructions are implemented. The possible values are:

- **0x0**: No SHA2 instructions are implemented. This is the value if the core implementation does not include the Cryptographic Extension.
- **0x1**: SHA256H, SHA256H2, SHA256U0, and SHA256U1 implemented. This is the value if the core implementation includes the Cryptographic Extension.

**SHA1, [11:8]**

Indicates whether SHA1 instructions are implemented. The possible values are:

- **0x0**: No SHA1 instructions implemented. This is the value if the core implementation does not include the Cryptographic Extension.
- **0x1**: SHA1C, SHA1P, SHA1M, SHA1SU0, and SHA1SU1 implemented. This is the value if the core implementation includes the Cryptographic Extension.

**AES, [7:4]**

Indicates whether AES instructions are implemented. The possible values are:

- **0x0**: No AES instructions implemented. This is the value if the core implementation does not include the Cryptographic Extension.
- **0x2**: AESE, AESD, AESMC, and AESMC implemented, plus PMULL and PMULL2 instructions operating on 64-bit data. This is the value if the core implementation includes the Cryptographic Extension.

**[3:0]**

Reserved, RES0.

**Configurations**

ID_AA64ISAR0_EL1 is architecturally mapped to external register ID_AA64ISAR0.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.*
B1.54  ID_AA64ISAR1_EL1, AArch64 Instruction Set Attribute Register 1, EL1

The ID_AA64ISAR1_EL1 provides information about the instructions implemented in AArch64 state.

**Bit field descriptions**

ID_AA64ISAR1_EL1 is a 64-bit register, and is part of the Identification registers functional group. This register is Read Only.

![Figure B1-31  ID_AA64ISAR1_EL1 bit assignments](image)

**RES0, [63:24]**

Reserved.

**LRCPC, [23:20]**

Indicates whether load-acquire (LDA) instructions are implemented for a Release Consistent core consistent RCPC model.

- 0x1: The LDAPRB, LDAPRH, and LDAPR instructions are implemented in AArch64.

**RES0, [19:4]**

Reserved.

**DC CVAP, [3:0]**

Indicates whether Data Cache, Clean to the Point of Persistence (DC CVAP) instructions are implemented.

- 0x1: DC CVAP is supported in AArch64.

**Configurations**

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.*
B1.55  ID_AA64MMFR0_EL1, AArch64 Memory Model Feature Register 0, EL1

The ID_AA64MMFR0_EL1 provides information about the implemented memory model and memory management support in the AArch64 Execution state.

**Bit field descriptions**

ID_AA64MMFR0_EL1 is a 64-bit register, and is part of the Identification registers functional group. This register is Read Only.

```
+----------------+----------------+----------------+----------------+----------------+----------------+----------------+----------------+----------------+----------------+
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>RES0</td>
<td>TGran4</td>
<td>TGran64</td>
<td>TGran16</td>
<td>SNSMem</td>
<td>BigEnd</td>
<td>ASIDBits</td>
<td>PARange</td>
<td>BigEnd</td>
<td>ASIDBits</td>
</tr>
</tbody>
</table>
+----------------+----------------+----------------+----------------+----------------+----------------+----------------+----------------+----------------+----------------+
```

**RES0, [63:32]**

RES0 Reserved.

**TGran4, [31:28]**

Support for 4KB memory translation granule size:

- 0x0 4KB granule supported.

**TGran64, [27:24]**

Support for 64KB memory translation granule size:

- 0x0 64KB granule supported.

**TGran16, [23:20]**

Support for 16KB memory translation granule size:

- 0x1 Indicates that the 16KB granule is supported.

**RES0, [19:16]**

RES0 Reserved.

**SNSMem, [15:12]**

Secure versus Non-secure Memory distinction:

- 0x1 Supports a distinction between Secure and Non-secure Memory.

**BigEnd, [11:8]**

Mixed-endian configuration support:

- 0x1 Mixed-endian support. The SCTRL_ELx.EE and SCTRL_EL1.E0E bits can be configured.

**ASIDBits, [7:4]**

Number of ASID bits:

- 0x2 16 bits.

**PARange, [3:0]**
Physical address range supported:

Ωx4 44 bits, 16TB.

The supported Physical Address Range is 44 bits. Other cores in the DSU might support a different Physical Address Range.

Configurations

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.
B1.56  **ID_AA64MMFR1_EL1, AArch64 Memory Model Feature Register 1, EL1**

The ID_AA64MMFR1_EL1 provides information about the implemented memory model and memory management support in the AArch64 Execution state.

**Bit field descriptions**

ID_AA64MMFR1_EL1 is a 64-bit register, and is part of the Identification registers functional group. This register is Read Only.

![Figure B1-33  ID_AA64MMFR1_EL1 bit assignments](image)

<table>
<thead>
<tr>
<th>Field</th>
<th>Width</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RES0</td>
<td>[63:32]</td>
<td>Reserved.</td>
</tr>
<tr>
<td>XNX</td>
<td>[31:28]</td>
<td>Indicates whether provision of EL0 vs EL1 execute never control at Stage 2 is supported. 0x1 EL0/EL1 execute control distinction at Stage 2 bit is supported. All other values are reserved.</td>
</tr>
<tr>
<td>SpecSEI</td>
<td>[27:24]</td>
<td>Describes whether the PE can generate SError interrupt exceptions from speculative reads of memory, including speculative instruction fetches. 0x0 The PE never generates an SError interrupt due to an external abort on a speculative read.</td>
</tr>
<tr>
<td>PAN</td>
<td>[23:20]</td>
<td>Privileged Access Never. Indicates support for the PAN bit in PSTATE, SPSR_EL1, SPSR_EL2, SPSR_EL3, and DSPSR_EL0. 0x2 PAN supported and AT S1E1RP and AT S1E1WP instructions supported.</td>
</tr>
<tr>
<td>LO</td>
<td>[19:16]</td>
<td>Indicates support for LORegions. 0x1 LORegions are supported.</td>
</tr>
<tr>
<td>HD</td>
<td>[15:12]</td>
<td>Presence of Hierarchical Disables. Enables an operating system or hypervisor to hand over up to 4 bits of the last level page table descriptor (bits[62:59] of the page table entry) for use by hardware for IMPLEMENTATION DEFINED usage. The value is: 0x2 Hierarchical Permission Disables and Hardware allocation of bits[62:59] supported.</td>
</tr>
<tr>
<td>VH</td>
<td>[11:8]</td>
<td>Indicates whether Virtualization Host Extensions are supported.</td>
</tr>
</tbody>
</table>

---

**RES0, [63:32]**

RES0  Reserved.

**XNX, [31:28]**

Indicates whether provision of EL0 vs EL1 execute never control at Stage 2 is supported.

- **0x1**: EL0/EL1 execute control distinction at Stage 2 bit is supported. All other values are reserved.

**SpecSEI, [27:24]**

Describes whether the PE can generate SError interrupt exceptions from speculative reads of memory, including speculative instruction fetches.

- **0x0**: The PE never generates an SError interrupt due to an external abort on a speculative read.

**PAN, [23:20]**

Privileged Access Never. Indicates support for the PAN bit in PSTATE, SPSR_EL1, SPSR_EL2, SPSR_EL3, and DSPSR_EL0.

- **0x2**: PAN supported and AT S1E1RP and AT S1E1WP instructions supported.

**LO, [19:16]**

Indicates support for LORegions.

- **0x1**: LORegions are supported.

**HD, [15:12]**

Presence of Hierarchical Disables. Enables an operating system or hypervisor to hand over up to 4 bits of the last level page table descriptor (bits[62:59] of the page table entry) for use by hardware for IMPLEMENTATION DEFINED usage. The value is:

- **0x2**: Hierarchical Permission Disables and Hardware allocation of bits[62:59] supported.

**VH, [11:8]**

Indicates whether Virtualization Host Extensions are supported.
0x1 Virtualization Host Extensions supported.

**VMID, [7:4]**

Indicates the number of VMID bits supported.

0x2 16 bits are supported.

**HAFDBS, [3:0]**

Indicates the support for hardware updates to Access flag and dirty state in translation tables.

0x2 Hardware update of both the Access flag and dirty state is supported in hardware.

**Configurations**

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.*
B1.57  ID_AA64MMFR2_EL1, AArch64 Memory Model Feature Register 2, EL1

The ID_AA64MMFR2_EL1 provides information about the implemented memory model and memory management support in the AArch64 Execution state.

**Bit field descriptions**

ID_AA64MMFR2_EL1 is a 64-bit register, and is part of the Identification registers functional group. This register is Read Only.

![Bit field assignments](image)

**RES0, [63:16]**

RES0  Reserved.

**IESB, [15:12]**

Indicates whether an implicit Error Synchronization Barrier has been inserted. The value is:

0x0  SCTLR_ELx.IESB implicit Error Synchronization Barrier control implemented.

**LSM, [11:8]**

Indicates whether STM ordering control bits are supported. The value is:

0x0  LSMAOE and nTLSMD bit not supported.

**UAO, [7:4]**

Indicates the presence of the *User Access Override* (UAO). The value is:

0x1  UAO is supported.

**CnP, [3:0]**

Common not Private. Indicates whether a TLB entry is pointed at a translation table base register that is a member of a common set. The value is:

0x1  CnP bit is supported.

**Configurations**

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.
B1.58  ID_AA64PFR0_EL1, AArch64 Processor Feature Register 0, EL1

The ID_AA64PFR0_EL1 provides additional information about implemented core features in AArch64.

The optional Advanced SIMD and floating-point support is not included in the base product of the core. Arm requires licensees to have contractual rights to obtain the Advanced SIMD and floating-point support.

**Bit field descriptions**

ID_AA64PFR0_EL1 is a 64-bit register, and is part of the Identification registers functional group. This register is Read Only.

<table>
<thead>
<tr>
<th>Bit Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>63-60</td>
<td>CSV3</td>
</tr>
<tr>
<td>59-56</td>
<td>CSV2</td>
</tr>
<tr>
<td>55-32</td>
<td>RES0</td>
</tr>
<tr>
<td>31-28</td>
<td>RAS</td>
</tr>
<tr>
<td>27-24</td>
<td>GIC</td>
</tr>
<tr>
<td>23-20</td>
<td>AdvSIMD</td>
</tr>
<tr>
<td>19-16</td>
<td>FP</td>
</tr>
<tr>
<td>15-12</td>
<td>EL3 handling</td>
</tr>
<tr>
<td>11-8</td>
<td>EL2 handling</td>
</tr>
<tr>
<td>7-4</td>
<td>EL1 handling</td>
</tr>
<tr>
<td>3-0</td>
<td>EL0 handling</td>
</tr>
</tbody>
</table>

**CSV3, [63:60]**

0x1  Data loaded under speculation with a permission or domain fault cannot be used to form an address or generate condition codes to be used by instructions newer than the load in the speculative sequence. This is the reset value.

All other values reserved.

**CSV2, [59:56]**

0x1  Branch targets trained in one context cannot affect speculative execution in a different hardware described context. This is the reset value.

All other values reserved.

**RES0, [55:32]**

RES0  Reserved.

**RAS, [31:28]**

RAS extension version. The possible values are:

0x1  Version 1 of the RAS extension is present.

**GIC, [27:24]**

GIC CPU interface:

0x0  GIC CPU interface is disabled, GICCDISABLE is HIGH, or not implemented.

0x1  GIC CPU interface is implemented and enabled, GICCDISABLE is LOW.

**AdvSIMD, [23:20]**

Advanced SIMD. The possible values are:

0x1  Advanced SIMD, including Half-precision support, is implemented.

0xF  Advanced SIMD is not implemented.

The FP and AdvSIMD both take the same value, as both must be implemented, or neither.
FP, [19:16]
Floating-point. The possible values are:
0x1 Floating-point, including Half-precision support, is implemented.
0xf Floating-point is not implemented.
The FP and AdvSIMD both take the same value, as both must be implemented, or neither.

EL3 handling, [15:12]
EL3 exception handling:
0x1 Instructions can be executed at EL3 in AArch64 state only.

EL2 handling, [11:8]
EL2 exception handling:
0x1 Instructions can be executed at EL3 in AArch64 state only.

EL1 handling, [7:4]
EL1 exception handling. The possible values are:
0x1 Instructions can be executed at EL3 in AArch64 state only.

EL0 handling, [3:0]
EL0 exception handling. The possible values are:
0x1 Instructions can be executed at EL3 in AArch64 state only.

Configurations
ID_AA64PFR0_EL1 is architecturally mapped to External register EDPFR.
Bit fields and details that are not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.
B1.59 ID_AA64PFR1_EL1, AArch64 Processor Feature Register 1, EL1

The ID_AA64PFR1_EL1 provides additional information about implemented core features in AArch64.

Bit field descriptions

ID_AA64PFR1_EL1 is a 64-bit register, and is part of the Identification registers functional group. This register is Read Only.

RES0, [63:8]

RES0 Reserved.

SSBS, [7:4]

AArch64 provides the PSTATE.SSBS mechanism to mark regions that are Speculative Store Bypassing Safe (SSBS).

0x01 The MSR/MRS instructions are not implemented to directly read and write the PSTATE.SSBS field.

RES0, [3:0]

RES0 Reserved.

Configurations

There are no configuration notes.

Bit fields and details not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.
B1.60 LORC_EL1, LORegion Control Register, EL1

The LORC_EL1 register enables and disables LORegions, and selects the current LORegion descriptor.

Bit field descriptions
LORC_EL1 is a 64-bit register and is part of the Virtual memory control registers functional group.

```
+-------------------+-------------------+-------------------+-------------------+
| 63               | 4                 | 3                 | 2                 |
| RES0             | [1] EN,           | [0]               |
+-------------------+-------------------+-------------------+

Figure B1-37 LORC_EL1 bit assignments
```

[63:4]
Reserved, RES0.

DS, [3:2]
Descriptor Select. Number that selects the current LORegion descriptor accessed by the LORSA_EL1, LOREA_EL1, and LORN_EL1 registers.

[1]
Reserved, RES0.

EN, [0]
Enable. The possible values are:

0  Disabled. This is the reset value.
1  Enabled.

Configurations
RW fields in this register reset to architecturally UNKNOWN values.

Bit fields and details that are not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.
B1.61 LORID_EL1, LORegion ID Register, EL1

The LORID_EL1 ID register indicates the supported number of LORegions and LORegion descriptors.

**Bit field descriptions**

LORID_EL1 is a 64-bit register.

![Figure B1-38 LORID_EL1 bit assignments](image)

<table>
<thead>
<tr>
<th>Bit Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[63:24]</td>
<td>Reserved, RESO.</td>
</tr>
<tr>
<td>LD, [23:16]</td>
<td>Number of LORegion descriptors supported by the implementation, expressed as binary 8-bit number. The value is: 0x04 Four LORegion descriptors are supported.</td>
</tr>
<tr>
<td>[15:8]</td>
<td>Reserved, RESO.</td>
</tr>
<tr>
<td>LR, [7:0]</td>
<td>Number of LORegions supported by the implementation, expressed as a binary 8-bit number. The value is: 0x04 Four LORegions are supported.</td>
</tr>
</tbody>
</table>

**Configurations**

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.*
**B1.62  LORN_EL1, LORegion Number Register, EL1**

The LORN_EL1 register holds the number of the LORegion described in the current LORegion descriptor selected by LORC_EL1.DS.

**Bit field descriptions**

LORN_EL1 is a 64-bit register and is part of the Virtual memory control registers functional group.

![Figure B1-39  LORN_EL1 bit assignments](image)

- **[63:2]** Reserve, RESO.
- **Num, [1:0]**
  Indicates the LORegion number.

**Configurations**

RW fields in this register reset to architecturally **UNKNOWN** values.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*. 
B1.63   **MDCR_EL3, Monitor Debug Configuration Register, EL3**

The MDCR_EL3 provides configuration options for Security to self-hosted debug.

**Bit field descriptions**
MDCR_EL3 is a 32-bit register, and is part of:
- The Debug registers functional group.
- The Security registers functional group.

![Figure B1-40  MDCR_EL3 bit assignments](image)

### EPMAD, [21]
External debugger access to Performance Monitors registers disabled. This disables access to these registers by an external debugger. The possible values are:

- 0: Access to Performance Monitors registers from external debugger is permitted.
- 1: Access to Performance Monitors registers from external debugger is disabled, unless overridden by authentication interface.

### EDAD, [20]
External debugger access to breakpoint and watchpoint registers disabled. This disables access to these registers by an external debugger. The possible values are:

- 0: Access to breakpoint and watchpoint registers from external debugger is permitted.
- 1: Access to breakpoint and watchpoint registers from external debugger is disabled, unless overridden by authentication interface.

### SPME, [17]
Secure performance monitors enable. This enables event counting exceptions from Secure state. The possible values are:

- 0: Event counting prohibited in Secure state.
- 1: Event counting allowed in Secure state.

### SPD32, [15:14]
RES0
Reserved.

### TDOSA, [10]
Trap accesses to the OS debug system registers, OSLAR_EL1, OSLSR_EL1, OSDLR_EL1, and DBGPRCR_EL1 OS.

- 0: Accesses are not trapped.
Accesses to the OS debug system registers are trapped to EL3.
The reset value is \textit{UNKNOWN}.

\textbf{TDA, [9]}

Trap accesses to the remaining sets of debug registers to EL3.

\begin{itemize}
  \item 0 \hspace{1cm} Accesses are not trapped.
  \item 1 \hspace{1cm} Accesses to the remaining debug system registers are trapped to EL3.
\end{itemize}

The reset value is \textit{UNKNOWN}.

\textbf{Configurations}

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the \textit{Arm\textsuperscript{\textregistered} Architecture Reference Manual Armv8, for Armv8-A architecture profile}. 
B1.64 MIDR_EL1, Main ID Register, EL1

The MIDR_EL1 provides identification information for the core, including an implementer code for the device and a device ID number.

Bit field descriptions

MIDR_EL1 is a 32-bit register, and is part of the Identification registers functional group.

This register is Read Only.

<table>
<thead>
<tr>
<th>31</th>
<th>24</th>
<th>20</th>
<th>16</th>
<th>15</th>
<th>4</th>
<th>3</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implementer</td>
<td>Variant</td>
<td>Architecture</td>
<td>PartNum</td>
<td>Revision</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure B1-41 MIDR_EL1 bit assignments

Implementer, [31:24]

Indicates the implementer code. This value is:

\(0x41\) ASCII character 'A' - implementer is Arm Limited.

Variant, [23:20]

Indicates the variant number of the core. This is the major revision number \(x\) in the \(rxp\) description of the product revision status. This value is:

\(0x1\) \(r1p1\).

Architecture, [19:16]

Indicates the architecture code. This value is:

\(0xF\) Defined by CPUID scheme.

PartNum, [15:4]

Indicates the primary part number. This value is:

\(0x0D06\) Cortex-A65 core.

Revision, [3:0]

Indicates the minor revision number of the core. This is the minor revision number \(y\) in the \(py\) part of the \(rxpy\) description of the product revision status. This value is:

\(0x1\) \(r1p1\).

Configurations

The MIDR_EL1 is architecturally mapped to external MIDR_EL1 register.

Bit fields and details that are not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.
B1.65  MPIDR_EL1, Multiprocessor Affinity Register, EL1

The MPIDR_EL1 provides an additional core identification mechanism for scheduling purposes in a cluster.

**Bit field descriptions**

MPIDR_EL1 is a 64-bit register, and is part of the Other system control registers functional group. This register is Read Only.

![MPIDR_EL1 bit assignments](image)

**RES0, [63:40]**

RES0  Reserved.

**Aff3, [39:32]**

Affinity level 3. Highest level affinity field.

**CLUSTERID**

Indicates the value read in the CLUSTERIDAFF3 configuration signal.

**RES1, [31]**

RES1  Reserved.

**U, [30]**

Indicates a single core system, as distinct from core 0 in a cluster. This value is:

0  Core is part of a multiprocessor system. This is the value for implementations with more than one core, and for implementations with an ACE or CHI master interface.

**RES0, [29:25]**

RES0  Reserved.

**MT, [24]**

Indicates whether the lowest level of affinity consists of logical cores that are implemented using a multithreading type approach. This value is:

1  Performance of PEs at the lowest affinity level is very interdependent.

Affinity0 represents threads, Cortex-A65 is multithreaded.

**Aff2, [23:16]**

Affinity level 2. Second highest level affinity field.

**CLUSTERID**

Indicates the value read in the CLUSTERIDAFF2 configuration signal.
**Aff1, [15:11]**
Part of Affinity level 1. Third highest level affinity field.

**RAZ**
Read-As-Zero.

**Aff1, [10:8]**
Part of Affinity level 1. Third highest level affinity field.

**CPUID**
Identification number for each CPU in the cluster:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>MP1: CPUID: 0. to</td>
</tr>
<tr>
<td>0x7</td>
<td>MP8: CPUID: 7.</td>
</tr>
</tbody>
</table>

**Aff0, [7:0]**
Affinity level 0. The level identifies individual threads within a multithreaded core. The Cortex-A65 core is dual-threaded, so this field has the value:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>Thread 0.</td>
</tr>
<tr>
<td>0x01</td>
<td>Thread 1.</td>
</tr>
</tbody>
</table>

**Configurations**

MPIDR_EL1[31:0] is mapped to external register EDDEVAFF0.

MPIDR_EL1[63:32] is mapped to external register EDDEVAFF1.

Bit fields and details that are not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.
B1.66 PAR_EL1, Physical Address Register, EL1

The PAR_EL1 returns the output address from an address translation instruction that executed successfully, or fault information if the instruction did not execute successfully.

**Bit field descriptions, PAR_EL1.F is 0**

The following figure shows the PAR bit assignments when PAR.F is 0.

![Diagram of PAR bit assignments]

**IMP DEF, [10]**

IMPLEMENTATION DEFINED. Bit[10] is RES0.

**F, [0]**

Indicates whether the instruction performed a successful address translation.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Address translation completed successfully.</td>
</tr>
<tr>
<td>1</td>
<td>Address translation aborted.</td>
</tr>
</tbody>
</table>

**Configurations**

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.

**Bit field descriptions, PAR_EL1.F is 1**

See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.
B1.67  **REVIDR_EL1, Revision ID Register, EL1**

The REVIDR_EL1 provides revision information, additional to MIDR_EL1, that identifies minor fixes (errata) which might be present in a specific implementation of the Cortex-A65 core.

**Bit field descriptions**

REVIDR_EL1 is a 32-bit register, and is part of the Identification registers functional group. This register is Read Only.

![Figure B1-44  REVIDR_EL1 bit assignments](image)

**Configurations**

There are no configuration notes. Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.*
B1.68 RMR_EL3, Reset Management Register

The RMR_EL3 controls the execution state that the core boots into and allows request of a Warm reset.

**Bit field descriptions**

RMR_EL3 is a 32-bit register, and is part of the Reset management registers functional group.

![Figure B1-45  RMR_EL3 bit assignments](image)

**RES0, [31:2]**

RES0  Reserved.

**RR, [1]**

Reset Request. The possible values are:

0  This is the reset value on both a Warm and a Cold reset.
1  Requests a Warm reset.

The bit is strictly a request.

**RES1, [0]**

RES1  Reserved.

**Configurations**

There are no configuration notes.

Details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*. 
B1.69 RVBAR_EL3, Reset Vector Base Address Register, EL3

RVBAR_EL3 contains the implementation defined address that execution starts from after reset.

Bit field descriptions

RVBAR_EL3 is a 64-bit register, and is part of the Reset management registers functional group.

This register is Read Only.

| 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|

Reset Vector Base Address

Figure B1-46 RVBAR_EL3 bit assignments

RVBA, [63:0]

Reset Vector Base Address. The address that execution starts from after reset. Bits[1:0] of this register are 0b00, as this address must be aligned, and bits [63:48] are 0x0000 because the address must be within the physical address size supported by the core.

Configurations

There are no configuration notes.

Bit fields and details not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.
B1.70  **SCTLR_EL1, System Control Register, EL1**

The SCTLR_EL1 provides top-level control of the system, including its memory system, at EL1 and EL0.

**Bit field descriptions**

SCTLR_EL1 is a 64-bit register, and is part of the Other system control registers functional group. This register resets to $0x0000000030D50838$.

![Figure B1-47  SCTLR_EL1 bit assignments](image)

### RES0, [63:45]

- **RES0**: Reserved.

### DSSBS, [44]

- **DSSBS** is used to set the new PSTATE bit, SSBS (Speculative Store Bypassing Safe).
  - $0x0$: PSTATE.SSBS is set to 0 on an exception taken to this Exception level. This is the reset value.
  - $0x1$: PSTATE.SSBS is set to 1 on an exception taken to this Exception level.

### RES0, [43:30]

- **RES0**: Reserved.

### RES1, [29:28]

- **RES1**: Reserved.

### RES0, [27]

- **RES0**: Reserved.

### EE, [25]

Exception endianness. The value of this bit controls the endianness for explicit data accesses at EL1. This value also indicates the endianness of the translation table data for translation table lookups. The possible values of this bit are:

- $0$: Little-endian.
- $1$: Big-endian.
ITD, [7]
   This field is RAZ/WI.

RES0, [6]
   RES0   Reserved.

CP15BEN, [5]
   CP15 barrier enable. The possible values are:
      0    CP15 barrier operations disabled. Their encodings are UNDEFINED.
      1    CP15 barrier operations enabled.

M, [0]
   MMU enable. The possible values are:
      0    EL1 and EL0 stage 1 MMU disabled.
      1    EL1 and EL0 stage 1 MMU enabled.

Configurations
   There are no configuration notes.

   Bit fields and details that are not provided in this description are architecturally defined. See the
B1.71  SCTLR_EL2, System Control Register, EL2

The SCTLR_EL2 provides top-level control of the system, including its memory system at EL2.

Bit field descriptions
SCTLR_EL2 is a 64-bit register, and is part of:
- The Virtualization registers functional group.
- The Other system control registers functional group.

![Figure B1-48 SCTLR_EL2 bit assignments]

This register resets to 0x30C50838.

DSSBS, [44]
DSSBS is used to set the new PSTATE bit, SSBS (Speculative Store Bypassing Safe).
SCTLR_EL2.DSSBS is held in bit[44] regardless of the value of HCR_EL2.E2H or HCR_EL2.TGE.
0x0  PSTATE.SSBS is set to 0 on an exception taken to this Exception level. This is the reset value.
0x1  PSTATE.SSBS is set to 1 on an exception taken to this Exception level.

Configurations
If EL2 is not implemented, this register is RES0 from EL3.

Bit fields and details that are not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.
B1.72  **SCTLR_EL3, System Control Register, EL3**

The SCTLR_EL3 provides top-level control of the system, including its memory system at EL3.

**Bit field descriptions**

SCTLR_EL3 is a 64-bit register, and is part of the Other system control registers functional group. This register resets to 0x30C50838.

<table>
<thead>
<tr>
<th>Bit Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[63:45]</td>
<td>RES0</td>
</tr>
<tr>
<td>[44]</td>
<td>DSSBS</td>
</tr>
<tr>
<td>[29:28]</td>
<td>RES1</td>
</tr>
<tr>
<td>[27:26]</td>
<td>RES0</td>
</tr>
<tr>
<td>[25]</td>
<td>EE</td>
</tr>
<tr>
<td>[12]</td>
<td>I</td>
</tr>
</tbody>
</table>

**RES0, [63:45]**

Reserved.

**DSSBS, [44]**

DSSBS is used to set the new PSTATE bit, SSBS (Speculative Store Bypassing Safe).

- 0x0: PSTATE.SSBS is set to 0 on an exception taken to this Exception level. This is the reset value.
- 0x1: PSTATE.SSBS is set to 1 on an exception taken to this Exception level.

**RES0, [43:30]**

Reserved.

**RES1, [29:28]**

Reserved.

**RES0, [27:26]**

Reserved.

**EE, [25]**

Exception endianness. This bit controls the endianness for:
- Explicit data accesses at EL3.
- Stage 1 translation table walks at EL3.

The possible values are:

- 0x0: Little endian.
- 0x1: Big endian.

The reset value is determined by the CFGEND configuration signal.

**I, [12]**

Global instruction cache enable. The possible values are:
Instruction caches disabled. This is the reset value.

0x1  Instruction caches enabled.

\textbf{C, [2]}

Global enable for data and unifies caches. The possible values are:

0x0  Disables data and unified caches. This is the reset value.

0x1  Enables data and unified caches.

\textbf{M, [0]}

Global enable for the EL3 MMU. The possible values are:

0x0  Disables EL3 MMU. This is the reset value.

0x1  Enables EL3 MMU.

\textbf{Configurations}

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the \textit{Arm\textsuperscript{\textregistered} Architecture Reference Manual Armv8, for Armv8-A architecture profile.}
**B1.73 TCR_EL1, Translation Control Register, EL1**

The TCR_EL1 determines which Translation Base registers define the base address register for a translation table walk required for stage 1 translation of a memory access from EL0 or EL1 and holds cacheability and shareability information.

**Bit field descriptions**

TCR_EL1 is a 64-bit register, and is part of the Virtual memory control registers functional group.

![Figure B1-50 TCR_EL1 bit assignments](image)

**HD, [40]**

Hardware management of dirty state in stage 1 translations from EL0 and EL1. The possible values are:

- **0** Stage 1 hardware management of dirty state disabled.
- **1** Stage 1 hardware management of dirty state enabled, only if the HA bit is also set to 1.

**HA, [39]**

Hardware Access flag update in stage 1 translations from EL0 and EL1. The possible values are:

- **0** Stage 1 Access flag update disabled.
- **1** Stage 1 Access flag update enabled.

**IPS, [34:32]**

Physical address size. The possible values are:

- **0b000** 32 bits, 4GB.
- **0b001** 36 bits, 64GB.
- **0b010** 40 bits, 1TB.
- **0b011** 42 bits, 4TB.
- **0b100** 44 bits, 16TB.

Other values are reserved.

**Configurations**

RW fields in this register reset to **UNKNOWN** values.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.
B1.74 TCR_EL2, Translation Control Register, EL2

The TCR_EL2 controls translation table walks required for stage 1 translation of a memory access from EL2 and holds cacheability and shareability information.

**Bit field descriptions**

TCR_EL2 is a 64-bit register.

TCR_EL2 is part of:
- The Virtual memory control registers functional group.
- The Hypervisor and virtualization registers functional group.

![TCR_EL2 bit assignments](image)

**HD, [22]**

Dirty bit update. The possible values are:

- 0 Dirty bit update is disabled.
- 1 Dirty bit update is enabled.

**HA, [21]**

Stage 1 Access flag update. The possible values are:

- 0 Stage 1 Access flag update is disabled.
- 1 Stage 1 Access flag update is enabled.

**PS, [18:16]**

Physical address size. The possible values are:

- \(0\bar{b}000\) 32 bits, 4GB.
- \(0\bar{b}001\) 36 bits, 64GB.
- \(0\bar{b}010\) 40 bits, 1TB.
- \(0\bar{b}011\) 42 bits, 4TB.
- \(0\bar{b}100\) 44 bits, 16TB.

Other values are reserved.

**Configurations**

When the Virtualization Host Extension is activated, TCR_EL2 has the same bit assignments as TCR_EL1.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*. 
B1.75 TCR_EL3, Translation Control Register, EL3

The TCR_EL3 controls translation table walks required for stage 1 translation of memory accesses from EL3 and holds cacheability and shareability information for the accesses.

Bit field descriptions

TCR_EL3 is a 32-bit register, and is part of the Virtual memory control registers functional group.

![Figure B1-52 TCR_EL3 bit assignments](image)

**HPD**, [24]
Hierarchical Permission Disables. This affects the hierarchical control bits, APTable, PXNTable, and UXNTable, except NSTable, in the translation tables pointed to by TTBR0_EL3. The possible values are:

0  Hierarchical Permissions are enabled.
1  Hierarchical Permissions are disabled.

--- Note ---
In this case bit[61] (APTable[0]) and bit[59] (PXNTable) of the next level descriptor attributes are required to be to be ignored by the PE, and are no longer reserved, allowing them to be used by software.

---

**HD**, [22]
Hardware management of dirty state in stage 1 translations from EL3. The possible values are:

0  Stage 1 hardware management of dirty state disabled.
1  Stage 1 hardware management of dirty state enabled, only if the HA bit is also set to 1.

Implementation of this bit is OPTIONAL, and, if not implemented, this bit is RES0.

**HA**, [21]
Hardware Access flag update in stage 1 translations from EL3. The possible values are:

0  Stage 1 Access flag update disabled.
1  Stage 1 Access flag update enabled.

**PS**, [18:16]
Physical address size. The possible values are:

- `0b00` 32 bits, 4GB.
- `0b01` 36 bits, 64GB.
- `0b10` 40 bits, 1TB.
- `0b11` 42 bits, 4TB.
0b100 44 bits, 16TB.

Other values are reserved.

**TG0, [15:14]**

TTBR0_EL3 granule size. The possible values are:

- 0b00 4KB.
- 0b10 16KB.
- 0b01 64KB.
- 0b11 Reserved.

All other values are not supported.

**SH0, [13:12]**

Shareability attribute for memory associated with translation table walks using TTBR0_EL3.

The possible values are:

- 0b00 Non-shareable.
- 0b01 Reserved.
- 0b10 Outer shareable.
- 0b11 Inner shareable.

Bit fields and details not provided in this description are architecturally defined. See the *Arm*®
The TTBR0_EL1 holds the base address of translation table 0, and information about the memory it occupies. This is one of the translation tables for the stage 1 translation of memory accesses from modes other than Hyp mode.

**Bit field descriptions**

TTBR0_EL1 is 64-bit register.

<table>
<thead>
<tr>
<th>Bit Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASID[63:48]</td>
<td>An ASID for the translation table base address. The TCR_EL1.A1 field selects either TTBR0_EL1.ASID or TTBR1_EL1.ASID.</td>
</tr>
<tr>
<td>BADDR[47:x][47:1]</td>
<td>Translation table base address, bits[47:x]. Bits [x-1:1] are RES0. x is based on the value of TCR_EL1.T0SZ, the stage of translation, and the memory translation granule size. For instructions on how to calculate it, see the <em>Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile</em>. The value of x determines the required alignment of the translation table, that must be aligned to $2^x$ bytes. If bits [x-1:1] are not all zero, this is a misaligned translation table base address. Its effects are CONSTRAINED UNPREDICTABLE, where bits [x-1:1] are treated as if all the bits are zero. The value read back from those bits is the value written.</td>
</tr>
<tr>
<td>CnP[0]</td>
<td>Common not Private. The possible values are: 0 CnP is not supported. 1 CnP is supported.</td>
</tr>
</tbody>
</table>

**Configurations**

There are no configuration notes. Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*. 

---

Non-Confidential
B1.77 TTBR0_EL2, Translation Table Base Register 0, EL2

The TTBR0_EL2 holds the base address of the translation table for the stage 1 translation of memory accesses from EL2.

Bit field descriptions

TTBR0_EL2 is a 64-bit register, and is part of the Virtual memory control registers functional group.

![TTBR0_EL2 bit assignments](image)

RES0, [63:48]

RES0 Reserved.

BADDR, [47:1]

Translation table base address, bits[47:x]. Bits [x-1:1] are RES0.

x is based on the value of TCR_EL2.T0SZ, the stage of translation, and the memory translation granule size.

For instructions on how to calculate it, see the Arm® Architecture Reference Manual Arm®v8, for Arm®v8-A architecture profile.

The value of x determines the required alignment of the translation table, that must be aligned to $2^x$ bytes.

If bits [x-1:1] are not all zero, this is a misaligned translation table base address. Its effects are CONSTRAINED UNPREDICTABLE, where bits [x-1:1] are treated as if all the bits are zero. The value read back from those bits is the value written.

CnP, [0]

Common not Private. The possible values are:

0  CnP is not supported.
1  CnP is supported.

Configurations

When the Virtualization Host Extension is activated, TTBR0_EL2 has the same bit assignments as TTBR0_EL1.

Bit fields and details that are not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.
B1.78 TTBR0_EL3, Translation Table Base Register 0, EL3

The TTBR0_EL3 holds the base address of the translation table for the stage 1 translation of memory accesses from EL3.

**Bit field descriptions**

TTBR0_EL3 is a 64-bit register.

![Figure B1-55  TTBR0_EL3 bit assignments](image)

<table>
<thead>
<tr>
<th>Bit field description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[63:48]</td>
<td>Reserved, RES0.</td>
</tr>
<tr>
<td>BADDR[47:x], [47:1]</td>
<td>Translation table base address, bits[47:x]. Bits [x-1:1] are RES0. x is based on the value of TCR_EL1.T0SZ, the stage of translation, and the memory translation granule size. For instructions on how to calculate it, see the <em>Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile</em>. The value of x determines the required alignment of the translation table, that must be aligned to $2^x$ bytes. If bits [x-1:1] are not all zero, this is a misaligned translation table base address. Its effects are CONSTRAINED UNPREDICTABLE, where bits [x-1:1] are treated as if all the bits are zero. The value read back from those bits is the value written.</td>
</tr>
</tbody>
</table>

**CnP, [0]**

Common not Private. The possible values are:

0 CnP is not supported.

1 CnP is supported.

**Configurations**

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*. 

---

100439_0101_00_en Copyright © 2018, 2019 Arm Limited or its affiliates. All rights reserved. Non-Confidential
B1.79 TTBR1_EL1, Translation Table Base Register 1, EL1

The TTBR1_EL1 holds the base address of translation table 1, and information about the memory it occupies. This is one of the translation tables for the stage 1 translation of memory accesses at EL0 and EL1.

Bit field descriptions

TTBR1_EL1 is a 64-bit register.

<table>
<thead>
<tr>
<th>63</th>
<th>48</th>
<th>47</th>
<th>BADDR[47:x]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASID</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CnP</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure B1-56 TTBR1_EL1 bit assignments

ASID, [63:48]

An ASID for the translation table base address. The TCR_EL1.A1 field selects either TTBR0_EL1.ASID or TTBR1_EL1.ASID.

BADDR[47:x], [47:1]

Translation table base address, bits[47:x]. Bits [x-1:0] are RES0.

x is based on the value of TCR_EL1.T0SZ, the stage of translation, and the memory translation granule size.

For instructions on how to calculate it, see the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.

The value of x determines the required alignment of the translation table, that must be aligned to $2^x$ bytes.

If bits [x-1:1] are not all zero, this is a misaligned Translation Table Base Address. Its effects are CONSTRAINED UNPREDICTABLE, where bits [x-1:1] are treated as if all the bits are zero. The value read back from those bits is the value written.

CnP, [0]

Common not Private. The possible values are:

0 CnP is not supported.
1 CnP is supported.

Configurations

There are no configuration notes.

Bit fields and details not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.
B1.80 TTBR1_EL2, Translation Table Base Register 1, EL2

TTBR1_EL2 has the same format and contents as TTBR1_EL1.

See B.79 TTBR1_EL1, Translation Table Base Register 1, EL1 on page B1-236.
**B1.81 VDISR_EL2, Virtual Deferred Interrupt Status Register, EL2**

The VDISR_EL2 records that a virtual SError interrupt has been consumed by an ESB instruction executed at Non-secure EL1.

**Bit field descriptions**

VDISR_EL2 is a 64-bit register, and is part of the Reliability, Availability, Serviceability (RAS) registers functional group.

**Configurations**

See B1.81.1 VDISR_EL2 at EL1 using AArch64 on page B1-238.

Bit fields and details not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.

This section contains the following subsection:

- B1.81.1 VDISR_EL2 at EL1 using AArch64 on page B1-238.

**B1.81.1 VDISR_EL2 at EL1 using AArch64**

VDISR_EL2 has a specific format when written at EL1.

The following figure shows the VDISR_EL2 bit assignments when written at EL1 using AArch64:

![Figure B1-57 VDISR_EL2 at EL1 using AArch64](image)

RES0, [63:32]

RES0 Reserved.

A, [31]

Set to 1 when ESB defers an asynchronous SError interrupt.

RES0, [30:25]

RES0 Reserved.

IDS, [24]

Contains the value from VSESRL2.IDS.

ISS, [23:0]

Contains the value from VSESRL2, bits[23:0].
B1.82 VSESR_EL2, Virtual SError Exception Syndrome Register

The VSESR_EL2 provides the syndrome value reported to software on taking a virtual SError interrupt exception.

**Bit field descriptions**

VSESR_EL2 is a 64-bit register, and is part of:

- The Exception and fault handling registers functional group.
- The Virtualization registers functional group.

If the virtual SError interrupt is taken to EL1, VSESR_EL2 provides the syndrome value reported in ESR_EL1.

**VSESR_EL2 bit assignments**

![Figure B1-58 VSESR_EL2 bit assignments]

- **RES0, [63:25]**
  
  `RES0` Reserved.

- **IDS, [24]**
  
  Indicates whether the deferred SError interrupt was of an IMPLEMENTATION DEFINED type. See ESR_EL1.IDS for a description of the functionality.

  On taking a virtual SError interrupt to EL1 using AArch64 because HCR_EL2.VSE == 1, ESR_EL1[24] is set to VSESR_EL2.IDS.

- **ISS, [23:0]**
  
  Syndrome information. See ESR_EL1.ISS for a description of the functionality.

**Configurations**

There are no configuration notes.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*. 
B1.83  VTCR_EL2, Virtualization Translation Control Register, EL2

The VTCR_EL2 controls the translation table walks required for the stage 2 translation of memory accesses from Non-secure EL0 and EL1.

It also holds cacheability and shareability information for the accesses.

**Bit field descriptions**

VTCR_EL2 is a 32-bit register, and is part of:

- The Virtualization registers functional group.
- The Virtual memory control registers functional group.

![VTCR_EL2 bit assignments](image)

---

**Note**

Bits[28:25] and bits[22:21], architecturally defined, are implemented in the core.

---

**TG0, [15:14]**

TTBR0_EL2 granule size. The possible values are:

- 00  4KB.
- 01  64KB.
- 10  16KB.
- 11  Reserved.

All other values are not supported.

**Configurations**

RW fields in this register reset to architecturally **UNKNOWN** values.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*. 
B1.84 VTTBR_EL2, Virtualization Translation Table Base Register, EL2

VTTBR_EL2 holds the base address of the translation table for the stage 2 translation of memory accesses from Non-secure EL0 and EL1.

Bit field descriptions

VTTBR_EL2 is a 64-bit register.

![Figure B1-60  VTTBR_EL2 bit assignments](image)

CnP, [0]

Common not Private. The possible values are:

0  CnP is not supported.

1  CnP is supported.

Configurations

There are no configuration notes.

Bit fields and details not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.
B1.84 VTTBR_EL2, Virtualization Translation Table Base Register, EL2
Chapter B2
Error system registers

This chapter describes the error registers accessed by the AArch64 error registers.

It contains the following sections:
• B2.1 Error system register summary on page B2-244.
• B2.2 ERR0CTLR, Error Record Control Register on page B2-245.
• B2.3 ERR0FR, Error Record Feature Register on page B2-247.
• B2.4 ERR0MISC0, Error Record Miscellaneous Register 0 on page B2-249.
• B2.5 ERR0PFGCDN, Error Pseudo Fault Generation Count Down Register on page B2-251.
• B2.6 ERR0PFGCTL, Error Pseudo Fault Generation Control Register on page B2-252.
• B2.7 ERR0PFGF, Error Pseudo Fault Generation Feature Register on page B2-254.
• B2.8 ERR0STATUS, Error Record Primary Status Register on page B2-256.
B2.1 Error system register summary

This section identifies the ERR0* core error record registers accessed by the AArch64 ERX* error registers.

For those registers not described in this chapter, see the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.

The following table describes the architectural error record registers.

<table>
<thead>
<tr>
<th>Register mnemonic</th>
<th>Size</th>
<th>Register name</th>
<th>Access aliases from AArch64</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERR0CTLR</td>
<td>64</td>
<td>B2.2 ERR0CTLR, Error Record Control Register on page B2-245</td>
<td>B1.37 ERXCTLR_EL1, Selected Error Record Control Register, EL1 on page B1-181</td>
</tr>
<tr>
<td>ERR0FR</td>
<td>64</td>
<td>B2.3 ERR0FR, Error Record Feature Register on page B2-247</td>
<td>B1.38 ERXFR_EL1, Selected Error Record Feature Register, EL1 on page B1-182</td>
</tr>
<tr>
<td>ERR0MISC0</td>
<td>64</td>
<td>B2.4 ERR0MISC0, Error Record Miscellaneous Register 0 on page B2-249</td>
<td>B1.39 ERXMISC0_EL1, Selected Error Record Miscellaneous Register 0, EL1 on page B1-183</td>
</tr>
<tr>
<td>ERR0STATUS</td>
<td>32</td>
<td>B2.8 ERR0STATUS, Error Record Primary Status Register on page B2-256</td>
<td>B1.43 ERXSTATUS_EL1, Selected Error Record Primary Status Register, EL1 on page B1-189</td>
</tr>
</tbody>
</table>

The following table describes the error record registers that are IMPLEMENTATION DEFINED.

<table>
<thead>
<tr>
<th>Register mnemonic</th>
<th>Size</th>
<th>Register name</th>
<th>Access aliases from AArch64</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERR0PFGCDN</td>
<td>32</td>
<td>B2.5 ERR0PFGCDN, Error Pseudo Fault Generation Count Down Register on page B2-251</td>
<td>B1.40 ERXPFGCDN_EL1, Selected Error Pseudo Fault Generation Count Down Register, EL1 on page B1-184</td>
</tr>
<tr>
<td>ERR0PFGCTL</td>
<td>32</td>
<td>B2.6 ERR0PFGCTL, Error Pseudo Fault Generation Control Register on page B2-252</td>
<td>B1.41 ERXPFGCTL_EL1, Selected Error Pseudo Fault Generation Control Register, EL1 on page B1-186</td>
</tr>
<tr>
<td>ERR0PFGF</td>
<td>32</td>
<td>B2.7 ERR0PFGF, Error Pseudo Fault Generation Feature Register on page B2-254</td>
<td>B1.42 ERXPFGF_EL1, Selected Pseudo Fault Generation Feature Register, EL1 on page B1-188</td>
</tr>
</tbody>
</table>
B2.2 ERR0CTLR, Error Record Control Register

The ERR0CTLR contains enable bits for the node that writes to this record:

- Enabling error detection and correction.
- Enabling an error recovery interrupt.
- Enabling a fault handling interrupt.
- Enabling error recovery reporting as a read or write error response.

Bit field descriptions

ERR0CTLR is a 64-bit register and is part of the Reliability, Availability, Serviceability (RAS) registers functional group.

ERR0CTLR resets to ED is 0x0. CFI [8], FI [3], and UI [2] are UNKNOWN. The rest of the register is RES0.

![Figure B2-1  ERR0CTLR bit assignments](image)

RES0, [63:9]

RES0 Reserved.

CFI, [8]

Fault handling interrupt for corrected errors enable.

The fault handling interrupt is generated when one of the standard CE counters on ERR0MISC0 overflows and the overflow bit is set. The possible values are:

0 Fault handling interrupt not generated for corrected errors.
1 Fault handling interrupt generated for corrected errors.

The interrupt is generated even if the error status is overwritten because the error record already records a higher priority error.

__________ Note __________

This applies to both reads and writes.

RES0, [7:4]

RES0 Reserved.

FI, [3]

Fault handling interrupt enable.

The fault handling interrupt is generated for all detected Deferred errors and Uncorrected errors. The possible values are:

0 Fault handling interrupt disabled.
1 Fault handling interrupt enabled.
UI, [2]
Uncorrected error recovery interrupt enable. When enabled, the error recovery interrupt is
generated for all detected Uncorrected errors that are not deferred. The possible values are:
0  Error recovery interrupt disabled.
1  Error recovery interrupt enabled.

Note
Applies to both reads and writes.

RES0, [1]
RES0  Reserved.

ED, [0]
Error Detection and correction enable. The possible values are:
0  Error detection and correction disabled.
1  Error detection and correction enabled.

Configurations
This register is accessible from the following registers when ERRSELR.SEL==0:

B1.37 ERXCTLR_EL1, Selected Error Record Control Register, EL1 on page B1-181.
**B2.3 ERR0FR, Error Record Feature Register**

The ERR0FR defines which of the common architecturally defined features are implemented and, of the implemented features, which are software programmable.

**Bit field descriptions**

ERR0FR is a 64-bit register, and is part of the Reliability, Availability, Serviceability (RAS) registers functional group.

The register is Read Only.

![ERR0FR bit assignments](image)

<table>
<thead>
<tr>
<th>Bit Position</th>
<th>Description</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[63:20]</td>
<td>RES0</td>
<td>Reserved.</td>
<td></td>
</tr>
<tr>
<td>[19:18]</td>
<td>CEO</td>
<td>00</td>
<td>Counts CE if a counter is implemented and keeps the previous error status. If the counter overflows, ERR0STATUS.OF is set to 1.</td>
</tr>
<tr>
<td>[17:16]</td>
<td>DUI</td>
<td>00</td>
<td>The core does not support this feature.</td>
</tr>
<tr>
<td>[15]</td>
<td>RP</td>
<td>1</td>
<td>A first repeat counter and a second other counter are implemented. The repeat counter is the same size as the primary error counter.</td>
</tr>
<tr>
<td>[14:12]</td>
<td>CEC</td>
<td>010</td>
<td>The node implements an 8-bit standard CE counter in ERR0MISC0[39:32].</td>
</tr>
<tr>
<td>[11:10]</td>
<td>CFI</td>
<td>10</td>
<td>The node implements a control for enabling fault handling interrupts on corrected errors.</td>
</tr>
<tr>
<td>[9:8]</td>
<td>UE</td>
<td>01</td>
<td>The node implements in-band uncorrected error reporting, that is external aborts.</td>
</tr>
<tr>
<td>[7:6]</td>
<td>FI</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Fault handling interrupt. The value is:
10  The node implements a fault handling interrupt and implements controls for enabling and disabling.

UI, [5:4]
Error recovery interrupt for uncorrected errors. The value is:
01  The node always enables uncorrected error recovery interrupt.

DE, [3:2]
Defers errors. The value is:
01  Defers errors is always enabled for the node.

ED, [1:0]
Error detection and correction. The value is:
10  The node implements controls for enabling or disabling error detection and correction.

Configurations
ERR0FR resets to 0x0000000000A996
ERR0FR is accessible from the following registers when ERRSEL.R.SEL==0:

B1.38 ERXFR_EL1, Selected Error Record Feature Register, EL1 on page B1-182.
B2.4 ERR0MISC0, Error Record Miscellaneous Register 0

The ERR0MISC0 is an error syndrome register. It contains corrected error counters, information to identify where the error was detected, and other state information not present in the corresponding status and address error record registers.

**Bit field descriptions**

ERR0MISC0 is a 64-bit register, and is part of the Reliability, Availability, Serviceability (RAS) registers functional group.

![ERR0MISC0 bit assignments](image)

**Bit 63:48**

- **RES0**: Reserved.

**OFO, [47]**

- Sticky overflow bit, other. The possible values of this bit are:
  - 0: Other counter has not overflowed.
  - 1: Other counter has overflowed.

  The fault handling interrupt is generated when the corrected fault handling interrupt is enabled and either overflow bit is set to 1.

**CECO, [46:40]**

- Corrected error count, other. Incremented for each Corrected error that does not match the recorded syndrome.

  This field resets to an implementation defined value which might be UNKNOWN on a Cold reset. If the reset value is UNKNOWN, then the value of this field remains UNKNOWN until software initializes it.

**OFR, [39]**

- Sticky overflow bit, repeat. The possible values of this bit are:
  - 0: Repeat counter has not overflowed.
  - 1: Repeat counter has overflowed.

  The fault handling interrupt is generated when the corrected fault handling interrupt is enabled and either overflow bit is set to 1.

**CECR, [38:32]**

- Corrected error count, repeat. Incremented for the first recorded error, which also records other syndrome, and then again for each Corrected error that matches the recorded syndrome.

  This field resets to an implementation defined which might be UNKNOWN on a Cold reset. If the reset value is UNKNOWN, then the value of this field remains UNKNOWN until software initializes it.
WAY, [31:28]
Indicates the way that contained the error.
- For the L3 cache all four bits are used
- For the L2 cache, L1 D-cache, L1 I-cache and L2 TLB only bits [31:30] are used

[27:18]
RES0 Reserved.

INDX, [17:6]
Indicates the index that contained the error.
Upper bits of the index are unused depending on the cache size.

[5:4]
RES0 Reserved.

LVL, [3:1]
Indicates the level that contained the error. The possible values are:
- 000 Level 1.
- 001 Level 2.

IND, [0]
Indicates the type of cache that contained the error. The possible values are:
- 0 L1 data cache, unified L2 cache, or TLB.
- 1 L1 instruction cache.

Configurations
ERR0MISC0 resets to [63:32] is 0x00000000, [31:0] is UNKNOWN.
When ERRSELR.SEL==0, this register is accessible from B1.39 ERXMISC0_EL1, Selected Error Record Miscellaneous Register 0, EL1 on page B1-183.
The following observations should be made about this register:
- If two or more memory errors occur in the same cycle, only one error is reported but the other error count will be incremented.
- If two or more first memory error events from different RAMs occur in the same cycle one of the errors is selected arbitrarily.
- If a new error arrives while the ERXSTATUS.V bit is set, the way, index, and level information is not updated, but the other error field or the repeat error field is updated.
- If two or more memory errors from different RAMs that do not match the level, way and index information in this register when the ERXSTATUS.V bit is set, occur in the same cycle, the Other error count field is only incremented once.
- This register is not reset on a warm reset.
B2.5 ERR0PFGCDN, Error Pseudo Fault Generation Count Down Register

ERR0PFGCDN is the Cortex-A65 node register that generates one of the errors that are enabled in the corresponding ERR0PFGCTL register.

Bit field descriptions

ERR0PFGCDN is a 32-bit register and is RW.

```
+-----+-----+-----+-----+-----+-----+-----+-----+
|     |     |     |     | CDN |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+
```

Figure B2-4 ERR0PFGCDN bit assignments

CDN, [31:0]

Count Down value. The reset value of the Error Generation Counter is used for the countdown.

Configurations

- There are no configuration options.
- ERR0PFGCDN resets to UNKNOWN.

When ERRSELR.SEL==0, ERR0PFGCDN is accessible from B1.40 ERXPFGCDN_EL1, Selected Error Pseudo Fault Generation Count Down Register; EL1 on page B1-184.
B2.6 ERR0PFGCTL, Error Pseudo Fault Generation Control Register

The ERR0PFGCTL is the Cortex-A65 node register that enables controlled fault generation.

Bit field descriptions

ERR0PFGCTL is a 32-bit register and is RW.

![Figure B2-5 ERR0PFGCTL bit assignments](image)

CDNEN, [31]

Count down enable. This bit controls transfers from the value that is held in the ERR0PFGCDN into the Error Generation Counter and enables this counter to start counting down. The possible values are:

- 0: The Error Generation Counter is disabled. This is the reset value.
- 1: The value that is held in the ERR0PFGCDN register is transferred into the Error Generation Counter. The Error Generation Counter counts down.

R, [30]

Restartable bit. When it reaches 0, the Error Generation Counter restarts from the ERR0PFGCDN value or stops. The possible values are:

- 0: When it reaches 0, the counter stops. This is the reset value.
- 1: When it reaches 0, the counter reloads the value that is stored in ERR0PFGCDN and starts counting down again.

[29:7]

Reserved, UNK/UNZP.

CE, [6]

Corrected error generation enable. The possible values are:

- 0: No corrected error is generated. This is the reset value.
- 1: A corrected error might be generated when the Error Generation Counter is triggered.

DE, [5]

Deferred Error generation enable. The possible values are:

- 0: No deferred error is generated. This is the reset value.
- 1: A deferred error might be generated when the Error Generation Counter is triggered.

UEO, [4]

Latent or Restartable Error generation. The value is:
No latent or restartable error is generated.

**UER, [3]**

Signaled or Recoverable Error generation. The value is:

- 0: No signaled or recoverable error is generated. This is the reset value.
- 1: This feature is controllable.

**RES0, [2]**

RES0: Reserved.

**UC, [1]**

Uncontainable error generation enable. The possible values are:

- 0: No uncontainable error is generated. This is the reset value.
- 1: An uncontainable error might be generated when the Error Generation Counter is triggered.

[0]

Reserved, UNK/SBZP.

**Configurations**

There are no configuration notes.

ERR0PFGCTRL resets to **UNKNOWN**.

When ERRSELR.SEL==0, ERR0PFGCTRL is accessible from *B1.41 ERXPFGCTL_EL1, Selected Error Pseudo Fault Generation Control Register, EL1* on page B1-186.
**B2.7 ERR0PFGF, Error Pseudo Fault Generation Feature Register**

The ERR0PFGF is the Cortex-A65 node register that defines which fault generation features are implemented.

**Bit field descriptions**

ERR0PFGF is a 32-bit register and is RO.

![ERR0PFGF bit assignments](image)

**PFG, [31]**

Pseudo Fault Generation. The value is:

1  The node implements a fault injection mechanism.

**R, [30]**

Restartable bit. When it reaches zero, the Error Generation Counter restarts from the ERR0PFGCDN value or stops. The value is:

1  This feature is controllable.

**[29:7]**

Reserved, UNK/SBZP.

**CE, [6]**

Corrected Error generation. The value is:

1  This feature is controllable.

**DE, [5]**

Deferred Error generation. The value is:

1  This feature is controllable.

**UEO, [4]**

Latent or Restartable Error generation. The value is:

0  The node does not support this feature.

**UER, [3]**

Signaled or Recoverable Error generation. The value is:

1  This feature is controllable.

**RES0, [2]**

RES0  Reserved.
UC, [1]

Uncontainable Error generation. The value is:

1  This feature is controllable.

[0]

Reserved, UNK/SBZP.

Configurations

There are no configuration notes.

ERR0PFGF resets to UNKNOWN.

When ERRSEL.R.SEL==0, ERR0PFGF is accessible from B1.42 ERXPFPGF_EL1, Selected Pseudo Fault Generation Feature Register; EL1 on page B1-188.
B2.8 ERR0STATUS, Error Record Primary Status Register

The ERR0STATUS contains information about the error record:

- Whether any error has been detected.
- Whether any detected error was not corrected and returned to a master.
- Whether any detected error was not corrected and deferred.
- Whether a second error of the same type was detected before software handled the first error.
- Whether any error has been reported.
- Whether the other error record registers contain valid information.

### Bit field descriptions

ERR0STATUS is a 32-bit register.

![ERR0STATUS bit assignments](image)

**AV, [31]**

Address Valid. The value is:

- 0: ERR0ADDR is not valid.

**V, [30]**

Status Register valid. The possible values are:

- 0: ERR0STATUS is not valid. This is the reset value.
- 1: ERR0STATUS is valid. At least one error has been recorded.

**UE, [29]**

Uncorrected error. The possible values are:

- 0: No error that could not be corrected or deferred has been detected. This is the reset value.
- 1: At least one error that could not be corrected or deferred has been detected. If error recovery interrupts are enabled, then the interrupt signal is asserted until this bit is cleared.

**ER, [28]**

Error reported. The possible values are:

- 0: No external abort has been reported. This is the reset value.
- 1: The node has reported an external abort to the master that is in access or making a transaction.

**OF, [27]**
Overflow. The possible values are:

0
  • If UE == 1, then no error status for an Uncorrected error has been discarded.
  • If UE == 0 and DE == 1, then no error status for a Deferred error has been discarded.
  • If UE == 0, DE == 0, and CE != 0x0, then:
    — If a Corrected error counter is implemented, it has not overflowed.
    — If no Corrected error counter is implemented, no error status for a Corrected error has been discarded.

This is the reset value.

1 More than one error has occurred and so details of the other error have been discarded.

MV, [26]
Miscellaneous Registers Valid. The possible values are:

0 ERR0MISC0 and ERR0MISC1 are not valid. This is the reset value.
1 This bit indicates that ERR0MISC0 contains additional information about any error recorded by this record.

CE, [25:24]
Corrected error. The possible values are:

0x0 No corrected errors recorded. This is the reset value.
0x2 At least one corrected error recorded.

DE, [23]
Deferred error. The possible values are:

0 No errors were deferred. This is the reset value.
1 At least one error was not corrected and deferred by poisoning.

PN, [22]
Poison. The value is:

0 No uncorrected errors were detected. This is the reset value.
1 At least one uncorrected error detected by reading a poisoned line.

UI, CT, [21:20]
Uninfected and Containable Error Type. The value is:

0x0 Uncontainable. This is the reset value.

RES0, [19:16]
RES0 Reserved.

IERR, [15:8]
IMPLEMENTATION DEFINED error code. The possible values are:

0x0 No error, or error on other RAMs.
0x1 Error on L1 dirty RAM.
0x2 Error on L3 snoop filter RAM.

SERR, [7:0]
Primary error code. The possible values are:

0x0 No error.
0x2 ECC error from internal data buffer.
0x6 ECC error on cache data RAM.
0x07  ECC error on cache tag or dirty RAM.
0x08  Parity error on TLB data RAM.
0x09  Parity error on TLB tag RAM.
0x12  Bus error.
0x15  Deferred error from slave not supported at the consumer. For example, poisoned data received from a slave by a master that cannot defer the error further.

Configurations

There are no configuration notes.

ERR0STATUS resets to 0x0000000000000000.

When ERRSELR.SEL==0, ERR0STATUS is accessible from B1.43 ERXSTATUS_EL1, Selected Error Record Primary Status Register, EL1 on page B1-189.

The following observations should be made about this register:

• When this record is written, the corresponding nFAULTIRQ pin will be asserted if the error is uncorrectable or if the correctable error counter has overflowed (if enabled in the ERXCTLR). To deassert the pin, software must set the Valid field to zero.
• When the UE field is set for an error on an L3 RAM, the nERRIRQ[0] pin is asserted (if enabled in the ERXCTLR). To deassert the pin, software must set the UE field to zero.
• When the UE field is set for an error on a L1 or L2 RAM, the nERRIRQ[n+1] pin is asserted for core n (if enabled in the ERXCTLR). To deassert the pin, software must set the UE field to zero.
• This register is not reset on a warm reset.
Chapter B3
GIC registers

This chapter describes the GIC registers.

It contains the following sections:

- **B3.1 CPU interface registers** on page B3-261.
- **B3.2 AArch64 physical GIC CPU interface system register summary** on page B3-262.
- **B3.3 ICC_AP0R0_EL1, Interrupt Controller Active Priorities Group 0 Register 0, EL1** on page B3-263.
- **B3.4 ICC_AP1R0_EL1, Interrupt Controller Active Priorities Group 1 Register 0 EL1** on page B3-264.
- **B3.5 ICC_BPR0_EL1, Interrupt Controller Binary Point Register 0, EL1** on page B3-265.
- **B3.6 ICC_BPR1_EL1, Interrupt Controller Binary Point Register 1, EL1** on page B3-266.
- **B3.7 ICC_CTLR_EL1, Interrupt Controller Control Register, EL1** on page B3-267.
- **B3.8 ICC_CTLR_EL3, Interrupt Controller Control Register, EL3** on page B3-269.
- **B3.9 ICC_SRE_EL1, Interrupt Controller System Register Enable Register, EL1** on page B3-271.
- **B3.10 ICC_SRE_EL2, Interrupt Controller System Register Enable register, EL2** on page B3-272.
- **B3.11 ICC_SRE_EL3, Interrupt Controller System Register Enable register, EL3** on page B3-274.
- **B3.12 AArch64 virtual GIC CPU interface register summary** on page B3-276.
- **B3.13 ICV_AP0R0_EL1, Interrupt Controller Virtual Active Priorities Group 0 Register 0, EL1** on page B3-277.
- **B3.14 ICV_AP1R0_EL1, Interrupt Controller Virtual Active Priorities Group 1 Register 0, EL1** on page B3-278.
- **B3.15 ICV_BPR0_EL1, Interrupt Controller Virtual Binary Point Register 0, EL1** on page B3-279.
- **B3.16 ICV_BPRI_EL1, Interrupt Controller Virtual Binary Point Register 1, EL1** on page B3-280.
- **B3.17 ICV_CTLR_EL1, Interrupt Controller Virtual Control Register, EL1** on page B3-281.
- **B3.18 AArch64 virtual interface control system register summary** on page B3-283.
• **B3.19 ICH_AP0R0_EL2, Interrupt Controller Hyp Active Priorities Group 0 Register 0, EL2** on page B3-284.
• **B3.20 ICH_AP1R0_EL2, Interrupt Controller Hyp Active Priorities Group 1 Register 0, EL2** on page B3-285.
• **B3.21 ICH_HCR_EL2, Interrupt Controller Hyp Control Register, EL2** on page B3-286.
• **B3.22 ICH_VMCR_EL2, Interrupt Controller Virtual Machine Control Register, EL2** on page B3-289.
• **B3.23 ICH_VTR_EL2, Interrupt Controller VGIC Type Register, EL2** on page B3-291.
B3.1 CPU interface registers

Each CPU interface block provides the interface for the Cortex-A65 core that interfaces with a GIC distributor within the system.

The Cortex-A65 core only supports system register access to the GIC CPU interface registers. The following table lists the three types of GIC CPU interface system registers supported in the Cortex-A65 core.

<table>
<thead>
<tr>
<th>Register prefix</th>
<th>Register type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICC</td>
<td>Physical GIC CPU interface system registers.</td>
</tr>
<tr>
<td>ICV</td>
<td>Virtual GIC CPU interface system registers.</td>
</tr>
<tr>
<td>ICH</td>
<td>Virtual interface control system registers.</td>
</tr>
</tbody>
</table>

Access to virtual GIC CPU interface system registers is only possible at Non-secure EL1.

Access to ICC registers or the equivalent ICV registers is determined by HCR_EL2. See B1.48 HCR_EL2, Hypervisor Configuration Register, EL2 on page B1-194.

For more information on the CPU interface, see the Arm® Generic Interrupt Controller Architecture Specification.
The following table lists the AArch64 physical GIC CPU interface system registers that have implementation defined bits.

See the Arm® Generic Interrupt Controller Architecture Specification for more information and a complete list of AArch64 physical GIC CPU interface system registers.

<table>
<thead>
<tr>
<th>Name</th>
<th>Op0</th>
<th>Op1</th>
<th>CRn</th>
<th>CRm</th>
<th>Op2</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICC_AP0R_EL1</td>
<td>3</td>
<td>0</td>
<td>12</td>
<td>8</td>
<td>4</td>
<td>RW</td>
<td>B3.3 ICC_AP0R_EL1, Interrupt Controller Active Priorities Group 0 Register 0, EL1 on page B3-263</td>
</tr>
<tr>
<td>ICC_AP1R_EL1</td>
<td>3</td>
<td>0</td>
<td>12</td>
<td>9</td>
<td>0</td>
<td>RW</td>
<td>B3.4 ICC_AP1R_EL1, Interrupt Controller Active Priorities Group 1 Register 0 EL1 on page B3-264</td>
</tr>
<tr>
<td>ICC_BPR0_EL1</td>
<td>3</td>
<td>0</td>
<td>12</td>
<td>8</td>
<td>3</td>
<td>RW</td>
<td>B3.5 ICC_BPR0_EL1, Interrupt Controller Binary Point Register 0, EL1 on page B3-265</td>
</tr>
<tr>
<td>ICC_BPR1_EL1</td>
<td>3</td>
<td>0</td>
<td>12</td>
<td>12</td>
<td>3</td>
<td>RW</td>
<td>B3.6 ICC_BPR1_EL1, Interrupt Controller Binary Point Register 1, EL1 on page B3-266</td>
</tr>
<tr>
<td>ICC_CTLR_EL1</td>
<td>3</td>
<td>0</td>
<td>12</td>
<td>12</td>
<td>4</td>
<td>RW</td>
<td>B3.7 ICC_CTLR_EL1, Interrupt Controller Control Register, EL1 on page B3-267</td>
</tr>
<tr>
<td>ICC_CTLR_EL3</td>
<td>3</td>
<td>6</td>
<td>12</td>
<td>12</td>
<td>4</td>
<td>RW</td>
<td>B3.8 ICC_CTLR_EL3, Interrupt Controller Control Register, EL3 on page B3-269</td>
</tr>
<tr>
<td>ICC_SRE_EL1</td>
<td>3</td>
<td>0</td>
<td>12</td>
<td>12</td>
<td>5</td>
<td>RW</td>
<td>B3.9 ICC_SRE_EL1, Interrupt Controller System Register Enable Register, EL1 on page B3-271</td>
</tr>
<tr>
<td>ICC_SRE_EL2</td>
<td>3</td>
<td>4</td>
<td>12</td>
<td>9</td>
<td>5</td>
<td>RW</td>
<td>B3.10 ICC_SRE_EL2, Interrupt Controller System Register Enable register, EL2 on page B3-272</td>
</tr>
<tr>
<td>ICC_SRE_EL3</td>
<td>3</td>
<td>6</td>
<td>12</td>
<td>12</td>
<td>5</td>
<td>RW</td>
<td>B3.11 ICC_SRE_EL3, Interrupt Controller System Register Enable register, EL3 on page B3-274</td>
</tr>
</tbody>
</table>
B3.3 ICC_AP0R0_EL1, Interrupt Controller Active Priorities Group 0 Register 0, EL1

The ICC_AP0R0_EL1 provides information about Group 0 active priorities.

**Bit descriptions**

This register is a 32-bit register and is part of:
- The GIC system registers functional group.
- The GIC control registers functional group.

The core implements 5 bits of priority with 32 priority levels, corresponding to the 32 bits [31:0] of the register. The possible values for each bit are:

- **0x00000000** No interrupt active. This is the reset value.
- **0x00000001** Interrupt active for priority 0x0.
- **0x00000002** Interrupt active for priority 0x8.
- ...
- **0x80000000** Interrupt active for priority 0xF8.
B3.4 ICC_AP1R0_EL1, Interrupt Controller Active Priorities Group 1 Register 0 EL1

The ICC_AP1R0_EL1 provides information about Group 1 active priorities.

**Bit descriptions**

This register is a 32-bit register and is part of:

- The GIC system registers functional group.
- The GIC control registers functional group.

The core implements 5 bits of priority with 32 priority levels, corresponding to the 32 bits [31:0] of the register. The possible values for each bit are:

- `0x00000000` No interrupt active. This is the reset value.
- `0x00000001` Interrupt active for priority 0x0.
- `0x00000002` Interrupt active for priority 0x8.

... 

- `0x80000000` Interrupt active for priority 0xF8.
B3.5 ICC_BPR0_EL1, Interrupt Controller Binary Point Register 0, EL1

ICC_BPR0_EL1 defines the point at which the priority value fields split into two parts, the group priority field and the subpriority field. The group priority field determines Group 0 interrupt preemption.

**Bit field descriptions**

ICC_BPR0_EL1 is a 32-bit register and is part of:
- The GIC system registers functional group.
- The GIC control registers functional group.

![Figure B3-1 ICC_BPR0_EL1 bit assignments](image)

**RES0, [31:3]**

RES0 Reserved.

**BinaryPoint, [2:0]**

The value of this field controls how the 8-bit interrupt priority field is split into a group priority field, that determines interrupt preemption, and a subpriority field. The minimum value that is implemented is:

\[0\times2\]

Bit fields and details that are not provided in this description are architecturally defined. See the Arm® Generic Interrupt Controller Architecture Specification.
B3.6 ICC_BPR1_EL1, Interrupt Controller Binary Point Register 1, EL1

ICC_BPR1_EL1 defines the point at which the priority value fields split into two parts, the group priority field and the subpriority field. The group priority field determines Group 1 interrupt preemption.

**Bit field descriptions**

ICC_BPR1_EL1 is a 32-bit register and is part of:
- The GIC system registers functional group.
- The GIC control registers functional group.

![Figure B3-2 ICC_BPR1_EL1 bit assignments](image)

RES0, [31:3]

RES0 Reserved.

**BinaryPoint, [2:0]**

The value of this field controls how the 8-bit interrupt priority field is split into a group priority field, that determines interrupt preemption, and a subpriority field.

The minimum value implemented of ICC_BPR1_EL1 Secure register is 0x2.

The minimum value implemented of ICC_BPR1_EL1 Non-secure register is 0x3.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Generic Interrupt Controller Architecture Specification*. 
B3.7 ICC_CTLR_EL1, Interrupt Controller Control Register, EL1

ICC_CTLR_EL1 controls aspects of the behavior of the GIC CPU interface and provides information about the features implemented.

Bit field descriptions
ICC_CTLR_EL1 is a 32-bit register and is part of:
- The GIC system registers functional group.
- The GIC control registers functional group.

RES0, [31:16]
RES0 Reserved.

A3V, [15]
Affinity 3 Valid. The value is:
1 The CPU interface logic supports non-zero values of Affinity 3 in SGI generation System registers.

SEIS, [14]
SEI Support. The value is:
0 The CPU interface logic does not support local generation of SEIs.

IDbits, [13:11]
Identifier bits. The value is:
0 The number of physical interrupt identifier bits supported is 16 bits.
This field is an alias of ICC_CTLR_EL3.IDbits.

PRIbits, [10:8]
Priority bits. The value is:
0x4 The core supports 32 levels of physical priority with 5 priority bits.

RES0, [7]
RES0 Reserved.

PMHE, [6]
0 Disables use of ICC_PMR as a hint for interrupt distribution.
1 Enables use of ICC_PMR as a hint for interrupt distribution.
RES0, [5:2]

RES0 Reserved.

EOImode, [1]

End of interrupt mode for the current security state. The possible values are:

0 ICC_EOIR0 and ICC_EOIR1 provide both priority drop and interrupt deactivation functionality. Accesses to ICC_DIR are UNPREDICTABLE.

1 ICC_EOIR0 and ICC_EOIR1 provide priority drop functionality only. ICC_DIR provides interrupt deactivation functionality.

CBPR, [0]

Common Binary Point Register. Control whether the same register is used for interrupt preemption of both Group 0 and Group 1 interrupt. The possible values are:

0 ICC_BPR0 determines the preemption group for Group 0 interrupts.

1 ICC_BPR1 determines the preemption group for Group 1 interrupts.

1 ICC_BPR0 and ICC_BPR1 determine the preemption group for Group 0 and Group 1 interrupts.

Bit fields and details that are not provided in this description are architecturally defined. See the Arm® Generic Interrupt Controller Architecture Specification.
### B3.8 ICC_CTLR_EL3, Interrupt Controller Control Register, EL3

ICC_CTLR_EL3 controls aspects of the behavior of the GIC CPU interface and provides information about the features implemented.

#### Bit field descriptions
ICC_CTLR_EL3 is a 32-bit register and is part of:
- The GIC system registers functional group.
- The Security registers functional group.
- The GIC control registers functional group.

#### RES0, [31:18]
RES0 Reserved.

#### nDS, [17]
Disable Security not supported. Read-only and writes are ignored. The value is:
- 1 The CPU interface logic does not support disabling of security, and requires that security is not disabled.

#### RES0, [16]
RES0 Reserved.

#### A3V, [15]
Affinity 3 Valid. This bit is RAO/WI.

#### SEIS, [14]
SEI Support. The value is:
- 0 The CPU interface logic does not support generation of SEIs.

#### IDbits, [13:11]
Identifier bits. The value is:
- 0x0 The number of physical interrupt identifier bits supported is 16 bits.

This field is an alias of ICC_CTLR_EL3.IDbits.
PRIbits, [10:8]
Priority bits. The value is:
0x4 The core supports 32 levels of physical priority with 5 priority bits.
Accesses to ICC_AP0R{1—3} and ICC_AP1R{1—3} are undefined.

RES0, [7]
Reserved, RES0.

PMHE, [6]
Priority Mask Hint Enable. The possible values are:
0 Disables use of ICC_PMR as a hint for interrupt distribution.
1 Enables use of ICC_PMR as a hint for interrupt distribution.

RM, [5]
Routing Modifier. This bit is RAZ/WI.

EOImode_EL1NS, [4]
EOI mode for interrupts handled at Non-secure EL1 and EL2.
Controls whether a write to an End of Interrupt register also deactivates the interrupt.

EOImode_EL1S, [3]
EOI mode for interrupts handled at Secure EL1.
Controls whether a write to an End of Interrupt register also deactivates the interrupt.

EOImode_EL3, [2]
EOI mode for interrupts handled at EL3.
Controls whether a write to an End of Interrupt register also deactivates the interrupt.

CBPR_EL1NS, [1]
Common Binary Point Register, EL1 Non-secure.
Control whether the same register is used for interrupt preemption of both Group 0 and Group 1 Non-secure interrupts at EL1 and EL2.

CBPR_EL1S, [0]
Common Binary Point Register, EL1 Secure.
Control whether the same register is used for interrupt preemption of both Group 0 and Group 1 Secure interrupt at EL1.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm* Generic Interrupt Controller Architecture Specification.
ICC_SRE_EL1, Interrupt Controller System Register Enable Register, EL1

ICC_SRE_EL1 controls whether the System register interface or the memory-mapped interface to the GIC CPU interface is used for EL0 and EL1.

Bit field descriptions
ICC_SRE_EL1 is a 32-bit register and is part of:
- The GIC system registers functional group.
- The GIC control registers functional group.

RES0, [31:3]
Reserved.

DIB, [2]
Disable IRQ bypass. The possible values are:
- 0x0     IRQ bypass enabled.
- 0x1     IRQ bypass disabled.

This bit is an alias of ICC_SRE_EL3.DIB

DFB, [1]
Disable FIQ bypass. The possible values are:
- 0x0     FIQ bypass enabled.
- 0x1     FIQ bypass disabled.

This bit is an alias of ICC_SRE_EL3.DFB

SRE, [0]
System Register Enable. The value is:
- 0x1     The System register interface for the current Security state is enabled.

This bit is RAO/WI. The core only supports a system register interface to the GIC CPU interface.

Bit fields and details that are not provided in this description are architecturally defined. See the Arm* Generic Interrupt Controller Architecture Specification.
ICC_SRE_EL2, Interrupt Controller System Register Enable register, EL2

ICC_SRE_EL2 controls whether the system register interface or the memory-mapped interface to the GIC CPU interface is used for EL2.

**Bit field descriptions**

ICC_SRE_EL2 is a 32-bit register and is part of:
- The GIC system registers functional group.
- The Virtualization registers functional group.
- The GIC control registers functional group.

![Figure B3-6 ICC_SRE_EL2 bit assignments](image)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| RES0 | [31:4] | RES0 | Reserved. |
| Enable, [3] | Enables lower Exception level access to ICC_SRE_EL1. The value is: |
| 0x1 | Non-secure EL1 accesses to ICC_SRE_EL1 do not trap to EL2. |
| This bit is RAO/WI. |
| DIB, [2] | Disable IRQ bypass. The possible values are: |
| 0x0 | IRQ bypass enabled. |
| 0x1 | IRQ bypass disabled. |
| This bit is an alias of ICC_SRE_EL3.DIB |
| DFB, [1] | Disable FIQ bypass. The possible values are: |
| 0x0 | FIQ bypass enabled. |
| 0x1 | FIQ bypass disabled. |
| This bit is an alias of ICC_SRE_EL3.DFB |
| SRE, [0] | System Register Enable. The value is: |
| 0x1 | The System register interface for the current Security state is enabled. |
This bit is RAO/WI. The core only supports a system register interface to the GIC CPU interface.

Bit fields and details that are not provided in this description are architecturally defined. See the Arm® Generic Interrupt Controller Architecture Specification.
B3.11 ICC_SRE_EL3, Interrupt Controller System Register Enable register, EL3

ICC_SRE_EL3 controls whether the System register interface or the memory-mapped interface to the GIC CPU interface is used for EL3.

Bit field descriptions
ICC_SRE_EL3 is a 32-bit register and is part of:
- The GIC system registers functional group.
- The Security registers functional group.
- The GIC control registers functional group.

RES0, [31:4]
RES0  Reserved.

Enable, [3]
Enables lower Exception level access to ICC_SRE_EL1 and ICC_SRE_EL2. The value is:
1  • Secure EL1 accesses to Secure ICC_SRE_EL1 do not trap to EL3.
   • EL2 accesses to Non-secure ICC_SRE_EL1 and ICC_SRE_EL2 do not trap to EL3.
   • Non-secure EL1 accesses to ICC_SRE_EL1 do not trap to EL3.

This bit is RAO/WI.

DIB, [2]
Disable IRQ bypass. The possible values are:
0  IRQ bypass enabled.
1  IRQ bypass disabled.

DFB, [1]
Disable FIQ bypass. The possible values are:
0  FIQ bypass enabled.
1  FIQ bypass disabled.

SRE, [0]
System Register Enable. The value is:
1  The System register interface for the current Security state is enabled.
This bit is RAO/WI. The core only supports a system register interface to the GIC CPU interface.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Generic Interrupt Controller Architecture Specification.*
The following table describes the AArch64 virtual GIC CPU interface system registers that have IMPLEMENTATION DEFINED bits.

See the Arm® Generic Interrupt Controller Architecture Specification for more information and a complete list of AArch64 virtual GIC CPU interface system registers.

<table>
<thead>
<tr>
<th>Name</th>
<th>Op0</th>
<th>Op1</th>
<th>CRn</th>
<th>CRm</th>
<th>Op2</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICV_AP0R0_EL1</td>
<td>3</td>
<td>0</td>
<td>12</td>
<td>8</td>
<td>4</td>
<td>RW</td>
<td>B3.13 ICV_AP0R0_EL1, Interrupt Controller Virtual Active Priorities Group 0 Register 0, EL1 on page B3-277</td>
</tr>
<tr>
<td>ICV_AP1R0_EL1</td>
<td>3</td>
<td>0</td>
<td>12</td>
<td>9</td>
<td>0</td>
<td>RW</td>
<td>B3.14 ICV_AP1R0_EL1, Interrupt Controller Virtual Active Priorities Group 1 Register 0, EL1 on page B3-278</td>
</tr>
<tr>
<td>ICV_BRP0_EL1</td>
<td>3</td>
<td>0</td>
<td>12</td>
<td>8</td>
<td>3</td>
<td>RW</td>
<td>B3.15 ICV_BPR0_EL1, Interrupt Controller Virtual Binary Point Register 0, EL1 on page B3-279</td>
</tr>
<tr>
<td>ICV_BPR1_EL1</td>
<td>3</td>
<td>0</td>
<td>12</td>
<td>12</td>
<td>3</td>
<td>RW</td>
<td>B3.16 ICV_BPR1_EL1, Interrupt Controller Virtual Binary Point Register 1, EL1 on page B3-280</td>
</tr>
<tr>
<td>ICV_CTLR_EL1</td>
<td>3</td>
<td>0</td>
<td>12</td>
<td>12</td>
<td>4</td>
<td>RW</td>
<td>B3.17 ICV_CTLR_EL1, Interrupt Controller Virtual Control Register, EL1 on page B3-281</td>
</tr>
</tbody>
</table>
B3.13 ICV_AP0R0_EL1, Interrupt Controller Virtual Active Priorities Group 0 Register 0, EL1

The ICV_AP0R0_EL1 register provides information about virtual Group 0 active priorities.

**Bit descriptions**

This register is a 32-bit register and is part of the virtual GIC system registers functional group.

The core implements 5 bits of priority with 32 priority levels, corresponding to the 32 bits [31:0] of the register. The possible values for each bit are:

- 0x00000000 No interrupt active. This is the reset value.
- 0x00000001 Interrupt active for priority 0x0.
- 0x00000002 Interrupt active for priority 0x8.

... 0x80000000 Interrupt active for priority 0xF8.

Details that are not provided in this description are architecturally defined. See the Arm® Generic Interrupt Controller Architecture Specification.
B3.14  **ICV_AP1R0_EL1, Interrupt Controller Virtual Active Priorities Group 1 Register 0, EL1**

The ICV_AP1R0_EL1 register provides information about virtual Group 1 active priorities.

**Bit descriptions**

This register is a 32-bit register and is part of the virtual GIC system registers functional group.

The core implements 5 bits of priority with 32 priority levels, corresponding to the 32 bits [31:0] of the register. The possible values for each bit are:

- 0x00000000  No interrupt active. This is the reset value.
- 0x00000001  Interrupt active for priority 0x0.
- 0x00000002  Interrupt active for priority 0x8.
  ...
- 0x80000000  Interrupt active for priority 0xF8.

Details that are not provided in this description are architecturally defined. See the *Arm® Generic Interrupt Controller Architecture Specification*. 
B3.15  ICV_BPR0_EL1, Interrupt Controller Virtual Binary Point Register 0, EL1

ICV_BPR0_EL1 defines the point at which the priority value fields split into two parts, the group priority field and the subpriority field. The group priority field determines virtual Group 0 interrupt preemption.

**Bit field descriptions**

ICC_BPR0_EL1 is a 32-bit register and is part of the virtual GIC system registers functional group.

![ICV_BPR0_EL1 bit assignments](image)

**RES0**

**RES0**, [31:3]

Reserved, RES0.

**BinaryPoint**, [2:0]

The value of this field controls how the 8-bit interrupt priority field is split into a group priority field, that determines interrupt preemption, and a subpriority field. The minimum value that is implemented is:

**0x2**

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Generic Interrupt Controller Architecture Specification*.
ICV_BPR1_EL1, Interrupt Controller Virtual Binary Point Register 1, EL1

ICV_BPR1_EL1 defines the point at which the priority value fields split into two parts, the group priority field and the subpriority field. The group priority field determines virtual Group 1 interrupt preemption.

**Bit field descriptions**

ICV_BPR1_EL1 is a 32-bit register and is part of the virtual GIC system registers functional group.

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

RES0, [31:3]

RES0  Reserved.

BinaryPoint, [2:0]

The value of this field controls how the 8-bit interrupt priority field is split into a group priority field, that determines interrupt preemption, and a subpriority field.

The minimum value that is implemented of ICV_BPR1_EL1 Secure register is 0x2.

The minimum value that is implemented of ICV_BPR1_EL1 Non-secure register is 0x3.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm* Generic Interrupt Controller Architecture Specification.
B3.17 ICV_CTLR_EL1, Interrupt Controller Virtual Control Register, EL1

ICV_CTLR_EL1 controls aspects of the behavior of the GIC virtual CPU interface and provides information about the features implemented.

Bit field descriptions

ICV_CTLR_EL1 is a 32-bit register and is part of the virtual GIC system registers functional group.

RES0, [31:16]

RES0 Reserved.

A3V, [15]

Affinity 3 Valid. The value is:

0x1 The virtual CPU interface logic supports non-zero values of Affinity 3 in SGI generation System registers.

SEIS, [14]

SEI Support. The value is:

0x0 The virtual CPU interface logic does not support local generation of SEIs.

IDbits, [13:11]

Identifier bits. The value is:

0x0 The number of physical interrupt identifier bits supported is 16 bits.

PRlbits, [10:8]

Priority bits. The value is:

0x4 Support 32 levels of physical priority (5 priority bits).

RES0, [7:2]

RES0 Reserved.

VEOImode, [1]

Virtual EOI mode. The possible values are:

0x0 ICV_EOIR0_EL1 and ICV_EOIR1_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to ICV_DIR_EL1 are UNPREDICTABLE.
ICV_EOIR0_EL1 and ICV_EOIR1_EL1 provide priority drop functionality only. ICV_DIR provides interrupt deactivation functionality.

VCBPR, [0]

Common Binary Point Register. Controls whether the same register is used for interrupt preemption of both virtual Group 0 and virtual Group 1 interrupts. The possible values are:

0

ICV_BPR0_EL1 determines the preemption group for virtual Group 0 interrupts only.

ICV_BPR1_EL1 determines the preemption group for virtual Group 1 interrupts.

1

ICV_BPR0_EL1 determines the preemption group for both virtual Group 0 and virtual Group 1 interrupts.

Reads of ICV_BPR1_EL1 return ICV_BPR0_EL1 plus one, saturated to 111. Writes to ICV_BPR1_EL1 are ignored.

Bit fields and details that are not provided in this description are architecturally defined. See the Arm® Generic Interrupt Controller Architecture Specification.
### AArch64 virtual interface control system register summary

The following table lists the AArch64 virtual interface control system registers that have implementation-defined bits.

See the Arm® Generic Interrupt Controller Architecture Specification for more information and a complete list of AArch64 virtual interface control system registers.

<table>
<thead>
<tr>
<th>Name</th>
<th>Op0</th>
<th>Op1</th>
<th>CRn</th>
<th>CRm</th>
<th>Op2</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICH_AP0R0_EL1</td>
<td>3</td>
<td>0</td>
<td>12</td>
<td>8</td>
<td>4</td>
<td>RW</td>
<td>B3.19 ICH_AP0R0_EL2, Interrupt Controller Hyp Active Priorities Group 0 Register 0, EL2 on page B3-284</td>
</tr>
<tr>
<td>ICH_AP1R0_EL1</td>
<td>3</td>
<td>0</td>
<td>19</td>
<td>9</td>
<td>0</td>
<td>RW</td>
<td>B3.20 ICH_AP1R0_EL2, Interrupt Controller Hyp Active Priorities Group 1 Register 0, EL2 on page B3-285</td>
</tr>
<tr>
<td>ICH_HCR_EL2</td>
<td>3</td>
<td>4</td>
<td>12</td>
<td>11</td>
<td>0</td>
<td>RW</td>
<td>B3.21 ICH_HCR_EL2, Interrupt Controller Hyp Control Register, EL2 on page B3-286</td>
</tr>
<tr>
<td>ICH_VTR_EL2</td>
<td>3</td>
<td>4</td>
<td>12</td>
<td>11</td>
<td>1</td>
<td>RO</td>
<td>B3.22 ICH_VMCR_EL2, Interrupt Controller Virtual Machine Control Register, EL2 on page B3-289</td>
</tr>
<tr>
<td>ICH_VMCR_EL2</td>
<td>3</td>
<td>4</td>
<td>12</td>
<td>11</td>
<td>7</td>
<td>RW</td>
<td>B3.23 ICH_VTR_EL2, Interrupt Controller VGIC Type Register, EL2 on page B3-291</td>
</tr>
</tbody>
</table>
B3.19 ICH_AP0R0_EL2, Interrupt Controller Hyp Active Priorities Group 0 Register 0, EL2

The ICH_AP0R0_EL2 provides information about Group 0 active priorities for EL2.

Bit field descriptions
This register is a 32-bit register and is part of:
• The GIC system registers functional group.
• The Virtualization registers functional group.
• The GIC host interface control registers functional group.

The core implements 5 bits of priority with 32 priority levels, corresponding to the 32 bits [31:0] of the register. The possible values for each bit are:

0x00000000  No interrupt active. This is the reset value.
0x00000001  Interrupt active for priority 0x0.
0x00000002  Interrupt active for priority 0x8.
...
0x80000000  Interrupt active for priority 0xF8.

Details that are not provided in this description are architecturally defined. See the Arm® Generic Interrupt Controller Architecture Specification.
The ICH_AP1R0_EL2 provides information about Group 1 active priorities for EL2.

**Bit field descriptions**

This register is a 32-bit register and is part of:
- The GIC system registers functional group.
- The Virtualization registers functional group.
- The GIC host interface control registers functional group.

The core implements 5 bits of priority with 32 priority levels, corresponding to the 32 bits [31:0] of the register. The possible values for each bit are:

- **0x00000000** No interrupt active. This is the reset value.
- **0x00000001** Interrupt active for priority 0x0.
- **0x00000002** Interrupt active for priority 0x8.
- ...
- **0x80000000** Interrupt active for priority 0xF8.

Details that are not provided in this description are architecturally defined. See the *Arm® Generic Interrupt Controller Architecture Specification*. 
B3.21  ICH_HCR_EL2, Interrupt Controller Hyp Control Register, EL2

ICH_HCR_EL2 controls the environment for VMs.

Bit field descriptions
ICH_HCR_EL2 is a 32-bit register and is part of:
- The GIC system registers functional group.
- The Virtualization registers functional group.
- The GIC host interface control registers functional group.

Figure B3-11  ICH_HCR_EL2 bit assignments

EOIcount, [31:27]
Number of outstanding deactivates.

RES0, [26:15]
RES0  Reserved.

TDIR, [14]
Trap Non-secure EL1 writes to ICC_DIR_EL1 and ICV_DIR_EL1. The possible values are:
- 0x0  Non-secure EL1 writes of ICC_DIR_EL1 and ICV_DIR_EL1 are not trapped to EL2, unless trapped by other mechanisms.
- 0x1  Non-secure EL1 writes of ICC_DIR_EL1 and ICV_DIR_EL1 are trapped to EL2.

TSEI, [13]
Trap all locally generated SEIs. The value is:
- 0  Locally generated SEIs do not cause a trap to EL2.

TALL1, [12]
Trap all Non-secure EL1 accesses to ICC_* and ICV_* System registers for Group 1 interrupts to EL2. The possible values are:
- 0x0  Non-secure EL1 accesses to ICC_* and ICV_* registers for Group 1 interrupts proceed as normal.
- 0x1  Non-secure EL1 accesses to ICC_* and ICV_* registers for Group 1 interrupts trap to EL2.

TALL0, [11]
Trap all Non-secure EL1 accesses to ICC_* and ICV_* System registers for Group 0 interrupts to EL2. The possible values are:
0x0  Non-secure EL1 accesses to ICC_* and ICV_* registers for Group 0 interrupts proceed as normal.

0x1  Non-secure EL1 accesses to ICC_* and ICV_* registers for Group 0 interrupts trap to EL2.

TC, [10]

Trap all Non-secure EL1 accesses to System registers that are common to Group 0 and Group 1 to EL2. The possible values are:

0x0  Non-secure EL1 accesses to common registers proceed as normal.

0x1  Non-secure EL1 accesses to common registers trap to EL2.

RES0, [9:8]

RES0  Reserved.

VGrp1DIE, [7]

VM Group 1 Disabled Interrupt Enable. The possible values are:

0  Maintenance interrupt disabled.

1  Maintenance interrupt signaled when ICH_VMCR_EL2.VENG1 is 0.

VGrp1EIE, [6]

VM Group 1 Enabled Interrupt Enable. The possible values are:

0  Maintenance interrupt disabled.

1  Maintenance interrupt signaled when ICH_VMCR_EL2.VENG1 is 1.

VGrp0DIE, [5]

VM Group 0 Disabled Interrupt Enable. The possible values are:

0  Maintenance interrupt disabled.

1  Maintenance interrupt signaled when ICH_VMCR_EL2.VENG0 is 0.

VGrp0EIE, [4]

VM Group 0 Enabled Interrupt Enable. The possible values are:

0  Maintenance interrupt disabled.

1  Maintenance interrupt signaled when ICH_VMCR_EL2.VENG0 is 1.

NPIE, [3]

No Pending Interrupt Enable. The possible values are:

0  Maintenance interrupt disabled.

1  Maintenance interrupt signaled while the List registers contain no interrupts in the pending state.

LRENPIE, [2]

List Register Entry Not Present Interrupt Enable. The possible values are:

0  Maintenance interrupt disabled.

1  Maintenance interrupt is asserted while the EOIcount field is not 0.
UIE, [1]

Underflow Interrupt Enable. The possible values are:

0    Maintenance interrupt disabled.
1    Maintenance interrupt is asserted if none, or only one, of the List register entries is marked as a valid interrupt.

En, [0]

Enable. The possible values are:

0    Virtual CPU interface operation disabled.
1    Virtual CPU interface operation enabled.

Bit fields and details that are not provided in this description are architecturally defined. See the Arm® Generic Interrupt Controller Architecture Specification.
ICH_VMCR_EL2, Interrupt Controller Virtual Machine Control Register, EL2

ICH_VMCR_EL2 enables the hypervisor to save and restore the virtual machine view of the GIC state.

Bit field descriptions
ICH_VMCR_EL2 is a 32-bit register and is part of:
- The GIC system registers functional group.
- The Virtualization registers functional group.
- The GIC host interface control registers functional group.

<table>
<thead>
<tr>
<th>Bit Field</th>
<th>Description</th>
<th>Value</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-24</td>
<td>VPMR, Virtual Priority Mask.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23-21</td>
<td>VBPR0, Virtual Binary Point Register, Group 0.</td>
<td>0x2</td>
<td></td>
</tr>
<tr>
<td>20-18</td>
<td>VBPR1, Virtual Binary Point Register, Group 1.</td>
<td>0x3</td>
<td></td>
</tr>
<tr>
<td>17-10</td>
<td>RES0, Reserved.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>VEOIM, Virtual EOI mode.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8-5</td>
<td>RES0, Reserved.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>VCBPR, Reserved.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure B3-12  ICH_VMCR_EL2 bit assignments
Virtual Common Binary Point Register. The possible values are:

0x0 ICV_BPR0_EL1 determines the preemption group for virtual Group 0 interrupts only.
ICV_BPR1_EL1 determines the preemption group for virtual Group 1 interrupts.

0x1 ICV_BPR0_EL1 determines the preemption group for both virtual Group 0 and virtual Group 1 interrupts.
Reads of ICV_BPR1_EL1 return ICV_BPR0_EL1 plus one, saturated to 111. Writes to ICV_BPR1_EL1 are IGNORED.

VFIQEn, [3]
Virtual FIQ enable. The value is:
0x1 Group 0 virtual interrupts are presented as virtual FIQs.

RES0, [2]
RES0 Reserved.

VENG1, [1]
Virtual Group 1 interrupt enable. The possible values are:
0x0 Virtual Group 1 interrupts are disabled.
0x1 Virtual Group 1 interrupts are enabled.

VENG0, [0]
Virtual Group 0 interrupt enable. The possible values are:
0x0 Virtual Group 0 interrupts are disabled.
0x1 Virtual Group 0 interrupts are enabled.

Bit fields and details that are not provided in this description are architecturally defined. See the Arm® Generic Interrupt Controller Architecture Specification.
B3.23 ICH_VTR_EL2, Interrupt Controller VGIC Type Register, EL2

ICH_VTR_EL2 reports supported GIC virtualization features.

**Bit field descriptions**
ICH_VTR_EL2 is a 32-bit register and is part of:
- The GIC system registers functional group.
- The Virtualization registers functional group.
- The GIC host interface control registers functional group.

![ICH_VTR_EL2 bit assignments](image)

**PRIbits, [31:29]**
Priority bits. The number of virtual priority bits implemented, minus one.
0x4 Priority implemented is 5-bit.

**PREbits, [28:26]**
The number of virtual preemption bits implemented, minus one. The value is:
0x4 Virtual preemption implemented is 5-bit.

**IDbits, [25:23]**
The number of virtual interrupt identifier bits supported. The value is:
0x0 Virtual interrupt identifier bits that are implemented is 16-bit.

**SEIS, [22]**
SEI Support. The value is:
0x0 The virtual CPU interface logic does not support generation of SEIs.

**A3V, [21]**
Affinity 3 Valid. The value is:
0x1 The virtual CPU interface logic supports non-zero values of Affinity 3 in SGI generation System registers.

**nV4, [20]**
Direct injection of virtual interrupts not supported. The value is:
0x0 The CPU interface logic supports direct injection of virtual interrupts.

**TDS, [19]**
Separate trapping of Non-secure EL1 writes to ICV_DIR_EL1 supported. The value is:

0x1  Implementation supports ICH_HCR_EL2.TDIR.

RES0, [18:5]

RES0  Reserved.

ListRegs, [4:0]

0x3  The number of implemented List registers, minus one.

Bit fields and details that are not provided in this description are architecturally defined. See the Arm® Generic Interrupt Controller Architecture Specification.
Chapter B4
Advanced SIMD and floating-point registers

This chapter describes the Advanced SIMD and floating-point registers.

It contains the following sections:

- **B4.1 AArch64 register summary** on page B4-294.
- **B4.2 FPCR, Floating-point Control Register** on page B4-295.
- **B4.3 FPSR, Floating-point Status Register** on page B4-297.
B4.1 AArch64 register summary

The core has several Advanced SIMD and floating-point system registers. Each register has a specific purpose, specific usage constraints, configurations, and attributes.

The following table gives a summary of the Cortex-A65 core Advanced SIMD and floating-point system registers.

Table B4-1  AArch64 Advanced SIMD and floating-point system registers

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FPCR</td>
<td>RW</td>
<td>0x00000000</td>
<td>See B4.2 FPCR, Floating-point Control Register on page B4-295.</td>
</tr>
<tr>
<td>FPSR</td>
<td>RW</td>
<td>0x00000000</td>
<td>See B4.3 FPSR, Floating-point Status Register on page B4-297.</td>
</tr>
</tbody>
</table>
B4.2  FPCR, Floating-point Control Register

The FPCR controls floating-point behavior.

**Bit field descriptions**

FPCR is a 32-bit register.

![FPCR bit assignments](image)

**RES0, [31:27]**

RES0  Reserved.

**AHP, [26]**

Alternative half-precision control bit. The possible values are:

0  IEEE half-precision format selected. This is the reset value.
1  Alternative half-precision format selected.

**DN, [25]**

Default NaN mode control bit. The possible values are:

0  NaN operands propagate through to the output of a floating-point operation. This is the reset value.
1  Any operation involving one or more NaNs returns the Default NaN.

**FZ, [24]**

Flush-to-zero mode control bit. The possible values are:

0  Flush-to-zero mode disabled. Behavior of the floating-point system is fully compliant with the IEEE 754 standard. This is the reset value.
1  Flush-to-zero mode enabled.

**RMode, [23:22]**

Rounding Mode control field. The encoding of this field is:

0b00  *Round to Nearest* (RN) mode. This is the reset value.
0b01  *Round towards Plus Infinity* (RP) mode.
0b10  *Round towards Minus Infinity* (RM) mode.
0b11  *Round towards Zero* (RZ) mode.

**RES0, [21:20]**

RES0  Reserved.
FZ16, [19]
Flush-to-zero mode control bit on half-precision data-processing instructions. The possible
values are:
0    Flush-to-zero mode disabled. Behavior of the floating-point system is fully compliant
    with the IEEE 754 standard. This is the default value.
1    Flush-to-zero mode enabled.

RES0, [18:0]
RES0    Reserved.

Configurations
There are no configurations.
Bit fields and details that are not provided in this description are architecturally defined. See the

Usage constraints
Accessing the FPCR
To access the FPCR:

MRS <Xt>, FPCR ; Read FPCR into Xt
MSR FPCR, <Xt> ; Write Xt to FPCR

Register access is encoded as follows:

<table>
<thead>
<tr>
<th>op0</th>
<th>op1</th>
<th>CRn</th>
<th>CRm</th>
<th>op2</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>011</td>
<td>0100</td>
<td>0100</td>
<td>000</td>
</tr>
</tbody>
</table>

Accessibility
This register is accessible as follows:

<table>
<thead>
<tr>
<th>EL0</th>
<th>EL1 (NS)</th>
<th>EL1 (S)</th>
<th>EL2 (SCR.NS = 1)</th>
<th>EL3 (SCR.NS = 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
</tr>
</tbody>
</table>
B4.3 FPSR, Floating-point Status Register

The FPSR provides floating-point system status information.

**Bit field descriptions**

FPSR is a 32-bit register.

![Diagram of FPSR bit assignments](image)

### N, [31]
Not in use, RES0. AArch64 floating-point comparisons set the PSTATE.N flag instead.

### Z, [30]
Not in use, RES0. AArch64 floating-point comparisons set the PSTATE.Z flag instead.

### C, [29]
Not in use, RES0. AArch64 floating-point comparisons set the PSTATE.C flag instead.

### V, [28]
Not in use, RES0. AArch64 floating-point comparisons set the PSTATE.V flag instead.

### QC, [27]
Cumulative saturation bit. This bit is set to 1 to indicate that an Advanced SIMD integer operation has saturated since a 0 was last written to this bit.

### RES0, [26:8]
Reserved, RES0.

### IDC, [7]
Input Denormal cumulative exception bit. This bit is set to 1 to indicate that the Input Denormal exception has occurred since 0 was last written to this bit.

### RES0, [6:5]
Reserved, RES0.

### IXC, [4]
Inexact cumulative exception bit. This bit is set to 1 to indicate that the Inexact exception has occurred since 0 was last written to this bit.

### UFC, [3]
Underflow cumulative exception bit. This bit is set to 1 to indicate that the Underflow exception has occurred since 0 was last written to this bit.
OFC, [2]
Overflow cumulative exception bit. This bit is set to 1 to indicate that the Overflow exception has occurred since 0 was last written to this bit.

DZC, [1]
Division by Zero cumulative exception bit. This bit is set to 1 to indicate that the Division by Zero exception has occurred since 0 was last written to this bit.

IOC, [0]
Invalid Operation cumulative exception bit. This bit is set to 1 to indicate that the Invalid Operation exception has occurred since 0 was last written to this bit.

Configurations
There are no configurations.

Bit fields and details that are not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Usage constraints

Accessing the FPSR

To access the FPSR:

```
MRS <Xt>, FPSR; Read FPSR into Xt
MSR FPSR, <Xt>; Write Xt to FPSR
```

Register access is encoded as follows:

<table>
<thead>
<tr>
<th>op0</th>
<th>op1</th>
<th>CRn</th>
<th>CRm</th>
<th>op2</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>011</td>
<td>0100</td>
<td>0100</td>
<td>001</td>
</tr>
</tbody>
</table>

Table B4-3 FPSR access encoding

Accessibility
This register is accessible as follows:

<table>
<thead>
<tr>
<th>EL0 (NS)</th>
<th>EL1 (S)</th>
<th>EL2 (SCR.NS = 1)</th>
<th>EL3 (SCR.NS = 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
</tr>
<tr>
<td>RW</td>
<td>RW</td>
<td>RW</td>
<td>RW</td>
</tr>
</tbody>
</table>
Part C
Debug descriptions
Chapter C1
Debug

This chapter describes the debug features of the core.

It contains the following sections:

- C1.1 About debug methods on page C1-302.
- C1.2 Debug functional description on page C1-303.
- C1.3 Debug register interfaces on page C1-305.
- C1.4 Debug events on page C1-307.
- C1.5 External debug interface on page C1-308.
C1.1 About debug methods

The core is part of a debug system and supports both self-hosted and external debug.

The following figure shows a typical external debug system.

![Internal debug system diagram]

**Debug host**
A computer, for example a personal computer, that is running a software debugger such as the DS-5 Debugger. With the debug host, you can issue high-level commands, such as setting a breakpoint at a certain location or examining the contents of a memory address.

**Protocol converter**
The debug host sends messages to the debug target using an interface such as Ethernet. However, the debug target typically implements a different interface protocol. A device such as DSTREAM is required to convert between the two protocols.

**Debug target**
The lowest level of the system implements system support for the protocol converter to access the debug unit using the *Advanced Peripheral Bus* (APB) slave interface. An example of a debug target is a development system with a test chip or a silicon part with a core.

**Debug unit**
Helps debugging software that is running on the core:
- Hardware systems that are based on the core.
- Operating systems.
- Application software.

With the debug unit, you can:
- Stop program execution.
- Examine and alter process and coprocessor state.
- Examine and alter memory and the state of the input or output peripherals.
- Restart the core.

For self-hosted debug, the debug target runs additional debug monitor software that runs on the Cortex-A65 core itself. This way, it does not require expensive interface hardware to connect a second host computer.
C1.2 Debug functional description

This section describes the trace, debug, and test features supported by Cortex-A65. It includes Armv8-A Debug, CoreSight Debug, and cache Debug.

Arm®v8-A debug architecture support

The Cortex-A65 core supports the Armv8-A debug architecture.

The core allows access to the internal debug functionality and registers either through a memory-mapped area on the external AMBA APBv3 slave port, or by using CP14 system coprocessor operations from software running on the core.

The core implements six hardware breakpoints, four watchpoints, and a Debug Communications Channel (DCC). Four of the breakpoints match only against virtual address, the other two breakpoints match against either virtual address or context ID. All watchpoints can be linked to either of the virtual address or context-ID matching breakpoints to allow a memory request to be trapped in a given process context.

Note

Armv7 debug map support

For backwards compatibility, and to reduce the address space required for the debug map, a 4k page-based memory map is also supported.

CoreSight debug

The Cortex-A65 core integrates several CoreSight debug related components to aid system debug in conjunction with CoreSight SoC.

These components include:

• Per-core Embedded Trace Macrocell (ETM).
• Per-core Cross Trigger Interface (CTI).
• Cross Trigger Matrix (CTM).
• Debug-over-power-down support.

The following figure shows the Cortex-A65 CoreSight debug components.

Note

The DAP connection is shown for completeness.
The debug components are split into two groups. Some components are in the cluster itself and the rest are in a separate block named the DebugBlock. It allows you to put the DebugBlock in a separate power domain and place it physically with other CoreSight logic in the SoC, rather than close to the cluster.

The connection between the cluster and the DebugBlock consists of a pair of APB interfaces, one in each direction. All debug traffic, except the authentication interface, takes place over this interface as read or write APB transactions. It includes register reads, writes, and CTI triggers.

All debug components are controlled through the primary Debug APB interface on the DebugBlock, and form a standard CoreSight interface. Requests on this bus are decoded by the APB decoder before being sent to the appropriate component in the DebugBlock or in the cluster. The per-core CTIs are connected to a CoreSight CTM.

Each core contains an ETM, PMU, and debug component that are accessed using the debug APB bus. This block conforms to the Armv8-A Debug Architecture Specification.

The core supports debug-over-power-down using modules contained in the DebugBlock that mirror key core information such as core ID. These allow the JTAG scan chain connection to be maintained while the core is powered down.

The ETM in each core outputs trace on a 32-bit AMBA 4 ATBv1.1 interface. There is one interface per core.
C1.3 Debug register interfaces

The core implements the Armv8-A Debug architecture and debug events.

They are described in the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.

The Debug architecture defines a set of debug registers. The debug register interfaces provide access to these registers from:

- Software running on the core.
- An external debugger.

C1.3.1 Core interfaces

System register access allows the core to directly access certain debug registers.

The external debug interface enables both external and self-hosted debug agents to access debug registers. Access to the debug registers is partitioned as follows:

**Debug registers**

This function is system register based and memory-mapped. You can access the debug register map using the APB slave port.

**Performance monitor**

This function is system register based and memory-mapped. You can access the performance monitor registers using the APB slave port.

**Trace registers**

This function is memory-mapped.

Related references

C1.5 External debug interface on page C1-308

C1.3.2 Breakpoints and watchpoints

The core supports six breakpoints, four watchpoints, and a standard Debug Communications Channel (DCC) for each thread.

A breakpoint consists of a breakpoint control register and a breakpoint value register. These two registers are referred to as a **Breakpoint Register Pair** (BRP).

Four of the breakpoints (BRP 0-3) match only to virtual address and the other two (BRP 4 and 5) match against either virtual address or context ID, or VMID. All the watchpoints can be linked to two breakpoints (BRP 4 and 5) to enable a memory request to be trapped in a given process context.

C1.3.3 Effects of resets on debug registers

The core has the following reset signals that affect the debug registers:

**nCPUPORESET**

This signal initializes the core logic, including the debug, ETM trace unit, breakpoint, watchpoint logic, and performance monitors logic. This maps to a Cold reset that covers reset of the core logic and the integrated debug functionality.

**nCORERESET**

This signal resets some of the debug and performance monitor logic. This maps to a Warm reset that covers reset of the core logic.

C1.3.4 External access permissions to debug registers

External access permission to the debug registers is subject to the conditions at the time of the access.

The following table describes the core response to accesses through the external debug interface.
## Table C1-1  External access conditions to registers

<table>
<thead>
<tr>
<th>Name</th>
<th>Condition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Off</td>
<td>EDPRSR.PU is 0</td>
<td>Core power domain is completely off, or in a low-power state where the core power domain registers cannot be accessed. If debug power is off, then all external debug and memory-mapped register accesses return an error.</td>
</tr>
<tr>
<td>DLK</td>
<td>DoubleLockStatus() == TRUE (EDPRSR.DLK is 1)</td>
<td>OS Double Lock is locked.</td>
</tr>
<tr>
<td>OSLK</td>
<td>OSLR_EL1.OSLK is 1</td>
<td>OS Lock is locked.</td>
</tr>
<tr>
<td>EDAD</td>
<td>AllowExternalDebugAccess() ==FALSE</td>
<td>External debug access is disabled. When an error is returned because of an EDAD condition code, and this is the highest priority error condition, EDPRSR.SDAD is set to 1. Otherwise SDAD is unchanged.</td>
</tr>
<tr>
<td>Default</td>
<td>-</td>
<td>None of the conditions apply, normal access.</td>
</tr>
</tbody>
</table>

The following table shows an example of external register access condition codes for access to a performance monitor register. To determine the access permission for the register, scan the columns from left to right. Stop at the first column a condition is true, the entry gives the access permission of the register and scanning stops.

## Table C1-2  External register condition code example

<table>
<thead>
<tr>
<th>Off</th>
<th>DLK</th>
<th>OSLK</th>
<th>EDAD</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>RO</td>
</tr>
</tbody>
</table>
C1.4 Debug events

A debug event can be a software debug event or a halting debug event.

A core responds to a debug event in one of the following ways:
• Ignores the debug event.
• Takes a debug exception.
• Enters debug state.

See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile for more information about the debug events.

C1.4.1 Watchpoint debug events

In the Cortex-A65 core, watchpoint debug events are always synchronous.

C1.4.2 Debug OS Lock

Debug OS Lock is set by the powerup reset, nCPUPORESET.

For normal behavior of debug events and debug register accesses, Debug OS Lock must be cleared. For more information, see the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.

Related references
C1.5 External debug interface on page C1-308
A3.1 About clocks, resets, and input synchronization on page A3-40
C1.5   External debug interface

For information about external debug interface, including debug memory map and debug signals, see the "Arm® DynamIQ™ Shared Unit Technical Reference Manual."
This chapter describes the *Performance Monitor Unit* (PMU).

It contains the following sections:

- *C2.1 About the PMU* on page C2-310.
- *C2.2 PMU functional description* on page C2-311.
- *C2.3 External register access permissions to the PMU registers* on page C2-312.
- *C2.4 PMU events* on page C2-313.
- *C2.5 PMU interrupts* on page C2-328.
- *C2.6 Exporting PMU events* on page C2-329.
C2.1 About the PMU

The Cortex-A65 core includes performance monitors that enable you to gather various statistics on the operation of the core and its memory system during runtime. These provide useful information about the behavior of the core that you can use when debugging or profiling code.

The PMU provides six counters. Each counter can count any of the events available in the core. The absolute counts recorded might vary because of pipeline effects. This has negligible effect except in cases where the counters are enabled for a very short time.

Related references
C2.4 PMU events on page C2-313
C2.2 PMU functional description

This section describes the functionality of the PMU.

The PMU includes the following interfaces and counters:

**Event interface**
Events from all other units from across the design are provided to the PMU.

**System register and APB interface**
You can program the PMU registers using the system registers or the external APB interface.

**Counters**
The PMU has 32-bit counters that increment when they are enabled, based on events, and a 64-bit cycle counter.

**PMU register interfaces**
The Cortex-A65 core supports access to the performance monitor registers from the internal system register interface and a memory-mapped interface.
C2.3 External register access permissions to the PMU registers

External access permission to the PMU registers is subject to the conditions at the time of the access. The following table describes the core response to accesses through the external debug and memory-mapped interfaces.

<table>
<thead>
<tr>
<th>Name</th>
<th>Condition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Off</td>
<td>EDPRSR.PU is 0</td>
<td>Core power domain is completely off, or in a low-power state where the core power domain registers cannot be accessed.</td>
</tr>
<tr>
<td>DLK</td>
<td>EDPRSR.DLK is 1</td>
<td>OS Double Lock is locked.</td>
</tr>
<tr>
<td>OSLK</td>
<td>OSLR_EL1.OSLK is 1</td>
<td>OS Lock is locked.</td>
</tr>
<tr>
<td>EPMAD</td>
<td>AllowExternalPMUAccess() == FALSE</td>
<td>External performance monitors access is disabled. When an error is returned because of an EPMAD condition code, and this is the highest priority error condition, EDPRSR.SPMAD is set to 1. Otherwise SPMAD is unchanged.</td>
</tr>
<tr>
<td>Default</td>
<td>-</td>
<td>None of the conditions apply, normal access.</td>
</tr>
</tbody>
</table>

The following table shows an example of external register condition codes for access to a performance monitor register. To determine the access permission for the register, scan the columns from left to right. Stop at the first column whose condition is true, the entry gives the register access permission and scanning stops.

<table>
<thead>
<tr>
<th>Off</th>
<th>DLK</th>
<th>OSLK</th>
<th>EPMAD</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>RO</td>
</tr>
</tbody>
</table>
C2.4 PMU events

The following table shows the events that are generated and the numbers that the PMU uses to reference the events. The table also shows the bit position of each event on the event bus. Event reference numbers that are not listed are reserved.

<table>
<thead>
<tr>
<th>Event number</th>
<th>PMU event bus (to trace)</th>
<th>Event mnemonic</th>
<th>Event description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>-</td>
<td>SW_INCR</td>
<td>Instruction architecturally executed, condition code check pass, software increment.</td>
</tr>
<tr>
<td>0x01</td>
<td>[0]</td>
<td>L1I_CACHE_REFILL</td>
<td>Level 1 instruction cache refill.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This event counts any instruction fetch which misses in the cache.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The following instructions are not counted:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Cache maintenance instructions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Non-cacheable accesses.</td>
</tr>
<tr>
<td>0x02</td>
<td>[1]</td>
<td>L1I_TLB_REFILL</td>
<td>Level 1 instruction TLB refill.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This event counts any refill of the instruction L1 TLB from the L2 TLB.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This includes refills which result in a translation fault.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The following instructions are not counted:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• TLB maintenance instructions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This event counts regardless of whether the MMU is enabled.</td>
</tr>
<tr>
<td>0x03</td>
<td>[2]</td>
<td>L1D_CACHE_REFILL</td>
<td>Level 1 data cache refill.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This event counts any load or store operation or pagewalk access which causes data to be read from outside the L1, including accesses which do not allocate into L1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The following instructions are not counted:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Cache maintenance instructions and prefetches.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Stores of an entire cache line, even if they make a coherency request outside the L1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Partial cache line writes which do not allocate into the L1 cache.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Non-cacheable accesses.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This event counts the sum of L1D_CACHE_REFILL_RD and L1D_CACHE_REFILL_WR.</td>
</tr>
<tr>
<td>0x04</td>
<td>[3]</td>
<td>L1D_CACHE</td>
<td>Level 1 data cache access.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This event counts any load or store operation or pagewalk access which looks up in the L1 data cache. In particular, any access which could count the L1D_CACHE_REFILL event causes this event to count.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The following instructions are not counted:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Cache maintenance instructions and prefetches.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Non-cacheable accesses.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This event counts the sum of L1D_CACHE_RD and L1D_CACHE_WR.</td>
</tr>
<tr>
<td>Event number</td>
<td>PMU event bus (to trace)</td>
<td>Event mnemonic</td>
<td>Event description</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------------</td>
<td>-------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>0x05</td>
<td>[4]</td>
<td>L1D_TLB_REFILL</td>
<td>Level 1 data TLB refill. This event counts any refill of the data L1 TLB from the L2 TLB. This includes refills which result in a translation fault.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The following instructions are not counted:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• TLB maintenance instructions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This event counts regardless of whether the MMU is enabled.</td>
</tr>
<tr>
<td>0x06</td>
<td>[5]</td>
<td>LD RETIRED</td>
<td>Instruction architecturally executed, condition code check pass, load. This event counts all load and prefetch instructions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This includes the Armv8.1-A atomic instructions, other than the ST* variants.</td>
</tr>
<tr>
<td>0x07</td>
<td>[6]</td>
<td>ST RETIRED</td>
<td>Instruction architecturally executed, condition code check pass, store. This event counts all store instructions and DC ZVA.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This includes all the Armv8.1-A atomic instructions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The following instructions are not counted:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Store-Exclusive instructions which fail.</td>
</tr>
<tr>
<td>0x08</td>
<td>[7]</td>
<td>INST RETIRED</td>
<td>Instruction architecturally executed. This event counts all retired instructions, including those that fail their condition check.</td>
</tr>
<tr>
<td>0x09</td>
<td>[8]</td>
<td>EXC TAKEN</td>
<td>Exception taken.</td>
</tr>
<tr>
<td>0x0A</td>
<td>[9]</td>
<td>EXC RETURN</td>
<td>Instruction architecturally executed, condition code check pass, exception return.</td>
</tr>
<tr>
<td>0x0B</td>
<td>[10]</td>
<td>CID WRITE RETIRED</td>
<td>Instruction architecturally executed, condition code check pass, write to CONTEXTIDR.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This event only counts writes via the CONTEXTIDR_EL1 mnemonic in AArch64.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The following instructions are not counted:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Writes to CONTEXTIDR_EL12 and CONTEXTIDR_EL2.</td>
</tr>
<tr>
<td>0x0C</td>
<td>[11]</td>
<td>PC WRITE RETIRED</td>
<td>Instruction architecturally executed, condition code check pass, software change of the PC.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This event counts all branches taken and popped from the branch monitor. This excludes exception entries, debug entries, and CCFAIL branches.</td>
</tr>
<tr>
<td>0x0D</td>
<td>[12]</td>
<td>BR IMMED RETIRED</td>
<td>Instruction architecturally executed, immediate branch.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This event counts all branches decoded as immediate branches, taken or not, and popped from the branch monitor. This excludes exception entries, debug entries, and CCFAIL branches.</td>
</tr>
<tr>
<td>Event number</td>
<td>PMU event bus (to trace)</td>
<td>Event mnemonic</td>
<td>Event description</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------</td>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>0x0E</td>
<td>[13]</td>
<td>BR_RETURN_RETIRED</td>
<td>Instruction architecturally executed, condition code check pass, procedure return.</td>
</tr>
<tr>
<td>0x10</td>
<td>[15]</td>
<td>BR_MIS_PRED</td>
<td>Mispredicted or not predicted branch speculatively executed. This event counts any predictable branch instruction which is mispredicted either due to dynamic misprediction or because the MMU is off and the branches are statically predicted not taken.</td>
</tr>
<tr>
<td>0x11</td>
<td>-</td>
<td>CPU_CYCLES</td>
<td>The counter increments on every cycle.</td>
</tr>
<tr>
<td>0x12</td>
<td>[16]</td>
<td>BR_PRED</td>
<td>Predictable branch speculatively executed. This event counts all predictable branches.</td>
</tr>
<tr>
<td>0x13</td>
<td>[17]</td>
<td>MEM_ACCESS</td>
<td>Data memory access. This event counts memory accesses due to load or store instructions. The following instructions are not counted: • Instruction fetches. • Cache maintenance instructions. • Translation table walks or prefetches. This event counts the sum of MEM_ACCESS_RD and MEM_ACCESS_WR.</td>
</tr>
<tr>
<td>0x14</td>
<td>[18]</td>
<td>L1I_CACHE</td>
<td>Level 1 instruction cache access. This event counts any instruction fetch which accesses the L1 instruction cache. The following instructions are not counted: • Cache maintenance instructions. • Non-cacheable accesses.</td>
</tr>
<tr>
<td>0x15</td>
<td>[19]</td>
<td>L1D_CACHE_WB</td>
<td>Level 1 data cache Write-Back. This event counts any write back of data from the L1 data cache to L2 or L3. This counts both victim line evictions and snoops, including cache maintenance operations. The following instructions are not counted: • Invalidations which do not result in data being transferred out of the L1. • Full-line writes which write to L2 without writing L1, such as write-streaming mode.</td>
</tr>
<tr>
<td>Event number</td>
<td>PMU event bus (to trace)</td>
<td>Event mnemonic</td>
<td>Event description</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------</td>
<td>----------------</td>
<td>-------------------</td>
</tr>
</tbody>
</table>
| 0x16         | [20]                     | L2D_CACHE      | Level 2 data cache access.  
• If the core is configured with a per-core L2 cache:  
  This event counts any transaction from L1 which looks up in the L2 cache, and any write-back from the L1 to the L2. Snoops from outside the core and cache maintenance operations are not counted.  
• If the core is not configured with a per-core L2 cache:  
  This event counts the cluster cache event, as defined by L3D_CACHE.  
• If there is neither a per-core cache nor a cluster cache configured, then this event is not implemented. |
| 0x17         | [21]                     | L2D_CACHE_REFILL | Level 2 data cache refill.  
• If the core is configured with a per-core L2 cache:  
  This event counts any cacheable transaction from L1 which causes data to be read from outside the core. L2 refills caused by stashes into L2 should not be counted.  
• If the core is not configured with a per-core L2 cache:  
  This event counts the cluster cache event, as defined by L3D_CACHE_REFILL.  
• If there is neither a per-core cache nor a cluster cache configured, then this event is not implemented. |
| 0x18         | [22]                     | L2D_CACHE_WB    | Level 2 data cache Write-Back.  
• If the core is configured with a per-core L2 cache:  
  This event counts any write back of data from the L2 cache to outside the core. This includes snoops to the L2 which return data, regardless of whether they cause an invalidation. Invalidations from the L2 which do not write data outside of the core and snoops which return data from the L1 are not counted.  
• If the core is not configured with a per-core L2 cache, this event is not implemented. |
| 0x19         | [23]                     | BUS_ACCESS      | Bus access.  
This event counts for every beat of data transferred over the data channels between the core and the SCU. If both read and write data beats are transferred on a given cycle, this event is counted twice on that cycle.  
This event counts the sum of BUS_ACCESS_RD and BUS_ACCESS_WR. |
| 0x1A         | [24]                     | MEMORY_ERROR    | Local memory error.  
This event counts any correctable or uncorrectable memory error (ECC or parity) in the protected core RAMs. |
| 0x1B         | -                        | INST_SPEC       | Operation speculatively executed.  
This event duplicates INST_RETIRED. |
<table>
<thead>
<tr>
<th>Event number</th>
<th>PMU event bus (to trace)</th>
<th>Event mnemonic</th>
<th>Event description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1C</td>
<td>[25]</td>
<td>TTBR_WRITE_RETIRED</td>
<td>Instruction architecturally executed, condition code check pass, write to TTBR. This event only counts writes to TTBR0_EL1/TTBR1_EL1 in AArch64. The following instructions are not counted: • Accesses to TTBR0_EL12/TTBR1_EL12 or TTBR0_EL2/TTBR1_EL2.</td>
</tr>
<tr>
<td>0x1D</td>
<td>-</td>
<td>BUS_CYCLES</td>
<td>Bus cycles. This event duplicates CPU_CYCLES.</td>
</tr>
<tr>
<td>0x1E</td>
<td>-</td>
<td>CHAIN</td>
<td>Odd performance counter chain mode.</td>
</tr>
<tr>
<td>0x20</td>
<td>[26]</td>
<td>L2D_CACHE_ALLOCATE</td>
<td>Level 2 data cache allocation without refill.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• If the core is configured with a per-core L2 cache:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This event counts any full cache line write into the L2 cache which does not cause a linefill, including write-backs from L1 to L2 and full-line writes which do not allocate into L1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• If the core is not configured with a per-core L2 cache:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This event counts the cluster cache event, as defined by L3D_CACHE_ALLOCATE.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• If there is neither a per-core cache nor a cluster cache configured, this event is not implemented.</td>
</tr>
<tr>
<td>0x21</td>
<td>[27]</td>
<td>BR_RETIRED</td>
<td>Instruction architecturally executed, branch. This event counts all branches, taken or not, popped from the branch monitor. This excludes exception entries, debug entries, and CCFAIL branches. In the Cortex-A65 core, an ISB is a branch, and even micro architectural ISBs are counted.</td>
</tr>
<tr>
<td>0x22</td>
<td>[28]</td>
<td>BR__MIS_PRED_RETIRED</td>
<td>Instruction architecturally executed, mispredicted branch. This event counts any branch counted by BR_RETIRED which is not correctly predicted and causes a pipeline flush.</td>
</tr>
<tr>
<td>0x23</td>
<td>[29]</td>
<td>STALL_FRONTEND</td>
<td>No operation issued because of the frontend. The counter counts on any cycle when no operations are issued due to the instruction queue being empty.</td>
</tr>
<tr>
<td>0x24</td>
<td>[30]</td>
<td>STALL_BACKEND</td>
<td>No operation issued because of the backend. The counter counts on any cycle when no operations are issued due to a pipeline stall.</td>
</tr>
<tr>
<td>Event number</td>
<td>PMU event bus (to trace)</td>
<td>Event mnemonic</td>
<td>Event description</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------</td>
<td>----------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>0x25</td>
<td>[31]</td>
<td>L1D_TLB</td>
<td>Level 1 data TLB access. This event counts any load or store operation which accesses the data L1 TLB. If both a load and a store are executed on a cycle, this event counts twice. This event counts regardless of whether the MMU is enabled.</td>
</tr>
<tr>
<td>0x26</td>
<td>[32]</td>
<td>L1I_TLB</td>
<td>Level 1 instruction TLB access. This event counts any instruction fetch which accesses the instruction L1 TLB. This event counts regardless of whether the MMU is enabled.</td>
</tr>
<tr>
<td>0x29</td>
<td>[33]</td>
<td>L3D_CACHE_ALLOCATE</td>
<td>Attributable Level 3 unified cache allocation without refill. • If the core is configured with a per-core L2 cache and the cluster is configured with an L3 cache: This event counts any full cache line write into the L3 cache which does not cause a linefill, including write-backs from L2 to L3 and full-line writes which do not allocate into L2. • If either the core is configured without a per-core L2 or the cluster is configured without an L3 cache, this event is not implemented.</td>
</tr>
<tr>
<td>0x2A</td>
<td>[34]</td>
<td>L3D_CACHE_REFILL</td>
<td>Attributable Level 3 unified cache refill. • If the core is configured with a per-core L2 cache and the cluster is configured with an L3 cache: This event counts for any cacheable read transaction returning data from the SCU for which the data source was outside the cluster. Transactions such as ReadUnique are counted here as “read” transactions, even though they can be generated by store instructions. • If either the core is configured without a per-core L2 or the cluster is configured without an L3 cache, this event is not implemented.</td>
</tr>
<tr>
<td>0x2B</td>
<td>[35]</td>
<td>L3D_CACHE</td>
<td>Attributable Level 3 unified cache access. • If the core is configured with a per-core L2 cache and the cluster is configured with an L3 cache: This event counts for any cacheable read transaction returning data from the SCU, or for any cacheable write to the SCU. • If either the core is configured without a per-core L2 or the cluster is configured without an L3 cache, this event is not implemented.</td>
</tr>
<tr>
<td>0x2D</td>
<td>[36]</td>
<td>L2D_TLB_REFILL</td>
<td>Attributable Level 2 unified TLB refill. This event counts on any refill of the L2 TLB, caused by either an instruction or data access. This event does not count if the MMU is disabled.</td>
</tr>
<tr>
<td>Event number</td>
<td>PMU event bus (to trace)</td>
<td>Event mnemonic</td>
<td>Event description</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------</td>
<td>----------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>0x2F</td>
<td>[37]</td>
<td>L2D_TLB</td>
<td>Attributable Level 2 unified TLB access. This event counts on any access to the L2 TLB (caused by a refill of any of the L1 TLBs). This event does not count if the MMU is disabled.</td>
</tr>
<tr>
<td>0x34</td>
<td>[39]</td>
<td>DTLB_WALK</td>
<td>Access to data TLB that caused a page table walk. This event counts on any data access which causes L2D_TLB_REFILL to count.</td>
</tr>
<tr>
<td>0x35</td>
<td>[40]</td>
<td>ITLB_WALK</td>
<td>Access to instruction TLB that caused a page table walk. This event counts on any instruction access which causes L2D_TLB_REFILL to count.</td>
</tr>
<tr>
<td>0x36</td>
<td>[41]</td>
<td>LL_CACHE_RD</td>
<td>Last level cache access, read.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• If CPUECTLR.EXTLLC is set: This event counts any cacheable read transaction which returns a data source of &quot;interconnect cache&quot;.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• If CPUECTLR.EXTLLC is not set: This event is a duplicate of the L*D_CACHE_RD event corresponding to the last level of cache implemented – L3D_CACHE_RD if both per-core L2 and cluster L3 are implemented, L2D_CACHE_RD if only one is implemented, or L1D_CACHE_RD if neither is implemented.</td>
</tr>
<tr>
<td>0x37</td>
<td>[42]</td>
<td>LL_CACHE_MISS_RD</td>
<td>Last level cache miss, read.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• If CPUECTLR.EXTLLC is set: This event counts any cacheable read transaction which returns a data source of &quot;DRAM&quot;, &quot;remote&quot; or &quot;inter-cluster peer&quot;.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• If CPUECTLR.EXTLLC is not set: This event is a duplicate of the L*D_CACHE_REFILL_RD event corresponding to the last level of cache implemented – L3D_CACHE_REFILL_RD if both per-core L2 and cluster L3 are implemented, L2D_CACHE_REFILL_RD if only one is implemented, or L1D_CACHE_REFILL_RD if neither is implemented.</td>
</tr>
<tr>
<td>0x38</td>
<td>[38]</td>
<td>REMOTE_ACCESS_RD</td>
<td>Access to another socket in a multi-socket system, read. This event counts any read transaction which returns a data source of &quot;remote&quot;.</td>
</tr>
<tr>
<td>Event number</td>
<td>PMU event bus (to trace)</td>
<td>Event mnemonic</td>
<td>Event description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------</td>
<td>----------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>0x40</td>
<td>-</td>
<td>L1D_CACHE_RD</td>
<td>Level 1 data cache access, read. This event counts any load operation or pagewalk access which looks up in the L1 data cache. In particular, any access which could count the L1D_CACHE_REFILL_RD event causes this event to count. The following instructions are not counted: • Cache maintenance instructions and prefetches. • Non-cacheable accesses.</td>
</tr>
<tr>
<td>0x41</td>
<td>-</td>
<td>L1D_CACHE_WR</td>
<td>Level 1 data cache access, write. This event counts any store operation which looks up in the L1 data cache. In particular, any access which could count the L1D_CACHE_REFILL event causes this event to count. The following instructions are not counted: • Cache maintenance instructions and prefetches. • Non-cacheable accesses.</td>
</tr>
<tr>
<td>0x42</td>
<td>-</td>
<td>L1D_CACHE_REFILL_RD</td>
<td>Level 1 data cache refill, read. This event counts any load operation or pagewalk access which causes data to be read from outside the L1, including accesses which do not allocate into L1. The following instructions are not counted: • Cache maintenance instructions and prefetches. • Non-cacheable accesses.</td>
</tr>
<tr>
<td>0x43</td>
<td>-</td>
<td>L1D_CACHE_REFILL_WR</td>
<td>Level 1 data cache refill, write. This event counts any store operation which causes data to be read from outside the L1, including accesses which do not allocate into L1. The following instructions are not counted: • Cache maintenance instructions and prefetches. • Stores of an entire cache line, even if they make a coherency request outside the L1. • Partial cache line writes which do not allocate into the L1 cache. • Non-cacheable accesses.</td>
</tr>
<tr>
<td>0x44</td>
<td>-</td>
<td>L1D_CACHE_REFILL_INNER</td>
<td>Level 1 data cache refill, inner. This event counts any L1 D-cache linefill (as counted by L1D_CACHE_REFILL) which hits in the L2 cache, L3 cache, or another core in the cluster.</td>
</tr>
<tr>
<td>0x45</td>
<td>-</td>
<td>L1D_CACHE_REFILL_OUTER</td>
<td>Level 1 data cache refill, outer. This event counts any L1 D-cache linefill (as counted by L1D_CACHE_REFILL) which does not hit in the L2 cache, L3 cache, or another core in the cluster, and instead obtains data from outside the cluster.</td>
</tr>
<tr>
<td>Event number</td>
<td>PMU event bus (to trace)</td>
<td>Event mnemonic</td>
<td>Event description</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------</td>
<td>----------------</td>
<td>-------------------</td>
</tr>
</tbody>
</table>
| 0x50         | -                        | L2D_CACHE_RD   | Level 2 cache access, read.  
• If the core is configured with a per-core L2 cache:  
This event counts any read transaction from L1 which looks up in the L2 cache. Snoops from outside the core are not counted.  
• If the core is configured without a per-core L2 cache:  
This event counts the cluster cache event, as defined by L3D_CACHE_RD.  
• If there is neither a per-core cache nor a cluster cache configured, this event is not implemented |
| 0x51         | -                        | L2D_CACHE_WR   | Level 2 cache access, write.  
• If the core is configured with a per-core L2 cache:  
This event counts any write transaction from L1 which looks up in the L2 cache or any write-back from L1 which allocates into the L2 cache. Snoops from outside the core are not counted.  
• If the core is configured without a per-core L2 cache:  
This event counts the cluster cache event, as defined by L3D_CACHE_WR.  
• If there is neither a per-core cache nor a cluster cache configured, this event is not implemented |
| 0x52         | -                        | L2D_CACHE_REFILL_RD | Level 2 cache refill, read.  
• If the core is configured with a per-core L2 cache:  
This event counts any cacheable read transaction from L1 which causes data to be read from outside the core. L2 refills caused by stashes into L2 should not be counted. Transactions such as ReadUnique are counted here as “read” transactions, even though they can be generated by store instructions.  
• If the core is configured without a per-core L2 cache:  
This event counts the cluster cache event, as defined by L3D_CACHE_REFILL_RD.  
• If there is neither a per-core cache nor a cluster cache configured, this event is not implemented |
| 0x53         | -                        | L2D_CACHE_REFILL_WR | Level 2 cache refill, write.  
• If the core is configured with a per-core L2 cache:  
This event counts any write transaction from L1 which causes data to be read from outside the core. L2 refills caused by stashes into L2 should not be counted. Transactions such as ReadUnique are not counted as write transactions.  
• If the core is configured without a per-core L2 cache:  
This event counts the cluster cache event, as defined by L3D_CACHE_REFILL_WR.  
• If there is neither a per-core cache nor a cluster cache configured, this event is not implemented |
<table>
<thead>
<tr>
<th>Event number</th>
<th>PMU event bus (to trace)</th>
<th>Event mnemonic</th>
<th>Event description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x60</td>
<td>-</td>
<td>BUS_ACCESS_RD</td>
<td>Bus access, read. This event counts for every beat of data transferred over the read data channel between the core and the SCU.</td>
</tr>
<tr>
<td>0x61</td>
<td>-</td>
<td>BUS_ACCESS_WR</td>
<td>Bus access, write. This event counts for every beat of data transferred over the write data channel between the core and the SCU.</td>
</tr>
<tr>
<td>0x66</td>
<td>-</td>
<td>MEM_ACCESS_RD</td>
<td>Data memory access, read. This event counts memory accesses due to load instructions. The following instructions are not counted: • Instruction fetches. • Cache maintenance instructions. • Translation table walks. • Prefetches.</td>
</tr>
<tr>
<td>0x67</td>
<td>-</td>
<td>MEM_ACCESS_WR</td>
<td>Data memory access, write. This event counts memory accesses due to store instructions. The following instructions are not counted: • Instruction fetches. • Cache maintenance instructions. • Translation table walks. • Prefetches.</td>
</tr>
<tr>
<td>0x68</td>
<td>-</td>
<td>UNALIGNED_LD_SPEC</td>
<td>Unaligned access, read</td>
</tr>
<tr>
<td>0x69</td>
<td>-</td>
<td>UNALIGNED_ST_SPEC</td>
<td>Unaligned access, write.</td>
</tr>
<tr>
<td>0x6A</td>
<td>-</td>
<td>UNALIGNED_LDST_SPEC</td>
<td>Unaligned access.</td>
</tr>
<tr>
<td>0x70</td>
<td>-</td>
<td>LD_SPEC</td>
<td>Operation speculatively executed, load. This event duplicates LD RETIRED.</td>
</tr>
<tr>
<td>0x71</td>
<td>-</td>
<td>ST_SPEC</td>
<td>Operation speculatively executed, store. This event duplicates ST RETIRED.</td>
</tr>
<tr>
<td>0x72</td>
<td>-</td>
<td>LDST_SPEC</td>
<td>Operation speculatively executed, load or store. This event counts the sum of LD_SPEC and ST_SPEC.</td>
</tr>
<tr>
<td>0x73</td>
<td>-</td>
<td>DP_SPEC</td>
<td>Operation speculatively executed, integer data processing. This event counts retired integer data-processing instructions.</td>
</tr>
<tr>
<td>0x74</td>
<td>-</td>
<td>ASE_SPEC</td>
<td>Operation speculatively executed, Advanced SIMD instruction. This event counts retired Advanced SIMD instructions.</td>
</tr>
<tr>
<td>0x75</td>
<td>-</td>
<td>VFP_SPEC</td>
<td>Instruction speculatively executed, Float Point</td>
</tr>
<tr>
<td>Event number</td>
<td>PMU event bus (to trace)</td>
<td>Event mnemonic</td>
<td>Event description</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------------</td>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>0x77</td>
<td>-</td>
<td>CRYPTO_SPEC</td>
<td>Operation speculatively executed, Cryptographic instruction. This event counts retired Cryptographic instructions.</td>
</tr>
<tr>
<td>0x78</td>
<td>-</td>
<td>BR_IMMED_SPEC</td>
<td>Branch speculatively executed, immediate branch. This event duplicates BR_IMMED_RETIRED.</td>
</tr>
<tr>
<td>0x79</td>
<td>-</td>
<td>BR_RETURN_SPEC</td>
<td>Branch speculatively executed, procedure return. This event duplicates BR_RETURN_RETIRED.</td>
</tr>
<tr>
<td>0x7A</td>
<td>-</td>
<td>BR_INDIRECT_SPEC</td>
<td>Branch speculatively executed, indirect branch. This event counts retired indirect branch instructions.</td>
</tr>
<tr>
<td>0x7C</td>
<td>-</td>
<td>ISB_SPEC</td>
<td>ISB speculatively executed</td>
</tr>
<tr>
<td>0x86</td>
<td>-</td>
<td>EXC_IRQ</td>
<td>Exception taken, IRQ.</td>
</tr>
<tr>
<td>0x87</td>
<td>-</td>
<td>EXC_FIQ</td>
<td>Exception taken, FIQ.</td>
</tr>
<tr>
<td>0xA0</td>
<td>-</td>
<td>L3D_CACHE_RD</td>
<td>Attributable Level 3 unified cache access, read. This event counts for any cacheable read transaction returning data from the SCU. If either the core is configured without a per-core L2 or the cluster is configured without an L3 cache, this event is not implemented.</td>
</tr>
<tr>
<td>0xA2</td>
<td>-</td>
<td>L3D_CACHE_REFILL_RD</td>
<td>Attributable Level 3 unified cache refill, read. This event duplicates L3D_CACHE_REFILL. If either the core is configured without a per-core L2 or the cluster is configured without an L3 cache, this event is not implemented.</td>
</tr>
<tr>
<td>0xC0</td>
<td>-</td>
<td>STB_STALL</td>
<td>Merge in the store buffer.</td>
</tr>
<tr>
<td>0xC1</td>
<td>-</td>
<td>BIU_EXT_MEM_REQ</td>
<td>External memory request.</td>
</tr>
<tr>
<td>0xC2</td>
<td>-</td>
<td>BIU_EXT_MEM_REQ_NC</td>
<td>External memory request to non-cacheable memory.</td>
</tr>
<tr>
<td>0xC3</td>
<td>-</td>
<td>L1D_PREF_LINE_FILL</td>
<td>Level 1 data cache refill started due to prefetch. Counts any linefills from the prefetcher which cause an allocation into the L1 D-cache.</td>
</tr>
<tr>
<td>0xC4</td>
<td>-</td>
<td>L2D_PREF_LINE_FILL</td>
<td>Level 2 cache refill due to prefetch.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• If the core is configured with a per-core L2 cache: This event does not count.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• If the core is configured without a per-core L2 cache: This event counts the cluster cache event, as defined by L3_PREF_LINE_FILL.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• If there is neither a per-core cache nor a cluster cache configured, this event is not implemented</td>
</tr>
<tr>
<td>Event number</td>
<td>PMU event bus (to trace)</td>
<td>Event mnemonic</td>
<td>Event description</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------</td>
<td>----------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>0xC5</td>
<td></td>
<td>L3_PREF_LINE_FILL</td>
<td>Level 3 cache refill due to prefetch. This event counts any linefills from the hardware prefetcher which cause an allocation into the L3 cache. <strong>Note</strong>: It might not be possible to distinguish between both hardware and software prefetches and also which prefetches cause an allocation. If so, only hardware prefetches should be counted, regardless of whether they allocate. If either the core is configured without a per-core L2 or the cluster is configured without an L3 cache, this event is not implemented.</td>
</tr>
<tr>
<td>0xC6</td>
<td></td>
<td>L1D_WS_MODE_ENTER</td>
<td>L1D entering write stream mode.</td>
</tr>
<tr>
<td>0xC7</td>
<td></td>
<td>L1D_WS_MODE</td>
<td>L1D is in write stream mode.</td>
</tr>
<tr>
<td>0xC8</td>
<td></td>
<td>L2D_WS_MODE</td>
<td>Level 2 cache write streaming mode. This event counts for each cycle where the core is in write-streaming mode and not allocating writes into the L2 cache.</td>
</tr>
<tr>
<td>0xC9</td>
<td></td>
<td>L3D_WS_MODE</td>
<td>Level 3 cache write streaming mode. This event counts for each cycle where the core is in write-streaming mode and not allocating writes into the L3 cache.</td>
</tr>
<tr>
<td>0xCA</td>
<td></td>
<td>TLB_L2TLB_LLWALK_ACCESS</td>
<td>Level 2 TLB last-level walk cache access. This event does not count if the MMU is disabled.</td>
</tr>
<tr>
<td>0xCB</td>
<td></td>
<td>TLB_L2TLB_LLWALK_REFILL</td>
<td>Level 2 TLB last-level walk cache refill. This event does not count if the MMU is disabled.</td>
</tr>
<tr>
<td>0xCC</td>
<td></td>
<td>TLB_L2TLB_L2WALK_ACCESS</td>
<td>Level 2 TLB level-2 walk cache access. This event counts accesses to the level-2 walk cache where the last-level walk cache has missed. The event only counts when the translation regime of the pagewalk uses level 2 descriptors. This event does not count if the MMU is disabled.</td>
</tr>
<tr>
<td>0xCD</td>
<td></td>
<td>TLB_L2TLB_L2WALK_REFILL</td>
<td>Level 2 TLB level-2 walk cache refill. This event does not count if the MMU is disabled.</td>
</tr>
<tr>
<td>0xCE</td>
<td></td>
<td>TLB_L2TLB_S2_ACCESS</td>
<td>Level 2 TLB IPA cache access. This event counts on each access to the IPA cache. • If a single pagewalk needs to make multiple accesses to the IPA cache, each access is counted. • If stage 2 translation is disabled, this event does not count.</td>
</tr>
<tr>
<td>Event number</td>
<td>PMU event bus (to trace)</td>
<td>Event mnemonic</td>
<td>Event description</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------</td>
<td>----------------</td>
<td>-------------------</td>
</tr>
</tbody>
</table>
| 0xCF         |                          | TLB_L2TLB_S2_REFILL | Level 2 TLB IPA cache refill. This event counts on each refill of the IPA cache.  
• If a single pagewalk needs to make multiple accesses to the IPA cache, each access which causes a refill is counted.  
• If stage 2 translation is disabled, this event does not count. |
| 0xD0         |                          | IFU_IC_MISS_WAIT   | I-Cache miss on an access from the prefetch block. |
| 0xD1         |                          | IFU_IUTLB_MISS_WAIT | Counts the cycles spent on a request for Level 2 TLB lookup after a Level 11 ITLB miss. |
| 0xD2         |                          | IFU_MICRO_COND_MISPRED | Micro-predictor conditional/direction mispredict, with respect to if3/if4 predictor. |
| 0xD3         |                          | IFU_MICRO_CADDR_MISPRED | Micro-predictor address mispredict, with respect to if3/if4 predictor. |
| 0xD4         |                          | IFU_MICRO_HIT      | Micro-predictor hit with immediate redirect. |
| 0xD6         |                          | IFU_MICRO_NEG_HIT  | Micro-predictor negative cache hit. |
| 0xD7         |                          | IFU_MICRO_CORRECTION | Micro-predictor correction. |
| 0xD8         |                          | IFU_MICRO_NO_INSTR | A 2nd instruction could have been pushed but was not because it was non-sequential. |
| 0xD9         |                          | IFU_MICRO_NO_PRED  | Micro-predictor miss. |
| 0xDA         |                          | IFU_FLUSHED_TLB_MISS  | Thread flushed due to TLB miss. |
| 0xDB         |                          | IFU_FLUSHED_EXCL_TLB_MISS  | Thread flushed due to reasons other than TLB miss. |
| 0xDC         |                          | IFU_ALL_THRDS_RDY   | This thread and the other thread both ready for scheduling in if0. |
| 0xDD         |                          | IFU_WIN_ARB_OTHER_RDY | This thread was arbitrated when the other thread was also ready for scheduling. |
| 0xDE         |                          | IFU_WIN_ARB_OTHER_ACT | This thread was arbitrated when the other thread was also active, but not necessarily ready. For example, waiting for I-Cache or TLB. |
| 0xDF         |                          | IFU_NOT_RDY_FOR_ARB | This thread was not arbitrated because it was not ready for scheduling. For example, due to a cache miss or TLB miss. |
| 0xE0         |                          | IFU_GOTO_IDLE       | The thread moved from an active state to an inactive state (long-term sleep state, causing deallocation of some resources). |
| 0xE1         |                          | IFU_IC_LOOKUP_UNDER_MISS  | I-Cache lookup under miss from other thread. |
| 0xE2         |                          | IFU_IC_MISS_UNDER_MISS | I-Cache miss under miss from other thread. |
| 0xE3         |                          | IFU_INSTR_PUSHED    | This thread pushed an instruction into the IQ. |
| 0xE4         |                          | IFU_IC_LF_SP        | I-Cache Speculative line fill. |
| 0xE8         |                          | DPU_BR_COND_RETIRED | Instruction retired, conditional branch. |
| 0xE9         |                          | DPU_BR_IND_MIS      | Instruction retired, indirect branch, mispredicted. |
Table C2-3 PMU events (continued)

<table>
<thead>
<tr>
<th>Event number</th>
<th>PMU event bus (to trace)</th>
<th>Event mnemonic</th>
<th>Event description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xEA</td>
<td>-</td>
<td>DPU_BR_COND_MIS</td>
<td>Instruction retired, conditional branch, mispredicted.</td>
</tr>
<tr>
<td>0xEB</td>
<td>-</td>
<td>DPU_MEM_ERR_IFU</td>
<td>Memory error (any type) from IFU.</td>
</tr>
<tr>
<td>0xEC</td>
<td>-</td>
<td>DPU_MEM_ERR_DCU</td>
<td>Memory error (any type) from DCU.</td>
</tr>
<tr>
<td>0xED</td>
<td>-</td>
<td>DPU_MEM_ERR_TLB</td>
<td>Memory error (any type) from TLB.</td>
</tr>
<tr>
<td>0xF0</td>
<td>-</td>
<td>L2_L1D_CACHE_WB_UNATT</td>
<td>Unattributable Level 1 data cache write-back. This event occurs when a requestor outside the PE makes a coherency request that results in write-back.</td>
</tr>
<tr>
<td>0xF1</td>
<td>-</td>
<td>L2_L2D_CACHE_UNATT</td>
<td>Unattributable Level 2 data cache access. This event occurs when a requestor outside the PE makes a coherency request that results in level 2 data cache access.</td>
</tr>
<tr>
<td>0xF2</td>
<td>-</td>
<td>L2_L2D_CACHE_RD_UNATT</td>
<td>Unattributable Level 2 data cache access, read. This event occurs when a requestor outside the PE makes a coherency request that results in level 2 data cache read access.</td>
</tr>
<tr>
<td>0xF3</td>
<td>-</td>
<td>L2_L3D_CACHE_UNATT</td>
<td>Unattributable Level 3 data cache access. This event occurs when a requestor outside the PE makes a coherency request that results in level 3 data cache read access.</td>
</tr>
<tr>
<td>0xF4</td>
<td>-</td>
<td>L2_L3D_CACHE_RD_UNATT</td>
<td>Unattributable Level 3 data cache access, read. This event occurs when a requestor outside the PE makes a coherency request that results in level 3 data cache read access.</td>
</tr>
<tr>
<td>0xF5</td>
<td>-</td>
<td>L2_L3D_CACHE_ALLOC_UNATT</td>
<td>Unattributable Level 3 data or unified cache allocation without refill. This event occurs when a requestor outside the PE makes a coherency request that results in level 3 cache allocate without refill.</td>
</tr>
<tr>
<td>0xF6</td>
<td>-</td>
<td>L2_L3D_CACHE_REFILL_UNATT</td>
<td>Unattributable Level 3 data or unified cache refill. This event occurs when a requestor outside the PE makes a coherency request that results in level 3 cache refill.</td>
</tr>
<tr>
<td>0xF7</td>
<td>-</td>
<td>L2D_CACHE_STASH_DROPPED</td>
<td>Level 2 cache stash dropped. This event counts on each stash request received from the interconnect or ACP, that is targeting L2 and gets dropped due to lack of buffer space to hold the request.</td>
</tr>
</tbody>
</table>

L2 and L3 cache events (L2D_CACHE*, L3D_CACHE*)

The behavior of these events depends on the configuration of the core.

If the private L2 cache is present, the L2D_CACHE* events count the activity in the private L2 cache, and the L3D_CACHE* events count the activity in the DSU L3 cache (if present).

If the private L2 cache is not present but the DSU L3 cache is present, the L2D_CACHE* events count activity in the DSU L3 cache and the L3D_CACHE* events do not count. The L2D_CACHE_WB, L2D_CACHE_WR and L2D_CACHE_REFILL_WR events do not count in this configuration.

If neither the private L2 cache nor the DSU L3 cache are present, neither the L2D_CACHE* or L3D_CACHE* events will count.
**Last Level cache events (LL_CACHE_*)**

The behavior of these events depends on the configuration of the core and the value of the CPUECTRL_EL1.EXTLLC bit.

**If the EXTLLC bit is 0:**

These events count activity in the last level of data cache implemented in the core. This is the DSU L3 cache if it is present, else the private L2 cache if it is present, otherwise the L1 data cache.

**If the EXTLLC bit is 1:**

These events count activity in a last level cache outside the core (if present). These events may not count in all implementations.
C2.5 PMU interrupts

The Cortex-A65 core asserts the nPMUIRQ signal when the PMU generates an interrupt.

You can route this signal to an external interrupt controller for prioritization and masking. This is the only mechanism that signals this interrupt to the core.

This interrupt is also driven as a trigger input to the CTI. See the Arm® DynamIQ™ Shared Unit Technical Reference Manual for more information.
C2.6 Exporting PMU events

Some of the PMU events are exported to the ETM trace unit to be monitored.

Note

The PMU EVENT bus is not exported to external components. This is because the event bus cannot safely cross an asynchronous boundary when events can be generated on every cycle.
Chapter C3
Embedded Trace Macrocell

This chapter describes the *Embedded Trace Macrocell* (ETM) for the Cortex-A65 core.

It contains the following sections:
- C3.1 About the ETM on page C3-332.
- C3.2 ETM trace unit generation options and resources on page C3-333.
- C3.3 ETM trace unit functional description on page C3-335.
- C3.4 Resetting the ETM on page C3-336.
- C3.5 Programming and reading ETM trace unit registers on page C3-337.
- C3.6 ETM trace unit register interfaces on page C3-338.
- C3.7 Interaction with the PMU and Debug on page C3-339.
C3.1 About the ETM

This module performs real-time instruction flow tracing that complies with the ETM architecture, ETMv4.2. As a CoreSight component, it is part of the Arm real-time debug solution.
C3.2 ETM trace unit generation options and resources

The following table shows the trace generation options implemented in the Cortex-A65 ETM trace unit.

### Table C3-1 ETM trace unit generation options implemented

<table>
<thead>
<tr>
<th>Description</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instruction address size in bytes</td>
<td>8</td>
</tr>
<tr>
<td>Data address size in bytes</td>
<td>0</td>
</tr>
<tr>
<td>Data value size in bytes</td>
<td>0</td>
</tr>
<tr>
<td>Virtual Machine ID size in bytes</td>
<td>4</td>
</tr>
<tr>
<td>Context ID size in bytes</td>
<td>4</td>
</tr>
<tr>
<td>Support for conditional instruction tracing</td>
<td>Not implemented</td>
</tr>
<tr>
<td>Support for tracing of data</td>
<td>Not implemented</td>
</tr>
<tr>
<td>Support for tracing of load and store instructions as P0 elements</td>
<td>Not implemented</td>
</tr>
<tr>
<td>Support for cycle counting in the instruction trace</td>
<td>Implemented</td>
</tr>
<tr>
<td>Support for branch broadcast tracing</td>
<td>Implemented</td>
</tr>
<tr>
<td>Number of events supported in the trace</td>
<td>4</td>
</tr>
<tr>
<td>Return stack support</td>
<td>Implemented</td>
</tr>
<tr>
<td>Tracing of SError exception support</td>
<td>Implemented</td>
</tr>
<tr>
<td>Instruction trace cycle counting minimum threshold</td>
<td>4</td>
</tr>
<tr>
<td>Size of Trace ID</td>
<td>7 bits</td>
</tr>
<tr>
<td>Synchronization period support</td>
<td>Read-write</td>
</tr>
<tr>
<td>Global timestamp size</td>
<td>64 bits</td>
</tr>
<tr>
<td>Number of cores available for tracing</td>
<td>1</td>
</tr>
<tr>
<td>ATB trigger support</td>
<td>Implemented</td>
</tr>
<tr>
<td>Low power behavior override</td>
<td>Implemented</td>
</tr>
<tr>
<td>Stall control support</td>
<td>Implemented</td>
</tr>
<tr>
<td>Support for overflow avoidance</td>
<td>Not implemented</td>
</tr>
<tr>
<td>Support for using CONTEXTIDR_EL2 in VMID comparator</td>
<td>Implemented</td>
</tr>
</tbody>
</table>

The following table shows the resources implemented in the Cortex-A65 ETM trace unit.

### Table C3-2 ETM trace unit resources implemented

<table>
<thead>
<tr>
<th>Description</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of resource selection pairs implemented</td>
<td>8</td>
</tr>
<tr>
<td>Number of external input selectors implemented</td>
<td>4</td>
</tr>
<tr>
<td>Number of external inputs implemented</td>
<td>32, 4 CTI + 28 PMU</td>
</tr>
<tr>
<td>Number of counters implemented</td>
<td>2</td>
</tr>
</tbody>
</table>
Table C3-2  ETM trace unit resources implemented (continued)

<table>
<thead>
<tr>
<th>Description</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reduced function counter implemented</td>
<td>Not implemented</td>
</tr>
<tr>
<td>Number of sequencer states implemented</td>
<td>4</td>
</tr>
<tr>
<td>Number of Virtual Machine ID comparators implemented</td>
<td>1</td>
</tr>
<tr>
<td>Number of Context ID comparators implemented</td>
<td>1</td>
</tr>
<tr>
<td>Number of address comparator pairs implemented</td>
<td>4</td>
</tr>
<tr>
<td>Number of single-shot comparator controls</td>
<td>1</td>
</tr>
<tr>
<td>Number of core comparator inputs implemented</td>
<td>0</td>
</tr>
<tr>
<td>Data address comparisons implemented</td>
<td>Not implemented</td>
</tr>
<tr>
<td>Number of data value comparators implemented</td>
<td>0</td>
</tr>
</tbody>
</table>
C3.3 ETM trace unit functional description

This section describes the functionality of each ETM trace unit.

The following figure shows the main functional blocks of each ETM trace unit.

![ETM functional blocks diagram](image)

**Figure C3-1 ETM functional blocks**

- **Core interface**

  This block monitors the behavior of the core and generates P0 elements that are essentially executed branches and exceptions traced in program order.

- **Trace generation**

  The trace generation block generates various trace packets based on P0 elements.

- **Filtering and triggering resources**

  You can limit the amount of trace data generated by each ETM through the process of filtering. For example, generating trace only in a certain address range. More complicated logic analyzer style filtering options are also available.

  The ETM trace unit can also generate a trigger that is a signal to the trace capture device to stop capturing trace.

- **FIFO**

  The trace generated by each ETM trace unit is in a highly-compressed form.

  The FIFO enables trace bursts to be flattened out. When the FIFO becomes full, the FIFO signals an overflow. The trace generation logic does not generate any new trace until the FIFO is emptied. This causes a gap in the trace when viewed in the debugger.

- **Trace out**

  Trace from FIFO is output on the AMBA ATB interface.

- **ATB funnel**

  Combines the ATB interfaces from each ETM into one ATB interface.
C3.4 Resetting the ETM

The reset for each ETM trace unit is the same as a Cold reset for the core.

Each ETM trace unit is not reset when Warm reset is applied to the core so that tracing through Warm core reset is possible.

If each ETM trace unit is reset, tracing stops until each ETM trace unit is reprogrammed and re-enabled. However, if the core is reset using Warm reset, the last few instructions provided by the core before the reset might not be traced.

--- Note ---

The ETM trace units cannot be reset individually, they can only be reset together.
C3.5 Programming and reading ETM trace unit registers

You program and read the ETM trace unit registers using the Debug APB interface.

The core does not have to be in debug state when you program the ETM trace unit registers.

When you are programming the ETM trace unit registers, you must enable all the changes at the same time. Otherwise, if you program the counter, it might start to count based on incorrect events before the correct setup is in place for the trigger condition.

To disable the ETM trace unit, use the TRCPRGCTLR.EN bit.

![Diagram of ETM trace unit programming](image)

**Figure C3-2 Programming ETM trace unit registers**

*Related references*

*D6.60 TRCPRGCTLR, Programming Control Register on page D6-493*
C3.6 ETM trace unit register interfaces

The Cortex-A65 core supports only memory-mapped interface to trace registers.

See the Arm® Embedded Trace Macrocell Architecture Specification ETMv4 for information on the behaviors on register accesses for different trace unit states and the different access mechanisms.

Note

Each thread can only access its own ETM trace unit.

Related references

C1.5 External debug interface on page C1-308
C3.7 Interaction with the PMU and Debug

This section describes the interaction with the PMU and the effect of debug double lock on trace register access.

Interaction with the PMU

The Cortex-A65 core includes a PMU that enables events, such as cache misses and instructions executed, to be counted over a period of time.

The PMU and ETM trace unit function together.

Use of PMU events by the ETM trace unit

The PMU architectural events described in C2.4 PMU events on page C2-313 are available to the ETM trace unit through the extended input facility.

See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile for more information about PMU events.

Each ETM trace unit uses four extended external input selectors to access the PMU events. Each selector can independently select one of the PMU events, that are then active for the cycles where the relevant events occur. These selected events can then be accessed by any of the event registers within the ETM trace unit. The PMU event table describes the PMU events.

Attributability

If an event is attributable to a given thread, then that event is only sent to the ETM associated with that thread. If an event is not attributable to a given thread, then that event is sent to both ETMs.

Related references

Chapter C2 Performance Monitor Unit on page C2-309
Part D
Debug registers
Chapter D1
AArch64 debug registers

This chapter describes the debug registers in the AArch64 Execution state and shows examples of how to use them.

It contains the following sections:
• D1.1 AArch64 debug register summary on page D1-344.
• D1.2 DBGBCRn_EL1, Debug Breakpoint Control Registers, EL1 on page D1-346.
• D1.3 DBGCLAIMSET_EL1, Debug Claim Tag Set Register, EL1 on page D1-349.
• D1.4 DBGWCRn_EL1, Debug Watchpoint Control Registers, EL1 on page D1-350.
• D1.5 MDSCR_EL1, Monitor Debug System Control Register, EL1 on page D1-352.
D1.1 AArch64 debug register summary

This section summarizes the debug control registers that are accessible.

These registers, listed in the following table, are accessed by the MRS and MSR instructions in the order of Op0, CRn, Op1, CRm, Op2.

See D2.1 Memory-mapped debug register summary on page D2-356 for a complete list of registers accessible from the external debug interface. The 64-bit registers cover two addresses on the external memory interface. For those registers not described in this chapter, see the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Reset</th>
<th>Width</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSDTRRX_EL1</td>
<td>RW</td>
<td>0x00000000</td>
<td>32</td>
<td>Debug Data Transfer Register, Receive, External View</td>
</tr>
<tr>
<td>DBGVR0_EL1</td>
<td>RW</td>
<td>-</td>
<td>64</td>
<td>Debug Breakpoint Value Register 0</td>
</tr>
<tr>
<td>DBGBCR0_EL1</td>
<td>RW</td>
<td>UNK</td>
<td>32</td>
<td>D1.2 DBGBCRn_EL1, Debug Breakpoint Control Registers, EL1</td>
</tr>
<tr>
<td>DBGWVR0_EL1</td>
<td>RW</td>
<td>-</td>
<td>64</td>
<td>Debug Watchpoint Value Register 0</td>
</tr>
<tr>
<td>DBGWCR0_EL1</td>
<td>RW</td>
<td>UNK</td>
<td>32</td>
<td>D1.4 DBGWCRn_EL1, Debug Watchpoint Control Registers, EL1</td>
</tr>
<tr>
<td>DBGVR1_EL1</td>
<td>RW</td>
<td>-</td>
<td>64</td>
<td>Debug Breakpoint Value Register 1</td>
</tr>
<tr>
<td>DBGBCR1_EL1</td>
<td>RW</td>
<td>UNK</td>
<td>32</td>
<td>D1.2 DBGBCRn_EL1, Debug Breakpoint Control Registers, EL1</td>
</tr>
<tr>
<td>DBGWVR1_EL1</td>
<td>RW</td>
<td>-</td>
<td>64</td>
<td>Debug Watchpoint Value Register 1</td>
</tr>
<tr>
<td>DBGWCR1_EL1</td>
<td>RW</td>
<td>UNK</td>
<td>32</td>
<td>D1.4 DBGWCRn_EL1, Debug Watchpoint Control Registers, EL1</td>
</tr>
<tr>
<td>MDCCINT_EL1</td>
<td>RW</td>
<td>0x00000000</td>
<td>32</td>
<td>Monitor Debug Comms Channel Interrupt Enable Register</td>
</tr>
<tr>
<td>MSCR_EL1</td>
<td>RW</td>
<td>-</td>
<td>32</td>
<td>D1.5 MSCR_EL1, Monitor Debug System Control Register, EL1</td>
</tr>
<tr>
<td>DBGVR2_EL1</td>
<td>RW</td>
<td>-</td>
<td>64</td>
<td>Debug Breakpoint Value Register 2</td>
</tr>
<tr>
<td>DBGBCR2_EL1</td>
<td>RW</td>
<td>UNK</td>
<td>32</td>
<td>D1.2 DBGBCRn_EL1, Debug Breakpoint Control Registers, EL1</td>
</tr>
<tr>
<td>DBGWVR2_EL1</td>
<td>RW</td>
<td>-</td>
<td>64</td>
<td>Debug Watchpoint Value Register 2</td>
</tr>
<tr>
<td>DBGWCR2_EL1</td>
<td>RW</td>
<td>UNK</td>
<td>32</td>
<td>D1.4 DBGWCRn_EL1, Debug Watchpoint Control Registers, EL1</td>
</tr>
<tr>
<td>OSDTRTX_EL1</td>
<td>RW</td>
<td>-</td>
<td>32</td>
<td>Debug Data Transfer Register, Transmit, External View</td>
</tr>
<tr>
<td>DBGVR3_EL1</td>
<td>RW</td>
<td>-</td>
<td>64</td>
<td>Debug Breakpoint Value Register 3</td>
</tr>
<tr>
<td>DBGBCR3_EL1</td>
<td>RW</td>
<td>UNK</td>
<td>32</td>
<td>D1.2 DBGBCRn_EL1, Debug Breakpoint Control Registers, EL1</td>
</tr>
<tr>
<td>DBGWVR3_EL1</td>
<td>RW</td>
<td>-</td>
<td>64</td>
<td>Debug Watchpoint Value Register 3</td>
</tr>
<tr>
<td>DBGWCR3_EL1</td>
<td>RW</td>
<td>UNK</td>
<td>32</td>
<td>D1.4 DBGWCRn_EL1, Debug Watchpoint Control Registers, EL1</td>
</tr>
<tr>
<td>DBGVR4_EL1</td>
<td>RW</td>
<td>-</td>
<td>64</td>
<td>Debug Breakpoint Value Register 4</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Reset</td>
<td>Width</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>------</td>
<td>--------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DBGBCR4_EL1</td>
<td>RW</td>
<td>UNK</td>
<td>32</td>
<td>D1.2 DBGBCRn_EL1, Debug Breakpoint Control Registers, EL1 on page D1-346</td>
</tr>
<tr>
<td>DBGVR5_EL1</td>
<td>RW</td>
<td>-</td>
<td>64</td>
<td>Debug Breakpoint Value Register 5</td>
</tr>
<tr>
<td>DBGBCR5_EL1</td>
<td>RW</td>
<td>UNK</td>
<td>32</td>
<td>D1.2 DBGBCRn_EL1, Debug Breakpoint Control Registers, EL1 on page D1-346</td>
</tr>
<tr>
<td>OSECCR_EL1</td>
<td>RW</td>
<td>0x00000000</td>
<td>32</td>
<td>Debug OS Lock Exception Catch Register</td>
</tr>
<tr>
<td>MDCCSR_EL0</td>
<td>RO</td>
<td>0x00000000</td>
<td>32</td>
<td>Monitor Debug Comms Channel Status Register</td>
</tr>
<tr>
<td>DBGDTR_EL0</td>
<td>RW</td>
<td>0x00000000</td>
<td>64</td>
<td>Debug Data Transfer Register, half-duplex</td>
</tr>
<tr>
<td>DBGDTRTX_EL0</td>
<td>WO</td>
<td>0x00000000</td>
<td>32</td>
<td>Debug Data Transfer Register, Transmit, Internal View</td>
</tr>
<tr>
<td>DBGDTRRX_EL0</td>
<td>RO</td>
<td>0x00000000</td>
<td>32</td>
<td>Debug Data Transfer Register, Receive, Internal View</td>
</tr>
<tr>
<td>MDRAR_EL1</td>
<td>RO</td>
<td>-</td>
<td>64</td>
<td>Debug ROM Address Register. This register is reserved, RES0</td>
</tr>
<tr>
<td>OSLAR_EL1</td>
<td>WO</td>
<td>-</td>
<td>32</td>
<td>Debug OS Lock Access Register</td>
</tr>
<tr>
<td>OSLSR_EL1</td>
<td>RO</td>
<td>0x0000000A</td>
<td>32</td>
<td>Debug OS Lock Status Register</td>
</tr>
<tr>
<td>OSDLR_EL1</td>
<td>RW</td>
<td>0x00000000</td>
<td>32</td>
<td>Debug OS Double Lock Register</td>
</tr>
<tr>
<td>DBGPRCR_EL1</td>
<td>RW</td>
<td>-</td>
<td>32</td>
<td>Debug Power/Reset Control Register</td>
</tr>
<tr>
<td>DBGCLAIMSET_EL1</td>
<td>RW</td>
<td>0x000000FF</td>
<td>32</td>
<td>D1.3 DBGCLAIMSET_EL1, Debug Claim Tag Set Register, EL1 on page D1-349</td>
</tr>
<tr>
<td>DBGCLAIMCLR_EL1</td>
<td>RW</td>
<td>0x00000000</td>
<td>32</td>
<td>Debug Claim Tag Clear Register</td>
</tr>
<tr>
<td>DBGAUTHSTATUS_EL1</td>
<td>RO</td>
<td>0x000000AA</td>
<td>32</td>
<td>Debug Authentication Status Register</td>
</tr>
</tbody>
</table>
D1.2 DBGBCRn_EL1, Debug Breakpoint Control Registers, EL1

The DBGBCRn_EL1 holds control information for a breakpoint. Each DBGBVR_EL1 is associated with a DBGBCR_EL1 to form a Breakpoint Register Pair (BRP). DBGBVRn_EL1 is associated with DBGBCRn_EL1 to form BRPn. The range of n for DBGBCR_EL1 is 0 to 5.

Bit field descriptions

The DBGBCRn_EL1 registers are 32-bit registers.

<table>
<thead>
<tr>
<th>31</th>
<th>24</th>
<th>20</th>
<th>19</th>
<th>16</th>
<th>15</th>
<th>14</th>
<th>12</th>
<th>9</th>
<th>8</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

RES0, [31:24]

RES0  Reserved.

BT, [23:20]

Breakpoint Type. This field controls the behavior of Breakpoint debug event generation. This includes the meaning of the value held in the associated DBGBVRn_EL1, indicating whether it is an instruction address match or mismatch, or a Context match. It also controls whether the breakpoint is linked to another breakpoint. The possible values are:

- 0b0000  Unlinked instruction address match.
- 0b0001  Linked instruction address match.
- 0b0010  Unlinked Context ID match.
- 0b0011  Linked Context ID match.
- 0b0100  Unlinked instruction address mismatch.
- 0b0101  Linked instruction address mismatch.
- 0b0110  Unlinked CONTEXTIDR_EL1 match.
- 0b0111  Linked CONTEXTIDR_EL1 match.
- 0b1000  Unlinked VMID match.
- 0b1001  Linked VMID match.
- 0b1010  Unlinked VMID + Context ID match.
- 0b1011  Linked VMID + Context ID match.
- 0b1100  Unlinked CONTEXTIDR_EL2 match.
- 0b1101  Linked CONTEXTIDR_EL2 match.
- 0b1110  Unlinked Full Context ID match.
- 0b1111  Linked Full Context ID match.

The field break down is:
- BT[3:1]: Base type. If the breakpoint is not context-aware, these bits are RES0. Otherwise, the possible values are:
  - 0b000  Match address. DBGBVRn_EL1 is the address of an instruction.
  - 0b001  Match context ID. DBGBVRn_EL1[31:0] is a context ID.
0b010  Address mismatch. Mismatch address. Behaves as type 0b000 if either:
— In an AArch64 translation regime.
— Halting debug-mode is enabled and halting is allowed.

Otherwise, DBGBVRn_EL1 is the address of an instruction to be stepped.

0b011  Match CONTEXTIDR_EL1. DBGBVRn_EL1[31:0] is a context ID.

0b100  Match VMID. DBGBVRn_EL1[47:32] is a VMID.

0b101  Match VMID and CONTEXTIDR_EL1. DBGBVRn_EL1[31:0] is a context ID, and DBGBVRn_EL1[47:32] is a VMID.

0b110  Match CONTEXTIDR_EL2. DBGBVRn_EL1[63:32] is a context ID.

0b111  Match CONTEXTIDR_EL1 and CONTEXTIDR_EL2. DBGBVRn_EL1[31:0] and DBGBVRn_EL1[63:32] are Context IDs.

• BT[0]: Enable linking.

LBN, [19:16]

Linked breakpoint number. For Linked address matching breakpoints, this specifies the index of the Context-matching breakpoint linked to.

SSC, [15:14]

Security State Control. Determines the Security states under which a Breakpoint debug event for breakpoint n is generated.

This field must be interpreted with the Higher Mode Control (HMC), and Privileged Mode Control (PMC), fields to determine the mode and security states that can be tested.

See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile for possible values of the HMC and PMC fields.

HMC, [13]

Hyp Mode Control bit. Determines the debug perspective for deciding when a breakpoint debug event for breakpoint n is generated.

This bit must be interpreted with the SSC and PMC fields to determine the mode and security states that can be tested.

See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile for possible values of the SSC and PMC fields.

RES0, [12:9]

RES0  Reserved.

BAS, [8:5]

Byte Address Select. Defines which half-words a regular breakpoint matches, regardless of the instruction set and execution state. A debugger must program this field as follows:

0x3  Match the T32 instruction at DBGBVRn_EL1.

0xC  Match the T32 instruction at DBGBVRn+2_EL1.

0xF  Match the A64 or A32 instruction at DBGBVRn_EL1, or context match.

All other values are reserved.


See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile for more information on how the BAS field is interpreted by hardware.

RES0, [4:3]
PMC, [2:1]

Privileged Mode Control. Determines the Exception level or levels that a breakpoint debug event for breakpoint \( n \) is generated.

This field must be interpreted with the SSC and HMC fields to determine the mode and security states that can be tested.

See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for possible values of the SSC and HMC fields.

Bits[2:1] have no effect for accesses made in Hyp mode.

E, [0]

Enable breakpoint. This bit enables the BRP:

0 \quad \text{BRP disabled.}

1 \quad \text{BRP enabled.}

A BRP never generates a breakpoint debug event when it is disabled.

The value of DBGBCR\(_n\) EL1.E is \textit{UNKNOWN} on reset. A debugger must ensure that DBGBCR\(_n\) EL1.E has a defined value before it enables debug.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*. 
### D1.3 DBGCLAIMSET_EL1, Debug Claim Tag Set Register, EL1

The DBGCLAIMSET_EL1 is used by software to set CLAIM bits to 1.

#### Bit field descriptions

The DBGCLAIMSET_EL1 is a 32-bit register.

![Figure D1-2 DBGCLAIMSET_EL1 bit assignments](image)

**RES0**, [31:8]

RES0  Reserved.

**CLAIM**, [7:0]

Claim set bits.

Writing a 1 to one of these bits sets the corresponding CLAIM bit to 1. This is an indirect write to the CLAIM bits.

A single write operation can set multiple bits to 1. Writing 0 to one of these bits has no effect.

Bit fields and details not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.
D1.4 DBGWCRn_EL1, Debug Watchpoint Control Registers, EL1

The DBGWCRn_EL1 holds control information for a watchpoint. Each DBGWCR_EL1 is associated with a DBGWVR_EL1 to form a Watchpoint Register Pair (WRP). DBGWCRn_EL1 is associated with DBGWVRn_EL1 to form WRPn. The range of n for DBGBCRn_EL1 is 0 to 3.

Bit field descriptions

The DBGWCRn_EL1 registers are 32-bit registers.

```
<table>
<thead>
<tr>
<th>Bit Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-29</td>
<td>MASK</td>
</tr>
<tr>
<td>28-24</td>
<td>LBN</td>
</tr>
<tr>
<td>23-21</td>
<td>SSC</td>
</tr>
<tr>
<td>20</td>
<td>BAS</td>
</tr>
<tr>
<td>19-16</td>
<td>LSC</td>
</tr>
<tr>
<td>15-14</td>
<td>PAC</td>
</tr>
<tr>
<td>13</td>
<td>HMC</td>
</tr>
<tr>
<td>12</td>
<td>WT</td>
</tr>
<tr>
<td>5-0</td>
<td>RES0</td>
</tr>
</tbody>
</table>
```

**Figure D1-3   DBGWCRn_EL1 bit assignments**

RES0, [31:29]

RES0          Reserved.

MASK, [28:24]

Address mask. Only objects up to 2GB can be watched using a single mask.

- 0b00000: No mask.
- 0b00001: Reserved.
- 0b00010: Reserved.

Other values mask the corresponding number of address bits, from 0b00011 masking 3 address bits (0x00000007 mask for address) to 0b11111 masking 31 address bits (0x7FFFFFFF mask for address).

RES0, [23:21]

RES0         Reserved.

WT, [20]

Watchpoint type. Possible values are:

- 0b0: Unlinked data address match.
- 0b1: Linked data address match.

On Cold reset, the field reset value is architecturally **UNKNOWN**.

LBN, [19:16]

Linked breakpoint number. For Linked data address watchpoints, this specifies the index of the Context-matching breakpoint linked to.

On Cold reset, the field reset value is architecturally **UNKNOWN**.

SSC, [15:14]

Security state control. Determines the Security states under which a watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the HMC and PAC fields.

On Cold reset, the field reset value is architecturally **UNKNOWN**.
HMC, [13]

Higher mode control. Determines the debug perspective for deciding when a watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the SSC and PAC fields.

On Cold reset, the field reset value is architecturally UNKNOWN.

BAS, [12:5]

Byte address select. Each bit of this field selects whether a byte from within the word or double-word addressed by DBGWVRn_EL1 is being watched. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile for more information.

LSC, [4:3]

Load/store access control. This field enables watchpoint matching on the type of access being made. The possible values are:

0b01 Match instructions that load from a watchpoint address.
0b10 Match instructions that store to a watchpoint address.
0b11 Match instructions that load from or store to a watchpoint address.

All other values are reserved, but must behave as if the watchpoint is disabled. Software must not rely on this property because the behavior of reserved values might change in a future revision of the architecture.

IGNORED if E is 0.

On Cold reset, the field reset value is architecturally UNKNOWN.

PAC, [2:1]

Privilege of access control. Determines the Exception level or levels at which a watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the SSC and HMC fields.

On Cold reset, the field reset value is architecturally UNKNOWN.

E, [0]

Enable watchpoint n. Possible values are:

0b0 Watchpoint disabled.
0b1 Watchpoint enabled.

On Cold reset, the field reset value is architecturally UNKNOWN.

Bit fields and details not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.
D1.5  MDSCR_EL1, Monitor Debug System Control Register, EL1

The MDSCR_EL1 main control register for the debug implementation.

Bit field descriptions

MDSCR_EL1 is a 32-bit register, and is part of the Debug registers functional group.

Figure D1-4  MDSCR_EL1 bit assignments

RES0, [31]

RES0 Reserved.

RXfull, [30]

Used for save/restore of EDSCR.RXfull

- When OSLSR_EL1.OSLK == 0 (the OS lock is unlocked), this bit is RO, and software must treat it as UNK/SBZP.
- When OSLSR_EL1.OSLK == 1 (the OS lock is locked), this bit is RW.

TXfull, [29]

Used for save/restore of EDSCR.TXfull

- When OSLSR_EL1.OSLK == 0 (the OS lock is unlocked), this bit is RO, and software must treat it as UNK/SBZP.
- When OSLSR_EL1.OSLK == 1 (the OS lock is locked), this bit is RW.

RES0, [28]

RES0 Reserved.

RXO, [27]

Used for save/restore of EDSCR.RXO.

- When OSLSR_EL1.OSLK == 0 (the OS lock is unlocked), this bit is RO. Software must treat it as UNKNOWN and use an SBZP policy for writes.
- When OSLSR_EL1.OSLK == 1 (the OS lock is locked), this bit is RW.

TXU, [26]

Used for save/restore of EDSCR.TXU.

- When OSLSR_EL1.OSLK == 0 (the OS lock is unlocked), this bit is RO. Software must treat it as UNKNOWN and use an SBZP policy for writes.
- When OSLSR_EL1.OSLK == 1 (the OS lock is locked), this bit is RW.

RES0, [25:24]

RES0 Reserved.
INTdis, [23:22]
Used for save/restore of EDSCR.INTdis.
- When OSLSR_EL1.OSLK == 0 (the OS lock is unlocked), this bit is RO. Software must treat it as \textit{UNKNOWN} and use an SBZP policy for writes.
- When OSLSR_EL1.OSLK == 1 (the OS lock is locked), this bit is RW.

TDA, [21]
Used for save/restore of EDSCR.TDA.
- When OSLSR_EL1.OSLK == 0 (the OS lock is unlocked), this bit is RO. Software must treat it as \textit{UNKNOWN} and use an SBZP policy for writes.
- When OSLSR_EL1.OSLK == 1 (the OS lock is locked), this bit is RW.

RES0, [20:19]
RES0 Reserved.

RAZ/WI, [18:16]
Reserved, RAZ/WI. Hardware must implement this as RAZ/WI. Software must not rely on this property as the behavior of reserved values might change in a future revision of the architecture.

MDE, [15]
Monitor debug events. Enable Breakpoint, Watchpoint, and Vector catch debug exceptions.
0 Breakpoint, Watchpoint, and Vector catch debug exceptions disabled.
1 Breakpoint, Watchpoint, and Vector catch debug exceptions enabled.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally \textit{UNKNOWN} on Warm reset.

HDE, [14]
Used for save/restore of EDSCR.HDE.
- When OSLSR_EL1.OSLK == 0 (the OS lock is unlocked), this bit is RO. Software must treat it as \textit{UNKNOWN} and use an SBZP policy for writes.
- When OSLSR_EL1.OSLK == 1 (the OS lock is locked), this bit is RW.

KDE, [13]
Local (kernel) debug enable. Enable Software debug events within EL\textsubscript{D}. Permitted values are:
0 Software debug events, other than Software breakpoint instructions, disabled within EL\textsubscript{D}.
1 Software debug events enabled within EL\textsubscript{D}.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally \textit{UNKNOWN} on Warm reset.

TDCC, [12]
Traps EL0 accesses to the DCC registers to EL1. Permitted values are:
0 This control does not cause any instructions to be trapped.
1 EL0 accesses to the MDCCSR_EL0, DBGDTR_EL0, DBGDTRTX_EL0, and DBGDTRRX_EL0 registers are trapped to EL1. Trap of AArch64 accesses to DBGDTR_EL0, DBGDTRRX_EL0, and DBGDTRTX_EL0 registers are ignored in Debug state.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally \textit{UNKNOWN} on Warm reset.

RES0, [11:7]
RES0 Reserved.
ERR, [6]
Used for save/restore of EDSCR.ERR.
• When OSLSR_EL1.OSLK == 0 (the OS lock is unlocked), this bit is RO. Software must treat it as \textit{UNKNOWN} and use an SBZP policy for writes.
• When OSLSR_EL1.OSLK == 1 (the OS lock is locked), this bit is RW.

RES0, [5:1]
\begin{itemize}
\item \texttt{RES0} \hspace{1cm} Reserved.
\end{itemize}

SS, [0]
Software step control bit. If EL_D is used, enables Software step. Permitted values are:
\begin{itemize}
\item 0 \hspace{1cm} Software step is disabled.
\item 1 \hspace{1cm} Software step is enabled.
\end{itemize}

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally \textit{UNKNOWN} on Warm reset.

\textbf{Configurations}
This register is in the Warm reset domain. Some or all RW fields of this register have defined reset values. On a Warm or Cold reset RW fields in this register reset to architecturally \textit{UNKNOWN} values.

Bit fields and details not provided in this description are architecturally defined. See the \textit{Arm* Architecture Reference Manual Armv8, for Armv8-A architecture profile}. 
Chapter D2
Memory-mapped debug registers

This chapter describes the memory-mapped debug registers and shows examples of how to use them.

It contains the following sections:

- D2.1 Memory-mapped debug register summary on page D2-356.
- D2.2 EDCIDR0, External Debug Component Identification Register 0 on page D2-360.
- D2.3 EDCIDR1, External Debug Component Identification Register 1 on page D2-361.
- D2.4 EDCIDR2, External Debug Component Identification Register 2 on page D2-362.
- D2.5 EDCIDR3, External Debug Component Identification Register 3 on page D2-363.
- D2.6 EDDEVID, External Debug Device ID Register 0 on page D2-364.
- D2.7 EDDEVID1, External Debug Device ID Register 1 on page D2-365.
- D2.8 EDDFR, External Debug Feature Register on page D2-366.
- D2.9 EDITCTRL, External Debug Integration Mode Control Register on page D2-368.
- D2.10 EDPFR, External Debug Processor Feature Register on page D2-369.
- D2.11 EDPIDR0, External Debug Peripheral Identification Register 0 on page D2-371.
- D2.12 EDPIDR1, External Debug Peripheral Identification Register 1 on page D2-372.
- D2.13 EDPIDR2, External Debug Peripheral Identification Register 2 on page D2-373.
- D2.14 EDPIDR3, External Debug Peripheral Identification Register 3 on page D2-374.
- D2.15 EDPIDR4, External Debug Peripheral Identification Register 4 on page D2-375.
- D2.16 EDPIDRn, External Debug Peripheral Identification Registers 5-7 on page D2-376.
- D2.17 EDRCR, External Debug Reserve Control Register on page D2-377.
D2.1 Memory-mapped debug register summary

The following table shows the offset address for the registers that are accessible from the external debug interface.

For those registers not described in this chapter, see the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Width</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000-0x1C</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x020</td>
<td>EDESR</td>
<td>RW</td>
<td>32</td>
<td>External Debug Event Status Register</td>
</tr>
<tr>
<td>0x024</td>
<td>EDECR</td>
<td>RW</td>
<td>32</td>
<td>External Debug Execution Control Register</td>
</tr>
<tr>
<td>0x028-0x2C</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x030</td>
<td>EDWAR[31:0]</td>
<td>RO</td>
<td>64</td>
<td>External Debug Watchpoint Address Register</td>
</tr>
<tr>
<td>0x034</td>
<td>EDWAR[63:32]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x038-0x07C</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x080</td>
<td>DBGDTRRX_EL0</td>
<td>RW</td>
<td>32</td>
<td>Debug Data Transfer Register, Receive</td>
</tr>
<tr>
<td>0x084</td>
<td>EDITR</td>
<td>WO</td>
<td>32</td>
<td>External Debug Instruction Transfer Register</td>
</tr>
<tr>
<td>0x088</td>
<td>EDSR</td>
<td>RW</td>
<td>32</td>
<td>External Debug Status and Control Register</td>
</tr>
<tr>
<td>0x08C</td>
<td>DBGDTRTX_EL0</td>
<td>WO</td>
<td>32</td>
<td>Debug Data Transfer Register, Transmit</td>
</tr>
<tr>
<td>0x090</td>
<td>EDRCR</td>
<td>WO</td>
<td>32</td>
<td>D2.17 EDRCR, External Debug Reserve Control Register on page D2-377</td>
</tr>
<tr>
<td>0x094</td>
<td>-</td>
<td>RW</td>
<td>32</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x098</td>
<td>EDECCR</td>
<td>RW</td>
<td>32</td>
<td>External Debug Exception Catch Control Register</td>
</tr>
<tr>
<td>0x09C</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0A0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0A4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0A8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0AC</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0B0-0x2FC</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x300</td>
<td>OSLAR_EL1</td>
<td>WO</td>
<td>32</td>
<td>OS Lock Access Register</td>
</tr>
<tr>
<td>0x304-0x30C</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x310</td>
<td>EDPRCR</td>
<td>RW</td>
<td>32</td>
<td>External Debug Power/Reset Control Register</td>
</tr>
<tr>
<td>0x314</td>
<td>EDPRSR</td>
<td>RO</td>
<td>32</td>
<td>External Debug Processor Status Register</td>
</tr>
<tr>
<td>0x318-0x3FC</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x400</td>
<td>DBGBVR0_EL1[31:0]</td>
<td>RW</td>
<td>64</td>
<td>Debug Breakpoint Value Register 0</td>
</tr>
<tr>
<td>0x404</td>
<td>DBGBVR0_EL1[63:32]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Offset</td>
<td>Name</td>
<td>Type</td>
<td>Width</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------</td>
<td>------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>0x408</td>
<td>DBGBCR0_EL1</td>
<td>RW</td>
<td>32</td>
<td>D1.2 DBGBCRn_EL1, Debug Breakpoint Control Registers, EL1 on page D1-346</td>
</tr>
<tr>
<td>0x40C</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x410</td>
<td>DBGVR1_EL1[31:0]</td>
<td>RW</td>
<td>64</td>
<td>Debug Breakpoint Value Register 1</td>
</tr>
<tr>
<td>0x414</td>
<td>DBGVR1_EL1[63:32]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x418</td>
<td>DBGCR1_EL1</td>
<td>RW</td>
<td>32</td>
<td>D1.2 DBGBCRn_EL1, Debug Breakpoint Control Registers, EL1 on page D1-346</td>
</tr>
<tr>
<td>0x41C</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x420</td>
<td>DBGVR2_EL1[31:0]</td>
<td>RW</td>
<td>64</td>
<td>Debug Breakpoint Value Register 2</td>
</tr>
<tr>
<td>0x424</td>
<td>DBGVR2_EL1[63:32]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x428</td>
<td>DBGCR2_EL1</td>
<td>RW</td>
<td>32</td>
<td>D1.2 DBGBCRn_EL1, Debug Breakpoint Control Registers, EL1 on page D1-346</td>
</tr>
<tr>
<td>0x42C</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x430</td>
<td>DBGVR3_EL1[31:0]</td>
<td>RW</td>
<td>64</td>
<td>Debug Breakpoint Value Register 3</td>
</tr>
<tr>
<td>0x434</td>
<td>DBGVR3_EL1[63:32]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x438</td>
<td>DBGCR3_EL1</td>
<td>RW</td>
<td>32</td>
<td>D1.2 DBGBCRn_EL1, Debug Breakpoint Control Registers, EL1 on page D1-346</td>
</tr>
<tr>
<td>0x43C</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x440</td>
<td>DBGVR4_EL1[31:0]</td>
<td>RW</td>
<td>64</td>
<td>Debug Breakpoint Value Register 4</td>
</tr>
<tr>
<td>0x444</td>
<td>DBGVR4_EL1[63:32]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x448</td>
<td>DBGCR4_EL1</td>
<td>RW</td>
<td>32</td>
<td>D1.2 DBGBCRn_EL1, Debug Breakpoint Control Registers, EL1 on page D1-346</td>
</tr>
<tr>
<td>0x44C</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x450</td>
<td>DBGVR5_EL1[31:0]</td>
<td>RW</td>
<td>64</td>
<td>Debug Breakpoint Value Register 5</td>
</tr>
<tr>
<td>0x454</td>
<td>DBGVR5_EL1[63:32]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x458</td>
<td>DBGCR5_EL1</td>
<td>RW</td>
<td>32</td>
<td>D1.2 DBGBCRn_EL1, Debug Breakpoint Control Registers, EL1 on page D1-346</td>
</tr>
<tr>
<td>0x45C-0x7FC</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x800</td>
<td>DBGWVR0_EL1[31:0]</td>
<td>RW</td>
<td>64</td>
<td>Debug Watchpoint Value Register 0</td>
</tr>
<tr>
<td>0x804</td>
<td>DBGWVR0_EL1[63:32]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x808</td>
<td>DBGWCR0_EL1</td>
<td>RW</td>
<td>32</td>
<td>D1.4 DBGWCRn_EL1, Debug Watchpoint Control Registers, EL1 on page D1-350</td>
</tr>
<tr>
<td>0x80C</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x810</td>
<td>DBGWVR1_EL1[31:0]</td>
<td>RW</td>
<td>64</td>
<td>Debug Watchpoint Value Register 1</td>
</tr>
<tr>
<td>0x814</td>
<td>DBGWVR1_EL1[63:32]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x818</td>
<td>DBGWCR1_EL1</td>
<td>RW</td>
<td>32</td>
<td>D1.4 DBGWCRn_EL1, Debug Watchpoint Control Registers, EL1 on page D1-350</td>
</tr>
<tr>
<td>Offset</td>
<td>Name</td>
<td>Type</td>
<td>Width</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------------------------</td>
<td>-------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>0x81C</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x820</td>
<td>DBGWVR2_EL1[31:0]</td>
<td>RW</td>
<td>64</td>
<td>Debug Watchpoint Value Register 2</td>
</tr>
<tr>
<td>0x824</td>
<td>DBGWVR2_EL1[63:32]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x828</td>
<td>DBGWCR2_EL1</td>
<td>RW</td>
<td>32</td>
<td>*D1.4 DBGWCRn_EL1, Debug Watchpoint Control Registers, EL1 on page D1-350</td>
</tr>
<tr>
<td>0x82C</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x830</td>
<td>DBGWVR3_EL1[31:0]</td>
<td>RW</td>
<td>64</td>
<td>Debug Watchpoint Value Register 0</td>
</tr>
<tr>
<td>0x834</td>
<td>DBGWVR3_EL1[63:32]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x838</td>
<td>DBGWCR3_EL1</td>
<td>RW</td>
<td>32</td>
<td>*D1.4 DBGWCRn_EL1, Debug Watchpoint Control Registers, EL1 on page D1-350</td>
</tr>
<tr>
<td>0x83C-0xCFC</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0xD00</td>
<td>MIDR</td>
<td>RO</td>
<td>32</td>
<td>*B1.64 MIDL_EL1, Main ID Register, EL1 on page B1-217</td>
</tr>
<tr>
<td>0xD04-0xD1C</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0xD20</td>
<td>EDPFR[31:0]</td>
<td>RO</td>
<td>64</td>
<td>*B1.58 ID_AA64PFR0_EL1, AArch64 Processor Feature Register 0, EL1 on page B1-209</td>
</tr>
<tr>
<td>0xD24</td>
<td>EDPFR[63:32]</td>
<td>RO</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td>0xD28</td>
<td>EDDFR[31:0]</td>
<td>RO</td>
<td>64</td>
<td>*B1.58 ID_AA64PFR0_EL1, AArch64 Processor Feature Register 0, EL1 on page B1-209</td>
</tr>
<tr>
<td>0xD2C</td>
<td>EDDFR[63:32]</td>
<td>RO</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td>0xD60-0xEFC</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0xF00</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0xF04-0xF9C</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0xFA0</td>
<td>DBGCLAIMSET_EL1</td>
<td>RW</td>
<td>32</td>
<td>*D1.3 DBGCLAIMSET_EL1, Debug Claim Tag Set Register, EL1 on page D1-349</td>
</tr>
<tr>
<td>0xFA4</td>
<td>DBGCLAIMCLR_EL1</td>
<td>RW</td>
<td>32</td>
<td>Debug Claim Tag Clear Register</td>
</tr>
<tr>
<td>0xFA8</td>
<td>EDDEVAFF0</td>
<td>RO</td>
<td>32</td>
<td>External Debug Device Affinity Register 0</td>
</tr>
<tr>
<td>0xFAC</td>
<td>EDDEVAFF1</td>
<td>RO</td>
<td>32</td>
<td>External Debug Device Affinity Register 1</td>
</tr>
<tr>
<td>0xFB0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0xFB4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0xFB8</td>
<td>DBGAUTHSTATUS_EL1</td>
<td>RO</td>
<td>32</td>
<td>Debug Authentication Status Register</td>
</tr>
<tr>
<td>0xFBC</td>
<td>EDDEVARCH</td>
<td>RO</td>
<td>32</td>
<td>External Debug Device Architecture Register</td>
</tr>
<tr>
<td>0xFC0</td>
<td>EDDEVID2</td>
<td>RO</td>
<td>32</td>
<td>External Debug Device ID Register 2, RES0</td>
</tr>
<tr>
<td>0xFC4</td>
<td>EDDEVID1</td>
<td>RO</td>
<td>32</td>
<td>*D2.7 EDDEVID1, External Debug Device ID Register 1 on page D2-365</td>
</tr>
<tr>
<td>0xFC8</td>
<td>EDDEVID</td>
<td>RO</td>
<td>32</td>
<td>*D2.6 EDDEVID, External Debug Device ID Register 0 on page D2-364</td>
</tr>
<tr>
<td>0xFCC</td>
<td>EDDEVTYPE</td>
<td>RO</td>
<td>32</td>
<td>External Debug Device Type Register</td>
</tr>
<tr>
<td>0xFD0</td>
<td>EDPIDR4</td>
<td>RO</td>
<td>32</td>
<td>*D2.15 EDPIDR4, External Debug Peripheral Identification Register 4 on page D2-375</td>
</tr>
<tr>
<td>Offset</td>
<td>Name</td>
<td>Type</td>
<td>Width</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------</td>
<td>------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>0xFD4-0xFDC</td>
<td>EDPIDR5-7</td>
<td>RO</td>
<td>32</td>
<td>D2.16 EDPIDRn, External Debug Peripheral Identification Registers 5-7 on page D2-376</td>
</tr>
<tr>
<td>0xFE0</td>
<td>EDPIDR0</td>
<td>RO</td>
<td>32</td>
<td>D2.11 EDPIDR0, External Debug Peripheral Identification Register 0 on page D2-371</td>
</tr>
<tr>
<td>0xFE4</td>
<td>EDPIDR1</td>
<td>RO</td>
<td>32</td>
<td>D2.12 EDPIDR1, External Debug Peripheral Identification Register 1 on page D2-372</td>
</tr>
<tr>
<td>0xFE8</td>
<td>EDPIDR2</td>
<td>RO</td>
<td>32</td>
<td>D2.13 EDPIDR2, External Debug Peripheral Identification Register 2 on page D2-373</td>
</tr>
<tr>
<td>0xFEC</td>
<td>EDPIDR3</td>
<td>RO</td>
<td>32</td>
<td>D2.14 EDPIDR3, External Debug Peripheral Identification Register 3 on page D2-374</td>
</tr>
<tr>
<td>0xFF0</td>
<td>EDCIDR0</td>
<td>RO</td>
<td>32</td>
<td>D2.2 EDCIDR0, External Debug Component Identification Register 0 on page D2-360</td>
</tr>
<tr>
<td>0xFF4</td>
<td>EDCIDR1</td>
<td>RO</td>
<td>32</td>
<td>D2.3 EDCIDR1, External Debug Component Identification Register 1 on page D2-361</td>
</tr>
<tr>
<td>0xFF8</td>
<td>EDCIDR2</td>
<td>RO</td>
<td>32</td>
<td>D2.4 EDCIDR2, External Debug Component Identification Register 2 on page D2-362</td>
</tr>
<tr>
<td>0xFFC</td>
<td>EDCIDR3</td>
<td>RO</td>
<td>32</td>
<td>D2.5 EDCIDR3, External Debug Component Identification Register 3 on page D2-363</td>
</tr>
</tbody>
</table>
D2.2 EDCIDR0, External Debug Component Identification Register 0

The EDCIDR0 provides information to identify an external debug component.

**Bit field descriptions**

The EDCIDR0 is a 32-bit register.

![EDCIDR0 Bit Assignments](image)

**RES0, [31:8]**

RES0  Reserved.

**PRMBL_0, [7:0]**

0x0D  Preamble byte 0.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The EDCIDR0 can be accessed through the external debug interface, offset 0xFF0.
D2.3  **EDCIDR1, External Debug Component Identification Register 1**

The EDCIDR1 provides information to identify an external debug component.

**Bit field descriptions**

The EDCIDR1 is a 32-bit register.

![Figure D2-2  EDCIDR1 bit assignments](image)

RES0, [31:8]  
**RES0** Reserved.

CLASS, [7:4]  
0x9  
Debug component.

PRMBL_1, [3:0]  
0x0  
Preamble.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.*

The EDCIDR1 can be accessed through the external debug interface, offset 0xFF4.
D2.4 EDCIDR2, External Debug Component Identification Register 2

The EDCIDR2 provides information to identify an external debug component.

**Bit field descriptions**

The EDCIDR2 is a 32-bit register.

![Figure D2-3 EDCIDR2 bit assignments](image)

**RES0, [31:8]**

`RES0` Reserved.

**PRMBL_2, [7:0]**

`0x05` Preamble byte 2.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The EDCIDR2 can be accessed through the external debug interface, offset `0xFF8`.
D2.5 EDCIDR3, External Debug Component Identification Register 3

The EDCIDR3 provides information to identify an external debug component.

**Bit field descriptions**

The EDCIDR3 is a 32-bit register.

<table>
<thead>
<tr>
<th>Bit Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-0</td>
<td>RES0</td>
</tr>
<tr>
<td>7-0</td>
<td>PRMBL_3</td>
</tr>
</tbody>
</table>

RES0, [31:8]

RES0 Reserved.

PRMBL_3, [7:0]

0xB1 Preamble byte 3.

Bit fields and details not provided in this description are architecturally defined. See the *Arm*® *Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The EDCIDR3 can be accessed through the external debug interface, offset 0xFFC.

![Figure D2-4 EDCIDR3 bit assignments](image-url)
D2.6 EDDEVID, External Debug Device ID Register 0

The EDDEVID provides extra information for external debuggers about features of the debug implementation.

**Bit field descriptions**

The EDDEVID is a 32-bit register.

![EDDEVID bit assignments](image)

**RES0, [31:28]**

  - **RES0** Reserved.

**AuxRegs, [27:24]**

  Indicates support for Auxiliary registers:
  - 0x0 None supported.

**RES0, [23:0]**

  - **RES0** Reserved.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The EDDEVID can be accessed through the external debug interface, offset 0xFC8.
D2.7 EDDEVID1, External Debug Device ID Register 1

The EDDEVID1 provides extra information for external debuggers about features of the debug implementation.

**Bit field descriptions**

The EDDEVID1 is a 32-bit register.

![EDDEVID1 bit assignments](image)

RES0, [31:0]  
RES0 Reserved.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The EDDEVID1 can be accessed through the external debug interface, offset 0xFC4.
D2.8 **EDDFR, External Debug Feature Register**

The EDDFR provides top level information about the debug system in AArch64.

**Bit field descriptions**

The EDDFR is a 64-bit register.

![Figure D2-7 EDDFR bit assignments](image)

RES0, [63:32]
RES0 Reserved.

CTX_CMPs, [31:28]
Number of breakpoints that are context-aware, minus 1. These are the highest numbered breakpoints.

RES0, [27:24]
RES0 Reserved.

WRPs, [23:20]
Number of watchpoints, minus 1. The value of 0b0000 is reserved.

RES0, [19:16]
RES0 Reserved.

BRPs, [15:12]
Number of breakpoints, minus 1. The value of 0b0000 is reserved.

PMUVer, [11:8]
Performance Monitors extension version. Indicates whether system register interface to Performance Monitors extension is implemented. Defined values are:

- 0x0000 Performance Monitors extension system registers not implemented.
- 0x0001 Performance Monitors extension system registers implemented, PMUv3.
- 0x1111 IMPLEMENTATION DEFINED form of performance monitors supported, PMUv3 not supported.

All other values are reserved.

TraceVer [7:4]
Trace support. Indicates whether system register interface to a trace macrocell is implemented. Defined values are:

- 0x0000 Trace macrocell system registers not implemented.
- 0x0001 Trace macrocell system registers implemented.

All other values are reserved.
A value of 0x0000 only indicates that no system register interface to a trace macrocell is implemented. A trace macrocell might nevertheless be implemented without a system register interface.

**UNKNOWN, [3:0]**

UNKNOWN    Reserved.

Bit fields and details not provided in this description are architecturally defined. See the \textit{Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile}.

EDDFR[31:0] can be accessed through the external debug interface, offset 0x0D28.

EDDFR[63:32] can be accessed through the external debug interface, offset 0x0D2C.
D2.9 EDITCTRL, External Debug Integration Mode Control Register

The EDITCTRL enables the external debug to switch from its default mode into integration mode, where test software can control directly the inputs and outputs of the core, for integration testing or topology detection.

**Bit field descriptions**

The ESITCTRL is a 32-bit register.

![EDITCTRL Bit Assignments](image)

**Figure D2-8 EDITCTRL bit assignments**

- **[31:1]**
  - **RES0**: Reserved.
  - **IME**: [0]
    - Integration Mode Enable.
    - **RES0**: The device does not revert to an integration mode to enable integration testing or topology detection.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The EDITCTRL can be accessed through the external debug interface, offset 0xF00.
D2.10 EDPFR, External Debug Processor Feature Register

The EDPFR provides additional information about implemented PE features in AArch64.

**Bit field descriptions**

The EDPFR is a 64-bit register.

![Figure D2-9 EDPFR bit assignments](image)

**RES0, [63:28]**

RES0 Reserved.

**GIC, [27:24]**

System register GIC interface. Defined values are:

- 0x0 No System register interface to the GIC is supported.
- 0x1 System register interface to the GIC CPU interface is supported.

All other values are reserved.

**AdvSIMD, [23:20]**

Advanced SIMD. Defined values are:

- 0x0 Advanced SIMD is implemented.
- 0xF Advanced SIMD is not implemented.

All other values are reserved.

**FP, [19:16]**

Floating-point. Defined values are:

- 0x0 Floating-point is implemented.
- 0xF Floating-point is not implemented.

All other values are reserved.

**EL3 handling, [15:12]**

EL3 exception handling:

- 0x2 Instructions can be executed at EL3.

**EL2 handling, [11:8]**

EL2 exception handling:

- 0x2 Instructions can be executed at EL2.

**EL1 handling, [7:4]**

EL1 exception handling. The possible values are:
$	ext{x}2$  Instructions can be executed at EL1.

**EL0 handling, [3:0]**

EL0 exception handling. The possible values are:

$	ext{x}2$  Instructions can be executed at EL0.

Bit fields and details not provided in this description are architecturally defined. See the *Arm*® *Architecture Reference Manual Armv8, for Armv8-A architecture profile.*

The EDPFR[31:0] can be accessed through the external debug interface, offset $0x2d0$.

The EDPFR[63:32] can be accessed through the external debug interface, offset $0x2d4$. 


D2.11 EDPIDR0, External Debug Peripheral Identification Register 0

The EDPIDR0 provides information to identify an external debug component.

**Bit field descriptions**

The EDPIDR0 is a 32-bit register.

![EDPIDR0 bit assignments](image_url)

**RES0, [31:8]**

RES0  Reserved.

**Part_0, [7:0]**

Least significant byte of the debug part number.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.*

The EDPIDR0 can be accessed through the external debug interface, offset 0xFE0.
D2.12 EDPIDR1, External Debug Peripheral Identification Register 1

The EDPIDR1 provides information to identify an external debug component.

**Bit field descriptions**

The EDPIDR1 is a 32-bit register.

<table>
<thead>
<tr>
<th>Bit Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RES0</td>
<td>Reserved.</td>
</tr>
<tr>
<td>DES_0</td>
<td>Arm Limited. This is the least significant nibble of JEP106 ID code.</td>
</tr>
<tr>
<td>Part_1</td>
<td>Most significant nibble of the debug part number.</td>
</tr>
</tbody>
</table>

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The EDPIDR1 can be accessed through the external debug interface, offset 0xFE4.
D2.13  EDPIDR2, External Debug Peripheral Identification Register 2

The EDPIDR2 provides information to identify an external debug component.

**Bit field descriptions**

The EDPIDR2 is a 32-bit register.

![Figure D2-12  EDPIDR2 bit assignments](image)

RES0, [31:8]

RES0  Reserved.

Revision, [7:4]

2  r1p1.

JEDEC, [3]

0b1  RAO. Indicates a JEP106 identity code is used.

DES_1, [2:0]

0b01  Arm Limited. This is the most significant nibble of JEP106 ID code.

Bit fields and details not provided in this description are architecturally defined. See the *Arm*® *Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The EDPIDR2 can be accessed through the external debug interface, offset 0xFE8.
D2.14  **EDPIDR3, External Debug Peripheral Identification Register 3**

The EDPIDR3 provides information to identify an external debug component.

**Bit field descriptions**

The EDPIDR3 is a 32-bit register.

![Figure D2-13  EDPIDR3 bit assignments](image)

- **RES0, [31:8]**
  - **RES0**  Reserved.

- **REV AND, [7:4]**
  - 0x0  Part minor revision.

- **CMOD, [3:0]**
  - 0x0  Customer modified.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The EDPIDR3 can be accessed through the external debug interface, offset 0xFEC.
D2.15  **EDPIDR4, External Debug Peripheral Identification Register 4**

The EDPIDR4 provides information to identify an external debug component.

**Bit field descriptions**

The EDPIDR4 is a 32-bit register.

![Figure D2-14  EDPIDR4 bit assignments](image)

**RES0, [31:8]**

RES0  Reserved.

**SIZE, [7:4]**

0x0  Size of the component. \( \log_2 \) the number of 4KB pages from the start of the component to the end of the component ID registers.

**DES_2, [3:0]**

0x4  Arm Limited This is the least significant nibble JEP106 continuation code.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The EDPIDR4 can be accessed through the external debug interface, offset 0xFD0.
D2.16  EDPIDRn, External Debug Peripheral Identification Registers 5-7

No information is held in the Peripheral ID5, Peripheral ID6, and Peripheral ID7 Registers. They are reserved for future use and are RES0.
D2.17 EDRCR, External Debug Reserve Control Register

The EDRCR is part of the Debug registers functional group. This register is used to allow imprecise entry to Debug state and clear sticky bits in EDSCR.

Bit field descriptions

The EDRCR is a 32-bit register.

<table>
<thead>
<tr>
<th>Bit Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RES0, [31:5]</td>
<td>Reserved.</td>
</tr>
<tr>
<td>CBRRQ, [4]</td>
<td>Allow imprecise entry to Debug state. Allow imprecise entry to Debug state,</td>
</tr>
<tr>
<td></td>
<td>for example by canceling pending bus accesses.</td>
</tr>
<tr>
<td></td>
<td>Setting this bit to 1 allows a debugger to request imprecise entry to Debug</td>
</tr>
<tr>
<td></td>
<td>state. An External Debug Request debug event must be pending before the</td>
</tr>
<tr>
<td></td>
<td>debugger sets this bit to 1.</td>
</tr>
<tr>
<td>CSPA, [3]</td>
<td>Clear Sticky Pipeline Advance. This bit is used to clear the EDSCR.PipeAdv</td>
</tr>
<tr>
<td></td>
<td>bit to 0.</td>
</tr>
<tr>
<td>CSE, [2]</td>
<td>Clear Sticky Error. Used to clear the EDSCR cumulative error bits to 0.</td>
</tr>
<tr>
<td></td>
<td>The core is in Debug state, the EDSCR.IRO bit, to 0.</td>
</tr>
<tr>
<td>RES0, [1:0]</td>
<td>Reserved.</td>
</tr>
</tbody>
</table>

Bit fields and details not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.

The EDRCR can be accessed through the external debug interface, offset 0x090.
D2 Memory-mapped debug registers
D2.17 EDRCR, External Debug Reserve Control Register
Chapter D3
AArch64 PMU registers

This chapter describes the AArch64 PMU registers and shows examples of how to use them.

It contains the following sections:

- **D3.1 AArch64 PMU register summary** on page D3-380.
- **D3.2 PMCEID0_EL0, Performance Monitors Common Event Identification Register 0, EL0** on page D3-382.
- **D3.3 PMCEID1_EL0, Performance Monitors Common Event Identification Register 1, EL0** on page D3-386.
- **D3.4 PMCR_EL0, Performance Monitors Control Register, EL0** on page D3-389.
D3.1 AArch64 PMU register summary

The PMU counters and their associated control registers are accessible in the AArch64 Execution state with MRS and MSR instructions.

The following table gives a summary of the Cortex-A65 PMU registers in the AArch64 Execution state. For those registers not described in this chapter, see the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Width</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMCR_EL0</td>
<td>RW</td>
<td>32</td>
<td>0x41463040</td>
<td>D3.4 PMCR_EL0, Performance Monitors Control Register, EL0 on page D3-389</td>
</tr>
<tr>
<td>PMCNTENSET_EL0</td>
<td>RW</td>
<td>32</td>
<td>UNK</td>
<td>Performance Monitors Count Enable Set Register</td>
</tr>
<tr>
<td>PMCNTENCLR_EL0</td>
<td>RW</td>
<td>32</td>
<td>UNK</td>
<td>Performance Monitors Count Enable Clear Register</td>
</tr>
<tr>
<td>PMOVSCLR_EL0</td>
<td>RW</td>
<td>32</td>
<td>UNK</td>
<td>Performance Monitors Overflow Flag Status Register</td>
</tr>
<tr>
<td>PMSWINC_EL0</td>
<td>WO</td>
<td>32</td>
<td>UNK</td>
<td>Performance Monitors Software Increment Register</td>
</tr>
<tr>
<td>PMSELR_EL0</td>
<td>RW</td>
<td>32</td>
<td>UNK</td>
<td>Performance Monitors Event Counter Selection Register</td>
</tr>
<tr>
<td>PMCEID0_EL0</td>
<td>RO</td>
<td>64</td>
<td></td>
<td>D3.2 PMCEID0_EL0, Performance Monitors Common Event Identification Register 0, EL0 on page D3-382</td>
</tr>
<tr>
<td>PMCEID1_EL0</td>
<td>RO</td>
<td>64</td>
<td></td>
<td>D3.3 PMCEID1_EL0, Performance Monitors Common Event Identification Register 1, EL0 on page D3-386</td>
</tr>
</tbody>
</table>
Table D3-1 PMU register summary in the AArch64 Execution state (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Width</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMCCNTR_EL0</td>
<td>RW</td>
<td>64</td>
<td>UNK</td>
<td>Performance Monitors Cycle Count Register</td>
</tr>
<tr>
<td>PMXEVTYPE_EL0</td>
<td>RW</td>
<td>32</td>
<td>UNK</td>
<td>Performance Monitors Selected Event Type and Filter Register</td>
</tr>
<tr>
<td>PMCCFILTR_EL0</td>
<td>RW</td>
<td>32</td>
<td>UNK</td>
<td>Performance Monitors Cycle Count Filter Register</td>
</tr>
<tr>
<td>PMXEVCNTR_EL0</td>
<td>RW</td>
<td>32</td>
<td>UNK</td>
<td>Performance Monitors Selected Event Count Register</td>
</tr>
<tr>
<td>PMUSERENR_EL0</td>
<td>RW</td>
<td>32</td>
<td>UNK</td>
<td>Performance Monitors User Enable Register</td>
</tr>
<tr>
<td>PMINTENSET_EL1</td>
<td>RW</td>
<td>32</td>
<td>UNK</td>
<td>Performance Monitors Interrupt Enable Set Register</td>
</tr>
<tr>
<td>PMINTECLR_EL1</td>
<td>RW</td>
<td>32</td>
<td>UNK</td>
<td>Performance Monitors Interrupt Enable Clear Register</td>
</tr>
<tr>
<td>PMOVSSSET_EL0</td>
<td>RW</td>
<td>32</td>
<td>UNK</td>
<td>Performance Monitors Overflow Flag Status Set Register</td>
</tr>
<tr>
<td>PMEVCNTR0_EL0</td>
<td>RW</td>
<td>32</td>
<td>UNK</td>
<td>Performance Monitors Event Count Registers</td>
</tr>
<tr>
<td>PMEVCNTR1_EL0</td>
<td>RW</td>
<td>32</td>
<td>UNK</td>
<td>Performance Monitors Event Count Registers</td>
</tr>
<tr>
<td>PMEVCNTR2_EL0</td>
<td>RW</td>
<td>32</td>
<td>UNK</td>
<td>Performance Monitors Event Count Registers</td>
</tr>
<tr>
<td>PMEVCNTR3_EL0</td>
<td>RW</td>
<td>32</td>
<td>UNK</td>
<td>Performance Monitors Event Count Registers</td>
</tr>
<tr>
<td>PMEVCNTR4_EL0</td>
<td>RW</td>
<td>32</td>
<td>UNK</td>
<td>Performance Monitors Event Count Registers</td>
</tr>
<tr>
<td>PMEVCNTR5_EL0</td>
<td>RW</td>
<td>32</td>
<td>UNK</td>
<td>Performance Monitors Event Count Registers</td>
</tr>
<tr>
<td>PMEVTYPER0_EL0</td>
<td>RW</td>
<td>32</td>
<td>UNK</td>
<td>Performance Monitors Event Type Registers</td>
</tr>
<tr>
<td>PMEVTYPER1_EL0</td>
<td>RW</td>
<td>32</td>
<td>UNK</td>
<td>Performance Monitors Event Type Registers</td>
</tr>
<tr>
<td>PMEVTYPER2_EL0</td>
<td>RW</td>
<td>32</td>
<td>UNK</td>
<td>Performance Monitors Event Type Registers</td>
</tr>
<tr>
<td>PMEVTYPER3_EL0</td>
<td>RW</td>
<td>32</td>
<td>UNK</td>
<td>Performance Monitors Event Type Registers</td>
</tr>
<tr>
<td>PMEVTYPER4_EL0</td>
<td>RW</td>
<td>32</td>
<td>UNK</td>
<td>Performance Monitors Event Type Registers</td>
</tr>
<tr>
<td>PMEVTYPER5_EL0</td>
<td>RW</td>
<td>32</td>
<td>UNK</td>
<td>Performance Monitors Event Type Registers</td>
</tr>
<tr>
<td>PMCCFILTR_EL0</td>
<td>RW</td>
<td>32</td>
<td>UNK</td>
<td>Performance Monitors Cycle Count Filter Register</td>
</tr>
</tbody>
</table>
D3.2 PMCEID0_EL0, Performance Monitors Common Event Identification Register 0, EL0

The PMCEID0_EL0 defines which common architectural and common microarchitectural feature events are implemented.

Bit field descriptions

![PMCEID0_EL0 bit assignments](image)

ID[31:0], [31:0]

Common architectural and microarchitectural feature events that can be counted by the PMU event counters.

The following table shows the PMCEID0_EL0 bit assignments with event implemented or not implemented when the associated bit is set to 1 or 0. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile for more information about these events.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Event number</th>
<th>Event mnemonic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[31]</td>
<td>0x1F</td>
<td>L1D_CACHE_ALLOCATE</td>
<td>L1 Data cache allocate:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 This event is not implemented.</td>
</tr>
<tr>
<td>[30]</td>
<td>0x1E</td>
<td>CHAIN</td>
<td>Chain. For odd-numbered counters, counts once for each overflow of the preceding even-numbered counter. For even-numbered counters, does not count:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 This event is implemented.</td>
</tr>
<tr>
<td>[29]</td>
<td>0x1D</td>
<td>BUS_CYCLES</td>
<td>Bus cycle:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 This event is implemented.</td>
</tr>
<tr>
<td>[28]</td>
<td>0x1C</td>
<td>TTBR_WRITE_RETIRED</td>
<td>TTBR write, architecturally executed, condition check pass - write to translation table base:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 This event is implemented.</td>
</tr>
<tr>
<td>[27]</td>
<td>0x1B</td>
<td>INST_SPEC</td>
<td>Instruction speculatively executed:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 This event is implemented.</td>
</tr>
<tr>
<td>[26]</td>
<td>0x1A</td>
<td>MEMORY_ERROR</td>
<td>Local memory error:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 This event is implemented.</td>
</tr>
<tr>
<td>[25]</td>
<td>0x19</td>
<td>BUS_ACCESS</td>
<td>Bus access:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 This event is implemented.</td>
</tr>
<tr>
<td>Bit</td>
<td>Event number</td>
<td>Event mnemonic</td>
<td>Description</td>
</tr>
<tr>
<td>-----</td>
<td>--------------</td>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| [24]| 0x18         | L2D_CACHE_WB       | L2 Data cache Write-Back:  
|     |              |                    | θ This event is not implemented if the Cortex-A65 core has been configured without an L2 cache.  
|     |              |                    | 1 This event is implemented if the Cortex-A65 core has been configured with an L2 cache.  |
| [23]| 0x17         | L2D_CACHE_REFILL   | L2 Data cache refill:  
|     |              |                    | θ This event is not implemented if the Cortex-A65 core has been configured without an L2 and L3 cache. If configured with only an L3 cache, the L3 event will become an L2 event.  
|     |              |                    | 1 This event is implemented if the Cortex-A65 core has been configured with an L2 or L3 cache.  |
| [22]| 0x16         | L2D_CACHE         | L2 Data cache access:  
|     |              |                    | θ This event is not implemented if the Cortex-A65 core has been configured without an L2 and L3 cache. If configured with only an L3 cache, the L3 event will become an L2 event.  
|     |              |                    | 1 This event is implemented if the Cortex-A65 core has been configured with an L2 or L3 cache.  |
| [21]| 0x15         | L1D_CACHE_WB       | L1 Data cache Write-Back:  
|     |              |                    | 1 This event is implemented.  |
| [20]| 0x14         | L1I_CACHE         | L1 Instruction cache access:  
|     |              |                    | 1 This event is implemented.  |
| [19]| 0x13         | MEM_ACCESS        | Data memory access:  
|     |              |                    | 1 This event is implemented.  |
| [18]| 0x12         | BR_PRED           | Predictable branch speculatively executed:  
|     |              |                    | 1 This event is implemented.  |
| [17]| 0x11         | CPU_CYCLES        | Cycle:  
|     |              |                    | 1 This event is implemented.  |
| [16]| 0x10         | BR_MIS_PRED       | Mispredicted or not predicted branch speculatively executed:  
|     |              |                    | 1 This event is implemented.  |
| [15]| 0x0F         | UNALIGNED_LDST_RETIRED | Instruction architecturally executed, condition check pass - unaligned load or store:  
<p>|     |              |                    | θ This event is not implemented.  |</p>
<table>
<thead>
<tr>
<th>Bit</th>
<th>Event number</th>
<th>Event mnemonic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[14]</td>
<td>0x0E</td>
<td>BR_RETURN RETIRED</td>
<td>Instruction architecturally executed, condition check pass - procedure return:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 This event is implemented.</td>
</tr>
<tr>
<td>[13]</td>
<td>0x0D</td>
<td>BR_IMMED RETIRED</td>
<td>Instruction architecturally executed - immediate branch:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 This event is implemented.</td>
</tr>
<tr>
<td>[12]</td>
<td>0x0C</td>
<td>PC_WRITE RETIRED</td>
<td>Instruction architecturally executed, condition check pass - software change of the PC:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 This event is implemented.</td>
</tr>
<tr>
<td>[11]</td>
<td>0x0B</td>
<td>CID_WRITE RETIRED</td>
<td>Instruction architecturally executed, condition check pass - write to CONTEXTIDR:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 This event is implemented.</td>
</tr>
<tr>
<td>[10]</td>
<td>0x0A</td>
<td>EXC_RETURN</td>
<td>Instruction architecturally executed, condition check pass - exception return:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 This event is implemented.</td>
</tr>
<tr>
<td>[9]</td>
<td>0x09</td>
<td>EXC_TAKEN</td>
<td>Exception taken:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 This event is implemented.</td>
</tr>
<tr>
<td>[8]</td>
<td>0x08</td>
<td>INST RETIRED</td>
<td>Instruction architecturally executed:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 This event is implemented.</td>
</tr>
<tr>
<td>[7]</td>
<td>0x07</td>
<td>ST RETIRED</td>
<td>Instruction architecturally executed, condition check pass - store:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 This event is implemented.</td>
</tr>
<tr>
<td>[6]</td>
<td>0x06</td>
<td>LD RETIRED</td>
<td>Instruction architecturally executed, condition check pass - load:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 This event is implemented.</td>
</tr>
<tr>
<td>[5]</td>
<td>0x05</td>
<td>L1D_TLB REFILL</td>
<td>L1 Data TLB refill:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 This event is implemented.</td>
</tr>
<tr>
<td>[4]</td>
<td>0x04</td>
<td>L1D_CACHE</td>
<td>L1 Data cache access:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 This event is implemented.</td>
</tr>
<tr>
<td>[3]</td>
<td>0x03</td>
<td>L1D_CACHE REFILL</td>
<td>L1 Data cache refill:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 This event is implemented.</td>
</tr>
<tr>
<td>[2]</td>
<td>0x02</td>
<td>L1I_TLB REFILL</td>
<td>L1 Instruction TLB refill:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 This event is implemented.</td>
</tr>
<tr>
<td>Bit</td>
<td>Event number</td>
<td>Event mnemonic</td>
<td>Description</td>
</tr>
<tr>
<td>-----</td>
<td>--------------</td>
<td>----------------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| [1] | 0x01         | L1I_CACHE_REFILL | L1 Instruction cache refill:  
  1 This event is implemented. |
| [0] | 0x00         | SW_INCR         | Instruction architecturally executed, condition check pass - software increment:  
  1 This event is implemented. |

Bit fields and details not provided in this description are architecturally defined. See the *Arm*® *Architecture Reference Manual Armv8, for Armv8-A architecture profile.*
D3.3 PMCEID1_EL0, Performance Monitors Common Event Identification Register 1, EL0

The PMCEID1_EL0 defines which common architectural and common microarchitectural feature events are implemented.

**Bit field descriptions**

![Figure D3-2 PMCEID1_EL0 bit assignments](image)

ID[63:32], [31:0]

Common architectural and microarchitectural feature events that can be counted by the PMU event counters.

For each bit described in the following table, the event is implemented if the bit is set to 1, or not implemented if the bit is set to 0.

**Table D3-3 PMU common events**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Event number</th>
<th>Event mnemonic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>1  This event is implemented.</td>
</tr>
<tr>
<td>[23]</td>
<td>0x37</td>
<td>LL_CACHE_MISS_RD</td>
<td>Last Level cache miss, read.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1  This event is implemented.</td>
</tr>
<tr>
<td>[22]</td>
<td>0x36</td>
<td>LL_CACHE_RD</td>
<td>Last Level cache access, read.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1  This event is implemented.</td>
</tr>
<tr>
<td>[21]</td>
<td>0x35</td>
<td>ITLB_WALK</td>
<td>Access to instruction TLB that caused a page table walk.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1  This event is implemented.</td>
</tr>
<tr>
<td>[20]</td>
<td>0x34</td>
<td>DTLB_WALK</td>
<td>Access to data TLB that caused a page table walk.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1  This event is implemented.</td>
</tr>
<tr>
<td>[16]</td>
<td>0x30</td>
<td>L2I_TLB</td>
<td>Attributable Level 2 instruction TLB access.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0  This event is not implemented.</td>
</tr>
<tr>
<td>[15]</td>
<td>0x2F</td>
<td>L2D_TLB</td>
<td>Attributable Level 2 data or unified TLB access.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1  This event is implemented.</td>
</tr>
<tr>
<td>[14]</td>
<td>0x2E</td>
<td>L2I_TLB_REFILL</td>
<td>Attributable Level 2 instruction TLB refill.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0  This event is not implemented.</td>
</tr>
<tr>
<td>Bit</td>
<td>Event number</td>
<td>Event mnemonic</td>
<td>Description</td>
</tr>
<tr>
<td>-----</td>
<td>--------------</td>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>[13]</td>
<td>0x2D</td>
<td>L2D_TLB_REFILL</td>
<td>Attributable Level 2 data or unified TLB refill.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 This event is implemented.</td>
</tr>
<tr>
<td>[12]</td>
<td>0x2C</td>
<td>L3D_CACHE_WB</td>
<td>Attributable Level 3 data or unified cache write-back.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 This event is not implemented.</td>
</tr>
<tr>
<td>[11]</td>
<td>0x2B</td>
<td>L3D_CACHE</td>
<td>Attributable Level 3 data or unified cache access.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 This event is implemented if L2 and L3 are present.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 This event is not implemented if L2 and L3 are present.</td>
</tr>
<tr>
<td>[10]</td>
<td>0x2A</td>
<td>L3D_CACHE_REFILL</td>
<td>Attributable Level 3 data or unified cache refill.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 This event is implemented if L2 and L3 are present.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 This event is not implemented if L2 and L3 are not present.</td>
</tr>
<tr>
<td>[9]</td>
<td>0x29</td>
<td>L3D_CACHE_ALLOCATE</td>
<td>Attributable Level 3 data or unified cache allocation without refill.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 This event is implemented if L2 and L3 are present.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 This event is not implemented if L2 and L3 are not present.</td>
</tr>
<tr>
<td>[8]</td>
<td>0x28</td>
<td>L2I_CACHE_REFILL</td>
<td>Attributable Level 2 instruction cache refill.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 This event is not implemented.</td>
</tr>
<tr>
<td>[7]</td>
<td>0x27</td>
<td>L2I_CACHE</td>
<td>Attributable Level 2 instruction cache access.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 This event is not implemented.</td>
</tr>
<tr>
<td>[6]</td>
<td>0x26</td>
<td>L1I_TLB</td>
<td>Level 1 instruction TLB access.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 This event is implemented.</td>
</tr>
<tr>
<td>[5]</td>
<td>0x25</td>
<td>L1D_TLB</td>
<td>Level 1 data or unified TLB access.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 This event is implemented.</td>
</tr>
<tr>
<td>[4]</td>
<td>0x24</td>
<td>STALL_BACKEND</td>
<td>No operation issued due to backend.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 This event is implemented.</td>
</tr>
<tr>
<td>[3]</td>
<td>0x23</td>
<td>STALL_FRONTEND</td>
<td>No operation issued due to the frontend.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 This event is implemented.</td>
</tr>
<tr>
<td>[2]</td>
<td>0x22</td>
<td>BR_MIS_PRED_RETIRED</td>
<td>Instruction architecturally executed, mispredicted branch.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 This event is implemented.</td>
</tr>
</tbody>
</table>
### Table D3-3  PMU common events (continued)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Event number</th>
<th>Event mnemonic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>0x21</td>
<td>BR_RETIRED</td>
<td>Instruction architecturally executed, branch.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 This event is implemented.</td>
</tr>
<tr>
<td>[0]</td>
<td>0x20</td>
<td>L2D_CACHE_ALLOCATE</td>
<td>Level 2 data cache allocation without refill.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 This event is implemented.</td>
</tr>
</tbody>
</table>

Bit fields and details not provided in this description are architecturally defined. See the *Arm*® *Architecture Reference Manual* *Armv8, for Armv8-A architecture profile*. 
D3.4 PMCR_EL0, Performance Monitors Control Register, EL0

The PMCR_EL0 provides details of the Performance Monitors implementation, including the number of counters implemented, and configures and controls the counters.

Bit field descriptions

<table>
<thead>
<tr>
<th>31</th>
<th>24</th>
<th>23</th>
<th>16</th>
<th>15</th>
<th>11</th>
<th>10</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMP</td>
<td>IDCODE</td>
<td>N</td>
<td>LC</td>
<td>DP</td>
<td>X</td>
<td>D</td>
<td>C</td>
<td>P</td>
<td>E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure D3-3  PMCR_EL0 bit assignments

IMP, [31:24]
Implementer code:
0x41 Arm.
This is a read-only field.

IDCODE, [23:16]
Identification code:
0x46 Cortex-A65.
This is a read-only field.

N, [15:11]
Number of event counters.
0b00110 Six counters.

RES0, [10:7]
RES0 Reserved.

LC, [6]
Long cycle count enable. Determines which PMCCNTR_EL0 bit generates an overflow recorded in PMOVSR[31]. The possible values are:
0 Overflow on increment that changes PMCCNTR_EL0[31] from 1 to 0.
1 Overflow on increment that changes PMCCNTR_EL0[63] from 1 to 0.

DP, [5]
Disable cycle counter, PMCCNTR_EL0 when event counting is prohibited:
0 Cycle counter operates regardless of the non-invasive debug authentication settings. This is the reset value.
1 Cycle counter is disabled if non-invasive debug is not permitted and enabled.
This bit is read/write.

X, [4]
Export enable. This bit permits events to be exported to another debug device, such as a trace macrocell, over an event bus:
0 Export of events is disabled. This is the reset value.
1 Export of events is enabled.

This bit is read/write and does not affect the generation of Performance Monitors interrupts on the nPMUIRQ pin.

D, [3]
Clock divider:
0 When enabled, PMCCNTR_EL0 counts every clock cycle. This is the reset value.
1 When enabled, PMCCNTR_EL0 counts every 64 clock cycles.

This bit is read/write.

C, [2]
Clock counter reset. This bit is WO. The effects of writing to this bit are:
0 No action. This is the reset value.
1 Reset PMCCNTR_EL0 to 0.

This bit is always RAZ.

Resetting PMCCNTR_EL0 does not clear the PMCCNTR_EL0 overflow bit to 0. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile for more information.

P, [1]
Event counter reset. This bit is WO. The effects of writing to this bit are:
0 No action. This is the reset value.
1 Reset all event counters, not including PMCCNTR_EL0, to zero.

This bit is always RAZ.

In Non-secure EL0 and EL1, a write of 1 to this bit does not reset event counters that MDCR_EL2.HPMN reserves for EL2 use.

In EL2 and EL3, a write of 1 to this bit resets all the event counters.

Resetting the event counters does not clear any overflow bits to 0.

E, [0]
Enable. The possible values of this bit are:
0 All counters, including PMCCNTR_EL0, are disabled. This is the reset value.
1 All counters are enabled.

This bit is RW.

In Non-secure EL0 and EL1, this bit does not affect the operation of event counters that MDCR_EL2.HPMN reserves for EL2 use.

On Warm reset, the field resets to 0.

Configurations

There are no configuration notes.

Bit fields and details not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.
Chapter D4
Memory-mapped PMU registers

This chapter describes the memory-mapped PMU registers and shows examples of how to use them.

It contains the following sections:
- D4.1 Memory-mapped PMU register summary on page D4-392.
- D4.2 PMCFG0, Performance Monitors Configuration Register on page D4-396.
- D4.3 PMCIDR0, Performance Monitors Component Identification Register 0 on page D4-397.
- D4.4 PMCIDR1, Performance Monitors Component Identification Register 1 on page D4-398.
- D4.5 PMCIDR2, Performance Monitors Component Identification Register 2 on page D4-399.
- D4.6 PMCIDR3, Performance Monitors Component Identification Register 3 on page D4-400.
- D4.7 PMPIDR0, Performance Monitors Peripheral Identification Register 0 on page D4-401.
- D4.8 PMPIDR1, Performance Monitors Peripheral Identification Register 1 on page D4-402.
- D4.9 PMPIDR2, Performance Monitors Peripheral Identification Register 2 on page D4-403.
- D4.10 PMPIDR3, Performance Monitors Peripheral Identification Register 3 on page D4-404.
- D4.11 PMPIDR4, Performance Monitors Peripheral Identification Register 4 on page D4-405.
- D4.12 PMPIDRn, Performance Monitors Peripheral Identification Register 5-7 on page D4-406.
## D4.1 Memory-mapped PMU register summary

There are PMU registers that are accessible through the external debug interface.

These registers are listed in the following table. For those registers not described in this chapter, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>PMEVCNTR0_EL0</td>
<td>RW</td>
<td>Performance Monitor Event Count Register 0</td>
</tr>
<tr>
<td>0x004</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x008</td>
<td>PMEVCNTR1_EL0</td>
<td>RW</td>
<td>Performance Monitor Event Count Register 1</td>
</tr>
<tr>
<td>0x00C</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x010</td>
<td>PMEVCNTR2_EL0</td>
<td>RW</td>
<td>Performance Monitor Event Count Register 2</td>
</tr>
<tr>
<td>0x014</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x018</td>
<td>PMEVCNTR3_EL0</td>
<td>RW</td>
<td>Performance Monitor Event Count Register 3</td>
</tr>
<tr>
<td>0x01C</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x020</td>
<td>PMEVCNTR4_EL0</td>
<td>RW</td>
<td>Performance Monitor Event Count Register 4</td>
</tr>
<tr>
<td>0x024</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x028</td>
<td>PMEVCNTR5_EL0</td>
<td>RW</td>
<td>Performance Monitor Event Count Register 5</td>
</tr>
<tr>
<td>0x02C - 0xF4</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0F8</td>
<td>PMCCNTR_EL0[31:0]</td>
<td>RW</td>
<td>Performance Monitor Cycle Count Register</td>
</tr>
<tr>
<td>0x0FC</td>
<td>PMCCNTR_EL0[63:32]</td>
<td>RW</td>
<td>Performance Monitor Cycle Count Register (alias)</td>
</tr>
<tr>
<td>0x200</td>
<td>PMPCSR[31:0]</td>
<td>RO</td>
<td>Program Counter Sample Register</td>
</tr>
<tr>
<td>0x204</td>
<td>PMPCSR[63:32]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x208</td>
<td>PMCID1SR</td>
<td>RO</td>
<td>CONTEXTIDR_EL1 Sample Register</td>
</tr>
<tr>
<td>0x20C</td>
<td>PMVIDSR</td>
<td>RO</td>
<td>VMID Sample Register</td>
</tr>
<tr>
<td>0x220</td>
<td>PMPCSR[31:0]</td>
<td>RO</td>
<td>Program Counter Sample Register (alias)</td>
</tr>
<tr>
<td>0x224</td>
<td>PMPCSR[63:32]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x228</td>
<td>PMCID1SR</td>
<td>RO</td>
<td>CONTEXTIDR_EL1 Sample Register (alias)</td>
</tr>
<tr>
<td>0x22C</td>
<td>PMCID2SR</td>
<td>RO</td>
<td>CONTEXTIDR_EL2 Sample Register</td>
</tr>
<tr>
<td>0x100 - 0x3FC</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>Offset</td>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------------</td>
<td>------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>0x418-0x478</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x47C</td>
<td>PMCCFILTR_EL0</td>
<td>RW</td>
<td>Performance Monitor Cycle Count Filter Register</td>
</tr>
<tr>
<td>0x600</td>
<td>PMPCSSR_LO</td>
<td>RO</td>
<td>D5.2 PMPCSSR, Snapshot Program Counter Sample Register on page D5-409</td>
</tr>
<tr>
<td>0x604</td>
<td>PMPCSSR_HI</td>
<td>RO</td>
<td></td>
</tr>
<tr>
<td>0x608</td>
<td>PMCIDSSR</td>
<td>RO</td>
<td>D5.3 PMCIDSSR, Snapshot CONTEXTIDR_EL1 Sample Register on page D5-410</td>
</tr>
<tr>
<td>0x60C</td>
<td>PMCID2SSR</td>
<td>RO</td>
<td>D5.4 PMCID2SSR, Snapshot CONTEXTIDR_EL2 Sample Register on page D5-411</td>
</tr>
<tr>
<td>0x610</td>
<td>PMSSSR</td>
<td>RO</td>
<td>D5.5 PMSSSR, PMU Snapshot Status Register on page D5-412</td>
</tr>
<tr>
<td>0x614</td>
<td>PMOVSSR</td>
<td>RO</td>
<td>D5.6 PMOVSSR, PMU Overflow Status Snapshot Register on page D5-413</td>
</tr>
<tr>
<td>0x618</td>
<td>PMCCNTSR_LO</td>
<td>RO</td>
<td>D5.7 PMCCNTSR, PMU Cycle Counter Snapshot Register on page D5-414</td>
</tr>
<tr>
<td>0x61C</td>
<td>PMCCNTSR_HI</td>
<td>RO</td>
<td></td>
</tr>
<tr>
<td>0x620+4×n</td>
<td>PMEVCNTSRn</td>
<td>RO</td>
<td>D5.8 PMEVCNTSRn, PMU Cycle Counter Snapshot Registers 0-5 on page D5-415</td>
</tr>
<tr>
<td>0x6F0</td>
<td>PMSSCR</td>
<td>WO</td>
<td>D5.9 PMSSCR, PMU Snapshot Capture Register on page D5-416</td>
</tr>
<tr>
<td>0xC00</td>
<td>PMCNTENSET_EL0</td>
<td>RW</td>
<td>Performance Monitor Count Enable Set Register</td>
</tr>
<tr>
<td>0xC04-0xC1C</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0xC20</td>
<td>PMCNTENCLR_EL0</td>
<td>RW</td>
<td>Performance Monitor Count Enable Clear Register</td>
</tr>
<tr>
<td>0xC24-0xC3C</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0xC40</td>
<td>PMIINTENSET_EL1</td>
<td>RW</td>
<td>Performance Monitor Interrupt Enable Set Register</td>
</tr>
<tr>
<td>0xC44-0xC5C</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0xC60</td>
<td>PMIINTENCLR_EL1</td>
<td>RW</td>
<td>Performance Monitor Interrupt Enable Clear Register</td>
</tr>
<tr>
<td>0xC64-0xC7C</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0xC80</td>
<td>PMOVSCLR_EL0</td>
<td>RW</td>
<td>Performance Monitor Overflow Flag Status Register</td>
</tr>
<tr>
<td>0xC84-0xC9C</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0xCA0</td>
<td>PMSWINC_EL0</td>
<td>WO</td>
<td>Performance Monitor Software Increment Register</td>
</tr>
<tr>
<td>Offset</td>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>------------------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>0xCA4-0xCBC</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0xCC0</td>
<td>PMOVSET_EL0</td>
<td>RW</td>
<td>Performance Monitor Overflow Flag Status Set Register</td>
</tr>
<tr>
<td>0xCC4-0xDFC</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0xE00</td>
<td>PMCFGR</td>
<td>RO</td>
<td>D4.2 PMCFGR, Performance Monitors Configuration Register on page D4-396</td>
</tr>
<tr>
<td>0xE04</td>
<td>PMCR_EL0</td>
<td>RW</td>
<td>Performance Monitors Control Register. This register is distinct from the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PMCR_EL0 system register. It does not have the same value.</td>
</tr>
<tr>
<td>0xE08-0xE1C</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0xE20</td>
<td>PMCEID0</td>
<td>RO</td>
<td>D3.2 PMCEID0_EL0, Performance Monitors Common Event Identification Register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0_EL0 on page D3-382</td>
</tr>
<tr>
<td>0xE24</td>
<td>PMCEID1</td>
<td>RO</td>
<td>D3.3 PMCEID1_EL0, Performance Monitors Common Event Identification Register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1_EL0 on page D3-386</td>
</tr>
<tr>
<td>0xE28</td>
<td>PMCEID2</td>
<td>RO</td>
<td>Reserved</td>
</tr>
<tr>
<td>0xE2C</td>
<td>PMCEID3</td>
<td>RO</td>
<td>Reserved</td>
</tr>
<tr>
<td>0xFA8</td>
<td>PMDEV AFF0</td>
<td>RO</td>
<td>B1.65 MPIDR_EL1, Multiprocessor Affinity Register, EL1 on page B1-218</td>
</tr>
<tr>
<td>0xFAC</td>
<td>PMDEV AFF1</td>
<td>RO</td>
<td>B1.65 MPIDR_EL1, Multiprocessor Affinity Register, EL1 on page B1-218</td>
</tr>
<tr>
<td>0xFB8</td>
<td>PMAUTHSTATUS</td>
<td>RO</td>
<td>Performance Monitor Authentication Status Register</td>
</tr>
<tr>
<td>0xFB8</td>
<td>PMDEVARCH</td>
<td>RO</td>
<td>Performance Monitor Device Architecture Register</td>
</tr>
<tr>
<td>0xFC0-0xFC8</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0xFD0</td>
<td>PMPIDR4</td>
<td>RO</td>
<td>D4.11 PMPIDR4, Performance Monitors Peripheral Identification Register 4 on</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>page D4-405</td>
</tr>
<tr>
<td>0xFD4</td>
<td>PMPIDR5</td>
<td>RO</td>
<td>D4.12 PMPIDRn, Performance Monitors Peripheral Identification Register 5-7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>on page D4-406</td>
</tr>
<tr>
<td>0xFD8</td>
<td>PMPIDR6</td>
<td>RO</td>
<td></td>
</tr>
<tr>
<td>0xFD0</td>
<td>PMPIDR7</td>
<td>RO</td>
<td></td>
</tr>
<tr>
<td>0xFE0</td>
<td>PMPIDR0</td>
<td>RO</td>
<td>D4.7 PMPIDR0, Performance Monitors Peripheral Identification Register 0 on</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>page D4-401</td>
</tr>
<tr>
<td>Offset</td>
<td>Name</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-----------</td>
<td>------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>0xFE4</td>
<td>PMPIDR1</td>
<td>RO</td>
<td>D4.8 PMPIDR1, Performance Monitors Peripheral Identification Register 1 on page D4-402</td>
</tr>
<tr>
<td>0xFE8</td>
<td>PMPIDR2</td>
<td>RO</td>
<td>D4.9 PMPIDR2, Performance Monitors Peripheral Identification Register 2 on page D4-403</td>
</tr>
<tr>
<td>0xFEC</td>
<td>PMPIDR3</td>
<td>RO</td>
<td>D4.10 PMPIDR3, Performance Monitors Peripheral Identification Register 3 on page D4-404</td>
</tr>
<tr>
<td>0xFF0</td>
<td>PMCIDR0</td>
<td>RO</td>
<td>D4.3 PMCIDR0, Performance Monitors Component Identification Register 0 on page D4-397</td>
</tr>
<tr>
<td>0xFF4</td>
<td>PMCIDR1</td>
<td>RO</td>
<td>D4.4 PMCIDR1, Performance Monitors Component Identification Register 1 on page D4-398</td>
</tr>
<tr>
<td>0xFF8</td>
<td>PMCIDR2</td>
<td>RO</td>
<td>D4.5 PMCIDR2, Performance Monitors Component Identification Register 2 on page D4-399</td>
</tr>
<tr>
<td>0xFFC</td>
<td>PMCIDR3</td>
<td>RO</td>
<td>D4.6 PMCIDR3, Performance Monitors Component Identification Register 3 on page D4-400</td>
</tr>
</tbody>
</table>
**D4.2 PMCFGR, Performance Monitors Configuration Register**

The PMCFGR contains PMU specific configuration data.

**Bit field descriptions**

The PMCFGR is a 32-bit register.

```
<table>
<thead>
<tr>
<th>31</th>
<th>17</th>
<th>16</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>8</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>RES0</td>
<td>Size</td>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCD</td>
<td>EX</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

**RES0, [31:17]**

RES0 Reserved.

**EX, [16]**

Export supported. The value is:

1 Export is supported. PMCR_EL0.EX is read/write.

**CCD, [15]**

Cycle counter has pre-scale. The value is:

1 PMCR_EL0.D is read/write.

**CC, [14]**

Dedicated cycle counter supported. The value is:

1 Dedicated cycle counter is supported.

**Size, [13:8]**

Counter size. The value is:

0b11111111 64-bit counters.

**N, [7:0]**

Number of event counters. The value is:

0x06 Six counters.

Bit fields and details not provided in this description are architecturally defined. See the *Arm* 

The PMCFGR can be accessed through the external debug interface, offset 0xE00.
D4.3 PMCIDR0, Performance Monitors Component Identification Register 0

The PMCIDR0 provides information to identify a Performance Monitor component.

**Bit field descriptions**

The PMCIDR0 is a 32-bit register.

![PMCIDR0 bit assignments](image)

RES0, [31:8]

RES0  Reserved.

PRMBL_0, [7:0]

0x0D  Preamble byte 0.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual* Armv8, for Armv8-A architecture profile.

The PMCIDR0 can be accessed through the external debug interface, offset 0xFF0.
D4.4 PMCIDR1, Performance Monitors Component Identification Register 1

The PMCIDR1 provides information to identify a Performance Monitor component.

**Bit field descriptions**

The PMCIDR1 is a 32-bit register.

![Figure D4-3 PMCIDR1 bit assignments](image)

**RES0, [31:8]**

RES0  Reserved.

**CLASS, [7:4]**

0x9  Debug component.

**PRMBL_1, [3:0]**

0x0  Preamble byte 1.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The PMCIDR1 can be accessed through the external debug interface, offset 0xFF4.
D4.5  PMCIDR2, Performance Monitors Component Identification Register 2

The PMCIDR2 provides information to identify a Performance Monitor component.

**Bit field descriptions**

The PMCIDR2 is a 32-bit register.

![PMCIDR2 bit assignments](image)

**RES0, [31:8]**

RES0  Reserved.

**PRMBL_2, [7:0]**

0x05  Preamble byte 2.

Bit fields and details not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.

The PMCIDR2 can be accessed through the external debug interface, offset 0xFF8.
D4.6  **PMCIDR3, Performance Monitors Component Identification Register 3**

The PMCIDR3 provides information to identify a Performance Monitor component.

**Bit field descriptions**

The PMCIDR3 is a 32-bit register.

![PMCIDR3 bit assignments](image)

**RES0, [31:8]**

RES0  Reserved.

**PRMBL_3, [7:0]**

0x81  Preamble byte 3.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The PMCIDR3 can be accessed through the external debug interface, offset 0xFFFc.
D4.7 PMPIDR0, Performance Monitors Peripheral Identification Register 0

The PMPIDR0 provides information to identify a Performance Monitor component.

**Bit field descriptions**

The PMPIDR0 is a 32-bit register.

![PMPIDR0 bit assignments](image)

**RES0, [31:8]**

RES0  
Reserved.

**Part_0, [7:0]**

0x06  
Least significant byte of the performance monitor part number.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The PMPIDR0 can be accessed through the external debug interface, offset 0xFE0.
D4.8  **PMPIDR1, Performance Monitors Peripheral Identification Register 1**

The PMPIDR1 provides information to identify a Performance Monitor component.

**Bit field descriptions**

The PMPIDR1 is a 32-bit register.

![Figure D4-7 PMPIDR1 bit assignments](image)

**RES0, [31:8]**

RES0  Reserved.

**DES_0, [7:4]**

0xB  Arm Limited. This is the least significant nibble of JEP106 ID code.

**Part_1, [3:0]**

0xD  Most significant nibble of the performance monitor part number.

Bit fields and details not provided in this description are architecturally defined. See the *Arm*<sup>®</sup> Architecture Reference Manual Armv8, for Armv8-A architecture profile.

The PMPIDR1 can be accessed through the external debug interface, offset 0xFE4.
D4.9  PMPIDR2, Performance Monitors Peripheral Identification Register 2

The PMPIDR2 provides information to identify a Performance Monitor component.

**Bit field descriptions**

The PMPIDR2 is a 32-bit register.

<table>
<thead>
<tr>
<th>Bit Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RES0 [31:8]</td>
<td>Reserved.</td>
</tr>
<tr>
<td>Revision [7:4]</td>
<td>0x2 r1p1.</td>
</tr>
<tr>
<td>JEDEC [3]</td>
<td>0b1 RAO. Indicates a JEP106 identity code is used.</td>
</tr>
<tr>
<td>DES_1 [2:0]</td>
<td>0b011 Arm Limited. This is the most significant nibble of JEP106 ID code.</td>
</tr>
</tbody>
</table>

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual* *Armv8, for Armv8-A architecture profile*.

The PMPIDR2 can be accessed through the external debug interface, offset 0xFE8.
D4.10 PMPIDR3, Performance Monitors Peripheral Identification Register 3

The PMPIDR3 provides information to identify a Performance Monitor component.

**Bit field descriptions**

The PMPIDR3 is a 32-bit register.

![Figure D4-9 PMPIDR3 bit assignments](image)

RES0, [31:8]

| RES0 | Reserved |

REV AND, [7:4]

| REV AND | Part minor revision |

CMOD, [3:0]

| CMOD | Customer modified |

Bit fields and details not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.

The PMPIDR3 can be accessed through the external debug interface, offset 0xFEC.
D4.11 PMPIDR4, Performance Monitors Peripheral Identification Register 4

The PMPIDR4 provides information to identify a Performance Monitor component.

**Bit field descriptions**

The PMPIDR4 is a 32-bit register.

![Figure D4-10 PMPIDR4 bit assignments](image)

RES0, [31:8]

RES0  Reserved.

Size, [7:4]

0x0  Size of the component. Log₂ the number of 4KB pages from the start of the component to the end of the component ID registers.

DES_2, [3:0]

0x4  Arm Limited. This is the least significant nibble JEP106 continuation code.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The PMPIDR4 can be accessed through the external debug interface, offset 0xFD0.
D4.12 PMPIDRn, Performance Monitors Peripheral Identification Register 5-7

No information is held in the Peripheral ID5, Peripheral ID6, and Peripheral ID7 Registers. They are reserved for future use and are RES0.
Chapter D5
PMU snapshot registers

PMU snapshot registers are an IMPLEMENTATION DEFINED extension to an Armv8-A compliant PMU to support an external core monitor that connects to a system profiler.

It contains the following sections:
• D5.1 PMU snapshot register summary on page D5-408.
• D5.2 PMPCSSR, Snapshot Program Counter Sample Register on page D5-409.
• D5.3 PMCIDSSR, Snapshot CONTEXTIDR_EL1 Sample Register on page D5-410.
• D5.4 PMCID2SSR, Snapshot CONTEXTIDR_EL2 Sample Register on page D5-411.
• D5.5 PMSSSR, PMU Snapshot Status Register on page D5-412.
• D5.6 PMOVSSR, PMU Overflow Status Snapshot Register on page D5-413.
• D5.7 PMCCNTSR, PMU Cycle Counter Snapshot Register on page D5-414.
• D5.8 PMEVCNTSRn, PMU Cycle Counter Snapshot Registers 0-5 on page D5-415.
• D5.9 PMSSCR, PMU Snapshot Capture Register on page D5-416.
D5.1 PMU snapshot register summary

The snapshot registers are visible in an IMPLEMENTATION DEFINED region of the PMU external debug interface. Each time the debugger sends a snapshot request, information is collected to see how the code is executed in the different cores.

The following table describes the PMU snapshot registers implemented in the core.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Width</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x600</td>
<td>PMPCSSR_LO</td>
<td>RO</td>
<td>32</td>
<td>D5.2 PMPCSSR, Snapshot Program Counter Sample Register on page D5-409</td>
</tr>
<tr>
<td>0x604</td>
<td>PMPCSSR_HI</td>
<td>RO</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>0x608</td>
<td>PMPCIDSSR</td>
<td>RO</td>
<td>32</td>
<td>D5.3 PMCIDSSR, Snapshot CONTEXTIDR_EL1 Sample Register on page D5-410</td>
</tr>
<tr>
<td>0x60C</td>
<td>PMPCID2SSR</td>
<td>RO</td>
<td>32</td>
<td>D5.4 PMCID2SSR, Snapshot CONTEXTIDR_EL2 Sample Register on page D5-411</td>
</tr>
<tr>
<td>0x610</td>
<td>PMSSSR</td>
<td>RO</td>
<td>32</td>
<td>D5.5 PMSSSR, PMU Snapshot Status Register on page D5-412</td>
</tr>
<tr>
<td>0x614</td>
<td>PMOVSSR</td>
<td>RO</td>
<td>32</td>
<td>D5.6 PMOVSSR, PMU Overflow Status Snapshot Register on page D5-413</td>
</tr>
<tr>
<td>0x618</td>
<td>PMCCNTSR_LO</td>
<td>RO</td>
<td>32</td>
<td>D5.7 PMCCNTSR, PMU Cycle Counter Snapshot Register on page D5-414</td>
</tr>
<tr>
<td>0x61C</td>
<td>PMCCNTSR_HI</td>
<td>RO</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>0x620 + 4×n</td>
<td>PMEVCNTSRn</td>
<td>RO</td>
<td>32</td>
<td>D5.8 PMEVCNTSRn, PMU Cycle Counter Snapshot Registers 0-5 on page D5-415</td>
</tr>
<tr>
<td>0x6F0</td>
<td>PMSSCR</td>
<td>WO</td>
<td>32</td>
<td>D5.9 PMSSCR, PMU Snapshot Capture Register on page D5-416</td>
</tr>
</tbody>
</table>
D5.2 PMPCSSR, Snapshot Program Counter Sample Register

The PMPCSSR is an alias for the PCSR register. However, unlike the other view of PCSR, it is not sensitive to reads. That is, reads of PMPCSSR through the PMU snapshot view do not cause a new sample capture and do not change CIDS, CID2SR, or VIDSR.

**Bit field descriptions**

The PMPCSSR is a 64-bit read-only register.

![PMPCSSR bit assignments](image)

- **NS, [63]**
  - Non-secure sample.
- **EL, [62:61]**
  - Exception level sample.
- **RES0, [60:56]**
  - Reserved, RES0.
- **PC, [55:0]**
  - Sampled PC.

**Configurations**

There are no configuration notes.

**Usage constraints**

Any access to PMPCSSR returns an error if any of the following occurs:

- The core power domain is off.
- DoubleLockStatus() == TRUE.
D5.3 PMCIDSSR, Snapshot CONTEXTIDR_EL1 Sample Register

The PMCIDSSR is an alias for the CIDSR register.

Configurations
There are no configuration notes.

Usage constraints
Any access to PMCIDSSR returns an error if any of the following occurs:

- The core power domain is off.
- DoubleLockStatus() == TRUE.
D5.4 PMCID2SSR, Snapshot CONTEXTIDR_EL2 Sample Register

The PMCID2SSR is an alias for the CID2SR register.

Configurations
There are no configuration notes.

Usage constraints
Any access to PMCID2SSR returns an error if any of the following occurs:
• The core power domain is off.
• DoubleLockStatus() == TRUE.
D5.5 PMSSSR, PMU Snapshot Status Register

The PMSSSR holds status information about the captured counters.

**Bit field descriptions**

The PMSSSR is a 32-bit read-only register.

![PMSSSR bit assignments](image)

**RES0, [31:1]**

Reserved, RES0.

**NC, [0]**

No capture. This bit indicates whether the PMU counters have been captured. The possible values are:

- 0 PMU counters are captured.
- 1 PMU counters are not captured.

If there is a security violation, the core does not capture the event counters. The external monitor is responsible for keeping track of whether it managed to capture the snapshot registers from the core.

This bit does not reflect the status of the captured Program Counter Sample registers.

The core resets this bit to 1 by a Warm reset but MPSSSR.NC is overwritten at the first capture.

**Configurations**

There are no configuration notes.

**Usage constraints**

Any access to PMSSSR returns an error if any of the following occurs:

- The core power domain is off.
- DoubleLockStatus() == TRUE.
D5.6   PMOVSSR, PMU Overflow Status Snapshot Register

The PMOVSSR is a captured copy of PMOVSR.

Once it is captured, the value in PMOVSSR is unaffected by writes to PMOVSET_EL0 and PMOVSCLR_EL0.

Configurations

There are no configuration notes.

Usage constraints

Any access to PMOVSSR returns an error if any of the following occurs:

- The core power domain is off.
- DoubleLockStatus() == TRUE.
D5.7  PMCCNTSR, PMU Cycle Counter Snapshot Register

The PMCCNTSR is a captured copy of PMCCNTR_EL0.
Once it is captured, the value in PMCCNTSR is unaffected by writes to PMCCNTR_EL0 and PMCR_EL0.C.

Configurations
There are no configuration notes.

Usage constraints
Any access to PMCCNTSR returns an error if any of the following occurs:
• The core power domain is off.
• DoubleLockStatus() == TRUE.
D5.8 PMEVCNTSRn, PMU Cycle Counter Snapshot Registers 0-5

The PMEVCNTSRn, are captured copies of PMEVCNTRn_EL0, n is 0-5. When they are captured, the value in PMSSEVCNTRn is unaffected by writes to PMSSEVCNTRn_EL0 and PMCR_EL0.P.

**Configurations**

There are no configuration notes.

**Usage constraints**

Any access to PMSSEVCNTRn returns an error if any of the following occurs:

- The core power domain is off.
- DoubleLockStatus() == TRUE.
D5.9 PMSSCR, PMU Snapshot Capture Register

The PMSSCR provides a mechanism for software to initiate a sample.

**Bit field descriptions**

The PMSSCR is a 32-bit write-only register.

![PMSSCR bit assignments](image)

<table>
<thead>
<tr>
<th>Bit Position</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-1</td>
<td>RES0</td>
<td>Reserved</td>
</tr>
<tr>
<td>0</td>
<td>SS</td>
<td>Capture now. The possible values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: IGNORED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Initiate a capture immediately.</td>
</tr>
</tbody>
</table>

**Configurations**

There are no configuration notes.

**Usage constraints**

Any access to PMSSCR returns an error if any of the following occurs:

- The core power domain is off.
- DoubleLockStatus() == TRUE.
Chapter D6
ETM registers

This chapter describes the ETM registers.

It contains the following sections:
- **D6.1 ETM register summary** on page D6-419.
- **D6.2 TRCACATRn, Address Comparator Access Type Registers 0-7** on page D6-423.
- **D6.3 TRCACVRn, Address Comparator Value Registers 0-7** on page D6-425.
- **D6.4 TRCAUTHSTATUS, Authentication Status Register** on page D6-426.
- **D6.5 TRCAUXCTLR, Auxiliary Control Register** on page D6-427.
- **D6.6 TRCBBCTLR, Branch Broadcast Control Register** on page D6-429.
- **D6.7 TRCCCCTRLR, Cycle Count Control Register** on page D6-430.
- **D6.8 TRCCIDCCTRLR0, Context ID Comparator Control Register 0** on page D6-431.
- **D6.9 TRCCIDCVR0, Context ID Comparator Value Register 0** on page D6-432.
- **D6.10 TRCCIDR0, ETM Component Identification Register 0** on page D6-433.
- **D6.11 TRCCIDR1, ETM Component Identification Register 1** on page D6-434.
- **D6.12 TRCCIDR2, ETM Component Identification Register 2** on page D6-435.
- **D6.13 TRCCIDR3, ETM Component Identification Register 3** on page D6-436.
- **D6.14 TRCLAIMCLR, Claim Tag Clear Register** on page D6-437.
- **D6.15 TRCLAIMSET, Claim Tag Set Register** on page D6-438.
- **D6.16 TRCCNTRCTRLR0, Counter Control Register 0** on page D6-439.
- **D6.17 TRCCNTRCTRLR1, Counter Control Register 1** on page D6-441.
- **D6.18 TRCCNTRLDVRn, Counter Reload Value Registers 0-1** on page D6-443.
- **D6.19 TRCCNTVRn, Counter Value Registers 0-1** on page D6-444.
- **D6.20 TRCCONFIGR, Trace Configuration Register** on page D6-445.
- **D6.21 TRCDEVAFF0, Device Affinity Register 0** on page D6-448.
- **D6.22 TRCDEVAFF1, Device Affinity Register 1** on page D6-449.
- **D6.23 TRCDEVARCH, Device Architecture Register** on page D6-450.
• D6.24 TRCDEVID, Device ID Register on page D6-451.
• D6.25 TRCDEVTYP, Device Type Register on page D6-452.
• D6.26 TRCEVENTCTL0R, Event Control 0 Register on page D6-453.
• D6.27 TRCEVENTCTL1R, Event Control 1 Register on page D6-455.
• D6.28 TRCEXTINSELR, External Input Select Register on page D6-456.
• D6.29 TRCIDR0, ID Register 0 on page D6-457.
• D6.30 TRCIDR1, ID Register 1 on page D6-459.
• D6.31 TRCIDR2, ID Register 2 on page D6-460.
• D6.32 TRCIDR3, ID Register 3 on page D6-462.
• D6.33 TRCIDR4, ID Register 4 on page D6-464.
• D6.34 TRCIDR5, ID Register 5 on page D6-466.
• D6.35 TRCIDR8, ID Register 8 on page D6-468.
• D6.36 TRCIDR9, ID Register 9 on page D6-469.
• D6.37 TRCIDR10, ID Register 10 on page D6-470.
• D6.38 TRCIDR11, ID Register 11 on page D6-471.
• D6.39 TRCIDR12, ID Register 12 on page D6-472.
• D6.40 TRCIDR13, ID Register 13 on page D6-473.
• D6.41 TRCIMSPEC0, IMPLEMENTATION SPECIFIC Register 0 on page D6-474.
• D6.42 TRCITATBIDR, Integration ATB Identification Register on page D6-475.
• D6.43 TRCITCTRL, Integration Mode Control Register on page D6-476.
• D6.44 TRCITATBINR, Integration Instruction ATB In Register on page D6-477.
• D6.45 TRCITATBOUTR, Integration Instruction ATB Out Register on page D6-478.
• D6.46 TRCITIDATAR, Integration Instruction ATB Data Register on page D6-479.
• D6.47 TRCLAR, Software Lock Access Register on page D6-480.
• D6.48 TRCLSR, Software Lock Status Register on page D6-481.
• D6.49 TRCCNTVRn, Counter Value Registers 0-1 on page D6-482.
• D6.50 TRCOSLR, OS Lock Access Register on page D6-483.
• D6.51 TRCOSLSR, OS Lock Status Register on page D6-484.
• D6.52 TRCPDCCR, Power Down Control Register on page D6-485.
• D6.53 TRCPDSR, Power Down Status Register on page D6-486.
• D6.54 TRCPIDR0, ETM Peripheral Identification Register 0 on page D6-487.
• D6.55 TRCPIDR1, ETM Peripheral Identification Register 1 on page D6-488.
• D6.56 TRCPIDR2, ETM Peripheral Identification Register 2 on page D6-489.
• D6.57 TRCPIDR3, ETM Peripheral Identification Register 3 on page D6-490.
• D6.58 TRCPIDR4, ETM Peripheral Identification Register 4 on page D6-491.
• D6.59 TRCPIDRn, ETM Peripheral Identification Registers 5-7 on page D6-492.
• D6.60 TRCPRGCTRL, Programming Control Register on page D6-493.
• D6.61 TRCRSCTLRn, Resource Selection Control Registers 2-16 on page D6-494.
• D6.62 TRCSEQEVn, Sequencer State Transition Control Registers 0-2 on page D6-495.
• D6.63 TRCSEQRSTEV, Sequencer Reset Control Register on page D6-497.
• D6.64 TRCSEQSTR, Sequencer State Register on page D6-498.
• D6.65 TRCSSCCR0, Single-Shot Comparator Control Register 0 on page D6-499.
• D6.66 TRCSSCSR0, Single-Shot Comparator Status Register 0 on page D6-500.
• D6.67 TRCSTALLCTLR, Stall Control Register on page D6-501.
• D6.68 TRCSTATR, Status Register on page D6-502.
### D6.1 ETM register summary

This section summarizes the ETM trace unit registers.

All ETM trace unit registers are 32-bit wide. The description of each register includes its offset from a base address. The base address is defined by the system integrator when placing the ETM trace unit in the Debug-APB memory map.

The following table lists all of the ETM trace unit registers.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Name</th>
<th>Type</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x04</td>
<td>TRCPRGCTRL</td>
<td>RW</td>
<td>0x00000000</td>
<td>D6.60 TRCPRGCTRL, Programming Control Register on page D6-493</td>
</tr>
<tr>
<td>0x0C</td>
<td>TRCSATR</td>
<td>RO</td>
<td>0x00000003</td>
<td>D6.68 TRCSATR, Status Register on page D6-502</td>
</tr>
<tr>
<td>0x10</td>
<td>TRCCONFIGR</td>
<td>RW</td>
<td>UNK</td>
<td>D6.20 TRCCONFIGR, Trace Configuration Register on page D6-445</td>
</tr>
<tr>
<td>0x18</td>
<td>TRCAUXCTLR</td>
<td>RW</td>
<td>0x00000000</td>
<td>D6.5 TRCAUXCTLR, Auxiliary Control Register on page D6-427</td>
</tr>
<tr>
<td>0x20</td>
<td>TRCEVENTCTLO</td>
<td>RW</td>
<td>UNK</td>
<td>D6.26 TRCEVENTCTLO, Event Control 0 Register on page D6-453</td>
</tr>
<tr>
<td>0x24</td>
<td>TRCEVENTCTL1R</td>
<td>RW</td>
<td>UNK</td>
<td>D6.27 TRCEVENTCTL1R, Event Control 1 Register on page D6-455</td>
</tr>
<tr>
<td>0x2C</td>
<td>TRCSTALLCTLR</td>
<td>RW</td>
<td>UNK</td>
<td>D6.67 TRCSTALLCTLR, Stall Control Register on page D6-501</td>
</tr>
<tr>
<td>0x30</td>
<td>TRCTSCTLR</td>
<td>RW</td>
<td>UNK</td>
<td>D6.71 TRCTSCTLR, Global Timestamp Control Register on page D6-505</td>
</tr>
<tr>
<td>0x34</td>
<td>TRCSYNCPR</td>
<td>RW</td>
<td>UNK</td>
<td>D6.69 TRCSYNCPR, Synchronization Period Register on page D6-503</td>
</tr>
<tr>
<td>0x38</td>
<td>TRCCCCTL</td>
<td>RW</td>
<td>UNK</td>
<td>D6.7 TRCCCCTL, Cycle Count Control Register on page D6-430</td>
</tr>
<tr>
<td>0x3C</td>
<td>TRCBBCTL</td>
<td>RW</td>
<td>UNK</td>
<td>D6.6 TRCBBCTL, Branch Broadcast Control Register on page D6-429</td>
</tr>
<tr>
<td>0x40</td>
<td>TRCTRACEIDR</td>
<td>RW</td>
<td>UNK</td>
<td>D6.70 TRCTRACEIDR, Trace ID Register on page D6-504</td>
</tr>
<tr>
<td>0x80</td>
<td>TRCVICTLR</td>
<td>RW</td>
<td>UNK</td>
<td>D6.72 TRCVICTLR, ViewInst Main Control Register on page D6-506</td>
</tr>
<tr>
<td>0x84</td>
<td>TRCVIIECTRL</td>
<td>RW</td>
<td>UNK</td>
<td>D6.73 TRCVIIECTRL, ViewInst Include-Exclude Control Register on page D6-508</td>
</tr>
<tr>
<td>0x88</td>
<td>TRCVISSCTRL</td>
<td>RW</td>
<td>UNK</td>
<td>D6.74 TRCVISSCTRL, ViewInst Start-Stop Control Register on page D6-509</td>
</tr>
<tr>
<td>0x100</td>
<td>TRCSEQEVR0</td>
<td>RW</td>
<td>UNK</td>
<td>D6.62 TRCSEQEVR0n, Sequencer State Transition Control Registers 0-2 on page D6-495</td>
</tr>
<tr>
<td>0x104</td>
<td>TRCSEQEVR1</td>
<td>RW</td>
<td>UNK</td>
<td>D6.62 TRCSEQEVR1n, Sequencer State Transition Control Registers 0-2 on page D6-495</td>
</tr>
<tr>
<td>0x108</td>
<td>TRCSEQEVR2</td>
<td>RW</td>
<td>UNK</td>
<td>D6.62 TRCSEQEVR2n, Sequencer State Transition Control Registers 0-2 on page D6-495</td>
</tr>
<tr>
<td>0x118</td>
<td>TRCSEQRSTEV</td>
<td>RW</td>
<td>UNK</td>
<td>D6.63 TRCSEQRSTEV, Sequencer Reset Control Register on page D6-497</td>
</tr>
<tr>
<td>0x11C</td>
<td>TRCSEQSTR</td>
<td>RW</td>
<td>UNK</td>
<td>D6.64 TRCSEQSTR, Sequencer State Register on page D6-498</td>
</tr>
<tr>
<td>0x120</td>
<td>TRCEXTINSELR</td>
<td>RW</td>
<td>UNK</td>
<td>D6.28 TRCEXTINSELR, External Input Select Register on page D6-456</td>
</tr>
<tr>
<td>0x140</td>
<td>TRCCNTRLDVR0</td>
<td>RW</td>
<td>UNK</td>
<td>D6.18 TRCCNTRLDVRn, Counter Reload Value Registers 0-1 on page D6-443</td>
</tr>
<tr>
<td>Offset</td>
<td>Name</td>
<td>Type</td>
<td>Reset</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------</td>
<td>------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>0x144</td>
<td>TRCCNTRLDVR1</td>
<td>RW</td>
<td>UNK</td>
<td>D6.18 TRCCNTRLDVRn, Counter Reload Value Registers 0-1 on page D6-443</td>
</tr>
<tr>
<td>0x150</td>
<td>TRCCNTCTRLR0</td>
<td>RW</td>
<td>UNK</td>
<td>D6.16 TRCCNTCTRLR0, Counter Control Register 0 on page D6-439</td>
</tr>
<tr>
<td>0x154</td>
<td>TRCCNTCTRLR1</td>
<td>RW</td>
<td>UNK</td>
<td>D6.17 TRCCNTCTRLR1, Counter Control Register 1 on page D6-441</td>
</tr>
<tr>
<td>0x160</td>
<td>TRCCNTVR0</td>
<td>RW</td>
<td>UNK</td>
<td>D6.19 TRCCNTVRn, Counter Value Registers 0-1 on page D6-444</td>
</tr>
<tr>
<td>0x164</td>
<td>TRCCNTVR1</td>
<td>RW</td>
<td>UNK</td>
<td>D6.19 TRCCNTVRn, Counter Value Registers 0-1 on page D6-444</td>
</tr>
<tr>
<td>0x180</td>
<td>TRCIDR8</td>
<td>RO</td>
<td>0x00000000</td>
<td>D6.35 TRCIDR8, ID Register 8 on page D6-468</td>
</tr>
<tr>
<td>0x184</td>
<td>TRCIDR9</td>
<td>RO</td>
<td>0x00000000</td>
<td>D6.36 TRCIDR9, ID Register 9 on page D6-469</td>
</tr>
<tr>
<td>0x188</td>
<td>TRCIDR10</td>
<td>RO</td>
<td>0x00000000</td>
<td>D6.37 TRCIDR10, ID Register 10 on page D6-470</td>
</tr>
<tr>
<td>0x18C</td>
<td>TRCIDR11</td>
<td>RO</td>
<td>0x00000000</td>
<td>D6.38 TRCIDR11, ID Register 11 on page D6-471</td>
</tr>
<tr>
<td>0x190</td>
<td>TRCIDR12</td>
<td>RO</td>
<td>0x00000000</td>
<td>D6.39 TRCIDR12, ID Register 12 on page D6-472</td>
</tr>
<tr>
<td>0x194</td>
<td>TRCIDR13</td>
<td>RO</td>
<td>0x00000000</td>
<td>D6.40 TRCIDR13, ID Register 13 on page D6-473</td>
</tr>
<tr>
<td>0x1C0</td>
<td>TRCIMSPEC0</td>
<td>RW</td>
<td>0x00000000</td>
<td>D6.41 TRCIMSPEC0, IMPLEMENTATION SPECIFIC Register 0 on page D6-474</td>
</tr>
<tr>
<td>0x1E0</td>
<td>TRCIDR0</td>
<td>RO</td>
<td>0x28000E1</td>
<td>D6.29 TRCIDR0, ID Register 0 on page D6-457</td>
</tr>
<tr>
<td>0x1E4</td>
<td>TRCIDR1</td>
<td>RO</td>
<td>0x4100F422</td>
<td>D6.30 TRCIDR1, ID Register 1 on page D6-459</td>
</tr>
<tr>
<td>0x1E8</td>
<td>TRCIDR2</td>
<td>RO</td>
<td>0x20001088</td>
<td>D6.31 TRCIDR2, ID Register 2 on page D6-460</td>
</tr>
<tr>
<td>0x1EC</td>
<td>TRCIDR3</td>
<td>RO</td>
<td>0x0D7B0004</td>
<td>D6.32 TRCIDR3, ID Register 3 on page D6-462</td>
</tr>
<tr>
<td>0x1F0</td>
<td>TRCIDR4</td>
<td>RO</td>
<td>0x11170004</td>
<td>D6.33 TRCIDR4, ID Register 4 on page D6-464</td>
</tr>
<tr>
<td>0x1F4</td>
<td>TRCIDR5</td>
<td>RO</td>
<td>0x28C7082F</td>
<td>D6.34 TRCIDR5, ID Register 5 on page D6-466</td>
</tr>
<tr>
<td>0x200</td>
<td>TRCRSCTRLRn</td>
<td>RW</td>
<td>UNK</td>
<td>D6.61 TRCRSCTRLRn, Resource Selection Control Registers 2-16 on page D6-494, n is 2, 15</td>
</tr>
<tr>
<td>0x280</td>
<td>TRCSSCCR0</td>
<td>RW</td>
<td>UNK</td>
<td>D6.65 TRCSSCCR0, Single-Shot Comparator Control Register 0 on page D6-499</td>
</tr>
<tr>
<td>0x2A0</td>
<td>TRCSSCSR0</td>
<td>RW</td>
<td>UNK</td>
<td>D6.66 TRCSSCSR0, Single-Shot Comparator Status Register 0 on page D6-500</td>
</tr>
<tr>
<td>0x300</td>
<td>TRCOSLAR</td>
<td>WO</td>
<td>0x00000001</td>
<td>D6.50 TRCOSLAR, OS Lock Access Register on page D6-483</td>
</tr>
<tr>
<td>0x304</td>
<td>TRCOSLSR</td>
<td>RO</td>
<td>0x0000000A</td>
<td>D6.51 TRCOSLSR, OS Lock Status Register on page D6-484</td>
</tr>
<tr>
<td>0x310</td>
<td>TRCPDCR</td>
<td>RW</td>
<td>0x00000000</td>
<td>D6.52 TRCPDCR, Power Down Control Register on page D6-485</td>
</tr>
<tr>
<td>0x314</td>
<td>TRCPDSR</td>
<td>RO</td>
<td>0x00000022</td>
<td>D6.53 TRCPDSR, Power Down Status Register on page D6-486</td>
</tr>
<tr>
<td>0x400</td>
<td>TRCACVRn</td>
<td>RW</td>
<td>UNK</td>
<td>D6.3 TRCACVRn, Address Comparator Value Registers 0-7 on page D6-425</td>
</tr>
<tr>
<td>Offset</td>
<td>Name</td>
<td>Type</td>
<td>Reset</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>---------------------</td>
<td>------</td>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>0x480</td>
<td>TRCACATRn</td>
<td>RW</td>
<td>UNK</td>
<td>D6.2 TRCACATRn, Address Comparator Access Type Registers 0-7 on page D6-423</td>
</tr>
<tr>
<td>0x600</td>
<td>TRCCIDCVR0</td>
<td>RW</td>
<td>UNK</td>
<td>D6.9 TRCCIDCVR0, Context ID Comparator Value Register 0 on page D6-432</td>
</tr>
<tr>
<td>0x640</td>
<td>TRCVMIDCVR0</td>
<td>RW</td>
<td>UNK</td>
<td>D6.76 TRCVMIDCVR0, VMID Comparator Value Register 0 on page D6-511</td>
</tr>
<tr>
<td>0x680</td>
<td>TRCCIDCCTRLR0</td>
<td>RW</td>
<td>UNK</td>
<td>D6.8 TRCCIDCCTRLR0, Context ID Comparator Control Register 0 on page D6-431</td>
</tr>
<tr>
<td>0x688</td>
<td>TRCVMIDCCTRL0</td>
<td>RW</td>
<td>UNK</td>
<td>D6.75 TRCVMIDCCTRL0, Virtual context identifier Comparator Control Register 0 on page D6-510</td>
</tr>
<tr>
<td>0xEE4</td>
<td>TRCITATBIDR</td>
<td>RW</td>
<td>UNK</td>
<td>D6.42 TRCITATBIDR, Integration ATB Identification Register on page D6-475</td>
</tr>
<tr>
<td>0xEEC</td>
<td>TRCITIDATAR</td>
<td>WO</td>
<td>UNK</td>
<td>D6.46 TRCITIDATAR, Integration Instruction ATB Data Register on page D6-479</td>
</tr>
<tr>
<td>0xEF4</td>
<td>TRCITIATBINR</td>
<td>RO</td>
<td>UNK</td>
<td>D6.44 TRCITIATBINR, Integration Instruction ATB In Register on page D6-477</td>
</tr>
<tr>
<td>0xEFC</td>
<td>TRCITIATBOUR</td>
<td>WO</td>
<td>UNK</td>
<td>D6.45 TRCITIATBOUR, Integration Instruction ATB Out Register on page D6-478</td>
</tr>
<tr>
<td>0xF00</td>
<td>TRCITCTRL</td>
<td>RW</td>
<td>0x00000000</td>
<td>D6.43 TRCITCTRL, Integration Mode Control Register on page D6-476</td>
</tr>
<tr>
<td>0xFA0</td>
<td>TRCCLAIMSET</td>
<td>RW</td>
<td>UNK</td>
<td>D6.15 TRCCLAIMSET, Claim Tag Set Register on page D6-438</td>
</tr>
<tr>
<td>0xFA4</td>
<td>TRCCLAIMCLR</td>
<td>RW</td>
<td>0x00000000</td>
<td>D6.14 TRCCLAIMCLR, Claim Tag Clear Register on page D6-437</td>
</tr>
<tr>
<td>0xFA8</td>
<td>TRCDEVAFF0</td>
<td>RO</td>
<td>UNK</td>
<td>D6.21 TRCDEVAFF0, Device Affinity Register 0 on page D6-448</td>
</tr>
<tr>
<td>0xFAC</td>
<td>TRCDEVAFF1</td>
<td>RO</td>
<td>UNK</td>
<td>D6.22 TRCDEVAFF1, Device Affinity Register 1 on page D6-449</td>
</tr>
<tr>
<td>0xFB0</td>
<td>TRCLAR</td>
<td>WO</td>
<td>UNK</td>
<td>D6.47 TRCLAR, Software Lock Access Register on page D6-480</td>
</tr>
<tr>
<td>0xFB4</td>
<td>TRCLSR</td>
<td>RO</td>
<td>0x00000000</td>
<td>D6.48 TRCLSR, Software Lock Status Register on page D6-481</td>
</tr>
<tr>
<td>0xFB8</td>
<td>TRCAUTHSTATUS</td>
<td>RO</td>
<td>UNK</td>
<td>D6.4 TRCAUTHSTATUS, Authentication Status Register on page D6-426</td>
</tr>
<tr>
<td>0xFBC</td>
<td>TRCDEVARCH</td>
<td>RO</td>
<td>0x47724A13</td>
<td>D6.23 TRCDEVARCH, Device Architecture Register on page D6-450</td>
</tr>
<tr>
<td>0xFC8</td>
<td>TRCDEVVID</td>
<td>RO</td>
<td>0x00000000</td>
<td>D6.24 TRCDEVVID, Device ID Register on page D6-451</td>
</tr>
<tr>
<td>0xFCC</td>
<td>TRCDEVTYPE</td>
<td>RO</td>
<td>0x00000013</td>
<td>D6.25 TRCDEVTYPE, Device Type Register on page D6-452</td>
</tr>
<tr>
<td>0xFE0</td>
<td>TRCPIDR0</td>
<td>RO</td>
<td>0x00000000A</td>
<td>D6.54 TRCPIDR0, ETM Peripheral Identification Register 0 on page D6-487</td>
</tr>
<tr>
<td>0xFE4</td>
<td>TRCPIDR1</td>
<td>RO</td>
<td>0x0000000BD</td>
<td>D6.55 TRCPIDR1, ETM Peripheral Identification Register 1 on page D6-488</td>
</tr>
<tr>
<td>0xFE8</td>
<td>TRCPIDR2</td>
<td>RO</td>
<td>0x0200000B</td>
<td>D6.56 TRCPIDR2, ETM Peripheral Identification Register 2 on page D6-489</td>
</tr>
<tr>
<td>0xFE0</td>
<td>TRCPIDR3</td>
<td>RO</td>
<td>0x00000000</td>
<td>D6.57 TRCPIDR3, ETM Peripheral Identification Register 3 on page D6-490</td>
</tr>
<tr>
<td>Offset</td>
<td>Name</td>
<td>Type</td>
<td>Reset</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>---------</td>
<td>------</td>
<td>-------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>0xFD0</td>
<td>TRCPIDR4</td>
<td>RO</td>
<td>0x00000004</td>
<td>D6.58 TRCPIDR4, ETM Peripheral Identification Register 4 on page D6-491</td>
</tr>
<tr>
<td>0xFD4-0xFD4</td>
<td>TRCPIDRn</td>
<td>RO</td>
<td>0x00000000</td>
<td>D6.59 TRCPIDRn, ETM Peripheral Identification Registers 5-7 on page D6-492</td>
</tr>
<tr>
<td>0xFF0</td>
<td>TRCCIDR0</td>
<td>RO</td>
<td>0x0000000D</td>
<td>D6.10 TRCCIDR0, ETM Component Identification Register 0 on page D6-433</td>
</tr>
<tr>
<td>0xFF4</td>
<td>TRCCIDR1</td>
<td>RO</td>
<td>0x00000090</td>
<td>D6.11 TRCCIDR1, ETM Component Identification Register 1 on page D6-434</td>
</tr>
<tr>
<td>0xFF8</td>
<td>TRCCIDR2</td>
<td>RO</td>
<td>0x00000005</td>
<td>D6.12 TRCCIDR2, ETM Component Identification Register 2 on page D6-435</td>
</tr>
<tr>
<td>0xFFC</td>
<td>TRCCIDR3</td>
<td>RO</td>
<td>0x000000B1</td>
<td>D6.13 TRCCIDR3, ETM Component Identification Register 3 on page D6-436</td>
</tr>
</tbody>
</table>
D6.2 TRCACATRn, Address Comparator Access Type Registers 0-7

The TRCACATRn control the access for the corresponding address comparators.

**Bit field descriptions**

The TRCACATRn is a 64-bit register.

![TRCACATRn bit assignments](image)

**RES0, [63:16]**

RES0 Reserved.

**EXLEVEL_NS, [15:12]**

Each bit controls whether a comparison can occur in Non-secure state for the corresponding Exception level. The possible values are:

\[\begin{align*}
0 & \quad \text{The trace unit can perform a comparison, in Non-secure state, for Exception level } n. \\
1 & \quad \text{The trace unit does not perform a comparison, in Non-secure state, for Exception level } n.
\end{align*}\]

The Exception levels are:

- **Bit[12]** Exception level 0.
- **Bit[13]** Exception level 1.
- **Bit[14]** Exception level 2.
- **Bit[15]** Always RES0.

**EXLEVEL_S, [11:8]**

Each bit controls whether a comparison can occur in Secure state for the corresponding Exception level. The possible values are:

\[\begin{align*}
0 & \quad \text{The trace unit can perform a comparison, in Secure state, for Exception level } n. \\
1 & \quad \text{The trace unit does not perform a comparison, in Secure state, for Exception level } n.
\end{align*}\]

The Exception levels are:

- **Bit[8]** Exception level 0.
- **Bit[9]** Exception level 1.
- **Bit[10]** Always RES0.
- **Bit[11]** Exception level 3.

**RES0, [7:4]**

RES0 Reserved.

**CONTEXT TYPE, [3:2]**

Controls whether the trace unit performs a Context ID comparison, a VMID comparison, or both comparisons:
The trace unit does not perform a Context ID comparison.

The trace unit performs a Context ID comparison using the Context ID comparator that the CONTEXT field specifies, and signals a match if both the Context ID comparator matches and the address comparator match.

The trace unit performs a VMID comparison using the VMID comparator that the CONTEXT field specifies, and signals a match if both the VMID comparator and the address comparator match.

The trace unit performs a Context ID comparison and a VMID comparison using the comparators that the CONTEXT field specifies, and signals a match if the Context ID comparator matches, the VMID comparator matches, and the address comparator matches.

**TYPE, [1:0]**

Type of comparison:

0b00 Instruction address, \texttt{RES0}.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The TRCACATRn can be accessed through the external debug interface, offset \texttt{0x480-0x488}.
D6.3 TRCACVRn, Address Comparator Value Registers 0-7

The TRCACVRn indicate the address for the address comparators.

**Bit field descriptions**

The TRCACVRn is a 64-bit register.

![Figure D6-2  TRCACVRn bit assignments](image)

**ADDRESS, [63:0]**

The address value to compare against.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The TRCACVRn can be accessed through the external debug interface, offset 0x400-0x43C.
D6.4 TRCAUTHSTATUS, Authentication Status Register

The TRCAUTHSTATUS indicates the current level of tracing permitted by the system.

**Bit field descriptions**

The TRCAUTHSTATUS is a 64-bit register.

```
+---------------------------------+---+
| 31:0 | 8:7 | 6:5 | 4 | 3 | 2 | 1 | 0 |
  | RES0 | SNID| SID|  |  |  |  |   |
```

RES0, [31:8]

RES0 Reserved.

SNID, [7:6]

Secure Non-invasive Debug:

0b10 Secure Non-invasive Debug implemented but disabled.
0b11 Secure Non-invasive Debug implemented and enabled.

SID, [5:4]

Secure Invasive Debug:

0b00 Secure Invasive Debug is not implemented.

NSNID, [3:2]

Non-secure Non-invasive Debug:

0b10 Non-secure Non-invasive Debug implemented but disabled, NIDEN=0.
0b11 Non-secure Non-invasive Debug implemented and enabled, NIDEN=1.

NSID, [1:0]

Non-secure Invasive Debug:

0b00 Non-secure Invasive Debug is not implemented.

Bit fields and details not provided in this description are architecturally defined. See the *Arm*® *Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The TRCAUTHSTATUS can be accessed through the external debug interface, offset 0xF8.
D6.5 TRCAUXCTLR, Auxiliary Control Register

The TRCAUXCTLR provides implementation-defined configuration and control options.

**Bit field descriptions**

The TRCAUXCTLR is a 32-bit register.

![Figure D6-4 TRCAUXCTLR bit assignments](image)

**RES0, [31:8]**

RES0 Reserved.

**COREIFEN, [7]**

Keep core interface enabled regardless of trace enable register state. The possible values are:

θ Core interface enabled is set by trace enable register state.

1 Enable core interface, regardless of trace enable register state.

**RES0, [6]**

RES0 Reserved.

**AUTHNOFLUSH, [5]**

Do not flush trace on de-assertion of authentication inputs. The possible values are:

θ ETM trace unit FIFO is flushed and ETM trace unit enters idle state when DBGEN or NIDEN is LOW.

1 ETM trace unit FIFO is not flushed and ETM trace unit does not enter idle state when DBGEN or NIDEN is LOW.

When this bit is set to 1, the trace unit behavior deviates from architecturally-specified behavior.

**TSNODELAY, [4]**

Do not delay timestamp insertion based on FIFO depth. The possible values are:

θ Timestamp packets are inserted into FIFO only when trace activity is LOW.

1 Timestamp packets are inserted into FIFO irrespective of trace activity.

**SYNCDELAY, [3]**

Delay periodic synchronization if FIFO is more than half-full. The possible values are:

θ SYNC packets are inserted into FIFO only when trace activity is low.

1 SYNC packets are inserted into FIFO irrespective of trace activity.
OVFLW, [2]

Force overflow if synchronization is not completed when second synchronization becomes due. The possible values are:

0  No FIFO overflow when SYNC packets are delayed.
1  Forces FIFO overflow when SYNC packets are delayed.

When this bit is set to 1, the trace unit behavior deviates from architecturally-specified behavior.

IDLEACK, [1]

Force idle-drain acknowledge high, CPU does not wait for trace to drain before entering WFX state. The possible values are:

0  ETM trace unit idle acknowledge is asserted only when the ETM trace unit is in idle state.
1  ETM trace unit idle acknowledge is asserted irrespective of the ETM trace unit idle state.

When this bit is set to 1, trace unit behavior deviates from architecturally-specified behavior.

AFREADY, [0]

Always respond to AFREADY immediately. Does not have any interaction with FIFO draining, even in WFI state. The possible values are:

0  ETM trace unit AFREADY output is asserted only when the ETM trace unit is in idle state or when all the trace bytes in FIFO before a flush request are output.
1  ETM trace unit AFREADY output is always asserted HIGH. When this bit is set to 1, trace unit behavior deviates from architecturally-specified behavior.

Bit fields and details not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.

The TRCAUXCTLR can be accessed through the external debug interface, offset 0x018.
The TRCBBCTLR controls how branch broadcasting behaves, and allows branch broadcasting to be enabled for certain memory regions.

**Bit field descriptions**

The TRCAUXCTLR is a 32-bit register.

**RES0, [31:9]**

RES0 Reserved.

**MODE, [8]**

Mode bit:

0 Exclude mode. Branch broadcasting is not enabled in the address range that RANGE defines.

If RANGE==0 then branch broadcasting is enabled for the entire memory map.

1 Include mode. Branch broadcasting is enabled in the address range that RANGE defines.

If RANGE==0 then the behavior of the trace unit is constrained unpredictable. That is, the trace unit might or might not consider any instructions to be in a branch broadcast region.

**RANGE, [7:0]**

Address range field.

Selects which address range comparator pairs are in use with branch broadcasting. Each bit represents an address range comparator pair, so bit[n] controls the selection of address range comparator pair n. If bit[n] is:

0 The address range that address range comparator pair n defines, is not selected.

1 The address range that address range comparator pair n defines, is selected.

Bit fields and details not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.

The TRCBBCTLR can be accessed through the external debug interface, offset 0x03C.
D6.7 TRCCCCTLR, Cycle Count Control Register

The TRCCCCTLR sets the threshold value for cycle counting.

**Bit field descriptions**

The TRCCCCTLR is a 32-bit register.

![TRCCCCTLR bit assignments](image)

**RES0, [31:12]**

RES0 Reserved.

**THRESHOLD, [11:0]**

Instruction trace cycle count threshold.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.*

The TRCCCCTLR can be accessed through the external debug interface, offset 0x038.
D6.8 TRCCIDCCTLR0, Context ID Comparator Control Register 0

The TRCCIDCCTLR0 controls the mask value for the context ID comparators.

**Bit field descriptions**

The TRCCIDCCTLR0 is a 32-bit register.

![Figure D6-7 TRCCIDCCTLR0 bit assignments]

**RES0, [31:4]**

RES0  Reserved.

**COMP0, [3:0]**

Controls the mask value that the trace unit applies to TRCCIDCVR0. Each bit in this field corresponds to a byte in TRCCIDCVR0. When a bit is:

0  The trace unit includes the relevant byte in TRCCIDCVR0 when it performs the Context ID comparison.

1  The trace unit ignores the relevant byte in TRCCIDCVR0 when it performs the Context ID comparison.

Bit fields and details not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.

The TRCCIDCCTLR0 can be accessed through the external debug interface, offset 0x680.
D6.9 TRCCIDCVR0, Context ID Comparator Value Register 0

The TRCCIDCVR0 contains a Context ID value.

**Bit field descriptions**

The TRCCIDCVR0 is a 64-bit register.

![TRCCIDCVR0 bit assignments](image)

**RES0, [63:32]**

RES0  Reserved.

**VALUE, [31:0]**

The data value to compare against.

Bit fields and details not provided in this description are architecturally defined. See the *Arm*® *Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The TRCCIDCVR0 can be accessed through the external debug interface, offset 0x600.
D6.10 TRCCIDR0, ETM Component Identification Register 0

The TRCCIDR0 provides information to identify a trace component.

**Bit field descriptions**

The TRCCIDR0 is a 32-bit register.

![Figure D6-9  TRCCIDR0 bit assignments](image)

RES0, [31:8]

RES0 Reserved.

PRMBL_0, [7:0]

0x0D Preamble byte 0.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The TRCCIDR0 can be accessed through the external debug interface, offset 0xFF0.
D6.11 TRCCIDR1, ETM Component Identification Register 1

The TRCCIDR1 provides information to identify a trace component.

**Bit field descriptions**

The TRCCIDR1 is a 32-bit register.

![Figure D6-10 TRCCIDR1 bit assignments](image)

<table>
<thead>
<tr>
<th>Bit Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RES0, [31:8]</td>
<td>Reserved.</td>
</tr>
<tr>
<td>PRMBL_1, [3:0]</td>
<td>Preamble byte 1.</td>
</tr>
</tbody>
</table>

Bit fields and details not provided in this description are architecturally defined. See the *Arm*® Architecture Reference Manual Armv8, for Armv8-A architecture profile.

The TRCCIDR1 can be accessed through the external debug interface, offset 0xFFF4.
D6.12 TRCCIDR2, ETM Component Identification Register 2

The TRCCIDR2 provides information to identify a CTI component.

**Bit field descriptions**

The TRCCIDR2 is a 32-bit register.

<table>
<thead>
<tr>
<th>Bit Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-8</td>
<td>RES0</td>
</tr>
<tr>
<td>7-0</td>
<td>PRMBL_2</td>
</tr>
</tbody>
</table>

Figure D6-11  TRCCIDR2 bit assignments

RES0, [31:8]  

RES0  Reserved.

PRMBL_2, [7:0]  

0x05  Preamble byte 2.

Bit fields and details not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.

The TRCCIDR2 can be accessed through the external debug interface, offset 0xFF8.
D6.13 TRCCIDR3, ETM Component Identification Register 3

The TRCCIDR3 provides information to identify a trace component.

**Bit field descriptions**

The TRCCIDR3 is a 32-bit register.

![TRCCIDR3 bit assignments](image)

- **RES0, [31:8]**
  - RES0: Reserved.
- **PRMBL_3, [7:0]**
  - 0xB1: Preamble byte 3.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The TRCCIDR3 can be accessed through the external debug interface, offset 0xFFC.
D6.14 TRCCLAIMCLR, Claim Tag Clear Register

The TRCCLAIMCLR clears bits in the claim tag and determines the current value of the claim tag.

**Bit field descriptions**

The TRCCLAIMCLR is a 32-bit register.

<table>
<thead>
<tr>
<th>Bit field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RES0</td>
<td>Reserved.</td>
</tr>
<tr>
<td>CLR</td>
<td>On reads, for each bit:</td>
</tr>
<tr>
<td></td>
<td>0: Claim tag bit is not set.</td>
</tr>
<tr>
<td></td>
<td>1: Claim tag bit is set.</td>
</tr>
<tr>
<td></td>
<td>On writes, for each bit:</td>
</tr>
<tr>
<td></td>
<td>0: Has no effect.</td>
</tr>
<tr>
<td></td>
<td>1: Clears the relevant bit of the claim tag.</td>
</tr>
</tbody>
</table>

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual* *Armv8, for Armv8-A architecture profile*.

The TRCCLAIMCLR can be accessed through the external debug interface, offset 0xFA4.
D6.15 TRCCLAIMSET, Claim Tag Set Register

The TRCCLAIMSET sets bits in the claim tag and determines the number of claim tag bits implemented.

Bit field descriptions
The TRCCLAIMSET is a 32-bit register.

![TRCCLAIMSET bit assignments](image)

RES0, [31:4]
RES0  Reserved.

SET, [3:0]
On reads, for each bit:
0  Claim tag bit is not implemented.
1  Claim tag bit is implemented.

On writes, for each bit:
0  Has no effect.
1  Sets the relevant bit of the claim tag.

Bit fields and details not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.

The TRCCLAIMSET can be accessed through the external debug interface, offset 0xFA0.
D6.16 TRCCNTCTLR0, Counter Control Register 0

The TRCCNTCTLR0 controls the counter.

**Bit field descriptions**

The TRCCNTCTLR0 is a 32-bit register.

![Figure D6-15 TRCCNTCTLR0 bit assignments](image)

**RES0, [31:17]**

**RES0** Reserved.

**RLDSELF, [16]**

Defines whether the counter reloads when it reaches zero:

- **0** The counter does not reload when it reaches zero. The counter only reloads based on RLDTYPE and RLDSEL.
- **1** The counter reloads when it reaches zero and the resource selected by CNTTYPE and CNTSEL is also active. The counter also reloads based on RLDTYPE and RLDSEL.

**RLDTYPE, [15]**

Selects the resource type for the reload:

- **0** Single selected resource.
- **1** Boolean combined resource pair.

**RES0, [14:12]**

**RES0** Reserved.

**RLDSEL, [11:8]**

Selects the resource number, based on the value of RLDTYPE:

- When RLDTYPE is 0, selects a single selected resource from 0-15 defined by bits[3:0].
- When RLDTYPE is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

**CNTTYPE, [7]**

Selects the resource type for the counter:

- **0** Single selected resource.
- **1** Boolean combined resource pair.

**RES0, [6:4]**

**RES0** Reserved.
CNTSEL, [3:0]

Selects the resource number, based on the value of CNTTYPE:

When CNTTYPE is 0, selects a single selected resource from 0-15 defined by bits[3:0].

When CNTTYPE is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

Bit fields and details not provided in this description are architecturally defined. See the *Arm* Architecture Reference Manual Armv8, for Armv8-A architecture profile.

The TRCCNTCTLR0 can be accessed through the external debug interface, offset 0x150.
D6.17 TRCCNTCTLR1, Counter Control Register 1

The TRCCNTCTLR1 controls the counter.

**Bit field descriptions**

The TRCCNTCTLR1 is a 32-bit register.

![Figure D6-16 TRCCNTCTLR1 bit assignments](image)

**RES0, [31:18]**

RES0 Reserved.

**CNTCHAIN, [17]**

Defines whether the counter decrements when the counter reloads. This enables two counters to be used in combination to provide a larger counter:

0 The counter operates independently from the counter. The counter only decrements based on CNTTYPE and CNTSEL.
1 The counter decrements when the counter reloads. The counter also decrements when the resource selected by CNTTYPE and CNTSEL is active.

**RLDSELF, [16]**

Defines whether the counter reloads when it reaches zero:

0 The counter does not reload when it reaches zero. The counter only reloads based on RLDTYPE and RLDSEL.
1 The counter reloads when it is zero and the resource selected by CNTTYPE and CNTSEL is also active. The counter also reloads based on RLDTYPE and RLDSEL.

**RLDTYPE, [15]**

Selects the resource type for the reload:

0 Single selected resource.
1 Boolean combined resource pair.

**RES0, [14:12]**

RES0 Reserved.

**RLDSEL, [11:8]**

Selects the resource number, based on the value of RLDTYPE:

When RLDTYPE is 0, selects a single selected resource from 0-15 defined by bits[3:0].
When RLDTYPE is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

**CNTTYPE, [7]**

Selects the resource type for the counter:
0  Single selected resource.
1  Boolean combined resource pair.

RES0, [6:4]
RES0  Reserved.

CNTSEL, [3:0]
Selects the resource number, based on the value of CNTTYPE:
When CNTTYPE is 0, selects a single selected resource from 0-15 defined by bits[3:0].
When CNTTYPE is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

Bit fields and details not provided in this description are architecturally defined. See the *Arm*®

The TRCCNTCTRLR1 can be accessed through the external debug interface, offset 0x154.
D6.18 TRCCNTRLDVRn, Counter Reload Value Registers 0-1

The TRCCNTRLDVRn define the reload value for the counter.

**Bit field descriptions**

The TRCCNTRLDVRn is a 32-bit register.

![Figure D6-17 TRCCNTRLDVRn bit assignments](image)

- **RES0**, [31:16]
  - RES0: Reserved.

- **VALUE**, [15:0]
  - Defines the reload value for the counter. This value is loaded into the counter each time the reload event occurs.

Bit fields and details not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.

The TRCCNTRLDVRn registers can be accessed through the external debug interface, offsets:

- **TRCCNTRLDVR0**
  - 0x140.

- **TRCCNTRLDVR1**
  - 0x144.
**D6.19 TRCCNTVRn, Counter Value Registers 0-1**

The TRCCNTVRn contain the current counter value.

**Bit field descriptions**

The TRCCNTVRn is a 32-bit register.

![Figure D6-18 TRCCNTVRn bit assignments](image)

**RES0, [31:16]**

RES0 Reserved.

**VALUE, [15:0]**

Contains the current counter value.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The TRCCNTRLDVRn registers can be accessed through the external debug interface, offsets:

**TRCCNTVR0**

0x160.

**TRCCNTVR1**

0x164.
D6.20 TRCCONFIGR, Trace Configuration Register

The TRCCONFIGR controls the tracing options.

Bit field descriptions

The TRCCONFIGR is a 32-bit register.

![Figure D6-19 TRCCONFIGR bit assignments](image)

RES0, [31:18]

Reserved.

DV, [17]

Enables data value tracing. The possible values are:

0  Disables data value tracing.
1  Enables data value tracing.

DA, [16]

Enables data address tracing. The possible values are:

0  Disables data address tracing.
1  Enables data address tracing.

VMIDOPT, [15]

Configures the Virtual context identifier value used by the trace unit, both for trace generation and in the Virtual context identifier comparators. The possible values are:

0b0  VTTBR_EL2.VMID is used. If the trace unit supports a Virtual context identifier larger than the VTTBR_EL2.VMID, the upper unused bits are always zero. If the trace unit supports a Virtual context identifier larger than 8 bits and if the VTCR_EL2.VS bit forces use of an 8-bit Virtual context identifier, bits [15:8] of the trace unit Virtual context identifier are always zero.
0b1  CONTEXTIDR_EL2 is used. TRCIDR2.VMIDOPT indicates whether this field is implemented.

QE, [14:13]

Enables Q element. The possible values are:

0b00  Q elements are disabled.
0b01  Q elements with instruction counts are disabled. Q elements without instruction counts are disabled.
0b10  Reserved.
0b11  Q elements with and without instruction counts are enabled.
RS, [12]
Enables the return stack. The possible values are:
0  Disables the return stack.
1  Enables the return stack.

Enables global timestamp tracing. The possible values are:
0  Disables global timestamp tracing.
1  Enables global timestamp tracing.

COND, [10:8]
Enables conditional instruction tracing. The possible values are:
0b000  Conditional instruction tracing is disabled.
0b001  Conditional load instructions are traced.
0b010  Conditional store instructions are traced.
0b011  Conditional load and store instructions are traced.
0b111  All conditional instructions are traced.

VMID, [7]
Enables VMID tracing. The possible values are:
0  Disables VMID tracing.
1  Enables VMID tracing.

CID, [6]
Enables context ID tracing. The possible values are:
0  Disables context ID tracing.
1  Enables context ID tracing.

RES0, [5]
RES0  Reserved.

CCI, [4]
Enables cycle counting instruction trace. The possible values are:
0  Disables cycle counting instruction trace.
1  Enables cycle counting instruction trace.

BB, [3]
Enables branch broadcast mode. The possible values are:
0  Disables branch broadcast mode.
1  Enables branch broadcast mode.

INSTP0, [2:1]
Controls whether load and store instructions are traced as P0 instructions. The possible values are:
0b00  Load and store instructions are not traced as P0 instructions.
0b01  Load instructions are traced as P0 instructions.
0b10  Store instructions are traced as P0 instructions.
0b11  Load and store instructions are traced as P0 instructions.

RES1, [0]
RES1  Reserved.

Bit fields and details not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.

The TRCCONFIGR can be accessed through the external debug interface, offset 0x010.
D6.21 TRCDEVAFF0, Device Affinity Register 0

The TRCDEVAFF0 provides an additional core identification mechanism for scheduling purposes in a cluster. TRCDEVAFF0 is a read-only copy of MPIDR accessible from the external debug interface.

**Bit field descriptions**

The TRCDEVAFF0 is a 32-bit register and is a copy of the MPIDR register. See *B1.65 MPIDR_EL1, Multiprocessor Affinity Register, EL1* on page B1-218 for full bit field descriptions.
D6.22 TRCDEVAFF1, Device Affinity Register 1

The TRCDEVAFF1 is a read-only copy of MPIDR_EL1[63:32] as seen from EL3, unaffected by VMPIDR_EL2.
D6.23 TRCDEVARCH, Device Architecture Register

The TRCDEVARCH identifies the ETM trace unit as an ETMv4 component.

**Bit field descriptions**

The TRCDEVARCH is a 32-bit register.

<table>
<thead>
<tr>
<th>31</th>
<th>21</th>
<th>19</th>
<th>16</th>
<th>15</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARCHITECT</td>
<td>REVISION</td>
<td>ARCHID</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**ARCHITECT, [31:21]**

Defines the architect of the component:

- 0x4  Arm JEP continuation.
- 0x3B  Arm JEP 106 code.

**PRESENT, [20]**

Indicates the presence of this register:

- 0b1  Register is present.

**REVISION, [19:16]**

Architecture revision:

- 0x2  Architecture revision 2.

**ARCHID, [15:0]**

Architecture ID:

- 0x4A13  ETMv4 component.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The TRCDEVARCH can be accessed through the external debug interface, offset 0xFBC.
D6.24 TRCDEVID, Device ID Register

The TRCDEVID indicates the capabilities of the ETM trace unit.

**Bit field descriptions**

The TRCDEVID is a 32-bit register.

```
| 31 |    |    |    |    |    | 0 |
|------------------|------------------|
| [DEVID]          | [DEVID]          |
|                  |                  |
|                  |                  |
| DEVID            | DEVID            |
```

Figure D6-21 TRCDEVID bit assignments

**DEVID, [31:0]**

RAZ. There are no component-defined capabilities.

Bit fields and details not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.

The TRCDEVID can be accessed through the external debug interface, offset 0xFC8.
D6.25 TRCDEVTYPE, Device Type Register

The TRCDEVTYPE indicates the type of the component.

Bit field descriptions

The TRCDEVTYPE is a 32-bit register.

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

RES0

RES0  Reserved.

SUB, [7:4]

The sub-type of the component:

0b0001  Core trace.

MAJOR, [3:0]

The main type of the component:

0b0011  Trace source.

Bit fields and details not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.

The TRCDEVTYPE can be accessed through the external debug interface, offset 0xFCC.
D6 TRCEVENTCTL0R, Event Control 0 Register

The TRCEVENTCTL0R controls the tracing of events in the trace stream. The events also drive the external outputs from the ETM trace unit. The events are selected from the Resource Selectors.

**Bit field descriptions**

The TRCEVENTCTL0R is a 32-bit register.

![Figure D6-23 TRCEVENTCTL0R bit assignments](image)

**TYPE3, [31]**
Selects the resource type for trace event 3:

- 0  Single selected resource.
- 1  Boolean combined resource pair.

**RES0, [30:28]**
Reserved.

**SEL3, [27:24]**
Selects the resource number, based on the value of TYPE3:

When TYPE3 is 0, selects a single selected resource from 0-15 defined by bits[3:0].

When TYPE3 is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

**TYPE2, [23]**
Selects the resource type for trace event 2:

- 0  Single selected resource.
- 1  Boolean combined resource pair.

**RES0, [22:20]**
Reserved.

**SEL2, [19:16]**
Selects the resource number, based on the value of TYPE2:

When TYPE2 is 0, selects a single selected resource from 0-15 defined by bits[3:0].

When TYPE2 is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

**TYPE1, [15]**
Selects the resource type for trace event 1:

- 0  Single selected resource.
- 1  Boolean combined resource pair.

**RES0, [14:12]**
RES0 Reserved.

SEL1, [11:8]
Selects the resource number, based on the value of TYPE1:
When TYPE1 is 0, selects a single selected resource from 0-15 defined by bits[3:0].
When TYPE1 is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

TYPE0, [7]
Selects the resource type for trace event 0:
0 Single selected resource.
1 Boolean combined resource pair.

RES0, [6:4]
RES0 Reserved.

SEL0, [3:0]
Selects the resource number, based on the value of TYPE0:
When TYPE0 is 0, selects a single selected resource from 0-15 defined by bits[3:0].
When TYPE0 is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

Bit fields and details not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.

The TRCEVENTCTL0R can be accessed through the external debug interface, offset 0x020.
D6.27 TRCEVENTCTL1R, Event Control 1 Register

The TRCEVENTCTL1R controls the behavior of the events that TRCEVENTCTL0R selects.

Bit field descriptions

The TRCEVENTCTL1R is a 32-bit register.

<table>
<thead>
<tr>
<th>Bit Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[31:13]</td>
<td>RES0, Reserved.</td>
</tr>
<tr>
<td>[12]</td>
<td>LPOVERRIDE, Low-power state behavior override:</td>
</tr>
<tr>
<td></td>
<td>0: Low-power state behavior unaffected.</td>
</tr>
<tr>
<td></td>
<td>1: Low-power state behavior overridden. The resources and Event trace generation are unaffected by entry to a low-power state.</td>
</tr>
<tr>
<td>[11]</td>
<td>ATB, ATB trigger enable:</td>
</tr>
<tr>
<td></td>
<td>0: ATB trigger disabled.</td>
</tr>
<tr>
<td></td>
<td>1: ATB trigger enabled.</td>
</tr>
<tr>
<td>[10:4]</td>
<td>RES0, Reserved.</td>
</tr>
<tr>
<td>[3:0]</td>
<td>EN, One bit per event, to enable generation of an event element in the instruction trace stream when the selected event occurs:</td>
</tr>
<tr>
<td></td>
<td>0: Event does not cause an event element.</td>
</tr>
<tr>
<td></td>
<td>1: Event causes an event element.</td>
</tr>
</tbody>
</table>

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The TRCEVENTCTL1R can be accessed through the external debug interface, offset 0x024.
D6.28 TRCEXTINSELR, External Input Select Register

The TRCEXTINSELR controls the selectors that choose an external input as a resource in the ETM trace unit. You can use the Resource Selectors to access these external input resources.

**Bit field descriptions**

The TRCEXTINSELR is a 32-bit register.

**TRCEXTINSELR bit assignments**

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>16</th>
<th>15</th>
<th>14</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEL3</td>
<td>SEL2</td>
<td>SEL1</td>
<td>SEL0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**RES0, [31:30]**

RES0 Reserved.

**SEL3, [29:24]**

Selects an event from the external input bus for External Input Resource 3.

**RES0, [23:22]**

RES0 Reserved.

**SEL2, [21:16]**

Selects an event from the external input bus for External Input Resource 2.

**RES0, [15:14]**

RES0 Reserved.

**SEL1, [13:8]**

Selects an event from the external input bus for External Input Resource 1.

**RES0, [7:6]**

RES0 Reserved.

**SEL0, [5:0]**

Selects an event from the external input bus for External Input Resource 0.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The TRCEXTINSELR can be accessed through the external debug interface, offset 0x120.
D6.29  TRCIDR0, ID Register 0

The TRCIDR0 returns the tracing capabilities of the ETM trace unit.

**Bit field descriptions**

The TRCIDR0 is a 32-bit register.

![Bit field assignments](image)

Figure D6-25  TRCIDR0 bit assignments

**RES0, [31:30]**

Reserved.

**COMMOPT, [29]**

Indicates the meaning of the commit field in some packets:

-   1 Commit mode 1.

**TSSIZE, [28:24]**

Global timestamp size field:

-   0b01000 Implementation supports a maximum global timestamp of 64 bits.

**RES0, [23:17]**

Reserved.

**QSUPP, [16:15]**

Indicates Q element support:

-   0b0 Q elements not supported.

**QFILT, [14]**

Indicates Q element filtering support:

-   0b0 Q element filtering not supported.

**CONDTYPE, [13:12]**

Indicates how conditional results are traced:

-   0b0 Conditional trace not supported.

**NUMEVENT, [11:10]**

Number of events supported in the trace, minus 1:

-   0b11 Four events supported.

**RETSSTACK, [9]**
Return stack support:
1 Return stack implemented.

RES0, [8]
RES0 Reserved.

TRCCCI, [7]
Support for cycle counting in the instruction trace:
1 Cycle counting in the instruction trace is implemented.

TRCCOND, [6]
Support for conditional instruction tracing:
0 Conditional instruction tracing is not supported.

TRCBB, [5]
Support for branch broadcast tracing:
1 Branch broadcast tracing is implemented.

TRCDATA, [4:3]
Conditional tracing field:
0b00 Tracing of data addresses and data values is not implemented.

INSTP0, [2:1]
P0 tracing support field:
0b00 Tracing of load and store instructions as P0 elements is not supported.

RES1, [0]
RES1 Reserved.

Bit fields and details not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.

The TRCIDR0 can be accessed through the external debug interface, offset 0x1E0.
D6.30  TRCIDR1, ID Register 1

The TRCIDR1 returns the base architecture of the trace unit.

**Bit field descriptions**

The TRCIDR1 is a 32-bit register.

![TRCIDR1 bit assignments](image)

**DESIGNER, [31:24]**

Indicates which company designed the trace unit:

0x41  Arm.

**RES0, [23:16]**

RES0  Reserved.

**RES1, [15:12]**

RES1  Reserved.

**TRCARCHMAJ, [11:8]**

Major trace unit architecture version number:

0b0100  ETMv4.

**TRCARCHMIN, [7:4]**

Minor trace unit architecture version number:

0x2  ETMv4.2

**REVISION, [3:0]**

Trace unit implementation revision number:

2  ETM revision.

Bit fields and details not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.

The TRCIDR1 can be accessed through the external debug interface, offset 0x1E4.
D6.31 TRCIDR2, ID Register 2

The TRCIDR2 returns the maximum size of six parameters in the trace unit.

The parameters are:
• Cycle counter.
• Data value.
• Data address.
• VMID.
• Context ID.
• Instruction address.

Bit field descriptions

The TRCIDR2 is a 32-bit register.

<table>
<thead>
<tr>
<th>31</th>
<th>29</th>
<th>28</th>
<th>25</th>
<th>24</th>
<th>20</th>
<th>19</th>
<th>15</th>
<th>14</th>
<th>10</th>
<th>9</th>
<th>5</th>
<th>4</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>RES0</td>
<td>VMIDOPT</td>
<td>CCSIZE</td>
<td>DVSIZE</td>
<td>DASIZE</td>
<td>VMIDSIZE</td>
<td>CIDSIZE</td>
<td>IASIZE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

RES0, [31]
RES0 Reserved.

VMIDOPT, [30:29]
Indicates the options for observing the Virtual context identifier:
0x1 VMIDOPT is implemented.

CCSIZE, [28:25]
Size of the cycle counter in bits minus 12:
0x0 The cycle counter is 12 bits in length.

DVSIZE, [24:20]
Data value size in bytes:
0x0 Data value tracing is not implemented.

DASIZE, [19:15]
Data address size in bytes:
0x0 Data address tracing is not implemented.

VMIDSIZE, [14:10]
Virtual Machine ID size:
0x4 Maximum of 32-bit Virtual Machine ID size.

CIDSIZE, [9:5]
Context ID size in bytes:
0x4  Maximum of 32-bit Context ID size.

**IASIZE, [4:0]**

Instruction address size in bytes:

0x8  Maximum of 64-bit address size.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The TRCIDR2 can be accessed through the external debug interface, offset 0x1E8.
The TRCIDR3 indicates:

- Whether TRCVICTLR is supported.
- The number of cores available for tracing.
- If an Exception level supports instruction tracing.
- The minimum threshold value for instruction trace cycle counting.
- Whether the synchronization period is fixed.
- Whether TRCSTALLCTLR is supported and if so whether it supports trace overflow prevention and supports stall control of the core.

**Bit field descriptions**

The TRCIDR3 is a 32-bit register.

![TRCIDR3 Bit Assignments](image)

**NOOVERFLOW, [31]**

Indicates whether TRCSTALLCTLR.NOOVERFLOW is implemented:

- 0: TRCSTALLCTLR.NOOVERFLOW is not implemented.

**NUMPROC, [30:28]**

Indicates the number of cores available for tracing:

- 0b000: The trace unit can trace one core, ETM trace unit sharing not supported.

**SYSSTALL, [27]**

Indicates whether stall control is implemented:

- 1: The system supports core stall control.

**STALLCTL, [26]**

Indicates whether TRCSTALLCTLR is implemented:

- 1: TRCSTALLCTLR is implemented.

This field is used in conjunction with SYSSTALL.

**SYNCPR, [25]**

Indicates whether there is a fixed synchronization period:
TRCSYNCPR is read-write so software can change the synchronization period.

**TRCERR, [24]**  
Indicates whether TRCVICTLR.TRCCERR is implemented:  
1 TRCVICTLR.TRCCERR is implemented.

**EXLEVEL_NS, [23:20]**  
Each bit controls whether instruction tracing in Non-secure state is implemented for the corresponding Exception level:  
0b0111 Instruction tracing is implemented for Non-secure EL0, EL1, and EL2 Exception levels.

**EXLEVEL_S, [19:16]**  
Each bit controls whether instruction tracing in Secure state is implemented for the corresponding Exception level:  
0b1011 Instruction tracing is implemented for Secure EL0, EL1, and EL3 Exception levels.

**RES0, [15:12]**  
RES0 Reserved.

**CCITMIN, [11:0]**  
The minimum value that can be programmed in TRCCCTRL.THRESHOLD:  
0x004 Instruction trace cycle counting minimum threshold is 4.

Bit fields and details not provided in this description are architecturally defined. See the *Arm* Architecture Reference Manual Armv8, for Armv8-A architecture profile.

The TRCIDR3 can be accessed through the external debug interface, offset 0x1EC.
**D6.33 TRCIDR4, ID Register 4**

The TRCIDR4 indicates the resources available in the ETM trace unit.

**Bit field descriptions**

The TRCIDR4 is a 32-bit register.

![Figure D6-29 TRCIDR4 bit assignments](image)

**NUMVMIDC, [31:28]**

Indicates the number of VMID comparators available for tracing:

0x1 One VMID comparator is available.

**NUMCIDC, [27:24]**

Indicates the number of CID comparators available for tracing:

0x1 One Context ID comparator is available.

**NUMSSCC, [23:20]**

Indicates the number of single-shot comparator controls available for tracing:

0x1 One single-shot comparator control is available.

**NUMRSPAIRS, [19:16]**

Indicates the number of resource selection pairs available for tracing:

0x7 Eight resource selection pairs are available.

**NUMPC, [15:12]**

Indicates the number of core comparator inputs available for tracing:

0x0 Core comparator inputs are not implemented.

**RES0, [11:9]**

RES0 Reserved.

**SUPPDAC, [8]**

Indicates whether the implementation supports data address comparisons: This value is:

0 Data address comparisons are not implemented.

**NUMDVC, [7:4]**

Indicates the number of data value comparators available for tracing:

0x0 Data value comparators not implemented.

**NUMACPAIRS, [3:0]**

Indicates the number of address comparator pairs available for tracing:
Four address comparator pairs are implemented.

Bit fields and details not provided in this description are architecturally defined. See the *Arm*® *Architecture Reference Manual Armv8*, for Armv8-A architecture profile.

The TRCIDR4 can be accessed through the external debug interface, offset 0x1F0.
**D6.34 TRCIDR5, ID Register 5**

The TRCIDR5 returns how many resources the trace unit supports.

**Bit field descriptions**

![Bit field assignments](image)

**REDFUNCNTR, [31]**

Reduced Function Counter implemented:

0 Reduced Function Counter not implemented.

**NUMCNTR, [30:28]**

Number of counters implemented:

0b010 Two counters implemented.

**NUMSEQSTATE, [27:25]**

Number of sequencer states implemented:

0b100 Four sequencer states implemented.

**RES0, [24]**

RES0 Reserved.

**LPOVERRIDE, [23]**

Low-power state override support:

1 Low-power state override support implemented.

**ATBTRIG, [22]**

ATB trigger support:

1 ATB trigger support implemented.

**TRACEIDSIZE, [21:16]**

Number of bits of trace ID:

0x7 Seven-bit trace ID implemented.

**RES0, [15:12]**

RES0 Reserved.

**NUMEXTINSEL, [11:9]**
Number of external input selectors implemented:
0b100 Four external input selectors implemented.

NUMEXTIN, [8:0]
Number of external inputs implemented:
0x2F 47 external inputs implemented.

Bit fields and details not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.

The TRCIDR5 can be accessed through the external debug interface, offset 0x1F4.
D6.35  TRCIDR8, ID Register 8

The TRCIDR8 returns the maximum speculation depth of the instruction trace stream.

Bit field descriptions

The TRCIDR8 is a 32-bit register.

![Figure D6-31 TRCIDR8 bit assignments](image)

MAXSPEC, [31:0]

The maximum number of P0 elements in the trace stream that can be speculative at any time.

\[ \theta \]  Maximum speculation depth of the instruction trace stream.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The TRCIDR8 can be accessed through the external debug interface, offset 0x180.
D6.36   TRCIDR9, ID Register 9

The TRCIDR9 returns the number of P0 right-hand keys that the trace unit can use.

**Bit field descriptions**

The TRCIDR9 is a 32-bit register.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | NUMP0KEY |

**Figure D6-32  TRCIDR9 bit assignments**

**NUMP0KEY, [31:0]**

The number of P0 right-hand keys that the trace unit can use.

0 Number of P0 right-hand keys.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The TRCIDR9 can be accessed through the external debug interface, offset 0x184.
D6.37 TRCIDR10, ID Register 10

The TRCIDR10 returns the number of P1 right-hand keys that the trace unit can use.

**Bit field descriptions**

The TRCIDR10 is a 32-bit register.

<table>
<thead>
<tr>
<th>31</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMP1KEY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**NUMP1KEY, [31:0]**

The number of P1 right-hand keys that the trace unit can use.

0 Number of P1 right-hand keys.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The TRCIDR10 can be accessed through the external debug interface, offset 0x188.
**D6.38 TRCIDR11, ID Register 11**

The TRCIDR11 returns the number of special P1 right-hand keys that the trace unit can use.

**Bit field descriptions**

The TRCIDR11 is a 32-bit register.

<table>
<thead>
<tr>
<th>Bit Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMP1SPC</td>
<td>Number of special P1 right-hand keys.</td>
</tr>
</tbody>
</table>

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The TRCIDR11 can be accessed through the external debug interface, offset 0x18C.
The TRCIDR12 returns the number of conditional instruction right-hand keys that the trace unit can use.

**Bit field descriptions**

The TRCIDR12 is a 32-bit register.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>NUMCONDKEY</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure D6-35 TRCIDR12 bit assignments**

**NUMCONDKEY, [31:0]**

The number of conditional instruction right-hand keys that the trace unit can use, including normal and special keys.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>NUMCONDKEY</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Number of conditional instruction right-hand keys.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The TRCIDR12 can be accessed through the external debug interface, offset 0x190.
D6.40 TRCIDR13, ID Register 13

The TRCIDR13 returns the number of special conditional instruction right-hand keys that the trace unit can use.

**Bit field descriptions**

The TRCIDR11 is a 32-bit register.

![Figure D6-36  TRCIDR13 bit assignments](image)

**NUMCONDSPC, [31:0]**

The number of special conditional instruction right-hand keys that the trace unit can use, including normal and special keys.

0 Number of special conditional instruction right-hand keys.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The TRCIDR13 can be accessed through the external debug interface, offset 0x194.
D6.41 TRCIMSPEC0, IMPLEMENTATION SPECIFIC Register 0

The TRCIMSPEC0 shows the presence of any IMPLEMENTATION SPECIFIC features, and enables any features that are provided.

**Bit field descriptions**

The TRCIMSPEC0 is a 32-bit register.

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

RES0, [31:4]

RES0 Reserved.

SUPPORT, [3:0]

0 No IMPLEMENTATION SPECIFIC extensions are supported.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The TRCIMSPEC0 can be accessed through the external debug interface, offset 0x1C0.
D6.42 TRCITATBIDR, Integration ATB Identification Register

The TRCITATBIDR sets the state of output pins, mentioned in the bit descriptions in this section.

**Bit field descriptions**

The TRCITATBIDR is a 32-bit register.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ID</td>
</tr>
</tbody>
</table>

**[31:7]**

Reserved. Read undefined.

**ID, [6:0]**

Drives the $\text{ATIDMn[6:0]}$ output pins.

When a bit is set to 0, the corresponding output pin is LOW.

When a bit is set to 1, the corresponding output pin is HIGH.

The TRCITATBIDR bit values correspond to the physical state of the output pins.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The TRCITATBIDR can be accessed through the external debug interface, offset $0x\text{EE4}$. 
D6.43 TRCITCTRL, Integration Mode Control Register

The TRCITCTRL enables topology detection or integration testing, by putting the ETM trace unit into integration mode.

Bit field descriptions

The TRCITCTRL is a 32-bit register.

![Figure D6-39 TRCITCTRL bit assignments](image)

RES0, [31:1]

RES0: Reserved.

IME, [0]

Integration mode enable bit. The possible values are:

0: The trace unit is not in integration mode.
1: The trace unit is in integration mode. This mode enables:
   • A debug agent to perform topology detection.
   • SoC test software to perform integration testing.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The TRCITCTRL can be accessed through the external debug interface, offset 0xF00.
D6.44 TRCITIATBINR, Integration Instruction ATB In Register

The TRCITIATBINR reads the state of the input pins described in this section.

**Bit field descriptions**

The TRCITIATBINR is a 32-bit register.

![Figure D6-40 TRCITIATBINR bit assignments](image)

For all non-reserved bits:
- When an input pin is LOW, the corresponding register bit is 0.
- When an input pin is HIGH, the corresponding register bit is 1.
- The TRCITIATBINR bit values always correspond to the physical state of the input pins.

[31:2]
Reserved. Read undefined.

**AFVALIDM, [1]**
Returns the value of the AFVALIDMn input pin.

**ATREADYM, [0]**
Returns the value of the ATREADYMn input pin.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The TRCITIATBINR can be accessed through the external debug interface, offset 0xEF4.
D6.45 TRCITIATBOUTR, Integration Instruction ATB Out Register

The TRCITIATBOUTR sets the state of the output pins mentioned in the bit descriptions in this section.

Bit field descriptions

The TRCITIATBOUTR is a 32-bit register.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


[7:2] Reserved. Read undefined.

AFREADY, [1] Drives the AFREADYMn output pin.

ATVALID, [0] Drives the ATVALIDMn output pin.

For all non-reserved bits:
- When a bit is set to 0, the corresponding output pin is LOW.
- When a bit is set to 1, the corresponding output pin is HIGH.
- The TRCITIATBOUTR bit values always correspond to the physical state of the output pins.

Bit fields and details not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.

The TRCITIATBOUTR can be accessed through the external debug interface, offset 0xEFC.
### D6.46 TRCITIDATAR, Integration Instruction ATB Data Register

The TRCITIDATAR sets the state of the ATDATAM<sub>n</sub> output pins shown in the TRCITIDATAR bit assignments table.

#### Bit field descriptions

The TRCITIDATAR is a 32-bit register.

![Figure D6-42 TRCITIDATAR bit assignments](image)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RES0, [31:5]</td>
<td>Reserved.</td>
</tr>
<tr>
<td>ATDATAM[0], [0]</td>
<td>Drives the ATDATAM[0] output.</td>
</tr>
</tbody>
</table>

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The TRCITIDATAR can be accessed through the external debug interface, offset 0xEEC.

---

<sup>b</sup> When a bit is set to 0, the corresponding output pin is LOW. When a bit is set to 1, the corresponding output pin is HIGH. The TRCITIDATAR bit values correspond to the physical state of the output pins.
D6.47 TRCLAR, Software Lock Access Register

The TRCLAR controls access to registers using the memory-mapped interface, when PADDRDBG31 is LOW.

When the software lock is set, write accesses using the memory-mapped interface to all ETM trace unit registers are IGNORED.

When the software lock is set, read accesses of TRCPDSR do not change the TRCPDSR.STICKYPD bit. Read accesses of all other registers are not affected.

**Bit field descriptions**

The TRCLAR is a 32-bit register.

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

**Figure D6-43** TRCLAR bit assignments

**RAZ/WI, [31:0]**

Read-As-Zero, write ignore.

Bit fields and details not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.

The TRCLAR can be accessed through the external debug interface, offset 0xFB0.
D6.48 TRCLSR, Software Lock Status Register

The TRCLSR determines whether the software lock is implemented, and indicates the current status of the software lock.

**Bit field descriptions**

The TRCLSR is a 32-bit register.

![Figure D6-44 TRCLSR bit assignments](image)

**RAZ/WI, [31:0]**

Read-As-Zero, write ignore.

Bit fields and details not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual, Armv8, for Armv8-A architecture profile.

The TRCLSR can be accessed through the external debug interface, offset 0xFB4.
D6.49 TRCCNTVRn, Counter Value Registers 0-1

The TRCCNTVRn contains the current counter value.

**Bit field descriptions**

The TRCCNTVRn is a 32-bit register.

![Figure D6-45 TRCCNTVRn bit assignments](image)

**RES0, [31:16]**

Res0: Reserved.

**VALUE, [15:0]**

Contains the current counter value.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The TRCCNTVRn registers can be accessed through the external debug interface, offsets:

- TRCCNTVR0: 0x160.
- TRCCNTVR1: 0x164.
D6.50 **TRCOSLAR, OS Lock Access Register**

The TRCOSLAR sets and clears the OS Lock, to lock out external debugger accesses to the ETM trace unit registers.

**Bit field descriptions**

The TRCOSLAR is a 32-bit register.

![Figure D6-46  TRCOSLAR bit assignments](image)

- **RES0, [31:1]**
  - **RES0**  Reserved.

- **OSLK, [0]**
  - OS Lock key value:
    - 0  Unlock the OS Lock.
    - 1  Lock the OS Lock.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The TRCOSLAR can be accessed through the external debug interface, offset 0x300.
D6.51 TRCOSLSR, OS Lock Status Register

The TRCOSLSR returns the status of the OS Lock.

**Bit field descriptions**

The TRCOSLSR is a 32-bit register.

![TRCOSLSR bit assignments](image)

RES0, [31:4]

RES0 Reserved.

OSLM[1], [3]

OS Lock model [1] bit. This bit is combined with OSLM[0] to form a two-bit field that indicates the OS Lock model is implemented.

The value of this field is always 0b10, indicating that the OS Lock is implemented.

nTT, [2]

This bit is RAZ, that indicates that software must perform a 32-bit write to update the TRCOSLAR.

OSLK, [1]

OS Lock status bit:

0 OS Lock is unlocked.
1 OS Lock is locked.

OSLM[0], [0]

OS Lock model [0] bit. This bit is combined with OSLM[1] to form a two-bit field that indicates the OS Lock model is implemented.

The value of this field is always 0b10, indicating that the OS Lock is implemented.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The TRCOSLSR can be accessed through the external debug interface, offset 0x304.
D6.52 TRCPDCR, Power Down Control Register

The TRCPDCR request to the system power controller to keep the ETM trace unit powered up.

Bit field descriptions

The TRCPDCR is a 32-bit register.

![TRCPDCR bit assignments](image)

RES0, [31:4]

RES0: Reserved.

PU, [3]

Powerup request, to request that power to the ETM trace unit and access to the trace registers is maintained:

0 Power not requested.
1 Power requested.

This bit is reset to 0 on a trace unit reset.

RES0, [2:0]

RES0: Reserved.

Bit fields and details not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.

The TRCPDCR can be accessed through the external debug interface, offset 0x310.
TRCPDSR, Power Down Status Register

The TRCPDSR indicates the power down status of the ETM trace unit.

Bit field descriptions

The TRCPDSR is a 32-bit register.

RES0, [31:6]

RES0 Reserved.

OSLK, [5]

OS lock status.

0 The OS Lock is unlocked.
1 The OS Lock is locked.

RES0, [4:2]

RES0 Reserved.

STICKYPD, [1]

Sticky power down state.

0 Trace register power has not been removed since the TRCPDSR was last read.
1 Trace register power has been removed since the TRCPDSR was last read.

This bit is set to 1 when power to the ETM trace unit registers is removed, to indicate that programming state has been lost. It is cleared after a read of the TRCPDSR.

POWER, [0]

Indicates the ETM trace unit is powered:

0 ETM trace unit is not powered. The trace registers are not accessible and they all return an error response.
1 ETM trace unit is powered. All registers are accessible.

If a system implementation allows the ETM trace unit to be powered off independently of the debug power domain, the system must handle accesses to the ETM trace unit appropriately.

Bit fields and details not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.

The TRCPDSR can be accessed through the external debug interface, offset 0x314.
D6.54 TRCPIDR0, ETM Peripheral Identification Register 0

The TRCPIDR0 provides information to identify a trace component.

Bit field descriptions

The TRCPIDR0 is a 32-bit register.

![Figure D6-50 TRCPIDR0 bit assignments](image)

**RES0, [31:8]**

RES0 Reserved.

**Part_0, [7:0]**

0x06 Least significant byte of the ETM trace unit part number.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The TRCPIDR0 can be accessed through the external debug interface, offset 0xFE0.
D6.55 TRCPIDR1, ETM Peripheral Identification Register 1

The TRCPIDR1 provides information to identify a trace component.

Bit field descriptions

The TRCPIDR1 is a 32-bit register.

<table>
<thead>
<tr>
<th>Bit Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RES0</td>
<td>Reserved</td>
</tr>
<tr>
<td>DES_0</td>
<td>Arm Limited. This is bits[3:0] of JEP106 ID code.</td>
</tr>
<tr>
<td>Part_1</td>
<td>Most significant four bits of the ETM trace unit part number.</td>
</tr>
</tbody>
</table>

Bit fields and details not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.

The TRCPIDR1 can be accessed through the external debug interface, offset 0xFE4.
D6.56 TRCPIDR2, ETM Peripheral Identification Register 2

The TRCPIDR2 provides information to identify a trace component.

**Bit field descriptions**

The TRCPIDR2 is a 32-bit register.

![TRCPIDR2 bit assignments](image)

- **RES0**, [31:8]
  - **RES0** Reserved.

- **Revision**, [7:4]
  - **0x2** r1p1.

- **JEDEC**, [3]
  - **0b1** RES1. Indicates a JEP106 identity code is used.

- **DES_1**, [2:0]
  - **0b011** Arm Limited. This is bits[6:4] of JEP106 ID code.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The TRCPIDR2 can be accessed through the external debug interface, offset 0xFE8.
D6.57 TRCPIDR3, ETM Peripheral Identification Register 3

The TRCPIDR3 provides information to identify a trace component.

**Bit field descriptions**

The TRCPIDR3 is a 32-bit register.

![Figure D6-53 TRCPIDR3 bit assignments](image)

- **RES0, [31:8]**
  - **RES0** Reserved.

- **REV AND, [7:4]**
  - **0x0** Part minor revision.

- **CMOD, [3:0]**
  - **0x0** Not customer modified.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The TRCPIDR3 can be accessed through the external debug interface, offset 0xFEC.
D6.58 TRCPIDR4, ETM Peripheral Identification Register 4

The TRCPIDR4 provides information to identify a trace component.

**Bit field descriptions**

The TRCPIDR4 is a 32-bit register.

![Figure D6-54 TRCPIDR4 bit assignments](image)

<table>
<thead>
<tr>
<th>Bit Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RES0, [31:8]</td>
<td>Reserved.</td>
</tr>
<tr>
<td>Size, [7:4]</td>
<td>Size of the component. Log2 the number of 4KB pages from the start of the component to the end of the component ID registers.</td>
</tr>
<tr>
<td>DES_2, [3:0]</td>
<td>Arm Limited. This is bits[3:0] of the JEP106 continuation code.</td>
</tr>
</tbody>
</table>

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The TRCPIDR4 can be accessed through the external debug interface, offset 0xFD0.
D6.59   TRCPIDRn, ETM Peripheral Identification Registers 5-7

No information is held in the Peripheral ID5, Peripheral ID6, and Peripheral ID7 Registers. They are reserved for future use and are RES0.
D6.60 TRCPRGCTLR, Programming Control Register

The TRCPRGCTLR enables the ETM trace unit.

**Bit field descriptions**

The TRCPRGCTLR is a 32-bit register.

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

- **RES0, [31:1]**
  - **RES0** Reserved.

- **EN, [0]**
  - Trace program enable:
    - 0 The ETM trace unit interface in the core is disabled, and clocks are enabled only when necessary to process APB accesses, or drain any already generated trace. This is the reset value.
    - 1 The ETM trace unit interface in the core is enabled, and clocks are enabled. Writes to most trace registers are ignored.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The TRCPRGCTLR can be accessed through the external debug interface, offset 0x004.
The TRCRSCTLRn controls the trace resources. There are eight resource pairs, the first pair is predefined as \( \{0,1,\text{pair}=0\} \) and having reserved select registers. This leaves seven pairs to be implemented as programmable selectors.

**Bit field descriptions**

The TRCRSCTLRn is a 32-bit register.

![Figure D6-56 TRCRSCTLRn bit assignments](image)

**RES0, [31:22]**

RES0 Reserved.

**PAIRINV, [21]**

Inverts the result of a combined pair of resources.

This bit is implemented only on the lower register for a pair of resource selectors.

**INV, [20]**

Inverts the selected resources:

0 Resource is not inverted.

1 Resource is inverted.

**RES0, [19]**

RES0 Reserved.

**GROUP, [18:16]**

Selects a group of resources. See the *Arm® ETM Architecture Specification, ETMv4* for more information.

**RES0, [15:8]**

RES0 Reserved.

**SELECT, [7:0]**

Selects one or more resources from the required group. One bit is provided for each resource from the group.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The TRCRSCTLRn can be accessed through the external debug interface, offset 0x208-0x023C.
The TRCSEQEVRn defines the sequencer transitions that progress to the next state or backwards to the previous state. The ETM trace unit implements a sequencer state machine with up to four states.

**Bit field descriptions**

The TRCSEQEVRn is a 32-bit register.

![Figure D6-57 TRCSEQEVRn bit assignments](image)

**RES0, [31:16]**

- **RES0** Reserved.

**B TYPE, [15]**

Selects the resource type to move backwards to this state from the next state:

- **0** Single selected resource.
- **1** Boolean combined resource pair.

**RES0, [14:12]**

- **RES0** Reserved.

**B SEL, [11:8]**

Selects the resource number, based on the value of B TYPE:

- When B TYPE is 0, selects a single selected resource from 0-15 defined by bits[3:0].
- When B TYPE is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

**F TYPE, [7]**

Selects the resource type to move forwards from this state to the next state:

- **0** Single selected resource.
- **1** Boolean combined resource pair.

**RES0, [6:4]**

- **RES0** Reserved.

**F SEL, [3:0]**

Selects the resource number, based on the value of F TYPE:

- When F TYPE is 0, selects a single selected resource from 0-15 defined by bits[3:0].
- When F TYPE is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.*

The TRCSEQEVRn registers can be accessed through the external debug interface, offsets:
TRCSEQEVR0
0x100.

TRCSEQEVR1
0x104.

TRCSEQEVR2
0x108.
D6.63 TRCSEQRSTEVR, Sequencer Reset Control Register

The TRCSEQRSTEVR resets the sequencer to state 0.

Bit field descriptions

The TRCSEQRSTEVR is a 32-bit register

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

**RES0, [31:8]**

| RES0   | Reserved. |

**RESETTYPE, [7]**

Selects the resource type to move back to state 0:

| 0 | Single selected resource. |
| 1 | Boolean combined resource pair. |

**RES0, [6:4]**

| RES0   | Reserved. |

**RESETSEL, [3:0]**

Selects the resource number, based on the value of RESETTYPE:

- When RESETTYPE is 0, selects a single selected resource from 0-15 defined by bits[3:0].
- When RESETTYPE is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

Bit fields and details not provided in this description are architecturally defined. See the *Arm*® *Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The TRCSEQRSTEVR can be accessed through the external debug interface, offset 0x118.
**D6.64 TRCSEQSTR, Sequencer State Register**

The TRCSEQSTR holds the value of the current state of the sequencer.

**Bit field descriptions**

The TRCSEQSTR is a 32-bit register

![TRCSEQSTR bit assignments](image)

**RES0, [31:2]**

- **RES0** Reserved.

**STATE, [1:0]**

- Current sequencer state:
  - `0b00` State 0.
  - `0b01` State 1.
  - `0b10` State 2.
  - `0b11` State 3.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The TRCSEQSTR can be accessed through the external debug interface, offset `0x11C`.
D6.65 TRCSSCCR0, Single-Shot Comparator Control Register 0

The TRCSSCCR0 controls the single-shot comparator.

**Bit field descriptions**

The TRCSSCSR0 is a 32-bit register.

![Figure D6-60 TRCSSCCR0 bit assignments](image)

**RES0, [31:25]**

RES0 Reserved.

**RST, [24]**

Enables the single-shot comparator resource to be reset when it occurs, to enable another comparator match to be detected:

1 Reset enabled. Multiple matches can occur.

**RES0, [23:20]**

RES0 Reserved.

**ARC, [19:16]**

Selects one or more address range comparators for single-shot control.

One bit is provided for each implemented address range comparator.

**RES0, [15:8]**

RES0 Reserved.

**SAC, [7:0]**

Selects one or more single address comparators for single-shot control.

One bit is provided for each implemented single address comparator.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The TRCSSCCR0 can be accessed through the external debug interface, offset 0x280.
D6.66 TRCSSCSR0, Single-Shot Comparator Status Register 0

The TRCSSCSR0 indicates the status of the single-shot comparator. TRCSSCSR0 is sensitive to instruction addresses.

**Bit field descriptions**

The TRCSSCSR0 is a 32-bit register

![TRCSSCSR0 bit assignments](image)

**STATUS, [31]**

Single-shot status. This indicates whether any of the selected comparators have matched:

- 0 Match has not occurred.
- 1 Match has occurred at least once.

When programming the ETM trace unit, if TRCSSCCRn.RST is b0, the STATUS bit must be explicitly written to 0 to enable this single-shot comparator control.

**RES0, [30:3]**

- RES0 Reserved.

**DV, [2]**

Data value comparator support:

- 0 Single-shot data value comparisons not supported.

**DA, [1]**

Data address comparator support:

- 0 Single-shot data address comparisons not supported.

**INST, [0]**

Instruction address comparator support:

- 1 Single-shot instruction address comparisons supported.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The TRCSSCSR0 can be accessed through the external debug interface, offset 0x2A0.
D6.67 TRCSTALLCTRLR, Stall Control Register

The TRCSTALLCTRLR enables the ETM trace unit to stall the Cortex-A65 core if the ETM trace unit FIFO overflows.

**Bit field descriptions**

The TRCSTALLCTRLR is a 32-bit register.

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

**Figure D6-62** TRCSTALLCTRLR bit assignments

RES0, [31:9]

RES0 Reserved.

ISTALL, [8]

Instruction stall bit. Controls if the trace unit can stall the core when the instruction trace buffer space is less than LEVEL:

- 0: The trace unit does not stall the core.
- 1: The trace unit can stall the core.

RES0, [7:4]

RES0 Reserved.

LEVEL, [3:2]

Threshold level field. The field can support 4 monotonic levels from 0b00 to 0b11, where:

- 0b00: Zero invasion. This setting has a greater risk of an ETM trace unit FIFO overflow.
- 0b11: Maximum invasion occurs but there is less risk of a FIFO overflow.

RES0, [1:0]

RES0 Reserved.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

The TRCSTALLCTRLR can be accessed through the external debug interface, offset 0x02C.
D6.68 TRCSTATR, Status Register

The TRCSTATR indicates the ETM trace unit status.

**Bit field descriptions**

The TRCSTATR is a 32-bit register.

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

- **RES0, [31:2]**
  - **RES0**  Reserved.

- **PMSTABLE, [1]**
  - Indicates whether the ETM trace unit registers are stable and can be read:
    - **0**  The programmers model is not stable.
    - **1**  The programmers model is stable.

- **IDLE, [0]**
  - Idle status:
    - **0**  The ETM trace unit is not idle.
    - **1**  The ETM trace unit is idle.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual* Armv8, for Armv8-A architecture profile.

The TRCSTATR can be accessed through the external debug interface, offset 0x00C.
D6.69 TRCSYNCPR, Synchronization Period Register

The TRCSYNCPR controls how often periodic trace synchronization requests occur.

**Bit field descriptions**

The TRCSYNCPR is a 32-bit register.

<table>
<thead>
<tr>
<th>Bit Field Description</th>
<th>Bit Positions</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERIOD</td>
<td>[4:0]</td>
</tr>
</tbody>
</table>

**RES0, [31:5]**

RES0 Reserved.

**PERIOD, [4:0]**

Defines the number of bytes of trace between synchronization requests as a total of the number of bytes generated by both the instruction and data streams. The number of bytes is $2^N$ where N is the value of this field:

- A value of zero disables these periodic synchronization requests, but does not disable other synchronization requests.
- The minimum value that can be programmed, other than zero, is 8, providing a minimum synchronization period of 256 bytes.
- The maximum value is 20, providing a maximum synchronization period of $2^{20}$ bytes.

Bit fields and details not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.

The TRCSYNCPR can be accessed through the external debug interface, offset 0x034.
D6.70 TRCTRACEIDR, Trace ID Register

The TRCTRACEIDR sets the trace ID for instruction trace.

Bit field descriptions

The TRCTRACEIDR is a 32-bit register.

<table>
<thead>
<tr>
<th>31</th>
<th>7-6</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>RES0</td>
<td>TRACEID</td>
<td></td>
</tr>
</tbody>
</table>

RES0, [31:7]

RES0  Reserved.

TRACEID, [6:0]

Trace ID value. When only instruction tracing is enabled, this provides the trace ID.

Bit fields and details not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.

The TRCTRACEIDR can be accessed through the external debug interface, offset 0x040.
D6.71 TRCTSCTLR, Global Timestamp Control Register

The TRCTSCTLR controls the insertion of global timestamps in the trace streams. When the selected event is triggered, the trace unit inserts a global timestamp into the trace streams. The event is selected from one of the Resource Selectors.

Bit field descriptions

The TRCTSCTLR is a 32-bit register.

![Figure D6-66 TRCTSCTLR bit assignments](image)

RES0, [31:8]

RES0 Reserved.

TYPE, [7]

Single or combined resource selector.

RES0, [6:4]

RES0 Reserved.

SEL, [3:1]

Identifies the resource selector to use.

Bit fields and details not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.

The TRCTSCTLR can be accessed through the external debug interface, offset 0x030.
D6.72 TRCVICTLR, ViewInst Main Control Register

The TRCVICTLR controls instruction trace filtering.

**Bit field descriptions**

The TRCVICTLR is a 32-bit register.

![TRCVICTLR bit assignments](image)

**RES0, [31:24]**

RES0  Reserved.

**EXLEVEL_NS, [23:20]**

In Non-secure state, each bit controls whether instruction tracing is enabled for the corresponding Exception level:

0 Trace unit generates instruction trace, in Non-secure state, for Exception level \( n \).
1 Trace unit does not generate instruction trace, in Non-secure state, for Exception level \( n \).

The Exception levels are:

- Bit[20]  Exception level 0.
- Bit[22]  Exception level 2.
- Bit[23]  RAZ/WI. Instruction tracing is not implemented for Exception level 3.

**EXLEVEL_S, [19:16]**

In Secure state, each bit controls whether instruction tracing is enabled for the corresponding Exception level:

0 Trace unit generates instruction trace, in Secure state, for Exception level \( n \).
1 Trace unit does not generate instruction trace, in Secure state, for Exception level \( n \).

The Exception levels are:

- Bit[16]  Exception level 0.
- Bit[17]  Exception level 1.
- Bit[18]  RAZ/WI. Instruction tracing is not implemented for Exception level 2.

**RES0, [15:12]**

RES0  Reserved.

**TRCERR, [11]**
Selects whether a system error exception must always be traced:

0  System error exception is traced only if the instruction or exception immediately before the system error exception is traced.
1  System error exception is always traced regardless of the value of ViewInst.

TRCRESET, [10]
Selects whether a reset exception must always be traced:

0  Reset exception is traced only if the instruction or exception immediately before the reset exception is traced.
1  Reset exception is always traced regardless of the value of ViewInst.

SSSTATUS, [9]
Indicates the current status of the start/stop logic:

0  Start/stop logic is in the stopped state.
1  Start/stop logic is in the started state.

RES0, [8]
RES0  Reserved.

TYPE, [7]
Selects the resource type for the viewinst event:

0  Single selected resource.
1  Boolean combined resource pair.

RES0, [6:4]
RES0  Reserved.

SEL, [3:0]
Selects the resource number to use for the viewinst event, based on the value of TYPE:

When TYPE is 0, selects a single selected resource from 0-15 defined by bits[3:0].

When TYPE is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0].

Bit fields and details not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.

The TRCVICTLR can be accessed through the external debug interface, offset 0x080.
D6.73 TRCVIIECTLR, ViewInst Include-Exclude Control Register

The TRCVIIECTLR defines the address range comparators that control the ViewInst Include/Exclude control.

**Bit field descriptions**

The TRCVIIECTLR is a 32-bit register.

![Bit assignments](image)

**RES0, [31:20]**

RES0 Reserved.

**EXCLUDE, [19:16]**

Defines the address range comparators for ViewInst exclude control. One bit is provided for each implemented Address Range Comparator.

**RES0, [15:4]**

RES0 Reserved.

**INCLUDE, [3:0]**

Defines the address range comparators for ViewInst include control.

Selecting no include comparators indicates that all instructions must be included. The exclude control indicates which ranges must be excluded.

One bit is provided for each implemented Address Range Comparator.

Bit fields and details not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.

The TRCVIIECTLR can be accessed through the external debug interface, offset 0x084.
D6.74   TRCVISSCTLR, ViewInst Start-Stop Control Register

The TRCVISSCTLR defines the single address comparators that control the ViewInst Start/Stop logic.

**Bit field descriptions**

The TRCVISSCTLR is a 32-bit register.

![TRCVISSCTLR bit assignments](image)

**RES0, [31:24]**

RES0  Reserved.

**STOP, [23:16]**

Defines the single address comparators to stop trace with the ViewInst Start/Stop control.

One bit is provided for each implemented single address comparator.

**RES0, [15:8]**

RES0  Reserved.

**START, [7:0]**

Defines the single address comparators to start trace with the ViewInst Start/Stop control.

One bit is provided for each implemented single address comparator.

Bit fields and details not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.

The TRCVISSCTLR can be accessed through the external debug interface, offset 0x088.
D6.75 TRCVMIDCCTLR0, Virtual context identifier Comparator Control Register 0

The TRCVMIDCCTLR0 contains the Virtual machine identifier mask value for the TRCVMIDCVR0 register.

**Bit field descriptions**

The TRCVMIDCCTLR0 is a 32-bit register.

![Figure D6-70 TRCVMIDCCTLR0 bit assignments](image)

**RES0, [31:4]**

RES0 Reserved.

**COMP0, [3:0]**

Controls the mask value that the trace unit applies to TRCVMIDCVR0. Each bit in this field corresponds to a byte in TRCVMIDCVR0. When a bit is:

0  The trace unit includes the relevant byte in TRCVMIDCVR0 when it performs the Virtual context ID comparison.

1  The trace unit ignores the relevant byte in TRCVMIDCVR0 when it performs the Virtual context ID comparison.

Bit fields and details not provided in this description are architecturally defined. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.*

The TRCVMIDCCTLR0 can be accessed through the external debug interface, offset 0x688.
D6.76 TRCVMIDCVR0, VMID Comparator Value Register 0

The TRCVMIDCVR0 contains a VMID value.

**Bit field descriptions**

The TRCVMIDCVR0 is a 64-bit register.

<table>
<thead>
<tr>
<th>63</th>
<th>32</th>
<th>31</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>RES0</td>
<td>VALUE</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**RES0, [63:32]**

RES0 Reserved.

**VALUE, [31:0]**

The VMID value.

Bit fields and details not provided in this description are architecturally defined. See the Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile.

The TRCVMIDCVR0 can be accessed through the external debug interface, offset 0x640.
D6 ETM registers
D6.76 TRCVMIDCVR0, VMID Comparator Value Register 0
Appendix A
Revisions

This appendix describes the technical changes between released issues of this book.
It contains the following section:
• A.1 Revisions on page Appx-A-516.
A.1 Revisions

This appendix describes the technical changes between released issues of this book.

### Table A-1  Issue 0000-02

<table>
<thead>
<tr>
<th>Change</th>
<th>Location</th>
<th>Affects</th>
</tr>
</thead>
<tbody>
<tr>
<td>First release</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

### Table A-2  Differences between Issue 0000-02 and Issue 0100-00

<table>
<thead>
<tr>
<th>Change</th>
<th>Location</th>
<th>Affects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Editorial changes</td>
<td>-</td>
<td>r1p0</td>
</tr>
<tr>
<td>Updated the product revision to r1p0</td>
<td>-</td>
<td>r1p0</td>
</tr>
<tr>
<td>Added support for PSTATE SSBS in the Armv8.5-A extension</td>
<td>A1.1 About the core on page A1-24</td>
<td>r1p0</td>
</tr>
<tr>
<td>Changed Dot Product support to always included</td>
<td>A1.3 Implementation options on page A1-26</td>
<td>r1p0</td>
</tr>
<tr>
<td>Added support for PSTATE SSBS in the Armv8.5-A extension</td>
<td>A1.4 Supported standards and specifications on page A1-27</td>
<td>r1p0</td>
</tr>
<tr>
<td>Added a Note for the asynchronous bridges</td>
<td>A2.1 Components on page A2-32</td>
<td>r1p0</td>
</tr>
<tr>
<td>Updated the Note for VCPU and VSYS</td>
<td>A4.2 Voltage domains on page A4-45</td>
<td>r1p0</td>
</tr>
<tr>
<td>Reordered the sections on Power control, Core power modes, and Thread power modes</td>
<td>A4.5 Power control on page A4-50, A4.6 Core power modes on page A4-51, A4.7 Thread power modes on page A4-57</td>
<td>r1p0</td>
</tr>
<tr>
<td>Added deactivated mode and new power mode transitions diagram</td>
<td>A4.6 Core power modes on page A4-51</td>
<td>r1p0</td>
</tr>
<tr>
<td>Changed shutdown to deactivated mode</td>
<td>A4.6.2 Off on page A4-53, A4.6.3 Off (emulated) on page A4-53</td>
<td>r1p0</td>
</tr>
<tr>
<td>Updated thread information</td>
<td>A4.6.4 SIMD dynamic retention on page A4-53</td>
<td>r1p0</td>
</tr>
<tr>
<td>Updated the section with more information about deactivated threads</td>
<td>A4.6.5 Core dynamic retention on page A4-54</td>
<td>r1p0</td>
</tr>
<tr>
<td>Updated table for On and Core dynamic retention.</td>
<td>A4.6.7 Encoding for core power modes on page A4-55</td>
<td>r1p0</td>
</tr>
<tr>
<td>Added deactivated mode</td>
<td>A4.7 Thread power modes on page A4-57</td>
<td>r1p0</td>
</tr>
<tr>
<td>Changed shutdown to deactivated mode</td>
<td>A4.7.3 Deactivated mode on page A4-57</td>
<td>r1p0</td>
</tr>
<tr>
<td>Added information about deactivated mode</td>
<td>A4.8 Relationship between power modes and power domains on page A4-59</td>
<td>r1p0</td>
</tr>
<tr>
<td>Added more information on threads</td>
<td>Treatment of intervening STR operations on page A6-80</td>
<td>r1p0</td>
</tr>
<tr>
<td>Added more information on threads</td>
<td>A6.4.3 Exclusive monitor on page A6-81</td>
<td>r1p0</td>
</tr>
<tr>
<td>Updated section on hard errors and detected persistent errors</td>
<td>CacheProtectionBehavior on page A8-95</td>
<td>r1p0</td>
</tr>
</tbody>
</table>
### Table A-2 Differences between Issue 0000-02 and Issue 0100-00 (continued)

<table>
<thead>
<tr>
<th>Change</th>
<th>Location</th>
<th>Affects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Removed MVFR0, MVFR1, and MVFR2</td>
<td><em>A10.2 Accessing the feature identification registers on page A10-109</em></td>
<td>r1p0</td>
</tr>
<tr>
<td>Added ID_AA64PFR1_EL1</td>
<td><em>B1.4 AArch64 registers by functional group on page B1-124</em></td>
<td>r1p0</td>
</tr>
<tr>
<td></td>
<td><em>B1.59 ID_AA64PFR1_EL1, AArch64 Processor Feature Register 1, EL1</em></td>
<td></td>
</tr>
<tr>
<td></td>
<td>on page B1-211</td>
<td></td>
</tr>
<tr>
<td>Added STBPFDIS, [22]</td>
<td><em>B1.25 CPUECTRL_EL1, CPU Extended Control Register, EL1</em></td>
<td>r1p0</td>
</tr>
<tr>
<td></td>
<td>on page B1-158</td>
<td></td>
</tr>
<tr>
<td>Updated CSV2 and CSV3</td>
<td><em>B1.58 ID_AA64PFR0_EL1, AArch64 Processor Feature Register 0, EL1</em></td>
<td>r1p0</td>
</tr>
<tr>
<td></td>
<td>on page B1-209</td>
<td></td>
</tr>
<tr>
<td>Added SSBS field</td>
<td><em>B1.59 ID_AA64PFR1_EL1, AArch64 Processor Feature Register 1, EL1</em></td>
<td>r1p0</td>
</tr>
<tr>
<td></td>
<td>on page B1-211</td>
<td></td>
</tr>
<tr>
<td>Added DSSBS field</td>
<td><em>B1.70 SCTLR_EL1, System Control Register, EL1 on page B1-224</em></td>
<td>r1p0</td>
</tr>
<tr>
<td></td>
<td><em>B1.71 SCTLR_EL2, System Control Register, EL2 on page B1-226</em></td>
<td></td>
</tr>
<tr>
<td></td>
<td><em>B1.72 SCTLR_EL3, System Control Register, EL3 on page B1-227</em></td>
<td></td>
</tr>
</tbody>
</table>

### Table A-3 Differences between Issue 0100-00 and Issue 0101-00

<table>
<thead>
<tr>
<th>Change</th>
<th>Location</th>
<th>Affects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Editorial changes</td>
<td>-</td>
<td>r1p1</td>
</tr>
<tr>
<td>Product name changed from Helios to Cortex-A65</td>
<td>Entire manual</td>
<td>r1p1</td>
</tr>
<tr>
<td>First release for r1p1</td>
<td>Document history table</td>
<td>r1p1</td>
</tr>
<tr>
<td></td>
<td>Product revision in <em>A1.7 Product revisions on page A1-30</em></td>
<td></td>
</tr>
<tr>
<td></td>
<td>MIDR reset value in <em>B1.4 AArch64 registers by functional group</em></td>
<td></td>
</tr>
<tr>
<td></td>
<td>on page B1-124</td>
<td></td>
</tr>
<tr>
<td></td>
<td><em>B1.64 MIDR_EL1, Main ID Register, EL1 on page B1-217</em></td>
<td></td>
</tr>
<tr>
<td></td>
<td><em>D2.13 EDPIDR2, External Debug Peripheral Identification Register 2</em></td>
<td></td>
</tr>
<tr>
<td></td>
<td>on page D2-373</td>
<td></td>
</tr>
<tr>
<td></td>
<td><em>D4.9 PMPIDR2, Performance Monitors Peripheral Identification Register 2</em></td>
<td></td>
</tr>
<tr>
<td></td>
<td>on page D4-403</td>
<td></td>
</tr>
<tr>
<td></td>
<td><em>D6.56 TRCPIDR2, ETM Peripheral Identification Register 2</em></td>
<td></td>
</tr>
<tr>
<td></td>
<td>on page D6-489</td>
<td></td>
</tr>
<tr>
<td>Removed ECC/parity implementation option for L1 and L2, since it is always included</td>
<td><em>A1.3 Implementation options on page A1-26</em></td>
<td>r1p1</td>
</tr>
<tr>
<td>Clarification added to the note for powering down cores independently</td>
<td><em>A4.2 Voltage domains on page A4-45</em></td>
<td>r1p1</td>
</tr>
<tr>
<td>Clarification added on execution of WFI/WFE instructions and changed shutdown to deactivated mode</td>
<td><em>A4.4 Architectural clock gating modes on page A4-48</em></td>
<td>r1p1</td>
</tr>
<tr>
<td>Clarification added on thread behavior</td>
<td><em>A4.6.1 On on page A4-53</em></td>
<td>r1p1</td>
</tr>
<tr>
<td></td>
<td><em>A4.6.4 SIMD dynamic retention on page A4-53</em></td>
<td></td>
</tr>
<tr>
<td>Change</td>
<td>Location</td>
<td>Affects</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>Encodings added to PSTATE[5:4]</td>
<td>A4.6.7 Encoding for core power modes on page A4-55</td>
<td>r1p1</td>
</tr>
<tr>
<td>Clarification added about activating the thread</td>
<td>A4.7.3 Deactivated mode on page A4-57</td>
<td>r1p1</td>
</tr>
<tr>
<td>Clarification added to SIMD dynamic retention description</td>
<td>A4.8 Relationship between power modes and power domains on page A4-59</td>
<td>r1p1</td>
</tr>
<tr>
<td>Clarification added to the TLB match description</td>
<td>A5.3 TLB match process on page A5-67</td>
<td>r1p1</td>
</tr>
<tr>
<td>Updated the TLB descriptor fields for Size, AP/HYP, S2AP, Domain, and DBM</td>
<td>A6.6.4 Main TLB RAM descriptor fields on page A6-86</td>
<td>r1p1</td>
</tr>
<tr>
<td>Updated Domain description and added table entries for Thread ID and CnP</td>
<td>A6.6.5 Walk cache descriptor fields on page A6-87</td>
<td>r1p1</td>
</tr>
<tr>
<td>Added ThreadID and CnP entries and updated Width for IPA, Unused, and Parity for the Tag RAM</td>
<td>A6.6.6 IPA cache descriptor fields on page A6-88</td>
<td>r1p1</td>
</tr>
<tr>
<td>Updated the Width for PA, Unused, and Parity</td>
<td>A6.6.6 IPA cache descriptor fields on page A6-88</td>
<td>r1p1</td>
</tr>
<tr>
<td>Error register names fixed to match the RAS specification</td>
<td>A8.7 Error injection on page A8-102</td>
<td>r1p1</td>
</tr>
<tr>
<td></td>
<td>B2.5 ERR0PFGCDN, Error Pseudo Fault Generation Count Down Register on page B2-251</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B2.6 ERR0PFGCTL, Error Pseudo Fault Generation Control Register on page B2-252</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B2.7 ERR0PFGF, Error Pseudo Fault Generation Feature Register on page B2-254</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B1.40 ERXPFGCDN_EL1, Selected Error Pseudo Fault Generation Count Down Register, EL1 on page B1-184</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B1.41 ERXPFGCTL_EL1, Selected Error Pseudo Fault Generation Control Register, EL1 on page B1-186</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B1.42 ERXPFGF_EL1, Selected Pseudo Fault Generation Feature Register, EL1 on page B1-188</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B2.1 Error system register summary on page B2-244</td>
<td></td>
</tr>
<tr>
<td>Removed ATCR, AVTCR, and AIDR registers that are not implemented</td>
<td>B1.3 AArch64 IMPLEMENTATION DEFINED register summary on page B1-123</td>
<td>r1p1</td>
</tr>
<tr>
<td></td>
<td>B1.4 AArch64 registers by functional group on page B1-124</td>
<td></td>
</tr>
<tr>
<td>Updated bit field descriptions in CPUECTLR_EL1</td>
<td>B1.25 CPUECTLR_EL1, CPU Extended Control Register, EL1 on page B1-158</td>
<td>r1p1</td>
</tr>
<tr>
<td>Updated configuration in CSSELR_EL1</td>
<td>B1.31 CSSELR_EL1, Cache Size Selection Register, EL1 on page B1-173</td>
<td>r1p1</td>
</tr>
<tr>
<td>Updated bit field description for ERRIDR_EL1</td>
<td>B1.35 ERRIDR_EL1, Error ID Register, EL1 on page B1-179</td>
<td>r1p1</td>
</tr>
<tr>
<td>Updated bit field description for ID_AA64PFR0_EL1</td>
<td>B1.58 ID_AA64PFR0_EL1, AArch64 Processor Feature Register 0, EL1 on page B1-209</td>
<td>r1p1</td>
</tr>
<tr>
<td>Change</td>
<td>Location</td>
<td>Affects</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>--------------------------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>Updated bit field description for LORID_EL1</td>
<td>B1.61 LORID_EL1, LORegion ID Register, EL1 on page B1-213</td>
<td>r1p1</td>
</tr>
<tr>
<td>Updated bit field description for MPIDR_EL1</td>
<td>B1.65 MPIDR_EL1, Multiprocessor Affinity Register, EL1 on page B1-218</td>
<td>r1p1</td>
</tr>
<tr>
<td>Updated TCR registers</td>
<td>B1.73 TCR_EL1, Translation Control Register, EL1 on page B1-229</td>
<td>r1p1</td>
</tr>
<tr>
<td></td>
<td>B1.74 TCR_EL2, Translation Control Register, EL2 on page B1-230</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B1.75 TCR_EL3, Translation Control Register, EL3 on page B1-231</td>
<td></td>
</tr>
<tr>
<td>Added information on breakpoints and watchpoints</td>
<td>C1.3.2 Breakpoints and watchpoints on page C1-305</td>
<td>r1p1</td>
</tr>
<tr>
<td>Updated TRCDEVAFF0</td>
<td>D6.21 TRCDEVAFF0, Device Affinity Register 0 on page D6-448</td>
<td>r1p1</td>
</tr>
<tr>
<td>Changed registers to Read-As-Zero/write ignore</td>
<td>D6.47 TRCLAR, Software Lock Access Register on page D6-480</td>
<td></td>
</tr>
<tr>
<td></td>
<td>D6.48 TRCLSR, Software Lock Status Register on page D6-481</td>
<td></td>
</tr>
</tbody>
</table>