



Netronome Network Flow Processor 6xxx

NFP SDK version 6.0

Micro-C Standard Library Reference Manual

- Proprietary and Confidential -

b677.dr6112

**Product code
040-00015-004**

Netronome Network Flow Processor 6xxx: Micro-C Standard Library Reference Manual

Copyright © 2008-2014 Netronome

COPYRIGHT

No part of this publication or documentation accompanying this Product may be reproduced in any form or by any means or used to make any derivative work by any means including but not limited to by translation, transformation or adaptation without permission from Netronome Systems, Inc., as stipulated by the United States Copyright Act of 1976. Contents are subject to change without prior notice.

WARRANTY

Netronome warrants that any media on which this documentation is provided will be free from defects in materials and workmanship under normal use for a period of ninety (90) days from the date of shipment. If a defect in any such media should occur during this 90-day period, the media may be returned to Netronome for a replacement.

NETRONOME DOES NOT WARRANT THAT THE DOCUMENTATION SHALL BE ERROR-FREE. THIS LIMITED WARRANTY SHALL NOT APPLY IF THE DOCUMENTATION OR MEDIA HAS BEEN (I) ALTERED OR MODIFIED; (II) SUBJECTED TO NEGLIGENCE, COMPUTER OR ELECTRICAL MALFUNCTION; OR (III) USED, ADJUSTED, OR INSTALLED OTHER THAN IN ACCORDANCE WITH INSTRUCTIONS FURNISHED BY NETRONOME OR IN AN ENVIRONMENT OTHER THAN THAT INTENDED OR RECOMMENDED BY NETRONOME.

EXCEPT FOR WARRANTIES SPECIFICALLY STATED IN THIS SECTION, NETRONOME HEREBY DISCLAIMS ALL EXPRESS OR IMPLIED WARRANTIES OF ANY KIND, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT AND FITNESS FOR A PARTICULAR PURPOSE.

Some jurisdictions do not allow the exclusion of implied warranties, so the above exclusion may not apply to some users of this documentation. This limited warranty gives users of this documentation specific legal rights, and users of this documentation may also have other rights which vary from jurisdiction to jurisdiction.

LIABILITY

Regardless of the form of any claim or action, Netronome's total liability to any user of this documentation for all occurrences combined, for claims, costs, damages or liability based on any cause whatsoever and arising from or in connection with this documentation shall not exceed the purchase price (without interest) paid by such user.

IN NO EVENT SHALL NETRONOME OR ANYONE ELSE WHO HAS BEEN INVOLVED IN THE CREATION, PRODUCTION, OR DELIVERY OF THE DOCUMENTATION BE LIABLE FOR ANY LOSS OF DATA, LOSS OF PROFITS OR LOSS OF USE OF THE DOCUMENTATION OR FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, EXEMPLARY, PUNITIVE, MULTIPLE OR OTHER DAMAGES, ARISING FROM OR IN CONNECTION WITH THE DOCUMENTATION EVEN IF NETRONOME HAS BEEN MADE AWARE OF THE POSSIBILITY OF SUCH DAMAGES. IN NO EVENT SHALL NETRONOME OR ANYONE ELSE WHO HAS BEEN INVOLVED IN THE CREATION, PRODUCTION, OR DELIVERY OF THE DOCUMENTATION BE LIABLE TO ANYONE FOR ANY CLAIMS, COSTS, DAMAGES OR LIABILITIES CAUSED BY IMPROPER USE OF THE DOCUMENTATION OR USE WHERE ANY PARTY HAS SUBSTITUTED PROCEDURES NOT SPECIFIED BY NETRONOME.

Revision History

Date	Revision	Description
April 2016	004	Updated for NFP SDK 6.0 Beta 1
September 2014	003	Updated for NFP SDK 5.1
January 2014	002	Updated for NFP SDK 5.0 Beta
August 2013	001	Initial release

Table of Contents

1. Introduction	32
1.1. About This Guide	32
1.2. Related Documentation	32
1.3. Conventions Used in this Manual	32
2. LibC Library Functions	34
2.1. String Literals	34
2.2. stdlib.h	35
2.3. nfptypes.h	35
2.3.1. Basic Types	35
2.3.2. Memory Region Types	36
2.4. memory.h	36
2.4.1. memcpy()	36
2.4.2. memmove()	39
2.4.3. memset()	40
2.4.4. memcmp()	41
2.4.5. memchr()	43
2.5. string.h	44
2.5.1. strlen()	44
2.5.2. strcmp()	44
2.5.3. strncmp()	45
2.5.4. strcpy()	46
2.5.5. strncpy()	47
2.5.6. strcat()	47
2.5.7. strncat()	48
2.5.8. strchr()	49
2.5.9. strrchr()	50
2.5.10. strstr()	50
2.5.11. strspn()	51
2.5.12. strcspn()	52
2.5.13. strpbrk()	52
2.5.14. strtok()	53
2.5.15. strtol()	54
2.5.16. strtoul()	55
3. Intrinsic Functions	56
3.1. Intrinsic Syntax Conventions	57
3.1.1. Transfer Register Modifiers	57
3.1.2. Limitations on Some I/O Functions	58
3.2. NFP-32xx Indirect Reference Formats	59
3.2.1. Indirect Reference 32xx Unions	59
3.3. NFP-6xxx Indirect Reference Formats	63
3.3.1. Indirect Reference 6xxx Unions	63
3.4. NFP Indirect Reference in General	64
3.4.1. Indirect Reference General Enumerations	64
3.4.2. Indirect Reference 6xxx Structs	66
3.4.3. Indirect Reference General Functions	68
3.5. ARM Intrinsics	70

3.5.1. ARM Functions	70
3.6. Cluster Local Scratch Intrinsics	72
3.6.1. Cluster Local Scratch Enumerations	72
3.6.2. Cluster Local Scratch Typedefs	73
3.6.3. Cluster Local Scratch Functions	73
3.7. Cluster Local Scratch Ring Intrinsics	196
3.7.1. CLS Ring Enumerations	196
3.7.2. CLS Ring Typedefs	198
3.7.3. CLS Ring Functions	199
3.8. Cluster Local Scratch Reflect Intrinsics	221
3.8.1. CLS Reflect Functions	222
3.9. Cluster Target Intrinsics	237
3.9.1. Cluster Target Enumerations	238
3.9.2. Cluster Target Unions	239
3.9.3. Cluster Target Typedefs	241
3.9.4. Cluster Target Functions	243
3.10. Control and Status Register Access Intrinsics	257
3.10.1. Control and Status Register Access Enumerations	257
3.10.2. Control and Status Register Access Unions	262
3.11. CRC Intrinsics	263
3.11.1. CRC Enumerations	263
3.11.2. CRC Functions	263
3.12. Crypto Intrinsics	276
3.12.1. Crypto Functions	276
3.13. MEM Intrinsics	279
3.14. MEM WorkQ Intrinsics	279
3.14.1. MU WorkQ Functions	281
3.15. MEM Ring Intrinsics	282
3.15.1. MU Ring Defines	282
3.15.2. MU Ring Enumerations	283
3.15.3. MU Ring Structs	283
3.15.4. MU Ring Unions	284
3.15.5. MU Ring Typedefs	285
3.15.6. MU Ring Functions	286
3.16. MEM Ticket Intrinsics	300
3.16.1. MU Ticket Structs	300
3.16.2. MU Ticket Typedefs	301
3.16.3. MU Ticket Functions	302
3.17. MEM Queue Lock Intrinsics	306
3.17.1. MU Queue Lock Structs	306
3.17.2. MU Queue Lock Unions	307
3.17.3. MU Queue Lock Typedefs	308
3.17.4. MU Queue Lock Functions	310
3.18. MEM Lock Intrinsics	318
3.18.1. MU Lock Structs	318
3.18.2. MU Lock Unions	319
3.18.3. MU Lock Typedefs	320
3.18.4. MU Lock Functions	323
3.19. MEM List Intrinsics	336
3.19.1. MU List Unions	336
3.19.2. MU List Typedefs	341
3.19.3. MU List Functions	345
3.20. MEM MicroQ Intrinsics	362

3.20.1. MU MicroQ Enumerations	362
3.20.2. MU MicroQ Structs	362
3.20.3. MU MicroQ Unions	363
3.20.4. MU MicroQ Typedefs	364
3.20.5. MU MicroQ Functions	366
3.21. MEM CAM IntrinsicS	374
3.21.1. MU CAM Structs	375
3.21.2. MU CAM Unions	376
3.21.3. MU CAM Typedefs	382
3.21.4. MU CAM Functions	390
3.22. MEM TCAM IntrinsicS	462
3.22.1. MU TCAM Defines	462
3.22.2. MU TCAM Structs	469
3.22.3. MU TCAM Unions	470
3.22.4. MU TCAM Typedefs	473
3.22.5. MU TCAM Functions	478
3.23. MEM Packet Engine IntrinsicS	564
3.23.1. MU PE Enumerations	564
3.23.2. MU PE Structs	565
3.23.3. MU PE Unions	566
3.23.4. MU PE Typedefs	568
3.23.5. MU PE Functions	570
3.24. MEM Statistics IntrinsicS	591
3.24.1. MU Statistics Defines	594
3.24.2. MU Statistics Enumerations	595
3.24.3. MU Statistics Unions	596
3.24.4. MU Statistics Typedefs	597
3.24.5. MU Statistics Functions	598
3.25. MEM Load Balancing IntrinsicS	602
3.25.1. MU LB Defines	602
3.25.2. MU LB Unions	603
3.25.3. MU LB Typedefs	607
3.25.4. MU LB Functions	610
3.26. MEM Lookup IntrinsicS	624
3.26.1. MU Lookup Structs	626
3.26.2. MU Lookup Enumerations	626
3.26.3. MU Lookup Unions	631
3.26.4. MU Lookup Typedefs	634
3.26.5. MU Lookup Functions	638
3.27. PCI Express IntrinsicS	639
3.27.1. PCI Express Functions	639
3.28. ILA IntrinsicS	648
3.28.1. ILA Enumerations	648
3.28.2. ILA Unions	649
3.28.3. ILA Typedefs	651
3.28.4. ILA Functions	653
3.29. NBI IntrinsicS	656
3.29.1. NBI Functions	656
3.30. SRAM IntrinsicS	657
3.31. Synchronization IntrinsicS	657
3.31.1. Synchronization Enumerations	658
3.31.2. Synchronization Structs	659
3.31.3. Synchronization Typedefs	659

3.31.4. Synchronization Functions	660
3.32. Math Intrinsics	667
3.32.1. MATH Functions	667
3.33. Unaligned Data Access	667
3.33.1. Unaligned Functions	668
3.34. Restrictions On Intrinsics	675
3.34.1. Intrinsic Function Arguments that Map to Transfer Registers in Microcode	675
3.35. Miscellaneous Intrinsics	677
3.35.1. Miscellaneous Enumerations	677
3.35.2. Miscellaneous Defines	681
3.35.3. Miscellaneous Unions	683
3.35.4. Miscellaneous Functions	683
4. Technical Support	706

List of Tables

1.1. Contents of this Guide	32
1.2. Conventions	33
2.1. memcpy() Prototypes	37
2.2. memmove() Prototypes	39
2.3. memset() Prototypes	41
2.4. memcmp() Prototypes	41
2.5. memchr() Prototypes	43
2.6. strlen() Prototypes	44
2.7. strcmp() Prototypes	45
2.8. strncmp() Prototypes	45
2.9. strcpy() Prototypes	46
2.10. strncpy() Prototypes	47
2.11. strcat() Prototypes	48
2.12. strncat() Prototypes	48
2.13. strchr() Prototypes	49
2.14. strrchr() Prototypes	50
2.15. strstr() Prototypes	51
2.16. strspn() Prototypes	51
2.17. strcspn() Prototypes	52
2.18. strpbrk() Prototypes	53
2.19. strtok() Prototypes	53
2.20. strtol() Prototypes	54
2.21. strtoul() Prototypes	55
3.1. union override_all_ind_t	59
3.2. union override_cnt_ind_t	59
3.3. union override_cnt_mask_ind_t	60
3.4. union override_mask_ind_t	60
3.5. union override_me_ctx_ind_t	60
3.6. union override_me_ctx_sig_ind_t	61
3.7. union override_me_xfer_ctx_cnt_ind_t	61
3.8. union override_me_xfer_ctx_ind_t	61
3.9. union override_me_xfer_imm_ind_t	62
3.10. union override_xfer_imm_cnt_ind_t	62
3.11. union override_xfer_imm_ind_t	63
3.12. union override_alu_ind_t	63
3.13. union override_csr_ind_t	64
3.14. enum ovr_field_t	64
3.15. struct generic_ind_t	67
3.16. ovr_init parameters	69
3.17. ovr_set parameters	70
3.18. arm_read parameters	70
3.19. arm_read_ind parameters	71
3.20. arm_write parameters	71
3.21. arm_write_ind parameters	72
3.22. enum CLS_METER_COLOR	72
3.23. enum CLS_METER_RFC	73
3.24. typedef CLS_METER_COLOR	73
3.25. typedef CLS_METER_RFC	73
3.26. cls_read_be_ptr32 parameters	74
3.27. cls_read_be_ptr40 parameters	74
3.28. cls_read_le_ptr32 parameters	75

3.29. <code>cls_read_le_ptr40</code> parameters	75
3.30. <code>cls_read_ptr32</code> parameters	76
3.31. <code>cls_read_ptr40</code> parameters	77
3.32. <code>cls_write_be_ptr32</code> parameters	77
3.33. <code>cls_write_be_ptr40</code> parameters	78
3.34. <code>cls_write_le_ptr32</code> parameters	78
3.35. <code>cls_write_le_ptr40</code> parameters	79
3.36. <code>cls_write8_be_ind_ptr40</code> parameters	79
3.37. <code>cls_write8_be_ind_ptr32</code> parameters	80
3.38. <code>cls_write8_le_ind_ptr40</code> parameters	80
3.39. <code>cls_write8_le_ind_ptr32</code> parameters	81
3.40. <code>cls_write8_ind_ptr40</code> parameters	82
3.41. <code>cls_write8_ind_ptr32</code> parameters	82
3.42. <code>cls_write_ptr32</code> parameters	83
3.43. <code>cls_write_ptr40</code> parameters	83
3.44. <code>cls_write8_be_ptr32</code> parameters	84
3.45. <code>cls_write8_be_ptr40</code> parameters	84
3.46. <code>cls_write8_le_ptr32</code> parameters	85
3.47. <code>cls_write8_le_ptr40</code> parameters	85
3.48. <code>cls_write8_ptr32</code> parameters	86
3.49. <code>cls_write8_ptr40</code> parameters	87
3.50. <code>cls_clear_bits_ind_ptr40</code> parameters	87
3.51. <code>cls_clear_bits_ind_ptr32</code> parameters	88
3.52. <code>cls_set_bits_ind_ptr40</code> parameters	89
3.53. <code>cls_set_bits_ind_ptr32</code> parameters	89
3.54. <code>cls_clear_bits_ptr32</code> parameters	90
3.55. <code>cls_clear_bits_ptr40</code> parameters	91
3.56. <code>cls_set_bits_ptr32</code> parameters	91
3.57. <code>cls_set_bits_ptr40</code> parameters	92
3.58. <code>cls_set_bits_imm_ptr32</code> parameters	93
3.59. <code>cls_set_bits_imm_ptr40</code> parameters	93
3.60. <code>cls_clear_bits_imm_ptr32</code> parameters	94
3.61. <code>cls_clear_bits_imm_ptr40</code> parameters	94
3.62. <code>cls_xor_bits_ind_ptr40</code> parameters	95
3.63. <code>cls_xor_bits_ind_ptr32</code> parameters	96
3.64. <code>cls_xor_bits_ptr32</code> parameters	96
3.65. <code>cls_xor_bits_ptr40</code> parameters	97
3.66. <code>cls_swap_ptr32</code> parameters	97
3.67. <code>cls_swap_ptr40</code> parameters	98
3.68. <code>cls_meter_ptr32</code> parameters	99
3.69. <code>cls_meter_ptr40</code> parameters	99
3.70. <code>cls_statistic_ptr32</code> parameters	100
3.71. <code>cls_statistic_ptr40</code> parameters	100
3.72. <code>cls_statistic_imm_ptr32</code> parameters	101
3.73. <code>cls_statistic_imm_ptr40</code> parameters	101
3.74. <code>cls_add_ptr32</code> parameters	102
3.75. <code>cls_add_ptr40</code> parameters	102
3.76. <code>cls_test_and_add_ptr32</code> parameters	103
3.77. <code>cls_test_and_add_ptr40</code> parameters	104
3.78. <code>cls_add64_ptr32</code> parameters	104
3.79. <code>cls_add64_ptr40</code> parameters	105
3.80. <code>cls_test_and_add64_ptr32</code> parameters	106
3.81. <code>cls_test_and_add64_ptr40</code> parameters	106

3.82. <code>cls_add_sat_ptr32</code> parameters	107
3.83. <code>cls_add_ind_ptr40</code> parameters	108
3.84. <code>cls_add_ind_ptr32</code> parameters	109
3.85. <code>cls_add64_ind_ptr40</code> parameters	109
3.86. <code>cls_add64_ind_ptr32</code> parameters	110
3.87. <code>cls_add_sat_ind_ptr40</code> parameters	111
3.88. <code>cls_add_sat_ind_ptr32</code> parameters	111
3.89. <code>cls_sub_ind_ptr40</code> parameters	112
3.90. <code>cls_sub_ind_ptr32</code> parameters	113
3.91. <code>cls_sub64_ind_ptr40</code> parameters	113
3.92. <code>cls_sub64_ind_ptr32</code> parameters	114
3.93. <code>cls_sub_sat_ind_ptr40</code> parameters	115
3.94. <code>cls_sub_sat_ind_ptr32</code> parameters	115
3.95. <code>cls_add_sat_ptr40</code> parameters	116
3.96. <code>cls_add_imm_ptr32</code> parameters	117
3.97. <code>cls_add_imm_ptr40</code> parameters	117
3.98. <code>cls_add_imm_sat_ptr32</code> parameters	118
3.99. <code>cls_add_imm_sat_ptr40</code> parameters	118
3.100. <code>cls_add64_imm_ptr32</code> parameters	119
3.101. <code>cls_add64_imm_ptr40</code> parameters	119
3.102. <code>cls_test_and_add_imm_ptr32</code> parameters	120
3.103. <code>cls_test_and_add_imm_ptr40</code> parameters	120
3.104. <code>cls_test_and_add64_imm_ptr32</code> parameters	121
3.105. <code>cls_test_and_add64_imm_ptr40</code> parameters	121
3.106. <code>cls_sub_ptr32</code> parameters	122
3.107. <code>cls_sub_ptr40</code> parameters	123
3.108. <code>cls_read_le_ind_ptr40</code> parameters	123
3.109. <code>cls_read_le_ind_ptr32</code> parameters	124
3.110. <code>cls_read_be_ind_ptr40</code> parameters	124
3.111. <code>cls_read_be_ind_ptr32</code> parameters	125
3.112. <code>cls_read_ind_ptr40</code> parameters	125
3.113. <code>cls_read_ind_ptr32</code> parameters	126
3.114. <code>cls_write_le_ind_ptr40</code> parameters	126
3.115. <code>cls_write_le_ind_ptr32</code> parameters	127
3.116. <code>cls_write_be_ind_ptr40</code> parameters	128
3.117. <code>cls_write_be_ind_ptr32</code> parameters	128
3.118. <code>cls_write_ind_ptr40</code> parameters	129
3.119. <code>cls_write_ind_ptr32</code> parameters	129
3.120. <code>cls_test_and_sub_ptr32</code> parameters	130
3.121. <code>cls_test_and_sub_ptr40</code> parameters	131
3.122. <code>cls_sub64_ptr32</code> parameters	132
3.123. <code>cls_sub64_ptr40</code> parameters	132
3.124. <code>cls_test_and_sub64_ptr32</code> parameters	133
3.125. <code>cls_test_and_sub64_ptr40</code> parameters	134
3.126. <code>cls_sub_sat_ptr32</code> parameters	135
3.127. <code>cls_sub_sat_ptr40</code> parameters	135
3.128. <code>cls_sub_imm_ptr32</code> parameters	136
3.129. <code>cls_sub_imm_ptr40</code> parameters	136
3.130. <code>cls_sub_imm_sat_ptr32</code> parameters	137
3.131. <code>cls_sub_imm_sat_ptr40</code> parameters	137
3.132. <code>cls_sub64_imm_ptr32</code> parameters	138
3.133. <code>cls_sub64_imm_ptr40</code> parameters	139
3.134. <code>cls_test_and_sub_imm_ptr32</code> parameters	139

3.135. <code>cls_test_and_sub_imm_ptr40</code> parameters	140
3.136. <code>cls_test_and_sub64_imm_ptr32</code> parameters	140
3.137. <code>cls_test_and_sub64_imm_ptr40</code> parameters	141
3.138. <code>cls_test_and_clear_bits_ptr32</code> parameters	141
3.139. <code>cls_test_and_clear_bits_ptr40</code> parameters	142
3.140. <code>cls_test_and_clear_bits_ind_ptr40</code> parameters	142
3.141. <code>cls_test_and_clear_bits_ind_ptr32</code> parameters	143
3.142. <code>cls_test_and_set_bits_ind_ptr40</code> parameters	143
3.143. <code>cls_test_and_set_bits_ind_ptr32</code> parameters	144
3.144. <code>cls_test_and_clear_bits_imm_ptr32</code> parameters	144
3.145. <code>cls_test_and_clear_bits_imm_ptr40</code> parameters	145
3.146. <code>cls_test_and_set_bits_ptr32</code> parameters	146
3.147. <code>cls_test_and_set_bits_ptr40</code> parameters	146
3.148. <code>cls_test_and_set_bits_imm_ptr32</code> parameters	147
3.149. <code>cls_test_and_set_bits_imm_ptr40</code> parameters	147
3.150. <code>cls_test_and_add_sat_ptr32</code> parameters	148
3.151. <code>cls_test_and_add_sat_ptr40</code> parameters	148
3.152. <code>cls_test_and_add_sat_ind_ptr40</code> parameters	149
3.153. <code>cls_test_and_add_sat_ind_ptr32</code> parameters	149
3.154. <code>cls_test_and_sub_sat_ind_ptr40</code> parameters	150
3.155. <code>cls_test_and_sub_sat_ind_ptr32</code> parameters	150
3.156. <code>cls_test_and_add_imm_sat_ptr32</code> parameters	151
3.157. <code>cls_test_and_add_imm_sat_ptr40</code> parameters	151
3.158. <code>cls_test_and_sub_sat_ptr32</code> parameters	152
3.159. <code>cls_test_and_sub_sat_ptr40</code> parameters	152
3.160. <code>cls_test_and_sub_imm_sat_ptr32</code> parameters	153
3.161. <code>cls_test_and_sub_imm_sat_ptr40</code> parameters	154
3.162. <code>cls_incr_ptr32</code> parameters	154
3.163. <code>cls_incr_ptr40</code> parameters	155
3.164. <code>cls_decr_ptr32</code> parameters	155
3.165. <code>cls_decr_ptr40</code> parameters	155
3.166. <code>cls_incr64_ptr32</code> parameters	156
3.167. <code>cls_incr64_ptr40</code> parameters	156
3.168. <code>cls_decr64_ptr32</code> parameters	156
3.169. <code>cls_decr64_ptr40</code> parameters	157
3.170. <code>cls_cmp_read_write_ptr32</code> parameters	157
3.171. <code>cls_cmp_read_write_ptr40</code> parameters	158
3.172. <code>cls_cmp_read_write_ind_ptr40</code> parameters	158
3.173. <code>cls_cmp_read_write_ind_ptr32</code> parameters	159
3.174. <code>cls_queue_lock_ind_ptr40</code> parameters	159
3.175. <code>cls_queue_lock_ind_ptr32</code> parameters	160
3.176. <code>cls_queue_lock_ptr32</code> parameters	160
3.177. <code>cls_queue_lock_ptr40</code> parameters	161
3.178. <code>cls_queue_unlock_ptr32</code> parameters	161
3.179. <code>cls_queue_unlock_ptr40</code> parameters	162
3.180. <code>cls_hash_mask_ptr32</code> parameters	162
3.181. <code>cls_hash_mask_ptr40</code> parameters	163
3.182. <code>cls_hash_mask_ind_ptr40</code> parameters	163
3.183. <code>cls_hash_mask_ind_ptr32</code> parameters	164
3.184. <code>cls_hash_mask_clear_ptr32</code> parameters	164
3.185. <code>cls_hash_mask_clear_ptr40</code> parameters	165
3.186. <code>cls_cam_lookup_ind_ptr40</code> parameters	165
3.187. <code>cls_cam_lookup_ind_ptr32</code> parameters	166

3.188. <code>cls_cam_lookup_add_ind_ptr40</code> parameters	167
3.189. <code>cls_cam_lookup_add_ind_ptr32</code> parameters	167
3.190. <code>cls_cam_lookup24_ind_ptr40</code> parameters	168
3.191. <code>cls_cam_lookup24_ind_ptr32</code> parameters	168
3.192. <code>cls_cam_lookup24_add_ind_ptr40</code> parameters	169
3.193. <code>cls_cam_lookup24_add_ind_ptr32</code> parameters	169
3.194. <code>cls_cam_lookup24_add_inc_ind_ptr40</code> parameters	170
3.195. <code>cls_cam_lookup24_add_inc_ind_ptr32</code> parameters	170
3.196. <code>cls_cam_lookup16_ind_ptr40</code> parameters	171
3.197. <code>cls_cam_lookup16_ind_ptr32</code> parameters	172
3.198. <code>cls_cam_lookup16_add_ind_ptr40</code> parameters	172
3.199. <code>cls_cam_lookup16_add_ind_ptr32</code> parameters	173
3.200. <code>cls_cam_lookup8_ind_ptr40</code> parameters	173
3.201. <code>cls_cam_lookup8_ind_ptr32</code> parameters	174
3.202. <code>cls_cam_lookup8_add_ind_ptr40</code> parameters	174
3.203. <code>cls_cam_lookup8_add_ind_ptr32</code> parameters	175
3.204. <code>cls_tcam_lookup_ind_ptr40</code> parameters	175
3.205. <code>cls_tcam_lookup_ind_ptr32</code> parameters	176
3.206. <code>cls_tcam_lookup24_ind_ptr40</code> parameters	177
3.207. <code>cls_tcam_lookup24_ind_ptr32</code> parameters	177
3.208. <code>cls_tcam_lookup16_ind_ptr40</code> parameters	178
3.209. <code>cls_tcam_lookup16_ind_ptr32</code> parameters	178
3.210. <code>cls_tcam_lookup8_ind_ptr40</code> parameters	179
3.211. <code>cls_tcam_lookup8_ind_ptr32</code> parameters	179
3.212. <code>cls_cam_lookup_ptr32</code> parameters	180
3.213. <code>cls_cam_lookup_ptr40</code> parameters	180
3.214. <code>cls_cam_lookup_add_ptr32</code> parameters	181
3.215. <code>cls_cam_lookup_add_ptr40</code> parameters	182
3.216. <code>cls_cam_lookup24_ptr32</code> parameters	182
3.217. <code>cls_cam_lookup24_ptr40</code> parameters	183
3.218. <code>cls_cam_lookup24_add_ptr32</code> parameters	183
3.219. <code>cls_cam_lookup24_add_ptr40</code> parameters	184
3.220. <code>cls_cam_lookup24_add_inc_ptr32</code> parameters	184
3.221. <code>cls_cam_lookup24_add_inc_ptr40</code> parameters	185
3.222. <code>cls_cam_lookup24_add_lock_ptr32</code> parameters	185
3.223. <code>cls_cam_lookup24_add_lock_ptr40</code> parameters	186
3.224. <code>cls_cam_lookup24_add_extend_ptr32</code> parameters	187
3.225. <code>cls_cam_lookup24_add_extend_ptr40</code> parameters	187
3.226. <code>cls_cam_lookup16_ptr32</code> parameters	188
3.227. <code>cls_cam_lookup16_ptr40</code> parameters	188
3.228. <code>cls_cam_lookup16_add_ptr32</code> parameters	189
3.229. <code>cls_cam_lookup16_add_ptr40</code> parameters	189
3.230. <code>cls_cam_lookup8_ptr32</code> parameters	190
3.231. <code>cls_cam_lookup8_ptr40</code> parameters	191
3.232. <code>cls_cam_lookup8_add_ptr32</code> parameters	191
3.233. <code>cls_cam_lookup8_add_ptr40</code> parameters	192
3.234. <code>cls_tcam_lookup_ptr32</code> parameters	192
3.235. <code>cls_tcam_lookup_ptr40</code> parameters	193
3.236. <code>cls_tcam_lookup24_ptr32</code> parameters	193
3.237. <code>cls_tcam_lookup24_ptr40</code> parameters	194
3.238. <code>cls_tcam_lookup16_ptr32</code> parameters	194
3.239. <code>cls_tcam_lookup16_ptr40</code> parameters	195
3.240. <code>cls_tcam_lookup8_ptr32</code> parameters	195

3.241. <code>cls_tcam_lookup8_ptr40</code> parameters	196
3.242. <code>enum cls_state_t</code>	196
3.243. <code>enum CLS_RING_SIZE</code>	197
3.244. <code>enum CLS_RING_EVENT_REPORTS</code>	198
3.245. <code>typedef CLS_RING_SIZE</code>	198
3.246. <code>typedef CLS_RING_EVENT_REPORTS</code>	199
3.247. <code>cls_state_test</code> parameters	199
3.248. <code>cls_ring_put_ptr32</code> parameters	199
3.249. <code>cls_ring_put_ptr40</code> parameters	200
3.250. <code>cls_ring_journal_ptr32</code> parameters	201
3.251. <code>cls_ring_journal_ptr40</code> parameters	201
3.252. <code>cls_ring_get_ptr32</code> parameters	202
3.253. <code>cls_ring_get_ptr40</code> parameters	202
3.254. <code>cls_ring_pop_ptr32</code> parameters	203
3.255. <code>cls_ring_pop_ptr40</code> parameters	204
3.256. <code>cls_ring_get_safe_ptr32</code> parameters	204
3.257. <code>cls_ring_get_safe_ptr40</code> parameters	205
3.258. <code>cls_ring_pop_safe_ptr32</code> parameters	205
3.259. <code>cls_ring_pop_safe_ptr40</code> parameters	206
3.260. <code>cls_ring_init_ptr32</code> parameters	207
3.261. <code>cls_ring_init_ptr40</code> parameters	208
3.262. <code>cls_ring_write_ptr32</code> parameters	209
3.263. <code>cls_ring_write_ptr40</code> parameters	210
3.264. <code>cls_ring_read_ptr32</code> parameters	210
3.265. <code>cls_ring_read_ptr40</code> parameters	211
3.266. <code>cls_ring_ordered_lock_ptr32</code> parameters	211
3.267. <code>cls_ring_ordered_lock_ptr40</code> parameters	212
3.268. <code>cls_ring_ordered_unlock_ptr32</code> parameters	212
3.269. <code>cls_ring_ordered_unlock_ptr40</code> parameters	213
3.270. <code>cls_ring_workq_add_thread_ptr32</code> parameters	213
3.271. <code>cls_ring_workq_add_thread_ptr40</code> parameters	214
3.272. <code>cls_ring_workq_add_work_ptr32</code> parameters	214
3.273. <code>cls_ring_workq_add_work_ptr40</code> parameters	215
3.274. <code>cls_ring_put_ind_ptr40</code> parameters	215
3.275. <code>cls_ring_put_ind_ptr32</code> parameters	216
3.276. <code>cls_ring_journal_ind_ptr40</code> parameters	217
3.277. <code>cls_ring_journal_ind_ptr32</code> parameters	217
3.278. <code>cls_ring_get_ind_ptr40</code> parameters	218
3.279. <code>cls_ring_get_ind_ptr32</code> parameters	218
3.280. <code>cls_ring_pop_ind_ptr40</code> parameters	219
3.281. <code>cls_ring_pop_ind_ptr32</code> parameters	219
3.282. <code>cls_ring_get_safe_ind_ptr40</code> parameters	220
3.283. <code>cls_ring_get_safe_ind_ptr32</code> parameters	220
3.284. <code>cls_ring_pop_safe_ind_ptr40</code> parameters	221
3.285. <code>cls_ring_pop_safe_ind_ptr32</code> parameters	221
3.286. <code>cls_reflect_write_sig_local</code> parameters	222
3.287. <code>cls_reflect_write_sig_local_ptr40</code> parameters	223
3.288. <code>cls_reflect_write_sig_remote</code> parameters	223
3.289. <code>cls_reflect_write_sig_remote_ptr40</code> parameters	224
3.290. <code>cls_reflect_write_sig_both</code> parameters	225
3.291. <code>cls_reflect_write_sig_both_ptr40</code> parameters	225
3.292. <code>cls_reflect_read_sig_remote</code> parameters	226
3.293. <code>cls_reflect_read_sig_remote_ptr40</code> parameters	226

3.294. cls_reflect_read_sig_local parameters	227
3.295. cls_reflect_read_sig_local_ptr40 parameters	228
3.296. cls_reflect_read_sig_both parameters	228
3.297. cls_reflect_read_sig_both_ptr40 parameters	229
3.298. cls_reflect_write_sig_local_ind_ptr40 parameters	230
3.299. cls_reflect_write_sig_local_ind parameters	230
3.300. cls_reflect_write_sig_remote_ind_ptr40 parameters	231
3.301. cls_reflect_write_sig_remote_ind parameters	232
3.302. cls_reflect_write_sig_both_ind_ptr40 parameters	232
3.303. cls_reflect_write_sig_both_ind parameters	233
3.304. cls_reflect_read_sig_remote_ind_ptr40 parameters	234
3.305. cls_reflect_read_sig_remote_ind parameters	234
3.306. cls_reflect_read_sig_local_ind_ptr40 parameters	235
3.307. cls_reflect_read_sig_local_ind parameters	236
3.308. cls_reflect_read_sig_both_ind_ptr40 parameters	236
3.309. cls_reflect_read_sig_both_ind parameters	237
3.310. enum CLUSTER_TARGET_REGISTER_TYPE	238
3.311. enum CLUSTER_TARGET_ADDRESS_MODE	238
3.312. enum CT_RING_SIZE	238
3.313. enum CT_RING_STATUS	239
3.314. union cluster_target_next_neighbour_write_address_format_t	239
3.315. union cluster_target_reflect_address_format_t	240
3.316. union cluster_target_signal_me_address_format_t	240
3.317. union cluster_target_xpb_address_format_t	241
3.318. typedef CLUSTER_TARGET_REGISTER_TYPE	241
3.319. typedef CLUSTER_TARGET_ADDRESS_MODE	241
3.320. typedef cluster_target_xpb_address_format_t	242
3.321. typedef cluster_target_reflect_address_format_t	242
3.322. typedef cluster_target_signal_me_address_format_t	242
3.323. typedef cluster_target_next_neighbour_write_address_format_t	242
3.324. typedef CT_RING_SIZE	242
3.325. typedef CT_RING_STATUS	243
3.326. cluster_target_xpb_write parameters	243
3.327. cluster_target_xpb_read parameters	244
3.328. cluster_target_reflect_write_sig_none parameters	245
3.329. cluster_target_reflect_read_sig_none parameters	245
3.330. cluster_target_reflect_write_sig_both parameters	246
3.331. cluster_target_reflect_read_sig_both parameters	247
3.332. cluster_target_reflect_write_sig_init parameters	247
3.333. cluster_target_reflect_read_sig_init parameters	248
3.334. cluster_target_reflect_write_sig_remote parameters	249
3.335. cluster_target_reflect_read_sig_remote parameters	249
3.336. cluster_target_sig_me_ctx parameters	250
3.337. cluster_target_next_neighbour_write parameters	251
3.338. cluster_target_next_neighbour_write_ind parameters	252
3.339. cluster_target_ring_put parameters	253
3.340. cluster_target_ring_get parameters	254
3.341. cluster_target_ring_init_ptr32 parameters	255
3.342. cluster_target_ring_init_ptr40 parameters	256
3.343. cluster_target_ring_init parameters	257
3.344. enum local_csr_t	257
3.345. union ACTIVE_CTX_STS_t	262
3.346. enum bytesSpecifier_t	263

3.347. crc_5_be parameters	264
3.348. crc_5_le parameters	264
3.349. crc_10_be parameters	265
3.350. crc_10_le parameters	265
3.351. crc_16_be parameters	266
3.352. crc_16_le parameters	266
3.353. crc_32_be parameters	267
3.354. crc_32_le parameters	267
3.355. crc_ccitt_be parameters	268
3.356. crc_ccitt_le parameters	268
3.357. crc_iscsi_be parameters	269
3.358. crc_iscsi_le parameters	269
3.359. crc_5_be_bit_swap parameters	270
3.360. crc_5_le_bit_swap parameters	270
3.361. crc_10_be_bit_swap parameters	271
3.362. crc_10_le_bit_swap parameters	271
3.363. crc_16_be_bit_swap parameters	272
3.364. crc_16_le_bit_swap parameters	272
3.365. crc_32_be_bit_swap parameters	273
3.366. crc_32_le_bit_swap parameters	273
3.367. crc_ccitt_be_bit_swap parameters	274
3.368. crc_ccitt_le_bit_swap parameters	274
3.369. crc_iscsi_be_bit_swap parameters	275
3.370. crc_iscsi_le_bit_swap parameters	275
3.371. crc_write parameters	276
3.372. crypto_read parameters	277
3.373. crypto_read_ind parameters	277
3.374. crypto_write parameters	277
3.375. crypto_write_ind parameters	278
3.376. crypto_write_fifo parameters	278
3.377. crypto_write_fifo_ind parameters	279
3.378. mem_workq_add_work parameters	281
3.379. mem_workq_add_work_imm parameters	282
3.380. mem_workq_add_thread parameters	282
3.381. MU Ring Defines	282
3.382. enum MEM_RING_SIZE	283
3.383. struct mem_ring_put_status_t	284
3.384. union mem_ring_desc_t	284
3.385. typedef MEM_RING_SIZE	285
3.386. typedef mem_ring_desc_t	285
3.387. typedef mem_ring_put_status_t	285
3.388. typedef mem_ring_desc_in_mem_t	285
3.389. typedef mem_ring_put_status_in_read_reg_t	286
3.390. mem_ring_read_desc parameters	286
3.391. mem_ring_write_desc parameters	287
3.392. mem_ring_push_desc parameters	287
3.393. mem_ring_read_buffer_unbounded parameters	288
3.394. mem_ring_read_buffer parameters	288
3.395. mem_ring_write_buffer_unbounded parameters	289
3.396. mem_ring_write_buffer parameters	290
3.397. mem_ring_add_tail parameters	290
3.398. mem_ring_put_buffer parameters	291
3.399. mem_ring_journal_buffer parameters	291

3.400. mem_ring_get_buffer parameters	292
3.401. mem_ring_get_buffer_eop parameters	293
3.402. mem_ring_get_buffer_freely parameters	294
3.403. mem_ring_pop_buffer parameters	295
3.404. mem_ring_pop_buffer_freely parameters	296
3.405. mem_ring_pop_buffer_eop parameters	296
3.406. mem_ring_fastjournal_imm parameters	297
3.407. mem_ring_init parameters	299
3.408. mem_ring_init_with_loc_and_page parameters	299
3.409. struct mem_ticket_line_push_t	300
3.410. struct mem_ticket_line_t	301
3.411. typedef mem_ticket_line_t	301
3.412. typedef mem_ticket_line_push_t	302
3.413. mem_ticket_release_ptr32 parameters	302
3.414. mem_ticket_release_ptr40 parameters	303
3.415. mem_ticket_release_push_ptr32 parameters	303
3.416. mem_ticket_release_push_ptr40 parameters	304
3.417. mem_ticket_line_init_ptr32 parameters	304
3.418. mem_ticket_line_init_ptr40 parameters	305
3.419. mem_ticket_line_push_init_ptr32 parameters	305
3.420. mem_ticket_line_push_init_ptr40 parameters	306
3.421. struct mem_lockq128_t	306
3.422. struct mem_lockq256_t	307
3.423. union mem_lockq_desc_t	307
3.424. union mem_lockq_entry_t	308
3.425. typedef mem_lockq_desc_t	308
3.426. typedef mem_lockq_entry_t	308
3.427. typedef mem_lockq128_t	308
3.428. typedef mem_lockq256_t	309
3.429. typedef mem_lockq128_in_mem_t	309
3.430. typedef mem_lockq128_ptr40_t	309
3.431. typedef mem_lockq256_in_mem_t	309
3.432. typedef mem_lockq256_ptr40_t	310
3.433. mem_lockq128_lock_ptr32 parameters	310
3.434. mem_lockq128_lock_ind_ptr32 parameters	310
3.435. mem_lockq128_lock_ptr40 parameters	311
3.436. mem_lockq128_lock_ind_ptr40 parameters	311
3.437. mem_lockq256_lock_ptr32 parameters	312
3.438. mem_lockq256_lock_ind_ptr32 parameters	312
3.439. mem_lockq256_lock_ptr40 parameters	313
3.440. mem_lockq256_lock_ind_ptr40 parameters	313
3.441. mem_lockq128_unlock_ptr32 parameters	313
3.442. mem_lockq128_unlock_ind_ptr32 parameters	314
3.443. mem_lockq128_unlock_ptr40 parameters	314
3.444. mem_lockq128_unlock_ind_ptr40 parameters	314
3.445. mem_lockq256_unlock_ptr32 parameters	315
3.446. mem_lockq256_unlock_ind_ptr32 parameters	315
3.447. mem_lockq256_unlock_ptr40 parameters	315
3.448. mem_lockq256_unlock_ind_ptr40 parameters	316
3.449. mem_lockq128_init_ptr32 parameters	316
3.450. mem_lockq128_init_ptr40 parameters	316
3.451. mem_lockq256_init_ptr32 parameters	317
3.452. mem_lockq256_init_ptr40 parameters	317

3.453. struct mem_lock128_t	318
3.454. struct mem_lock256_t	318
3.455. struct mem_lock384_t	318
3.456. struct mem_lock512_t	319
3.457. union mem_lock_in_t	319
3.458. union mem_lock_out_t	319
3.459. typedef mem_lock128_t	320
3.460. typedef mem_lock256_t	320
3.461. typedef mem_lock384_t	320
3.462. typedef mem_lock512_t	321
3.463. typedef mem_lock128_in_mem_t	321
3.464. typedef mem_lock256_in_mem_t	321
3.465. typedef mem_lock384_in_mem_t	321
3.466. typedef mem_lock512_in_mem_t	321
3.467. typedef mem_lock128_ptr40_t	321
3.468. typedef mem_lock256_ptr40_t	322
3.469. typedef mem_lock384_ptr40_t	322
3.470. typedef mem_lock512_ptr40_t	322
3.471. typedef mem_lock_in_t	322
3.472. typedef mem_lock_out_t	322
3.473. typedef mem_lock_out_in_read_reg_t	323
3.474. mem_lock128_init_ptr32 parameters	323
3.475. mem_lock128_init_ptr40 parameters	323
3.476. mem_lock256_init_ptr32 parameters	324
3.477. mem_lock256_init_ptr40 parameters	324
3.478. mem_lock384_init_ptr32 parameters	324
3.479. mem_lock384_init_ptr40 parameters	325
3.480. mem_lock512_init_ptr32 parameters	325
3.481. mem_lock512_init_ptr40 parameters	325
3.482. mem_lock128_ptr32 parameters	326
3.483. mem_lock128_ptr40 parameters	326
3.484. mem_lock256_ptr32 parameters	327
3.485. mem_lock256_ptr40 parameters	328
3.486. mem_lock384_ptr32 parameters	329
3.487. mem_lock384_ptr40 parameters	329
3.488. mem_lock512_ptr32 parameters	330
3.489. mem_lock512_ptr40 parameters	331
3.490. mem_unlock128_ptr32 parameters	332
3.491. mem_unlock128_ptr40 parameters	332
3.492. mem_unlock256_ptr32 parameters	333
3.493. mem_unlock256_ptr40 parameters	333
3.494. mem_unlock384_ptr32 parameters	334
3.495. mem_unlock384_ptr40 parameters	334
3.496. mem_unlock512_ptr32 parameters	335
3.497. mem_unlock512_ptr40 parameters	335
3.498. union mem_list_desc_t0	336
3.499. union mem_list_desc_t1	336
3.500. union mem_list_desc_t2	337
3.501. union mem_list_desc_t3	337
3.502. union mem_list_link_mem_t0	338
3.503. union mem_list_link_mem_t1	338
3.504. union mem_list_link_mem_t2	339
3.505. union mem_list_link_mem_t3	339

3.506. union mem_list_link_t0	339
3.507. union mem_list_link_t1	340
3.508. union mem_list_link_t2	340
3.509. union mem_list_link_t3	340
3.510. typedef MEM_LIST_TYPE	341
3.511. typedef mem_list_desc_t0	341
3.512. typedef mem_list_link_t0	341
3.513. typedef mem_list_link_mem_t0	341
3.514. typedef mem_list_desc_in_mem_t0	342
3.515. typedef mem_list_link_in_mem_t0	342
3.516. typedef mem_list_desc_t1	342
3.517. typedef mem_list_link_t1	342
3.518. typedef mem_list_link_mem_t1	342
3.519. typedef mem_list_desc_in_mem_t1	343
3.520. typedef mem_list_link_in_mem_t1	343
3.521. typedef mem_list_desc_t2	343
3.522. typedef mem_list_link_t2	343
3.523. typedef mem_list_link_mem_t2	344
3.524. typedef mem_list_desc_in_mem_t2	344
3.525. typedef mem_list_link_in_mem_t2	344
3.526. typedef mem_list_desc_t3	344
3.527. typedef mem_list_link_t3	344
3.528. typedef mem_list_link_mem_t3	345
3.529. typedef mem_list_desc_in_mem_t3	345
3.530. typedef mem_list_link_in_mem_t3	345
3.531. mem_list_read_desc_t0 parameters	346
3.532. mem_list_read_desc_t1 parameters	346
3.533. mem_list_read_desc_t2 parameters	347
3.534. mem_list_read_desc_t3 parameters	347
3.535. mem_list_push_desc_t0 parameters	348
3.536. mem_list_push_desc_t1 parameters	348
3.537. mem_list_push_desc_t2 parameters	349
3.538. mem_list_push_desc_t3 parameters	349
3.539. mem_list_write_desc_t0 parameters	350
3.540. mem_list_write_desc_t1 parameters	351
3.541. mem_list_write_desc_t2 parameters	351
3.542. mem_list_write_desc_t3 parameters	352
3.543. mem_list_enqueue_t0 parameters	352
3.544. mem_list_enqueue_t1 parameters	353
3.545. mem_list_enqueue_t2 parameters	354
3.546. mem_list_enqueue_t3 parameters	354
3.547. mem_list_enqueue_tail_t0 parameters	355
3.548. mem_list_enqueue_tail_t1 parameters	356
3.549. mem_list_enqueue_tail_t2 parameters	356
3.550. mem_list_enqueue_tail_t3 parameters	357
3.551. mem_list_dequeue_t0 parameters	357
3.552. mem_list_dequeue_t1 parameters	358
3.553. mem_list_dequeue_t2 parameters	359
3.554. mem_list_dequeue_t3 parameters	359
3.555. mem_list_init parameters	361
3.556. mem_list_init_with_loc_and_page parameters	361
3.557. enum MICROQ_ENTRY_SIZE	362
3.558. struct mem_microq128_t	362

3.559. struct mem_microq256_t	363
3.560. union mem_microq_desc_t	363
3.561. typedef MICROQ_ENTRY_SIZE	364
3.562. typedef mem_microq_desc_t	364
3.563. typedef mem_microq128_t	364
3.564. typedef mem_microq256_t	365
3.565. typedef mem_microq128_in_mem_t	365
3.566. typedef mem_microq128_ptr40_t	365
3.567. typedef mem_microq256_in_mem_t	365
3.568. typedef mem_microq256_ptr40_t	366
3.569. mem_microq128_put_ptr32 parameters	366
3.570. mem_microq128_put_ptr40 parameters	367
3.571. mem_microq256_put_ptr32 parameters	367
3.572. mem_microq256_put_ptr40 parameters	368
3.573. mem_microq128_get_ptr32 parameters	368
3.574. mem_microq128_get_ptr40 parameters	369
3.575. mem_microq256_get_ptr32 parameters	369
3.576. mem_microq256_get_ptr40 parameters	370
3.577. mem_microq128_pop_ptr32 parameters	370
3.578. mem_microq128_pop_ptr40 parameters	371
3.579. mem_microq256_pop_ptr32 parameters	371
3.580. mem_microq256_pop_ptr40 parameters	372
3.581. mem_microq128_init_ptr32 parameters	372
3.582. mem_microq128_init_ptr40 parameters	373
3.583. mem_microq256_init_ptr32 parameters	374
3.584. mem_microq256_init_ptr40 parameters	374
3.585. struct mem_cam128_t	375
3.586. struct mem_cam256_t	375
3.587. struct mem_cam384_t	375
3.588. struct mem_cam512_t	375
3.589. struct mem_cam_lookup32_in_t	376
3.590. union mem_cam_lookup16_add_out_t	376
3.591. union mem_cam_lookup16_in_t	376
3.592. union mem_cam_lookup16_out_t	377
3.593. union mem_cam_lookup24_add_inc_out_t	377
3.594. union mem_cam_lookup24_add_in_t	378
3.595. union mem_cam_lookup24_add_out_t	378
3.596. union mem_cam_lookup24_in_t	379
3.597. union mem_cam_lookup24_out_t	379
3.598. union mem_cam_lookup32_add_out_t	380
3.599. union mem_cam_lookup32_out_t	380
3.600. union mem_cam_lookup8_add_out_t	381
3.601. union mem_cam_lookup8_in_t	381
3.602. union mem_cam_lookup8_out_t	382
3.603. typedef mem_cam128_t	382
3.604. typedef mem_cam256_t	382
3.605. typedef mem_cam384_t	382
3.606. typedef mem_cam512_t	383
3.607. typedef mem_cam128_in_mem_t	383
3.608. typedef mem_cam256_in_mem_t	383
3.609. typedef mem_cam384_in_mem_t	383
3.610. typedef mem_cam512_in_mem_t	383
3.611. typedef mem_cam128_ptr40_t	383

3.612. <code>typedef mem_cam256_ptr40_t</code>	384
3.613. <code>typedef mem_cam384_ptr40_t</code>	384
3.614. <code>typedef mem_cam512_ptr40_t</code>	384
3.615. <code>typedef mem_cam_lookup8_in_t</code>	384
3.616. <code>typedef mem_cam_lookup8_out_t</code>	384
3.617. <code>typedef mem_cam_lookup8_add_out_t</code>	385
3.618. <code>typedef mem_cam_lookup8_out_in_read_reg_t</code>	385
3.619. <code>typedef mem_cam_lookup8_add_out_in_read_reg_t</code>	385
3.620. <code>typedef mem_cam_lookup16_in_t</code>	385
3.621. <code>typedef mem_cam_lookup16_out_t</code>	386
3.622. <code>typedef mem_cam_lookup16_add_out_t</code>	386
3.623. <code>typedef mem_cam_lookup16_out_in_read_reg_t</code>	386
3.624. <code>typedef mem_cam_lookup16_add_out_in_read_reg_t</code>	386
3.625. <code>typedef mem_cam_lookup24_in_t</code>	387
3.626. <code>typedef mem_cam_lookup24_add_in_t</code>	387
3.627. <code>typedef mem_cam_lookup24_add_inc_in_t</code>	387
3.628. <code>typedef mem_cam_lookup24_out_t</code>	387
3.629. <code>typedef mem_cam_lookup24_add_out_t</code>	388
3.630. <code>typedef mem_cam_lookup24_add_inc_out_t</code>	388
3.631. <code>typedef mem_cam_lookup24_out_in_read_reg_t</code>	388
3.632. <code>typedef mem_cam_lookup24_add_out_in_read_reg_t</code>	388
3.633. <code>typedef mem_cam_lookup24_add_inc_out_in_read_reg_t</code>	389
3.634. <code>typedef mem_cam_lookup32_in_t</code>	389
3.635. <code>typedef mem_cam_lookup32_out_t</code>	389
3.636. <code>typedef mem_cam_lookup32_add_out_t</code>	389
3.637. <code>typedef mem_cam_lookup32_out_in_read_reg_t</code>	389
3.638. <code>typedef mem_cam_lookup32_add_out_in_read_reg_t</code>	390
3.639. <code>mem_cam128_init_ptr32</code> parameters	390
3.640. <code>mem_cam128_init_ptr40</code> parameters	390
3.641. <code>mem_cam256_init_ptr32</code> parameters	391
3.642. <code>mem_cam256_init_ptr40</code> parameters	391
3.643. <code>mem_cam384_init_ptr32</code> parameters	391
3.644. <code>mem_cam384_init_ptr40</code> parameters	392
3.645. <code>mem_cam512_init_ptr32</code> parameters	392
3.646. <code>mem_cam512_init_ptr40</code> parameters	392
3.647. <code>mem_cam128_set8_ptr32</code> parameters	393
3.648. <code>mem_cam128_set8_ptr40</code> parameters	393
3.649. <code>mem_cam256_set8_ptr32</code> parameters	394
3.650. <code>mem_cam256_set8_ptr40</code> parameters	394
3.651. <code>mem_cam384_set8_ptr32</code> parameters	395
3.652. <code>mem_cam384_set8_ptr40</code> parameters	395
3.653. <code>mem_cam512_set8_ptr32</code> parameters	396
3.654. <code>mem_cam512_set8_ptr40</code> parameters	397
3.655. <code>mem_cam128_set16_ptr32</code> parameters	397
3.656. <code>mem_cam128_set16_ptr40</code> parameters	398
3.657. <code>mem_cam256_set16_ptr32</code> parameters	398
3.658. <code>mem_cam256_set16_ptr40</code> parameters	399
3.659. <code>mem_cam384_set16_ptr32</code> parameters	399
3.660. <code>mem_cam384_set16_ptr40</code> parameters	400
3.661. <code>mem_cam512_set16_ptr32</code> parameters	400
3.662. <code>mem_cam512_set16_ptr40</code> parameters	401
3.663. <code>mem_cam128_set24_ptr32</code> parameters	402
3.664. <code>mem_cam128_set24_ptr40</code> parameters	402

3.665. mem_cam256_set24_ptr32 parameters	403
3.666. mem_cam256_set24_ptr40 parameters	403
3.667. mem_cam384_set24_ptr32 parameters	404
3.668. mem_cam384_set24_ptr40 parameters	405
3.669. mem_cam512_set24_ptr32 parameters	405
3.670. mem_cam512_set24_ptr40 parameters	406
3.671. mem_cam128_set32_ptr32 parameters	406
3.672. mem_cam128_set32_ptr40 parameters	407
3.673. mem_cam256_set32_ptr32 parameters	407
3.674. mem_cam256_set32_ptr40 parameters	408
3.675. mem_cam384_set32_ptr32 parameters	408
3.676. mem_cam384_set32_ptr40 parameters	409
3.677. mem_cam512_set32_ptr32 parameters	410
3.678. mem_cam512_set32_ptr40 parameters	410
3.679. mem_cam128_lookup8_ptr32 parameters	411
3.680. mem_cam128_lookup8_ptr40 parameters	411
3.681. mem_cam256_lookup8_ptr32 parameters	412
3.682. mem_cam256_lookup8_ptr40 parameters	413
3.683. mem_cam384_lookup8_ptr32 parameters	414
3.684. mem_cam384_lookup8_ptr40 parameters	414
3.685. mem_cam512_lookup8_ptr32 parameters	415
3.686. mem_cam512_lookup8_ptr40 parameters	416
3.687. mem_cam128_lookup8_add_ptr32 parameters	416
3.688. mem_cam128_lookup8_add_ptr40 parameters	417
3.689. mem_cam256_lookup8_add_ptr32 parameters	418
3.690. mem_cam256_lookup8_add_ptr40 parameters	419
3.691. mem_cam384_lookup8_add_ptr32 parameters	419
3.692. mem_cam384_lookup8_add_ptr40 parameters	420
3.693. mem_cam512_lookup8_add_ptr32 parameters	421
3.694. mem_cam512_lookup8_add_ptr40 parameters	422
3.695. mem_cam128_lookup16_ptr32 parameters	422
3.696. mem_cam128_lookup16_ptr40 parameters	423
3.697. mem_cam256_lookup16_ptr32 parameters	424
3.698. mem_cam256_lookup16_ptr40 parameters	425
3.699. mem_cam384_lookup16_ptr32 parameters	425
3.700. mem_cam384_lookup16_ptr40 parameters	426
3.701. mem_cam512_lookup16_ptr32 parameters	427
3.702. mem_cam512_lookup16_ptr40 parameters	427
3.703. mem_cam128_lookup16_add_ptr32 parameters	428
3.704. mem_cam128_lookup16_add_ptr40 parameters	429
3.705. mem_cam256_lookup16_add_ptr32 parameters	430
3.706. mem_cam256_lookup16_add_ptr40 parameters	430
3.707. mem_cam384_lookup16_add_ptr32 parameters	431
3.708. mem_cam384_lookup16_add_ptr40 parameters	432
3.709. mem_cam512_lookup16_add_ptr32 parameters	433
3.710. mem_cam512_lookup16_add_ptr40 parameters	433
3.711. mem_cam128_lookup24_ptr32 parameters	434
3.712. mem_cam128_lookup24_ptr40 parameters	435
3.713. mem_cam256_lookup24_ptr32 parameters	436
3.714. mem_cam256_lookup24_ptr40 parameters	436
3.715. mem_cam384_lookup24_ptr32 parameters	437
3.716. mem_cam384_lookup24_ptr40 parameters	438
3.717. mem_cam512_lookup24_ptr32 parameters	438

3.718. mem_cam512_lookup24_ptr40 parameters	439
3.719. mem_cam128_lookup24_add_ptr32 parameters	440
3.720. mem_cam128_lookup24_add_ptr40 parameters	440
3.721. mem_cam256_lookup24_add_ptr32 parameters	441
3.722. mem_cam256_lookup24_add_ptr40 parameters	442
3.723. mem_cam384_lookup24_add_ptr32 parameters	443
3.724. mem_cam384_lookup24_add_ptr40 parameters	443
3.725. mem_cam512_lookup24_add_ptr32 parameters	444
3.726. mem_cam512_lookup24_add_ptr40 parameters	445
3.727. mem_cam128_lookup24_add_inc_ptr32 parameters	445
3.728. mem_cam128_lookup24_add_inc_ptr40 parameters	446
3.729. mem_cam256_lookup24_add_inc_ptr32 parameters	447
3.730. mem_cam256_lookup24_add_inc_ptr40 parameters	448
3.731. mem_cam384_lookup24_add_inc_ptr32 parameters	448
3.732. mem_cam384_lookup24_add_inc_ptr40 parameters	449
3.733. mem_cam512_lookup24_add_inc_ptr32 parameters	450
3.734. mem_cam512_lookup24_add_inc_ptr40 parameters	450
3.735. mem_cam128_lookup32_ptr32 parameters	451
3.736. mem_cam128_lookup32_ptr40 parameters	452
3.737. mem_cam256_lookup32_ptr32 parameters	453
3.738. mem_cam256_lookup32_ptr40 parameters	453
3.739. mem_cam384_lookup32_ptr32 parameters	454
3.740. mem_cam384_lookup32_ptr40 parameters	455
3.741. mem_cam512_lookup32_ptr32 parameters	455
3.742. mem_cam512_lookup32_ptr40 parameters	456
3.743. mem_cam128_lookup32_add_ptr32 parameters	457
3.744. mem_cam128_lookup32_add_ptr40 parameters	457
3.745. mem_cam256_lookup32_add_ptr32 parameters	458
3.746. mem_cam256_lookup32_add_ptr40 parameters	459
3.747. mem_cam384_lookup32_add_ptr32 parameters	460
3.748. mem_cam384_lookup32_add_ptr40 parameters	460
3.749. mem_cam512_lookup32_add_ptr32 parameters	461
3.750. mem_cam512_lookup32_add_ptr40 parameters	462
3.751. MU TCAM Defines	462
3.752. struct mem_tcam128_t	469
3.753. struct mem_tcam256_t	469
3.754. struct mem_tcam384_t	469
3.755. struct mem_tcam512_t	470
3.756. struct mem_tcam_lookup32_in_t	470
3.757. union mem_tcam_lookup16_in_t	470
3.758. union mem_tcam_lookup16_out_t	470
3.759. union mem_tcam_lookup24_in_t	471
3.760. union mem_tcam_lookup24_out_t	471
3.761. union mem_tcam_lookup32_out_t	471
3.762. union mem_tcam_lookup8_in_t	472
3.763. union mem_tcam_lookup8_out_t	472
3.764. typedef mem_tcam128_t	473
3.765. typedef mem_tcam256_t	473
3.766. typedef mem_tcam384_t	473
3.767. typedef mem_tcam512_t	473
3.768. typedef mem_tcam128_in_mem_t	473
3.769. typedef mem_tcam256_in_mem_t	474
3.770. typedef mem_tcam384_in_mem_t	474

3.771. <code>typedef mem_tcam512_in_mem_t</code>	474
3.772. <code>typedef mem_tcam128_ptr40_t</code>	474
3.773. <code>typedef mem_tcam256_ptr40_t</code>	474
3.774. <code>typedef mem_tcam384_ptr40_t</code>	474
3.775. <code>typedef mem_tcam512_ptr40_t</code>	475
3.776. <code>typedef mem_tcam_lookup8_in_t</code>	475
3.777. <code>typedef mem_tcam_lookup8_out_t</code>	475
3.778. <code>typedef mem_tcam_lookup8_out_in_read_reg_t</code>	475
3.779. <code>typedef mem_tcam_lookup16_in_t</code>	476
3.780. <code>typedef mem_tcam_lookup16_out_t</code>	476
3.781. <code>typedef mem_tcam_lookup16_out_in_read_reg_t</code>	476
3.782. <code>typedef mem_tcam_lookup24_in_t</code>	476
3.783. <code>typedef mem_tcam_lookup24_out_t</code>	476
3.784. <code>typedef mem_tcam_lookup24_out_in_read_reg_t</code>	477
3.785. <code>typedef mem_tcam_lookup32_in_t</code>	477
3.786. <code>typedef mem_tcam_lookup32_out_t</code>	477
3.787. <code>typedef mem_tcam_lookup32_out_in_read_reg_t</code>	477
3.788. <code>mem_tcam128_init_ptr32</code> parameters	478
3.789. <code>mem_tcam128_init_ptr40</code> parameters	478
3.790. <code>mem_tcam256_init_ptr32</code> parameters	479
3.791. <code>mem_tcam256_init_ptr40</code> parameters	479
3.792. <code>mem_tcam384_init_ptr32</code> parameters	479
3.793. <code>mem_tcam384_init_ptr40</code> parameters	480
3.794. <code>mem_tcam512_init_ptr32</code> parameters	480
3.795. <code>mem_tcam512_init_ptr40</code> parameters	481
3.796. <code>mem_tcam128_set8_word_ptr32</code> parameters	481
3.797. <code>mem_tcam128_set8_word_ptr40</code> parameters	482
3.798. <code>mem_tcam128_set8_entry_value_ptr32</code> parameters	482
3.799. <code>mem_tcam128_set8_entry_value_ptr40</code> parameters	483
3.800. <code>mem_tcam128_set8_entry_mask_ptr32</code> parameters	483
3.801. <code>mem_tcam128_set8_entry_mask_ptr40</code> parameters	484
3.802. <code>mem_tcam128_set8_entry_ptr32</code> parameters	484
3.803. <code>mem_tcam128_set8_entry_ptr40</code> parameters	485
3.804. <code>mem_tcam256_set8_word_ptr32</code> parameters	486
3.805. <code>mem_tcam256_set8_word_ptr40</code> parameters	486
3.806. <code>mem_tcam256_set8_entry_value_ptr32</code> parameters	487
3.807. <code>mem_tcam256_set8_entry_value_ptr40</code> parameters	487
3.808. <code>mem_tcam256_set8_entry_mask_ptr32</code> parameters	488
3.809. <code>mem_tcam256_set8_entry_mask_ptr40</code> parameters	489
3.810. <code>mem_tcam256_set8_entry_ptr32</code> parameters	489
3.811. <code>mem_tcam256_set8_entry_ptr40</code> parameters	490
3.812. <code>mem_tcam384_set8_word_ptr32</code> parameters	490
3.813. <code>mem_tcam384_set8_word_ptr40</code> parameters	491
3.814. <code>mem_tcam384_set8_entry_value_ptr32</code> parameters	491
3.815. <code>mem_tcam384_set8_entry_value_ptr40</code> parameters	492
3.816. <code>mem_tcam384_set8_entry_mask_ptr32</code> parameters	493
3.817. <code>mem_tcam384_set8_entry_mask_ptr40</code> parameters	493
3.818. <code>mem_tcam384_set8_entry_ptr32</code> parameters	494
3.819. <code>mem_tcam384_set8_entry_ptr40</code> parameters	494
3.820. <code>mem_tcam512_set8_word_ptr32</code> parameters	495
3.821. <code>mem_tcam512_set8_word_ptr40</code> parameters	496
3.822. <code>mem_tcam512_set8_entry_value_ptr32</code> parameters	496
3.823. <code>mem_tcam512_set8_entry_value_ptr40</code> parameters	497

3.824. mem_tcam512_set8_entry_mask_ptr32 parameters	497
3.825. mem_tcam512_set8_entry_mask_ptr40 parameters	498
3.826. mem_tcam512_set8_entry_ptr32 parameters	499
3.827. mem_tcam512_set8_entry_ptr40 parameters	499
3.828. mem_tcam128_set16_word_ptr32 parameters	500
3.829. mem_tcam128_set16_word_ptr40 parameters	500
3.830. mem_tcam128_set16_entry_value_ptr32 parameters	501
3.831. mem_tcam128_set16_entry_value_ptr40 parameters	501
3.832. mem_tcam128_set16_entry_mask_ptr32 parameters	502
3.833. mem_tcam128_set16_entry_mask_ptr40 parameters	503
3.834. mem_tcam128_set16_entry_ptr32 parameters	503
3.835. mem_tcam128_set16_entry_ptr40 parameters	504
3.836. mem_tcam256_set16_word_ptr32 parameters	505
3.837. mem_tcam256_set16_word_ptr40 parameters	505
3.838. mem_tcam256_set16_entry_value_ptr32 parameters	506
3.839. mem_tcam256_set16_entry_value_ptr40 parameters	506
3.840. mem_tcam256_set16_entry_mask_ptr32 parameters	507
3.841. mem_tcam256_set16_entry_mask_ptr40 parameters	507
3.842. mem_tcam256_set16_entry_ptr32 parameters	508
3.843. mem_tcam256_set16_entry_ptr40 parameters	509
3.844. mem_tcam384_set16_word_ptr32 parameters	509
3.845. mem_tcam384_set16_word_ptr40 parameters	510
3.846. mem_tcam384_set16_entry_value_ptr32 parameters	510
3.847. mem_tcam384_set16_entry_value_ptr40 parameters	511
3.848. mem_tcam384_set16_entry_mask_ptr32 parameters	512
3.849. mem_tcam384_set16_entry_mask_ptr40 parameters	512
3.850. mem_tcam384_set16_entry_ptr32 parameters	513
3.851. mem_tcam384_set16_entry_ptr40 parameters	513
3.852. mem_tcam512_set16_word_ptr32 parameters	514
3.853. mem_tcam512_set16_word_ptr40 parameters	515
3.854. mem_tcam512_set16_entry_value_ptr32 parameters	515
3.855. mem_tcam512_set16_entry_value_ptr40 parameters	516
3.856. mem_tcam512_set16_entry_mask_ptr32 parameters	516
3.857. mem_tcam512_set16_entry_mask_ptr40 parameters	517
3.858. mem_tcam512_set16_entry_ptr32 parameters	518
3.859. mem_tcam512_set16_entry_ptr40 parameters	518
3.860. mem_tcam128_set32_word_ptr32 parameters	519
3.861. mem_tcam128_set32_word_ptr40 parameters	519
3.862. mem_tcam128_set32_entry_value_ptr32 parameters	520
3.863. mem_tcam128_set32_entry_value_ptr40 parameters	520
3.864. mem_tcam128_set32_entry_mask_ptr32 parameters	521
3.865. mem_tcam128_set32_entry_mask_ptr40 parameters	522
3.866. mem_tcam128_set32_entry_ptr32 parameters	522
3.867. mem_tcam128_set32_entry_ptr40 parameters	523
3.868. mem_tcam256_set32_word_ptr32 parameters	523
3.869. mem_tcam256_set32_word_ptr40 parameters	524
3.870. mem_tcam256_set32_entry_value_ptr32 parameters	525
3.871. mem_tcam256_set32_entry_value_ptr40 parameters	525
3.872. mem_tcam256_set32_entry_mask_ptr32 parameters	526
3.873. mem_tcam256_set32_entry_mask_ptr40 parameters	526
3.874. mem_tcam256_set32_entry_ptr32 parameters	527
3.875. mem_tcam256_set32_entry_ptr40 parameters	527
3.876. mem_tcam384_set32_word_ptr32 parameters	528

3.877. mem_tcam384_set32_word_ptr40 parameters	529
3.878. mem_tcam384_set32_entry_value_ptr32 parameters	529
3.879. mem_tcam384_set32_entry_value_ptr40 parameters	530
3.880. mem_tcam384_set32_entry_mask_ptr32 parameters	530
3.881. mem_tcam384_set32_entry_mask_ptr40 parameters	531
3.882. mem_tcam384_set32_entry_ptr32 parameters	531
3.883. mem_tcam384_set32_entry_ptr40 parameters	532
3.884. mem_tcam512_set32_word_ptr32 parameters	533
3.885. mem_tcam512_set32_word_ptr40 parameters	533
3.886. mem_tcam512_set32_entry_value_ptr32 parameters	534
3.887. mem_tcam512_set32_entry_value_ptr40 parameters	534
3.888. mem_tcam512_set32_entry_mask_ptr32 parameters	535
3.889. mem_tcam512_set32_entry_mask_ptr40 parameters	535
3.890. mem_tcam512_set32_entry_ptr32 parameters	536
3.891. mem_tcam512_set32_entry_ptr40 parameters	537
3.892. mem_tcam128_lookup8_ptr32 parameters	537
3.893. mem_tcam128_lookup8_ptr40 parameters	538
3.894. mem_tcam256_lookup8_ptr32 parameters	539
3.895. mem_tcam256_lookup8_ptr40 parameters	539
3.896. mem_tcam384_lookup8_ptr32 parameters	540
3.897. mem_tcam384_lookup8_ptr40 parameters	541
3.898. mem_tcam512_lookup8_ptr32 parameters	542
3.899. mem_tcam512_lookup8_ptr40 parameters	542
3.900. mem_tcam128_lookup16_ptr32 parameters	543
3.901. mem_tcam128_lookup16_ptr40 parameters	544
3.902. mem_tcam256_lookup16_ptr32 parameters	544
3.903. mem_tcam256_lookup16_ptr40 parameters	545
3.904. mem_tcam384_lookup16_ptr32 parameters	546
3.905. mem_tcam384_lookup16_ptr40 parameters	546
3.906. mem_tcam512_lookup16_ptr32 parameters	547
3.907. mem_tcam512_lookup16_ptr40 parameters	548
3.908. mem_tcam128_lookup24_ptr32 parameters	549
3.909. mem_tcam128_lookup24_ptr40 parameters	549
3.910. mem_tcam256_lookup24_ptr32 parameters	550
3.911. mem_tcam256_lookup24_ptr40 parameters	551
3.912. mem_tcam384_lookup24_ptr32 parameters	551
3.913. mem_tcam384_lookup24_ptr40 parameters	552
3.914. mem_tcam512_lookup24_ptr32 parameters	553
3.915. mem_tcam512_lookup24_ptr40 parameters	553
3.916. mem_tcam128_lookup32_ptr32 parameters	554
3.917. mem_tcam128_lookup32_ptr40 parameters	555
3.918. mem_tcam256_lookup32_ptr32 parameters	556
3.919. mem_tcam256_lookup32_ptr40 parameters	556
3.920. mem_tcam384_lookup32_ptr32 parameters	557
3.921. mem_tcam384_lookup32_ptr40 parameters	558
3.922. mem_tcam512_lookup32_ptr32 parameters	558
3.923. mem_tcam512_lookup32_ptr40 parameters	559
3.924. mem_tcam_word_to_entry_index8 parameters	560
3.925. mem_tcam_entry_value_to_word_index8 parameters	560
3.926. mem_tcam_entry_mask_to_word_index8 parameters	561
3.927. mem_tcam_word_to_entry_index16 parameters	561
3.928. mem_tcam_entry_value_to_word_index16 parameters	562
3.929. mem_tcam_entry_mask_to_word_index16 parameters	562

3.930. mem_tcam_word_to_entry_index32 parameters	563
3.931. mem_tcam_entry_value_to_word_index32 parameters	563
3.932. mem_tcam_entry_mask_to_word_index32 parameters	564
3.933. enum MEM_PACKET_REWRITE_SCRIPT_OFFSET	564
3.934. enum MEM_PACKET_MASTER_BUCKET	565
3.935. enum MEM_PACKET_LENGTH	565
3.936. enum MEM_PACKET_TRANSFER_TYPE	565
3.937. struct mem_packet_header_t	566
3.938. union mem_packet_alloc_response_t	566
3.939. union mem_packet_complete_request_t	567
3.940. union mem_packet_read_status_response_t	567
3.941. typedef MEM_PACKET_REWRITE_SCRIPT_OFFSET	568
3.942. typedef MEM_PACKET_MASTER_BUCKET	568
3.943. typedef MEM_PACKET_LENGTH	568
3.944. typedef MEM_PACKET_TRANSFER_TYPE	568
3.945. typedef mem_packet_read_status_response_t	569
3.946. typedef mem_packet_read_status_response_in_read_reg_t	569
3.947. typedef mem_packet_alloc_response_t	569
3.948. typedef mem_packet_alloc_response_in_read_reg_t	569
3.949. typedef mem_packet_complete_request_t	569
3.950. typedef mem_packet_header_t	570
3.951. mem_packet_credit_get parameters	570
3.952. mem_packet_alloc parameters	571
3.953. mem_packet_alloc_poll parameters	572
3.954. mem_packet_free parameters	573
3.955. mem_packet_free_and_return_pointer parameters	573
3.956. mem_packet_free_and_signal parameters	573
3.957. mem_packet_return_pointer parameters	574
3.958. mem_packet_complete_drop parameters	574
3.959. mem_packet_complete_unicast parameters	575
3.960. mem_packet_complete_multicast parameters	575
3.961. mem_packet_complete_multicast_free parameters	576
3.962. mem_packet_read_packet_status parameters	576
3.963. mem_packet_wait_packet_status parameters	577
3.964. mem_packet_add_thread parameters	577
3.965. mem_pe_dma_to_memory_packet parameters	579
3.966. mem_pe_dma_to_memory_packet_free parameters	580
3.967. mem_pe_dma_to_memory_packet_swap parameters	580
3.968. mem_pe_dma_to_memory_packet_free_swap parameters	581
3.969. mem_pe_dma_to_memory_buffer parameters	582
3.970. mem_pe_dma_to_memory_buffer_swap parameters	583
3.971. mem_pe_dma_to_memory_buffer_le parameters	584
3.972. mem_pe_dma_to_memory_buffer_le_swap parameters	585
3.973. mem_pe_dma_from_memory_buffer parameters	585
3.974. mem_pe_dma_from_memory_buffer_swap parameters	586
3.975. mem_pe_dma_from_memory_buffer_le parameters	587
3.976. mem_pe_dma_from_memory_buffer_le_swap parameters	588
3.977. mem_pe_dma_to_memory_indirect parameters	589
3.978. mem_pe_dma_to_memory_indirect_swap parameters	589
3.979. mem_pe_dma_to_memory_indirect_free parameters	590
3.980. mem_pe_dma_to_memory_indirect_free_swap parameters	590
3.981. MU Statistics Defines	594
3.982. enum MEM_STATS_BASE_ADDRESS_SELECT	595

3.983. enum MEM_STATS_ADDRESS_PACK_CONFIG	596
3.984. union mem_stats_log_command_address_format_t	596
3.985. union mem_stats_packed_address_detail_t	596
3.986. union mem_stats_read_command_address_format_t	596
3.987. union mem_stats_unpacked_address_detail_t	597
3.988. typedef MEM_STATS_BASE_ADDRESS_SELECT	597
3.989. typedef MEM_STATS_ADDRESS_PACK_CONFIG	597
3.990. typedef mem_stats_log_command_address_format_t	598
3.991. typedef mem_stats_read_command_address_format_t	598
3.992. typedef mem_stats_packed_address_detail_t	598
3.993. typedef mem_stats_unpacked_address_detail_t	598
3.994. mem_stats_read parameters	599
3.995. mem_stats_read_and_clear parameters	599
3.996. mem_stats_log parameters	600
3.997. mem_stats_log_saturate parameters	600
3.998. mem_stats_log_event parameters	601
3.999. mem_stats_log_event_saturate parameters	602
3.1000. MU LB Defines	602
3.1001. union mem_lb_bucket_dcache_address_format_t	603
3.1002. union mem_lb_bucket_local_address_format_t	604
3.1003. union mem_lb_dcache_stats_address_format_t	604
3.1004. union mem_lb_descriptor_address_format_t	604
3.1005. union mem_lb_desc_t	604
3.1006. union mem_lb_id_table_address_format_t	605
3.1007. union mem_lb_local_stats_address_format_t	605
3.1008. union mem_lb_lookup_bundle_id_result_t	606
3.1009. union mem_lb_lookup_command_format_t	606
3.1010. union mem_lb_lookup_direct_result_t	606
3.1011. typedef MEM_LB_LOCATION	607
3.1012. typedef mem_lb_desc_t	607
3.1013. typedef mem_lb_desc_in_write_reg_t	607
3.1014. typedef mem_lb_desc_in_read_reg_t	607
3.1015. typedef mem_lb_local_stats_address_format_t	608
3.1016. typedef mem_lb_dcache_stats_address_format_t	608
3.1017. typedef mem_lb_descriptor_address_format_t	608
3.1018. typedef mem_lb_id_table_address_format_t	608
3.1019. typedef mem_lb_bucket_local_address_format_t	608
3.1020. typedef mem_lb_bucket_dcache_address_format_t	609
3.1021. typedef mem_lb_lookup_command_format_t	609
3.1022. typedef mem_lb_lookup_command_format_in_write_reg_t	609
3.1023. typedef mem_lb_lookup_bundle_id_result_t	609
3.1024. typedef mem_lb_lookup_bundle_id_result_in_read_reg_t	609
3.1025. typedef mem_lb_lookup_direct_result_t	610
3.1026. typedef mem_lb_lookup_direct_result_in_read_reg_t	610
3.1027. mem_lb_write_desc parameters	610
3.1028. mem_lb_read_desc parameters	611
3.1029. mem_lb_write_id_table parameters	611
3.1030. mem_lb_read_id_table parameters	612
3.1031. mem_lb_bucket_write_local parameters	613
3.1032. mem_lb_bucket_read_local parameters	613
3.1033. mem_lb_bucket_write_dcache parameters	614
3.1034. mem_lb_bucket_read_dcache parameters	614
3.1035. mem_lb_stats_read_and_clear_local parameters	615

3.1036. mem_lb_stats_read_local parameters	615
3.1037. mem_lb_stats_read_and_clear_dcache parameters	616
3.1038. mem_lb_stats_read_dcache parameters	616
3.1039. mem_lb_lookup_bundle_id parameters	619
3.1040. mem_lb_lookup_dcache parameters	620
3.1041. mem_lb_lookup_id_table parameters	621
3.1042. mem_lb_init parameters	623
3.1043. struct mem_lookup_result_t	626
3.1044. enum MEM_LOOKUP_DLUT_SMALL_SIZE	627
3.1045. enum MEM_LOOKUP_DLUT_LARGE_SIZE	627
3.1046. enum MEM_LOOKUP_RESULT	628
3.1047. enum MEM_LOOKUP_DLUT_TYPE	628
3.1048. enum MEM_LOOKUP_ALUT_SIZE	628
3.1049. enum MEM_LOOKUP_ALUT_COMMANDS	628
3.1050. enum MEM_LOOKUP_HASH_TABLE_SIZE	629
3.1051. enum MEM_LOOKUP_HASH_STARTING_BIT	630
3.1052. enum MEM_LOOKUP_HASH_COMMANDS	630
3.1053. union mem_lookup_alut_t	631
3.1054. union mem_lookup_dlut_large_recursive_t	632
3.1055. union mem_lookup_dlut_large_t	632
3.1056. union mem_lookup_dlut_small_recursive_t	633
3.1057. union mem_lookup_dlut_small_t	633
3.1058. union mem_lookup_hash_table_t	633
3.1059. union mem_lookup_recursive_hash_table_t	634
3.1060. typedef MEM_LOOKUP_DLUT_SMALL_SIZE	635
3.1061. typedef MEM_LOOKUP_DLUT_LARGE_SIZE	635
3.1062. typedef MEM_LOOKUP_RESULT	635
3.1063. typedef MEM_LOOKUP_DLUT_TYPE	635
3.1064. typedef mem_lookup_dlut_small_t	635
3.1065. typedef mem_lookup_dlut_large_t	636
3.1066. typedef mem_lookup_dlut_small_recursive_t	636
3.1067. typedef mem_lookup_dlut_large_recursive_t	636
3.1068. typedef mem_lookup_result_t	636
3.1069. typedef mem_lookup_result_in_read_reg_t	636
3.1070. typedef MEM_LOOKUP_ALUT_SIZE	637
3.1071. typedef MEM_LOOKUP_ALUT_COMMANDS	637
3.1072. typedef mem_lookup_alut_t	637
3.1073. typedef MEM_LOOKUP_HASH_TABLE_SIZE	637
3.1074. typedef MEM_LOOKUP_HASH_STARTING_BIT	637
3.1075. typedef MEM_LOOKUP_HASH_COMMANDS	638
3.1076. typedef mem_lookup_hash_table_t	638
3.1077. typedef mem_lookup_recursive_hash_table_t	638
3.1078. mem_lookup parameters	639
3.1079. pcie_read_ptr40 parameters	639
3.1080. pcie_write_ptr40 parameters	640
3.1081. pcie_read_pci_ptr40 parameters	640
3.1082. pcie_write_pci_ptr40 parameters	641
3.1083. pcie_read_rid_ptr40 parameters	641
3.1084. pcie_write_rid_ptr40 parameters	642
3.1085. pcie_read_ptr32 parameters	642
3.1086. pcie_write_ptr32 parameters	643
3.1087. pcie_read_ind_ptr32 parameters	643
3.1088. pcie_write_ind_ptr32 parameters	643

3.1089. pcie_read_pci_ptr32 parameters	644
3.1090. pcie_read_pci_ind_ptr32 parameters	644
3.1091. pcie_write_pci_ptr32 parameters	645
3.1092. pcie_write_pci_ind_ptr32 parameters	645
3.1093. pcie_read_rid_ptr32 parameters	646
3.1094. pcie_read_rid_ind_ptr32 parameters	646
3.1095. pcie_write_rid_ptr32 parameters	647
3.1096. pcie_write_rid_ind_ptr32 parameters	647
3.1097. enum ILA_OPERATION_MODE	648
3.1098. enum ILA_LOGIC_RESET	648
3.1099. enum ILA_CHANNEL	648
3.1100. union ila_config_control_register_format_t	649
3.1101. union ila_cpp_to_ilabar_register_format_t	649
3.1102. union ila_dma_command_register_format_t	650
3.1103. union ila_internal_address_format_t	650
3.1104. union ila_to_cpp_bar_register_format_t	651
3.1105. typedef ILA_OPERATION_MODE	651
3.1106. typedef ILA_LOGIC_RESET	651
3.1107. typedef ILA_CHANNEL	652
3.1108. typedef ila_config_control_register_format_t	652
3.1109. typedef ila_internal_address_format_t	652
3.1110. typedef ila_to_cpp_bar_register_format_t	652
3.1111. typedef ila_cpp_to_ilabar_register_format_t	652
3.1112. typedef ila_dma_command_register_format_t	653
3.1113. ila_write_ptr40 parameters	653
3.1114. ila_read_ptr40 parameters	653
3.1115. ila_write_check_error_ptr40 parameters	654
3.1116. ila_read_check_error_ptr40 parameters	655
3.1117. ila_write_internal_ptr40 parameters	655
3.1118. ila_read_internal_ptr40 parameters	656
3.1119. nbi_write parameters	656
3.1120. nbi_read parameters	657
3.1121. enum sync_t	658
3.1122. enum signal_t	658
3.1123. struct SIGNAL_PAIR	659
3.1124. typedef SIGNAL_MASK	659
3.1125. typedef SIGNAL	659
3.1126. typedef SIGNAL_PAIR	659
3.1127. __signal_number parameters	660
3.1128. __wait_for_any parameters	661
3.1129. __wait_for_any_single parameters	661
3.1130. __wait_for_all parameters	662
3.1131. __wait_for_all_single parameters	662
3.1132. __signals parameters	663
3.1133. ctx_wait parameters	664
3.1134. signal_test parameters	664
3.1135. signal_same_ME parameters	664
3.1136. signal_same_ME_next_ctx parameters	665
3.1137. signal_prev_ME parameters	665
3.1138. signal_prev_ME_this_ctx parameters	666
3.1139. signal_next_ME parameters	666
3.1140. signal_next_ME_this_ctx parameters	666
3.1141. log2 parameters	667

3.1142. ua_get_s8 parameters	668
3.1143. ua_get_s16 parameters	668
3.1144. ua_get_s32 parameters	669
3.1145. ua_get_s64 parameters	669
3.1146. ua_get_u8 parameters	670
3.1147. ua_get_u16 parameters	670
3.1148. ua_get_u32 parameters	670
3.1149. ua_get_u64 parameters	671
3.1150. ua_set_8 parameters	671
3.1151. ua_set_16 parameters	672
3.1152. ua_set_32 parameters	672
3.1153. ua_set_64 parameters	672
3.1154. ua_memcpy parameters	673
3.1155. ua_get_s64_mem parameters	673
3.1156. ua_get_u64_cls parameters	674
3.1157. ua_memcpy_lmem0_7_sxfer_w_clr parameters	674
3.1158. ua_memcpy_lmem0_7_dxfer_w_clr parameters	675
3.1159. enum inp_state_t	677
3.1160. enum swpack_t	680
3.1161. Miscellaneous Defines	681
3.1162. union cam_lookup_t	683
3.1163. assert_range parameters	684
3.1164. rotr parameters	684
3.1165. rotl parameters	684
3.1166. byte_align_block_be parameters	685
3.1167. byte_align_block_le parameters	685
3.1168. ffs parameters	686
3.1169. pop_count parameters	686
3.1170. bswap parameters	687
3.1171. bitswap parameters	687
3.1172. cam_write parameters	688
3.1173. cam_lookup parameters	688
3.1174. cam_write_state parameters	688
3.1175. cam_read_tag parameters	689
3.1176. cam_read_state parameters	689
3.1177. multiply_24x8 parameters	690
3.1178. multiply_16x16 parameters	690
3.1179. multiply_32x32_lo parameters	690
3.1180. multiply_32x32_hi parameters	691
3.1181. multiply_32x32 parameters	691
3.1182. __set_timestamp parameters	692
3.1183. __timestamp_stop parameters	692
3.1184. __sleep parameters	693
3.1185. nn_ring_enqueue_incr parameters	695
3.1186. __xfer_reg_number parameters	696
3.1187. __implicit_read parameters	697
3.1188. __implicit_write parameters	697
3.1189. __implicit_undef parameters	698
3.1190. __assign_relative_register parameters	699
3.1191. __set_profile_count parameters	700
3.1192. __profile_count_stop parameters	701
3.1193. inp_state_test parameters	701
3.1194. __free_write_buffer parameters	702

3.1195. __global_label parameters	703
3.1196. __LoadTimeConstant parameters	704
3.1197. __link_sym parameters	704
3.1198. __alloc_resource parameters	704
3.1199. __critical_path parameters	705
3.1200. __switch_pack parameters	705

1. Introduction

1.1 About This Guide

This document specifies the subset and the extensions to the language as well as the intrinsic functions that support the unique features of the Netronome Network Flow Processor NFP-6xxx product line.



Note

For simplicity throughout this document, the compiler will be referred to as the *C compiler*, or in some cases simply *the compiler*. Also, the Netronome NFP-6xxx network processor may be referred to as *the network processor*.

Table 1.1. Contents of this Guide

Chapter	Description
Chapter 2	LibC Library Functions.
Chapter 3	Intrinsic Functions.

1.2 Related Documentation

Further information related to the Netronome Systems NFP-6xxx product line is located in:

- *Netronome Network Flow Processor 6xxx: Datasheet*
- *Netronome Network Flow Processor 6xxx: Development Tools User's Guide*
- *Netronome Network Flow Processor 6xxx: Network Flow Assembler System User's Guide*
- *Netronome Network Flow Processor 6xxx: Databook*
- *Netronome Network Flow Processor 6xxx: Network Flow C Compiler User's Guide*
- *Netronome Network Flow Processor 6xxx: Microengine Programmer's Reference Manual*
- *Netronome Netowrk Flow Processor 6xxx: Microcode Standard Library Reference Manual*

1.3 Conventions Used in this Manual

The following conventions are used in this manual.

Table 1.2. Conventions

keyword=<required value>	Angle brackets show mandatory value that must be supplied.
...	Ellipsis indicates that an item may be repeated.
[Option]	Items in square brackets are optional.
[Option1...]	Optional items can have multiples. The equivalent of [Option1 [Option2]...].
Option=1..5	Range of allowable values. Equivalent to Option=1, 2, 3, 4, or 5.
Command1 Command2 Command3	(For Windows*) Selecting cascading options. Indicates that you should follow these steps: <ul style="list-style-type: none">• Click on Command1, which offers options including Command2• Click on Command2, which offers options including Command3• Click on Command3. For example: Start Programs Accessories Command Prompt .
SDK x.y	The x represents the current version, and y the latest point release of SDK that is installed on your system. This could be 4.7, for example.
NFP-6124 file	Keyboard input, keywords and code items are shown in monospaced font.
Netronome Systems NFP-6xxx product line	The family of Netronome Systems NFP-6xxx network processors, where 6xxx=the four-digit designator of the target chip.
Adobe* Acrobat*	An asterisk at the end of a word or name indicates it is a third-party product trademark.
Legacy Mode	Legacy mode refers to the NFP-32xx extended mode indirect reference format mode.

2. LibC Library Functions

This library provides support for a modified subset of the standard C library. Those standard library headers defined in C89/C99 that are required for the Netronome Systems NFP-6xxx network processors are supported. The `memory.h`, `string.h` and `stdlib.h` header files are implemented with significantly fewer functions supported than required by C89/C99.

The LibC library functions are similar to the standard C library functions with the following exceptions:

- Function names without attached memory types operate in SRAM. For example, the `memcpy()` function operates on buffers in SRAM while `memcpy_sram_mem()` copies a buffer from MEM (EMEM/IMEM/CTM) to SRAM.
- All pointers are aligned on a natural boundary according to memory region type. That is, pointers are 32 bits in SRAM, and local memory and 64-bits for MEM (EMEM/CTM/IMEM) and CLUSTER LOCAL SCRATCH.
- Support for the memory types that was included in the Netronome Systems NFP-32xx network processor version of the Microengine C LibC library have been removed. The following types do not exist in this version of LibC:
 - `scratch`
 - `ustore`
- DRAM has been changed to MEM to include IMEM, EMEM and CTM
- SRAM has been deprecated but is still supported for backwards compatibility

2.1 String Literals

String literals are placed into IMEM. You should not use a string literal in a position which expects a pointer to a non-IMEM memory region, unless a static initialization of a character array is being performed. This will produce an error. For example:

```
void foo(__declspec(cls) char *str_in_cls) { ... }

foo("string"); // ERROR: "string" is in IMEM and cannot be passed to foo()

foo((__declspec(cls) char *)"string");           // RUNTIME ERROR: address of "string"
                                                // is not a valid CLS address
{
__declspec(cls) char *ptr = "string";           // ERROR: "ptr" must be a character
                                                // array
__declspec(cls) char arr[7] = "string";          // CORRECT: static initialization of
                                                // character array
                                                // CORRECT: type of parameter matches
                                                // type of argument
}
```

2.2 stdlib.h

This section describes the stdlib.h macros and functions supported by the LibC library.

<code>int abs(int n);</code>	Returns the absolute value of n .
<code>long labs(long n);</code>	Returns absolute value of n (same as <code>abs()</code>).
<code>int atoi(__declspec(sram) CHAR *s); __int64 atoi64(__declspec(sram) CHAR *s)</code>	Converts string s to 32-bit integer or a 64-bit integer.
<code>int rand();</code>	Generates a random number between 0 and RAND_MAX.
<code>void srand(unsigned int seed);</code>	Seeds a new pseudo-random number sequence.
<code>unsigned int _rotr(unsigned int v, int shift); unsigned int _rotl(unsigned int v, int shift);</code>	These functions, which are non-ANSI-C compatible, rotate bits to the right or left for 32-bit ints.
<code>int tolower(int c);</code>	Returns lowercase of character c .
<code>int toupper(int c);</code>	Returns uppercase of character c .
<code>int islower(int c);</code>	Returns 1 if c is a lower-case character, and 0 otherwise.
<code>int isupper(int c);</code>	Returns 1 if c is an upper-case character, and 0 otherwise.
<code>void exit(int);</code>	Aborts or exits the current context. For the <code>exit</code> function, the code returned is the integer argument. The <code>exit()</code> function is implementation-dependent and is defined in <code>rtl.c</code> . The current implementation kills the current thread with an <code>ctx_arb[kill]</code> .

2.3 nfptypes.h

The nfptypes.h file contains abbreviated type definitions for your convenience. Section 2.3.1 and Section 2.3.2 describe these type definitions.

2.3.1 Basic Types

<code>typedef int</code>	<code>bool;</code>
<code>typedef void</code>	<code>VOID;</code>
<code>typedef unsigned char</code>	<code>U8;</code>
<code>typedef char</code>	<code>I8;</code>

```
typedef unsigned short           U16;
typedef short                   I16;
typedef unsigned int            U32;
typedef int                     I32;
typedef unsigned long long      U64;
typedef long long               I64;
typedef char                    CHAR;
typedef unsigned char           UCHAR;
```

2.3.2 Memory Region Types

All of the combinations of the Netronome Systems NFP-6xxx network processor memory regions and the basic data types (with the exception of UCHAR and bool) have been defined as single types. `ustore` and `scratch` have been deprecated. `dram` has been changed to `mem`. The following are some examples of these combinations.

```
typedef __declspec(sram, CHAR)           SRAM_CHAR;
typedef __declspec(local_mem, U32)        LMEM_U32;
typedef __declspec(mem, I64)              MEM_I64;
typedef __declspec(cls, I16)              CLS_I16;
```

2.4 memory.h

The `memory.h` file contains four memory functions that operate in the same way as the standard C memory functions with the exception of the `memchr()` function, which returns an offset to the first occurrence of a character within a buffer rather than the address of the character.

A memory function that does not contain a memory region in its name works only with buffers in SRAM. Each of the four memory functions, however, have memory-specific counterparts that work on buffers from different memory regions. For example, the `memcpy_sram_mem()` function is used to copy a buffer from MEM to a buffer in SRAM. The function naming convention used with all memory-specific functions is to specify the destination memory region first followed by the source memory region.

The following sections describe each of the functions contained in the `memory.h` header file.

2.4.1 memcpy()

```
__declspec(sram) void *memcpy( __declspec(sram) void *dest,
                             __declspec(sram) const void *src,
```

```
        unsigned int count);
```

Like the standard C `memcpy` function, this function copies up to **count** characters from memory buffer **src** into buffer **dest** and returns **dest**. Both memory buffers are in SRAM. This function correctly handles non optimal alignment cases.

Similar functions are provided for operation in all memory regions. Table 2.1 summarizes the prototypes for each of the memory-specific `memcpy()` functions.

Table 2.1. `memcpy()` Prototypes

Prototype	Description
<code>__declspec(mem) void* memcpy_mem_mem(__declspec(mem) void *dest, __declspec(mem) const void *src, unsigned int count);</code>	Copies from src in MEM (IMEM/EMEM/CTM) to dest in MEM (IMEM/EMEM/CTM).
<code>__declspec(mem) void* memcpy_mem_lmem(__declspec(mem) void *dest, __declspec(local_mem) const void *src, unsigned int count);</code>	Copies from src in LMEM to dest in MEM (IMEM/EMEM/CTM).
<code>__declspec(mem) void* memcpy_mem_sram(__declspec(mem) void *dest, __declspec(sram) const void *src, unsigned int count);</code>	Copies from src in SRAM to dest in MEM (IMEM/EMEM/CTM).
<code>__declspec(local_mem) memcpy_lmem_mem(void* __declspec(local_mem) void *dest, __declspec(mem) const void *src, unsigned int count);</code>	Copies from src in MEM (IMEM/EMEM/CTM) to dest in LMEM.
<code>__declspec(local_mem) memcpy_lmem_lmem(void* __declspec(local_mem) void *dest, __declspec(local_mem) const void *src, unsigned int count);</code>	Copies from src in LMEM to dest in LMEM.
<code>__declspec(local_mem)* memcpy_lmem_sram(void __declspec(local_mem) void *dest, __declspec(sram) const void *src, unsigned int count);</code>	Copies from src in SRAM to dest in LMEM.
<code>__declspec(sram) void* memcpy_sram_mem(__declspec(sram) const void *src, unsigned int count);</code>	Copies from src in MEM (IMEM/EMEM/CTM) to dest in SRAM.

Prototype	Description
<code>__declspec(sram) void *dest, __declspec(mem) const void *src, unsigned int count);</code>	
<code>__declspec(sram) void* memcpy_sram_lmem(__declspec(sram) void *dest, __declspec(local_mem) const void *src, unsigned int count);</code>	Copies from src in local memory (LMEM) to dest in SRAM.
<code>__declspec(sram) void* memcpy_sram_sram(__declspec(sram) void *dest, __declspec(sram) const void *src, unsigned int count);</code>	Copies from src in SRAM to dest in SRAM. This is identical to the <code>memcpy()</code> function.
<code>__declspec(sram) void* memcpy_sram_cls(__declspec(sram) void *dest, __declspec(cls) const void *src, unsigned int count);</code>	Copies from src in CLUSTER LOCAL SCRATCH to dest in SRAM.
<code>__declspec(mem) void* memcpy_mem_cls(__declspec(mem) void *dest, __declspec(cls) const void *src, unsigned int count);</code>	Copies from src in CLUSTER LOCAL SCRATCH to dest in MEM (IMEM/EMEM/CTM).
<code>__declspec(cls) void* memcpy_cls_sram(__declspec(cls) void *dest, __declspec(sram) const void *src, unsigned int count);</code>	Copies from src in SRAM to dest in CLUSTER LOCAL SCRATCH.
<code>__declspec(cls) void* memcpy_cls_mem(__declspec(cls) void *dest, __declspec(mem) const void *src, unsigned int count);</code>	Copies from src in MEM (IMEM/EMEM/CTM) to dest in CLUSTER LOCAL SCRATCH.
<code>__declspec(cls) void* memcpy_cls_cls(__declspec(cls) void *dest, __declspec(cls) const void *src, unsigned int count);</code>	Copies from src in CLUSTER LOCAL SCRATCH to dest in CLUSTER LOCAL SCRATCH.
<code>__declspec(cls) void* memcpy_cls_lmem(__declspec(cls) void *dest, __declspec(lmem) const void *src, unsigned int count);</code>	Copies from src in LOCAL MEMORY to dest in CLUSTER LOCAL SCRATCH.

Prototype	Description
<code>__declspec(lmem) void* memcpy_lmem_cls(</code> <code> __declspec(lmem) void *dest,</code> <code> __declspec(cls) const void *src,</code> <code> unsigned int count);</code>	Copies from src in CLUSTER LOCAL SCRATCH to dest in LOCAL MEMORY.

2.4.2 memmove()

```

__declspec(sram) void *memmove( __declspec(sram) void *dest,
                               __declspec(sram) const void *src,
                               unsigned int count);

```

Like the standard C memcpy function, this function copies up to **count** characters from memory buffer **src** into buffer **dest** and returns **dest**. Unlike memcpy(), however, this function handles the case of overlapping buffers. Both memory buffers are in SRAM. This function correctly handles non optimal alignment cases.

Similar functions are provided for operation in all memory regions. Table 2.2 summarizes the prototypes for each of the memory-specific memmove() functions.

Table 2.2. memmove() Prototypes

Prototype	Description
<code>__declspec(sram) void * memmove_sram_sram(</code> <code> __declspec(sram) void *dest,</code> <code> __declspec(sram) const void *src,</code> <code> unsigned int count);</code>	Copies from src in SRAM to dest in SRAM. This is identical to the memcpy() function.
<code>__declspec(sram) void * memmove_sram_lmem(</code> <code> __declspec(sram) void *dest,</code> <code> __declspec(local_mem) const void *src,</code> <code> unsigned int count);</code>	Copies from src in local memory (LMEM) to dest in SRAM.
<code>__declspec(sram) void * memmove_sram_mem(</code> <code> __declspec(sram) void *dest,</code> <code> __declspec(mem) const void *src,</code> <code> unsigned int count);</code>	Copies from src in MEM (IMEM/EMEM/CTM) to dest in SRAM.
<code>__declspec(local_mem) void * memmove_lmem_sram(</code> <code> __declspec(local_mem) void *dest,</code> <code> __declspec(sram) const void *src,</code> <code> unsigned int count);</code>	Copies from src in SRAM to dest in LMEM.

Prototype	Description
<code>__declspec(local_mem) void * memmove_lmem_lmem(__declspec(local_mem) void *dest, __declspec(local_mem) const void *src, unsigned int count);</code>	Copies from src in LMEM to dest in LMEM.
<code>__declspec(local_mem) void * memmove_lmem_mem(__declspec(local_mem) void *dest, __declspec(mem) const void *src, unsigned int count);</code>	Copies from src in MEM (IMEM/EMEM/CTM) to dest in LMEM.
<code>__declspec(mem) void * memmove_mem_sram(__declspec(mem) void *dest, __declspec(sram) const void *src, unsigned int count);</code>	Copies from src in SRAM to dest in MEM (IMEM/EMEM/CTM).
<code>__declspec(mem) void * memmove_mem_lmem(__declspec(mem) void *dest, __declspec(local_mem) const void *src, unsigned int count);</code>	Copies from src in LMEM to dest in MEM (IMEM/EMEM/CTM).
<code>__declspec(mem) void * memmove_mem_mem(__declspec(mem) void *dest, __declspec(mem) const void *src, unsigned int count);</code>	Copies from src in MEM (IMEM/EMEM/CTM) to dest in MEM (IMEM/EMEM/CTM).

2.4.3 memset()

```
__declspec(sram) void *memset( __declspec(sram) void *dest,  
                                  CHAR c,  
                                  unsigned int count);
```

This function fills the aligned memory buffer **dest** in SRAM with repeated 8-bit characters **c** for the first **count** characters and returns **dest**.

Similar functions are provided for operation in all memory regions. Table 2.3 summarizes the prototypes for each of the memory-specific memcmp() functions.

Table 2.3. memset() Prototypes

Prototype	Definition
<code>__declspec(sram) void *</code> <code>memset_sram(__declspec(sram) void *dest, CHAR c, unsigned int count);</code>	Fills the dest buffer in SRAM with count number of 8-bit characters specified by c . This is identical to the <code>memset()</code> function.
<code>__declspec(local_mem) void *</code> <code>memset_lmem(__declspec(local_mem) void *dest, CHAR c, unsigned int count);</code>	Fills the dest buffer in local memory with count number of 8-bit characters specified by c .
<code>__declspec(mem) void *</code> <code>memset_mem(__declspec(mem) void *dest, CHAR c, unsigned int count);</code>	Fills the dest buffer in MEM (IMEM/EMEM/CTM) with count number of 8-bit characters specified by c .
<code>__declspec(cls) void *</code> <code>memset_cls(__declspec(cls) void *dest, CHAR c, unsigned int count);</code>	Fills the dest buffer in CLUSTER LOCAL SCRATCH with count number of 8-bit characters specified by c .

2.4.4 memcmp()

```
int memcmp( __declspec(sram) const void *buf1,
            __declspec(sram) const void *buf2,
            unsigned int count);
```

Compares the first **count** characters in two possibly misaligned memory buffers in SRAM. Performs unsigned comparison on each byte. Returns a negative value if **buf1** is smaller than **buf2**, 0 if **buf1** is equal to **buf2**, or positive value if **buf1** is greater than **buf2**.

Similar functions are provided for operation in all memory regions. Table 2.4 summarizes the prototypes for each of the memory-specific `memcmp()` functions.

Table 2.4. memcmp() Prototypes

Prototype	Description
<code>int memcmp_sram_sram(__declspec(sram) const void *buf1, __declspec(sram) const void *buf2, unsigned int count);</code>	Compares count number of bytes in buf1 and buf2 , both of which are in SRAM. This is identical to the <code>memcmp()</code> function.
<code>int memcmp_sram_lmem(__declspec(sram) const void *buf1, __declspec(local_mem) const void *buf2, unsigned int count);</code>	Compares count number of bytes between buf1 in SRAM and buf2 in LMEM.
<code>int memcmp_sram_mem(__declspec(sram) const void *buf1, __declspec(mem) const void *buf2,</code>	Compares count number of bytes between buf1 in SRAM and buf2 in MEM (IMEM/EMEM/CTM).

Prototype	Description
unsigned int count);	
int memcmp_lmem_sram(__declspec(local_mem) const void *buf1, __declspec(sram) const void *buf2, unsigned int count);	Compares count number of bytes between buf1 in local memory and buf2 in SRAM.
int memcmp_lmem_lmem(__declspec(local_mem) const void *buf1, __declspec(local_mem) const void *buf2, unsigned int count);	Compares count number of bytes between buf1 in local memory and buf2 in local memory.
int memcmp_lmem_mem(__declspec(local_mem) const void *buf1, __declspec(mem) const void *buf2, unsigned int count);	Compares count number of bytes between buf1 in local memory and buf2 in MEM (IMEM/EMEM/CTM).
int memcmp_mem_sram(__declspec(mem) const void *buf1, __declspec(sram) const void *buf2, unsigned int count);	Compares count number of bytes between buf1 in MEM (IMEM/EMEM/CTM) and buf2 in SRAM.
int memcmp_mem_lmem(__declspec(mem) const void *buf1, __declspec(local_mem) const void *buf2, unsigned int count);	Compares count number of bytes between buf1 in MEM (IMEM/EMEM/CTM) and buf2 in LMEM.
int memcmp_mem_mem(__declspec(mem) const void *buf1, __declspec(mem) const void *buf2, unsigned int count);	Compares count number of bytes between buf1 in MEM (IMEM/EMEM/CTM) and buf2 in MEM (IMEM/EMEM/CTM).
int memcmp_sram_cls(__declspec(sram) const void *buf1, __declspec(cls) const void *buf2, unsigned int count);	Compares count number of bytes between buf1 in SRAM and buf2 in CLUSTER LOCAL SCRATCH.
int memcmp_mem_cls __declspec(mem) const void *buf1, __declspec(cls) const void *buf2, unsigned int count);	Compares count number of bytes between buf1 in MEM (IMEM/EMEM/CTM) and buf2 in CLUSTER LOCAL SCRATCH.
int memcmp_cls_sram __declspec(cls) const void *buf1, __declspec(sram) const void *buf2, unsigned int count);	Compares count number of bytes between buf1 in CLUSTER LOCAL SCRATCH and buf2 in SRAM.
int memcmp_cls_mem __declspec(cls) const void *buf1, __declspec(mem) const void *buf2, unsigned int count);	Compares count number of bytes between buf1 in CLUSTER LOCAL SCRATCH and buf2 in MEM (IMEM/EMEM/CTM).
int memcmp_cls_cls __declspec(cls) const void *buf1, __declspec(cls) const void *buf2, unsigned int count);	Compares count number of bytes between buf1 in CLUSTER LOCAL SCRATCH and buf2 in CLUSTER LOCAL SCRATCH.

Prototype	Description
int memcmp_cls_lmem __declspec(cls) const void *buf1, __declspec(lmem) const void *buf2, unsigned int count);	Compares count number of bytes between buf1 in CLUSTER LOCAL SCRATCH and buf2 in LOCAL MEMORY.
int memcmp_lmem_cls __declspec(lmem) const void *buf1, __declspec(cls) const void *buf2, unsigned int count);	Compares count number of bytes between buf1 in LOCAL MEMORY and buf2 in CLUSTER LOCAL SCRATCH.

2.4.5 memchr()

```
int memchr( __declspec(sram) const void *buf,  
              CHAR c,  
              unsigned int count);
```

Returns offset to the first occurrence of 8-bit character **c** in memory buffer **buf** in SRAM, or -1 if none is found in the first **count** characters. Note that this function returns an offset rather than the address of the character as in standard C library.

Similar functions are provided for operation in all memory regions. Table 2.5 summarizes the prototypes for each of the memory-specific memchr() functions.

Table 2.5. memchr() Prototypes

Prototype	Description
int memchr_sram(__declspec(sram) const void *buf, CHAR c, unsigned int count);	Returns an offset to the first occurrence of c in buf in SRAM.
int memchr_lmem(__declspec(local_mem) const void *buf, CHAR c, unsigned int count);	Returns an offset to the first occurrence of c in buf in LMEM.
int memchr_mem(__declspec(mem) const void *buf, CHAR c, unsigned int count);	Returns an offset to the first occurrence of c in buf in MEM (IMEM/EMEM/CTM).
int memchr_cls(__declspec(cls) const void *buf, CHAR c,	Returns an offset to the first occurrence of c in buf in CLUSTER LOCAL SCRATCH.

Prototype	Description
unsigned int count);	

2.5 string.h

The string.h file provides the same set of string manipulation functions as standard C. As with the functions in memory.h, string function names that do not specify a memory region work on strings in SRAM.

2.5.1 strlen()

```
unsigned int strlen( __declspec(sram) const CHAR *s);
```

Returns length of string **s** in SRAM in bytes.

Similar functions are provided for operation in all memory regions. Table 2.6 summarizes the prototypes for each of the memory-specific strlen() functions.

Table 2.6. strlen() Prototypes

Prototype	Description
unsigned int strlen_sram(__declspec(sram) const CHAR *s);	Returns the length of string s in SRAM.
unsigned int strlen_lmem(__declspec(local_mem) const CHAR *s);	Returns the length of string s in LMEM.
unsigned int strlen_mem(__declspec(mem) const CHAR *s);	Returns the length of string s in MEM (IMEM/EMEM/CTM).
unsigned int strlen_cls(__declspec(cls) const CHAR *s);	Returns the length of string s in CLUSTER LOCAL SCRATCH.

2.5.2 strcmp()

```
int strcmp( __declspec(sram) const CHAR *s1,
            __declspec(sram) const CHAR *s2);
```

Compares two strings in SRAM. Perform unsigned comparison on each CHAR.

Return:

- | | |
|--------------------|------------------------|
| negative value if: | s1 is smaller than s2, |
| 0 if | s1 is equal to s2, and |
| positive value if | s1 is greater than s2. |

Similar functions are provided for operation in all memory regions. Table 2.7 summarizes the prototypes for each of the memory-specific strcmp() functions.

Table 2.7. strcmp() Prototypes

Prototype	Description
unsigned int strcmp_sram(__declspec(sram) const CHAR *s1, __declspec(sram) const CHAR *s2);	Compares string s1 with string s2 in SRAM. This is identical to the strcmp() function
unsigned int strcmp_lmem(__declspec(local_mem) const CHAR *s1, __declspec(local_mem) const CHAR *s2);	Compares string s1 with string s2 in LMEM
unsigned int strcmp_mem(__declspec(mem) const CHAR *s1, __declspec(mem) const CHAR *s2);	Compares string s1 with string s2 in MEM (IMEM/EMEM/CTM)
unsigned int strcmp_cls(__declspec(cls) const CHAR *s1, __declspec(cls) const CHAR *s2);	Compares string s1 with string s2 in CLUSTER LOCAL SCRATCH.

2.5.3 strncmp()

```
int strncmp( __declspec(sram) const CHAR *s1,
             __declspec(sram) const CHAR *s2,
             unsigned int count);
```

Compares two strings in SRAM up to a specified count of characters. Performs unsigned comparison on each CHAR. Cross product is not supported.

Return:

negative value if:	s1 is smaller than s2,
0 if	s1 is equal to s2, and
positive value if	s1 is greater than s2.

Similar functions are provided for operation in all memory regions. Table 2.8 summarizes the prototypes for each of the memory-specific strncmp() functions.

Table 2.8. strncmp() Prototypes

Prototype	Description
int strncmp_sram(__declspec(sram) const CHAR *s1, __declspec(sram) const CHAR *s2, unsigned int count);	Compares count number of bytes in string s1 with string s2 in SRAM. This is identical to the strncmp() function.
int strncmp_lmem(__declspec(local_mem) const CHAR *s1,	Compares count number of bytes in string s1 with string s2 in LMEM.

Prototype	Description
<pre style="margin: 0;">__declspec(local_mem) const CHAR *s2, unsigned int count);</pre>	
<pre style="margin: 0;">int strncmp_mem(__declspec(mem) const CHAR *s1, __declspec(mem) const CHAR *s2, unsigned int count);</pre>	Compares count number of bytes in string s1 with string s2 in MEM (IMEM/EMEM/CTM).
<pre style="margin: 0;">int strncmp_cls(__declspec(cls) const CHAR *s1, __declspec(cls) const CHAR *s2, unsigned int count);</pre>	Compares count number of bytes in string s1 with string s2 in CLUSTER LOCAL SCRATCH.

2.5.4 strcpy()

```
__declspec(sram) CHAR *strcpy( __declspec(sram) CHAR *dest,
__declspec(sram) const CHAR *src);
```

Copies source string **src** in SRAM to string **dest** in SRAM and returns the original **dest**.

Similar functions are provided for operation in all memory regions. Table 2.9 summarizes the prototypes for each of the memory-specific strcpy() functions.

Table 2.9. strcpy() Prototypes

Prototype	Description
<pre style="margin: 0;">__declspec(sram) CHAR * strcpy_sram(__declspec(sram) CHAR *dest, __declspec(sram) const CHAR *src);</pre>	Copies string src to string dest in SRAM. This is identical to the strcpy() function.
<pre style="margin: 0;">__declspec(local_mem) CHAR * strcpy_lmem(__declspec(local_mem) CHAR *dest, __declspec(local_mem) const CHAR *src);</pre>	Copies string src to string dest in LMEM.
<pre style="margin: 0;">__declspec(mem) CHAR * strcpy_mem(__declspec(mem) CHAR *dest, __declspec(mem) const CHAR *src);</pre>	Copies string src to string dest in MEM (IMEM/EMEM/CTM).
<pre style="margin: 0;">__declspec(cls) CHAR * strcpy_cls(__declspec(cls) CHAR *dest, __declspec(cls) const CHAR *src);</pre>	Copies string src to string dest in CLUSTER LOCAL SCRATCH.

2.5.5 `strncpy()`

```
__declspec(sram) CHAR *strncpy( __declspec(sram) CHAR *dest,
                                __declspec(sram) const CHAR *src,
                                unsigned int count);
```

Copies source string **src** in SRAM to string **dest** in SRAM, up to a specified count of characters, and returns the original **dest**.

Similar functions are provided for operation in all memory regions. Table 2.10 summarizes the prototypes for each of the memory-specific `strncpy()` functions.

Table 2.10. `strncpy()` Prototypes

Prototype	Description
<code>__declspec(sram) CHAR * strncpy_sram(__declspec(sram) CHAR *dest, __declspec(sram) const CHAR *src, unsigned int count);</code>	Copies count number of bytes from string src to string dest in SRAM. This is identical to the <code>strncpy()</code> function.
<code>__declspec(local_mem) CHAR * strncpy_lmem(__declspec(local_mem) CHAR *dest, __declspec(local_mem) const CHAR *src, unsigned int count);</code>	Copies count number of bytes from string src to string dest in LMEM.
<code>__declspec(mem) CHAR * strncpy_mem(__declspec(mem) CHAR *dest, __declspec(mem) const CHAR *src, unsigned int count);</code>	Copies count number of bytes from string src to string dest in MEM (IMEM/EMEM/CTM).
<code>__declspec(cls) CHAR * strncpy_cls(__declspec(cls) CHAR *dest, __declspec(cls) const CHAR *src, unsigned int count);</code>	Copies count number of bytes from string src to string dest in CLUSTER LOCAL SCRATCH.

2.5.6 `strcat()`

```
__declspec(sram) CHAR *strcat( __declspec(sram) CHAR *dest,  
                              __declspec(sram) const CHAR *src);
```

Appends string **src** in SRAM to string **dest** in SRAM and returns the original **dest**.

Similar functions are provided for operation in all memory regions. Table 2.11 summarizes the prototypes for each of the memory-specific `strcat()` functions.

Table 2.11. `strcat()` Prototypes

Prototype	Description
<code>__declspec(sram) CHAR * strcat_sram(__declspec(sram) CHAR *dest, __declspec(sram) const CHAR *src);</code>	Appends string src to string dest in SRAM. This is identical to the <code>strcat()</code> function.
<code>__declspec(local_mem) CHAR * strcat_lmem(__declspec(local_mem) CHAR *dest, __declspec(local_mem) const CHAR *src);</code>	Appends string src to string dest in LMEM.
<code>__declspec(mem) CHAR * strcat_mem(__declspec(mem) CHAR *dest, __declspec(mem) const CHAR *src);</code>	Appends string src to string dest in MEM (IMEM/EMEM/CTM).
<code>__declspec(cls) CHAR * strcat_cls(__declspec(cls) CHAR *dest, __declspec(cls) const CHAR *src);</code>	Appends string src to string dest in CLUSTER LOCAL SCRATCH.

2.5.7 `strncat()`

```
__declspec(sram) CHAR *strncat( __declspec(sram) CHAR *dest,  
                                  __declspec(sram) const CHAR *src,  
                                  unsigned int count);
```

Appends string **src** in SRAM to string **dest** in SRAM, up to a specified **count** of characters, and returns the original **dest**.

Similar functions are provided for operation in all memory regions. Table 2.12 summarizes the prototypes for each of the memory-specific `strncat()` functions.

Table 2.12. `strncat()` Prototypes

Prototype	Description
<code>__declspec(sram) CHAR * strncat_sram(__declspec(sram) CHAR *dest, __declspec(sram) const CHAR *src, unsigned int count);</code>	Appends count number of bytes from string src to string dest in SRAM. This is identical to the <code>strncat()</code> function.
<code>__declspec(local_mem) CHAR * strncat_lmem(__declspec(local_mem) CHAR *dest,</code>	Appends count number of bytes from string src to string dest in LMEM.

Prototype	Description
<code style="padding-left: 40px;">__declspec(local_mem) const CHAR *src, unsigned int count);</code>	
<code style="padding-left: 40px;">__declspec(mem) CHAR * strncat_mem(__declspec(mem) CHAR *dest, __declspec(mem) const CHAR *src, unsigned int count);</code>	Appends count number of bytes from string src to string dest in MEM (IMEM/EMEM/CTM).
<code style="padding-left: 40px;">__declspec(cls) CHAR * strncat_cls(__declspec(cls) CHAR *dest, __declspec(cls) const CHAR *src, unsigned int count);</code>	Appends count number of bytes from string src to string dest in CLUSTER LOCAL SCRATCH.

2.5.8 strchr()

```
__declspec(sram) CHAR *strchr( __declspec(sram)  
                                  const CHAR *s, CHAR c);
```

Returns address of the first occurrence of character **c** (could be 0x00) in string **s** in SRAM, or NULL if none.

Similar functions are provided for operation in all memory regions. Table 2.13 summarizes the prototypes for each of the memory-specific strchr() functions.

Table 2.13. strchr() Prototypes

Prototype	Description
<code style="padding-left: 40px;">__declspec(sram) CHAR * strchr_sram(__declspec(sram) CHAR *s, CHAR *c)</code>	Returns the address of the first occurrence of c in string s in SRAM. This is identical to the strchr() function.
<code style="padding-left: 40px;">__declspec(local_mem) CHAR * strcat_lmem(__declspec(local_mem) CHAR *s, CHAR *c);</code>	Returns the address of the first occurrence of c in string s in LMEM.
<code style="padding-left: 40px;">__declspec(mem) CHAR * strchr_mem(__declspec(mem) CHAR *s, CHAR *c);</code>	Returns the address of the first occurrence of c in string s in MEM (IMEM/EMEM/CTM).
<code style="padding-left: 40px;">__declspec(cls) CHAR * strchr_cls(__declspec(cls) CHAR *s, CHAR *c);</code>	Returns the address of the first occurrence of c in string s in CLUSTER LOCAL SCRATCH.

2.5.9 strrchr()

```
__declspec(sram) CHAR *strrchr( __declspec(sram)
                               const CHAR *s, CHAR c);
```

Returns address of the last occurrence of character **c** (could be 0x00) in string **s** in SRAM, or NULL if none.

Similar functions are provided for operation in all memory regions. Table 2.14 summarizes the prototypes for each of the memory-specific strrchr() functions.

Table 2.14. strrchr() Prototypes

Prototype	Description
<code>__declspec(sram) CHAR * strrchr_sram(__declspec(sram) CHAR *s, CHAR *c);</code>	Returns the address of the last occurrence of c in string s in SRAM. This is identical to the strchr() function.
<code>__declspec(local_mem) CHAR * strrcat_lmem(__declspec(local_mem) CHAR *s, CHAR *c);</code>	Returns the address of the last occurrence of c in string s in LMEM.
<code>__declspec(mem) CHAR * strrchr_mem(__declspec(mem) CHAR *s, CHAR *c);</code>	Returns the address of the last occurrence of c in string s in MEM (IMEM/EMEM/CTM).
<code>__declspec(cls) CHAR * strrchr_cls(__declspec(cls) CHAR *s, CHAR *c);</code>	Returns the address of the last occurrence of c in string s in CLUSTER LOCAL SCRATCH.

2.5.10 strstr()

```
__declspec(sram) CHAR *strstr( __declspec(sram) const CHAR *s1,
                             __declspec(sram) const CHAR *s2);
```

Returns address of the first occurrence of sub-string **s2** in string **s1**, both in SRAM, or NULL if none.

Similar functions are provided for operation in all memory regions. Table 2.15 summarizes the prototypes for each of the memory-specific strstr() functions.

Table 2.15. strstr() Prototypes

Prototype	Description
<code>__declspec(sram) CHAR * strstr_sram(__declspec(sram) const CHAR *s1, __declspec(sram) const CHAR *s2);</code>	Returns the address of the first occurrence of substring s2 in string s1 in SRAM. This is identical to the <code>strstr()</code> function.
<code>__declspec(local_mem) CHAR * strstr_lmem(__declspec(local_mem) const CHAR *s1, __declspec(sram) const CHAR *s2);</code>	Returns the address of the first occurrence of substring s2 in string s1 in LMEM.
<code>__declspec(mem) CHAR * strstr_mem(__declspec(mem) const CHAR *s1, __declspec(sram) const CHAR *s2);</code>	Returns the address of the first occurrence of substring s2 in string s1 in MEM (IMEM/EMEM/CTM).
<code>__declspec(cls) CHAR * strstr_cls(__declspec(cls) const CHAR *s1, __declspec(sram) const CHAR *s2);</code>	Returns the address of the first occurrence of substring s2 in string s1 in CLUSTER LOCAL SCRATCH.

2.5.11 strspn()

```
unsigned int strspn( __declspec(sram) const CHAR *s,  
                  __declspec(sram) const CHAR *chset);
```

Returns length of sub-string in string **s** in SRAM that consists entirely of characters in 8-bit character **chset** in SRAM.

Similar functions are provided for operation in all memory regions. Table 2.16 summarizes the prototypes for each of the memory-specific `strspn()` functions.

Table 2.16. strspn() Prototypes

Prototype	Description
<code>unsigned int strspn_sram(__declspec(sram) const CHAR *s, __declspec(sram) const CHAR *chset);</code>	Returns the length of the substring in string s that consists entirely of characters from chset in SRAM. This is identical to the <code>strspn()</code> function.
<code>unsigned int strspn_lmem(__declspec(local_mem) const CHAR *s, __declspec(sram) const CHAR *chset);</code>	Returns the length of the substring in string s that consists entirely of characters from chset in LMEM.

Prototype	Description
unsigned int strspn_mem(__declspec(mem) const CHAR *s, __declspec(sram) const CHAR *chset);	Returns the length of the substring in string s that consists entirely of characters from chset in MEM (IMEM/EMEM/CTM).
unsigned int strspn_cls(__declspec(cls) const CHAR *s, __declspec(sram) const CHAR *chset);	Returns the length of the substring in string s that consists entirely of characters from chset in CLUSTER LOCAL SCRATCH.

2.5.12 strcspn()

```
unsigned int strcspn( __declspec(sram) const CHAR *s,  
                      __declspec(sram) const CHAR *chset);
```

Returns length of sub-string in string **s** in SRAM that consists entirely of characters not in 8-bit character set **chset** in SRAM.

Similar functions are provided for operation in all memory regions. Table 2.17 summarizes the prototypes for each of the memory-specific strcspn() functions.

Table 2.17. strcspn() Prototypes

Prototype	Description
unsigned int strcspn_sram(__declspec(sram) const CHAR *s, __declspec(sram) const CHAR *chset);	Returns the length of the substring in string s that consists entirely of characters not contained in chset in SRAM. This is identical to the strcspn() function.
unsigned int strcspn_lmem(__declspec(local_mem) const CHAR *s, __declspec(sram) const CHAR *chset);	Returns the length of the substring in string s that consists entirely of characters not contained in chset in LMEM.
unsigned int strcspn_mem(__declspec(mem) const CHAR *chset; __declspec(sram) const CHAR *chset);	Returns the length of the substring in string s that consists entirely of characters not contained in chset in MEM (IMEM/EMEM/CTM).
unsigned int strcspn_cls(__declspec(cls) const CHAR *s, __declspec(sram) const CHAR *chset);	Returns the length of the substring in string s that consists entirely of characters not contained in chset in CLUSTER LOCAL SCRATCH.

2.5.13 strpbrk()

```
__declspec(sram) CHAR *strpbrk( __declspec(sram) const CHAR *s,
```

```
__declspec(sram) const CHAR *chset);
```

Returns address to the first character in string **s** that comes from 8-bit character set chset, both in SRAM, or NULL if none.

Similar functions are provided for operation in all memory regions. Table 2.18 summarizes the prototypes for each of the memory-specific strpbrk() functions.

Table 2.18. strpbrk() Prototypes

Prototype	Description
<code>__declspec(sram) CHAR * strpbrk_sram(</code> <code> __declspec(sram) const CHAR *s1,</code> <code> __declspec(sram) const CHAR *s2);</code>	Returns the address of the first character in string s1 that comes from s2 in SRAM. This is identical to the strpbrk() function.
<code>__declspec(local_mem) CHAR * strpbrk_lmem(</code> <code> __declspec(local_mem) const CHAR *s1,</code> <code> __declspec(sram) const CHAR *s2);</code>	Returns the address of the first character in string s that comes from s2 in LMEM.
<code>__declspec(mem) CHAR * strpbrk_mem(</code> <code> __declspec(mem) const CHAR *s1,</code> <code> __declspec(sram) const CHAR *s2);</code>	Returns the address of the first character in string s that comes from s2 in MEM (IMEM/EMEM/CTM).
<code>__declspec(cls) CHAR * strpbrk_cls(</code> <code> __declspec(cls) const CHAR *s1,</code> <code> __declspec(sram) const CHAR *s2);</code>	Returns the address of the first character in string s that comes from s2 in CLUSTER LOCAL SCRATCH.

2.5.14 strtok()

```
__declspec(sram) CHAR *strtok( __declspec(sram) CHAR *s,
                          __declspec(sram) const CHAR *delimit);
```

Returns address to the next token in string **s** delimited by a character from the 8-bit character set delimit, both in SRAM.

Similar functions are provided for operation in all memory regions. Table 2.19 summarizes the prototypes for each of the memory-specific strtok() functions.

Table 2.19. strtok() Prototypes

Prototype	Description
<code>__declspec(sram) CHAR * strtok_sram(</code>	Returns the address to the next token in string s delimited by

Prototype	Description
<code>__declspec(sram) CHAR *s,</code> <code>__declspec(sram) const CHAR *delimit);</code>	a character from delimit in SRAM. This is identical to the strtok() function.
<code>__declspec(local_mem) CHAR * strtok_lmem(</code> <code> __declspec(local_mem) CHAR *s,</code> <code> __declspec(sram) const CHAR *delimit);</code>	Returns the address to the next token in string s delimited by a character from delimit in LMEM.
<code>__declspec(mem) CHAR * strtok_mem(</code> <code> __declspec(mem) CHAR *s,</code> <code> __declspec(sram) const CHAR *delimit);</code>	Returns the address to the next token in string s delimited by a character from delimit in MEM (IMEM/EMEM/CTM).
<code>__declspec(cls) CHAR * strtok_cls(</code> <code> __declspec(cls) CHAR *s,</code> <code> __declspec(sram) const CHAR *delimit);</code>	Returns the address to the next token in string s delimited by a character from delimit in CLUSTER LOCAL SCRATCH.

2.5.15 strtol()

```
long strtol( __declspec(sram) const CHAR *s,
              __declspec(sram) CHAR **ps_end, int base);
```

Converts string **s** in SRAM to a long integer using specified base (between 2 to 36, or 0 to interpret string in C-style). Returns LONG_MAX or LONG_MIN if overflow or underflow, respectively. Adjusts pointer ***ps_end** to the first character that causes conversion to fail.

Similar functions are provided for operation in all memory regions. Table 2.20 summarizes the prototypes for each of the memory-specific strtol() functions.

Table 2.20. strtol() Prototypes

Prototype	Description
<code>long strtol_sram(__declspec(sram) const CHAR *s,</code> <code> __declspec(sram) CHAR **ps_end,</code> <code> int base);</code>	Converts string s to a long integer using the specified base in SRAM. This is identical to the strtol() function.
<code>long strtol_lmem(__declspec(local_mem) const CHAR *s,</code> <code> __declspec(local_mem) CHAR **ps_end,</code> <code> int base);</code>	Converts string s to a long integer using the specified base in LMEM.
<code>long strtol_mem(__declspec(mem) const CHAR *s,</code> <code> __declspec(mem) CHAR **ps_end,</code> <code> int base);</code>	Converts string s to a long integer using the specified base in MEM (IMEM/EMEM/CTM).

Prototype	Description
long strtol_cls(__declspec(cls) const CHAR *s, __declspec(cls) CHAR **ps_end, int base);	Converts string s to a long integer using the specified base in CLUSTER LOCAL SCRATCH.

2.5.16 strtoul()

```
unsigned long strtoul( __declspec(sram) const CHAR *s,
                  __declspec(sram) CHAR **ps_end, int base);
```

Converts string **s** in SRAM to an unsigned long integer using specified base (between 2 to 36, or 0 to interpret string in C-style). Returns ULONG_MAX if overflow. Adjusts pointer *ps_end to the first character that causes conversion to fail.

Similar functions are provided for operation in all memory regions. Table 2.21 summarizes the prototypes for each of the memory-specific strtoul() functions.

Table 2.21. strtoul() Prototypes

Prototype	Description
long strtoul_sram(__declspec(sram) const CHAR *s, __declspec(sram) CHAR **ps_end, int base);	Converts string s to an unsigned long integer using the specified base in SRAM. This is identical to the strtoul() function.
long strtoul_lmem(__declspec(local_mem) const CHAR *s, __declspec(local_mem) CHAR **ps_end, int base);	Converts string s to an unsigned long integer using the specified base in LMEM.
long strtoul_mem(__declspec(mem) const CHAR *s, __declspec(mem) CHAR **ps_end, int base);	Converts string s to an unsigned long integer using the specified base in MEM (IMEM/EMEM/CTM).
long strtoul_cls(__declspec(cls) const CHAR *s, __declspec(cls) CHAR **ps_end, int base);	Converts string s to an unsigned long integer using the specified base in CLUSTER LOCAL SCRATCH.

3. Intrinsic Functions

Intrinsic functions support features of the Netronome NFP-6xxx Network Processors that are not easily accessible using standard C. They are either expanded in-line in the compiler or defined and inlined in intrinsics. Also defined are many enumerated data types and structures to encapsulate signal and CSR names or other such special information.

The header `nfp.h` defines the C compiler intrinsic functions. The descriptions here are not intended to be complete descriptions of the intrinsic functions—they must be supplemented with the information in the *Netronome Network Flow Processor 6xxx Databook*.

There are restrictions placed on the data argument specified to CSR and intrinsic functions. These restrictions are discussed throughout this chapter and in Section 3.34.

The intrinsic functions fall into the following categories:

- ARM Intrinsics
- Cluster Local Scratch Intrinsics
- Cluster Local Scratch Ring Intrinsics
- Cluster Local Scratch Reflect Intrinsics
- Cluster Target Intrinsics
- Control and Status Register Access Intrinsics
- CRC Intrinsics
- Crypto Intrinsics
- MEM Intrinsics
- MEM WorkQ Intrinsics
- MEM Ring Intrinsics
- MEM Ticket Intrinsics
- MEM LockQ Intrinsics
- MEM Lock Intrinsics
- MEM List Intrinsics
- MEM MicroQ Intrinsics
- MEM CAM Intrinsics
- MEM TCAM Intrinsics
- MEM Packet Engine Intrinsics
- MEM Statistics Intrinsics
- MEM Load Balancing Intrinsics
- MEM Lookup Intrinsics
- PCI Intrinsics

- ILA Intrinsic
- NBI Intrinsic
- SRAM Intrinsic
- Synchronization Intrinsic
- Unaligned Data Access Intrinsic
- Miscellaneous Intrinsic
- Math Intrinsic

Intrinsic functions are known to the compiler and do not generate a function call. They generally translate to one or two machine instructions, excluding evaluation of the arguments.

3.1 Intrinsic Syntax Conventions

The intrinsic functions described in this chapter can be used in NFP-32xx Indirect Reference Mode or NFP-6xxx Indirect Reference Mode or both modes. When the mode is not indicated both modes apply.

The data for memory and I/O operations is generally one or more 32-bit or 64-bit words. The arguments for the data are specified as `void *` to allow you to pass any structure or array of the appropriate size for these operations. There are restrictions placed upon the data argument as described at the end of this chapter (refer to Section 3.34). These restrictions are needed to satisfy the transfer register operand restrictions imposed by the microcode underlying the intrinsic and to deal with the asynchronous programming model exposed by the microcode. Take special care to properly call the `__free_write_buffer()` intrinsic when performing an asynchronous memory write operation (i.e. a write that waits on “sig_done”). Without a call to this intrinsic, the compiler may prematurely reuse the write transfer registers involved in the operation before the write has completed.

For those intrinsics that accept `SIGNAL_PAIR`, the sync argument must be `sig_done`; therefore, the caller must wait for the signal using `__wait_for_all()`, `__wait_for_any()`, or `signal_test()`, etc. Also, asynchronous memory reads may require the use of `__implicit_read()` if not all the data is used.

3.1.1 Transfer Register Modifiers

The following `__declspec` modifiers are used in most memory and CSR accessing intrinsics to help you cope with the restrictions detailed in Section 3.34. You receive an error if you do not specify a `declspec` modifier to an intrinsic that expects one. If you pass a parameter not annotated with the appropriate `__declspec` modifier, the compiler gives both a warning and an error. For example, if an intrinsic is expecting a `read_reg` and you pass it a `write_reg`, you receive a warning and a compiler error.

There is no distinction between any transfer registers in NFP-6xxx. Therefore, all xfer registers should be as follows:

```
__declspec (read_reg)
__declspec (write_reg)
```

Some intrinsics expect constant parameters (i.e., count, sync). For count, the compiler tries to generate an indirect reference if a non-constant is passed and will run with a loss of performance. This will also occur for count values greater than 8 as the direct form of instructions only support count values up to 8.

For sync or csr, etc., however, the compiler reports an error if an intrinsic that expects a constant does not receive one.

3.1.2 Limitations on Some I/O Functions

Some hardware instructions only take one operand for both source and destination transfer register operand, but the intrinsic functions provide two separate parameters. These functions are:

- sram_swap
- sram_swap_ind
- sram_test_and_clear_bits
- sram_test_and_set_bits
- sram_test_and_add
- sram_put_ring
- cls_swap
- cls_test_and_clear_bits
- cls_test_and_set_bits
- cls_test_and_add
- cls_test_and_sub

The compiler internally maps both the read and write transfer registers to the same physical register number. As a result, certain usage of these functions is not legal. For example:

```
__declspec(read_reg) int r1[2], r2, r3;
__declspec(write_reg) int w1, w2, w3;
__declspec(sram) int p;
SIGNAL_PAIR s1;

...
    sram_swap(&r1[0], &w1, &p, sig_done, &s1);           // r1[0] and w1 are mapped to the
    //same physical register number
    __wait_for_all(&s1);
    sram_swap(&r1[1], &w1, &p, sig_done, &s1);           // Error: r1[1] and w1 can not be
    //mapped to the same physical
    //register number

    __wait_for_all(&s1);
...
```

Another case:

```
sram_swap(&r2, &w1, &p, ctx_swap, &s1);           // r2 and w1 are mapped to the same
                                                    // physical register number
__wait_for_all(&s1);
sram_swap(&r3, &w1, &p, ctx_swap, &s1);           // r3 and w1 are mapped to the same
                                                    // physical register number
__wait_for_all(&s1);
__asm alu[X, --, B, r2]                          // Error: the second sram_swap corrupted
                                                    // r2 because r2 and r3 are mapped to the same
                                                    // physical register.
```

3.2 NFP-32xx Indirect Reference Formats

This section describes the C language unions and types enabling the manipulation of the indirect words for NFP-32xx indirect format. This compiler option is enabled when indirect reference mode (compile time directive) is set to `-indirect_ref_format_nfp32xx`.

3.2.1 Indirect Reference 32xx Unions

3.2.1.1 override_all_ind_t

Override all fields in NFP-32xx indirect format.

Table 3.1. union override_all_ind_t

Type	Name	Description
unsigned int	encoding:2	0x3.
unsigned int	reserved:1	Reserved.
unsigned int	byte_mask:8	Byte mask.
unsigned int	sig_ctx:3	Signal context.
unsigned int	master:5	Signal and data master.
unsigned int	xadd_imm:8	Transfer register or immediate.
unsigned int	ref_count:5	Number of bytes, long/quad-words to access.
unsigned int	value	Accessor to all bits simultaneously.

3.2.1.2 override_cnt_ind_t

Override length only in NFP-32xx indirect format.

Table 3.2. union override_cnt_ind_t

Type	Name	Description
unsigned int	encoding:4	0x2.

Type	Name	Description
unsigned int	reserved:23	Reserved.
unsigned int	ref_count:5	Number of bytes, long/quad-words to access.
unsigned int	value	Accessor to all bits simultaneously.

3.2.1.3 override_cnt_mask_ind_t

Override length and byte mask in NFP-32xx indirect format.

Table 3.3. union override_cnt_mask_ind_t

Type	Name	Description
unsigned int	encoding:4	0x3.
unsigned int	reserved:15	Reserved.
unsigned int	ref_count:5	Number of bytes, long/quad-words to access.
unsigned int	byte_mask:8	Mask of bytes to write.
unsigned int	value	Accessor to all bits simultaneously.

3.2.1.4 override_mask_ind_t

Override byte mask only in NFP-32xx indirect format.

Table 3.4. union override_mask_ind_t

Type	Name	Description
unsigned int	encoding:4	0x4.
unsigned int	reserved:20	Reserved.
unsigned int	byte_mask:8	Mask of bytes to write.
unsigned int	value	Accessor to all bits simultaneously.

3.2.1.5 override_me_ctx_ind_t

Override ME and context in NFP-32xx indirect format.

Table 3.5. union override_me_ctx_ind_t

Type	Name	Description
unsigned int	encoding:4	0x5.
unsigned int	reserved_1:1	Reserved.
unsigned int	master:8	Address of signal or data master.
unsigned int	ctx:3	Context.
unsigned int	reserved_2:16	Reserved.

Type	Name	Description
unsigned int	value	Accessor to all bits simultaneously.

3.2.1.6 override_me_ctx_sig_ind_t

Override ME, context and signal in NFP-32xx indirect format.

Table 3.6. union override_me_ctx_sig_ind_t

Type	Name	Description
unsigned int	encoding:4	0x6.
unsigned int	reserved_1:1	Reserved.
unsigned int	master:8	Address of signal or data master.
unsigned int	ctx:3	Context.
unsigned int	sig:4	Signal number.
unsigned int	reserved_2:12	Reserved.
unsigned int	value	Accessor to all bits simultaneously.

3.2.1.7 override_me_xfer_ctx_cnt_ind_t

Override ME, Xfer register, context and length in NFP-32xx indirect format.

Table 3.7. union override_me_xfer_ctx_cnt_ind_t

Type	Name	Description
unsigned int	encoding:4	0x7
unsigned int	reserved_1:1	Reserved.
unsigned int	master:8	Address of signal or data master.
unsigned int	ctx:3	Context.
unsigned int	reserved_2:3	Reserved.
unsigned int	xadd_imm:8	Transfer register or immediate.
unsigned int	ref_count:5	Number of bytes, long/quad-words to access.
unsigned int	value	Accessor to all bits simultaneously.

3.2.1.8 override_me_xfer_ctx_ind_t

Override ME, Xfer register and context in NFP-32xx indirect format.

Table 3.8. union override_me_xfer_ctx_ind_t

Type	Name	Description
unsigned int	encoding:4	0xA or 0xB.

Type	Name	Description
unsigned int	reserved_1:1	Reserved.
unsigned int	master:8	Address of signal or data master.
unsigned int	xadd_imm:14	Transfer register or immediate.
unsigned int	reserved_2:2	Reserved.
unsigned int	ctx:3	Context.
unsigned int	value	Accessor to all bits simultaneously.

3.2.1.9 override_me_xfer_imm_ind_t

Override ME and Xfer register in NFP-32xx indirect format.

Table 3.9. union override_me_xfer_imm_ind_t

Type	Name	Description
unsigned int	encoding:4	0x8 or 0x9.
unsigned int	reserved_1:1	Reserved.
unsigned int	master:8	Address of signal or data master.
unsigned int	xadd_imm:14	Transfer register or immediate.
unsigned int	reserved_2:5	Reserved.
unsigned int	value	Accessor to all bits simultaneously.

3.2.1.10 override_xfer_imm_cnt_ind_t

Override Xfer register and length in NFP-32xx indirect format.

Table 3.10. union override_xfer_imm_cnt_ind_t

Type	Name	Description
unsigned int	encoding:4	0x1.
unsigned int	reserved_1:9	Reserved.
unsigned int	xadd_imm:14	Transfer register or immediate.
unsigned int	ref_count:5	Number of bytes, long/quad-words to access.
unsigned int	value	Accessor to all bits simultaneously.

3.2.1.11 override_xfer_imm_ind_t

Override Xfer register only in NFP-32xx indirect format.

Table 3.11. union override_xfer_imm_ind_t

Type	Name	Description
unsigned int	encoding:4	0x0.
unsigned int	reserved_1:9	Reserved.
unsigned int	xadd_imm:14	Transfer register or immediate.
unsigned int	reserved_2:5	Reserved.
unsigned int	value	Accessor to all bits simultaneously.

3.3 NFP-6xxx Indirect Reference Formats

This section describes the C language unions and types enabling the manipulation of the indirect words for NFP-6xxx indirect format. This compiler option is enabled by default.

3.3.1 Indirect Reference 6xxx Unions

3.3.1.1 override_alu_ind_t

Override ALU Preset value in NFP-6xxx indirect format.

Table 3.12. union override_alu_ind_t

Type	Name	Description
unsigned int	data16:16	Override data reference or byte mask if data reference is not used.
unsigned int	reserved:1	Reserved.
unsigned int	override_signal_ctx:1	Override signal context.
unsigned int	override_signal_num:1	Override signal number.
unsigned int	ref_count:5	Number of bytes, long/quad-words to access.
unsigned int	override_length:1	Override length.
unsigned int	override_bytemask_csr:1	Override byte mask in IND_CSR.
unsigned int	override_data:3	Override data. 0x01: Full data reference. 0x02: Immediate data. 0x03: Data context. 0x04: Data and signal context. 0x05: Per context offset of data reference. 0x06: Byte mask.
unsigned int	override_master:2	Override master. 0x01: Island and master. 0x02: Island, data master, signal master. 0x03: Island.

Type	Name	Description
unsigned int	override_signal_master:1	Override signal master.
unsigned int	value	Accessor to all bits simultaneously.

3.3.1.2 override_csr_ind_t

This union provides overriding using CSR Indirect in NFP-6xxx indirect format.

Table 3.13. union override_csr_ind_t

Type	Name	Description
unsigned int	reserved_1:2	Reserved.
unsigned int	island:6	Override island when field is set in override_alu_ind_t.
unsigned int	master:4	Override data master when field is set in override_alu_ind_t.
unsigned int	signal_master:4	Override signal master when field is set in override_alu_ind_t.
unsigned int	signal_ctx:3	Override signal context when field is set in override_alu_ind_t.
unsigned int	signal_num:4	Override signal number when field is set in override_alu_ind_t.
unsigned int	reserved_2:1	Reserved.
unsigned int	byte_mask:8	Override byte mask when field is set in override_alu_ind_t.
unsigned int	value	Accessor to all bits simultaneously.

3.4 NFP Indirect Reference in General

This section describes the enumerations and functions applicable to NFP-32xx and NFP-6xxx indirect format.

3.4.1 Indirect Reference General Enumerations

3.4.1.1 ovr_field_t

Override field specifier enum.

This enum is used by `ovr_init()` and `ovr_set()` to specify which fields to override or set

Table 3.14. enum ovr_field_t

Name	Description
ovr_me	Override ME.

Name	Description
	NFP-32xx indirect ref mode only. In NFP-32xx indirect ref mode bit 3 of the value is set in <code>ovr_set()</code> to ensure the value is interpreted as a ME number when placed on the bus.
<code>ovr_xfer</code>	Override transfer register. NFP-32xx indirect ref mode only
<code>ovr_ctx</code>	Override context. NFP-32xx indirect ref mode only
<code>ovr_data_master</code>	Override data master. NFP-32xx indirect ref mode only
<code>ovr_signal_and_data_master</code>	Override signal and data master only. NFP-32xx indirect ref mode only
<code>ovr_byte_mask</code>	Override bytemask. In NFP-6xxx indirect ref mode the bytemask is in Indirect CSR
<code>ovr_length</code>	Override the count/length.
<code>ovr_signal_ctx</code>	Override signal context.
<code>ovr_signal_master</code>	Override signal master.
<code>ovr_signal_number</code>	Override signal number.
<code>ovr_data_full_ref</code>	Override the full data reference. NFP-6xxx indirect ref mode only
<code>ovr_data_16bit_imm</code>	Override 16-bit immediate data. NFP-6xxx indirect ref mode only
<code>ovr_data_ctx</code>	Override data context only. NFP-6xxx indirect ref mode only
<code>ovr_data_and_signal_ctx</code>	Override data and signal contexts from DATA16. NFP-6xxx indirect ref mode only
<code>ovr_data_ctx_offset</code>	Override per-context offset of the data reference. NFP-6xxx indirect ref mode only
<code>ovr_data_byte_mask</code>	Override byte mask from DATA16. NFP-6xxx indirect ref mode only
<code>ovr_island_and_data_master</code>	Override island and data master. NFP-6xxx indirect ref mode only

Name	Description
ovr_island_master	Override island master. NFP-6xxx indirect ref mode only
ovr_signal_island_and_data_master	Override signal, island and data master. NFP-6xxx indirect ref mode only

3.4.2 Indirect Reference 6xxx Structs

3.4.2.1 generic_ind_t

In **NFP-32xx** indirect format, `generic_ind_t` is a union of all the indirect qualifiers along with an unsigned int.

This enables one to set the entire indirect word with an integer assignment as opposed to setting each field individually. See example of the code below where length and bytemask is overridden.

```
{
    SIGNAL          sig;
    generic_ind_t   ind;

    unsigned int write_data = 0x12;
    unsigned int len = 11;                                // NOTE: ovr_length zero indexed
    ...
    ovr_init(&ind, ovr_byte_mask | ovr_length);
    ovr_set(&ind, ovr_length, len - 1);                  // override length in previous ALU result
    ovr_set(&ind, ovr_byte_mask, write_data);             // override bytemask in previous ALU result

    __asm
    {
        alu[--, --, B, ind]
        mem[test_add_imm, rd_back, address, 0, --], sig_done[sig], num_sigs[1], indirect_ref
    }
    __wait_for_all(&sig);
}
```

In **NFP-6xxx** indirect format, `generic_ind_t` has two words for indirect format. Previous ALU result as well as Indirect CSR can be used to override. Each field that is overridden has its own enable bit. Any combination of fields can be overridden. Override any combination of fields, where each field has its own enable bit. The previous ALU result covers the most commonly used override fields and the remaining override fields are specified by the Indirect CSR. To access the ALU indirect word or CSR indirect word, use `ALU_INDIRECT_TYPE()` and `CSR_INDIRECT_TYPE()`. See the code below.



Note

`ovr_byte_mask` overrides the bytemask from the indirect CSR. Use `ovr_data_byte_mask` to override from ALU indirect word

```
{
    SIGNAL          sig;
    generic_ind_t   ind;

    unsigned int write_data = 0x12;
    unsigned int length = 11;           // NOTE: ovr_length zero indexed
    ...
    ovr_init(&ind, ovr_byte_mask | ovr_length);
    ovr_set(&ind, ovr_length, len - 1);      // override length in previous ALU result
    ovr_set(&ind, ovr_byte_mask, write_data); // overrides the bytemask from the indirect CSR.

    __asm
    {
        local_csr_wr[CMD_INDIRECT_REF_0, CSR_INDIRECT_TYPE(ind)]
        alu[--, --, B, ALU_INDIRECT_TYPE(ind)]
        mem[test_add_imm, rd_back, address, 0, --], sig_done[sig], num_sigs[1], indirect_ref
    }
    __wait_for_all(&sig);
}
```

Table 3.15. struct generic_ind_t

Type	Name	Description
override_all_ind_t	all_ind	Override all fields in NFP 32xx indirect format.
override_xfer_imm_ind_t	xfer_ind	Override Xfer register only in NFP 32xx indirect format.
override_xfer_imm_cnt_ind_t	xfer_cnt_ind	Override Xfer register and length in NFP 32xx indirect format.
override_cnt_mask_ind_t	cnt_mask_ind	Override length and byte mask in NFP 32xx indirect format.
override_cnt_ind_t	cnt_ind	Override length only in NFP 32xx indirect format.
override_mask_ind_t	mask_ind	Override byte mask in NFP 32xx indirect format.
override_me_ctx_ind_t	me_ctx_ind	Override ME and context in NFP 32xx indirect format.
override_me_ctx_sig_ind_t	me_ctx_sig_ind	Override ME, context and signal in NFP 32xx indirect format.
override_me_xfer_imm_ind_t	me_xfer_ind	Override ME and Xfer register in NFP 32xx indirect format.
override_me_xfer_ctx_ind_t	me_xfer_ctx_ind	Override ME, Xfer register and context in NFP 32xx indirect format.

Type	Name	Description
override_me_xfer_ctx_cnt_ind_t	me_xfer_ctx_cnt_ind	Override ME, Xfer register, context and length in NFP 32xx indirect format.
unsigned int	value	
override_alu_ind_t	alu_ind	ALU result in NFP-6xxx indirect reference format.
override_csr_ind_t	csr_ind	Indirect CSR word used in NFP-6xxx indirect reference mode.

3.4.3 Indirect Reference General Functions

3.4.3.1 ovr_init

Prototype:

```
void ovr_init(generic_ind_t* ind, unsigned int fields)
```

Description:

Initialize an override word.

This function is used to specify which fields the user want to override with an indirect word. The encoding to use is determined from the fields specified and subsequent calls to `ovr_set()` only allows the corresponding field values to be set. NFP-32xx: An exception is `ovr_me` and `ovr_signal_and_data_master`. In NFP-32xx indirect reference mode `ovr_me` can be set when `ovr_signal_and_data_master` is specified in `ovr_init()` or vice versa. Specifying `ovr_me` in `ovr_set` will force bit 3 to be set to ensure the value placed on the bus is interpreted as an ME number for NFP-32xx indirect reference mode.

A couple of examples are given below which illustrates the usage of `ovr_init()` and `ovr_set()`. The first example creates an override word which will correctly override the byte mask and length in both NFP-32xx and NFP-6xxx indirect reference mode.

```
{
    __xwrite unsigned int xfer_wr[16];
    SIGNAL      sig;
    unsigned int length = 15;
    generic_ind_t ind;      // NFP-32xx: this is the union of all override fields
                           // NFP-6xxx: this is ALU override word and CSR indirect word
    ...
    ovr_init(&ind, ovr_byte_mask | ovr_length);
    ovr_set(&ind, ovr_length, length - 1);           // NOTE: zero indexed
    ovr_set(&ind, ovr_byte_mask, 0x0f);

    mem_write_ind((void*)&xfer_wr, 0x100, length, ind, sig_done, &sig);
}
```

```
    __wait_for_all(&sig);
}
```

In NFP-6xxx indirect reference format any combination of fields can be overridden. A few examples are given below.

The example below shows how to override the island, signal and data master in NFP-6xxx indirect reference mode.

```
{
    SIGNAL                                signal;
    generic_ind_t                         indirect;
    __cls_n(32) __addr40 unsigned int   address[2];
    volatile __xrw unsigned int          xfer_read_write[2];
    unsigned int                           override_island = 35; // i25
    unsigned int                           override_me = 4;      // ME 4

    xfer_read_write[0] = 0x11223344;
    xfer_read_write[1] = 0x44332211;

    cls_write_ptr40((void *)xfer_read_write, address, 2, sig_done, &signal);
    wait_for_all(&signal);

    ovr_init(&indirect, ovr_island_and_data_master | ovr_signal_master);
    ovr_set(&indirect, ovr_island_and_data_master, override_island << 4 | override_me + 4);
    ovr_set(&indirect, ovr_signal_master, override_me + 4 );

    cls_read_ind_ptr40((void *)xfer_read_write, address, 2, indirect, sig_done, &signal);
}
```

The example below shows how to override the length and signal number in NFP-6xxx indirect reference mode.

```
{
    generic_ind_t                         add_sat_ind;
    SIGNAL                                override_signal;
    unsigned int                           length = 12;
    volatile __xread unsigned int          xfer_read[12];
    volatile __xwrite unsigned int         xfer_write[12];

    ovr_init(&add_sat_ind, ovr_length | ovr_signal_number);
    ovr_set(&add_sat_ind, ovr_length, length - 1);
    ovr_set(&add_sat_ind, ovr_signal_number, __signal_number(&override_signal));

    cls_test_and_add_sat_ind_ptr40(xfer_read, xfer_write, address, length, add_sat_ind, sig_done, &signal);
    wait_for_all(&override_signal);
}
```

Table 3.16. ovr_init parameters

Type	Name	Description
generic_ind_t*	<i>ind</i>	NFP-32xx: Pointer to indirect word being constructed. NFP-6xxx: Pointer to two words: ALU and CSR indirect.
unsigned int	<i>fields</i>	Fields to override. It is specified by OR-ing the values from ovr_field_t.

3.4.3.2 ovr_set

Prototype:

```
void ovr_set(generic_ind_t* ind, ovr_field_t field, int value)
```

Description:

Set the value of an override field.

Table 3.17. ovr_set parameters

Type	Name	Description
generic_ind_t*	<i>ind</i>	Xfer register(s) to store data read from Memory
ovr_field_t	<i>field</i>	Field to set.
int	<i>value</i>	The value of the overridden field.

See Also:

- `ovr_init()`

3.5 ARM Intrinsics

This section discusses the ARM intrinsic functions available for data transfer to and from the ARM.

3.5.1 ARM Functions

3.5.1.1 arm_read

Prototype:

```
void arm_read(__xread void* data, volatile void __sram* address, unsigned int count,  
sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read data from ARM into transfer registers.

Table 3.18. arm_read parameters

Type	Name	Description
<code>__xread void*</code>	<i>data</i>	Address to read data into
<code>volatile void __sram*</code>	<i>address</i>	Address to read data from

Type	Name	Description
unsigned int	<i>count</i>	Number of 32-bit words to read
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.5.1.2 arm_read_ind

Prototype:

```
void arm_read_ind(__xread void* data, volatile void __sram* address, unsigned int max_nn,
generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read data from ARM into transfer register indirect mode.

Table 3.19. arm_read_ind parameters

Type	Name	Description
__xread void*	<i>data</i>	Address to read data into
volatile void __sram*	<i>address</i>	Address to read data from
unsigned int	<i>max_nn</i>	Maximum number of 32-bit words to read
generic_ind_t	<i>ind</i>	Indirect word
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.5.1.3 arm_write

Prototype:

```
void arm_write(__xwrite void* data, volatile void __sram* address, unsigned int count,
sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write data to ARM from transfer register.

Table 3.20. arm_write parameters

Type	Name	Description
__xwrite void*	<i>data</i>	Data to write
volatile void __sram*	<i>address</i>	Address to write data to
unsigned int	<i>count</i>	Number of 32-bit words to write
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)

Type	Name	Description
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.5.1.4 arm_write_ind

Prototype:

```
void arm_write_ind(__xwrite void* data, volatile void __sram* address, unsigned int
max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write data to ARM from transfer register indirect mode.

Table 3.21. arm_write_ind parameters

Type	Name	Description
__xwrite void*	<i>data</i>	Data to write
volatile void __sram*	<i>address</i>	Address to write data to
unsigned int	<i>max_nn</i>	Maximum number of 32-bit words to write
generic_ind_t	<i>ind</i>	Indirect word
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.6 Cluster Local Scratch Intrinsics

This section discusses Cluster Local Scratch functions.

3.6.1 Cluster Local Scratch Enumerations

3.6.1.1 CLS_METER_COLOR

Color for CLS Metering operations.

Table 3.22. enum CLS_METER_COLOR

Name	Description
CLS_METER_COLOR_GREEN	Packet is green.
CLS_METER_COLOR_YELLOW	Packet is yellow.
CLS_METER_COLOR_RED	Packet is red.

3.6.1.2 CLS_METER_RFC

RFC type to use for CLS Metering operations.

Table 3.23. enum CLS_METER_RFC

Name	Description
CLS_METER_RFC2698	RFC 2698 defines one method for two rate three color metering.
CLS_METER_RFC2697	RFC 2697 defines one method for two rate three color metering.
CLS_METER_RFC4115	RFC 4115 defines second method for two rate three color metering.

3.6.2 Cluster Local Scratch Typedefs

3.6.2.1 CLS_METER_COLOR

Color for CLS Metering operations.

Table 3.24. typedef CLS_METER_COLOR

Type	Definition
CLS_METER_COLOR	enum CLS_METER_COLOR

3.6.2.2 CLS_METER_RFC

RFC type to use for CLS Metering operations.

Table 3.25. typedef CLS_METER_RFC

Type	Definition
CLS_METER_RFC	enum CLS_METER_RFC

3.6.3 Cluster Local Scratch Functions

3.6.3.1 cls_read_be_ptr32

Prototype:

```
void cls_read_be_ptr32(__xread void* data, volatile void __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read 32-bit words from Cluster Local Scratch in Big Endian mode.

Table 3.26. `cls_read_be_ptr32` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Read Transfer Registers the read data will be placed in
<code>volatile void __cls*</code>	<code>address</code>	Address to read from
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to read
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

See Also:

- `cls_read_ptr32()`

3.6.3.2 `cls_read_be_ptr40`

Prototype:

```
void cls_read_be_ptr40(__xread void* data, volatile void __addr40 __cls* address, unsigned
int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read 32-bit words from 40 bit Cluster Local Scratch in Big Endian mode.

Table 3.27. `cls_read_be_ptr40` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Read Transfer Registers the read data will be placed in
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address to read from where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to read
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

See Also:

- `cls_read_ptr40()`

3.6.3.3 `cls_read_le_ptr32`

Prototype:

```
void cls_read_le_ptr32(__xread void* data, volatile void __cls* address, unsigned int
count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read 32-bit words from Cluster Local Scratch in Little Endian mode.

Table 3.28. `cls_read_le_ptr32` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Read Transfer Registers the read data will be placed in
<code>volatile void __cls*</code>	<code>address</code>	Address to read from
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to read
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

See Also:

- `cls_read_ptr32()`

3.6.3.4 `cls_read_le_ptr40`

Prototype:

```
void cls_read_le_ptr40(__xread void* data, volatile void __addr40 __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read 32-bit words from (40 bit) Cluster Local Scratch in Little Endian mode.

Table 3.29. `cls_read_le_ptr40` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Read Transfer Registers the read data will be placed in
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address to read from where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to read
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

See Also:

- `cls_read_ptr40()`

3.6.3.5 `cls_read_ptr32`

Prototype:

```
void cls_read_ptr32(__xread void* data, volatile void __cls* address, unsigned int count,
sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read 32-bit words from Cluster Local Scratch.

Read count number of 32-bit words from Cluster Local Scratch from address and place them in the transfer registers pointed to by data. The signal sig_ptr will be raised on completion and sync determines if the operation switch context or completes asynchronously.

The address is a byte address and does not need to be naturally aligned. However, LW aligned reads will be faster.

Data read will be returned either in Little Endian or Big Endian depending on the compiler setting.



Note

If count is not a constant, the compiler will automatically generate an instruction with an indirect_ref optional token.

Table 3.30. `cls_read_ptr32` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Read Transfer Registers the read data will be placed in
<code>volatile void __cls*</code>	<code>address</code>	Address to read from
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to read
<code>sync_t</code>	<code>sync</code>	Type of synchronization (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

3.6.3.6 `cls_read_ptr40`

Prototype:

```
void cls_read_ptr40(__xread void* data, volatile void __addr40 __cls* address, unsigned
int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read 32-bit words from 40 bit Cluster Local Scratch.

Read count number of 32-bit words from Cluster Local Scratch from address and place them in the transfer registers pointed to by data. The signal sig_ptr will be raised on completion and sync determines if the operation switch context or completes asynchronously.

The address is a byte address and does not need to be naturally aligned. However, LW aligned reads will be faster.

Data read will be returned either in Little Endian or Big Endian depending on the compiler setting.



Note

If count is not a constant, the compiler will automatically generate an instruction with an indirect_ref optional token.

Table 3.31. `cls_read_ptr40` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Read Transfer Registers the read data will be placed in
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address to read from where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to read
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

3.6.3.7 `cls_write_be_ptr32`

Prototype:

```
void cls_write_be_ptr32(__xwrite void* data, volatile void __addr32 __cls* address,
unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write 32-bit words to Cluster Local Scratch in Big Endian mode.

Table 3.32. `cls_write_be_ptr32` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Write Transfer Registers to be written
<code>volatile void __addr32 __cls*</code>	<code>address</code>	Address to write to
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to written
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

See Also:

- `cls_write_ptr32()`

3.6.3.8 `cls_write_be_ptr40`

Prototype:

```
void cls_write_be_ptr40(__xwrite void* data, volatile void __addr40 __cls* address,
unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write 32-bit words to 40 bit Cluster Local Scratch in Big Endian mode.

Table 3.33. `cls_write_be_ptr40` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Write Transfer Registers to be written
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address to write to where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to written
<code>sync_t</code>	<code>sync</code>	Type of synchronization (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

See Also:

- `cls_write_ptr32()`

3.6.3.9 `cls_write_le_ptr32`

Prototype:

```
void cls_write_le_ptr32(__xwrite void* data, volatile void __cls* address, unsigned int
count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write 32-bit words to Cluster Local Scratch in Little Endian mode.

Table 3.34. `cls_write_le_ptr32` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Write Transfer Registers to be written
<code>volatile void __cls*</code>	<code>address</code>	Address to write to
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to be written
<code>sync_t</code>	<code>sync</code>	Type of synchronization (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

See Also:

- `cls_write_ptr32()`

3.6.3.10 cls_write_le_ptr40

Prototype:

```
void cls_write_le_ptr40(__xwrite void* data, volatile void __addr40 __cls* address,
unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write 32-bit words to 40 bit Cluster Local Scratch in Little Endian mode.

Table 3.35. cls_write_le_ptr40 parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Write Transfer Registers to be written
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address to write to where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to be written
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

See Also:

- `cls_write_ptr40()`

3.6.3.11 cls_write8_be_ind_ptr40

Prototype:

```
void cls_write8_be_ind_ptr40(__xwrite void* data, volatile void __addr40 __cls* address,
unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write bytes to Cluster Local Scratch in Big Endian mode in indirect mode.

Table 3.36. cls_write8_be_ind_ptr40 parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Write Transfer Registers data to write to Cluster Local Scratch
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 Bit address to write to
<code>unsigned int</code>	<code>max_nn</code>	Maximum number of bytes to write (1-32)
<code>generic_ind_t</code>	<code>ind</code>	Indirection type
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

See Also:

- `cls_write_ind()`

3.6.3.12 `cls_write8_be_ind_ptr32`

Prototype:

```
void cls_write8_be_ind_ptr32(__xwrite void* data, volatile void __cls* address, unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write bytes to Cluster Local Scratch in Big Endian mode in indirect mode.

Table 3.37. `cls_write8_be_ind_ptr32` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Write Transfer Registers data to write to Cluster Local Scratch
<code>volatile void __cls*</code>	<code>address</code>	32 Bit address to write to
<code>unsigned int</code>	<code>max_nn</code>	Maximum number of bytes to write (1-32)
<code>generic_ind_t</code>	<code>ind</code>	Indirection type
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

See Also:

- `cls_write_ind()`

3.6.3.13 `cls_write8_le_ind_ptr40`

Prototype:

```
void cls_write8_le_ind_ptr40(__xwrite void* data, volatile void __addr40 __cls* address, unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write bytes to Cluster Local Scratch in Little Endian mode in indirect mode.

Table 3.38. `cls_write8_le_ind_ptr40` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Write Transfer Registers data to write to Cluster Local Scratch
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 Bit address to write to
<code>unsigned int</code>	<code>max_nn</code>	Maximum number of bytes to write (1-32)

Type	Name	Description
generic_ind_t	<i>ind</i>	Indirection type
sync_t	<i>sync</i>	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to be raised on completion

See Also:

- [cls_write_ind\(\)](#)

3.6.3.14 `cls_write8_le_ind_ptr32`

Prototype:

```
void cls_write8_le_ind_ptr32(__xwrite void* data, volatile void __cls* address, unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write bytes to Cluster Local Scratch in Little Endian mode in indirect mode.

Table 3.39. `cls_write8_le_ind_ptr32` parameters

Type	Name	Description
__xwrite void*	<i>data</i>	Write Transfer Registers data to write to Cluster Local Scratch
volatile void __cls*	<i>address</i>	32 Bit address to write to
unsigned int	<i>max_nn</i>	Maximum number of bytes to write (1-32)
generic_ind_t	<i>ind</i>	Indirection type
sync_t	<i>sync</i>	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to be raised on completion

See Also:

- [cls_write_ind\(\)](#)

3.6.3.15 `cls_write8_ind_ptr40`

Prototype:

```
void cls_write8_ind_ptr40(__xwrite void* data, volatile void __addr40 __cls* address, unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write bytes to Cluster Local Scratch in Big or Little Endian format in indirect mode.

Table 3.40. `cls_write8_ind_ptr40` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Write Transfer Registers holds the data to be written
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 Bit address to write to
<code>unsigned int</code>	<code>max_nn</code>	Maximum number of bytes to write (1-32)
<code>generic_ind_t</code>	<code>ind</code>	Indirection type
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

See Also:

- `cls_write_ind()`

3.6.3.16 `cls_write8_ind_ptr32`

Prototype:

```
void cls_write8_ind_ptr32(__xwrite void* data, volatile void __cls* address, unsigned
int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write bytes to Cluster Local Scratch in Big or Little Endian format in indirect mode.

Table 3.41. `cls_write8_ind_ptr32` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Write Transfer Registers holds the data to be written
<code>volatile void __cls*</code>	<code>address</code>	32 Bit address to write to
<code>unsigned int</code>	<code>max_nn</code>	Maximum number of bytes to write (1-32)
<code>generic_ind_t</code>	<code>ind</code>	Indirection type
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

See Also:

- `cls_write_ind()`

3.6.3.17 `cls_write_ptr32`

Prototype:

```
void cls_write_ptr32(__xwrite void* data, volatile void __cls* address, unsigned int
count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write 32-bit words to Cluster Local Scratch.

Write count number of 32-bit words from the write transfer registers pointed to by data to Cluster Local Scratch at address. The signal `sig_ptr` will be raised on completion and `sync` determines if the operation switch context or completes asynchronously.

The address has to be long word aligned.

Data will be treated as Big Endian or Little Endian depending on the compiler setting.

Table 3.42. `cls_write_ptr32` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Write Transfer Registers to be written
<code>volatile void __cls*</code>	<code>address</code>	Address to write to
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to be written
<code>sync_t</code>	<code>sync</code>	Type of synchronization (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

3.6.3.18 `cls_write_ptr40`

Prototype:

```
void cls_write_ptr40(__xwrite void* data, volatile void __addr40 __cls* address, unsigned
int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write 32-bit words to 40 bit Cluster Local Scratch.

Write count number of 32-bit words from the write transfer registers pointed to by data to Cluster Local Scratch at address. The signal `sig_ptr` will be raised on completion and `sync` determines if the operation switch context or completes asynchronously.

The address has to be long word aligned.

Data will be treated as Big Endian or Little Endian depending on the compiler setting.

Table 3.43. `cls_write_ptr40` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Write Transfer Registers to be written
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address to write to where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to be written

Type	Name	Description
sync_t	<i>sync</i>	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to be raised on completion

3.6.3.19 **cls_write8_be_ptr32**

Prototype:

```
void cls_write8_be_ptr32(__xwrite void* data, volatile void __cls* address, unsigned int
count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write bytes to Cluster Local Scratch in Big Endian mode.

Table 3.44. *cls_write8_be_ptr32* parameters

Type	Name	Description
__xwrite void*	<i>data</i>	Write Transfer Registers to be written
volatile void __cls*	<i>address</i>	Address to write to
unsigned int	<i>count</i>	Number of bytes to be written
sync_t	<i>sync</i>	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to be raised on completion

See Also:

- *cls_write8_ptr32()*

3.6.3.20 **cls_write8_be_ptr40**

Prototype:

```
void cls_write8_be_ptr40(__xwrite void* data, volatile void __addr40 __cls* address,
unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write bytes to 40 bit Cluster Local Scratch in Big Endian mode.

Table 3.45. *cls_write8_be_ptr40* parameters

Type	Name	Description
__xwrite void*	<i>data</i>	Write Transfer Registers to be written
volatile void __addr40 __cls*	<i>address</i>	40 bit address to write to where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
unsigned int	<i>count</i>	Number of bytes to be written

Type	Name	Description
sync_t	sync	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to be raised on completion

See Also:

- `cls_write8_ptr32()`

3.6.3.21 `cls_write8_le_ptr32`

Prototype:

```
void cls_write8_le_ptr32(__xwrite void* data, volatile void __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write bytes to Cluster Local Scratch in Little Endian mode.

Table 3.46. `cls_write8_le_ptr32` parameters

Type	Name	Description
__xwrite void*	data	Write Transfer Registers to be written
volatile void __cls*	address	Address to write to
unsigned int	count	Number of bytes to be written
sync_t	sync	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to be raised on completion

See Also:

- `cls_write8_ptr32()`

3.6.3.22 `cls_write8_le_ptr40`

Prototype:

```
void cls_write8_le_ptr40(__xwrite void* data, volatile void __addr40 __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write bytes to to 40 bit Cluster Local Scratch in Little Endian mode.

Table 3.47. `cls_write8_le_ptr40` parameters

Type	Name	Description
__xwrite void*	data	Write Transfer Registers to be written

Type	Name	Description
volatile void __addr40 __cls*	address	40 bit address to write to where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
unsigned int	count	Number of bytes to be written
sync_t	sync	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to be raised on completion

See Also:

- `cls_write8_ptr32()`

3.6.3.23 `cls_write8_ptr32`

Prototype:

```
void cls_write8_ptr32(__xwrite void* data, volatile void __cls* address, unsigned int
count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write bytes to Cluster Local Scratch.

Write count number of bytes from the write transfer registers pointed to by data to Cluster Local Scratch at address. The signal `sig_ptr` will be raised on completion and `sync` determines if the operation switch context or completes asynchronously.

The address can be an arbitrary byte address.

Table 3.48. `cls_write8_ptr32` parameters

Type	Name	Description
__xwrite void*	data	Write Transfer Registers to be written
volatile void __cls*	address	Address to write to
unsigned int	count	Number of bytes to be written
sync_t	sync	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to be raised on completion

3.6.3.24 `cls_write8_ptr40`

Prototype:

```
void cls_write8_ptr40(__xwrite void* data, volatile void __addr40 __cls* address, unsigned
int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write bytes to 40 bit Cluster Local Scratch.

Write count number of bytes from the write transfer registers pointed to by data to Cluster Local Scratch at address. The signal `sig_ptr` will be raised on completion and `sync` determines if the operation switch context or completes asynchronously.

The address can be an arbitrary byte address.

Table 3.49. `cls_write8_ptr40` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Write Transfer Registers to be written
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address to write to where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>count</code>	Number of bytes to be written
<code>sync_t</code>	<code>sync</code>	Type of synchronization (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

3.6.3.25 `cls_clear_bits_ind_ptr40`

Prototype:

```
void cls_clear_bits_ind_ptr40(__xwrite unsigned int* mask, volatile void __addr40 __cls* address, unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Clear bits in 40 bit Cluster Local Scratch with indirect word.

Clears the bits specified in the mask from the address onwards. Bits in multiple 32-bit words can be cleared by specifying `max_nn` greater than 1. In this case mask needs to point to an array of at least `max_nn` mask LWS.

If `max_nn` is greater than 1, then the address needs to be QW aligned otherwise it has to be LW aligned.



Note

Alignment is not currently enforced.

Table 3.50. `cls_clear_bits_ind_ptr40` parameters

Type	Name	Description
<code>__xwrite unsigned int*</code>	<code>mask</code>	Mask of bits to clear
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address in which to clear bits where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>max_nn</code>	Number of 32-bit words to clear bits from
<code>generic_ind_t</code>	<code>ind</code>	Indirection type
<code>sync_t</code>	<code>sync</code>	Type of synchronization (<code>sig_done</code> or <code>ctx_swap</code>)

Type	Name	Description
SIGNAL*	<i>sig_ptr</i>	Signal to be raised on completion

3.6.3.26 `cls_clear_bits_ind_ptr32`

Prototype:

```
void cls_clear_bits_ind_ptr32(__xwrite unsigned int* mask, volatile void __cls* address,
unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Clear bits in Cluster Local Scratch memory with indirect word.

Clears the bits specified in the mask from the address onwards. Bits in multiple 32-bit words can be cleared by specifying `max_nn` greater than 1. In this case mask needs to point to an array of at least `max_nn` mask LWs.

If `max_nn` is greater than 1, then the address needs to be QW aligned otherwise it has to be LW aligned.



Note

Alignment is not currently enforced.

Table 3.51. `cls_clear_bits_ind_ptr32` parameters

Type	Name	Description
<code>__xwrite unsigned int*</code>	<i>mask</i>	Mask of bits to clear
<code>volatile void __cls*</code>	<i>address</i>	Address of 32-bit word in which to clear bits
<code>unsigned int</code>	<i>max_nn</i>	Number of 32-bit words to clear bits from
<code>generic_ind_t</code>	<i>ind</i>	Indirection type
<code>sync_t</code>	<i>sync</i>	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to be raised on completion

3.6.3.27 `cls_set_bits_ind_ptr40`

Prototype:

```
void cls_set_bits_ind_ptr40(__xwrite unsigned int* mask, volatile void __addr40 __cls*
address, unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set bits in Cluster Local Scratch with indirect word.

Sets the bits specified in the mask from the address onwards. Bits in multiple 32-bit words can be set by specifying `max_nn` greater than 1. In this case mask needs to point to an array of at least `max_nn` mask LWs.

If max_nn is greater than 1, then the address needs to be QW aligned otherwise it has to be LW aligned.



Note

Alignment is not currently enforced.

Table 3.52. `cls_set_bits_ind_ptr40` parameters

Type	Name	Description
<code>__xwrite unsigned int*</code>	<code>mask</code>	Mask of bits to set
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address in which to set bits where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>max_nn</code>	Number of 32-bit words to set bits from
<code>generic_ind_t</code>	<code>ind</code>	Indirection type
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

3.6.3.28 `cls_set_bits_ind_ptr32`

Prototype:

```
void cls_set_bits_ind_ptr32(__xwrite unsigned int* mask, volatile void __cls* address,
                           unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set bits in 40 bit Cluster Local Scratch with indirect word.

Sets the bits specified in the mask from the address onwards. Bits in multiple 32-bit words can be set by specifying max_nn greater than 1. In this case mask needs to point to an array of at least max_nn mask LWS.

If max_nn is greater than 1, then the address needs to be QW aligned otherwise it has to be LW aligned.



Note

Alignment is not currently enforced.

Table 3.53. `cls_set_bits_ind_ptr32` parameters

Type	Name	Description
<code>__xwrite unsigned int*</code>	<code>mask</code>	Mask of bits to set
<code>volatile void __cls*</code>	<code>address</code>	Address of 32-bit word in which to set bits
<code>unsigned int</code>	<code>max_nn</code>	Number of 32-bit words to set bits from
<code>generic_ind_t</code>	<code>ind</code>	Indirection type

Type	Name	Description
sync_t	sync	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to be raised on completion

3.6.3.29 cls_clear_bits_ptr32

Prototype:

```
void cls_clear_bits_ptr32(__xwrite unsigned int* mask, volatile void __cls* address,
unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Clear bits in Cluster Local Scratch memory.

Clears the bits specified in the mask from the address onwards. Bits in multiple 32-bit words can be cleared by specifying count greater than 1. In this case mask needs to point to an array of at least count mask LWS.

If count is greater than 1, then the address needs to be QW aligned otherwise it has to be LW aligned.



Note

Alignment is not currently enforced.

Table 3.54. cls_clear_bits_ptr32 parameters

Type	Name	Description
__xwrite unsigned int*	mask	Mask of bits to clear
volatile void __cls*	address	Address of 32-bit word in which to clear bits
unsigned int	count	Number of 32-bit words to clear bits from
sync_t	sync	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to be raised on completion

3.6.3.30 cls_clear_bits_ptr40

Prototype:

```
void cls_clear_bits_ptr40(__xwrite unsigned int* mask, volatile void __addr40 __cls* address,
unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Clear bits in 40 bit Cluster Local Scratch.

Clears the bits specified in the mask from the address onwards. Bits in multiple 32-bit words can be cleared by specifying count greater than 1. In this case mask needs to point to an array of at least count mask LWS.

If count is greater than 1, then the address needs to be QW aligned otherwise it has to be LW aligned.



Note

Alignment is not currently enforced.

Table 3.55. `cls_clear_bits_ptr40` parameters

Type	Name	Description
<code>__xwrite unsigned int*</code>	<code>mask</code>	Mask of bits to clear
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address in which to clear bits where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to clear bits from
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

3.6.3.31 `cls_set_bits_ptr32`

Prototype:

```
void cls_set_bits_ptr32(__xwrite unsigned int* mask, volatile void __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set bits in Cluster Local Scratch memory.

Set the bits specified in the mask from the address onwards. Bits in multiple 32-bit words can be set by specifying count greater than 1. In this case mask needs to point to an array of at least count mask LWS.

If count is greater than 1, then the address needs to be QW aligned otherwise it has to be LW aligned.



Note

Alignment is not currently enforced.

Table 3.56. `cls_set_bits_ptr32` parameters

Type	Name	Description
<code>__xwrite unsigned int*</code>	<code>mask</code>	Mask of bits to set
<code>volatile void __cls*</code>	<code>address</code>	Address of 32-bit word in which to set bits
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to clear bits from
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

3.6.3.32 `cls_set_bits_ptr40`

Prototype:

```
void cls_set_bits_ptr40(__xwrite unsigned int* mask, volatile void __addr40 __cls* address,  
unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set bits in 40 bit Cluster Local Scratch.

Set the bits specified in the mask from the address onwards. Bits in multiple 32-bit words can be set by specifying count greater than 1. In this case mask needs to point to an array of at least count mask LWS.

If count is greater than 1, then the address needs to be QW aligned otherwise it has to be LW aligned.



Note

Alignment is not currently enforced.

Table 3.57. `cls_set_bits_ptr40` parameters

Type	Name	Description
<code>__xwrite unsigned int*</code>	<code>mask</code>	Mask of bits to set
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address in which to set bits where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to set bits from
<code>sync_t</code>	<code>sync</code>	Type of synchronization (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

3.6.3.33 `cls_set_bits_imm_ptr32`

Prototype:

```
void cls_set_bits_imm_ptr32(volatile void __cls* address, unsigned int value)
```

Description:

Set bits in Cluster Local Scratch memory immediate.

Set the bits specified in the mask in the word pointed to by address. The address needs to be LW aligned.

The immediate value is encoded in the instruction itself (in the count field). For values greater than 3 bits (the width of the count field) an indirect override is used. This allows for only the bottom 14 bits (bits 0-13) of the CLS word to be set in immediate mode.

The passed in value can actually be up to 16 bits wide. However, bit 14 is special. It works as a "extend" bit. If bit 14 is set in value and if bit 13 is set, then bits 13-31 will be set in the address 32-bit word in CLS. Thus, by passing a value of 0x7fff all bits in the addressed LW will be set.

Bit 15 of value is ignored.

Table 3.58. `cls_set_bits_imm_ptr32` parameters

Type	Name	Description
volatile void __cls*	address	Address of 32-bit word in which to set bits
unsigned int	value	Mask of bits to set

3.6.3.34 `cls_set_bits_imm_ptr40`

Prototype:

```
void cls_set_bits_imm_ptr40(volatile void __addr40 __cls* address, unsigned int value)
```

Description:

Set bits in 40 bit Cluster Local Scratch immediate.

Set the bits specified in the mask in the word pointed to by address. The address needs to be LW aligned.

The immediate value is encoded in the instruction itself (in the count field). For values greater than 3 bits (the width of the count field) an indirect override is used. This allows for only the bottom 14 bits (bits 0-13) of the CLS word to be set in immediate mode.

The passed in value can actually be up to 16 bits wide. However, bit 14 is special. It works as a "extend" bit. If bit 14 is set in value and if bit 13 is set, then bits 13-31 will be set in the address 32-bit word in CLS. Thus, by passing a value of 0x7fff all bits in the addressed LW will be set.

Bit 15 of value is ignored.

Table 3.59. `cls_set_bits_imm_ptr40` parameters

Type	Name	Description
volatile void __addr40 __cls*	address	40 bit address in which to set bits where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
unsigned int	value	Mask of bits to set

3.6.3.35 `cls_clear_bits_imm_ptr32`

Prototype:

```
void cls_clear_bits_imm_ptr32(volatile void __cls* address, unsigned int value)
```

Description:

Clear bits in Cluster Local Scratch memory immediate.

Clear the bits specified in the mask in the word pointed to by address. The address needs to be LW aligned.

The immediate value is encoded in the instruction itself (in the count field). For values greater than 3 bits (the width of the count field) and indirect override is used. This allows for only the bottom 14 bits of the CLS word to be cleared in immediate mode.

The passed in value can actually be up to 16 bits wide. However, bit 14 is special. It works as a "extend" bit. If bit 14 is set in value and if bit 13 is set, then bits 13-31 will be cleared set in the address 32-bit word in CLS. Thus, by passing a value of 0x7fff all bits in the addressed LW will be cleared.

Bit 15 of value is ignored.

Table 3.60. `cls_clear_bits_imm_ptr32` parameters

Type	Name	Description
volatile void __cls*	address	Address of 32-bit word in which to clear bits
unsigned int	value	Mask of bits to clear

3.6.3.36 `cls_clear_bits_imm_ptr40`

Prototype:

```
void cls_clear_bits_imm_ptr40(volatile void __addr40 __cls* address, unsigned int value)
```

Description:

Clear bits in 40 bit Cluster Local Scratch memory immediate.

Clear the bits specified in the mask in the word pointed to by address. The address needs to be LW aligned.

The immediate value is encoded in the instruction itself (in the count field). For values greater than 3 bits (the width of the count field) and indirect override is used. This allows for only the bottom 14 bits of the CLS word to be cleared in immediate mode.

The passed in value can actually be up to 16 bits wide. However, bit 14 is special. It works as a "extend" bit. If bit 14 is set in value and if bit 13 is set, then bits 13-31 will be cleared set in the address 32-bit word in CLS. Thus, by passing a value of 0x7fff all bits in the addressed LW will be cleared.

Bit 15 of value is ignored.

Table 3.61. `cls_clear_bits_imm_ptr40` parameters

Type	Name	Description
volatile void __addr40 __cls*	address	40 bit address in which to clear bits where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
unsigned int	value	Mask of bits to clear

3.6.3.37 cls_xor_bits_ind_ptr40

Prototype:

```
void cls_xor_bits_ind_ptr40(__xwrite void* mask, volatile void __addr40 __cls* address,  
unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

XOR bits in 40 bit Cluster Local Scratch with indirect.

XOR the bits specified in the mask with the 32-bit words at address onwards. Bits in multiple 32-bit words can be XORed by specifying max_nn greater than 1. In this case mask needs to point to an array of at least max_nn mask LWS.

If max_nn is greater than 1, then the address needs to be QW aligned otherwise it has to be LW aligned.



Note

Alignment is not currently enforced.

Table 3.62. cls_xor_bits_ind_ptr40 parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>mask</code>	Mask of bits to XOR
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address in which to XOR bits where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>max_nn</code>	Number of 32-bit words to XOR bits from
<code>generic_ind_t</code>	<code>ind</code>	Indirection word
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

3.6.3.38 cls_xor_bits_ind_ptr32

Prototype:

```
void cls_xor_bits_ind_ptr32(__xwrite void* mask, volatile void __cls* address, unsigned  
int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

XOR bits in 32 bit Cluster Local Scratch with indirect.



Note

Alignment is not currently enforced.

Table 3.63. `cls_xor_bits_ind_ptr32` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>mask</code>	Mask of bits to XOR
<code>volatile void __cls*</code>	<code>address</code>	32 Bit address to xor from
<code>unsigned int</code>	<code>max_nn</code>	Maximum number of 32-bit words to xor (1-32)
<code>generic_ind_t</code>	<code>ind</code>	Indirection type
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

3.6.3.39 `cls_xor_bits_ptr32`

Prototype:

```
void cls_xor_bits_ptr32(__xwrite void* mask, volatile void __cls* address, unsigned int
count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

XOR bits in Cluster Local Scratch memory.

XOR the bits specified in the mask with the 32-bit words at address onwards. Bits in multiple 32-bit words can be XORed by specifying count greater than 1. In this case mask needs to point to an array of at least count mask LWS.

If count is greater than 1, then the address needs to be QW aligned otherwise it has to be LW aligned.



Note

Alignment is not currently enforced.

Table 3.64. `cls_xor_bits_ptr32` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>mask</code>	Mask of bits to XOR
<code>volatile void __cls*</code>	<code>address</code>	Address of 32-bit word in which to XOR bits
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to XOR bits from
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

3.6.3.40 cls_xor_bits_ptr40

Prototype:

```
void cls_xor_bits_ptr40(__xwrite void* mask, volatile void __addr40 __cls* address,
unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

XOR bits in 40 bit Cluster Local Scratch.

XOR the bits specified in the mask with the 32-bit words at address onwards. Bits in multiple 32-bit words can be XORed by specifying count greater than 1. In this case mask needs to point to an array of at least count mask LWS.

If count is greater than 1, then the address needs to be QW aligned otherwise it has to be LW aligned.



Note

Alignment is not currently enforced.

Table 3.65. cls_xor_bits_ptr40 parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>mask</code>	Mask of bits to XOR
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address in which to XOR bits where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to XOR bits from
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

3.6.3.41 cls_swap_ptr32

Prototype:

```
void cls_swap_ptr32(__xwrite void* data, volatile void __cls* address, unsigned int count,
sync_t sync, SIGNAL* sig_ptr)
```

Description:

Swap bits in Cluster Local Scratch memory.

Swap the bits specified in the transfer register with the 32-bit words at address onwards.

Table 3.66. cls_swap_ptr32 parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Data to swap

Type	Name	Description
volatile void __cls*	address	Address of 32-bit word in which to swap bits
unsigned int	count	Number of 32-bit words to swap (maximum is 32)
sync_t	sync	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to be raised on completion

3.6.3.42 cls_swap_ptr40

Prototype:

```
void cls_swap_ptr40(__xwrite void* data, volatile void __addr40 __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Swap bits in 40 bit Cluster Local Scratch.

Swap the bits specified in the transfer register with the 32-bit words at address onwards.

Table 3.67. cls_swap_ptr40 parameters

Type	Name	Description
__xwrite void*	data	Data to swap
volatile void __addr40 __cls*	address	40 bit address in which to swap bits where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
unsigned int	count	Number of 32-bit words to swap (maximum is 32)
sync_t	sync	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to be raised on completion

3.6.3.43 cls_meter_ptr32

Prototype:

```
void cls_meter_ptr32(__xread unsigned int* val, __xwrite unsigned int* data, volatile void __cls* address, unsigned int rfc_type, unsigned int color, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Perform a metering operation in Cluster Local Scratch memory.

Perform a metering operation.

Table 3.68. `cls_meter_ptr32` parameters

Type	Name	Description
<code>__xread unsigned int*</code>	<code>val</code>	Return meter command color result i.e. 0 (green), 1 (yellow), 2 (red)
<code>__xwrite unsigned int*</code>	<code>data</code>	Data to write
<code>volatile void __cls*</code>	<code>address</code>	Address to write metering information
<code>unsigned int</code>	<code>rfc_type</code>	0 (RFC2698), 1 (RFC2697/4115). See enum <code>CLS_METER_RFC</code>
<code>unsigned int</code>	<code>color</code>	Color of packet: 0 (green), 1 (yellow), 2 (red). See enum <code>CLS_METER_COLOR</code>
<code>sync_t</code>	<code>sync</code>	Type of synchronization (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

3.6.3.44 `cls_meter_ptr40`

Prototype:

```
void cls_meter_ptr40(__xread unsigned int* val, __xwrite unsigned int* data, volatile
void __addr40 __cls* address, unsigned int rfc_type, unsigned int color, sync_t sync,
SIGNAL* sig_ptr)
```

Description:

Perform a metering operation in 40 bit Cluster Local Scratch memory.

Perform a metering operation.

Table 3.69. `cls_meter_ptr40` parameters

Type	Name	Description
<code>__xread unsigned int*</code>	<code>val</code>	Return meter command color result i.e. 0 (green), 1 (yellow), 2 (red)
<code>__xwrite unsigned int*</code>	<code>data</code>	Data to write
<code>volatile void __addr40 __cls*</code>	<code>address</code>	Address to write metering information
<code>unsigned int</code>	<code>rfc_type</code>	0 (RFC2698), 1 (RFC2697/4115). See enum <code>CLS_METER_RFC</code>
<code>unsigned int</code>	<code>color</code>	Color of packet: 0 (green), 1 (yellow), 2 (red). See enum <code>CLS_METER_COLOR</code>
<code>sync_t</code>	<code>sync</code>	Type of synchronization (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

3.6.3.45 `cls_statistic_ptr32`

Prototype:

```
void cls_statistic_ptr32(__xwrite void* data, volatile void __cls* address, unsigned int
count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Perform a statistic in Cluster Local Scratch memory.

Add statistic to CLS memory.

Table 3.70. cls_statistic_ptr32 parameters

Type	Name	Description
__xwrite void*	data	Add 32-bit data to address specified
volatile void __cls*	address	Address in CLS in which to add statistic
unsigned int	count	Number of words to add (maximum is 32)
sync_t	sync	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to be raised on completion

3.6.3.46 cls_statistic_ptr40

Prototype:

```
void cls_statistic_ptr40(__xwrite void* data, volatile void __addr40 __cls* address,
unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Perform a statistic in 40 bit Cluster Local Scratch.

Add statistic to CLS memory.

Table 3.71. cls_statistic_ptr40 parameters

Type	Name	Description
__xwrite void*	data	Add 32-bit data to address specified
volatile void __addr40 __cls*	address	40 bit CLS address in which to perform statistic where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
unsigned int	count	Number of words to add (maximum is 32)
sync_t	sync	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to be raised on completion

3.6.3.47 cls_statistic_imm_ptr32

Prototype:

```
void cls_statistic_imm_ptr32(volatile void __cls* address, unsigned int value)
```

Description:

Add 16-bit immediate statistic data to address in Cluster Local Scratch memory.

Add the value to the 32-bit word at the specified address. If the value is less than 3 bits it is directly encoded in the instruction, otherwise an indirect reference is used.

Table 3.72. `cls_statistic_imm_ptr32` parameters

Type	Name	Description
volatile void __cls*	address	Address of 32-bit word in which to write statistic
unsigned int	value	Value to write

3.6.3.48 `cls_statistic_imm_ptr40`

Prototype:

```
void cls_statistic_imm_ptr40(volatile void __addr40 __cls* address, unsigned int value)
```

Description:

Add 16-bit immediate statistic data to address in 40 bit Cluster Local Scratch memory.

Add the value to the 32-bit word at the specified address. If the value is less than 3 bits it is directly encoded in the instruction, otherwise an indirect reference is used.

Table 3.73. `cls_statistic_imm_ptr40` parameters

Type	Name	Description
volatile void __addr40 __cls*	address	40 bit address in which to write statistic where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
unsigned int	value	Value to write

3.6.3.49 `cls_add_ptr32`

Prototype:

```
void cls_add_ptr32(__xwrite void* data, volatile void __cls* address, unsigned int count,
sync_t sync, SIGNAL* sig_ptr)
```

Description:

Add value to Cluster Local Scratch memory address.

Add the value in data to the 32-bit word at the specified address. Multiple add operations (count > 1) can be performed in which case data needs to point to an array of the appropriate size.

If count is greater than 1, then the address needs to be QW aligned otherwise it has to be LW aligned.



Note

Alignment is not currently enforced.

Table 3.74. `cls_add_ptr32` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Value(s) to add
<code>volatile void __cls*</code>	<code>address</code>	Address of 32-bit word to which to add values
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to add values to
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

3.6.3.50 `cls_add_ptr40`

Prototype:

```
void cls_add_ptr40(__xwrite void* data, volatile void __addr40 __cls* address, unsigned
int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Add value to 40 bit Cluster Local Scratch.

Add the value in data to the 32-bit word at the specified address. Multiple add operations (count > 1) can be performed in which case data needs to point to an array of the appropriate size.

If count is greater than 1, then the address needs to be QW aligned otherwise it has to be LW aligned.



Note

Alignment is not currently enforced.

Table 3.75. `cls_add_ptr40` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Value(s) to add
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address in which to add values where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to add values to
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

3.6.3.51 `cls_test_and_add_ptr32`

Prototype:

```
void cls_test_and_add_ptr32(__xread unsigned int* val, __xwrite unsigned int* data,  
volatile void __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Test and add value to Cluster Local Scratch memory address.

Add the value in data to the 32-bit word at the specified address. Multiple add operations can be performed in which case data needs to point to an array of the appropriate size.

If count is greater than 1, then the address needs to be QW aligned otherwise it has to be LW aligned.



Note

Alignment is not currently enforced.

Table 3.76. `cls_test_and_add_ptr32` parameters

Type	Name	Description
<code>__xread unsigned int*</code>	<code>val</code>	Returned pre-modified value
<code>__xwrite unsigned int*</code>	<code>data</code>	Value(s) to add
<code>volatile void __cls*</code>	<code>address</code>	Address of 32-bit word to which to add values
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to add values to (1 to 32)
<code>sync_t</code>	<code>sync</code>	Type of synchronization (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

3.6.3.52 `cls_test_and_add_ptr40`

Prototype:

```
void cls_test_and_add_ptr40(__xread unsigned int* val, __xwrite unsigned int* data,  
volatile void __addr40 __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Test and add value to 40 bit Cluster Local Scratch memory address.

Add the value in data to the 32-bit word at the specified address. Multiple add operations can be performed in which case data needs to point to an array of the appropriate size.

If count is greater than 1, then the address needs to be QW aligned otherwise it has to be LW aligned.



Note

Alignment is not currently enforced.

Table 3.77. `cls_test_and_add_ptr40` parameters

Type	Name	Description
<code>__xread unsigned int*</code>	<code>val</code>	Returned pre-modified value
<code>__xwrite unsigned int*</code>	<code>data</code>	Value(s) to add
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address in which to add values where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to add values to (1 to 32)
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

3.6.3.53 `cls_add64_ptr32`

Prototype:

```
void cls_add64_ptr32(__xwrite void* data, volatile void __cls* address, unsigned int
count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Add 64-bit value to Cluster Local Scratch memory address.

Add the value in data to the 64-bit at the specified address using a 64-bit add. Multiple add operations (count > 2) can be performed in which case data needs to point to an array of the appropriate size. Valid count values are 2, 4, 6, 8, 10, 12, 14 and 16.

Address must be 64-bit aligned.



Note

Alignment is not currently enforced.

Table 3.78. `cls_add64_ptr32` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Value(s) to add
<code>volatile void __cls*</code>	<code>address</code>	Address of 64-bit word to which to add values
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to add values to and must be multiple of 2.
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

3.6.3.54 `cls_add64_ptr40`

Prototype:

```
void cls_add64_ptr40(__xwrite void* data, volatile void __addr40 __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Add 64-bit value to 40 bit Cluster Local Scratch memory address.

Add the value in data to the 64-bit at the specified address using a 64-bit add. Multiple add operations (count > 2) can be performed in which case data needs to point to an array of the appropriate size. Valid count values are 2, 4, 6, 8, 10, 12, 14 and 16.

Address must be 64-bit aligned.



Note

Alignment is not currently enforced.

Table 3.79. `cls_add64_ptr40` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Value(s) to add
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address in which to add values where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to add values to and must be multiple of 2.
<code>sync_t</code>	<code>sync</code>	Type of synchronization (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

3.6.3.55 `cls_test_and_add64_ptr32`

Prototype:

```
void cls_test_and_add64_ptr32(__xread unsigned int* val, __xwrite unsigned int* data, volatile void __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Test and add 64-bit value to Cluster Local Scratch memory address.

Add the value in data to the 64-bit at the specified address. Multiple add operations can be performed in which case data needs to point to an array of the appropriate size.

If count is greater than 1, then the address needs to be QW aligned otherwise it has to be LW aligned.

Same as `cls_add64` command, but also returns pre-modified memory contents.



Note

Alignment is not currently enforced.

Table 3.80. `cls_test_and_add64_ptr32` parameters

Type	Name	Description
<code>__xread unsigned int*</code>	<code>val</code>	Returned pre-modified value
<code>__xwrite unsigned int*</code>	<code>data</code>	Value(s) to add
<code>volatile void __cls*</code>	<code>address</code>	Address of 64-bit word to which to add values
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words and must be multiple of 2 up to 16
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

See Also:

- `cls_add64()`

3.6.3.56 `cls_test_and_add64_ptr40`

Prototype:

```
void cls_test_and_add64_ptr40(__xread unsigned int* val, __xwrite unsigned int* data,
volatile void __addr40 __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Test and add 64-bit value to 40 bit Cluster Local Scratch memory address.

Add the value in data to the 64-bit word at the specified address. Multiple add operations can be performed in which case data needs to point to an array of the appropriate size.

If count is greater than 1, then the address needs to be QW aligned otherwise it has to be LW aligned.

Same as `cls_add64` command, but also returns pre-modified memory contents.



Note

Alignment is not currently enforced.

Table 3.81. `cls_test_and_add64_ptr40` parameters

Type	Name	Description
<code>__xread unsigned int*</code>	<code>val</code>	Returned pre-modified value

Type	Name	Description
<code>__xwrite unsigned int*</code>	<code>data</code>	Value(s) to add
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address in which to add values where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words and must be multiple of 2 up to 16
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

See Also:

- `cls_add64()`

3.6.3.57 `cls_add_sat_ptr32`

Prototype:

```
void cls_add_sat_ptr32(__xwrite void* data, volatile void __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Add value to Cluster Local Scratch memory address with saturation.

Add the value in data to the 32-bit word at the specified address. Multiple add operations (`count > 1`) can be performed in which case data needs to point to an array of the appropriate size.

If the addition would overflow, the 32-bit word is set to 0xffffffff.

If `count` is greater than 1, then the address needs to be QW aligned otherwise it has to be LW aligned.



Note

Alignment is not currently enforced.

Table 3.82. `cls_add_sat_ptr32` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Value(s) to add
<code>volatile void __cls*</code>	<code>address</code>	Address of 32-bit word to which to add values
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to add values to
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

3.6.3.58 cls_add_ind_ptr40

Prototype:

```
void cls_add_ind_ptr40(__xwrite void* data, volatile void __addr40 __cls* address, unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Add value to 40 bit Cluster Local Scratch.

Add the value in data to the 32-bit word at the specified address. Multiple add operations (`max_nn > 1`) can be performed in which case data needs to point to an array of the appropriate size.

If `max_nn` is greater than 1, then the address needs to be QW aligned otherwise it has to be LW aligned.



Note

Alignment is not currently enforced.

Table 3.83. `cls_add_ind_ptr40` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Value(s) to add
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address in which to add values where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>max_nn</code>	Number of 32-bit words to add values to
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

3.6.3.59 cls_add_ind_ptr32

Prototype:

```
void cls_add_ind_ptr32(__xwrite void* data, volatile void __cls* address, unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Add value to 32 bit Cluster Local Scratch.

Add the value in data to the 32-bit word at the specified address. Multiple add operations (`max_nn > 1`) can be performed in which case data needs to point to an array of the appropriate size.

If `max_nn` is greater than 1, then the address needs to be QW aligned otherwise it has to be LW aligned.



Note

Alignment is not currently enforced.

Table 3.84. `cls_add_ind_ptr32` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Value(s) to add
<code>volatile void __cls*</code>	<code>address</code>	Address of 32-bit word to which to add values
<code>unsigned int</code>	<code>max_nn</code>	Number of 32-bit words to add values to
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

3.6.3.60 `cls_add64_ind_ptr40`

Prototype:

```
void cls_add64_ind_ptr40(__xwrite void* data, volatile void __addr40 __cls* address,
unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Add the 64-bit value(s) in the transfer register pairs to the 64-bit value(s) at the specified address.

Add the value in data to the 64-bit word at the specified address. Multiple add operations (`max_nn > 1`) can be performed in which case data needs to point to an array of the appropriate size.

The address needs to be QW aligned



Note

Alignment is not currently enforced.

Table 3.85. `cls_add64_ind_ptr40` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Value(s) to add
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address in which to add values where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>max_nn</code>	Number of 32-bit words to add values to (2,4,6,8,10,12,14 or 16)
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

3.6.3.61 `cls_add64_ind_ptr32`

Prototype:

```
void cls_add64_ind_ptr32(__xwrite void* data, volatile void __cls* address, unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Add the 64-bit value(s) in the transfer register pairs to the 64-bit value(s) at the specified address.

Add the value in data to the 64-bit word at the specified address. Multiple add operations ($\text{max_nn} > 1$) can be performed in which case data needs to point to an array of the appropriate size.

The address needs to be QW aligned



Note

Alignment is not currently enforced.

Table 3.86. `cls_add64_ind_ptr32` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Value(s) to add
<code>volatile void __cls*</code>	<code>address</code>	Address of 64-bit word to which to add values
<code>unsigned int</code>	<code>max_nn</code>	Number of 32-bit words to add values to (2,4,6,8,10,12,14 or 16)
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

3.6.3.62 `cls_add_sat_ind_ptr40`

Prototype:

```
void cls_add_sat_ind_ptr40(__xwrite void* data, volatile void __addr40 __cls* address, unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Add value to 40 bit Cluster Local Scratch, with saturation limits.

Add the value in data to the 32-bit word at the specified address. Multiple add operations ($\text{max_nn} > 1$) can be performed in which case data needs to point to an array of the appropriate size.

If `max_nn` is greater than 1, then the address needs to be QW aligned otherwise it has to be LW aligned.



Note

Alignment is not currently enforced.

Table 3.87. `cls_add_sat_ind_ptr40` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Value(s) to add
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address in which to add values where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>max_nn</code>	Number of 32-bit words to add values to
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

3.6.3.63 `cls_add_sat_ind_ptr32`

Prototype:

```
void cls_add_sat_ind_ptr32(__xwrite void* data, volatile void __cls* address, unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Add value to 32 bit Cluster Local Scratch, with saturation limits.

Add the value in data to the 32-bit word at the specified address. Multiple add operations (`max_nn > 1`) can be performed in which case data needs to point to an array of the appropriate size.

If `max_nn` is greater than 1, then the address needs to be QW aligned otherwise it has to be LW aligned.



Note

Alignment is not currently enforced.

Table 3.88. `cls_add_sat_ind_ptr32` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Value(s) to add
<code>volatile void __cls*</code>	<code>address</code>	Address of 32-bit word to which to add values
<code>unsigned int</code>	<code>max_nn</code>	Number of 32-bit words to add values to
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

3.6.3.64 cls_sub_ind_ptr40

Prototype:

```
void cls_sub_ind_ptr40(__xwrite void* data, volatile void __addr40 __cls* address, unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Subtract the 32-bit value(s) in the transfer register(s) from the 32-bit value(s) at the specified address.

Subtract the value in data from the 32-bit word at the specified address. Multiple add operations (`max_nn > 1`) can be performed in which case data needs to point to an array of the appropriate size.

If `max_nn` is greater than 1, then the address needs to be QW aligned otherwise it has to be LW aligned.



Note

Alignment is not currently enforced.

Table 3.89. `cls_sub_ind_ptr40` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Value(s) to add
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address in which to subtract values where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>max_nn</code>	Number of 32-bit words to subtract values from
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

3.6.3.65 cls_sub_ind_ptr32

Prototype:

```
void cls_sub_ind_ptr32(__xwrite void* data, volatile void __cls* address, unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Subtract value to 32 bit Cluster Local Scratch.

Subtract the value in data from the 32-bit word at the specified address. Multiple add operations (`max_nn > 1`) can be performed in which case data needs to point to an array of the appropriate size.

If `max_nn` is greater than 1, then the address needs to be QW aligned otherwise it has to be LW aligned.



Note

Alignment is not currently enforced.

Table 3.90. `cls_sub_ind_ptr32` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Value(s) to subtract
<code>volatile void __cls*</code>	<code>address</code>	Address of 32-bit word to which to subtract values
<code>unsigned int</code>	<code>max_nn</code>	Number of 32-bit words to subtract values from
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

3.6.3.66 `cls_sub64_ind_ptr40`

Prototype:

```
void cls_sub64_ind_ptr40(__xwrite void* data, volatile void __addr40 __cls* address,
unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Subtract the 64-bit value(s) in the transfer register pairs to the 64-bit value(s) at the specified address.

Subtract the value in data to the 64-bit word at the specified address. Multiple add operations (`max_nn > 1`) can be performed in which case data needs to point to an array of the appropriate size.

The address needs to be QW aligned



Note

Alignment is not currently enforced.

Table 3.91. `cls_sub64_ind_ptr40` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Value(s) to subtract
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address in which to add values where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>max_nn</code>	Number of 32-bit words to subtract values to (2,4,6,8,10,12,14 or 16)
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

3.6.3.67 `cls_sub64_ind_ptr32`

Prototype:

```
void cls_sub64_ind_ptr32(__xwrite void* data, volatile void __cls* address, unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Subtract the 64-bit value(s) in the transfer register pairs to the 64-bit value(s) at the specified address.

Subtract the value in data to the 64-bit word at the specified address. Multiple add operations (`max_nn > 1`) can be performed in which case data needs to point to an array of the appropriate size.

The address needs to be QW aligned



Note

Alignment is not currently enforced.

Table 3.92. `cls_sub64_ind_ptr32` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Value(s) to subtract
<code>volatile void __cls*</code>	<code>address</code>	Address of 64-bit word to which to subtract values
<code>unsigned int</code>	<code>max_nn</code>	Number of 32-bit word to add values to (2,4,6,8,10,12,14 or 16)
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

3.6.3.68 `cls_sub_sat_ind_ptr40`

Prototype:

```
void cls_sub_sat_ind_ptr40(__xwrite void* data, volatile void __addr40 __cls* address, unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Subtract value to 40 bit Cluster Local Scratch, with saturation limits.

Subtract the value in data to the 32-bit word at the specified address. Multiple subtract operations (`max_nn > 1`) can be performed in which case data needs to point to an array of the appropriate size.

If `max_nn` is greater than 1, then the address needs to be QW aligned otherwise it has to be LW aligned.



Note

Alignment is not currently enforced.

Table 3.93. `cls_sub_sat_ind_ptr40` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Value(s) to sub
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address in which to add values where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>max_nn</code>	Number of 32-bit words to subtract values to
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

3.6.3.69 `cls_sub_sat_ind_ptr32`

Prototype:

```
void cls_sub_sat_ind_ptr32(__xwrite void* data, volatile void __cls* address, unsigned
int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Subtract value to 32 bit Cluster Local Scratch, with saturation limits.

Subtract the value in data to the 32-bit word at the specified address. Multiple add operations (`max_nn > 1`) can be performed in which case data needs to point to an array of the appropriate size.

If `max_nn` is greater than 1, then the address needs to be QW aligned otherwise it has to be LW aligned.



Note

Alignment is not currently enforced.

Table 3.94. `cls_sub_sat_ind_ptr32` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Value(s) to subtract
<code>volatile void __cls*</code>	<code>address</code>	Address of 32-bit word to which to subtract values
<code>unsigned int</code>	<code>max_nn</code>	Number of 32-bit words to subtract values to
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

3.6.3.70 cls_add_sat_ptr40

Prototype:

```
void cls_add_sat_ptr40(__xwrite void* data, volatile void __addr40 __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Add value to 40 bit Cluster Local Scratch memory address with saturation.

Add the value in data to the 32-bit word at the specified address. Multiple add operations (count > 1) can be performed in which case data needs to point to an array of the appropriate size.

If the addition would overflow, the 32-bit word is set to 0xffffffff.

If count is greater than 1, then the address needs to be QW aligned otherwise it has to be LW aligned.



Note

Alignment is not currently enforced.

Table 3.95. cls_add_sat_ptr40 parameters

Type	Name	Description
__xwrite void*	data	Value(s) to add
volatile void __addr40 __cls*	address	40 bit address in which to add values where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
unsigned int	count	Number of 32-bit words to add values to
sync_t	sync	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to be raised on completion

3.6.3.71 cls_add_imm_ptr32

Prototype:

```
void cls_add_imm_ptr32(volatile void __cls* address, unsigned int value)
```

Description:

Add an immediate value to Cluster Local Scratch memory address.

Add the value in data to the 32-bit word at the specified address. If the value is less than 3 bits it is directly encoded in the instruction, otherwise a indirect reference is used. The value can be up to 14 bits long.

This function accepts values for up to 16 bits with bit 14 being used as an optional sign extension. If bit 14 is 0 no sign extension is used. If bit 14 is set, the sign of value is extended to 32-bit before the operation is performed

Bit 15 of value is ignored.

Table 3.96. `cls_add_imm_ptr32` parameters

Type	Name	Description
volatile void __cls*	address	Address of 32-bit word to which to add values
unsigned int	value	Value to add

3.6.3.72 `cls_add_imm_ptr40`

Prototype:

```
void cls_add_imm_ptr40(volatile void __addr40 __cls* address, unsigned int value)
```

Description:

Add an immediate value to 40 bit Cluster Local Scratch memory address.

Add the value in data to the 32-bit word at the specified address. If the value is less than 3 bits it is directly encoded in the instruction, otherwise a indirect reference is used. The value can be up to 14 bits long.

This function accepts values for up to 16 bits with bit 14 being used as an optional sign extension. If bit 14 is 0 no sign extension is used. If bit 14 is set, the sign of value is extended to 32-bit before the operation is performed

Bit 15 of value is ignored.

Table 3.97. `cls_add_imm_ptr40` parameters

Type	Name	Description
volatile void __addr40 __cls*	address	40 bit address in which to add values where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
unsigned int	value	Value to add

3.6.3.73 `cls_add_imm_sat_ptr32`

Prototype:

```
void cls_add_imm_sat_ptr32(volatile void __cls* address, unsigned int value)
```

Description:

Add an immediate value to Cluster Local Scratch memory address with saturation.

Add the value in data to the 32-bit word at the specified address. If the value is less than 3 bits it is directly encoded in the instruction, otherwise a indirect reference is used. If the addition overflows the 32-bit word is set to 0xffffffff. Values can be up to 14 bits long.

This function accepts values for up to 16 bits with bit 14 being used as an optional sign extension. If bit 14 is 0 no sign extension is used. If bit 14 is set, the sign of value is extended to 32-bit before the operation is performed

Bit 15 of value is ignored.

Table 3.98. `cls_add_imm_sat_ptr32` parameters

Type	Name	Description
volatile void __cls*	address	Address of 32-bit word to which to add values
unsigned int	value	Value to add

3.6.3.74 `cls_add_imm_sat_ptr40`

Prototype:

```
void cls_add_imm_sat_ptr40(volatile void __addr40 __cls* address, unsigned int value)
```

Description:

Add an immediate value to 40 bit Cluster Local Scratch memory address with saturation.

Add the value in data to the 32-bit word at the specified address. If the value is less than 3 bits it is directly encoded in the instruction, otherwise a indirect reference is used. If the addition overflows the 32-bit word is set to 0xffffffff. Values can be up to 14 bits long.

This function accepts values for up to 16 bits with bit 14 being used as an optional sign extension. If bit 14 is 0 no sign extension is used. If bit 14 is set, the sign of value is extended to 32-bit before the operation is performed

Bit 15 of value is ignored.

Table 3.99. `cls_add_imm_sat_ptr40` parameters

Type	Name	Description
volatile void __addr40 __cls*	address	40 bit address in which to add values where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
unsigned int	value	Value to add

3.6.3.75 `cls_add64_imm_ptr32`

Prototype:

```
void cls_add64_imm_ptr32(volatile void __cls* address, unsigned int value)
```

Description:

Add an immediate value to Cluster Local Scratch 64-bit memory location.

Add the value in data to the 64-bit word at the specified address. If the value is less than 3 bits it is directly encoded in the instruction, otherwise a indirect reference is used. The value can be up to 14 bits long.

This function accepts values for up to 16 bits with the top 2 bits being used as an optional sign extension. If the top two bits are 00 no sign extension is used. The top bits set to 01 indicate sign extend to 64-bit value, top bits set

to 10 indicate no sign extension but duplicate data in top and bottom fields, top bits equal 11 indicate two sign extended 32-bit values.

Table 3.100. `cls_add64_imm_ptr32` parameters

Type	Name	Description
volatile void __cls*	address	Address of 64-bit word to which to add values
unsigned int	value	Value to add

3.6.3.76 `cls_add64_imm_ptr40`

Prototype:

```
void cls_add64_imm_ptr40(volatile void __addr40 __cls* address, unsigned int value)
```

Description:

Add an immediate value to 40 bit Cluster Local Scratch 64-bit memory location.

Add the value in data to the 64-bit word at the specified address. If the value is less than 3 bits it is directly encoded in the instruction, otherwise a indirect reference is used. The value can be up to 14 bits long.

This function accepts values for up to 16 bits with the top 2 bits being used as an optional sign extension. If the top two bits are 00 no sign extension is used. The top bits set to 01 indicate sign extend to 64-bit value, top bits set to 10 indicate no sign extension but duplicate data in top and bottom fields, top bits equal 11 indicate two sign extended 32-bit values.

Table 3.101. `cls_add64_imm_ptr40` parameters

Type	Name	Description
volatile void __addr40 __cls*	address	40 bit address in which to add values where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
unsigned int	value	Value to add

3.6.3.77 `cls_test_and_add_imm_ptr32`

Prototype:

```
void cls_test_and_add_imm_ptr32(__xread unsigned int* val, volatile void __cls* address,  
unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch memory test and add immediate.

Same as `cls_add_imm` command, but also returns pre-modified memory contents.

Table 3.102. `cls_test_and_add_imm_ptr32` parameters

Type	Name	Description
<code>__xread unsigned int*</code>	<code>val</code>	Returned pre-modified value
<code>volatile void __cls*</code>	<code>address</code>	Cluster Local Scratch address to add to
<code>unsigned int</code>	<code>value</code>	Immediate value to add (1 - 31)
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

See Also:

- `cls_add_imm_ptr32()`

3.6.3.78 `cls_test_and_add_imm_ptr40`

Prototype:

```
void cls_test_and_add_imm_ptr40(__xread unsigned int* val, volatile void __addr40 __cls* address, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch memory test and add immediate.

Same as `cls_add_imm` command, but also returns pre-modified memory contents.

Table 3.103. `cls_test_and_add_imm_ptr40` parameters

Type	Name	Description
<code>__xread unsigned int*</code>	<code>val</code>	Returned pre-modified value
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address in which to add values where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>value</code>	Immediate value to add (1 - 31)
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

See Also:

- `cls_add_imm_ptr40()`

3.6.3.79 `cls_test_and_add64_imm_ptr32`

Prototype:

```
void cls_test_and_add64_imm_ptr32(__xread unsigned int* val, volatile void __cls* address, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch memory test and add immediate to 64bit location.

Same as `cls_add_imm64` command, but also returns pre-modified memory contents.

Table 3.104. `cls_test_and_add64_imm_ptr32` parameters

Type	Name	Description
<code>__xread unsigned int*</code>	<code>val</code>	Returned pre-modified value
<code>volatile void __cls*</code>	<code>address</code>	Cluster Local Scratch address to add to
<code>unsigned int</code>	<code>value</code>	Immediate value to add (1 - 31)
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

See Also:

- `cls_add64_imm_ptr32()`

3.6.3.80 `cls_test_and_add64_imm_ptr40`

Prototype:

```
void cls_test_and_add64_imm_ptr40(__xread unsigned int* val, volatile void __addr40 __cls*
address, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch memory test and add immediate to 64bit location.

Same as `cls_add_imm64` command, but also returns pre-modified memory contents.

Table 3.105. `cls_test_and_add64_imm_ptr40` parameters

Type	Name	Description
<code>__xread unsigned int*</code>	<code>val</code>	Returned pre-modified value
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address in which to add values where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>value</code>	Immediate value to add (1 - 31)
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

See Also:

- `cls_add64_imm_ptr40()`

3.6.3.81 cls_sub_ptr32

Prototype:

```
void cls_sub_ptr32(__xwrite void* data, volatile void __cls* address, unsigned int count,  
sync_t sync, SIGNAL* sig_ptr)
```

Description:

Subtract value from Cluster Local Scratch memory address.

Subtract the value in data from the 32-bit word at the specified address. Multiple subtract operations (count > 1) can be performed in which case data needs to point to an array of the appropriate size.

If count is greater than 1, then the address needs to be QW aligned otherwise it has to be LW aligned.



Note

Alignment is not currently enforced.

Table 3.106. cls_sub_ptr32 parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Value(s) to subtract
<code>volatile void __cls*</code>	<code>address</code>	Address of 32-bit word from which to subtract values
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to subtract values from
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

3.6.3.82 cls_sub_ptr40

Prototype:

```
void cls_sub_ptr40(__xwrite void* data, volatile void __addr40 __cls* address, unsigned  
int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Subtract value from 40 bit Cluster Local Scratch memory address.

Subtract the value in data from the 32-bit word at the specified address. Multiple subtract operations (count > 1) can be performed in which case data needs to point to an array of the appropriate size.

If count is greater than 1, then the address needs to be QW aligned otherwise it has to be LW aligned.



Note

Alignment is not currently enforced.

Table 3.107. `cls_sub_ptr40` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Value(s) to subtract
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address in which to subtract values where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to subtract values from
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

3.6.3.83 `cls_read_le_ind_ptr40`

Prototype:

```
void cls_read_le_ind_ptr40(__xread void* data, volatile void __addr40 __cls* address,
                           unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read 32-bit words from Cluster Local Scratch in Little Endian mode in indirect mode.

Table 3.108. `cls_read_le_ind_ptr40` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Read Transfer Registers the read data will be placed in
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 Bit address to read from
<code>unsigned int</code>	<code>max_nn</code>	Maximum number of 32-bit words to read (1-32)
<code>generic_ind_t</code>	<code>ind</code>	Indirection type
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

See Also:

- `cls_read_ind()`

3.6.3.84 `cls_read_le_ind_ptr32`

Prototype:

```
void cls_read_le_ind_ptr32(__xread void* data, volatile void __cls* address, unsigned
int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read 32-bit words from Cluster Local Scratch in Little Endian mode in indirect mode.

Table 3.109. `cls_read_le_ind_ptr32` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Read Transfer Registers the read data will be placed in
<code>volatile void __cls*</code>	<code>address</code>	32 Bit address to read from
<code>unsigned int</code>	<code>max_nn</code>	Maximum number of 32-bit words to read (1-32)
<code>generic_ind_t</code>	<code>ind</code>	Indirection type
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

See Also:

- `cls_read_ind()`

3.6.3.85 `cls_read_be_ind_ptr40`

Prototype:

```
void cls_read_be_ind_ptr40(__xread void* data, volatile void __addr40 __cls* address,
unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read 32-bit words from Cluster Local Scratch in Big Endian mode in indirect mode.

Table 3.110. `cls_read_be_ind_ptr40` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Read Transfer Registers the read data will be placed in
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 Bit address to read from
<code>unsigned int</code>	<code>max_nn</code>	Maximum number of 32-bit words to read (1-32)
<code>generic_ind_t</code>	<code>ind</code>	Indirection type
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

See Also:

- `cls_read_ind()`

3.6.3.86 cls_read_be_ind_ptr32

Prototype:

```
void cls_read_be_ind_ptr32(__xread void* data, volatile void __cls* address, unsigned
int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read 32-bit words from Cluster Local Scratch in Big Endian mode in indirect mode.

Table 3.111. cls_read_be_ind_ptr32 parameters

Type	Name	Description
__xread void*	data	Read Transfer Registers the read data will be placed in
volatile void __cls*	address	32 Bit address to read from
unsigned int	max_nn	Maximum number of 32-bit words to read (1-32)
generic_ind_t	ind	Indirection type
sync_t	sync	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to be raised on completion

See Also:

- [cls_read_ind\(\)](#)

3.6.3.87 cls_read_ind_ptr40

Prototype:

```
void cls_read_ind_ptr40(__xread void* data, volatile void __addr40 __cls* address, unsigned
int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read 32-bit words from Cluster Local Scratch in Big Endian mode in indirect mode.

Table 3.112. cls_read_ind_ptr40 parameters

Type	Name	Description
__xread void*	data	Read Transfer Registers the read data will be placed in
volatile void __addr40 __cls*	address	40 Bit address to read from
unsigned int	max_nn	Maximum number of 32-bit words to read (1-32)
generic_ind_t	ind	Indirection type
sync_t	sync	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to be raised on completion

See Also:

- `cls_read_ind()`

3.6.3.88 `cls_read_ind_ptr32`

Prototype:

```
void cls_read_ind_ptr32(__xread void* data, volatile void __cls* address, unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read 32-bit words from Cluster Local Scratch, endian mode according compiler option in indirect mode.

Table 3.113. `cls_read_ind_ptr32` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Read Transfer Registers the read data will be placed in
<code>volatile void __cls*</code>	<code>address</code>	32 Bit address to read from
<code>unsigned int</code>	<code>max_nn</code>	Maximum number of 32-bit words to read (1-32)
<code>generic_ind_t</code>	<code>ind</code>	Indirection type
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

See Also:

- `cls_read_ind()`

3.6.3.89 `cls_write_le_ind_ptr40`

Prototype:

```
void cls_write_le_ind_ptr40(__xwrite void* data, volatile void __addr40 __cls* address, unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write 32-bit words to Cluster Local Scratch in Little Endian mode in indirect mode.

Table 3.114. `cls_write_le_ind_ptr40` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Write Transfer Registers data to write to Cluster Local Scratch
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 Bit address to write to
<code>unsigned int</code>	<code>max_nn</code>	Maximum number of 32-bit words to write (1-32)

Type	Name	Description
generic_ind_t	<i>ind</i>	Indirection type
sync_t	<i>sync</i>	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to be raised on completion

See Also:

- [cls_write_ind\(\)](#)

3.6.3.90 `cls_write_le_ind_ptr32`

Prototype:

```
void cls_write_le_ind_ptr32(__xwrite void* data, volatile void __cls* address, unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write 32-bit words to Cluster Local Scratch in Little Endian mode in indirect mode.

Table 3.115. `cls_write_le_ind_ptr32` parameters

Type	Name	Description
__xwrite void*	<i>data</i>	Write Transfer Registers data to write to Cluster Local Scratch
volatile void __cls*	<i>address</i>	32 Bit address to write to
unsigned int	<i>max_nn</i>	Maximum number of 32-bit words to write (max_1 - max_32)
generic_ind_t	<i>ind</i>	Indirection type
sync_t	<i>sync</i>	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to be raised on completion

See Also:

- [cls_write_ind\(\)](#)

3.6.3.91 `cls_write_be_ind_ptr40`

Prototype:

```
void cls_write_be_ind_ptr40(__xwrite void* data, volatile void __addr40 __cls* address, unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write 32-bit words to Cluster Local Scratch in Big Endian mode in indirect mode.

Table 3.116. `cls_write_be_ind_ptr40` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Write Transfer Registers data to write to Cluster Local Scratch
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 Bit address to write to
<code>unsigned int</code>	<code>max_nn</code>	Maximum number of 32-bit words to write (1-32)
<code>generic_ind_t</code>	<code>ind</code>	Indirection type
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

See Also:

- `cls_write_ind()`

3.6.3.92 `cls_write_be_ind_ptr32`

Prototype:

```
void cls_write_be_ind_ptr32(__xwrite void* data, volatile void __cls* address, unsigned
int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write 32-bit words to Cluster Local Scratch in Big Endian mode in indirect mode.

Table 3.117. `cls_write_be_ind_ptr32` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Write Transfer Registers data to write to Cluster Local Scratch
<code>volatile void __cls*</code>	<code>address</code>	32 Bit address to write to
<code>unsigned int</code>	<code>max_nn</code>	Maximum number of 32-bit words to write (max_1 - max_32)
<code>generic_ind_t</code>	<code>ind</code>	Indirection type
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

See Also:

- `cls_write_ind()`

3.6.3.93 `cls_write_ind_ptr40`

Prototype:

```
void cls_write_ind_ptr40(__xwrite void* data, volatile void __addr40 __cls* address,
unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write 32-bit words to Cluster Local Scratch in Big or Little Endian format in indirect mode.

Table 3.118. `cls_write_ind_ptr40` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Write Transfer Registers holds the data to be written
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 Bit address to write to
<code>unsigned int</code>	<code>max_nn</code>	Maximum number of 32-bit words to write (1-32)
<code>generic_ind_t</code>	<code>ind</code>	Indirection type
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

See Also:

- `cls_write_ind()`

3.6.3.94 `cls_write_ind_ptr32`

Prototype:

```
void cls_write_ind_ptr32(__xwrite void* data, volatile void __cls* address, unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write 32-bit words to Cluster Local Scratch in Big or Little Endian format in indirect mode.

Table 3.119. `cls_write_ind_ptr32` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Write Transfer Registers holds the data to be written
<code>volatile void __cls*</code>	<code>address</code>	32 Bit address to write to
<code>unsigned int</code>	<code>max_nn</code>	Maximum number of 32-bit words to write (1-32)
<code>generic_ind_t</code>	<code>ind</code>	Indirection type
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

See Also:

- `cls_write_ind()`

3.6.3.95 `cls_test_and_sub_ptr32`

Prototype:

```
void cls_test_and_sub_ptr32(__xread unsigned int* val, __xwrite unsigned int* data,  
volatile void __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Test and subtract value to Cluster Local Scratch memory address.

Subtract the value in data to the 32-bit word at the specified address. Multiple subtract operations can be performed in which case data needs to point to an array of the appropriate size.

If count is greater than 1, then the address needs to be QW aligned otherwise it has to be LW aligned.

Same as `cls_sub` command, but also returns pre-modified memory contents.



Note

Alignment is not currently enforced.

Table 3.120. `cls_test_and_sub_ptr32` parameters

Type	Name	Description
<code>__xread unsigned int*</code>	<code>val</code>	Returned pre-modified value
<code>__xwrite unsigned int*</code>	<code>data</code>	Value(s) to subtract
<code>volatile void __cls*</code>	<code>address</code>	Address of 32-bit word to which to add values
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to subtract values (1 to 32)
<code>sync_t</code>	<code>sync</code>	Type of synchronization (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

See Also:

- `cls_sub_ptr32()`

3.6.3.96 `cls_test_and_sub_ptr40`

Prototype:

```
void cls_test_and_sub_ptr40(__xread unsigned int* val, __xwrite unsigned int* data,  
volatile void __addr40 __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Test and subtract value from 40 bit Cluster Local Scratch memory address.

Subtract the value in data to the 32-bit word at the specified address. Multiple subtract operations can be performed in which case data needs to point to an array of the appropriate size.

If count is greater than 1, then the address needs to be QW aligned otherwise it has to be LW aligned.

Same as `cls_sub` command, but also returns pre-modified memory contents.



Note

Alignment is not currently enforced.

Table 3.121. `cls_test_and_sub_ptr40` parameters

Type	Name	Description
<code>__xread unsigned int*</code>	<code>val</code>	Returned pre-modified value
<code>__xwrite unsigned int*</code>	<code>data</code>	Value(s) to subtract
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address in which to subtract values where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to subtract values (1 to 32)
<code>sync_t</code>	<code>sync</code>	Type of synchronization (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

See Also:

- `cls_sub_ptr40()`

3.6.3.97 `cls_sub64_ptr32`

Prototype:

```
void cls_sub64_ptr32(__xwrite void* data, volatile void __cls* address, unsigned int
count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Subtract value from Cluster Local Scratch 64-bit memory location.

Subtract the 64-bit value(s) in data from the 64-bit word at the specified address. Multiple subtract operations (count > 2 and must be multiple of 2) can be performed in which case data needs to point to an array of the appropriate size.

Only up to 8 32-bit words are supported by passing it directly. More than 8 32-bit words subtract operations will be handled in indirect format.

Address must be 64-bit aligned.



Note

Alignment is not currently enforced.

Table 3.122. `cls_sub64_ptr32` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Value(s) to subtract
<code>volatile void __cls*</code>	<code>address</code>	Address of 64-bit word from which to subtract values
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to subtract values from (2, 4, .. 16)
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

3.6.3.98 `cls_sub64_ptr40`

Prototype:

```
void cls_sub64_ptr40(__xwrite void* data, volatile void __addr40 __cls* address, unsigned
int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Subtract value from 40 bit Cluster Local Scratch 64bit memory location.

Subtract the 64-bit value(s) in data from the 64-bit word at the specified address. Multiple subtract operations (count > 2 and must be multiple of 2) can be performed in which case data needs to point to an array of the appropriate size.

Only up to 8 32-bit words are supported by passing it directly. More than 8 32-bit words subtract operations will be handled in indirect format.

Address must be 64-bit aligned.



Note

Alignment is not currently enforced.

Table 3.123. `cls_sub64_ptr40` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Value(s) to subtract
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address in which to subtract values where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to subtract values from (2, 4, .. 16)
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)

Type	Name	Description
SIGNAL*	<i>sig_ptr</i>	Signal to be raised on completion

3.6.3.99 `cls_test_and_sub64_ptr32`

Prototype:

```
void cls_test_and_sub64_ptr32(__xread unsigned int* val, __xwrite unsigned int* data,
volatile void __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Test and subtract 64-bit value to Cluster Local Scratch memory address.

Subtract the value in data to the 64-bit word at the specified address. Multiple subtract operations can be performed in which case data needs to point to an array of the appropriate size.

If count is greater than 1, then the address needs to be QW aligned otherwise it has to be LW aligned.

Same as `cls_sub64_ptr32` command, but also returns pre-modified memory contents.



Note

Alignment is not currently enforced.

Table 3.124. `cls_test_and_sub64_ptr32` parameters

Type	Name	Description
<code>__xread unsigned int*</code>	<i>val</i>	Returned pre-modified value
<code>__xwrite unsigned int*</code>	<i>data</i>	Value(s) to add
<code>volatile void __cls*</code>	<i>address</i>	Address of 64-bit word from which to subtract values
<code>unsigned int</code>	<i>count</i>	Number of 32-bit words and must be multiple of 2 up to 16
<code>sync_t</code>	<i>sync</i>	Type of synchronization (<code>sig_done</code> or <code>ctx_swap</code>)
SIGNAL*	<i>sig_ptr</i>	Signal to be raised on completion

See Also:

- `cls_sub64_ptr32()`

3.6.3.100 `cls_test_and_sub64_ptr40`

Prototype:

```
void cls_test_and_sub64_ptr40(__xread unsigned int* val, __xwrite unsigned int* data,
volatile void __addr40 __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Test and subtract 64-bit value to Cluster Local Scratch memory address.

Subtract the value in data from the 64-bit word at the specified address. Multiple subtract operations can be performed in which case data needs to point to an array of the appropriate size.

If count is greater than 1, then the address needs to be QW aligned otherwise it has to be LW aligned.

Same as `cls_sub64_ptr40` command, but also returns pre-modified memory contents.



Note

Alignment is not currently enforced.

Table 3.125. `cls_test_and_sub64_ptr40` parameters

Type	Name	Description
<code>__xread unsigned int*</code>	<code>val</code>	Returned pre-modified value
<code>__xwrite void*</code>	<code>data</code>	Value(s) to add
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address in which to subtract values where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words and must be multiple of 2 up to 16
<code>sync_t</code>	<code>sync</code>	Type of synchronization (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

See Also:

- `cls_sub64_ptr40()`

3.6.3.101 `cls_sub_sat_ptr32`

Prototype:

```
void cls_sub_sat_ptr32(__xwrite void* data, volatile void __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Subtract value from Cluster Local Scratch memory address with saturation.

Subtract the value in data from the 32-bit word at the specified address. Multiple subtract operations (`count > 1`) can be performed in which case data needs to point to an array of the appropriate size.

If the subtraction underflows the value of the 32-bit word will be set to 0x0.

If count is greater than 1, then the address needs to be QW aligned otherwise it has to be LW aligned.



Note

Alignment is not currently enforced.

Table 3.126. `cls_sub_sat_ptr32` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Value(s) to subtract
<code>volatile void __cls*</code>	<code>address</code>	Address of 32-bit word from which to subtract values
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to subtract values from
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

3.6.3.102 `cls_sub_sat_ptr40`

Prototype:

```
void cls_sub_sat_ptr40(__xwrite void* data, volatile void __addr40 __cls* address, unsigned
int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Subtract value from 40 bit Cluster Local Scratch memory address with saturation.

Subtract the value in data from the 32-bit word at the specified address. Multiple subtract operations (`count > 1`) can be performed in which case data needs to point to an array of the appropriate size.

If the subtraction underflows the value of the 32-bit word will be set to 0x0.

If `count` is greater than 1, then the address needs to be QW aligned otherwise it has to be LW aligned.



Note

Alignment is not currently enforced.

Table 3.127. `cls_sub_sat_ptr40` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Value(s) to subtract
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address in which to subtract values where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to subtract values from
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

3.6.3.103 cls_sub_imm_ptr32

Prototype:

```
void cls_sub_imm_ptr32(volatile void __cls* address, unsigned int value)
```

Description:

Subtract an immediate value from Cluster Local Scratch memory address.

Subtract the value in data from the 32-bit word at the specified address. If the value is less than 3 bits it is directly encoded in the instruction, otherwise a indirect reference is used. Values up to 14 bits can be passed in.

This function accepts values for up to 16 bits with bit 14 being used as an optional sign extension. If bit 14 is 0, no sign extension is used. If bit 14 is set, the sign of the value is extended to 32-bit before performing the subtraction.

Bit 15 of value is ignored.

Table 3.128. cls_sub_imm_ptr32 parameters

Type	Name	Description
volatile void __cls*	address	Address of 32-bit word to which to add values
unsigned int	value	Value to subtract

3.6.3.104 cls_sub_imm_ptr40

Prototype:

```
void cls_sub_imm_ptr40(volatile void __addr40 __cls* address, unsigned int value)
```

Description:

Subtract an immediate value from 40 bit Cluster Local Scratch memory address.

Subtract the value in data from the 32-bit word at the specified address. If the value is less than 3 bits it is directly encoded in the instruction, otherwise a indirect reference is used. Values up to 14 bits can be passed in.

This function accepts values for up to 16 bits with bit 14 being used as an optional sign extension. If bit 14 is 0, no sign extension is used. If bit 14 is set, the sign of the value is extended to 32-bit before performing the subtraction.

Bit 15 of value is ignored.

Table 3.129. cls_sub_imm_ptr40 parameters

Type	Name	Description
volatile void __addr40 __cls*	address	40 bit address in which to subtract values where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
unsigned int	value	Value to subtract

3.6.3.105 cls_sub_imm_sat_ptr32

Prototype:

```
void cls_sub_imm_sat_ptr32(volatile void __cls* address, unsigned int value)
```

Description:

Sub a immediate value from Cluster Local Scratch memory address with saturation.

Subtract the value in data to the 32-bit word at the specified address. If the value is less than 3 bits it is directly encoded in the instruction, otherwise a indirect reference is used. If the subtraction underflows the 32-bit word is set to 0x0. Values subtracted can be up to 14 bits long.

This function accepts values for up to 16 bits with bit 14 being used as an optional sign extension. If bit 14 is 0, no sign extension is used. If bit 14 is set, the sign of the value is extended to 32-bit before performing the subtraction.

Bit 15 of value is ignored.

Table 3.130. cls_sub_imm_sat_ptr32 parameters

Type	Name	Description
volatile void __cls*	address	Address of 32-bit word to which to subtract value
unsigned int	value	Value to subtract

3.6.3.106 cls_sub_imm_sat_ptr40

Prototype:

```
void cls_sub_imm_sat_ptr40(volatile void __addr40 __cls* address, unsigned int value)
```

Description:

Sub a immediate value from 40 bit Cluster Local Scratch memory address with saturation.

Subtract the value in data to the 32-bit word at the specified address. If the value is less than 3 bits it is directly encoded in the instruction, otherwise a indirect reference is used. If the subtraction underflows the 32-bit word is set to 0x0. Values subtracted can be up to 14 bits long.

This function accepts values for up to 16 bits with bit 14 being used as an optional sign extension. If bit 14 is 0, no sign extension is used. If bit 14 is set, the sign of the value is extended to 32-bit before performing the subtraction.

Bit 15 of value is ignored.

Table 3.131. cls_sub_imm_sat_ptr40 parameters

Type	Name	Description
volatile void __addr40	address	40 bit address in which to subtract values where upper 6 bits indicate the __cls*. See 6xxx databook for recommended addressing mode.

Type	Name	Description
unsigned int	value	Value to subtract

3.6.3.107 cls_sub64_imm_ptr32

Prototype:

```
void cls_sub64_imm_ptr32(volatile void __cls* address, unsigned int value)
```

Description:

Subtract an immediate value from 64-bit Cluster Local Scratch memory address.

Subtract the value in data from the 64-bit word at the specified address. If the value is less than 3 bits, it is directly encoded in the instruction, otherwise a indirect reference is used. Values can be up to 14 bits long.

Address needs to be 64-bit aligned.

This function accepts values for up to 16 bits with the top 2 bits being used as an optional sign extension. If the top two bits are 00, no sign extension is used. The top bits set to 01 indicate sign extend to 64-bit value, top bits 10 indicate no sign extension but duplicate data in top and bottom fields, top bits equal 11 indicate two sign extended 32-bit values.

Table 3.132. cls_sub64_imm_ptr32 parameters

Type	Name	Description
volatile void __cls*	address	Address of 64-bit word to which to add values
unsigned int	value	Value to subtract

3.6.3.108 cls_sub64_imm_ptr40

Prototype:

```
void cls_sub64_imm_ptr40(volatile void __addr40 __cls* address, unsigned int value)
```

Description:

Subtract an immediate value from 64-bit Cluster Local Scratch memory address.

Subtract the value in data from the 64-bit word at the specified address. If the value is less than 3 bits, it is directly encoded in the instruction, otherwise a indirect reference is used. Values can be up to 14 bits long.

Address needs to be 64-bit aligned.

This function accepts values for up to 16 bits with the top 2 bits being used as an optional sign extension. If the top two bits are 00, no sign extension is used. The top bits set to 01 indicate sign extend to 64-bit value, top bits 10 indicate no sign extension but duplicate data in top and bottom fields, top bits equal 11 indicate two sign extended 32-bit values.

Table 3.133. `cls_sub64_imm_ptr40` parameters

Type	Name	Description
volatile void __addr40 __cls*	address	40 bit address in which to subtract values where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
unsigned int	value	Value to subtract

3.6.3.109 `cls_test_and_sub_imm_ptr32`

Prototype:

```
void cls_test_and_sub_imm_ptr32(__xread unsigned int* val, volatile void __cls* address,
unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch memory test and subtract immediate.

Same as `cls_sub_imm` command, but also returns pre-modified memory contents.

Table 3.134. `cls_test_and_sub_imm_ptr32` parameters

Type	Name	Description
__xread unsigned int*	val	Returned pre-modified value
volatile void __cls*	address	Cluster Local Scratch address to subtract from
unsigned int	value	Immediate value to subtract (1 - 31)
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

See Also:

- `cls_sub_imm_ptr32()`

3.6.3.110 `cls_test_and_sub_imm_ptr40`

Prototype:

```
void cls_test_and_sub_imm_ptr40(__xread unsigned int* val, volatile void __addr40 __cls*
address, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

40-bit Local scratch memory test and subtract immediate.

Same as `cls_sub_imm` command, but also returns pre-modified memory contents.

Table 3.135. `cls_test_and_sub_imm_ptr40` parameters

Type	Name	Description
<code>__xread unsigned int*</code>	<code>val</code>	Returned pre-modified value
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address in which to subtract values where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>value</code>	Immediate value to subtract (1 - 31)
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

See Also:

- `cls_sub_imm_ptr40()`

3.6.3.111 `cls_test_and_sub64_imm_ptr32`

Prototype:

```
void cls_test_and_sub64_imm_ptr32(__xread unsigned int* val, volatile void __cls* address,
unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch memory test and sub immediate from 64bit location.

Same as `cls_sub64_imm` command, but also returns pre-modified memory contents.

Table 3.136. `cls_test_and_sub64_imm_ptr32` parameters

Type	Name	Description
<code>__xread unsigned int*</code>	<code>val</code>	Returned pre-modified value
<code>volatile void __cls*</code>	<code>address</code>	Cluster Local Scratch address to subtract from
<code>unsigned int</code>	<code>value</code>	Immediate value to subtract (1 - 31)
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

See Also:

- `cls_sub64_imm_ptr32()`

3.6.3.112 `cls_test_and_sub64_imm_ptr40`

Prototype:

```
void cls_test_and_sub64_imm_ptr40(__xread unsigned int* val, volatile void __addr40 __cls*
address, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

40-bit Local scratch memory test and sub immediate from 64bit location.

Same as `cls_sub_imm64` command, but also returns pre-modified memory contents.

Table 3.137. `cls_test_and_sub64_imm_ptr40` parameters

Type	Name	Description
<code>__xread unsigned int*</code>	<code>val</code>	Returned pre-modified value
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address in which to subtract values where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>value</code>	Immediate value to subtract (1 - 31)
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

See Also:

- `cls_sub64_imm_ptr32()`

3.6.3.113 `cls_test_and_clear_bits_ptr32`

Prototype:

```
void cls_test_and_clear_bits_ptr32(__xread unsigned int* val, __xwrite unsigned int* mask, volatile void __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Clear Cluster Local Scratch memory bits and return pre-modified value to transfer register.

Table 3.138. `cls_test_and_clear_bits_ptr32` parameters

Type	Name	Description
<code>__xread unsigned int*</code>	<code>val</code>	Returned pre-modified value
<code>__xwrite unsigned int*</code>	<code>mask</code>	Mask of bits to clear
<code>volatile void __cls*</code>	<code>address</code>	Cluster Local Scratch address to clear
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to clear
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.6.3.114 `cls_test_and_clear_bits_ptr40`

Prototype:

```
void cls_test_and_clear_bits_ptr40(__xread unsigned int* val, __xwrite unsigned int* mask, volatile void __addr40 __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Clear 40 bit Cluster Local Scratch memory bits and return pre-modified value to transfer register.

Table 3.139. `cls_test_and_clear_bits_ptr40` parameters

Type	Name	Description
<code>__xread unsigned int*</code>	<code>val</code>	Returned pre-modified value
<code>__xwrite unsigned int*</code>	<code>mask</code>	Mask of bits to clear
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address in which to clear bits where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to clear
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.6.3.115 `cls_test_and_clear_bits_ind_ptr40`

Prototype:

```
void cls_test_and_clear_bits_ind_ptr40(__xread unsigned int* val, __xwrite unsigned int* mask, volatile void __addr40 __cls* address, unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Test and clear 40 bit Cluster Local Scratch memory bits and return pre-modified value to transfer register, with indirect.

Table 3.140. `cls_test_and_clear_bits_ind_ptr40` parameters

Type	Name	Description
<code>__xread unsigned int*</code>	<code>val</code>	Returned pre-modified value
<code>__xwrite unsigned int*</code>	<code>mask</code>	Mask of bits to clear
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address in which to clear bits where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>max_nn</code>	Number of 32-bit words to clear
<code>generic_ind_t</code>	<code>ind</code>	Indirection type
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.6.3.116 `cls_test_and_clear_bits_ind_ptr32`

Prototype:

```
void cls_test_and_clear_bits_ind_ptr32(__xread unsigned int* val, __xwrite unsigned int* mask, volatile void __cls* address, unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Test and clear Cluster Local Scratch memory bits and return pre-modified value to transfer register, with indirect.

Table 3.141. `cls_test_and_clear_bits_ind_ptr32` parameters

Type	Name	Description
<code>__xread unsigned int*</code>	<code>val</code>	Returned pre-modified value
<code>__xwrite unsigned int*</code>	<code>mask</code>	Mask of bits to clear
<code>volatile void __cls*</code>	<code>address</code>	Cluster Local Scratch address to clear
<code>unsigned int</code>	<code>max_nn</code>	Number of 32-bit words to clear
<code>generic_ind_t</code>	<code>ind</code>	Indirection type
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.6.3.117 `cls_test_and_set_bits_ind_ptr40`

Prototype:

```
void cls_test_and_set_bits_ind_ptr40(__xread unsigned int* val, __xwrite unsigned int* mask, volatile void __addr40 __cls* address, unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Test and set 40 bit Cluster Local Scratch memory bits and return pre-modified value to transfer, with indirect register.

Table 3.142. `cls_test_and_set_bits_ind_ptr40` parameters

Type	Name	Description
<code>__xread unsigned int*</code>	<code>val</code>	Returned pre-modified value
<code>__xwrite unsigned int*</code>	<code>mask</code>	Mask of bits to clear
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address in which to clear bits where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>max_nn</code>	Number of 32-bit words to clear
<code>generic_ind_t</code>	<code>ind</code>	Indirection type

Type	Name	Description
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.6.3.118 `cls_test_and_set_bits_ind_ptr32`

Prototype:

```
void cls_test_and_set_bits_ind_ptr32(__xread unsigned int* val, __xwrite unsigned int* mask, volatile void __cls* address, unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Test and set bits Cluster Local Scratch memory bits and return pre-modified value to transfer register, with indirect.

Table 3.143. `cls_test_and_set_bits_ind_ptr32` parameters

Type	Name	Description
__xread unsigned int*	<i>val</i>	Returned pre-modified value
__xwrite unsigned int*	<i>mask</i>	Mask of bits to clear
volatile void __cls*	<i>address</i>	Cluster Local Scratch address to clear
unsigned int	<i>max_nn</i>	Number of 32-bit words to clear
generic_ind_t	<i>ind</i>	Indirection type
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.6.3.119 `cls_test_and_clear_bits_imm_ptr32`

Prototype:

```
void cls_test_and_clear_bits_imm_ptr32(__xread unsigned int* val, volatile void __cls* address, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch memory test and clear bits immediate.

Same as `cls_clear_bits_imm` command, but also returns pre-modified memory contents.

Table 3.144. `cls_test_and_clear_bits_imm_ptr32` parameters

Type	Name	Description
__xread unsigned int*	<i>val</i>	Returned pre-modified value
volatile void __cls*	<i>address</i>	Cluster Local Scratch address to clear

Type	Name	Description
unsigned int	<i>value</i>	Immediate value to clear (1 - 31)
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

See Also:

- `cls_clear_bits_imm_ptr32()`

3.6.3.120 `cls_test_and_clear_bits_imm_ptr40`

Prototype:

```
void cls_test_and_clear_bits_imm_ptr40(__xread unsigned int* val, volatile void __addr40
__cls* address, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

40-bit Local scratch memory test and clear bits immediate.

Same as `cls_clear_bits_imm` command, but also returns pre-modified memory contents.

Table 3.145. `cls_test_and_clear_bits_imm_ptr40` parameters

Type	Name	Description
__xread unsigned int*	<i>val</i>	Returned pre-modified value
volatile void __addr40 __cls*	<i>address</i>	40 bit address in which to clear bits where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
unsigned int	<i>value</i>	Immediate value to clear (1 - 31)
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

See Also:

- `cls_clear_bits_imm_ptr40()`

3.6.3.121 `cls_test_and_set_bits_ptr32`

Prototype:

```
void cls_test_and_set_bits_ptr32(__xread unsigned int* val, __xwrite unsigned int* mask,
volatile void __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set Cluster Local Scratch memory bits and return pre-modified value to transfer register.

Table 3.146. `cls_test_and_set_bits_ptr32` parameters

Type	Name	Description
<code>__xread unsigned int*</code>	<code>val</code>	Returned pre-modified value
<code>__xwrite unsigned int*</code>	<code>mask</code>	Mask of bits to set
<code>volatile void __cls*</code>	<code>address</code>	Cluster Local Scratch address to set
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to set
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.6.3.122 `cls_test_and_set_bits_ptr40`

Prototype:

```
void cls_test_and_set_bits_ptr40(__xread unsigned int* val, __xwrite unsigned int* mask,
volatile void __addr40 __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set 40 bit Cluster Local Scratch memory bits and return pre-modified value to transfer register.

Table 3.147. `cls_test_and_set_bits_ptr40` parameters

Type	Name	Description
<code>__xread unsigned int*</code>	<code>val</code>	Returned pre-modified value
<code>__xwrite unsigned int*</code>	<code>mask</code>	Mask of bits to set
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address in which to set bits where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to set
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.6.3.123 `cls_test_and_set_bits_imm_ptr32`

Prototype:

```
void cls_test_and_set_bits_imm_ptr32(__xread unsigned int* val, volatile void __cls* address,
unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch memory test and set bits immediate.

Same as `cls_set_bits_imm` command, but also returns pre-modified memory contents.

Table 3.148. `cls_test_and_set_bits_imm_ptr32` parameters

Type	Name	Description
<code>__xread unsigned int*</code>	<code>val</code>	Returned pre-modified value
<code>volatile void __cls*</code>	<code>address</code>	Cluster Local Scratch address to set
<code>unsigned int</code>	<code>value</code>	Immediate value to set (1 - 31)
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be <code>sig_done</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

See Also:

- `cls_set_bits_imm_ptr32()`

3.6.3.124 `cls_test_and_set_bits_imm_ptr40`

Prototype:

```
void cls_test_and_set_bits_imm_ptr40(__xread unsigned int* val, volatile void __addr40
__cls* address, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

40-bit Local scratch memory test and set bits immediate.

Same as `cls_set_bits_imm` command, but also returns pre-modified memory contents.

Table 3.149. `cls_test_and_set_bits_imm_ptr40` parameters

Type	Name	Description
<code>__xread unsigned int*</code>	<code>val</code>	Returned pre-modified value
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address in which to set bits where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>value</code>	Immediate value to set (1 - 31)
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be <code>sig_done</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

See Also:

- `cls_set_bits_imm_ptr40()`

3.6.3.125 `cls_test_and_add_sat_ptr32`

Prototype:

```
void cls_test_and_add_sat_ptr32(__xread unsigned int* val, __xwrite unsigned int* data,
volatile void __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Add with saturation data to Cluster Local Scratch memory and return pre-modified value to transfer register.

Table 3.150. `cls_test_and_add_sat_ptr32` parameters

Type	Name	Description
<code>__xread unsigned int*</code>	<code>val</code>	Returned pre-modified value
<code>__xwrite unsigned int*</code>	<code>data</code>	Data to add
<code>volatile void __cls*</code>	<code>address</code>	Cluster Local Scratch address to add data to
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to add
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.6.3.126 `cls_test_and_add_sat_ptr40`

Prototype:

```
void cls_test_and_add_sat_ptr40(__xread unsigned int* val, __xwrite unsigned int* data,
volatile void __addr40 __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Add with saturation data to 40 bit Cluster Local Scratch memory and return pre-modified value to transfer register.

Table 3.151. `cls_test_and_add_sat_ptr40` parameters

Type	Name	Description
<code>__xread unsigned int*</code>	<code>val</code>	Returned pre-modified value
<code>__xwrite unsigned int*</code>	<code>data</code>	Data to add
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address in which to add value where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to add
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.6.3.127 `cls_test_and_add_sat_ind_ptr40`

Prototype:

```
void cls_test_and_add_sat_ind_ptr40(__xread unsigned int* val, __xwrite unsigned int* data,
volatile void __addr40 __cls* address, unsigned int max_nn, generic_ind_t ind,
sync_t sync, SIGNAL* sig_ptr)
```

Description:

Add with saturation data to 40 bit Cluster Local Scratch memory and return pre-modified value to transfer register, with indirect.

Table 3.152. `cls_test_and_add_sat_ind_ptr40` parameters

Type	Name	Description
<code>__xread unsigned int*</code>	<code>val</code>	Returned pre-modified value
<code>__xwrite unsigned int*</code>	<code>data</code>	Data to add
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address in which to add value where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>max_nn</code>	Number of 32-bit words to add
<code>generic_ind_t</code>	<code>ind</code>	Indirection type
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.6.3.128 `cls_test_and_add_sat_ind_ptr32`

Prototype:

```
void cls_test_and_add_sat_ind_ptr32(__xread unsigned int* val, __xwrite unsigned int* data, volatile void __cls* address, unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Add with saturation data to 32 bit Cluster Local Scratch memory and return pre-modified value to transfer register, with indirect.

Table 3.153. `cls_test_and_add_sat_ind_ptr32` parameters

Type	Name	Description
<code>__xread unsigned int*</code>	<code>val</code>	Returned pre-modified value
<code>__xwrite unsigned int*</code>	<code>data</code>	Data to add
<code>volatile void __cls*</code>	<code>address</code>	Cluster Local Scratch address to clear
<code>unsigned int</code>	<code>max_nn</code>	Number of 32-bit words to clear
<code>generic_ind_t</code>	<code>ind</code>	Indirection type
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.6.3.129 `cls_test_and_sub_sat_ind_ptr40`

Prototype:

```
void cls_test_and_sub_sat_ind_ptr40(__xread unsigned int* val, __xwrite unsigned int* data, volatile void __addr40 __cls* address, unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Subtract with saturation data to 40 bit Cluster Local Scratch memory and return pre-modified value to transfer register, with indirect.

Table 3.154. `cls_test_and_sub_sat_ind_ptr40` parameters

Type	Name	Description
__xread unsigned int*	val	Returned pre-modified value
__xwrite unsigned int*	data	Data to subtract
volatile void __addr40 __cls*	address	40 bit address in which to add value where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
unsigned int	max_nn	Number of 32-bit words to subtract
generic_ind_t	ind	Indirection type
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

3.6.3.130 `cls_test_and_sub_sat_ind_ptr32`

Prototype:

```
void cls_test_and_sub_sat_ind_ptr32(__xread unsigned int* val, __xwrite unsigned int* data, volatile void __cls* address, unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Subtract with saturation data to 32 bit Cluster Local Scratch memory and return pre-modified value to transfer register, with indirect.

Table 3.155. `cls_test_and_sub_sat_ind_ptr32` parameters

Type	Name	Description
__xread unsigned int*	val	Returned pre-modified value
__xwrite unsigned int*	data	Data to subtract
volatile void __cls*	address	Cluster Local Scratch address to subtract
unsigned int	max_nn	Number of 32-bit words to subtract
generic_ind_t	ind	Indirection type
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

3.6.3.131 cls_test_and_add_imm_sat_ptr32

Prototype:

```
void cls_test_and_add_imm_sat_ptr32(__xread unsigned int* val, volatile void __cls*
address, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch memory test and add immediate with saturate.

Same as cls_add_imm_sat command, but also returns pre-modified memory contents.

Table 3.156. cls_test_and_add_imm_sat_ptr32 parameters

Type	Name	Description
<code>__xread unsigned int*</code>	<code>val</code>	Returned pre-modified value
<code>volatile void __cls*</code>	<code>address</code>	Cluster Local Scratch address to add
<code>unsigned int</code>	<code>value</code>	Immediate value to add (1 - 31)
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be <code>sig_done</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

See Also:

- `cls_add_imm_sat_ptr32()`

3.6.3.132 cls_test_and_add_imm_sat_ptr40

Prototype:

```
void cls_test_and_add_imm_sat_ptr40(__xread unsigned int* val, volatile void __addr40
__cls* address, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

40-bit Local scratch memory test and add immediate with saturate.

Same as cls_add_imm_sat command, but also returns pre-modified memory contents.

Table 3.157. cls_test_and_add_imm_sat_ptr40 parameters

Type	Name	Description
<code>__xread unsigned int*</code>	<code>val</code>	Returned pre-modified value
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address in which to add value where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>value</code>	Immediate value to add (1 - 31)
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be <code>sig_done</code>)

Type	Name	Description
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

See Also:

- `cls_add_imm_sat_ptr40()`

3.6.3.133 `cls_test_and_sub_sat_ptr32`

Prototype:

```
void cls_test_and_sub_sat_ptr32(__xread unsigned int* val, __xwrite unsigned int* data,
volatile void __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Subtract with saturation data from Cluster Local Scratch memory and return pre-modified value to transfer register.

Table 3.158. `cls_test_and_sub_sat_ptr32` parameters

Type	Name	Description
<code>__xread unsigned int*</code>	<i>val</i>	Returned pre-modified value
<code>__xwrite unsigned int*</code>	<i>data</i>	Data to subtract
<code>volatile void __cls*</code>	<i>address</i>	Cluster Local Scratch address to subtract data from
<code>unsigned int</code>	<i>count</i>	Number of 32-bit words to sub
<code>sync_t</code>	<i>sync</i>	Type of synchronization to use (<code>sig_done</code> or <code>ctx_swap</code>)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.6.3.134 `cls_test_and_sub_sat_ptr40`

Prototype:

```
void cls_test_and_sub_sat_ptr40(__xread unsigned int* val, __xwrite unsigned int* data,
volatile void __addr40 __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Subtract with saturation data from 40 bit Cluster Local Scratch memory and return pre-modified value to transfer register.

Table 3.159. `cls_test_and_sub_sat_ptr40` parameters

Type	Name	Description
<code>__xread unsigned int*</code>	<i>val</i>	Returned pre-modified value
<code>__xwrite unsigned int*</code>	<i>data</i>	Data to subtract

Type	Name	Description
volatile void __addr40 __cls*	address	40 bit address in which to subtract value where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
unsigned int	count	Number of 32-bit words to sub
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

3.6.3.135 cls_test_and_sub_imm_sat_ptr32

Prototype:

```
void cls_test_and_sub_imm_sat_ptr32(__xread unsigned int* val, volatile void __cls* address, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch memory test and sub immediate with saturate.

Same as cls_sub_imm command, but also returns pre-modified memory contents.

Table 3.160. cls_test_and_sub_imm_sat_ptr32 parameters

Type	Name	Description
__xread unsigned int*	val	Returned pre-modified value
volatile void __cls*	address	Cluster Local Scratch address to sub
unsigned int	value	Immediate value to sub (1 - 31)
sync_t	sync	Type of synchronization to use (must be sig_done)
SIGNAL*	sig_ptr	Signal to raise upon completion

See Also:

- `cls_sub_imm_ptr32()`

3.6.3.136 cls_test_and_sub_imm_sat_ptr40

Prototype:

```
void cls_test_and_sub_imm_sat_ptr40(__xread unsigned int* val, volatile void __addr40 __cls* address, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

40-bit Local scratch memory test and sub immediate with saturate.

Same as cls_sub_imm command, but also returns pre-modified memory contents.

Table 3.161. `cls_test_and_sub_imm_sat_ptr40` parameters

Type	Name	Description
<code>__xread unsigned int*</code>	<code>val</code>	Returned pre-modified value
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address in which to subtract value where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>value</code>	Immediate value to sub (1 - 31)
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be <code>sig_done</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

See Also:

- `cls_sub_imm_ptr40()`

3.6.3.137 `cls_incr_ptr32`

Prototype:

```
void cls_incr_ptr32(volatile void __addr32 __cls* address)
```

Description:

Increment by one (1) a 32-bit value in Cluster local scratch memory.

Implemented as sub_imm with an immediate value of 0xff (-1).

Table 3.162. `cls_incr_ptr32` parameters

Type	Name	Description
<code>volatile void __addr32 __cls*</code>	<code>address</code>	32-bit Cluster Local Scratch address to increment

3.6.3.138 `cls_incr_ptr40`

Prototype:

```
void cls_incr_ptr40(volatile void __addr40 __cls* address)
```

Description:

Increment by one (1) a 32-bit value in Cluster local scratch memory.

Implemented as sub_imm with an immediate value of 0xff (-1).

Table 3.163. `cls_incr_ptr40` parameters

Type	Name	Description
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40-bit Cluster Local Scratch address to increment

3.6.3.139 `cls_decr_ptr32`

Prototype:

```
void cls_decr_ptr32(volatile void __addr32 __cls* address)
```

Description:

Decrement by one (1) a 32-bit value in Cluster local scratch memory.

Implemented as add_imm with an immediate value of 0xff (-1).

Table 3.164. `cls_decr_ptr32` parameters

Type	Name	Description
<code>volatile void __addr32 __cls*</code>	<code>address</code>	32-bit Cluster Local Scratch address to decrement

3.6.3.140 `cls_decr_ptr40`

Prototype:

```
void cls_decr_ptr40(volatile void __addr40 __cls* address)
```

Description:

Decrement by one (1) a 32-bit value in Cluster local scratch memory.

Implemented as add_imm with an immediate value of 0xff (-1).

Table 3.165. `cls_decr_ptr40` parameters

Type	Name	Description
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40-bit Cluster Local Scratch address to decrement

3.6.3.141 `cls_incr64_ptr32`

Prototype:

```
void cls_incr64_ptr32(volatile void __addr32 __cls* address)
```

Description:

Increment by one (1) a 64-bit value in Cluster local scratch memory.

Implemented as sub64_imm with an immediate value of 0xff (-1).

Table 3.166. `cls_incr64_ptr32` parameters

Type	Name	Description
volatile void __addr32 __cls*	address	32-bit Cluster Local Scratch address to increment

3.6.3.142 `cls_incr64_ptr40`

Prototype:

```
void cls_incr64_ptr40(volatile void __addr40 __cls* address)
```

Description:

Increment by one (1) a 64-bit value in Cluster local scratch memory.

Implemented as sub64_imm with an immediate value of 0xff (-1).

Table 3.167. `cls_incr64_ptr40` parameters

Type	Name	Description
volatile void __addr40 __cls*	address	40-bit Cluster Local Scratch address to increment

3.6.3.143 `cls_decr64_ptr32`

Prototype:

```
void cls_decr64_ptr32(volatile void __addr32 __cls* address)
```

Description:

Decrement by one (1) a 64-bit value in Cluster local scratch memory.

Implemented as add64_imm with an immediate value of 0xff (-1).

Table 3.168. `cls_decr64_ptr32` parameters

Type	Name	Description
volatile void __addr32 __cls*	address	32-bit Cluster Local Scratch address to decrement

3.6.3.144 cls_decr64_ptr40

Prototype:

```
void cls_decr64_ptr40(volatile void __addr40 __cls* address)
```

Description:

Decrement by one (1) a 64-bit value in Cluster local scratch memory.

Implemented as add64_imm with an immediate value of 0xff (-1).

Table 3.169. cls_decr64_ptr40 parameters

Type	Name	Description
volatile void __addr40 __cls*	address	40-bit Cluster Local Scratch address to decrement

3.6.3.145 cls_cmp_read_write_ptr32

Prototype:

```
void cls_cmp_read_write_ptr32(__xrw unsigned int* data, unsigned int mask, volatile void __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read, compare then write data from Cluster Local scratch to transfer register.

The byte mask bits indicate which bytes must match pull data for write of all the pull data to happen.

If it matches the 'pull' data in every byte where the byte mask is 1, then the SRAM data is overwritten with the pull data.

Table 3.170. cls_cmp_read_write_ptr32 parameters

Type	Name	Description
__xrw unsigned int*	data	Data to read/write
unsigned int	mask	Byte mask bits value
volatile void __cls*	address	Cluster Local Scratch address to read/write from
unsigned int	count	Number of 4-byte 32-bit words to read/write
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

3.6.3.146 cls_cmp_read_write_ptr40

Prototype:

```
void cls_cmp_read_write_ptr40(__xrw unsigned int* data, unsigned int mask, volatile void
__addr40 __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read, compare then write data from 40 bit Cluster Local Scratch to transfer register.

The byte mask bits indicate which bytes must match pull data for write of all the pull data to happen.

If it matches the 'pull' data in every byte where the byte mask is 1, then the SRAM data is overwritten with the pull data.

Table 3.171. `cls_cmp_read_write_ptr40` parameters

Type	Name	Description
<code>__xrw unsigned int*</code>	<code>data</code>	Data to read/write
<code>unsigned int</code>	<code>mask</code>	Byte mask bits value
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address in which to read/write from where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>count</code>	Number of 4-byte 32-bit words to read/write
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.6.3.147 `cls_cmp_read_write_ind_ptr40`

Prototype:

```
void cls_cmp_read_write_ind_ptr40(__xrw void* data, volatile void __addr40 __cls* address,
unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read, compare then write data from Cluster Local scratch to transfer register in indirect format.

Table 3.172. `cls_cmp_read_write_ind_ptr40` parameters

Type	Name	Description
<code>__xrw void*</code>	<code>data</code>	Data to read/write
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address in which to read/write from where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>max_nn</code>	Max number of 4-byte 32-bit words to read/write
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.6.3.148 cls_cmp_read_write_ind_ptr32

Prototype:

```
void cls_cmp_read_write_ind_ptr32(__xrw void* data, volatile void __cls* address, unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read, compare then write data from Cluster Local scratch to transfer register in indirect format.

Table 3.173. cls_cmp_read_write_ind_ptr32 parameters

Type	Name	Description
<code>__xrw void*</code>	<code>data</code>	Data to read/write
<code>volatile void __cls*</code>	<code>address</code>	Cluster Local Scratch address to read/write from
<code>unsigned int</code>	<code>max_nn</code>	Max number of 4-byte 32-bit words to read/write
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.6.3.149 cls_queue_lock_ind_ptr40

Prototype:

```
void cls_queue_lock_ind_ptr40(volatile void __addr40 __cls* address, generic_ind_t ind, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Local scratch Queue Lock Claim in indirect mode.

Table 3.174. cls_queue_lock_ind_ptr40 parameters

Type	Name	Description
<code>volatile void __addr40 __cls*</code>	<code>address</code>	Cluster Local Scratch 40 bit address to read into
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done only)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal pair to raise upon completion

3.6.3.150 cls_queue_lock_ind_ptr32

Prototype:

```
void cls_queue_lock_ind_ptr32(volatile void __cls* address, generic_ind_t ind, sync_t
sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Local scratch Queue Lock Claim in indirect mode.

Table 3.175. `cls_queue_lock_ind_ptr32` parameters

Type	Name	Description
volatile void __cls*	address	Cluster Local Scratch 32 bit address to read into
generic_ind_t	ind	Indirect word
sync_t	sync	Type of synchronization to use (sig_done only)
SIGNAL_PAIR*	sig_pair_ptr	Signal pair to raise upon completion

3.6.3.151 `cls_queue_lock_ptr32`

Prototype:

```
void cls_queue_lock_ptr32(volatile void __cls* address, sync_t sync, SIGNAL_PAIR*
sig_pair_ptr)
```

Description:

Local scratch Queue Lock Claim.

Attempt to acquire lock and respond with sig_name when lock is acquired. If lock can not be immediately acquired, enqueue the signal to send once it can be acquired. Up to 5 additional signals can be enqueued. If the queue is already full, respond with sig_name and sig_name+1.

Table 3.176. `cls_queue_lock_ptr32` parameters

Type	Name	Description
volatile void __cls*	address	Cluster Local Scratch address to read into
sync_t	sync	Type of synchronization to use (sig_done only)
SIGNAL_PAIR*	sig_pair_ptr	Signal pair to raise upon completion. Sending sig_name is used to indicate that the operation completed. If sig_name+1 is also sent, it indicates that the operation completed, but failed because the queue was full.

3.6.3.152 `cls_queue_lock_ptr40`

Prototype:

```
void cls_queue_lock_ptr40(volatile void __addr40 __cls* address, sync_t sync, SIGNAL_PAIR*
sig_pair_ptr)
```

Description:

40-bit Local scratch Queue Lock Claim.

Attempt to acquire lock and respond with sig_name when lock is acquired. If lock can not be immediately acquired, enqueue the signal to send once it can be acquired. Up to 5 additional signals can be enqueued. If the queue is already full, respond with sig_name and sig_name+1.

Table 3.177. cls_queue_lock_ptr40 parameters

Type	Name	Description
volatile void __addr40 __cls*	address	40 bit address in which to read into where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
sync_t	sync	Type of synchronization to use (sig_done only)
SIGNAL_PAIR*	sig_pair_ptr	Signal pair to raise upon completion. Sending sig_name is used to indicate that the operation completed. If sig_name+1 is also sent, it indicates that the operation completed, but failed because the queue was full.

3.6.3.153 cls_queue_unlock_ptr32

Prototype:

```
void cls_queue_unlock_ptr32(volatile void __cls* address)
```

Description:

Local scratch Queue UnLock Claim.

Release an already acquired lock. If the lock-queue has any pending signals enqueued do not release the lock, but instead dequeue the first signal off the queue and trigger it. If there are no pending signals in the queue, then no signals are triggered and the queue is marked as empty.

Table 3.178. cls_queue_unlock_ptr32 parameters

Type	Name	Description
volatile void __cls*	address	Cluster Local Scratch address to read into

3.6.3.154 cls_queue_unlock_ptr40

Prototype:

```
void cls_queue_unlock_ptr40(volatile void __addr40 __cls* address)
```

Description:

40-bit Local scratch Queue UnLock Claim.

Release an already acquired lock. If the lock-queue has any pending signals enqueued do not release the lock, but instead dequeue the first signal off the queue and trigger it. If there are no pending signals in the queue, then no signals are triggered and the queue is marked as empty.

Table 3.179. `cls_queue_unlock_ptr40` parameters

Type	Name	Description
volatile void __addr40 __cls*	address	40 bit address in which to read into where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.

3.6.3.155 `cls_hash_mask_ptr32`

Prototype:

```
void cls_hash_mask_ptr32(__xwrite void* data, volatile void __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch memory Hash.

Create a 64-bit hash_index over the transfer registers as follows: hash_index_select = address[18:16], hash_mask_address = address[15:0], hash_index = hash_indexes[hash_index_select].

For each 64-bit data value in the transfer register pairs: data = data AND cls_read64(hash_mask_address), calculate new hash_index as indicated above, hash_mask_address = hash_mask_address + 8, hash_indexes[hash_index_select] = hash_index.

Table 3.180. `cls_hash_mask_ptr32` parameters

Type	Name	Description
__xwrite void*	data	Data to hash
volatile void __cls*	address	Address to hash
unsigned int	count	Number of 32-bit words to produce hash_index over. Valid values in multiples of 2 in the range 2-32.
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

3.6.3.156 `cls_hash_mask_ptr40`

Prototype:

```
void cls_hash_mask_ptr40(__xwrite void* data, volatile void __addr40 __cls* address,  
unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

40-bit Local scratch memory Hash.

Create a 64-bit hash_index over the transfer registers as follows: hash_index_select = address[18:16], hash_mask_address = address[15:0], hash_index = hash_indexes[hash_index_select].

For each 64-bit data value in the transfer register pairs: data = data AND cls_read64(hash_mask_address), calculate new hash_index as indicated above, hash_mask_address = hash_mask_address + 8, hash_indexes[hash_index_select] = hash_index.

Table 3.181. `cls_hash_mask_ptr40` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Data to hash
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address to hash where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to produce hash_index over. Valid values in multiples of 2 in the range 2-32.
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.6.3.157 `cls_hash_mask_ind_ptr40`

Prototype:

```
void cls_hash_mask_ind_ptr40(__xwrite void* data, volatile void __addr40 __cls* address,
unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch memory Hash in indirect mode.

Table 3.182. `cls_hash_mask_ind_ptr40` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Data to hash
<code>volatile void __addr40 __cls*</code>	<code>address</code>	Address to hash, 40 bit
<code>unsigned int</code>	<code>max_nn</code>	Max number of 32-bit words to produce hash_index over
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.6.3.158 `cls_hash_mask_ind_ptr32`

Prototype:

```
void cls_hash_mask_ind_ptr32(__xwrite void* data, volatile void __cls* address, unsigned
int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch memory Hash in indirect mode.

Table 3.183. `cls_hash_mask_ind_ptr32` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Data to hash
<code>volatile void __cls*</code>	<code>address</code>	Address to hash, 32 bit
<code>unsigned int</code>	<code>max_nn</code>	Max number of 32-bit words to produce hash_index over
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.6.3.159 `cls_hash_mask_clear_ptr32`

Prototype:

```
void cls_hash_mask_clear_ptr32(__xwrite void* data, volatile void __cls* address, unsigned
int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch memory Hash Clear.

Same as `hash_mask` command, but also clears the initial value of `hash_index` before calculating the new value.

Table 3.184. `cls_hash_mask_clear_ptr32` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Data to hash
<code>volatile void __cls*</code>	<code>address</code>	Address to hash
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to produce hash_index over. Valid values in multiples of 2 in the range 2-32.
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

See Also:

- `cls_hash_mask()`

3.6.3.160 cls_hash_mask_clear_ptr40

Prototype:

```
void cls_hash_mask_clear_ptr40(__xwrite void* data, volatile void __addr40 __cls* address,
unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

40-bit Local scratch memory Hash Clear.

Same as hash_mask command, but also clears the initial value of hash_index before calculating the new value.

Table 3.185. cls_hash_mask_clear_ptr40 parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Data to hash
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address to hash where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to produce hash_index over. Valid values in multiples of 2 in the range 2-32.
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

See Also:

- `cls_hash_mask()`

3.6.3.161 cls_cam_lookup_ind_ptr40

Prototype:

```
void cls_cam_lookup_ind_ptr40(__xread void* data_out, __xwrite void* data_in, volatile
void __addr40 __cls* address, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch Content Addressable Memory lookup word in indirect mode.

Table 3.186. cls_cam_lookup_ind_ptr40 parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __addr40 __cls*</code>	<code>address</code>	Local scratch CAM 40 bit address to lookup, 40 bit
<code>generic_ind_t</code>	<code>ind</code>	Indirect word

Type	Name	Description
sync_t	<i>sync</i>	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to be raised upon completion

See Also:

- [cls_cam_lookup\(\)](#)

3.6.3.162 `cls_cam_lookup_ind_ptr32`

Prototype:

```
void cls_cam_lookup_ind_ptr32(__xread void* data_out, __xwrite void* data_in, volatile
void __cls* address, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch Content Addressable Memory lookup word in indirect mode.

Table 3.187. `cls_cam_lookup_ind_ptr32` parameters

Type	Name	Description
__xread void*	<i>data_out</i>	Data looked up
__xwrite void*	<i>data_in</i>	Data to lookup
volatile void __cls*	<i>address</i>	Local scratch CAM 32 bit address to lookup
generic_ind_t	<i>ind</i>	Indirect word
sync_t	<i>sync</i>	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to be raised upon completion

See Also:

- [cls_cam_lookup\(\)](#)

3.6.3.163 `cls_cam_lookup_add_ind_ptr40`

Prototype:

```
void cls_cam_lookup_add_ind_ptr40(__xread void* data_out, __xwrite void* data_in, volatile
void __addr40 __cls* address, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch Content Addressable Memory lookup word with add.

Table 3.188. `cls_cam_lookup_add_ind_ptr40` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __addr40 __cls*</code>	<code>address</code>	Local scratch CAM 40 bit address to lookup, 40 bit
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised upon completion

See Also:

- `cls_cam_lookup()`

3.6.3.164 `cls_cam_lookup_add_ind_ptr32`

Prototype:

```
void cls_cam_lookup_add_ind_ptr32(__xread void* data_out, __xwrite void* data_in, volatile void __cls* address, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch Content Addressable Memory lookup word with add.

Table 3.189. `cls_cam_lookup_add_ind_ptr32` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __cls*</code>	<code>address</code>	Local scratch CAM 32 bit address to lookup
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised upon completion

See Also:

- `cls_cam_lookup()`

3.6.3.165 `cls_cam_lookup24_ind_ptr40`

Prototype:

```
void cls_cam_lookup24_ind_ptr40(__xread void* data_out, __xwrite void* data_in, volatile void __addr40 __cls* address, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch Content Addressable Memory lookup 24-bits in indirect mode.

Table 3.190. `cls_cam_lookup24_ind_ptr40` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __addr40 __cls*</code>	<code>address</code>	Local scratch CAM 40 bit address to lookup, 40 bit
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised upon completion

See Also:

- `cls_cam_lookup()`

3.6.3.166 `cls_cam_lookup24_ind_ptr32`

Prototype:

```
void cls_cam_lookup24_ind_ptr32(__xread void* data_out, __xwrite void* data_in, volatile void __cls* address, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch Content Addressable Memory lookup 24-bits in indirect mode.

Table 3.191. `cls_cam_lookup24_ind_ptr32` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __cls*</code>	<code>address</code>	Local scratch CAM 32 bit address to lookup
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised upon completion

See Also:

- `cls_cam_lookup()`

3.6.3.167 cls_cam_lookup24_add_ind_ptr40

Prototype:

```
void cls_cam_lookup24_add_ind_ptr40(__xread void* data_out, __xwrite void* data_in,
volatile void __addr40 __cls* address, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch Content Addressable Memory lookup 24-bits with add in indirect mode.

Table 3.192. cls_cam_lookup24_add_ind_ptr40 parameters

Type	Name	Description
__xread void*	<i>data_out</i>	Data looked up
__xwrite void*	<i>data_in</i>	Data to lookup
volatile void __addr40 __cls*	<i>address</i>	Local scratch CAM 40 bit address to lookup, 40 bit
generic_ind_t	<i>ind</i>	Indirect word
sync_t	<i>sync</i>	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to be raised upon completion

See Also:

- [cls_cam_lookup\(\)](#)

3.6.3.168 cls_cam_lookup24_add_ind_ptr32

Prototype:

```
void cls_cam_lookup24_add_ind_ptr32(__xread void* data_out, __xwrite void* data_in,
volatile void __cls* address, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch Content Addressable Memory lookup 24-bits with add in indirect mode.

Table 3.193. cls_cam_lookup24_add_ind_ptr32 parameters

Type	Name	Description
__xread void*	<i>data_out</i>	Data looked up
__xwrite void*	<i>data_in</i>	Data to lookup
volatile void __cls*	<i>address</i>	Local scratch CAM 32 bit address to lookup
generic_ind_t	<i>ind</i>	Indirect word
sync_t	<i>sync</i>	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to be raised upon completion

See Also:

- `cls_cam_lookup()`

3.6.3.169 `cls_cam_lookup24_add_inc_ind_ptr40`

Prototype:

```
void cls_cam_lookup24_add_inc_ptr40(__xread void* data_out, __xwrite void* data_in,
volatile void __addr40 __cls* address, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch Content Addressable Memory lookup 24-bits with add or increment usage in indirect mode.

Table 3.194. `cls_cam_lookup24_add_inc_ptr40` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __addr40 __cls*</code>	<code>address</code>	Local scratch CAM 40 bit address to lookup, 40 bit
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised upon completion

See Also:

- `cls_cam_lookup()`

3.6.3.170 `cls_cam_lookup24_add_inc_ptr32`

Prototype:

```
void cls_cam_lookup24_add_inc_ptr32(__xread void* data_out, __xwrite void* data_in,
volatile void __cls* address, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch Content Addressable Memory lookup 24-bits with add or increment usage in indirect mode.

Table 3.195. `cls_cam_lookup24_add_inc_ptr32` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __cls*</code>	<code>address</code>	Local scratch CAM 32 bit address to lookup

Type	Name	Description
generic_ind_t	<i>ind</i>	Indirect word
sync_t	<i>sync</i>	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to be raised upon completion

See Also:

- [cls_cam_lookup\(\)](#)

3.6.3.171 `cls_cam_lookup16_ind_ptr40`

Prototype:

```
void cls_cam_lookup16_ind_ptr40(__xread void* data_out, __xwrite void* data_in, volatile
void __addr40 __cls* address, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch Content Addressable Memory lookup 16-bits in indirect mode.

Table 3.196. `cls_cam_lookup16_ind_ptr40` parameters

Type	Name	Description
__xread void*	<i>data_out</i>	Data looked up
__xwrite void*	<i>data_in</i>	Data to lookup
volatile void __addr40 __cls*	<i>address</i>	Local scratch CAM 40 bit address to lookup, 40 bit
generic_ind_t	<i>ind</i>	Indirect word
sync_t	<i>sync</i>	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to be raised upon completion

See Also:

- [cls_cam_lookup\(\)](#)

3.6.3.172 `cls_cam_lookup16_ind_ptr32`

Prototype:

```
void cls_cam_lookup16_ind_ptr32(__xread void* data_out, __xwrite void* data_in, volatile
void __cls* address, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch Content Addressable Memory lookup 16-bits in indirect mode.

Table 3.197. `cls_cam_lookup16_ind_ptr32` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __cls*</code>	<code>address</code>	Local scratch CAM 32 bit address to lookup
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised upon completion

See Also:

- `cls_cam_lookup()`

3.6.3.173 `cls_cam_lookup16_add_ind_ptr40`

Prototype:

```
void cls_cam_lookup16_add_ind_ptr40(__xread void* data_out, __xwrite void* data_in,
volatile void __addr40 __cls* address, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch Content Addressable Memory lookup 16-bits with add in indirect mode.

Table 3.198. `cls_cam_lookup16_add_ind_ptr40` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __addr40 __cls*</code>	<code>address</code>	Local scratch CAM 40 bit address to lookup, 40 bit
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised upon completion

See Also:

- `cls_cam_lookup()`

3.6.3.174 `cls_cam_lookup16_add_ind_ptr32`

Prototype:

```
void cls_cam_lookup16_add_ind_ptr32(__xread void* data_out, __xwrite void* data_in,
volatile void __cls* address, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch Content Addressable Memory lookup 16-bits with add in indirect mode.

Table 3.199. `cls_cam_lookup16_add_ind_ptr32` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __cls*</code>	<code>address</code>	Local scratch CAM 32 bit address to lookup
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised upon completion

See Also:

- `cls_cam_lookup()`

3.6.3.175 `cls_cam_lookup8_ind_ptr40`

Prototype:

```
void cls_cam_lookup8_ind_ptr40(__xread void* data_out, __xwrite void* data_in, volatile void __addr40 __cls* address, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch Content Addressable Memory lookup 8-bits in indirect mode.

Table 3.200. `cls_cam_lookup8_ind_ptr40` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __addr40 __cls*</code>	<code>address</code>	Local scratch CAM 40 bit address to lookup, 40 bit
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised upon completion

See Also:

- `cls_cam_lookup()`

3.6.3.176 `cls_cam_lookup8_ind_ptr32`

Prototype:

```
void cls_cam_lookup8_ind_ptr32(__xread void* data_out, __xwrite void* data_in, volatile
void __cls* address, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch Content Addressable Memory lookup 8-bits in indirect mode.

Table 3.201. `cls_cam_lookup8_ind_ptr32` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __cls*</code>	<code>address</code>	Local scratch CAM 32 bit address to lookup
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised upon completion

See Also:

- `cls_cam_lookup()`

3.6.3.177 `cls_cam_lookup8_add_ind_ptr40`

Prototype:

```
void cls_cam_lookup8_add_ind_ptr40(__xread void* data_out, __xwrite void* data_in, volatile
void __addr40 __cls* address, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch Content Addressable Memory lookup 8-bits with add in indirect mode.

Table 3.202. `cls_cam_lookup8_add_ind_ptr40` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __addr40 __cls*</code>	<code>address</code>	Local scratch CAM 40 bit address to lookup, 40 bit
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised upon completion

See Also:

- `cls_cam_lookup()`

3.6.3.178 `cls_cam_lookup8_add_ind_ptr32`

Prototype:

```
void cls_cam_lookup8_add_ind_ptr32(__xread void* data_out, __xwrite void* data_in, volatile
void __cls* address, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch Content Addressable Memory lookup 8-bits with add in indirect mode.

Table 3.203. `cls_cam_lookup8_add_ind_ptr32` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __cls*</code>	<code>address</code>	Local scratch CAM 32 bit address to lookup
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised upon completion

See Also:

- `cls_cam_lookup()`

3.6.3.179 `cls_tcam_lookup_ind_ptr40`

Prototype:

```
void cls_tcam_lookup_ind_ptr40(__xread void* data_out, __xwrite void* data_in, volatile
void __addr40 __cls* address, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch TCAM lookup word in indirect mode.

Table 3.204. `cls_tcam_lookup_ind_ptr40` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __addr40 __cls*</code>	<code>address</code>	Local scratch TCAM 40 bit address to lookup, 40 bit

Type	Name	Description
generic_ind_t	<i>ind</i>	Indirect word
sync_t	<i>sync</i>	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to be raised upon completion

See Also:

- [cls_cam_lookup\(\)](#)

3.6.3.180 `cls_tcam_lookup_ind_ptr32`

Prototype:

```
void cls_tcam_lookup_ind_ptr32(__xread void* data_out, __xwrite void* data_in, volatile
void __cls* address, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch TCAM lookup word in indirect mode.

Table 3.205. `cls_tcam_lookup_ind_ptr32` parameters

Type	Name	Description
__xread void*	<i>data_out</i>	Data looked up
__xwrite void*	<i>data_in</i>	Data to lookup
volatile void __cls*	<i>address</i>	Local scratch TCAM 32 bit address to lookup
generic_ind_t	<i>ind</i>	Indirect word
sync_t	<i>sync</i>	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to be raised upon completion

See Also:

- [cls_cam_lookup\(\)](#)

3.6.3.181 `cls_tcam_lookup24_ind_ptr40`

Prototype:

```
void cls_tcam_lookup24_ind_ptr40(__xread void* data_out, __xwrite void* data_in, volatile
void __addr40 __cls* address, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch TCAM lookup 24-bits in indirect mode.

Table 3.206. `cls_tcam_lookup24_ind_ptr40` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __addr40 __cls*</code>	<code>address</code>	Local scratch TCAM 40 bit address to lookup, 40 bit
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised upon completion

See Also:

- `cls_cam_lookup()`

3.6.3.182 `cls_tcam_lookup24_ind_ptr32`

Prototype:

```
void cls_tcam_lookup24_ind_ptr32(__xread void* data_out, __xwrite void* data_in, volatile void __cls* address, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch TCAM lookup 24-bits in indirect mode.

Table 3.207. `cls_tcam_lookup24_ind_ptr32` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __cls*</code>	<code>address</code>	Local scratch TCAM 32 bit address to lookup
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised upon completion

See Also:

- `cls_cam_lookup()`

3.6.3.183 `cls_tcam_lookup16_ind_ptr40`

Prototype:

```
void cls_tcam_lookup16_ind_ptr40(__xread void* data_out, __xwrite void* data_in, volatile void __addr40 __cls* address, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch TCAM lookup 16-bits in indirect mode.

Table 3.208. `cls_tcam_lookup16_ind_ptr40` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __addr40 __cls*</code>	<code>address</code>	Local scratch TCAM 40 bit address to lookup, 40 bit
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised upon completion

See Also:

- `cls_cam_lookup()`

3.6.3.184 `cls_tcam_lookup16_ind_ptr32`

Prototype:

```
void cls_tcam_lookup16_ind_ptr32(__xread void* data_out, __xwrite void* data_in, volatile void __cls* address, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch TCAM lookup 16-bits in indirect mode.

Table 3.209. `cls_tcam_lookup16_ind_ptr32` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __cls*</code>	<code>address</code>	Local scratch TCAM 32 bit address to lookup
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised upon completion

See Also:

- `cls_cam_lookup()`

3.6.3.185 `cls_tcam_lookup8_ind_ptr40`

Prototype:

```
void cls_tcam_lookup8_ind_ptr40(__xread void* data_out, __xwrite void* data_in, volatile
void __addr40 __cls* address, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch TCAM lookup 8-bits in indirect mode.

Table 3.210. `cls_tcam_lookup8_ind_ptr40` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __addr40 __cls*</code>	<code>address</code>	Local scratch TCAM 40 bit address to lookup, 40 bit
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised upon completion

See Also:

- `cls_cam_lookup()`

3.6.3.186 `cls_tcam_lookup8_ind_ptr32`

Prototype:

```
void cls_tcam_lookup8_ind_ptr32(__xread void* data_out, __xwrite void* data_in, volatile
void __cls* address, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch TCAM lookup 8-bits in indirect mode.

Table 3.211. `cls_tcam_lookup8_ind_ptr32` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __cls*</code>	<code>address</code>	Local scratch TCAM 32 bit address to lookup
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised upon completion

See Also:

- `cls_cam_lookup()`

3.6.3.187 `cls_cam_lookup_ptr32`

Prototype:

```
void cls_cam_lookup_ptr32(__xread void* data_out, __xwrite void* data_in, volatile void
__cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch Content Addressable Memory lookup word.

Use 32-bit values from transfer register and compare with 32-bit input data.

Table 3.212. `cls_cam_lookup_ptr32` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __cls*</code>	<code>address</code>	32 bit local scratch CAM address to lookup
<code>unsigned int</code>	<code>count</code>	Size of CAM in number of 64-bit words (1-32)
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised upon completion

3.6.3.188 `cls_cam_lookup_ptr40`

Prototype:

```
void cls_cam_lookup_ptr40(__xread void* data_out, __xwrite void* data_in, volatile void
__addr40 __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch Content Addressable Memory lookup word.

Use 32-bit values from transfer register and compare with 32-bit input data.

Table 3.213. `cls_cam_lookup_ptr40` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit local scratch CAM address to lookup where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.

Type	Name	Description
unsigned int	count	Size of CAM in number of 64-bit words (1-32)
sync_t	sync	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to be raised upon completion

3.6.3.189 cls_cam_lookup_add_ptr32

Prototype:

```
void cls_cam_lookup_add_ptr32(__xread void* data_out, __xwrite void* data_in, volatile
void __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch Content Addressable Memory lookup word with add.

Use 32-bit values from transfer register and compare with 32-bit input data. If no match was found insert input data into first zero entry in the CAM.

Table 3.214. cls_cam_lookup_add_ptr32 parameters

Type	Name	Description
__xread void*	data_out	Data looked up
__xwrite void*	data_in	Data to lookup
volatile void __cls*	address	Local scratch CAM address to lookup
unsigned int	count	Size of CAM in number of 64-bit words (1-16)
sync_t	sync	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to be raised upon completion

3.6.3.190 cls_cam_lookup_add_ptr40

Prototype:

```
void cls_cam_lookup_add_ptr40(__xread void* data_out, __xwrite void* data_in, volatile
void __addr40 __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

40-bit Local scratch Content Addressable Memory lookup word with add.

Use 32-bit values from transfer register and compare with 32-bit input data. If no match was found insert input data into first zero entry in the CAM.

Table 3.215. `cls_cam_lookup_add_ptr40` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit CAM address to lookup where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>count</code>	Size of CAM in number of 64-bit words (1-16)
<code>sync_t</code>	<code>sync</code>	Type of synchronization (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised upon completion

3.6.3.191 `cls_cam_lookup24_ptr32`

Prototype:

```
void cls_cam_lookup24_ptr32(__xread void* data_out, __xwrite void* data_in, volatile void __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch Content Addressable Memory lookup 24-bit.

Use 32-bit values from transfer register and compare lower 24-bits with inputdata.

Table 3.216. `cls_cam_lookup24_ptr32` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __cls*</code>	<code>address</code>	Local scratch CAM address to lookup
<code>unsigned int</code>	<code>count</code>	Size of CAM in number of 64-bit words (1-32)
<code>sync_t</code>	<code>sync</code>	Type of synchronization (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised upon completion

3.6.3.192 `cls_cam_lookup24_ptr40`

Prototype:

```
void cls_cam_lookup24_ptr40(__xread void* data_out, __xwrite void* data_in, volatile void __addr40 __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

40-bit Local scratch Content Addressable Memory lookup 24-bit.

Use 32-bit values from transfer register and compare lower 24-bits with inputdata.

Table 3.217. `cls_cam_lookup24_ptr40` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit CAM address to lookup where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>count</code>	Size of CAM in number of 64-bit words (1-32)
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised upon completion

3.6.3.193 `cls_cam_lookup24_add_ptr32`

Prototype:

```
void cls_cam_lookup24_add_ptr32(__xread void* data_out, __xwrite void* data_in, volatile void __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch Content Addressable Memory lookup 24-bits with add.

Use 32-bit values from transfer register and compare lower 24-bits with input data. If no match was found insert input data (full 32-bit) into first zero entry in the CAM.

Table 3.218. `cls_cam_lookup24_add_ptr32` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __cls*</code>	<code>address</code>	Local scratch CAM address to lookup
<code>unsigned int</code>	<code>count</code>	Size of CAM in number of 64-bit words (1-16)
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised upon completion

3.6.3.194 `cls_cam_lookup24_add_ptr40`

Prototype:

```
void cls_cam_lookup24_add_ptr40(__xread void* data_out, __xwrite void* data_in, volatile void __addr40 __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

40-bit Local scratch Content Addressable Memory lookup 24-bits with add.

Use 32-bit values from transfer register and compare lower 24-bits with input data. If no match was found insert input data (full 32-bit) into first zero entry in the CAM.

Table 3.219. `cls_cam_lookup24_add_ptr40` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit CAM address to lookup where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>count</code>	Size of CAM in number of 64-bit words (1-16)
<code>sync_t</code>	<code>sync</code>	Type of synchronization (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised upon completion

3.6.3.195 `cls_cam_lookup24_add_inc_ptr32`

Prototype:

```
void cls_cam_lookup24_add_inc_ptr32(__xread void* data_out, __xwrite void* data_in,
volatile void __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch Content Addressable Memory lookup 24-bits with add or increment usage.

Use 32-bit values from transfer register and compare lower 24-bits with input data. If a match was found, increment the counter in the top 8-bits of the matching entry with one (without saturation). If no match was found, insert input data (full 32-bit) into first zero entry in the CAM.

Table 3.220. `cls_cam_lookup24_add_inc_ptr32` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __cls*</code>	<code>address</code>	Local scratch CAM address to lookup
<code>unsigned int</code>	<code>count</code>	Size of CAM in number of 64-bit words + 16 (1-16 plus 16)
<code>sync_t</code>	<code>sync</code>	Type of synchronization (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised upon completion

3.6.3.196 `cls_cam_lookup24_add_inc_ptr40`

Prototype:

```
void cls_cam_lookup24_add_inc_ptr40(__xread void* data_out, __xwrite void* data_in,
volatile void __addr40 __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

40-bit Local scratch Content Addressable Memory lookup 24-bits with add or increment usage.

Use 32-bit values from transfer register and compare lower 24-bits with input data. If a match was found, increment the counter in the top 8-bits of the matching entry with one (without saturation). If no match was found, insert input data (full 32-bit) into first zero entry in the CAM.

Table 3.221. `cls_cam_lookup24_add_inc_ptr40` parameters

Type	Name	Description
__xread void*	data_out	Data looked up
__xwrite void*	data_in	Data to lookup
volatile void __addr40 __cls*	address	40 bit CAM address to lookup where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
unsigned int	count	Size of CAM in number of 64-bit words + 16 (1-16 plus 16)
sync_t	sync	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to be raised upon completion

3.6.3.197 `cls_cam_lookup24_add_lock_ptr32`

Prototype:

```
void cls_cam_lookup24_add_lock_ptr32(__xread void* data_out, __xwrite void* data_in,
volatile void __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch Content Addressable Memory lookup 24-bits with add lock.

Use 32-bit values from transfer register and compare lower 24-bits with input data. If match is found then bit[31] of the matching memory data location is set indicating that the entry is now locked. The pre-modified memory data top byte is returned in the push data[15:8] so that the software can learn if location was already locked. If match is not found and there is room for the entry then new entry is added with data[31] set indicating that the entry is now locked. If match is not found and there is no free entry then no memory update is performed and (0x000000ff) is returned.

Table 3.222. `cls_cam_lookup24_add_lock_ptr32` parameters

Type	Name	Description
__xread void*	data_out	Data looked up
__xwrite void*	data_in	Data to lookup
volatile void __cls*	address	Local scratch CAM address to lookup

Type	Name	Description
unsigned int	<i>count</i>	Size of CAM in number of 64-bit words (1-16)
sync_t	<i>sync</i>	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to be raised upon completion

3.6.3.198 **cls_cam_lookup24_add_lock_ptr40**

Prototype:

```
void cls_cam_lookup24_add_lock_ptr40(__xread void* data_out, __xwrite void* data_in,
volatile void __addr40 __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

40-bit Local scratch Content Addressable Memory lookup 24-bits with add lock.

Use 32-bit values from transfer register and compare lower 24-bits with input data. If match is found then bit[31] of the matching memory data location is set indicating that the entry is now locked. The pre-modified memory data top byte is returned in the push data[15:8] so that the software can learn if location was already locked. If match is not found and there is room for the entry then new entry is added with data[31] set indicating that the entry is now locked. If match is not found and there is no free entry then no memory update is performed and (0x000000ff) is returned.

Table 3.223. *cls_cam_lookup24_add_lock_ptr40* parameters

Type	Name	Description
__xread void*	<i>data_out</i>	Data looked up
__xwrite void*	<i>data_in</i>	Data to lookup
volatile void __addr40 __cls*	<i>address</i>	40 bit CAM address to lookup where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
unsigned int	<i>count</i>	Size of CAM in number of 64-bit words (1-16)
sync_t	<i>sync</i>	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to be raised upon completion

3.6.3.199 **cls_cam_lookup24_add_extend_ptr32**

Prototype:

```
void cls_cam_lookup24_add_extend_ptr32(__xread void* data_out, __xwrite void* data_in,
volatile void __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch Content Addressable Memory lookup 24-bits with add extend.

Use 32-bit values from transfer register and compare lower 24-bits with input data. If match is found then bit[31] of the matching memory data location is set indicating that the entry is now locked. The pre-modified memory data top byte is returned in the push data[15:8] so that the software can learn if location was already locked. If match is not found and there is no free entry then set bit[31] of the first memory location used for the CAM command. It will also push back the previous value of bit[31] of the first memory location in bit[15] of the push data (which is normally zero) along with 0xff in push data[7:0].

Table 3.224. `cls_cam_lookup24_add_extend_ptr32` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __cls*</code>	<code>address</code>	Local scratch CAM address to lookup
<code>unsigned int</code>	<code>count</code>	Size of CAM in number of 64-bit words (1-16)
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised upon completion

3.6.3.200 `cls_cam_lookup24_add_extend_ptr40`

Prototype:

```
void cls_cam_lookup24_add_extend_ptr40(__xread void* data_out, __xwrite void* data_in,
volatile void __addr40 __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

40-bit Local scratch Content Addressable Memory lookup 24-bits with add extend.

Use 32-bit values from transfer register and compare lower 24-bits with input data. If match is found then bit[31] of the matching memory data location is set indicating that the entry is now locked. The pre-modified memory data top byte is returned in the push data[15:8] so that the software can learn if location was already locked. If match is not found and there is no free entry then set bit[31] of the first memory location used for the CAM command. It will also push back the previous value of bit[31] of the first memory location in bit[15] of the push data (which is normally zero) along with 0xff in push data[7:0].

Table 3.225. `cls_cam_lookup24_add_extend_ptr40` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit CAM address to lookup where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>count</code>	Size of CAM in number of 64-bit words (1-16)
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised upon completion

3.6.3.201 `cls_cam_lookup16_ptr32`

Prototype:

```
void cls_cam_lookup16_ptr32(__xread void* data_out, __xwrite void* data_in, volatile void
__cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch Content Addressable Memory lookup 16-bits.

Fetch 16-bit values from transfer register and compare with lower 16-bits of input data.

Table 3.226. `cls_cam_lookup16_ptr32` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __cls*</code>	<code>address</code>	Local scratch CAM address to lookup
<code>unsigned int</code>	<code>count</code>	Size of CAM in number of 64-bit words (1-32)
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised upon completion

3.6.3.202 `cls_cam_lookup16_ptr40`

Prototype:

```
void cls_cam_lookup16_ptr40(__xread void* data_out, __xwrite void* data_in, volatile void
__addr40 __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

40-bit Local scratch Content Addressable Memory lookup 16-bits.

Fetch 16-bit values from transfer register and compare with lower 16-bits of input data.

Table 3.227. `cls_cam_lookup16_ptr40` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit CAM address to lookup where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>count</code>	Size of CAM in number of 64-bit words (1-32)
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)

Type	Name	Description
SIGNAL*	<i>sig_ptr</i>	Signal to be raised upon completion

3.6.3.203 `cls_cam_lookup16_add_ptr32`

Prototype:

```
void cls_cam_lookup16_add_ptr32(__xread void* data_out, __xwrite void* data_in, volatile
void __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch Content Addressable Memory lookup 16-bits with add.

Use 16-bit values from transfer register and compare with lower 16-bits of input data. If no match was found, insert 16-bit input data into first zero entry in the CAM.

Table 3.228. `cls_cam_lookup16_add_ptr32` parameters

Type	Name	Description
<code>__xread void*</code>	<i>data_out</i>	Data looked up
<code>__xwrite void*</code>	<i>data_in</i>	Data to lookup
volatile void <code>__cls*</code>	<i>address</i>	Local scratch CAM address to lookup
unsigned int	<i>count</i>	Size of CAM in number of 64-bit words (1-16)
<code>sync_t</code>	<i>sync</i>	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to be raised upon completion

3.6.3.204 `cls_cam_lookup16_add_ptr40`

Prototype:

```
void cls_cam_lookup16_add_ptr40(__xread void* data_out, __xwrite void* data_in, volatile
void __addr40 __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

40-bit Local scratch Content Addressable Memory lookup 16-bits with add.

Use 16-bit values from transfer register and compare with lower 16-bits of input data. If no match was found, insert 16-bit input data into first zero entry in the CAM.

Table 3.229. `cls_cam_lookup16_add_ptr40` parameters

Type	Name	Description
<code>__xread void*</code>	<i>data_out</i>	Data looked up
<code>__xwrite void*</code>	<i>data_in</i>	Data to lookup

Type	Name	Description
volatile void __addr40 __cls*	address	40 bit CAM address to lookup where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
unsigned int	count	Size of CAM in number of 64-bit words (1-16)
sync_t	sync	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to be raised upon completion

3.6.3.205 cls_cam_lookup8_ptr32

Prototype:

```
void cls_cam_lookup8_ptr32(__xread void* data_out, __xwrite void* data_in, volatile void __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch Content Addressable Memory lookup 8-bits.

Use 8-bit values from transfer register and compare with lower 8-bits of input data.

Table 3.230. cls_cam_lookup8_ptr32 parameters

Type	Name	Description
__xread void*	data_out	Data looked up
__xwrite void*	data_in	Data to lookup
volatile void __cls*	address	Local scratch CAM address to lookup
unsigned int	count	Size of CAM in number of 64-bit words (1-32)
sync_t	sync	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to be raised upon completion

3.6.3.206 cls_cam_lookup8_ptr40

Prototype:

```
void cls_cam_lookup8_ptr40(__xread void* data_out, __xwrite void* data_in, volatile void __addr40 __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

40-bit Local scratch Content Addressable Memory lookup 8-bits.

Use 8-bit values from transfer register and compare with lower 8-bits of input data.

Table 3.231. `cls_cam_lookup8_ptr40` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit CAM address to lookup where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>count</code>	Size of CAM in number of 64-bit words (1-32)
<code>sync_t</code>	<code>sync</code>	Type of synchronization (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised upon completion

3.6.3.207 `cls_cam_lookup8_add_ptr32`

Prototype:

```
void cls_cam_lookup8_add_ptr32(__xread void* data_out, __xwrite void* data_in, volatile void __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch Content Addressable Memory lookup 8-bits with add.

Use 8-bit values from transfer register and compare with lower 8-bits of input data. If no match was found, insert 8-bit input data into first zero entry in the CAM.

Table 3.232. `cls_cam_lookup8_add_ptr32` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __cls*</code>	<code>address</code>	Local scratch CAM address to lookup
<code>unsigned int</code>	<code>count</code>	Size of CAM in number of 64-bit words (1-16)
<code>sync_t</code>	<code>sync</code>	Type of synchronization (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised upon completion

3.6.3.208 `cls_cam_lookup8_add_ptr40`

Prototype:

```
void cls_cam_lookup8_add_ptr40(__xread void* data_out, __xwrite void* data_in, volatile void __addr40 __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

40-bit Local scratch Content Addressable Memory lookup 8-bits with add.

Use 8-bit values from transfer register and compare with lower 8-bits of input data. If no match was found, insert 8-bit input data into first zero entry in the CAM.

Table 3.233. `cls_cam_lookup8_add_ptr40` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit CAM address to lookup where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>count</code>	Size of CAM in number of 64-bit words (1-16)
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised upon completion

3.6.3.209 `cls_tcam_lookup_ptr32`

Prototype:

```
void cls_tcam_lookup_ptr32(__xread void* data_out, __xwrite void* data_in, volatile void
__cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch TCAM lookup word.

Compare whole 32-bit TCAM word with 32-bit input data.

Table 3.234. `cls_tcam_lookup_ptr32` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __cls*</code>	<code>address</code>	Local scratch TCAM address to lookup
<code>unsigned int</code>	<code>count</code>	Number of 64-bit words
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised upon completion

3.6.3.210 `cls_tcam_lookup_ptr40`

Prototype:

```
void cls_tcam_lookup_ptr40(__xread void* data_out, __xwrite void* data_in, volatile void
__addr40 __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

40-bit Local scratch TCAM lookup word.

Compare whole 32-bit TCAM word with 32-bit input data.

Table 3.235. `cls_tcam_lookup_ptr40` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit TCAM address to lookup where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>count</code>	Number of 64-bit words
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised upon completion

3.6.3.211 `cls_tcam_lookup24_ptr32`

Prototype:

```
void cls_tcam_lookup24_ptr32(__xread void* data_out, __xwrite void* data_in, volatile
void __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch TCAM lookup 24-bits.

Compare lower 24-bits of TCAM word with lower 24-bits of input data. The upper 8 bits of TCAM match word and TCAM mask are ignored.

Table 3.236. `cls_tcam_lookup24_ptr32` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __cls*</code>	<code>address</code>	Local scratch TCAM address to lookup
<code>unsigned int</code>	<code>count</code>	Number of 64-bit words
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised upon completion

3.6.3.212 `cls_tcam_lookup24_ptr40`

Prototype:

```
void cls_tcam_lookup24_ptr40(__xread void* data_out, __xwrite void* data_in, volatile
void __addr40 __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

40-bit Local scratch TCAM lookup 24-bits.

Compare lower 24-bits of TCAM word with lower 24-bits of input data. The upper 8 bits of TCAM match word and TCAM mask are ignored.

Table 3.237. `cls_tcam_lookup24_ptr40` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit TCAM address to lookup where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>count</code>	Number of 64-bit words
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised upon completion

3.6.3.213 `cls_tcam_lookup16_ptr32`

Prototype:

```
void cls_tcam_lookup16_ptr32(__xread void* data_out, __xwrite void* data_in, volatile void __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch TCAM lookup 16-bits.

Split TCAM match word into two 16-bit words and compare each with lower 16-bits of input data.

Table 3.238. `cls_tcam_lookup16_ptr32` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __cls*</code>	<code>address</code>	Local scratch TCAM address to lookup
<code>unsigned int</code>	<code>count</code>	Number of 64-bit words
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised upon completion

3.6.3.214 `cls_tcam_lookup16_ptr40`

Prototype:

```
void cls_tcam_lookup16_ptr40(__xread void* data_out, __xwrite void* data_in, volatile
void __addr40 __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

40-bit Local scratch TCAM lookup 16-bits.

Split TCAM match word into two 16-bit words and compare each with lower 16-bits of input data.

Table 3.239. `cls_tcam_lookup16_ptr40` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit TCAM address to lookup where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>count</code>	Number of 64-bit words
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised upon completion

3.6.3.215 `cls_tcam_lookup8_ptr32`

Prototype:

```
void cls_tcam_lookup8_ptr32(__xread void* data_out, __xwrite void* data_in, volatile void
__cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Local scratch TCAM lookup 8-bits.

Split TCAM match word into four 8-bit words and compare each with lower 8-bits of input data.

Table 3.240. `cls_tcam_lookup8_ptr32` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __cls*</code>	<code>address</code>	Local scratch TCAM address to lookup
<code>unsigned int</code>	<code>count</code>	Number of 64-bit words
<code>sync_t</code>	<code>sync</code>	Type of synchronization (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised upon completion

3.6.3.216 `cls_tcam_lookup8_ptr40`

Prototype:

```
void cls_tcam_lookup8_ptr40(__xread void* data_out, __xwrite void* data_in, volatile void
__addr40 __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

40-bit Local scratch TCAM lookup 8-bits.

Split TCAM match word into four 8-bit words and compare each with lower 8-bits of input data.

Table 3.241. `cls_tcam_lookup8_ptr40` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data_out</code>	Data looked up
<code>__xwrite void*</code>	<code>data_in</code>	Data to lookup
<code>volatile void __addr40</code>	<code>address</code>	40 bit TCAM address to lookup where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>count</code>	Number of 64-bit words
<code>sync_t</code>	<code>sync</code>	Type of synchronization (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised upon completion

3.7 Cluster Local Scratch Ring Intrinsics

This section discusses Cluster Local Scratch Ring functions.

3.7.1 CLS Ring Enumerations

3.7.1.1 `cls_state_t`

This enumeration defines the state values that can be tested with the `cls_state_test()` intrinsic.

Table 3.242. enum `cls_state_t`

Name	Description
<code>cls_state_ring0_status</code>	Indicates status of CLS Ring 0.
<code>cls_state_ring1_status</code>	Indicates status of CLS Ring 1.
<code>cls_state_ring2_status</code>	Indicates status of CLS Ring 2.
<code>cls_state_ring3_status</code>	Indicates status of CLS Ring 3.

Name	Description
cls_state_ring4_status	Indicates status of CLS Ring 4.
cls_state_ring5_status	Indicates status of CLS Ring 5.
cls_state_ring6_status	Indicates status of CLS Ring 6.
cls_state_ring7_status	Indicates status of CLS Ring 7.
cls_state_ring8_status	Indicates status of CLS Ring 8.
cls_state_ring9_status	Indicates status of CLS Ring 9.
cls_state_ring10_status	Indicates status of CLS Ring 10.
cls_state_ring11_status	Indicates status of CLS Ring 11.
cls_state_ring12_status	Indicates status of CLS Ring 12.
cls_state_ring13_status	Indicates status of CLS Ring 13.
cls_state_ring14_status	Indicates status of CLS Ring 14.
cls_state_ring15_status	Indicates status of CLS Ring 15.

3.7.1.2 CLS_RING_SIZE

Ring word size.

Table 3.243. enum CLS_RING_SIZE

Name	Description
CLS_RING_SIZE_32	32 words. base address should be aligned to 128 byte boundary, full is >=24 words.
CLS_RING_SIZE_64	64 words. base address should be aligned to 256 byte boundary, full is >=48 words.
CLS_RING_SIZE_128	128 words. base address should be aligned to 512 byte boundary, full is >=96 words.
CLS_RING_SIZE_256	256 words. base address should be aligned to 1024 byte boundary, full is >=192 words.
CLS_RING_SIZE_512	512 words. base address should be aligned to 2048 byte boundary, full is >=384 words.
CLS_RING_SIZE_1024	1024 words. base address should be aligned to 4096 byte boundary, full is >=768 words.

3.7.1.3 CLS_RING_EVENT_REPORTS

Ring events to generate system events on the event bus.

The reported event has the source set to the ring number and the event specified to the relevant event type (0=>not empty, 1=>not full, 8=>underflow, 9=>overflow)

Table 3.244. enum CLS_RING_EVENT_REPORTS

Name	Description
CLS_RING_NO_EVENTS	No events generated.
CLS_RING_UNDERFLOW_EVENT	Generate event on event bus if ring underflows. If asserted then a system event is generated whenever a command is handled which attempts to underflow the ring (remove elements when the ring is empty). This is generated even though the ring operation is aborted.
CLS_RING_OVERFLOW_EVENT	Generate event on event bus if ring overflows. If asserted then a system event is generated whenever a command is handled which attempts to overflow the ring; this is generated even though the ring operation is aborted, to indicate to the system that data has probably been lost.
CLS_RING_NOT_EMPTY_EVENT	Generate event on event bus if ring become not empty. If asserted then a system event is generated whenever the ring goes from empty to not empty.
CLS_RING_NOT_FULL_EVENT	Generate event on event bus if ring become not full. If asserted then a system event is generated whenever the ring goes from full (i.e. >3/4 entries used) to not full (<3/4 entries used).

3.7.2 CLS Ring Typedefs

3.7.2.1 CLS_RING_SIZE

Ring word size.

Table 3.245. typedef CLS_RING_SIZE

Type	Definition
CLS_RING_SIZE	enum CLS_RING_SIZE

3.7.2.2 CLS_RING_EVENT_REPORTS

Ring events to generate system events on the event bus.

The reported event has the source set to the ring number and the event specified to the relevant event type (0=>not empty, 1=>not full, 8=>underflow, 9=>overflow)

Table 3.246. `typedef CLS_RING_EVENT_REPORTS`

Type	Definition
<code>CLS_RING_EVENT_REPORTS</code>	enum <code>CLS_RING_EVENT_REPORTS</code>

3.7.3 CLS Ring Functions

3.7.3.1 `cls_state_test`

Prototype:

```
int cls_state_test(cls_state_t state)
```

Description:

Tests the value of the specified CLS state name.

This function tests the value of the specified state name and returns a 1 if the state is set or 0 if clear. The argument state must be a constant literal as required by the microcode assembler; otherwise, the compiler generates a runtime check, if possible, with loss of performance.

Table 3.247. `cls_state_test` parameters

Type	Name	Description
<code>cls_state_t</code>	<code>state</code>	State to test

3.7.3.2 `cls_ring_put_ptr32`

Prototype:

```
void cls_ring_put_ptr32(__xwrite void* data, volatile void __addr32 __cls* address,  
unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Put data on Cluster Local Scratch memory ring.

Write 32-bit words to tail of ring and update tail pointer. If there are not enough empty slots in the ring, no words are added and the tail pointer is not updated. This will raise an overflow event if the ring is configured to deliver one.

Table 3.248. `cls_ring_put_ptr32` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Data to put onto ring

Type	Name	Description
volatile void __addr32 __cls*	address	Ring address to put data into where the ring number is specified in address[5:2]
unsigned int	count	Number of 32-bit words to put into ring
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

3.7.3.3 cls_ring_put_ptr40

Prototype:

```
void cls_ring_put_ptr40(__xwrite void* data, volatile void __addr40 __cls* address,
unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Put data on Cluster Local Scratch memory ring.

Write 40-bit words to tail of ring and update tail pointer. If there are not enough empty slots in the ring, no words are added and the tail pointer is not updated. This will raise an overflow event if the ring is configured to deliver one.

Table 3.249. cls_ring_put_ptr40 parameters

Type	Name	Description
__xwrite void*	data	Data to put onto ring
volatile void __addr40 __cls*	address	40 bit ring address to put data into where upper 6 bits indicate the island. The ring number is specified in address[5:2] See 6xxx databook for recommended addressing mode.
unsigned int	count	Number of 32-bit words to put into ring
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

3.7.3.4 cls_ring_journal_ptr32

Prototype:

```
void cls_ring_journal_ptr32(__xwrite void* data, volatile void __addr32 __cls* address,
unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Add data on Cluster Local Scratch memory ring.

Journal adds data into specified ring in Cluster Local Scratch memory. A ring journal command ignores any full indication. The consumer of a journal will start at the tail pointer and read backwards from there through time, up to the count of the journal.

Table 3.250. `cls_ring_journal_ptr32` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Data to add onto ring
<code>volatile void __addr32 __cls*</code>	<code>address</code>	Ring address to add data into where the ring number is specified in address[5:2]
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to add into ring
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.7.3.5 `cls_ring_journal_ptr40`

Prototype:

```
void cls_ring_journal_ptr40(__xwrite void* data, volatile void __addr40 __cls* address,
                           unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Add data on Cluster Local Scratch memory ring.

Journal adds data into specified ring in Cluster Local Scratch memory. A ring journal command ignores any full indication. The consumer of a journal will start at the tail pointer and read backwards from there through time, up to the count of the journal.

Table 3.251. `cls_ring_journal_ptr40` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Data to add onto ring
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit ring address to put data into where upper 6 bits indicate the island. The ring number is specified in address[5:2] See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to add into ring
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.7.3.6 `cls_ring_get_ptr32`

Prototype:

```
void cls_ring_get_ptr32(__xread void* data, volatile void __addr32 __cls* address, unsigned
int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read data from Cluster Local Scratch memory ring and put in transfer register.

Read 32-bit words from head of ring and update head pointer. If there are not enough entries in the ring, no valid data is returned and the head pointer is not updated. This will raise an underflow event if the ring is configured to deliver one.

Table 3.252. `cls_ring_get_ptr32` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Data read from ring
<code>volatile void __addr32 __cls*</code>	<code>address</code>	Ring address to read data from where the ring number is specified in address[5:2]
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to read
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.7.3.7 `cls_ring_get_ptr40`

Prototype:

```
void cls_ring_get_ptr40(__xread void* data, volatile void __addr40 __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read data from Cluster Local Scratch memory ring and put in transfer register.

Read 32-bit words from head of ring and update head pointer. If there are not enough entries in the ring, no valid data is returned and the head pointer is not updated. This will raise an underflow event if the ring is configured to deliver one.

Table 3.253. `cls_ring_get_ptr40` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Data read from ring
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit ring address to read data from where upper 6 bits indicate the island. The ring number is specified in address[5:2] See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to read
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.7.3.8 cls_ring_pop_ptr32

Prototype:

```
void cls_ring_pop_ptr32(__xread void* data, volatile void __addr32 __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Remove data from Cluster Local Scratch memory ring and put in transfer register.

Read 32-bit words from tail of ring and update tail pointer. If there are not enough entries in the ring, no valid data is returned and the tail pointer is not updated. This will raise an underflow event if the ring is configured to deliver one.



Note

A single pop operation will return the words in the order they were added. If the ring contains the words (A, B, C, D, E), then a single pop operation of 3 words will return the values (C, D, E) - in that order.

Table 3.254. cls_ring_pop_ptr32 parameters

Type	Name	Description
__xread void*	data	Data read from ring
volatile void __addr32 __cls*	address	Ring address to read data from where the ring number is specified in address[5:2]
unsigned int	count	Number of 32-bit words to read
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

3.7.3.9 cls_ring_pop_ptr40

Prototype:

```
void cls_ring_pop_ptr40(__xread void* data, volatile void __addr40 __cls* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Remove data from Cluster Local Scratch memory ring and put in transfer register.

Read 32-bit words from tail of ring and update tail pointer. If there are not enough entries in the ring, no valid data is returned and the tail pointer is not updated. This will raise an underflow event if the ring is configured to deliver one.



Note

A single pop operation will return the words in the order they were added. If the ring contains the words (A, B, C, D, E), then a single pop operation of 3 words will return the values (C, D, E) - in that order.

Table 3.255. `cls_ring_pop_ptr40` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Data read from ring
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit ring address to read data from where upper 6 bits indicate the island. The ring number is specified in address[5:2] See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to read
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.7.3.10 `cls_ring_get_safe_ptr32`

Prototype:

```
void cls_ring_get_safe_ptr32(__xread void* data, volatile void __addr32 __cls* address,
unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read data from Cluster Local Scratch memory ring in a safe way and put in transfer register.

The ring get safe attempts to remove the count of 32-bit words from the ring. It will permit any number of entries to be removed up to the desired length. If there are fewer entries on the ring, then the ring will be emptied, and zero will be returned for each entry that was not available from the ring.

Table 3.256. `cls_ring_get_safe_ptr32` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Data read from ring
<code>volatile void __addr32 __cls*</code>	<code>address</code>	Ring address to read data from where the ring number is specified in address[5:2]
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to read
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.7.3.11 `cls_ring_get_safe_ptr40`

Prototype:

```
void cls_ring_get_safe_ptr40(__xread void* data, volatile void __addr40 __cls* address,
unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read data from Cluster Local Scratch memory ring in a safe way and put in transfer register.

The ring get safe attempts to remove the count of 32-bit words from the ring. It will permit any number of entries to be removed up to the desired length. If there are fewer entries on the ring, then the ring will be emptied, and zero will be returned for each entry that was not available from the ring.

Table 3.257. `cls_ring_get_safe_ptr40` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Data read from ring
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit ring address to read data from where upper 6 bits indicate the island. The ring number is specified in address[5:2] See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to read
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.7.3.12 `cls_ring_pop_safe_ptr32`

Prototype:

```
void cls_ring_pop_safe_ptr32(__xread void* data, volatile void __addr32 __cls* address,
unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Remove data from Cluster Local Scratch memory ring in a safe way and put in transfer register.

The ring pop attempts to remove the count of 32-bit words from the ring. It will permit any number of entries to be removed up to the desired length. If there are fewer entries on the ring, then the ring will be emptied, and zero will be returned for each entry that was not available from the ring.



Note

A single pop operation will return the words in the order they were added. If the ring contains the words (A, B, C, D, E), then two pop operations of 3 words will return the values (C, D, E) and (A, B, 0) - in that order.

Table 3.258. `cls_ring_pop_safe_ptr32` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Data read from ring

Type	Name	Description
volatile void __addr32 __cls*	address	Ring address to read data from where the ring number is specified in address[5:2]
unsigned int	count	Number of 32-bit words to read
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

3.7.3.13 cls_ring_pop_safe_ptr40

Prototype:

```
void cls_ring_pop_safe_ptr40(__xread void* data, volatile void __addr40 __cls* address,
unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Remove data from Cluster Local Scratch memory ring in a safe way and put in transfer register.

The ring pop attempts to remove the count of 32-bit words from the ring. It will permit any number of entries to be removed up to the desired length. If there are fewer entries on the ring, then the ring will be emptied, and zero will be returned for each entry that was not available from the ring.



Note

A single pop operation will return the words in the order they were added. If the ring contains the words (A, B, C, D, E), then two pop operations of 3 words will return the values (C, D, E) and (A, B, 0) - in that order.

Table 3.259. cls_ring_pop_safe_ptr40 parameters

Type	Name	Description
__xread void*	data	Data read from ring
volatile void __addr40 __cls*	address	40 bit ring address to read data from where upper 6 bits indicate the island. The ring number is specified in address[5:2] See 6xxx databook for recommended addressing mode.
unsigned int	count	Number of 32-bit words to read
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

3.7.3.14 cls_ring_init_ptr32

Prototype:

```
void cls_ring_init_ptr32(unsigned int ring_number, volatile void __addr32 __cls*
base_address, enum CLS_RING_SIZE size, unsigned int report_events, sync_t sync, SIGNAL*
sig_ptr)
```

Description:

Initialise a Cluster Local Scratch memory ring.

Setup a ring of specified ring number, size and base address in Cluster Local Scratch memory.

Table 3.260. `cls_ring_init_ptr32` parameters

Type	Name	Description
unsigned int	<i>ring_number</i>	Ring number to initialise (0 to 15)
volatile void __addr32 __cls*	<i>base_address</i>	Base address of ring which could be anywhere in CLS. The base address should also be aligned to either 128, 256, 512, 1024 or 2048 byte boundary as specified in enum CLS_RING_SIZE.
enum CLS_RING_SIZE	<i>size</i>	Size of ring (0 to 5) of type enum CLS_RING_SIZE.
unsigned int	<i>report_events</i>	List of events of enum CLS_RING_EVENT_REPORTS which should be generated
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.7.3.15 `cls_ring_init_ptr40`

Prototype:

```
void cls_ring_init_ptr40(unsigned int ring_number, volatile void __addr40 __cls*
base_address, enum CLS_RING_SIZE size, unsigned int report_events, sync_t sync, SIGNAL*
sig_ptr)
```

Description:

Initialise a Cluster Local Scratch memory ring.

Setup a ring of specified ring number, size and base address in Cluster Local Scratch memory.

The following example shows how to create a ring in cluster local scratch on i34.

```
{
    SIGNAL                                sig;
    unsigned int                           ring_index = 15;

    __addr40 __cls void *ring_address = (__addr40 __cls void *)
        (LoadTimeConstant("__ADDR_I34_CLS") | (ring_index << 2));
    __declspec(i34.cls, addr40, aligned(32*sizeof(unsigned int))) unsigned int ring_base[32];

    // initialise ring
    {
        cls_ring_init_ptr40(
```

```

        ring_index,
        ring_base,
        CLS_RING_SIZE_32,
        0,
        sig_done,
        &sig
    );

    __wait_for_all(&sig);
}

// write four 32-bit words to the ring
{
    __xwrite unsigned int write_data[4];

    write_data[0] = 0x11223344;
    write_data[1] = 0xaabbccdd;
    write_data[2] = 0x55667788;
    write_data[3] = 0x9900aabb;
    cls_ring_put_ptr40((void*)write_data, ring_address, 4, ctx_swap, &sig);
}

// read one 32-bit word from the tail of the ring
{
    __xread unsigned int read_data;

    // pop from tail
    cls_ring_pop_ptr40((void*)&read_data, ring_address, 1, ctx_swap, &sig);

    if (read_data != 0x9900aabb)
    {
        return 0;           // We have an error
    }
}
// read one 32-bit word from the head of the ring
{
    __xread unsigned int read_data;

    // get from head
    cls_ring_get_ptr40((void*)&read_data, ring_address, 1, ctx_swap, &sig);

    if (read_data != 0x11223344)
    {
        return 0;           // We have an error
    }
}

return 1;
}

```

Table 3.261. `cls_ring_init_ptr40` parameters

Type	Name	Description
unsigned int	<i>ring_number</i>	Ring number to initialise (0 to 15)
volatile void __addr40 __cls*	<i>base_address</i>	40 bit pointer base address of ring which could be anywhere in CLS. The base address should also be aligned to either 128, 256, 512, 1024 or 2048 byte boundary as specified in enum <code>CLS_RING_SIZE</code> . The top bits of

Type	Name	Description
		the address must be set to the island as in 6xxx databook for recommended addressing mode for higher 8 bits of address parameter.
enum CLS_RING_SIZE	<i>size</i>	Size of ring (0 to 5) of type enum CLS_RING_SIZE.
unsigned int	<i>report_events</i>	List of events of enum CLS_RING_EVENT_REPORTS which should be generated
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.7.3.16 cls_ring_write_ptr32

Prototype:

```
void cls_ring_write_ptr32(__xwrite void* data, unsigned int ring_number, unsigned int
count, unsigned int offset, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write data on Cluster Local Scratch memory ring, starting at the specified offset.

Write to a specified offset within a ring, bounded by the ring size in Cluster Local Scratch memory. This is used to add entries at an offset beyond the current tail pointer without updating the tail, and to later update the tail pointer without adding any entries. This can be used to, e.g., reorder the packets in a queue.

Table 3.262. cls_ring_write_ptr32 parameters

Type	Name	Description
__xwrite void*	<i>data</i>	Data to write to ring
unsigned int	<i>ring_number</i>	Ring number to write data to (0 to 15)
unsigned int	<i>count</i>	Number of 32-bit words to put into ring (1 to 32)
unsigned int	<i>offset</i>	Offset from base for ring to write; will be bounded by the ring size
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.7.3.17 cls_ring_write_ptr40

Prototype:

```
void cls_ring_write_ptr40(__xwrite void* data, unsigned int ring_number, unsigned int
count, unsigned int offset, unsigned int island_id, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write data on Cluster Local Scratch memory ring, starting at the specified offset.

Write to a specified offset within a ring, bounded by the ring size in Cluster Local Scratch memory specified by the island_id. This is used to add entries at an offset beyond the current tail pointer without updating the tail, and to later update the tail pointer without adding any entries. This can be used to, e.g., reorder the packets in a queue.

Table 3.263. cls_ring_write_ptr40 parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Data to write to ring
<code>unsigned int</code>	<code>ring_number</code>	Ring number to write data to (0 to 15)
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to put into ring (1 to 32)
<code>unsigned int</code>	<code>offset</code>	Offset from base for ring to write; will be bounded by the ring size
<code>unsigned int</code>	<code>island_id</code>	Island id indicating CLS locality. See 6xxx databook for Local Scratch recommended addressing mode.
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.7.3.18 cls_ring_read_ptr32

Prototype:

```
void cls_ring_read_ptr32(__xread void* data, unsigned int ring_number, unsigned int count,
                        unsigned int offset, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read data on Cluster Local Scratch memory ring, starting at the specified offset.

Read from a specified offset within a ring, bounded by the ring size in Cluster Local Scratch memory.

Table 3.264. cls_ring_read_ptr32 parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Data read from ring
<code>unsigned int</code>	<code>ring_number</code>	Ring number to read data from (0 to 15)
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to read (1 to 16)
<code>unsigned int</code>	<code>offset</code>	Offset from base for ring to read; will be bounded by the ring size
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.7.3.19 cls_ring_read_ptr40

Prototype:

```
void cls_ring_read_ptr40(__xread void* data, unsigned int ring_number, unsigned int count,
                        unsigned int offset, unsigned int island_id, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read data on Cluster Local Scratch memory ring, starting at the specified offset.

Read from a specified offset within a ring, bounded by the ring size in Cluster Local Scratch memory.

Table 3.265. `cls_ring_read_ptr40` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Data read from ring
<code>unsigned int</code>	<code>ring_number</code>	Ring number to read data from (0 to 15)
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to read (1 to 16)
<code>unsigned int</code>	<code>offset</code>	Offset from base for ring to read; will be bounded by the ring size
<code>unsigned int</code>	<code>island_id</code>	Island id indicating CLS locality. See 6xxx databook for Local Scratch recommended addressing mode.
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.7.3.20 `cls_ring_ordered_lock_ptr32`

Prototype:

```
void cls_ring_ordered_lock_ptr32(unsigned int ring_number, unsigned int sequence_number,
sync_t sync, SIGNAL* sig_ptr)
```

Description:

Lock a Cluster Local Scratch memory ring for reordering.

Permits a ring to be locked for reordering. Compare sequence number with ring head pointer on a specified ring. If match is found then push signal only, otherwise write the signal information to the ring at offset specified by sequence number.

Table 3.266. `cls_ring_ordered_lock_ptr32` parameters

Type	Name	Description
<code>unsigned int</code>	<code>ring_number</code>	Ring number to read data from (0 to 15)
<code>unsigned int</code>	<code>sequence_number</code>	Sequence number to reorder on.
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.7.3.21 `cls_ring_ordered_lock_ptr40`

Prototype:

```
void cls_ring_ordered_lock_ptr40(unsigned int ring_number, unsigned int sequence_number,
unsigned int island_id, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Lock a Cluster Local Scratch memory ring for reordering.

Permits a ring to be locked for reordering. Compare sequence number with ring head pointer on a specified ring. If match is found then push signal only, otherwise write the signal information to the ring at offset specified by sequence number.

Table 3.267. `cls_ring_ordered_lock_ptr40` parameters

Type	Name	Description
unsigned int	<i>ring_number</i>	Ring number to read data from (0 to 15)
unsigned int	<i>sequence_number</i>	Sequence number to reorder on.
unsigned int	<i>island_id</i>	Island id indicating CLS locality. See 6xxx databook for Local Scratch recommended addressing mode.
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.7.3.22 `cls_ring_ordered_unlock_ptr32`

Prototype:

```
void cls_ring_ordered_unlock_ptr32(unsigned int ring_number, unsigned int sequence_number)
```

Description:

Unlock a Cluster Local Scratch memory ring after reordering.

Unlocks a ring after reordering. Increment the head pointer on the specified ring. Read the ring at the new head pointer and overwrite with zero, if the data was non-zero then push to the signal data that was read.

Table 3.268. `cls_ring_ordered_unlock_ptr32` parameters

Type	Name	Description
unsigned int	<i>ring_number</i>	Ring number to read data from (0 to 15)
unsigned int	<i>sequence_number</i>	Sequence number

Availability:

NFP-6xxx Indirect Reference Mode

3.7.3.23 `cls_ring_ordered_unlock_ptr40`

Prototype:

```
void cls_ring_ordered_unlock_ptr40(unsigned int ring_number, unsigned int sequence_number,
unsigned int island_id)
```

Description:

Unlock a Cluster Local Scratch memory ring after reordering.

Unlocks a ring after reordering. Increment the head pointer on the specified ring. Read the ring at the new head pointer and overwrite with zero, if the data was non-zero then push to the signal data that was read.

Table 3.269. `cls_ring_ordered_unlock_ptr40` parameters

Type	Name	Description
unsigned int	<i>ring_number</i>	Ring number to read data from (0 to 15)
unsigned int	<i>sequence_number</i>	Sequence number
unsigned int	<i>island_id</i>	Island id indicating CLS locality. See 6xxx databook for Local Scratch recommended addressing mode.

Availability:

NFP-6xxx Indirect Reference Mode

3.7.3.24 `cls_ring_workq_add_thread_ptr32`

Prototype:

```
void cls_ring_workq_add_thread_ptr32(__xread void* data, unsigned int ring_number, unsigned
int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Add a thread to a Cluster Local Scratch memory ring (queue).

Add a thread to the work queue. Adding a thread to a queue that contains work will get the first work and deliver it to the thread. If there are no work on the queue, the thread is added to the ring.

Table 3.270. `cls_ring_workq_add_thread_ptr32` parameters

Type	Name	Description
<code>__xread void*</code>	<i>data</i>	Work received from ring
unsigned int	<i>ring_number</i>	Ring number to read data from (0 to 15)
unsigned int	<i>count</i>	Number of 32-bit words to read (1 to 16)
<code>sync_t</code>	<i>sync</i>	Type of synchronization to use (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<i>sig_ptr</i>	Signal to raise upon completion

3.7.3.25 `cls_ring_workq_add_thread_ptr40`

Prototype:

```
void cls_ring_workq_add_thread_ptr40(__xread void* data, unsigned int ring_number, unsigned
int count, unsigned int island_id, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Add a thread to a Cluster Local Scratch memory ring (queue).

Add a thread to the work queue. Adding a thread to a queue that contains work will get the first work and deliver it to the thread. If there are no work on the queue, the thread is added to the ring.

Table 3.271. `cls_ring_workq_add_thread_ptr40` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Work received from ring
<code>unsigned int</code>	<code>ring_number</code>	Ring number to read data from (0 to 15)
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to read (1 to 16)
<code>unsigned int</code>	<code>island_id</code>	Island id indicating CLS locality. See 6xxx databook for Local Scratch recommended addressing mode.
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.7.3.26 `cls_ring_workq_add_work_ptr32`

Prototype:

```
void cls_ring_workq_add_work_ptr32(__xwrite void* data, unsigned int ring_number, unsigned
int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Add work to a Cluster Local Scratch memory ring (queue).

Adding work to a queue that contains threads will get the first thread and deliver the work to it. If there are no threads on the ring, the work is written to the ring.

Table 3.272. `cls_ring_workq_add_work_ptr32` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Work written to ring
<code>unsigned int</code>	<code>ring_number</code>	Ring number to read data from (0 to 15)
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to read (1 to 16)
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (<code>sig_done</code> or <code>ctx_swap</code>)

Type	Name	Description
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.7.3.27 **cls_ring_workq_add_work_ptr40**

Prototype:

```
void cls_ring_workq_add_work_ptr40(__xwrite void* data, unsigned int ring_number, unsigned
int count, unsigned int island_id, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Add work to a Cluster Local Scratch memory ring (queue).

Adding work to a queue that contains threads will get the first thread and deliver the work to it. If there are no threads on the ring, the work is written to the ring.

Table 3.273. *cls_ring_workq_add_work_ptr40* parameters

Type	Name	Description
<u>xwrite</u> void*	<i>data</i>	Work written to ring
unsigned int	<i>ring_number</i>	Ring number to read data from (0 to 15)
unsigned int	<i>count</i>	Number of 32-bit words to read (1 to 16)
unsigned int	<i>island_id</i>	Island id indicating CLS locality. See 6xxx databook for Local Scratch recommended addressing mode.
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.7.3.28 **cls_ring_put_ind_ptr40**

Prototype:

```
void cls_ring_put_ind_ptr40(__xwrite void* data, volatile void __addr40 __cls* address,
unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write data to tail of ring in Cluster Local Scratch memory in indirect mode.

If ring full nothing added and pointer not updated

Table 3.274. *cls_ring_put_ind_ptr40* parameters

Type	Name	Description
<u>xwrite</u> void*	<i>data</i>	Data to write to tail of ring

Type	Name	Description
volatile void __addr40 __cls*	address	40 bit Cluster Local Scratch address See 6xxx databook for recommended addressing mode.
unsigned int	max_nn	Maximum number of 32-bit words to put into ring
generic_ind_t	ind	Indirect word
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

3.7.3.29 cls_ring_put_ind_ptr32

Prototype:

```
void cls_ring_put_ind_ptr32(__xwrite void* data, volatile void __cls* address, unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write data to tail of ring in Cluster Local Scratch memory in indirect mode.

If ring full nothing added and pointer not updated

Table 3.275. cls_ring_put_ind_ptr32 parameters

Type	Name	Description
__xwrite void*	data	Data to write to tail of ring
volatile void __cls*	address	32 bit Cluster Local Scratch address
unsigned int	max_nn	Maximum number of 32-bit words to put into ring
generic_ind_t	ind	Indirect word
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

3.7.3.30 cls_ring_journal_ind_ptr40

Prototype:

```
void cls_ring_journal_ind_ptr40(__xwrite void* data, volatile void __addr40 __cls* address, unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write data to tail of ring in Cluster Local Scratch memory in indirect mode.

Always update pointer, no concept of overflow

Table 3.276. `cls_ring_journal_ind_ptr40` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Data to write to tail of ring
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit Cluster Local Scratch address See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>max_nn</code>	Maximum number of 32-bit words to put into ring
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.7.3.31 `cls_ring_journal_ind_ptr32`

Prototype:

```
void cls_ring_journal_ind_ptr32(__xwrite void* data, volatile void __cls* address, unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write data to tail of ring in Cluster Local Scratch memory in indirect mode.

Always update pointer, no concept of overflow

Table 3.277. `cls_ring_journal_ind_ptr32` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Data to write to tail of ring
<code>volatile void __cls*</code>	<code>address</code>	32 bit Cluster Local Scratch address
<code>unsigned int</code>	<code>max_nn</code>	Maximum number of 32-bit words to put into ring
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.7.3.32 `cls_ring_get_ind_ptr40`

Prototype:

```
void cls_ring_get_ind_ptr40(__xread void* data, volatile void __addr40 __cls* address, unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read data from Cluster Local Scratch memory head of ring and put into transfer register, with indirect mode Empty then underflow event, data invalid.

Table 3.278. `cls_ring_get_ind_ptr40` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Data read from ring
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>max_nn</code>	Maximum number of 32-bit words to read
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.7.3.33 `cls_ring_get_ind_ptr32`

Prototype:

```
void cls_ring_get_ind_ptr32(__xread void* data, volatile void __cls* address, unsigned
int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read data from Cluster Local Scratch memory head of ring and put into transfer register in, with indirect mode
Empty then underflow event, data invalid.

Table 3.279. `cls_ring_get_ind_ptr32` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Data read from ring
<code>volatile void __cls*</code>	<code>address</code>	Cluster Local Scratch 32 bit address
<code>unsigned int</code>	<code>max_nn</code>	Maximum number of 32-bit words to read
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.7.3.34 `cls_ring_pop_ind_ptr40`

Prototype:

```
void cls_ring_pop_ind_ptr40(__xread void* data, volatile void __addr40 __cls* address,
unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read data from Cluster Local Scratch memory tail of ring and put into transfer register in indirect mode.

Table 3.280. `cls_ring_pop_ind_ptr40` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Data read from ring
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>max_nn</code>	Maximum number of 32-bit words to read
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.7.3.35 `cls_ring_pop_ind_ptr32`

Prototype:

```
void cls_ring_pop_ind_ptr32(__xread void* data, volatile void __cls* address, unsigned
int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read data from Cluster Local Scratch memory tail of ring and put into transfer register in indirect mode.

Table 3.281. `cls_ring_pop_ind_ptr32` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Data read from ring
<code>volatile void __cls*</code>	<code>address</code>	Cluster Local Scratch 32 bit address
<code>unsigned int</code>	<code>max_nn</code>	Maximum number of 32-bit words to read
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.7.3.36 `cls_ring_get_safe_ind_ptr40`

Prototype:

```
void cls_ring_get_safe_ind_ptr40(__xread void* data, volatile void __addr40 __cls* address,
unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read data from Cluster Local Scratch memory head of ring and put into transfer register, with indirect mode Empty then no underflow event, data will be zero.

Table 3.282. `cls_ring_get_safe_ind_ptr40` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Data read from ring
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>max_nn</code>	Maximum number of 32-bit words to read
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.7.3.37 `cls_ring_get_safe_ind_ptr32`

Prototype:

```
void cls_ring_get_safe_ind_ptr32(__xread void* data, volatile void __cls* address, unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read data from Cluster Local Scratch memory head of ring and put into transfer register in, with indirect mode Empty then no underflow event, data will be zero.

Table 3.283. `cls_ring_get_safe_ind_ptr32` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Data read from ring
<code>volatile void __cls*</code>	<code>address</code>	Cluster Local Scratch 32 bit address
<code>unsigned int</code>	<code>max_nn</code>	Maximum number of 32-bit words to read
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.7.3.38 `cls_ring_pop_safe_ind_ptr40`

Prototype:

```
void cls_ring_pop_safe_ind_ptr40(__xread void* data, volatile void __addr40 __cls* address, unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read data from Cluster Local Scratch memory tail of ring and put into transfer register, with indirect mode Empty then no underflow event, data will be zero.

Table 3.284. `cls_ring_pop_safe_ind_ptr40` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Data read from ring
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address See 6xxx databook for recommended addressing mode.
<code>unsigned int</code>	<code>max_nn</code>	Maximum number of 32-bit words to read
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.7.3.39 `cls_ring_pop_safe_ind_ptr32`

Prototype:

```
void cls_ring_pop_safe_ind_ptr32(__xread void* data, volatile void __cls* address, unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read data from Cluster Local Scratch memory tail of ring and put into transfer register, with indirect mode Empty then no underflow event, data will be zero.

Table 3.285. `cls_ring_pop_safe_ind_ptr32` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Data read from ring
<code>volatile void __cls*</code>	<code>address</code>	32 bit address
<code>unsigned int</code>	<code>max_nn</code>	Maximum number of 32-bit words to read
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.8 Cluster Local Scratch Reflect Intrinsics

This section discusses Cluster Local Scratch Reflect functions.

3.8.1 CLS Reflect Functions

3.8.1.1 `cls_reflect_write_sig_local`

Prototype:

```
void cls_reflect_write_sig_local(__xwrite void* data, unsigned int remote_ME, unsigned int remote_xfer, unsigned int count, sync_t sync, volatile SIGNAL* sig_ptr)
```

Description:

Copy local transfer registers to remote microengine.

Signal local ME (source) after transfer registers are read. This intrinsic was formerly named `cls_reflect_from_sig_src()`.



Note

`__xfer_reg_number()` can be used to get remote_xfer register number.

Table 3.286. `cls_reflect_write_sig_local` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Data to reflect
<code>unsigned int</code>	<code>remote_ME</code>	Remote FPC or ME number
<code>unsigned int</code>	<code>remote_xfer</code>	Remote xfer register reg number
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to reflect
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (<code>sig_done</code> or <code>ctx_swap</code>)
<code>volatile SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.8.1.2 `cls_reflect_write_sig_local_ptr40`

Prototype:

```
void cls_reflect_write_sig_local_ptr40(__xwrite void* data, unsigned int remote_island, unsigned int remote_ME, unsigned int remote_xfer, unsigned int count, sync_t sync, volatile SIGNAL* sig_ptr)
```

Description:

Copy local transfer registers to remote microengine.

Signal local ME (source) after transfer registers are read.



Note

`__xfer_reg_number()` can be used to get remote_xfer register number.

Table 3.287. `cls_reflect_write_sig_local_ptr40` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Data to reflect
<code>unsigned int</code>	<code>remote_island</code>	Remote island number (0 for local island)
<code>unsigned int</code>	<code>remote_ME</code>	Remote FPC or ME number
<code>unsigned int</code>	<code>remote_xfer</code>	Remote xfer register reg number
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to reflect
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>volatile SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.8.1.3 `cls_reflect_write_sig_remote`

Prototype:

```
void cls_reflect_write_sig_remote(__xwrite void* data, unsigned int remote_ME, unsigned int remote_xfer, int sig, unsigned int count)
```

Description:

Copy local transfer registers to remote microengine.

Signal remote ME (destination) after transfer registers are written. This intrinsic was formerly named `cls_reflect_from_sig_dst()`.



Note

`__xfer_reg_number()` can be used to get remote_xfer register number.

Table 3.288. `cls_reflect_write_sig_remote` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Data to reflect
<code>unsigned int</code>	<code>remote_ME</code>	Remote FPC or ME number
<code>unsigned int</code>	<code>remote_xfer</code>	Remote xfer register reg number
<code>int</code>	<code>sig</code>	Triggered signal number
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to reflect

3.8.1.4 cls_reflect_write_sig_remote_ptr40

Prototype:

```
void cls_reflect_write_sig_remote_ptr40(__xwrite void* data, unsigned int remote_island,  
unsigned int remote_ME, unsigned int remote_xfer, int sig, unsigned int count)
```

Description:

Copy local transfer registers to remote microengine.

Signal remote ME (destination) after transfer registers are written.

Note

`__xfer_reg_number()` can be used to get remote_xfer register number.

Table 3.289. `cls_reflect_write_sig_remote_ptr40` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Data to reflect
<code>unsigned int</code>	<code>remote_island</code>	Remote island number (0 for local island)
<code>unsigned int</code>	<code>remote_ME</code>	Remote FPC or ME number
<code>unsigned int</code>	<code>remote_xfer</code>	Remote xfer register reg number
<code>int</code>	<code>sig</code>	Triggered signal number
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to reflect

3.8.1.5 cls_reflect_write_sig_both

Prototype:

```
void cls_reflect_write_sig_both(__xwrite void* data, unsigned int remote_ME, unsigned  
int remote_xfer, unsigned int count, sync_t sync, volatile SIGNAL* sig_ptr)
```

Description:

Copy local transfer registers to remote microengine.

Signal local ME after transfer registers are read and remote ME after transfer registers are written. This intrinsic was formerly named `cls_reflect_from_sig_both()`.

Note

`__xfer_reg_number()` can be used to get remote_xfer register number.

Table 3.290. `cls_reflect_write_sig_both` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Data to reflect
<code>unsigned int</code>	<code>remote_ME</code>	Remote FPC or ME number
<code>unsigned int</code>	<code>remote_xfer</code>	Remote transfer register
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to reflect
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (<code>sig_done</code> or <code>ctx_swap</code>)
<code>volatile SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.8.1.6 `cls_reflect_write_sig_both_ptr40`

Prototype:

```
void cls_reflect_write_sig_both_ptr40(__xwrite void* data, unsigned int remote_island,
unsigned int remote_ME, unsigned int remote_xfer, unsigned int count, sync_t sync, volatile
SIGNAL* sig_ptr)
```

Description:

Copy local transfer registers to remote microengine.

Signal local ME after transfer registers are read and remote ME after transfer registers are written.



Note

`__xfer_reg_number()` can be used to get remote_xfer register number.

Table 3.291. `cls_reflect_write_sig_both_ptr40` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Data to reflect
<code>unsigned int</code>	<code>remote_island</code>	Remote island number (0 for local island)
<code>unsigned int</code>	<code>remote_ME</code>	Remote FPC or ME number
<code>unsigned int</code>	<code>remote_xfer</code>	Remote transfer register
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to reflect
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (<code>sig_done</code> or <code>ctx_swap</code>)
<code>volatile SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.8.1.7 `cls_reflect_read_sig_remote`

Prototype:

```
void cls_reflect_read_sig_remote(__xread void* data, unsigned int remote_ME, unsigned
int remote_xfer, int sig, unsigned int count)
```

Description:

Copy transfer registers from remote microengine.

Signal remote ME (source) after transfer registers are read. This intrinsic was formerly named `cls_reflect_to_sig_src()`.



Note

`__xfer_reg_number()` can be used to get remote_xfer register number.

Table 3.292. `cls_reflect_read_sig_remote` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Data from reflect
<code>unsigned int</code>	<code>remote_ME</code>	Remote FPC or ME number
<code>unsigned int</code>	<code>remote_xfer</code>	Remote xfer register reg number
<code>int</code>	<code>sig</code>	Triggered signal number
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to reflect

3.8.1.8 `cls_reflect_read_sig_remote_ptr40`

Prototype:

```
void cls_reflect_read_sig_remote_ptr40(__xread void* data, unsigned int remote_island,
unsigned int remote_ME, unsigned int remote_xfer, int sig, unsigned int count)
```

Description:

Copy transfer registers from remote microengine.

Signal remote ME (source) after transfer registers are read.



Note

`__xfer_reg_number()` can be used to get remote_xfer register number.

Table 3.293. `cls_reflect_read_sig_remote_ptr40` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Data from reflect
<code>unsigned int</code>	<code>remote_island</code>	Remote island number (0 for local island)
<code>unsigned int</code>	<code>remote_ME</code>	Remote FPC or ME number

Type	Name	Description
unsigned int	<i>remote_xfer</i>	Remote xfer register reg number
int	<i>sig</i>	Triggered signal number
unsigned int	<i>count</i>	Number of 32-bit words to reflect

3.8.1.9 `cls_reflect_read_sig_local`

Prototype:

```
void cls_reflect_read_sig_local(__xread void* data, unsigned int remote_ME, unsigned int
remote_xfer, unsigned int count, sync_t sync, volatile SIGNAL* sig_ptr)
```

Description:

Copy transfer registers from remote microengine.

Signal local ME (destination) after transfer registers are written. This intrinsic was formerly named `cls_reflect_to_sig_dst()`.



Note

`__xfer_reg_number()` can be used to get remote_xfer register number.

Table 3.294. `cls_reflect_read_sig_local` parameters

Type	Name	Description
<code>__xread void*</code>	<i>data</i>	Data from reflect
unsigned int	<i>remote_ME</i>	Remote FPC or ME number
unsigned int	<i>remote_xfer</i>	Remote xfer register reg number
unsigned int	<i>count</i>	Number of 32-bit words to reflect
<code>sync_t</code>	<i>sync</i>	Type of synchronization to use (<code>sig_done</code> or <code>ctx_swap</code>)
<code>volatile SIGNAL*</code>	<i>sig_ptr</i>	Signal to raise upon completion

3.8.1.10 `cls_reflect_read_sig_local_ptr40`

Prototype:

```
void cls_reflect_read_sig_local_ptr40(__xread void* data, unsigned int remote_island,
unsigned int remote_ME, unsigned int remote_xfer, unsigned int count, sync_t sync, volatile
SIGNAL* sig_ptr)
```

Description:

Copy transfer registers from remote microengine.

Signal local ME (destination) after transfer registers are written.



Note

`__xfer_reg_number()` can be used to get remote_xfer register number.

Table 3.295. `cls_reflect_read_sig_local_ptr40` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Data from reflect
<code>unsigned int</code>	<code>remote_island</code>	Remote island number (0 for local island)
<code>unsigned int</code>	<code>remote_ME</code>	Remote FPC or ME number
<code>unsigned int</code>	<code>remote_xfer</code>	Remote xfer register reg number
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to reflect
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>volatile SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.8.1.11 `cls_reflect_read_sig_both`

Prototype:

```
void cls_reflect_read_sig_both(__xread void* data, unsigned int remote_ME, unsigned int
remote_xfer, unsigned int count, sync_t sync, volatile SIGNAL* sig_ptr)
```

Description:

Copy transfer registers from remote microengine.

Signal remote ME after transfer registers are read and local ME after transfer registers are written. This intrinsic was formerly named `cls_reflect_to_sig_both()`.



Note

`__xfer_reg_number()` can be used to get remote_xfer register number.

Table 3.296. `cls_reflect_read_sig_both` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Data from reflect
<code>unsigned int</code>	<code>remote_ME</code>	Remote FPC or ME number
<code>unsigned int</code>	<code>remote_xfer</code>	Remote xfer register number
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to reflect
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)

Type	Name	Description
volatile SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.8.1.12 cls_reflect_read_sig_both_ptr40

Prototype:

```
void cls_reflect_read_sig_both_ptr40(__xread void* data, unsigned int remote_island,
unsigned int remote_ME, unsigned int remote_xfer, unsigned int count, sync_t sync, volatile
SIGNAL* sig_ptr)
```

Description:

Copy transfer registers from remote microengine.

Signal remote ME after transfer registers are read and local ME after transfer registers are written.



Note

`__xfer_reg_number()` can be used to get remote_xfer register number.

Table 3.297. cls_reflect_read_sig_both_ptr40 parameters

Type	Name	Description
<code>__xread void*</code>	<i>data</i>	Data from reflect
unsigned int	<i>remote_island</i>	Remote island number (0 for local island)
unsigned int	<i>remote_ME</i>	Remote FPC or ME number
unsigned int	<i>remote_xfer</i>	Remote xfer register number
unsigned int	<i>count</i>	Number of 32-bit words to reflect
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
volatile SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.8.1.13 cls_reflect_write_sig_local_ind_ptr40

Prototype:

```
void cls_reflect_write_sig_local_ind_ptr40(__xwrite void* data, unsigned int remote_island,
unsigned int remote_ME, unsigned int remote_xfer_reg_number, unsigned int max_nn,
generic_ind_t ind, sync_t sync, volatile SIGNAL* sig_ptr)
```

Description:

Copy local transfer registers to remote microengine.

Signal local ME (source) after transfer registers are read in indirect mode, 40 bit address mode



Note

`__xfer_reg_number()` can be used to get remote_xfer register number.

Table 3.298. `cls_reflect_write_sig_local_ind_ptr40` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Data to reflect
<code>unsigned int</code>	<code>remote_island</code>	Remote island number (0 for local island)
<code>unsigned int</code>	<code>remote_ME</code>	Remote FPC or ME number
<code>unsigned int</code>	<code>remote_xfer_reg_number</code>	Remote xfer register reg number
<code>unsigned int</code>	<code>max_nn</code>	Max Number of 32-bit words to reflect
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>volatile SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.8.1.14 `cls_reflect_write_sig_local_ind`

Prototype:

```
void cls_reflect_write_sig_local_ind(__xwrite void* data, unsigned int remote_ME, unsigned int remote_xfer_reg_number, unsigned int max_nn, generic_ind_t ind, sync_t sync, volatile SIGNAL* sig_ptr)
```

Description:

Copy local transfer registers to remote microengine.

Signal local ME (source) after transfer registers are read in indirect mode, 32 bit address mode This intrinsic was formerly named `cls_reflect_from_sig_src_ind()`.



Note

`__xfer_reg_number()` can be used to get remote_xfer register number.

Table 3.299. `cls_reflect_write_sig_local_ind` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Data to reflect
<code>unsigned int</code>	<code>remote_ME</code>	Remote FPC or ME number
<code>unsigned int</code>	<code>remote_xfer_reg_number</code>	Remote xfer register reg number
<code>unsigned int</code>	<code>max_nn</code>	Max Number of 32-bit words to reflect
<code>generic_ind_t</code>	<code>ind</code>	Indirect word

Type	Name	Description
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
volatile SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.8.1.15 `cls_reflect_write_sig_remote_ind_ptr40`

Prototype:

```
void cls_reflect_write_sig_remote_ind_ptr40(__xwrite void* data, unsigned int
remote_island, unsigned int remote_ME, unsigned int remote_xfer_reg_number, int sig,
unsigned int max_nn, generic_ind_t ind)
```

Description:

Copy local transfer registers to remote microengine.

Signal remote ME (destination) after transfer registers are written in indirect mode, 40 bit address mode



Note

`__xfer_reg_number()` can be used to get remote_xfer register number.

Table 3.300. `cls_reflect_write_sig_remote_ind_ptr40` parameters

Type	Name	Description
<code>__xwrite void*</code>	<i>data</i>	Data to reflect
unsigned int	<i>remote_island</i>	Remote island number (0 for local island)
unsigned int	<i>remote_ME</i>	Remote FPC or ME number
unsigned int	<i>remote_xfer_reg_number</i>	Remote xfer register reg number
int	<i>sig</i>	Triggered signal number
unsigned int	<i>max_nn</i>	Max number of 32-bit words to reflect
generic_ind_t	<i>ind</i>	Indirect word

3.8.1.16 `cls_reflect_write_sig_remote_ind`

Prototype:

```
void cls_reflect_write_sig_remote_ind(__xwrite void* data, unsigned int remote_ME, unsigned
int remote_xfer_reg_number, int sig, unsigned int max_nn, generic_ind_t ind)
```

Description:

Copy local transfer registers to remote microengine.

Signal remote ME (destination) after transfer registers are written in indirect mode, 32 bit address mode This intrinsic was formerly named cls_reflect_from_sig_dst_ind().



Note

`__xfer_reg_number()` can be used to get remote_xfer register number.

Table 3.301. `cls_reflect_write_sig_remote_ind` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Data to reflect
<code>unsigned int</code>	<code>remote_ME</code>	Remote FPC or ME number
<code>unsigned int</code>	<code>remote_xfer_reg_number</code>	Remote xfer register reg number
<code>int</code>	<code>sig</code>	Triggered signal number
<code>unsigned int</code>	<code>max_nn</code>	Max number of 32-bit words to reflect
<code>generic_ind_t</code>	<code>ind</code>	Indirect word

3.8.1.17 `cls_reflect_write_sig_both_ind_ptr40`

Prototype:

```
void cls_reflect_write_sig_both_ind_ptr40(__xwrite void* data, unsigned int remote_island,
                                         unsigned int remote_ME, unsigned int remote_xfer_reg_number, unsigned int max_nn,
                                         generic_ind_t ind, sync_t sync, volatile SIGNAL* sig_ptr)
```

Description:

Local Scratch reflect from signal both in indirect mode.

Copy local transfer registers to remote microengine. Signal local ME after transfer registers are read and remote ME after transfer registers are written with indirection, 40 bit address mode



Note

`__xfer_reg_number()` can be used to get remote_xfer register number.

Table 3.302. `cls_reflect_write_sig_both_ind_ptr40` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Data to reflect
<code>unsigned int</code>	<code>remote_island</code>	Remote island number (0 for local island)
<code>unsigned int</code>	<code>remote_ME</code>	Remote FPC or ME number
<code>unsigned int</code>	<code>remote_xfer_reg_number</code>	Remote xfer register reg number
<code>unsigned int</code>	<code>max_nn</code>	Max Number of 32-bit words to reflect

Type	Name	Description
generic_ind_t	<i>ind</i>	Indirect word
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
volatile SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.8.1.18 cls_reflect_write_sig_both_ind

Prototype:

```
void cls_reflect_write_sig_both_ind(__xwrite void* data, unsigned int remote_ME, unsigned int remote_xfer_reg_number, unsigned int max_nn, generic_ind_t ind, sync_t sync, volatile SIGNAL* sig_ptr)
```

Description:

Local Scratch reflect from signal both in indirect mode.

Copy local transfer registers to remote microengine. Signal local ME after transfer registers are read and remote ME after transfer registers are written with indirection, 32 bit address mode This intrinsic was formerly named cls_reflect_from_sig_both_ind().



Note

`__xfer_reg_number()` can be used to get remote_xfer register number.

Table 3.303. cls_reflect_write_sig_both_ind parameters

Type	Name	Description
<code>__xwrite void*</code>	<i>data</i>	Data to reflect
unsigned int	<i>remote_ME</i>	Remote FPC or ME number
unsigned int	<i>remote_xfer_reg_number</i>	Remote xfer register reg number
unsigned int	<i>max_nn</i>	Max Number of 32-bit words to reflect
generic_ind_t	<i>ind</i>	Indirect word
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
volatile SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.8.1.19 cls_reflect_read_sig_remote_ind_ptr40

Prototype:

```
void cls_reflect_read_sig_remote_ind_ptr40(__xread void* data, unsigned int remote_island, unsigned int remote_ME, unsigned int remote_xfer_reg_number, int sig, unsigned int max_nn, generic_ind_t ind)
```

Description:

Copy transfer registers from remote microengine.

Signal remote ME (source) after transfer registers are read in indirect mode, 40 bit address mode



Note

`__xfer_reg_number()` can be used to get remote_xfer register number.

Table 3.304. `cls_reflect_read_sig_remote_ind_ptr40` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Data from reflect
<code>unsigned int</code>	<code>remote_island</code>	Remote island number (0 for local island)
<code>unsigned int</code>	<code>remote_ME</code>	Remote FPC or ME number
<code>unsigned int</code>	<code>remote_xfer_reg_number</code>	Remote xfer register reg number
<code>int</code>	<code>sig</code>	Triggered signal number
<code>unsigned int</code>	<code>max_nn</code>	Max number of 32-bit words to reflect
<code>generic_ind_t</code>	<code>ind</code>	Indirect word

3.8.1.20 `cls_reflect_read_sig_remote_ind`

Prototype:

```
void cls_reflect_read_sig_remote_ind(__xread void* data, unsigned int remote_ME, unsigned int remote_xfer_reg_number, int sig, unsigned int max_nn, generic_ind_t ind)
```

Description:

Copy transfer registers from remote microengine.

Signal remote ME (source) after transfer registers are read in indirect mode, 32 bit address mode This intrinsic was formerly named `cls_reflect_to_sig_src_ind()`.



Note

`__xfer_reg_number()` can be used to get remote_xfer register number.

Table 3.305. `cls_reflect_read_sig_remote_ind` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Data from reflect
<code>unsigned int</code>	<code>remote_ME</code>	Remote FPC or ME number
<code>unsigned int</code>	<code>remote_xfer_reg_number</code>	Remote xfer register reg number

Type	Name	Description
int	<i>sig</i>	Triggered signal number
unsigned int	<i>max_nn</i>	Max number of 32-bit words to reflect
generic_ind_t	<i>ind</i>	Indirect word

3.8.1.21 `cls_reflect_read_sig_local_ind_ptr40`

Prototype:

```
void cls_reflect_read_sig_local_ind_ptr40(__xread void* data, unsigned int remote_island,
unsigned int remote_ME, unsigned int remote_xfer_reg_number, unsigned int max_nn,
generic_ind_t ind, sync_t sync, volatile SIGNAL* sig_ptr)
```

Description:

Copy transfer registers from remote microengine Signal local ME (destination) after transfer registers are written in indirect mode, 40 bit address mode.



Note

`__xfer_reg_number()` can be used to get remote_xfer register number.

Table 3.306. `cls_reflect_read_sig_local_ind_ptr40` parameters

Type	Name	Description
<code>__xread void*</code>	<i>data</i>	Data from reflect
unsigned int	<i>remote_island</i>	Remote island number (0 for local island)
unsigned int	<i>remote_ME</i>	Remote FPC or ME number
unsigned int	<i>remote_xfer_reg_number</i>	Remote xfer register reg number
unsigned int	<i>max_nn</i>	Max Number of 32-bit words to reflect
generic_ind_t	<i>ind</i>	Indirect word
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
volatile SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.8.1.22 `cls_reflect_read_sig_local_ind`

Prototype:

```
void cls_reflect_read_sig_local_ind(__xread void* data, unsigned int remote_ME, unsigned
int remote_xfer_reg_number, unsigned int max_nn, generic_ind_t ind, sync_t sync, volatile
SIGNAL* sig_ptr)
```

Description:

Copy transfer registers from remote microengine Signal local ME (destination) after transfer registers are written in indirect mode, 32 bit address mode This intrinsic was formerly named `cls_reflect_to_sig_dst_ind()`.



Note

`_xfer_reg_number()` can be used to get remote_xfer register number.

Table 3.307. `cls_reflect_read_sig_local_ind` parameters

Type	Name	Description
<code>_xread void*</code>	<code>data</code>	Data from reflect
<code>unsigned int</code>	<code>remote_ME</code>	Remote FPC or ME number
<code>unsigned int</code>	<code>remote_xfer_reg_number</code>	Remote xfer register reg number
<code>unsigned int</code>	<code>max_nn</code>	Max Number of 32-bit words to reflect
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>volatile SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.8.1.23 `cls_reflect_read_sig_both_ind_ptr40`

Prototype:

```
void cls_reflect_read_sig_both_ind_ptr40(_xread void* data, unsigned int remote_island,
                                         unsigned int remote_ME, unsigned int remote_xfer_reg_number, unsigned int max_nn,
                                         generic_ind_t ind, sync_t sync, volatile SIGNAL* sig_ptr)
```

Description:

Copy transfer registers from remote microengine.

Signal remote ME after transfer registers are read and local ME after transfer registers are written in indirect mode, 40 bit address mode



Note

`_xfer_reg_number()` can be used to get remote_xfer register number.

Table 3.308. `cls_reflect_read_sig_both_ind_ptr40` parameters

Type	Name	Description
<code>_xread void*</code>	<code>data</code>	Data from reflect
<code>unsigned int</code>	<code>remote_island</code>	Remote island number (0 for local island)
<code>unsigned int</code>	<code>remote_ME</code>	Remote FPC or ME number
<code>unsigned int</code>	<code>remote_xfer_reg_number</code>	Remote xfer register reg number

Type	Name	Description
unsigned int	<i>max_nn</i>	Max Number of 32-bit words to reflect
generic_ind_t	<i>ind</i>	Indirect word
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
volatile SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.8.1.24 cls_reflect_read_sig_both_ind

Prototype:

```
void cls_reflect_read_sig_both_ind(__xread void* data, unsigned int remote_ME, unsigned
int remote_xfer_reg_number, unsigned int max_nn, generic_ind_t ind, sync_t sync, volatile
SIGNAL* sig_ptr)
```

Description:

Copy transfer registers from remote microengine.

Signal remote ME after transfer registers are read and local ME after transfer registers are written in indirect mode, 32 bit address mode This intrinsic was formerly named cls_reflect_to_sig_both_ind().



Note

`__xfer_reg_number()` can be used to get remote_xfer register number.

Table 3.309. `cls_reflect_read_sig_both_ind` parameters

Type	Name	Description
<code>__xread void*</code>	<i>data</i>	Data from reflect
unsigned int	<i>remote_ME</i>	Remote FPC or ME number
unsigned int	<i>remote_xfer_reg_number</i>	Remote xfer register reg number
unsigned int	<i>max_nn</i>	Max Number of 32-bit words to reflect
generic_ind_t	<i>ind</i>	Indirect word
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
volatile SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.9 Cluster Target Intrinsics

This section describes the cluster target functions.

3.9.1 Cluster Target Enumerations

3.9.1.1 CLUSTER_TARGET_REGISTER_TYPE

Enum for xfer register or csr register.

Table 3.310. enum CLUSTER_TARGET_REGISTER_TYPE

Name	Description
CT_REGISTER_TYPE_XFER	xfer register type.
CT_REGISTER_TYPE_CSR	CSR register type.

3.9.1.2 CLUSTER_TARGET_ADDRESS_MODE

Enum for addressing mode.

Table 3.311. enum CLUSTER_TARGET_ADDRESS_MODE

Name	Description
CT_ADDRESS_MODE_INDEX	NN register FIFO mode is used.
CT_ADDRESS_MODE_ABSOLUTE	NN register number specified is the first to be written to.

3.9.1.3 CT_RING_SIZE

Ring word size.

Table 3.312. enum CT_RING_SIZE

Name	Description
CT_RING_SIZE_128	128 words. Base address should be aligned to 512 byte boundary,
CT_RING_SIZE_256	256 words. Base address should be aligned to 1024 byte boundary.
CT_RING_SIZE_512	512 words. Base address should be aligned to 2048 byte boundary.
CT_RING_SIZE_1024	1024 words (1K). Base address should be aligned to 4096 byte boundary.
CT_RING_SIZE_2048	2048 words (2K). Base address should be aligned to 8192 byte boundary.

Name	Description
CT_RING_SIZE_4096	4096 words (4K). Base address should be aligned to 16384 byte boundary.
CT_RING_SIZE_8192	8192 words (8K). Base address should be aligned to 32768 byte boundary.
CT_RING_SIZE_16384	16384 words (16K). Base address should be aligned to 65536 byte boundary.

3.9.1.4 CT_RING_STATUS

The type of status reported on the status bus for the ring.

Table 3.313. enum CT_RING_STATUS

Name	Description
CT_RING_EMPTY	If ring contains no valid entries.
CT_RING_FULL	If ring is 3/4 full or greater.

3.9.2 Cluster Target Unions

3.9.2.1 cluster_target_next_neighbour_write_address_format_t

Layout of 32-bit address used with the next neighbour write command.

Table 3.314. union cluster_target_next_neighbour_write_address_format_t

Type	Name	Description
unsigned int	reserved_3:2	Reserved.
unsigned int	remote_island:6	Island id of remote master.
unsigned int	reserved_2:3	Reserved.
unsigned int	master:4	Master within specified island. See NFP-6xxx Pull IDs in 6xxx databook.
unsigned int	signal_number:7	If non-zero, signal number to send to the ME on completion.
CLUSTER_TARGET_ADDRESS_MODE	address_mode:1	Address mode where 0 = NN register FIFO mode, 1 = absolute mode.
unsigned int	NN_register_number:7	Next neighbour register number.
unsigned int	reserved_1:2	Reserved.

Type	Name	Description
unsigned int	value	Accessor to entire lookup detail structure.

3.9.2.2 cluster_target_reflect_address_format_t

Layout of 32-bit address used with reflector commands.

Table 3.315. union cluster_target_reflect_address_format_t

Type	Name	Description
unsigned int	reserved_4:2	Reserved.
unsigned int	remote_island:6	Island Id of remote master.
unsigned int	reserved_3:7	Reserved.
CLUSTER_TARGET_REGISTER_TYPE	register_type:1	Register type where 0 = transfer registers, 1 = CSR registers.
unsigned int	reserved_2:2	Reserved.
unsigned int	master:4	Master within specified island. See NFP-6xxx Pull IDs in 6xxx databook.
unsigned int	address:8	Transfer Register Address or Local ME CSR Address, depending on XferCsrRegSel.
unsigned int	reserved_1:2	Reserved.
unsigned int	value	Accessor to entire lookup detail structure.

3.9.2.3 cluster_target_signal_me_address_format_t

Layout of 32-bit address used with interthread signalling commands.

Table 3.316. union cluster_target_signal_me_address_format_t

Type	Name	Description
unsigned int	reserved_3:2	Reserved.
unsigned int	remote_island:6	Island id of remote master.
unsigned int	reserved_2:11	Reserved.
unsigned int	remote_master:4	Remote master id within specified island. See NFP-6xxx Pull IDs in 6xxx databook.
unsigned int	remote_context:3	Context number (thread number) of remote thread.
unsigned int	signal_number:4	Signal number or signal reference.
unsigned int	reserved_1:2	Reserved.
unsigned int	value	Accessor to entire lookup detail structure.

3.9.2.4 cluster_target_xpb_address_format_t

Layout of 32-bit address used with XPB commands.

Table 3.317. union cluster_target_xpb_address_format_t

Type	Name	Description
unsigned int	reserved_2:1	Reserved.
unsigned int	global_xpb:1	Global steering bit. Local = 0, global = 1.
unsigned int	target_island:6	Target island. 0 if own island (local).
unsigned int	slave:2	Slave island number, for those islands that have slave XPB buses.
unsigned int	device:6	XPB device number with the island/slave island.
unsigned int	address:14	XPB register address within the XPB target.
unsigned int	reserved_1:2	Reserved.
unsigned int	value	Accessor to entire lookup detail structure.

3.9.3 Cluster Target Typedefs

3.9.3.1 CLUSTER_TARGET_REGISTER_TYPE

Enum for xfer register or csr register.

Table 3.318. typedef CLUSTER_TARGET_REGISTER_TYPE

Type	Definition
CLUSTER_TARGET_REGISTER_TYPE	enum CLUSTER_TARGET_REGISTER_TYPE

3.9.3.2 CLUSTER_TARGET_ADDRESS_MODE

Enum for addressing mode.

Table 3.319. typedef CLUSTER_TARGET_ADDRESS_MODE

Type	Definition
CLUSTER_TARGET_ADDRESS_MODE	enum CLUSTER_TARGET_ADDRESS_MODE

3.9.3.3 cluster_target_xpb_address_format_t

Layout of 32-bit address used with XPB commands.

Table 3.320. typedef cluster_target_xpb_address_format_t

Type	Definition
cluster_target_xpb_address_format_t	union cluster_target_xpb_address_format_t

3.9.3.4 cluster_target_reflect_address_format_t

Layout of 32-bit address used with reflector commands.

Table 3.321. typedef cluster_target_reflect_address_format_t

Type	Definition
cluster_target_reflect_address_format_t	union cluster_target_reflect_address_format_t

3.9.3.5 cluster_target_signal_me_address_format_t

Layout of 32-bit address used with interthread signalling commands.

Table 3.322. typedef cluster_target_signal_me_address_format_t

Type	Definition
cluster_target_signal_me_address_format_t	union cluster_target_signal_me_address_format_t

3.9.3.6 cluster_target_next_neighbour_write_address_format_t

Layout of 32-bit address used with the next neighbour write command.

Table 3.323. typedef cluster_target_next_neighbour_write_address_format_t

Type	Definition
cluster_target_next_neighbour_write_address_format_t	union cluster_target_next_neighbour_write_address_format_t

3.9.3.7 CT_RING_SIZE

Ring word size.

Table 3.324. typedef CT_RING_SIZE

Type	Definition
CT_RING_SIZE	enum CT_RING_SIZE

3.9.3.8 CT_RING_STATUS

The type of status reported on the status bus for the ring.

Table 3.325. typedef CT_RING_STATUS

Type	Definition
CT_RING_STATUS	enum CT_RING_STATUS

3.9.4 Cluster Target Functions

3.9.4.1 cluster_target_xpb_write

Prototype:

```
void cluster_target_xpb_write(__xwrite void* xfer, volatile
cluster_target_xpb_address_format_t* address, unsigned int count, sync_t sync, SIGNAL*
sig_ptr)
```

Description:

Write data to XPB target over the XPB bus.

Transactions from the ARM may be global, but from other islands must be to within their island or its slaves.



Note

The valid range for count is between 1 and 16 for Silicon revision B0 or higher and between 1 and 14 for Silicon revision less than B0. See Errata 'Misc engine Pull-buffer which stores Pull-data can erroneously overflow if the post-pull id Command FIFO is not full.'

Table 3.326. cluster_target_xpb_write parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>xfer</code>	Transfer registers containing data to write.
<code>volatile cluster_target_xpb_address_format_t*</code>	<code>address</code>	32 bit pointer.
<code>unsigned int</code>	<code>count</code>	Length in 32-bit words to write (valid values 1 - 16).
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap).
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion.

3.9.4.2 cluster_target_xpb_read

Prototype:

```
void cluster_target_xpb_read(__xread void* xfer, volatile
cluster_target_xpb_address_format_t* address, unsigned int count, sync_t sync, SIGNAL*
sig_ptr)
```

Description:

Read data from XPB target over the XPB bus.

Transactions from the ARM may be global, but from other islands must be to within their island or its slaves.

Table 3.327. cluster_target_xpb_read parameters

Type	Name	Description
__xread void*	xfer	Transfer registers containing data read.
volatile cluster_target_xpb_address_format_t*	address	32 bit pointer.
unsigned int	count	Length in 32-bit words to write (valid values 1 - 16).
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap).
SIGNAL*	sig_ptr	Signal to raise upon completion.

3.9.4.3 cluster_target_reflect_write_sig_none

Prototype:

```
void cluster_target_reflect_write_sig_none(__xwrite void* xfer, volatile
cluster_target_reflect_address_format_t* address, unsigned int count)
```

Description:

Write data from transfer registers of initiating ME to the transfer registers of another ME.

Read data from the initiating ME and write it to a remote ME without providing a signal. Specify command arguments as per `cluster_target_reflect_address_format_t` command address field.



Note

The valid range for count is between 1 and 16 for Silicon revision B0 or higher and between 1 and 14 for Silicon revision less than B0. See Errata 'Misc engine Pull-buffer which stores Pull-data can erroneously overflow if the post-pull id Command FIFO is not full.'

Table 3.328. cluster_target_reflect_write_sig_none parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>xfer</code>	Transfer registers containing data to write.
<code>volatile cluster_target_reflect_address_format_t*</code>	<code>address</code>	32 bit pointer.
<code>unsigned int</code>	<code>count</code>	Length in 32-bit words to write (valid values 1 - 16).

3.9.4.4 cluster_target_reflect_read_sig_none

Prototype:

```
void cluster_target_reflect_read_sig_none(__xread void* xfer, volatile
cluster_target_reflect_address_format_t* address, unsigned int count)
```

Description:

Read data from transfer registers of remote ME and write to the transfer registers of the initiating ME.

Read data from the remote ME and write it to a initiating ME without providing a signal. Specify command arguments as per `cluster_target_reflect_address_format_t` command address field.



Note

The valid range for count is between 1 and 16 for Silicon revision B0 or higher and between 1 and 14 for Silicon revision less than B0. See Errata 'Misc engine Pull-buffer which stores Pull-data can erroneously overflow if the post-pull id Command FIFO is not full.'

Table 3.329. cluster_target_reflect_read_sig_none parameters

Type	Name	Description
<code>__xread void*</code>	<code>xfer</code>	Transfer registers containing data to read.
<code>volatile cluster_target_reflect_address_format_t*</code>	<code>address</code>	32 bit pointer.
<code>unsigned int</code>	<code>count</code>	Length in 32-bit words to write (valid values 1 - 16).

3.9.4.5 cluster_target_reflect_write_sig_both

Prototype:

```
void cluster_target_reflect_write_sig_both(__xwrite void* xfer, volatile
cluster_target_reflect_address_format_t* address, unsigned int count, sync_t sync, SIGNAL*
sig_ptr)
```

Description:

Write data from transfer registers of initiating ME to the transfer registers of another ME.

Read data from the initiating ME and write it to a remote ME signalling both MEs. Specify command arguments as per `cluster_target_reflect_address_format_t` command address field.



Note

The valid range for count is between 1 and 16 for Silicon revision B0 or higher and between 1 and 14 for Silicon revision less than B0. See Errata 'Misc engine Pull-buffer which stores Pull-data can erroneously overflow if the post-pull id Command FIFO is not full.'

Table 3.330. `cluster_target_reflect_write_sig_both` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>xfer</code>	Transfer registers containing data to write.
<code>volatile cluster_target_reflect_address_format_t*</code>	<code>address</code>	32 bit pointer.
<code>unsigned int</code>	<code>count</code>	Length in 32-bit words to write (valid values 1 - 16).
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap).
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion.

3.9.4.6 `cluster_target_reflect_read_sig_both`

Prototype:

```
void cluster_target_reflect_read_sig_both(__xread void* xfer, volatile
cluster_target_reflect_address_format_t* address, unsigned int count, sync_t sync, SIGNAL*
sig_ptr)
```

Description:

Read data from transfer registers of remote ME and write to the transfer registers of the initiating ME.

Read data from the remote ME and write it to a initiating ME signalling both MEs. Specify command arguments as per `cluster_target_reflect_address_format_t` command address field.



Note

The valid range for count is between 1 and 16 for Silicon revision B0 or higher and between 1 and 14 for Silicon revision less than B0. See Errata 'Misc engine Pull-buffer which stores Pull-data can erroneously overflow if the post-pull id Command FIFO is not full.'

Table 3.331. cluster_target_reflect_read_sig_both parameters

Type	Name	Description
<code>__xread void*</code>	<code>xfer</code>	Transfer registers containing data to read.
<code>volatile cluster_target_reflect_address_format_t*</code>	<code>address</code>	32 bit pointer.
<code>unsigned int</code>	<code>count</code>	Length in 32-bit words to write (valid values 1 - 16).
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap).
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion.

3.9.4.7 cluster_target_reflect_write_sig_init

Prototype:

```
void cluster_target_reflect_write_sig_init(__xwrite void* xfer, volatile
cluster_target_reflect_address_format_t* address, unsigned int count, sync_t sync, SIGNAL*
sig_ptr)
```

Description:

Write data from transfer registers of initiating ME to the transfer registers of another ME.

Read data from the initiating ME and write it to a remote ME signalling the initiating ME. Specify command arguments as per `cluster_target_reflect_address_format_t` command address field.



Note

The valid range for count is between 1 and 16 for Silicon revision B0 or higher and between 1 and 14 for Silicon revision less than B0. See Errata 'Misc engine Pull-buffer which stores Pull-data can erroneously overflow if the post-pull id Command FIFO is not full.'

Table 3.332. cluster_target_reflect_write_sig_init parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>xfer</code>	Transfer registers containing data to write.
<code>volatile cluster_target_reflect_address_format_t*</code>	<code>address</code>	32 bit pointer.
<code>unsigned int</code>	<code>count</code>	Length in 32-bit words to write (valid values 1 - 16).
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap).
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion.

3.9.4.8 cluster_target_reflect_read_sig_init

Prototype:

```
void cluster_target_reflect_read_sig_init(__xread void* xfer, volatile
cluster_target_reflect_address_format_t* address, unsigned int count, sync_t sync, SIGNAL*
sig_ptr)
```

Description:

Read data from transfer registers of remote ME and write to the transfer registers of the initiating ME.

Read data from the remote ME and write it to a initiating ME signalling initiating ME. Specify command arguments as per `cluster_target_reflect_address_format_t` command address field.



Note

The valid range for count is between 1 and 16 for Silicon revision B0 or higher and between 1 and 14 for Silicon revision less than B0. See Errata 'Misc engine Pull-buffer which stores Pull-data can erroneously overflow if the post-pull id Command FIFO is not full.'

Table 3.333. `cluster_target_reflect_read_sig_init` parameters

Type	Name	Description
<code>__xread void*</code>	<code>xfer</code>	Transfer registers containing data to read.
<code>volatile cluster_target_reflect_address_format_t*</code>	<code>address</code>	32 bit pointer.
<code>unsigned int</code>	<code>count</code>	Length in 32-bit words to write (valid values 1 - 16).
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap).
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion.

3.9.4.9 cluster_target_reflect_write_sig_remote

Prototype:

```
void cluster_target_reflect_write_sig_remote(__xwrite void* xfer, volatile
cluster_target_reflect_address_format_t* address, unsigned int count, sync_t sync, SIGNAL*
sig_ptr)
```

Description:

Write data from transfer registers of initiating ME to the transfer registers of another ME.

Read data from the initiating ME and write it to a remote ME signalling the remote ME. Specify command arguments as per `cluster_target_reflect_address_format_t` command address field.



Note

The valid range for count is between 1 and 16 for Silicon revision B0 or higher and between 1 and 14 for Silicon revision less than B0. See Errata 'Misc engine Pull-buffer which stores Pull-data can erroneously overflow if the post-pull id Command FIFO is not full.'

Table 3.334. cluster_target_reflect_write_sig_remote parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>xfer</code>	Transfer registers containing data to write.
<code>volatile cluster_target_reflect_address_format_t*</code>	<code>address</code>	32 bit pointer.
<code>unsigned int</code>	<code>count</code>	Length in 32-bit words to write (valid values 1 - 16).
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done only).
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion.

3.9.4.10 cluster_target_reflect_read_sig_remote

Prototype:

```
void cluster_target_reflect_read_sig_remote(__xread void* xfer, volatile
cluster_target_reflect_address_format_t* address, unsigned int count, sync_t sync, SIGNAL*
sig_ptr)
```

Description:

Read data from transfer registers of remote ME and write to the transfer registers of the initiating ME.

Read data from the remote ME and write it to a initiating ME signalling remote ME. Specify command arguments as per `cluster_target_reflect_address_format_t` command address field.



Note

The valid range for count is between 1 and 16 for Silicon revision B0 or higher and between 1 and 14 for Silicon revision less than B0. See Errata 'Misc engine Pull-buffer which stores Pull-data can erroneously overflow if the post-pull id Command FIFO is not full.'

Table 3.335. cluster_target_reflect_read_sig_remote parameters

Type	Name	Description
<code>__xread void*</code>	<code>xfer</code>	Transfer registers containing data to read.
<code>volatile cluster_target_reflect_address_format_t*</code>	<code>address</code>	32 bit pointer.

Type	Name	Description
unsigned int	<i>count</i>	Length in 32-bit words to write (valid values 1 - 16).
sync_t	<i>sync</i>	Type of synchronization to use (sig_done only).
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion.

3.9.4.11 cluster_target_sig_me_ctx

Prototype:

```
void cluster_target_sig_me_ctx(volatile cluster_target_signal_me_address_format_t* address)
```

Description:

Send a signal to another ME.

Signal another ME as specified in the address.

```
{
    cluster_target_signal_me_address_format_t    signal_command;
    SIGNAL                                         signal;

    signal_command.value = 0;
    signal_command.remote_island = 34;           // island 34 (i34)
    signal_command.remote_context = 0;           // thread/context 0
    signal_command.remote_master = 5;            // Master IDs 4..15 for MEs 0..11 on i32..i38
    signal_command.signal_number = __signal_number(&signal);

    // signal remote island 34, ME 1
    cluster_target_sig_me_ctx(&signal_command);
}
```

Table 3.336. cluster_target_sig_me_ctx parameters

Type	Name	Description
volatile cluster_target_signal_me_address_format_t*	<i>address</i>	32 bit pointer.

3.9.4.12 cluster_target_next_neighbour_write

Prototype:

```
void cluster_target_next_neighbour_write(__xwrite void* xfer, volatile
cluster_target_next_neighbour_write_address_format_t* address, unsigned int count, sync_t
sync, SIGNAL* sig_ptr)
```

Description:

Write data to the next neighbor register or FIFO of a ME.

A ME does a write data to the next neighbour from the xfer registers.



Note

The valid range for count is between 1 and 16 for Silicon revision B0 or higher and between 1 and 14 for Silicon revision less than B0. See Errata 'Misc engine Pull-buffer which stores Pull-data can erroneously overflow if the post-pull id Command FIFO is not full.'

Table 3.337. cluster_target_next_neighbour_write parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>xfer</code>	Transfer registers containing data to write.
<code>volatile cluster_target_next_neighbour_write_address_format_t*</code>	<code>address</code>	32 bit pointer.
<code>unsigned int</code>	<code>count</code>	Length in 32-bit words to write (valid values 1 - 16).
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap).
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion.

3.9.4.13 cluster_target_next_neighbour_write_ind

Prototype:

```
void cluster_target_next_neighbour_write_ind(__xwrite void* xfer, volatile
cluster_target_next_neighbour_write_address_format_t* address, unsigned int max_nn,
generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write data to the next neighbor register or FIFO of a ME.

A ME does a write data to the next neighbour but the data can be pulled from another ME. If ctnn write without indirect address then island master, data master, signal master, signal number can be left 0. See DB Figure 9.10 for an example of using indirect address.



Note

The valid range for max_nn is between 1 and 16 for Silicon revision B0 or higher and between 1 and 14 for Silicon revision less than B0. See Errata 'Misc engine Pull-buffer which stores Pull-data can erroneously overflow if the post-pull id Command FIFO is not full.'

Below is an example where an island (i.e. i33) does next neighbour write to i32 and the data is pulled from i34.

```

cluster_target_next_neighbour_write_address_format_t      command;
unsigned int      master = 4;
unsigned int      signal_master = 4;
SIGNAL           sig;
unsigned int      count = 16;
generic_ind_t    ind;
unsigned int      data_island = 34; // i34

xfer_write[0] = 0x12345678;
xfer_write[1] = 0x12345678+10;

command.value = 0;
command.remote_island = 32;          // i32
command.master = master;
command.signal_number = 0x1;
command.address_mode = CT_ADDRESS_MODE_ABSOLUTE;
command.NN_register_number = 0;       // start at register 0

ovr_init(&ind, ovr_signal_island_and_data_master | ovr_signal_number | ovr_length);
ovr_set(
    &ind,
    ovr_signal_island_and_data_master,
    data_island << 8 | master << 4 | signal_master
);
ovr_set(&ind, ovr_signal_number, 1);
ovr_set(&ind, ovr_length, count - 1);

cluster_target_next_neighbour_write_ind
(
    (void *)&xfer_write[0],
    &command,
    16,
    ind,
    ctx_swap,
    &sig
);

```

Table 3.338. cluster_target_next_neighbour_write_ind parameters

Type	Name	Description
__xwrite void*	xfer	Transfer registers containing data to write.
volatile cluster_target_next_neighbour_write_address_format_t*	address	32 bit pointer.
unsigned int	max_nn	Maximum number of 32-bit words to read (1 - 16)
generic_ind_t	ind	Indirect word
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap).
SIGNAL*	sig_ptr	Signal to raise upon completion.

Availability:

NFP-6xxx Indirect Reference Mode

3.9.4.14 cluster_target_ring_put

Prototype:

```
void cluster_target_ring_put(__xwrite void* data, unsigned int island_number, unsigned int ring_number, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Put data on Cluster Target ring.

Write 32-bit words to tail of ring and update tail pointer. If there are not enough empty slots in the ring, no words are added and the tail pointer is not updated. This will raise a status of CT_RING_FULL on the status bus of the ring if 3/4 or greater.



Note

The valid range for count is between 1 and 16 for Silicon revision B0 or higher and between 1 and 14 for Silicon revision less than B0. See Errata 'Misc engine Pull-buffer which stores Pull-data can erroneously overflow if the post-pull id Command FIFO is not full.'

Table 3.339. cluster_target_ring_put parameters

Type	Name	Description
__xwrite void*	data	Data to put onto ring.
unsigned int	island_number	Island number of the target CTM.
unsigned int	ring_number	Ring number to initialise (0 to 15).
unsigned int	count	Number of 32-bit words to put into ring (valid values 1 - 16).
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap).
SIGNAL*	sig_ptr	Signal to raise upon completion.

3.9.4.15 cluster_target_ring_get

Prototype:

```
void cluster_target_ring_get(__xread void* data, unsigned int island_number, unsigned int ring_number, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read data from Cluster Target ring and put in transfer register.

Read 32-bit words from head of ring and update head pointer. If there are not enough entries in the ring, no valid data is returned and the head pointer is not updated. This will raise a status of CT_RING_FULL on the status bus of the ring if 3/4 or greater.

Table 3.340. cluster_target_ring_get parameters

Type	Name	Description
__xread void*	data	Data read from ring.
unsigned int	island_number	Island number of the target CTM.
unsigned int	ring_number	Ring number to initialise (0 to 15).
unsigned int	count	Number of 32-bit words to read (valid values 1 - 16).
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

3.9.4.16 cluster_target_ring_init_ptr32

Prototype:

```
void cluster_target_ring_init_ptr32(unsigned int ring_number, volatile void __addr32
__ctm* base_ptr, enum CT_RING_SIZE size, enum CT_RING_STATUS status_events, sync_t sync,
SIGNAL* sig_ptr)
```

Description:

Initialise a Cluster Target ring on the local island.

Setup a ring of specified ring number, size and address in the local island in Cluster Target memory. In the example the CT ring is created on the local island.

```
__ctm __addr32 __align(128*sizeof(int)) unsigned int ring_base[128];

unsigned int          ring_index = 2;
unsigned int          island = __island();
unsigned int          count = 3;
SIGNAL               sig;

cluster_target_ring_init_ptr32(
    ring_index,
    ring_base,
    CT_RING_SIZE_128,
    CT_RING_FULL,
    ctx_swap,
    &sig
);

// put data on the ring
{
    __xwrite unsigned int wr_data[3];

    wr_data[0] = 0x12345678;
    wr_data[1] = 0x87654321;
```

```

    wr_data[2] = 0x10101010;

    cluster_target_ring_put((void *)&wr_data[0], island, ring_index, count, ctx_swap, &sig);
}

// get data from the ring
{
    __xread unsigned int rd_data[3];

    cluster_target_ring_get((void *)&rd_data[0], island, ring_index, count, ctx_swap, &sig);

    if (rd_data[0] != 0x12345678)
    {
        return 0;           // We have an error
    }
}

return 1;

```

Table 3.341. cluster_target_ring_init_ptr32 parameters

Type	Name	Description
unsigned int	<i>ring_number</i>	Ring number to initialise (0 to 15)
volatile void __addr32 __ctm*	<i>base_ptr</i>	32-bit pointer to physical address in CTM
enum CT_RING_SIZE	<i>size</i>	Size of ring of enum CT_RING_SIZE
enum CT_RING_STATUS	<i>status_events</i>	List of statuses of enum CT_RING_STATUS which should be generated
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.9.4.17 cluster_target_ring_init_ptr40

Prototype:

```
void cluster_target_ring_init_ptr40(unsigned int ring_number, volatile void __addr40
__ctm* base_ptr, enum CT_RING_SIZE size, enum CT_RING_STATUS status_events, sync_t sync,
SIGNAL* sig_ptr)
```

Description:

Initialise a Cluster Target ring on a specific (non-local) island.

Setup a ring of specified ring number, size and address in Cluster Target memory. Island is taken from the base_ptr[37:32]. In the example the CT ring is created on remote island 33.

```

__ctm_n(1) __addr40 __align(128*sizeof(uint32_t)) uint32_t ring_base_addr[128];

unsigned int      ring_index = 2;
unsigned int      island = 33;
```

```

unsigned int      count = 3;
SIGNAL           sig;

cluster_target_ring_init_ptr40(
    ring_index,
    ring_base_addr,
    CT_RING_SIZE_128,
    CT_RING_FULL,
    ctx_swap,
    &sig
);

// put data on the ring
{
    __xwrite unsigned int wr_data[3];

    wr_data[0] = 0x12345678;
    wr_data[1] = 0x87654321;
    wr_data[2] = 0x10101010;

    cluster_target_ring_put((void *)&wr_data[0], island, ring_index, count, ctx_swap, &sig);
}

// get data from the ring
{
    __xread unsigned int rd_data[3];

    cluster_target_ring_get((void *)&rd_data[0], island, ring_index, count, ctx_swap, &sig);

    if (rd_data[0] != 0x12345678)
    {
        return 0;          // We have an error
    }
}

return 1;

```

Table 3.342. cluster_target_ring_init_ptr40 parameters

Type	Name	Description
unsigned int	<i>ring_number</i>	Ring number to initialise (0 to 15)
volatile void __addr40 __ctm*	<i>base_ptr</i>	40-bit pointer to physical address in CTM with island in top bits.
enum CT_RING_SIZE	<i>size</i>	Size of ring of enum CT_RING_SIZE
enum CT_RING_STATUS	<i>status_events</i>	List of statuses of enum CT_RING_STATUS which should be generated
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.9.4.18 cluster_target_ring_init

Prototype:

```
void cluster_target_ring_init(unsigned int island_number, unsigned int ring_number,
unsigned int base_address, enum CT_RING_SIZE size, enum CT_RING_STATUS status_events,
sync_t sync, SIGNAL* sig_ptr)
```

Description:

Initialise a Cluster Target ring.

Setup a ring of specified ring number, size and base address in the specified island in Cluster Target memory. The parameter base_address is shifted right by a number depending on the size of the ring. This gives the actual byte address of the base of the ring.

Table 3.343. cluster_target_ring_init parameters

Type	Name	Description
unsigned int	<i>island_number</i>	Island number of the target CTM.
unsigned int	<i>ring_number</i>	Ring number to initialise (0 to 15)
unsigned int	<i>base_address</i>	Base address of ring which could be anywhere in CT.
enum CT_RING_SIZE	<i>size</i>	Size of ring of enum CT_RING_SIZE
enum CT_RING_STATUS	<i>status_events</i>	List of statuses of enum CT_RING_STATUS which should be generated
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.10 Control and Status Register Access Intrinsics

This section discusses the functions used to access the Control and Status Register (CSR) registers.

3.10.1 Control and Status Register Access Enumerations

3.10.1.1 local_csr_t

This enumeration specifies the CSRs used in local_csr_read and local_csr_write functions.

Table 3.344. enum local_csr_t

Name	Description
local_csr_ustore_address	
local_csr_ustore_data_lower	
local_csr_ustore_data_upper	
local_csr_ustore_error_status	

Name	Description
local_csr_alu_out	
local_csr_ctx_arb_ctrl	
local_csr_ctx_enables	
local_csr_cc_enable	
local_csr_csr_ctx_pointer	
local_csr_pc_breakpoint_0	
local_csr_pc_breakpoint_1	
local_csr_pc_breakpoint_status	
local_csr_lm_register_error_status	
local_csr_lm_error_status	
local_csr_lm_error_mask	
local_csr_indirect_ctx_sts	
local_csr_active_ctx_sts	
local_csr_indirect_ctx_sig_events	
local_csr_active_ctx_sig_events	
local_csr_indirect_ctx_wakeup_events	
local_csr_active_ctx_wakeup_events	
local_csr_indirect_ctx_future_count	
local_csr_active_ctx_future_count	
local_csr_byte_index	
local_csr_t_index	
local_csr_indirect_future_count_signal	
local_csr_active_future_count_signal	
local_csr_nn_put	
local_csr_nn_get	
local_csr_indirect_lm_addr_0	
local_csr_active_lm_addr_0	
local_csr_indirect_lm_addr_1	
local_csr_active_lm_addr_1	
local_csr_indirect_lm_addr_2	
local_csr_active_lm_addr_2	
local_csr_indirect_lm_addr_3	
local_csr_active_lm_addr_3	
local_csr_indirect_lm_addr_0_byte_index	
local_csr_active_lm_addr_0_byte_index	

Name	Description
local_csr_indirect_lm_addr_1_byte_index	
local_csr_active_lm_addr_1_byte_index	
local_csr_indirect_lm_addr_2_byte_index	
local_csr_active_lm_addr_2_byte_index	
local_csr_indirect_lm_addr_3_byte_index	
local_csr_active_lm_addr_3_byte_index	
local_csr_indirect_predicate_cc	
local_csr_t_index_byte_index	
local_csr_timestamp_low	
local_csr_timestamp_high	
local_csr_next_neighbor_signal	
local_csr_prev_neighbor_signal	
local_csr_same_me_signal	
local_csr_crc_remainder	
local_csr_profile_count	
local_csr_pseudo_random_number	
local_csr_misc_control	
pc_breakpoint_0_mask	
pc_breakpoint_1_mask	
local_csr_mailbox0	
local_csr_mailbox1	
local_csr_mailbox2	
local_csr_mailbox3	
local_csr_mailbox_0	
local_csr_mailbox_1	
local_csr_mailbox_2	
local_csr_mailbox_3	
local_csr_cmd_indirect_ref0	
local_csr_cmd_indirect_ref1	
local_csr_cmd_indirect_ref_0	
local_csr_cmd_indirect_ref_1	
local_csr_reserved	
local_csr_ustore_address	Used to load programs into the Control Store.
local_csr_ustore_data_lower	Control Store Data - lower.
local_csr_ustore_data_upper	Control Store Data - upper.

Name	Description
local_csr_ustore_error_status	ECC errors during Control Store reads.
local_csr_alu_out	Debug to show state of ALU.
local_csr_ctx_arb_CNTL	Context Arbiter Control - used by the context arbiter and for debug.
local_csr_ctx_enables	Context Enables - used by the context arbiter and for debug.
local_csr_cc_enable	Condition Code Enable.
local_csr_csr_ctx_pointer	CSR Context Pointer.
local_csr_pc_breakpoint_0	PC Breakpoint 1 - PCB system.
local_csr_pc_breakpoint_1	PC Breakpoint 1 - PCB system.
local_csr_pc_breakpoint_status	PC Breakpoint - Status register associated with the PCB system.
local_csr_lm_register_error_status	Information about parity errors detected on Datapath Regs.
local_csr_lm_error_status	Status on ECC errors recorded on Local Memory reads.
local_csr_lm_error_mask	Controls Error Injection bits into an LM data-path word.
local_csr_indirect_ctx_sts	Indirect Context Status Register.
local_csr_active_ctx_sts	Active Context Status Register.
local_csr_indirect_ctx_sig_events	Indirect Context Signal Events Register.
local_csr_active_ctx_sig_events	Active Context Signal Events Register.
local_csr_indirect_ctx_wakeup_events	Indirect Context Wakeup Events Register.
local_csr_active_ctx_wakeup_events	Active Context Wakeup Events Register.
local_csr_indirect_ctx_future_count	Indirect Context Future Count Register.
local_csr_active_ctx_future_count	Active Context Future Count Register.
local_csr_byte_index	Byte Index Register.
local_csr_t_index	Transfer Index Register.
local_csr_indirect_future_count_signal	Which signal to set when FUTURE_COUNT == TIMESTAMP.
local_csr_active_future_count_signal	Which signal to set when FUTURE_COUNT == TIMESTAMP.
local_csr_nn_put	Next Neighbor Put Register.
local_csr_nn_get	Next Neighbor Get Register.
local_csr_indirect_lm_addr_0	Indirect Local Memory Address 0 Register.
local_csr_active_lm_addr_0	Active Local Memory Address 0 Register.
local_csr_indirect_lm_addr_1	Indirect Local Memory Address 1 Register.
local_csr_active_lm_addr_1	Active Local Memory Address 1 Register.
local_csr_indirect_lm_addr_2	Indirect Local Memory Address 2 Register.
local_csr_active_lm_addr_2	Active Local Memory Address 2 Register.

Name	Description
local_csr_indirect_lm_addr_3	Indirect Local Memory Address 3 Register.
local_csr_active_lm_addr_3	Active Local Memory Address 3 Register.
local_csr_indirect_lm_addr_0_byte_index	Alias of IndLMAAddr0 and ByteIndex.
local_csr_active_lm_addr_0_byte_index	Alias of ActLMAAddr0 and ByteIndex.
local_csr_indirect_lm_addr_1_byte_index	Alias of IndLMAAddr1 and ByteIndex.
local_csr_active_lm_addr_1_byte_index	Alias of ActLMAAddr1 and ByteIndex.
local_csr_indirect_lm_addr_2_byte_index	Alias of IndLMAAddr2 and ByteIndex.
local_csr_active_lm_addr_2_byte_index	Alias of ActLMAAddr2 and ByteIndex.
local_csr_indirect_lm_addr_3_byte_index	Alias of IndLMAAddr3 and ByteIndex.
local_csr_active_lm_addr_3_byte_index	Alias of ActLMAAddr3 and ByteIndex.
local_csr_indirect_predicate_cc	Indirect Predicate CC select.
local_csr_t_index_byte_index	This register is used when Transfer registers are accessed via indexed mode.
local_csr_timestamp_low	Timestamp is 64 bits. It counts up by one every sixteen cycles.
local_csr_timestamp_high	Timestamp is 64 bits. It counts up by one every sixteen cycles.
local_csr_next_neighbor_signal	Signal a Context in Next Neighbor.
local_csr_prev_neighbor_signal	Signal a Context in Previous Neighbor.
local_csr_same_me_signal	Signal another Context in same Microengine.
local_csr_crc_remainder	Result of the CRC operation after a crc instruction.
local_csr_profile_count	The profile count is used for code profiling and tuning.
local_csr_pseudo_random_number	Random number generator.
local_csr_misc_control	Miscellaneous Control Register.
pc_breakpoint_0_mask	Mask register associated with PC Breakpoint 0.
pc_breakpoint_1_mask	Mask register associated with PC Breakpoint 1.
local_csr_mailbox0	Mailbox Register 0.
local_csr_mailbox1	Mailbox Register 1.
local_csr_mailbox2	Mailbox Register 2.
local_csr_mailbox3	Mailbox Register 3.
local_csr_mailbox_0	Alias of Mailbox0.
local_csr_mailbox_1	Alias of Mailbox1.
local_csr_mailbox_2	Alias of Mailbox2.
local_csr_mailbox_3	Alias of Mailbox3.
local_csr_cmd_indirect_ref0	Indirect reference to control register 0.

Name	Description
local_csr_cmd_indirect_ref1	Indirect reference to control register 1.
local_csr_cmd_indirect_ref_0	Alias of local_csr_cmd_indirect_ref0.
local_csr_cmd_indirect_ref_1	Alias of local_csr_cmd_indirect_ref1.
local_csr_reserved	Reserved.

3.10.2 Control and Status Register Access Unions

3.10.2.1 ACTIVE_CTX_STS_t

Active_CTX_Status CSR format.

The Active_CTX_Status CSR maintains the context number of the Context currently executing. Each Microengine supports eight contexts (0 through 7).

Table 3.345. union ACTIVE_CTX_STS_t

Type	Name	Description
unsigned int	AB0:1	If set, the Microengine has a Context in the Executing state. If clear, no Context is in Executing State.
unsigned int	ISLAND_ID:6	Island number where the ME is instantiated.
unsigned int	ATXPC:17	Program count of the Executing Context. Only valid if AB0 is a 1. Valid values are 0 to 131071. This field provides a snapshot value of the PC. This value is used for tracking/code profiling purposes. When issued as a local_csr_read from the Microengine, the PC value may not be the exact PC value of the local_csr_rd instruction.
unsigned int	RESERVED:1	Reserved.
unsigned int	ME_ID:4	The unique ME number which identifies it within the cluster where it happens to be instantiated.
unsigned int	ACNO:3	The number of the Executing context. Only valid if AB0 bit is a 1.
unsigned int	value	Accessor to all bits simultaneously.

FIX ME doxy_Control_and_Status_Register_Accessor_functions.xml

3.11 CRC Intrinsics

This section describes the functions that provide access to the Cyclic Redundancy Check (CRC) unit of the Netronome NFP-6000 network processor.

3.11.1 CRC Enumerations

3.11.1.1 bytesSpecifier_t

CRC bytes specifier.

The bytesSpecifier_t enumeration is used as an argument to the CRC functions and specifies one or more contiguous bytes within a 32-bit word of big-endian or little-endian data. For example, the bytes_0_3 item in this enumeration refers to bytes 0 through 3. When using the big-endian CRC functions, byte 0 refers to the most significant byte and byte 3 refers to the least significant byte. When using the little-endian CRC functions, byte 0 refers to the least significant byte and byte 3 refers to the most significant byte. This enumeration type specifies the bytes to be used with CRC operations.

Table 3.346. enum bytesSpecifier_t

Name	Description
bytes_0_3	BE: 0, 1, 2, 3 LE: 3, 2, 1, 0.
bytes_0_2	BE: 0, 1, 2 LE: 2, 1, 0.
bytes_0_1	BE: 0, 1 LE: 1, 0.
byte_0	BE: 0 LE: 0.
bytes_1_3	BE: 1, 2, 3 LE: 3, 2, 1 .
bytes_2_3	BE: 2, 3 LE: 3, 2 .
byte_3	BE: 3 LE: 3 .

3.11.2 CRC Functions

3.11.2.1 crc_5_be

Prototype:

```
__intrinsic unsigned int crc_5_be(unsigned int data, bytesSpecifier_t bspec)
```

Description:

CRC 5 computation in Big-endian format.

Perform a CRC-5 computation on specified bytes of the data argument that are assumed to be in Big-endian format and return the unmodified value of data.

`crc_write()` can be used to initialize the CRC value and the result can be obtained from `crc_read()`.

Table 3.347. `crc_5_be` parameters

Type	Name	Description
unsigned int	<code>data</code>	Data on which to perform the CRC 5 computation
<code>bytesSpecifier_t</code>	<code>bspec</code>	Specified bytes in the data argument on which to perform the computation

See Also:

- `crc_read()` and `crc_write()`

3.11.2.2 `crc_5_le`

Prototype:

```
_intrinsic unsigned int crc_5_le(unsigned int data, bytesSpecifier_t bspec)
```

Description:

CRC 5 computation in Little-endian format.

Perform a CRC-5 computation on specified bytes of the data argument that are assumed to be in Little-endian format and return the unmodified value of data.

`crc_write()` can be used to initialize the CRC value and the result can be obtained from `crc_read()`.

Table 3.348. `crc_5_le` parameters

Type	Name	Description
unsigned int	<code>data</code>	Data on which to perform the CRC 5 computation
<code>bytesSpecifier_t</code>	<code>bspec</code>	Specified bytes in the data argument on which to perform the computation

See Also:

- `crc_read()` and `crc_write()`

3.11.2.3 `crc_10_be`

Prototype:

```
_intrinsic unsigned int crc_10_be(unsigned int data, bytesSpecifier_t bspec)
```

Description:

CRC 10 computation in Big-endian format.

Perform a CRC-10 computation on specified bytes of the data argument that are assumed to be in Big-endian format and return the unmodified value of data.

`crc_write()` can be used to initialize the CRC value and the result can be obtained from `crc_read()`.

Table 3.349. `crc_10_be` parameters

Type	Name	Description
unsigned int	<code>data</code>	Data on which to perform the CRC 10 computation
<code>bytesSpecifier_t</code>	<code>bspec</code>	Specified bytes in the data argument on which to perform the computation

See Also:

- `crc_read()` and `crc_write()`

3.11.2.4 `crc_10_le`

Prototype:

```
_intrinsic unsigned int crc_10_le(unsigned int data, bytesSpecifier_t bspec)
```

Description:

CRC 10 computation in Little-endian format.

Perform a CRC-10 computation on specified bytes of the data argument that are assumed to be in Little-endian format and return the unmodified value of data.

`crc_write()` can be used to initialize the CRC value and the result can be obtained from `crc_read()`.

Table 3.350. `crc_10_le` parameters

Type	Name	Description
unsigned int	<code>data</code>	Data on which to perform the CRC 10 computation
<code>bytesSpecifier_t</code>	<code>bspec</code>	Specified bytes in the data argument on which to perform the computation

See Also:

- `crc_read()` and `crc_write()`

3.11.2.5 `crc_16_be`

Prototype:

```
_intrinsic unsigned int crc_16_be(unsigned int data, bytesSpecifier_t bspec)
```

Description:

16-bit CCITT CRC computation in Big-endian format.

Perform a 16-bit CCITT CRC computation on specified bytes in the data argument that is assumed to be in Big-endian format and return the unmodified value of data.

`crc_write()` can be used to initialize the CRC value and the result can be obtained from `crc_read()`.

Table 3.351. `crc_16_be` parameters

Type	Name	Description
unsigned int	<code>data</code>	Data on which to perform the 16-bit CCITT CRC computation
<code>bytesSpecifier_t</code>	<code>bspec</code>	Specified bytes in the data argument on which to perform the computation

See Also:

- `crc_read()` and `crc_write()`

3.11.2.6 `crc_16_le`

Prototype:

```
_intrinsic unsigned int crc_16_le(unsigned int data, bytesSpecifier_t bspec)
```

Description:

16-bit CCITT CRC computation in Little-endian format.

Perform a 16-bit CCITT CRC computation on specified bytes in the data argument that is assumed to be in Little-endian format and return the unmodified value of data.

`crc_write()` can be used to initialize the CRC value and the result can be obtained from `crc_read()`.

Table 3.352. `crc_16_le` parameters

Type	Name	Description
unsigned int	<code>data</code>	Data on which to perform the 16-bit CCITT CRC computation
<code>bytesSpecifier_t</code>	<code>bspec</code>	Specified bytes in the data argument on which to perform the computation

See Also:

- `crc_read()` and `crc_write()`

3.11.2.7 `crc_32_be`

Prototype:

```
_intrinsic unsigned int crc_32_be(unsigned int data, bytesSpecifier_t bspec)
```

Description:

32-bit iSCSI CRC computation in Big-endian format.

Perform a CRC-32 computation on specified bytes in the data argument that is assumed to be in Big-endian format and return the unmodified value of data.

`crc_write()` can be used to initialize the CRC value and the result can be obtained from `crc_read()`.

Table 3.353. `crc_32_be` parameters

Type	Name	Description
unsigned int	<code>data</code>	Data on which to perform the 32-bit iSCSI CRC computation
<code>bytesSpecifier_t</code>	<code>bspec</code>	Specified bytes in the data argument on which to perform the computation

See Also:

- `crc_read()` and `crc_write()`

3.11.2.8 `crc_32_le`

Prototype:

```
_intrinsic unsigned int crc_32_le(unsigned int data, bytesSpecifier_t bspec)
```

Description:

32-bit iSCSI CRC computation in Little-endian format.

Perform a CRC-32 computation on specified bytes in the data argument that is assumed to be in Little-endian format and return the unmodified value of data.

`crc_write()` can be used to initialize the CRC value and the result can be obtained from `crc_read()`.

Table 3.354. `crc_32_le` parameters

Type	Name	Description
unsigned int	<code>data</code>	Data on which to perform the 32-bit iSCSI CRC computation
<code>bytesSpecifier_t</code>	<code>bspec</code>	Specified bytes in the data argument on which to perform the computation

See Also:

- `crc_read()` and `crc_write()`

3.11.2.9 `crc_ccitt_be`

Prototype:

```
_intrinsic unsigned int crc_ccitt_be(unsigned int data, bytesSpecifier_t bspec)
```

Description:

16-bit CCITT CRC computation in Big-endian format.

Perform a 16-bit CCITT CRC computation on specified bytes in the data argument that is assumed to be in Big-endian format and return the unmodified value of data.

`crc_write()` can be used to initialize the CRC value and the result can be obtained from `crc_read()`.

Table 3.355. `crc_ccitt_be` parameters

Type	Name	Description
unsigned int	<code>data</code>	Data on which to perform the 16-bit CCITT CRC computation
<code>bytesSpecifier_t</code>	<code>bspec</code>	Specified bytes in the data argument on which to perform the computation

See Also:

- `crc_read()` and `crc_write()`

3.11.2.10 `crc_ccitt_le`

Prototype:

```
_intrinsic unsigned int crc_ccitt_le(unsigned int data, bytesSpecifier_t bspec)
```

Description:

16-bit CCITT CRC computation in Little-endian format.

Perform a 16-bit CCITT CRC computation on specified bytes in the data argument that is assumed to be in Little-endian format and return the unmodified value of data.

`crc_write()` can be used to initialize the CRC value and the result can be obtained from `crc_read()`.

Table 3.356. `crc_ccitt_le` parameters

Type	Name	Description
unsigned int	<code>data</code>	Data on which to perform the 16-bit CCITT CRC computation
<code>bytesSpecifier_t</code>	<code>bspec</code>	Specified bytes in the data argument on which to perform the computation

See Also:

- `crc_read()` and `crc_write()`

3.11.2.11 `crc_iscsi_be`

Prototype:

```
_intrinsic unsigned int crc_iscsi_be(unsigned int data, bytesSpecifier_t bspec)
```

Description:

32-bit iSCSI CRC computation in Big-endian format.

Perform a 32-bit iSCSI CRC computation on specified bytes in the data argument that is assumed to be in Big-endian format and return the unmodified value of data.

`crc_write()` can be used to initialize the CRC value and the result can be obtained from `crc_read()`.

Table 3.357. `crc_iscsi_be` parameters

Type	Name	Description
unsigned int	<code>data</code>	Data on which to perform the 32-bit iSCSI CRC computation
<code>bytesSpecifier_t</code>	<code>bspec</code>	Specified bytes in the data argument on which to perform the computation

See Also:

- `crc_read()` and `crc_write()`

3.11.2.12 `crc_iscsi_le`

Prototype:

```
_intrinsic unsigned int crc_iscsi_le(unsigned int data, bytesSpecifier_t bspec)
```

Description:

32-bit iSCSI CRC computation in Little-endian format.

Perform a 32-bit iSCSI CRC computation on specified bytes in the data argument that is assumed to be in Little-endian format and return the unmodified value of data.

`crc_write()` can be used to initialize the CRC value and the result can be obtained from `crc_read()`.

Table 3.358. `crc_iscsi_le` parameters

Type	Name	Description
unsigned int	<code>data</code>	Data on which to perform the 32-bit iSCSI CRC computation
<code>bytesSpecifier_t</code>	<code>bspec</code>	Specified bytes in the data argument on which to perform the computation

See Also:

- `crc_read()` and `crc_write()`

3.11.2.13 `crc_5_be_bit_swap`

Prototype:

```
unsigned int crc_5_be_bit_swap(unsigned int data, bytesSpecifier_t bspec)
```

Description:

CRC 5 computation and bit swap in Big-endian format.

Perform a CRC-5 computation on specified bytes of the data argument that are assumed to be in Big-endian format and return the unmodified value of data. The bits in each byte are swapped before the computation begins.

`crc_write()` can be used to initialize the CRC value and the result can be obtained from `crc_read()`.

Table 3.359. `crc_5_be_bit_swap` parameters

Type	Name	Description
unsigned int	<code>data</code>	Data on which to perform the CRC 5 computation
<code>bytesSpecifier_t</code>	<code>bspec</code>	Specified bytes in the data argument on which to perform the computation

See Also:

- `crc_read()` and `crc_write()`

3.11.2.14 `crc_5_le_bit_swap`

Prototype:

```
unsigned int crc_5_le_bit_swap(unsigned int data, bytesSpecifier_t bspec)
```

Description:

CRC 5 computation and bit swap in Little-endian format.

Perform a CRC-5 computation on specified bytes of the data argument that are assumed to be in Little-endian format and return the unmodified value of data. The bits in each byte are swapped before the computation begins.

`crc_write()` can be used to initialize the CRC value and the result can be obtained from `crc_read()`.

Table 3.360. `crc_5_le_bit_swap` parameters

Type	Name	Description
unsigned int	<code>data</code>	Data on which to perform the CRC 5 computation
<code>bytesSpecifier_t</code>	<code>bspec</code>	Specified bytes in the data argument on which to perform the computation

See Also:

- `crc_read()` and `crc_write()`

3.11.2.15 `crc_10_be_bit_swap`

Prototype:

```
unsigned int crc_10_be_bit_swap(unsigned int data, bytesSpecifier_t bspec)
```

Description:

CRC 10 computation and bit swap in Big-endian format.

Perform a CRC-10 computation on specified bytes of the data argument that are assumed to be in Big-endian format and return the unmodified value of data. The bits in each byte are swapped before the computation begins.

`crc_write()` can be used to initialize the CRC value and the result can be obtained from `crc_read()`.

Table 3.361. `crc_10_be_bit_swap` parameters

Type	Name	Description
unsigned int	<code>data</code>	Data on which to perform the CRC 10 computation
<code>bytesSpecifier_t</code>	<code>bspec</code>	Specified bytes in the data argument on which to perform the computation

See Also:

- `crc_read()` and `crc_write()`

3.11.2.16 `crc_10_le_bit_swap`

Prototype:

```
unsigned int crc_10_le_bit_swap(unsigned int data, bytesSpecifier_t bspec)
```

Description:

CRC 10 computation and bit swap in Little-endian format.

Perform a CRC-10 computation on specified bytes of the data argument that are assumed to be in Little-endian format and return the unmodified value of data. The bits in each byte are swapped before the computation begins.

`crc_write()` can be used to initialize the CRC value and the result can be obtained from `crc_read()`.

Table 3.362. `crc_10_le_bit_swap` parameters

Type	Name	Description
unsigned int	<code>data</code>	Data on which to perform the CRC 10 computation
<code>bytesSpecifier_t</code>	<code>bspec</code>	Specified bytes in the data argument on which to perform the computation

See Also:

- `crc_read()` and `crc_write()`

3.11.2.17 `crc_16_be_bit_swap`

Prototype:

```
unsigned int crc_16_be_bit_swap(unsigned int data, bytesSpecifier_t bspec)
```

Description:

16-bit CCITT CRC computation and bit swap in Big-endian format.

Perform a 16-bit CCITT CRC computation on specified bytes of the data argument that are assumed to be in Big-endian format and return the unmodified value of data. The bits in each byte are swapped before the computation begins.

`crc_write()` can be used to initialize the CRC value and the result can be obtained from `crc_read()`.

Table 3.363. `crc_16_be_bit_swap` parameters

Type	Name	Description
unsigned int	<code>data</code>	Data on which to perform the 16-bit CCITT CRC computation
<code>bytesSpecifier_t</code>	<code>bspec</code>	Specified bytes in the data argument on which to perform the computation

See Also:

- `crc_read()` and `crc_write()`

3.11.2.18 `crc_16_le_bit_swap`

Prototype:

```
unsigned int crc_16_le_bit_swap(unsigned int data, bytesSpecifier_t bspec)
```

Description:

16-bit CCITT CRC computation and bit swap in Little-endian format.

Perform a 16-bit CCITT CRC computation on specified bytes of the data argument that are assumed to be in Little-endian format and return the unmodified value of data. The bits in each byte are swapped before the computation begins.

`crc_write()` can be used to initialize the CRC value and the result can be obtained from `crc_read()`.

Table 3.364. `crc_16_le_bit_swap` parameters

Type	Name	Description
unsigned int	<code>data</code>	Data on which to perform the 16-bit CCITT CRC computation
<code>bytesSpecifier_t</code>	<code>bspec</code>	Specified bytes in the data argument on which to perform the computation

See Also:

- `crc_read()` and `crc_write()`

3.11.2.19 `crc_32_be_bit_swap`

Prototype:

```
unsigned int crc_32_be_bit_swap(unsigned int data, bytesSpecifier_t bspec)
```

Description:

32-bit iSCSI CRC computation and bit swap in Big-endian format.

Perform a 32-bit iSCSI CRC computation on specified bytes of the data argument that are assumed to be in Big-endian format and return the unmodified value of data. The bits in each byte are swapped before the computation begins.

`crc_write()` can be used to initialize the CRC value and the result can be obtained from `crc_read()`.

Table 3.365. `crc_32_be_bit_swap` parameters

Type	Name	Description
unsigned int	<code>data</code>	Data on which to perform the 32-bit iSCSI CRC computation
<code>bytesSpecifier_t</code>	<code>bspec</code>	Specified bytes in the data argument on which to perform the computation

See Also:

- `crc_read()` and `crc_write()`

3.11.2.20 `crc_32_le_bit_swap`

Prototype:

```
unsigned int crc_32_le_bit_swap(unsigned int data, bytesSpecifier_t bspec)
```

Description:

32-bit iSCSI CRC computation and bit swap in Little-endian format.

Perform a 32-bit iSCSI CRC computation on specified bytes of the data argument that are assumed to be in Little-endian format and return the unmodified value of data. The bits in each byte are swapped before the computation begins.

`crc_write()` can be used to initialize the CRC value and the result can be obtained from `crc_read()`.

Table 3.366. `crc_32_le_bit_swap` parameters

Type	Name	Description
unsigned int	<code>data</code>	Data on which to perform the 32-bit iSCSI CRC computation
<code>bytesSpecifier_t</code>	<code>bspec</code>	Specified bytes in the data argument on which to perform the computation

See Also:

- `crc_read()` and `crc_write()`

3.11.2.21 `crc_ccitt_be_bit_swap`

Prototype:

```
unsigned int crc_ccitt_be_bit_swap(unsigned int data, bytesSpecifier_t bspec)
```

Description:

16-bit CCITT CRC computation and bit swap in Big-endian format.

Perform a 16-bit CCITT CRC computation on specified bytes of the data argument that are assumed to be in Big-endian format and return the unmodified value of data. The bits in each byte are swapped before the computation begins.

`crc_write()` can be used to initialize the CRC value and the result can be obtained from `crc_read()`.

Table 3.367. `crc_ccitt_be_bit_swap` parameters

Type	Name	Description
unsigned int	<code>data</code>	Data on which to perform the 16-bit CCITT CRC computation
<code>bytesSpecifier_t</code>	<code>bspec</code>	Specified bytes in the data argument on which to perform the computation

See Also:

- `crc_read()` and `crc_write()`

3.11.2.22 `crc_ccitt_le_bit_swap`

Prototype:

```
unsigned int crc_ccitt_le_bit_swap(unsigned int data, bytesSpecifier_t bspec)
```

Description:

16-bit CCITT CRC computation and bit swap in Little-endian format.

Perform a 16-bit CCITT CRC computation on specified bytes of the data argument that are assumed to be in Little-endian format and return the unmodified value of data. The bits in each byte are swapped before the computation begins.

`crc_write()` can be used to initialize the CRC value and the result can be obtained from `crc_read()`.

Table 3.368. `crc_ccitt_le_bit_swap` parameters

Type	Name	Description
unsigned int	<code>data</code>	Data on which to perform the 16-bit CCITT CRC computation
<code>bytesSpecifier_t</code>	<code>bspec</code>	Specified bytes in the data argument on which to perform the computation

See Also:

- `crc_read()` and `crc_write()`

3.11.2.23 `crc_iscsi_be_bit_swap`

Prototype:

```
unsigned int crc_iscsi_be_bit_swap(unsigned int data, bytesSpecifier_t bspec)
```

Description:

32-bit iSCSI CRC computation and bit swap in Big-endian format.

Perform a 32-bit iSCSI CRC computation on specified bytes of the data argument that are assumed to be in Big-endian format and return the unmodified value of data. The bits in each byte are swapped before the computation begins.

`crc_write()` can be used to initialize the CRC value and the result can be obtained from `crc_read()`.

Table 3.369. `crc_iscsi_be_bit_swap` parameters

Type	Name	Description
unsigned int	<code>data</code>	Data on which to perform the 32-bit iSCSI CRC computation
bytesSpecifier_t	<code>bspec</code>	Specified bytes in the data argument on which to perform the computation

See Also:

- `crc_read()` and `crc_write()`

3.11.2.24 `crc_iscsi_le_bit_swap`

Prototype:

```
unsigned int crc_iscsi_le_bit_swap(unsigned int data, bytesSpecifier_t bspec)
```

Description:

32-bit iSCSI CRC computation and bit swap in Little-endian format.

Perform a 32-bit iSCSI CRC computation on specified bytes of the data argument that are assumed to be in Little-endian format and return the unmodified value of data. The bits in each byte are swapped before the computation begins.

`crc_write()` can be used to initialize the CRC value and the result can be obtained from `crc_read()`.

Table 3.370. `crc_iscsi_le_bit_swap` parameters

Type	Name	Description
unsigned int	<code>data</code>	Data on which to perform the 32-bit iSCSI CRC computation
bytesSpecifier_t	<code>bspec</code>	Specified bytes in the data argument on which to perform the computation

See Also:

- `crc_read()` and `crc_write()`

3.11.2.25 crc_read

Prototype:

```
unsigned int crc_read(void)
```

Description:

Read CRC remainder accumulated so far.

3.11.2.26 crc_write

Prototype:

```
void crc_write(unsigned int residue)
```

Description:

Write the CRC remainder.

Table 3.371. crc_write parameters

Type	Name	Description
unsigned int	residue	Value to initialize the CRC remainder

3.12 Crypto Intrinsics

This section discusses the crypto functions.

3.12.1 Crypto Functions

3.12.1.1 crypto_read

Prototype:

```
void crypto_read(__xread void* data, unsigned int address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read data from Crypto into transfer register.

Table 3.372. crypto_read parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Xfer register(s) to store data read from memory
<code>unsigned int</code>	<code>address</code>	32 bit address in Crypto SRAM to read data from
<code>unsigned int</code>	<code>count</code>	Number of 64-bit words to read (1 - 16)
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.12.1.2 crypto_read_ind

Prototype:

```
void crypto_read_ind(__xread void* data, unsigned int address, unsigned int max_nn,
generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read data from Crypto into DRAM in indirect mode.

Table 3.373. crypto_read_ind parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Xfer register(s) to store data read from memory
<code>unsigned int</code>	<code>address</code>	32 bit address in Crypto SRAM to read data from
<code>unsigned int</code>	<code>max_nn</code>	Maximum number of 64-bit words to read (1 - 16)
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.12.1.3 crypto_write

Prototype:

```
void crypto_write(__xwrite void* data, unsigned int address, unsigned int count, sync_t
sync, SIGNAL* sig_ptr)
```

Description:

Write data to Crypto from MEM.

Table 3.374. crypto_write parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Xfer register(s) containing data to write to memory

Type	Name	Description
unsigned int	<i>address</i>	32 bit address in Crypto SRAM to write data to
unsigned int	<i>count</i>	Number of 64-bit words to write (1 - 16)
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.12.1.4 crypto_write_ind

Prototype:

```
void crypto_write_ind(__xwrite void* data, unsigned int address, unsigned int max_nn,
generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write data to Crypto from DRAM in indirect mode.

Table 3.375. crypto_write_ind parameters

Type	Name	Description
__xwrite void*	<i>data</i>	Xfer register(s) containing data to write to memory
unsigned int	<i>address</i>	32 bit address in Crypto SRAM to write data to
unsigned int	<i>max_nn</i>	Maximum number of 64-bit words to write (1 - 16)
generic_ind_t	<i>ind</i>	Indirect word
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.12.1.5 crypto_write_fifo

Prototype:

```
void crypto_write_fifo(__xwrite void* data, unsigned int address, unsigned int count,
sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write data to Crypto FIFO.

Table 3.376. crypto_write_fifo parameters

Type	Name	Description
__xwrite void*	<i>data</i>	Xfer register(s) containing data (command) to write to FIFO
unsigned int	<i>address</i>	32 bit address in Crypto SRAM to write data to
unsigned int	<i>count</i>	Number of 64-bit words to write (1 - 16)

Type	Name	Description
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.12.1.6 crypto_write_fifo_ind

Prototype:

```
void crypto_write_fifo_ind(__xwrite void* data, unsigned int address, unsigned int max_nn,
                           generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write data to Crypto FIFO from DRAM in indirect mode.

Table 3.377. crypto_write_fifo_ind parameters

Type	Name	Description
__xwrite void*	<i>data</i>	Xfer register(s) containing data (command) to write to FIFO
unsigned int	<i>address</i>	32 bit address in Crypto SRAM to write data to
unsigned int	<i>max_nn</i>	Maximum number of 64-bit words to write
generic_ind_t	<i>ind</i>	Indirect word
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.13 MEM Intrinsics

This section discusses the enumerations, structs, unions, types and functions related to MEM operations.

FIX ME doxy_MEMDefines.xml

FIX ME doxy_MEMTypedefs.xml

FIX ME doxy_MEMFunctions.xml

3.14 MEM WorkQ Intrinsics

This section discusses the enumerations, structs, unions, types and functions related to MEM WorkQ operations.

Work queues enable work to be scheduled among a pool of threads. Threads are added to the work queue using the mem_workq_add_thread() intrinsic. The call will only return when work is available or becomes available.

Work is added to a work queue using mem_workq_add_work_imm() or mem_workq_add_work(). These calls will never block and overflow will occur when more work is added than is consumed.

Work queues are based on type 2 rings and the mem_ring_init() function can be used to configure a new work queue before adding work or threads.

Work can consist of 1 to 16 32-bit words and should be the same size when added as when consumed by mem_workq_add_thread().

Below is an example of workq function with work on ring number 4 in emem1:

```
// ring head and tail on i25 or emem1
__declspec(i25.mem, addr40, aligned(512*sizeof(unsigned int))) unsigned int mem_workq[512];

// place workq on emem1 with ring number 4
unsigned int mu_island = 1;
unsigned int ring_number = (mu_island << 10) | 4;
SIGNAL signal;

mem_ring_init(ring_number, RING_SIZE_512, mem_workq, mem_workq, 0);

// Add work
{
    __declspec(write_reg) unsigned int write_register;
    write_register = 0x01;
    mem_workq_add_work(ring_number, &write_register, 1, sig_done, &signal);
    __wait_for_all(&signal);
}

// Add more work
{
    unsigned int work = 0x123;
    mem_workq_add_work_imm(ring_number, work);
}

{
    __declspec(read_reg) unsigned int read_register;

    // Add thread, should get back 0x01
    mem_workq_add_thread(ring_number, &read_register, 1, ctx_swap, &signal);

    if (read_register != 0x01)
    {
        return 0;          // We have an error
    }

    // Add thread, should get back 0x123
    mem_workq_add_thread(ring_number, &read_register, 1, ctx_swap, &signal);

    if (read_register != 0x123)
    {
        return 0;          // We have an error
    }
}

return 1;
```

}

3.14.1 MU WorkQ Functions

3.14.1.1 mem_workq_add_work

Prototype:

```
void mem_workq_add_work(unsigned int ring_number, __xwrite void* xfer, unsigned int count,  
sync_t sync, SIGNAL* sig_ptr)
```

Description:

Add work specified in transfer register(s) and place on the work queue.



Note

No work queue overflow checking is performed.

Table 3.378. mem_workq_add_work parameters

Type	Name	Description
unsigned int	ring_number	MEM ring number to get data from (0-1023)
__xwrite void*	xfer	Transfer register(s) containing the work
unsigned int	count	Number of 32-bit words specifying the work (1 - 16)
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion.

3.14.1.2 mem_workq_add_work_imm

Prototype:

```
void mem_workq_add_work_imm(unsigned int ring_number, unsigned int work_longword)
```

Description:

Add a single long word describing work on the work queue.



Note

No work queue overflow checking is performed and no signals are used.

Table 3.379. mem_workq_add_work_imm parameters

Type	Name	Description
unsigned int	<i>ring_number</i>	MEM ring number to put get data from (0-1023)
unsigned int	<i>work_longword</i>	32-bit word describing the work

3.14.1.3 mem_workq_add_thread

Prototype:

```
void mem_workq_add_thread(unsigned int ring_number, __xread unsigned int* xfer, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Fetch work from a work queue into transfer register(s).



Note

This call will block until there is work on the queue.

Table 3.380. mem_workq_add_thread parameters

Type	Name	Description
unsigned int	<i>ring_number</i>	MEM ring number to get data from (0-1023)
<u>xread</u> unsigned int*	<i>xfer</i>	Transfer register(s) receiving the work
unsigned int	<i>count</i>	Number of 32-bit words to get (1 - 16)
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.15 MEM Ring Intrinsics

This section discusses the enumerations, structs, unions, types and functions related to MEM Ring operations.

3.15.1 MU Ring Defines

Table 3.381. MU Ring Defines

Defined	Definition
MEM_RING_HEAD(desc)	((desc).head_ptr << 2) (((desc).tail_ptr & ~0xFFFFFFF) << 2)) Return the byte address of the head of a ring for a given descriptor.

Defined	Definition
MEM_RING_TAIL(desc)	((desc).tail_ptr << 2) Return the byte address of the tail of a ring for a given descriptor.

3.15.2 MU Ring Enumerations

3.15.2.1 MEM_RING_SIZE

Ring word size.

Table 3.382. enum MEM_RING_SIZE

Name	Description
RING_SIZE_512	512 Long Words
RING_SIZE_1K	1 K
RING_SIZE_2K	2 K
RING_SIZE_4K	4 K
RING_SIZE_8K	8 K
RING_SIZE_16K	16 K
RING_SIZE_32K	32 K
RING_SIZE_64K	64 K
RING_SIZE_128K	128 K
RING_SIZE_256K	256 K
RING_SIZE_512K	512 K
RING_SIZE_1M	1M
RING_SIZE_2M	2M
RING_SIZE_4M	4M
RING_SIZE_8M	8M
RING_SIZE_16M	16M

3.15.3 MU Ring Structs

3.15.3.1 mem_ring_put_status_t

mem_ring_put return type.

Table 3.383. struct mem_ring_put_status_t

Type	Name	Description
unsigned int	success:1	Bit is set when data was placed on ring.
unsigned int	reserved:7	Reserved.
unsigned int	words_in_ring_before_put:24	Number of words on ring before the operation.

3.15.4 MU Ring Unions

3.15.4.1 mem_ring_desc_t

Push descriptor type.

Table 3.384. union mem_ring_desc_t

Type	Name	Description
unsigned int	ring_size:4	Ring size specification and must be one of MEM_RING_SIZE.
unsigned int	reserved_1:2	Reserved.
unsigned int	head_ptr:24	24 - bit pointer - Byte address of ring buffer. <div style="border: 1px solid blue; padding: 10px; text-align: center;">  Note Must be aligned to the byte size of the ring. </div>
unsigned int	eop:1	This bit is set when an overflow occurred on journal command. <code>mem_ring_journal_buffer, mem_ring_fastjournal_imm</code>
unsigned int	reserved_2:1	Reserved.
unsigned int	tail_ptr:30	Byte address of the tail of the ring.
unsigned int	q_type:2	Is set to 2 for ring intrinsic library functions.
unsigned int	q_loc:2	Locality specification of data.
unsigned int	reserved_3:4	Reserved.
unsigned int	q_page:2	Page of ring data.
unsigned int	q_count:24	Number of words on the ring.
unsigned int	reserved_4	
unsigned int	value[4]	Accessor to entire descriptor structure.

3.15.5 MU Ring Typedefs

3.15.5.1 MEM_RING_SIZE

Ring word size.

Table 3.385. typedef MEM_RING_SIZE

Type	Definition
MEM_RING_SIZE	enum MEM_RING_SIZE

3.15.5.2 mem_ring_desc_t

Push descriptor type.

Table 3.386. typedef mem_ring_desc_t

Type	Definition
mem_ring_desc_t	union mem_ring_desc_t

3.15.5.3 mem_ring_put_status_t

mem_ring_put return type.

Table 3.387. typedef mem_ring_put_status_t

Type	Definition
mem_ring_put_status_t	struct mem_ring_put_status_t

3.15.5.4 mem_ring_desc_in_mem_t

Descriptor type definition to ensure correct alignment in MEM.

Table 3.388. typedef mem_ring_desc_in_mem_t

Type	Definition
mem_ring_desc_in_mem_t	<code>__addr40 __align16 mem_ring_desc_t</code>

3.15.5.5 mem_ring_put_status_in_read_reg_t

mem_ring_put_status_t in a read register

Table 3.389. `typedef mem_ring_put_status_in_read_reg_t`

Type	Definition
<code>mem_ring_put_status_in_read_reg_t</code>	<code>__xread mem_ring_put_status_t</code>

3.15.6 MU Ring Functions

3.15.6.1 `mem_ring_read_desc`

Prototype:

```
void mem_ring_read_desc(unsigned int ring_number, mem_ring_desc_in_mem_t*
mem_ring_desc_in_mem)
```

Description:

Read MEM queue descriptor from MEM into queue array.

Table 3.390. `mem_ring_read_desc` parameters

Type	Name	Description
unsigned int	<i>ring_number</i>	Ring descriptor number to load. The range of the ring number identifies the external memory to use. <ul style="list-style-type: none"> • 0 - 1023: emem0 or island 24, • 1024 - 2047: emem1 or island 25, • 2048 - 3071: emem2 or island 26
<code>mem_ring_desc_in_mem_t*</code>	<i>mem_ring_desc_in_mem</i>	40 bit pointer to the Ring Descriptor in external memory including the MU island. The pointer must be 16 byte aligned.

3.15.6.2 `mem_ring_write_desc`

Prototype:

```
void mem_ring_write_desc(unsigned int ring_number, mem_ring_desc_in_mem_t*
mem_ring_desc_in_mem)
```

Description:

Write MEM queue descriptor from queue array into MEM.

Table 3.391. mem_ring_write_desc parameters

Type	Name	Description
unsigned int	<i>ring_number</i>	Ring descriptor to write. The range of the ring number identifies the external memory to use. <ul style="list-style-type: none"> • 0 - 1023: emem0 or island 24, • 1024 - 2047: emem1 or island 25, • 2048 - 3071: emem2 or island 26
mem_ring_desc_in_mem_t*	<i>mem_ring_desc_in_mem</i>	40 bit pointer where to place ring descriptor in external memory including the MU island. The pointer must be 16 byte aligned.

3.15.6.3 mem_ring_push_desc

Prototype:

```
void mem_ring_push_desc(unsigned int ring_number, __xread void* xfer, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Push MEM queue descriptor from queue array into transfer registers.

Table 3.392. mem_ring_push_desc parameters

Type	Name	Description
unsigned int	<i>ring_number</i>	Ring number descriptor to write. The range of the ring number identifies the external memory to use. <ul style="list-style-type: none"> • 0 - 1023: emem0 or island 24, • 1024 - 2047: emem1 or island 25, • 2048 - 3071: emem2 or island 26
__xread void*	<i>xfer</i>	Pointer to read xfer registers where the descriptor will be written
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.15.6.4 mem_ring_read_buffer_unbounded

Prototype:

```
void mem_ring_read_buffer_unbounded(unsigned int ring_number, __xread unsigned int* xfer,
                                    unsigned int count, unsigned int offset, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read data from MEM ring at an offset from the base and place in transfer register(s) without checking for ring bounds.

Table 3.393. mem_ring_read_buffer_unbounded parameters

Type	Name	Description
unsigned int	<i>ring_number</i>	MEM ring number to get data from. The range of the ring number identifies the external memory to use. <ul style="list-style-type: none"> • 0 - 1023: emem0 or island 24, • 1024 - 2047: emem1 or island 25, • 2048 - 3071: emem2 or island 26
<u>__xread</u> unsigned int*	<i>xfer</i>	Transfer register(s) receiving data
unsigned int	<i>count</i>	Number of 4-byte long words to get (1-16)
unsigned int	<i>offset</i>	Long word count offset from base of ring
sync_t	<i>sync</i>	Type of synchronization to use
SIGNAL*	<i>sig_ptr</i>	Signal to raised upon completion

3.15.6.5 mem_ring_read_buffer

Prototype:

```
void mem_ring_read_buffer(unsigned int ring_number, __xread unsigned int* xfer, unsigned int count, unsigned int offset, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read data from MEM ring at an offset from the base and place in transfer register(s).

Table 3.394. mem_ring_read_buffer parameters

Type	Name	Description
unsigned int	<i>ring_number</i>	MEM ring number to get data from. The range of the ring number identifies the external memory to use. <ul style="list-style-type: none"> • 0 - 1023: emem0 or island 24, • 1024 - 2047: emem1 or island 25, • 2048 - 3071: emem2 or island 26
<u>__xread</u> unsigned int*	<i>xfer</i>	Transfer register(s) receiving data
unsigned int	<i>count</i>	Number of 4-byte long words to get (1-16)
unsigned int	<i>offset</i>	Long word count offset from base of ring
sync_t	<i>sync</i>	Type of synchronization to use

Type	Name	Description
SIGNAL*	<i>sig_ptr</i>	Signal to raised upon completion

3.15.6.6 mem_ring_write_buffer_unbounded

Prototype:

```
void mem_ring_write_buffer_unbounded(unsigned int ring_number, __xwrite void* xfer,
unsigned int count, unsigned int offset, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write data at an offset in a ring without boundary checking.

This operation is used to access memory within the bounds of the maximum ring size of the queue engine.

Table 3.395. mem_ring_write_buffer_unbounded parameters

Type	Name	Description
unsigned int	<i>ring_number</i>	Ring number. The range of the ring number identifies the external memory to use. <ul style="list-style-type: none"> • 0 - 1023: emem0 or island 24, • 1024 - 2047: emem1 or island 25, • 2048 - 3071: emem2 or island 26
<u>xwrite</u> void*	<i>xfer</i>	Pointer to write xfer register(s) containing the data to write
unsigned int	<i>count</i>	Number of 32-bit word to put on the ring (1-16)
unsigned int	<i>offset</i>	Long word count offset from base of 16M word ring
sync_t	<i>sync</i>	Type of synchronization to use
SIGNAL*	<i>sig_ptr</i>	Signal to raised upon completion

3.15.6.7 mem_ring_write_buffer

Prototype:

```
void mem_ring_write_buffer(unsigned int ring_number, __xwrite void* xfer, unsigned int
count, unsigned int offset, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write data at an offset into a ring.

Table 3.396. mem_ring_write_buffer parameters

Type	Name	Description
unsigned int	<i>ring_number</i>	Ring number. The range of the ring number identifies the external memory to use. <ul style="list-style-type: none"> • 0 - 1023: emem0 or island 24, • 1024 - 2047: emem1 or island 25, • 2048 - 3071: emem2 or island 26
__xwrite void*	<i>xfer</i>	Pointer to write xfer register(s) containing the data to write
unsigned int	<i>count</i>	Number of 32-bit word to put on the ring (1-16)
unsigned int	<i>offset</i>	Long word count offset from base of ring
sync_t	<i>sync</i>	Type of synchronization to use
SIGNAL*	<i>sig_ptr</i>	Signal to raised upon completion

3.15.6.8 mem_ring_add_tail

Prototype:

```
void mem_ring_add_tail(unsigned int ring_number, unsigned int count)
```

Description:

Add a value to the tail of a ring wrapping it appropriately.

Table 3.397. mem_ring_add_tail parameters

Type	Name	Description
unsigned int	<i>ring_number</i>	Ring number descriptor to write. The range of the ring number identifies the external memory to use. <ul style="list-style-type: none"> • 0 - 1023: emem0 or island 24, • 1024 - 2047: emem1 or island 25, • 2048 - 3071: emem2 or island 26
unsigned int	<i>count</i>	24 bit count to add to the tail

3.15.6.9 mem_ring_put_buffer

Prototype:

```
mem_ring_put_status_in_read_reg_t* mem_ring_put_buffer(unsigned int ring_number, __xrw
void* xfer, unsigned int count, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Add data to the tail of a ring checking for overflow.

Table 3.398. mem_ring_put_buffer parameters

Type	Name	Description
unsigned int	<i>ring_number</i>	Ring number. The range of the ring number identifies the external memory to use. <ul style="list-style-type: none">• 0 - 1023: emem0 or island 24,• 1024 - 2047: emem1 or island 25,• 2048 - 3071: emem2 or island 26
<u>_xrw void*</u>	<i>xfer</i>	Pointer to read_write xfer register(s) containing the data to put on the ring
unsigned int	<i>count</i>	Number of 32-bit word to put on the ring (1-16)
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to read transfer register containing the status of the operation

3.15.6.10 mem_ring_journal_buffer

Prototype:

```
void mem_ring_journal_buffer(unsigned int ring_number, __xwrite void* xfer, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Add data to the tail of a ring without checking for overflow.

Table 3.399. mem_ring_journal_buffer parameters

Type	Name	Description
unsigned int	<i>ring_number</i>	Ring number. The range of the ring number identifies the external memory to use. <ul style="list-style-type: none">• 0 - 1023: emem0 or island 24,• 1024 - 2047: emem1 or island 25,• 2048 - 3071: emem2 or island 26
<u>_xwrite void*</u>	<i>xfer</i>	Pointer to write xfer register(s) containing the data to put on the ring
unsigned int	<i>count</i>	Number of 32-bit word to put on the ring (1-16)
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

Returns:

Read transfer register containing the status of the operation

3.15.6.11 mem_ring_get_buffer

Prototype:

```
void mem_ring_get_buffer(unsigned int ring_number, __xread unsigned int* xfer, unsigned
int count, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Fetch data from the head of MEM ring and place in transfer register(s).

Table 3.400. mem_ring_get_buffer parameters

Type	Name	Description
unsigned int	<i>ring_number</i>	MEM ring number to get data from. The range of the ring number identifies the external memory to use. <ul style="list-style-type: none"> • 0 - 1023: emem0 or island 24, • 1024 - 2047: emem1 or island 25, • 2048 - 3071: emem2 or island 26
<u>xread</u> unsigned int*	<i>xfer</i>	Transfer register(s) receiving data
unsigned int	<i>count</i>	Number of 32-bit words to get (1-16)
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Even signal is raised upon completion. Odd signal is raised on underflow.

3.15.6.12 mem_ring_get_buffer_eop

Prototype:

```
void mem_ring_get_buffer_eop(unsigned int ring_number, __xread unsigned int* xfer, unsigned
int count, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Fetch data from the head of MEM ring, place in transfer register(s) and check for journal overflow.

The EOP bit is cleared after this command

Table 3.401. mem_ring_get_buffer_eop parameters

Type	Name	Description
unsigned int	<i>ring_number</i>	MEM ring number to get data from. The range of the ring number identifies the external memory to use. <ul style="list-style-type: none"> • 0 - 1023: emem0 or island 24, • 1024 - 2047: emem1 or island 25, • 2048 - 3071: emem2 or island 26
<u>xread</u> unsigned int*	<i>xfer</i>	Transfer register(s) receiving data
unsigned int	<i>count</i>	Number of 32-bit words to get (1-16)
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Even signal is raised upon completion. Odd signal is raised on underflow or when overflow occurred during journal operation

3.15.6.13 mem_ring_get_buffer_freely

Prototype:

```
void mem_ring_get_buffer_freely(unsigned int ring_number, __xread unsigned int* xfer,
unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Fetch data from the head of MEM ring and place in transfer register(s).

If not enough entries to return, do not return an error signal but return 0 for the remaining length after valid entries have been removed.

The following example shows how to create a ring in emem2 (i26). Note the setup of the ring_number to indicate that the ring is in emem2.

```
// ring head and tail on i26 or emem2
__emem_n(2) __addr40 __align(512*sizeof(unsigned int)) unsigned int ring_mem[512];
unsigned int          mu_island = 2;
unsigned int          ring_number = (mu_island << 10) | 2; // placing ring on emem2
SIGNAL_PAIR           sig_pair;
SIGNAL                sig;

mem_ring_init(ring_number, RING_SIZE_512, ring_mem, ring_mem, 0);

// Put some words on ring
{
    __xrw unsigned int          data[3];
    __xread mem_ring_put_status_t *ring_put_status;
    __xread unsigned int          status;

    data[0] = 0x12345678;
    data[1] = 0x87654321;
```

```

data[2] = 0x10101010;

ring_put_status = mem_ring_put_buffer(ring_number, data, 3, sig_done, &sig_pair);
wait_for_all(&sig_pair);

if (!ring_put_status->success)
{
    return 0;           // We have an error
}
if (ring_put_status->words_in_ring_before_put != 0)
{
    return 0;           // We have an error
}
}

// Remove 4 entries from the head of the ring. The ring only contains 3 entries.
// The last xfer(entry) returned will be 0.
{
    __xread unsigned int      data_get[4];

    mem_ring_get_buffer_freely(ring_number, data_get, 4, ctx_swap, &sig);

    // First entry
    if (data_get[0] != 0x12345678)
    {
        return 0;           // We have an error
    }
    // Last xfer should be 0
    if (data_get[3] != 0x00)
    {
        return 0;           // We have an error
    }
}
}

```

Table 3.402. mem_ring_get_buffer_freely parameters

Type	Name	Description
unsigned int	<i>ring_number</i>	MEM ring number to get data from. The range of the ring number identifies the external memory to use. <ul style="list-style-type: none"> • 0 - 1023: emem0 or island 24, • 1024 - 2047: emem1 or island 25, • 2048 - 3071: emem2 or island 26
<i>__xread unsigned int*</i>	<i>xfer</i>	Transfer register(s) receiving data
unsigned int	<i>count</i>	Number of 32-bit words to get (1-16)
<i>sync_t</i>	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.15.6.14 mem_ring_pop_buffer

Prototype:

```
void mem_ring_pop_buffer(unsigned int ring_number, __xread unsigned int* xfer, unsigned
int count, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Remove data from the tail of MEM ring and place in transfer register(s).

If there are not enough entries in the ring, no valid data is returned and the tail pointer is not updated. An odd signal will also be delivered.

Note that a single pop operation will return the words in the order they were added. If the ring contains the words (A, B, C, D, E), then a single pop operation of 3 words will return the values (C, D, E) in that order.

Table 3.403. mem_ring_pop_buffer parameters

Type	Name	Description
unsigned int	<i>ring_number</i>	MEM ring number to get data from. The range of the ring number identifies the external memory to use. <ul style="list-style-type: none"> • 0 - 1023: emem0 or island 24, • 1024 - 2047: emem1 or island 25, • 2048 - 3071: emem2 or island 26
<u>xread</u> unsigned int*	<i>xfer</i>	Transfer register(s) receiving data
unsigned int	<i>count</i>	Number of 32-bit words to get (1-16)
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Even signal is raised upon completion. Odd signal is raised on underflow.

3.15.6.15 mem_ring_pop_buffer_freely

Prototype:

```
void mem_ring_pop_buffer_freely(unsigned int ring_number, __xread unsigned int* xfer,
unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Fetch data from the tail of MEM ring and place in transfer register(s).

If not enough entries to return, do not return an error signal but return 0 for the remaining length after valid entries have been removed.

Table 3.404. mem_ring_pop_buffer_freely parameters

Type	Name	Description
unsigned int	<i>ring_number</i>	MEM ring number to get data from. The range of the ring number identifies the external memory to use. <ul style="list-style-type: none"> • 0 - 1023: emem0 or island 24, • 1024 - 2047: emem1 or island 25, • 2048 - 3071: emem2 or island 26
<u>xread</u> unsigned int*	<i>xfer</i>	Transfer register(s) receiving data
unsigned int	<i>count</i>	Number of 32-bit words to get (1-16)
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.15.6.16 mem_ring_pop_buffer_eop

Prototype:

```
void mem_ring_pop_buffer_eop(unsigned int ring_number, xread unsigned int* xfer, unsigned int count, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Fetch data from the tail of MEM ring, place in transfer register(s) and check for journal overflow.

Table 3.405. mem_ring_pop_buffer_eop parameters

Type	Name	Description
unsigned int	<i>ring_number</i>	MEM ring number to get data from. The range of the ring number identifies the external memory to use. <ul style="list-style-type: none"> • 0 - 1023: emem0 or island 24, • 1024 - 2047: emem1 or island 25, • 2048 - 3071: emem2 or island 26
<u>xread</u> unsigned int*	<i>xfer</i>	Transfer register(s) receiving data
unsigned int	<i>count</i>	Number of 32-bit words to get (1-16)
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Even signal is raised upon completion. Odd signal is raised on underflow or when the EOP bit is set in the descriptor.

3.15.6.17 mem_ring_fastjournal_imm

Prototype:

```
void mem_ring_fastjournal_imm(unsigned int ring_number, unsigned int longword)
```

Description:

FastJournal / Put Freely immediate data to the tail of a ring without checking for overflow.

Note

This function does not return a signal and the ME will automatically be throttled when it generates too much data for the queue engine.

Table 3.406. mem_ring_fastjournal_imm parameters

Type	Name	Description
unsigned int	<i>ring_number</i>	Ring number. The range of the ring number identifies the external memory to use. <ul style="list-style-type: none"> • 0 - 1023: emem0 or island 24, • 1024 - 2047: emem1 or island 25, • 2048 - 3071: emem2 or island 26
unsigned int	<i>longword</i>	32-bit word to put on the ring

3.15.6.18 mem_ring_init

Prototype:

```
void mem_ring_init(unsigned int ring_number, enum MEM_RING_SIZE size, __addr40 __mem
unsigned int* head_ptr, __addr40 __mem unsigned int* tail_ptr, unsigned int count)
```

Description:

Helper function to initialize a ring.

This function configures the ring to be a type 2 in the queue engine

The following example shows how to create a ring in emem2 (i26). Note the setup of the *ring_number* to indicate that the ring is in emem2.

```
// ring head and tail on i26 or emem2
__emem_n(2) __addr40 __align(512*sizeof(unsigned int)) unsigned int ring_mem[512];
unsigned int mu_island = 2;
unsigned int ring_number = (mu_island << 10) | 2;      // placing ring on emem2
SIGNAL_PAIR sig_pair;
SIGNAL sig;
```

```
mem_ring_init(ring_number, RING_SIZE_512, ring_mem, ring_mem, 0);

// Put some words on ring
{
    __xrw unsigned int          data[3];
    __xread mem_ring_put_status_t      *ring_put_status;
    __xread unsigned int          status;

    data[0] = 0x12345678;
    data[1] = 0x87654321;
    data[2] = 0x10101010;

    ring_put_status = mem_ring_put_buffer(ring_number, data, 3, sig_done, &sig_pair);
    wait_for_all(&sig_pair);

    if (!ring_put_status->success)
    {
        return 0;           // We have an error
    }
    if (ring_put_status->words_in_ring_before_put != 0)
    {
        return 0;           // We have an error
    }
}

// Verify count
{
    __xread mem_ring_desc_t descriptor;
    mem_ring_push_desc(ring_number, &descriptor, ctx_swap, &sig);

    if (descriptor.q_count != 3)
    {
        return 0;           // We have an error
    }
}

// Get buffer from head of ring
{
    __xread unsigned int          data_get;

    mem_ring_get_buffer(ring_number, &data_get, 1, sig_done, &sig_pair);
    wait_for_all_single(&sig_pair.even);

    if (data_get != 0x12345678)
    {
        return 0;           // We have an error
    }
}

// Pop buffer from tail of ring
{
    __xread unsigned int          data_pop;

    mem_ring_pop_buffer(ring_number, &data_pop, 1, sig_done, &sig_pair);
    wait_for_all_single(&sig_pair.even);

    if (data_pop != 0x10101010)
    {
```

```

        return 0;           // We have an error
    }
}
return 1;

```

Table 3.407. mem_ring_init parameters

Type	Name	Description
unsigned int	<i>ring_number</i>	Ring number. The range of the ring number identifies the external memory to use. <ul style="list-style-type: none"> • 0 - 1023: emem0 or island 24, • 1024 - 2047: emem1 or island 25, • 2048 - 3071: emem2 or island 26
enum MEM_RING_SIZE	<i>size</i>	The ring size specified as one of enum MEM_RING_SIZE
<u>addr40 __mem</u> unsigned int*	<i>head_ptr</i>	Pointer to the head of the ring. Must be aligned to byte size of ring.
<u>addr40 __mem</u> unsigned int*	<i>tail_ptr</i>	Initial pointer to the tail of the ring. For an empty ring this is set to the head.
unsigned int	<i>count</i>	Initial number of words on the ring. For an empty ring this must be zero.

3.15.6.19 mem_ring_init_with_loc_and_page

Prototype:

```
void mem_ring_init_with_loc_and_page(unsigned int ring_number, enum MEM_RING_SIZE size,
addr40 __mem unsigned int* head_ptr, addr40 __mem unsigned int* tail_ptr, unsigned
int count, unsigned int q_locality, unsigned int q_page)
```

Description:

Helper function to initialize a ring with a locality and page specification.

This function uses a type 2 queue for the ring.

The byte address of the head is given by {locality[1:0] 4b0 page[1:0] tail[29:24] head[23:0] 2b0} and the tail {locality[1:0] 4b0 page[1:0] tail[29:0] 2b0}

Table 3.408. mem_ring_init_with_loc_and_page parameters

Type	Name	Description
unsigned int	<i>ring_number</i>	Ring number. The range of the ring number identifies the external memory to use. <ul style="list-style-type: none"> • 0 - 1023: emem0 or island 24, • 1024 - 2047: emem1 or island 25, • 2048 - 3071: emem2 or island 26

Type	Name	Description
enum MEM_RING_SIZE	<i>size</i>	The ring size specified as one of enum MEM_RING_SIZE
<u>__addr40 __mem unsigned int*</u>	<i>head_ptr</i>	Pointer to the head of the ring. Must be aligned to byte size of ring.
<u>__addr40 __mem unsigned int*</u>	<i>tail_ptr</i>	Initial pointer to the tail of the ring. For an empty ring this is set to the head.
unsigned int	<i>count</i>	Initial number of words on the ring. For an empty ring this must be zero.
unsigned int	<i>q_locality</i>	Locality of the ring. Please refer to Ring/Circular buffer section in PRM for detail.
unsigned int	<i>q_page</i>	4 Gigabyte page of ring

3.16 MEM Ticket Intrinsics

This section discusses the enumerations, structs, unions, types and functions related to MEM Ticket operations.

3.16.1 MU Ticket Structs

3.16.1.1 mem_ticket_line_push_t

Ticket line push.

The ticket line is 128 bit memory structure managing the released tickets. It differs from Ticket line in that the top 22 bits encodes a remote data master and data reference to push the ticket release result to.

Table 3.409. struct mem_ticket_line_push_t

Type	Name	Description
unsigned int	master_cluster:4	Cluster ID of ME that should receive the result.
unsigned int	master_is_me:1	Bit indicating if the result receiver is a ME. This bit should be set to 1.
unsigned int	master_id:3	Microengine ID that should receive the result.
unsigned int	dataref_type:3	Data reference type. 0x0 - Transfer Register address 0x3 - Next Neighbor register address 0x2 - CSR address. In this case dataref_address[10] should be set to 0
unsigned int	dataref_address:11	Register or CSR address.
unsigned int	current_ticket:10	Current ticket number released.
unsigned int	bitmap0:32	Bitmap of tickets presented for release.

Type	Name	Description
		Current ticket+1 to current ticket+31.
unsigned int	bitmap1:32	Current ticket+32 to current ticket+63.
unsigned int	bitmap2:32	Current ticket+64 to current ticket+95.

3.16.1.2 mem_ticket_line_t

Ticket line.

The ticket line is 128 bit memory structure managing the released tickets.

Table 3.410. struct mem_ticket_line_t

Type	Name	Description
unsigned int	reserved:22	Reserved.
unsigned int	current_ticket:10	Current ticket number released.
unsigned int	bitmap0:32	Bitmap of tickets presented for release. Current ticket+1 to current ticket+31.
unsigned int	bitmap1:32	Current ticket+32 to current ticket+63.
unsigned int	bitmap2:32	Current ticket+64 to current ticket+95.

3.16.2 MU Ticket Typedefs

3.16.2.1 mem_ticket_line_t

Ticket line.

The ticket line is 128 bit memory structure managing the released tickets.

Table 3.411. typedef mem_ticket_line_t

Type	Definition
mem_ticket_line_t	struct mem_ticket_line_t

3.16.2.2 mem_ticket_line_push_t

Ticket line push.

The ticket line is 128 bit memory structure managing the released tickets. It differs from Ticket line in that the top 22 bits encodes a remote data master and data reference to push the ticket release result to.

Table 3.412. `typedef mem_ticket_line_push_t`

Type	Definition
<code>mem_ticket_line_push_t</code>	<code>struct mem_ticket_line_push_t</code>

3.16.3 MU Ticket Functions

3.16.3.1 `mem_ticket_release_ptr32`

Prototype:

```
void mem_ticket_release_ptr32(__xrw void* xfer, volatile void __addr32 __mem* address,
sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Release the supplied ticket number from the ticket line at a 32-bit addressed address.

Returns (via xfer) the number of consecutive released tickets or MEM_TICKET_RELEASE_ERROR.

Table 3.413. `mem_ticket_release_ptr32` parameters

Type	Name	Description
<code>__xrw void*</code>	<code>xfer</code>	[In] Ticket number to release [Out] Release result
<code>volatile void __addr32 __mem*</code>	<code>address</code>	MEM address where the 128-bit ticket line starts
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be sig_done)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal to raise upon completion

3.16.3.2 `mem_ticket_release_ptr40`

Prototype:

```
void mem_ticket_release_ptr40(__xrw void* xfer, volatile void __addr40 __mem* address,
sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Release the supplied ticket number from the ticket line at a 40-bit address.

Returns (via xfer) the number of consecutive released tickets or MEM_TICKET_RELEASE_ERROR.

Table 3.414. mem_ticket_release_ptr40 parameters

Type	Name	Description
<code>__xrw void*</code>	<code>xfer</code>	[In] Ticket number to release [Out] Release result
<code>volatile void __addr40 __mem*</code>	<code>address</code>	40-bit MEM address where the 128-bit ticket line starts
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be <code>sig_done</code>)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal to raise upon completion

3.16.3.3 mem_ticket_release_push_ptr32

Prototype:

```
void mem_ticket_release_push_ptr32(__xrw void* xfer, volatile void __addr32 __mem* address,
sync_t sync, SIGNAL* sig_ptr)
```

Description:

Release the supplied ticket number from the ticket line at a 32-bit addressed address.

The result is 'pushed' to the data master and reference setup in the `mem_ticket_line_t` structure pasted to `mem_init_ticket_line`.



Note

No signal is delivered to the data master and one would typically push to a NN ring (`dataref_type` = 0x3) to prevent possible data loss.

Table 3.415. mem_ticket_release_push_ptr32 parameters

Type	Name	Description
<code>__xrw void*</code>	<code>xfer</code>	[In] Ticket number to release
<code>volatile void __addr32 __mem*</code>	<code>address</code>	MEM address where the 128-bit ticket line starts
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be <code>sig_done</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise in caller upon completion

3.16.3.4 mem_ticket_release_push_ptr40

Prototype:

```
void mem_ticket_release_push_ptr40(__xrw void* xfer, volatile void __addr40 __mem* address,
sync_t sync, SIGNAL* sig_ptr)
```

Description:

Release the supplied ticket number from the ticket line at a 40-bit address.

The result is 'pushed' to the data master and reference setup in the `mem_ticket_line_t` structure pasted to `mem_init_ticket_line`.



Note

No signal is delivered to the data master and one would typically push to a NN ring (dataref_type = 0x3) to prevent possible data loss.

Table 3.416. mem_ticket_release_push_ptr40 parameters

Type	Name	Description
<code>__xrw void*</code>	<code>xfer</code>	[In] Ticket number to release
<code>volatile void __addr40 __mem*</code>	<code>address</code>	40-bit MEM address where the 128-bit ticket line starts
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be sig_done)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise in caller upon completion

3.16.3.5 mem_ticket_line_init_ptr32

Prototype:

```
void mem_ticket_line_init_ptr32(mem_ticket_line_t* mem_ticket_line, volatile void __addr32 __mem* address, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Helper function for initializing the ticket release functionality in 32-bit addressed MEM.

Table 3.417. mem_ticket_line_init_ptr32 parameters

Type	Name	Description
<code>mem_ticket_line_t*</code>	<code>mem_ticket_line</code>	Pre-populated <code>mem_ticket_line_t</code> struct to initialize MEM with
<code>volatile void __addr32 __mem*</code>	<code>address</code>	MEM address at which the 128-bit ticket structure should start
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be sig_done)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.16.3.6 mem_ticket_line_init_ptr40

Prototype:

```
void mem_ticket_line_init_ptr40(mem_ticket_line_t* mem_ticket_line, volatile void __addr40 __mem* address, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Helper function for initializing the ticket release functionality in 40-bit addressed MEM.

Table 3.418. mem_ticket_line_init_ptr40 parameters

Type	Name	Description
mem_ticket_line_t*	<i>mem_ticket_line</i>	Pre-populated mem_ticket_line_t struct to initialize MEM with
volatile void __addr40 __mem*	<i>address</i>	40-bit MEM address at which the 128-bit ticket structure should start
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.16.3.7 mem_ticket_line_push_init_ptr32

Prototype:

```
void mem_ticket_line_push_init_ptr32(mem_ticket_line_push_t* mem_ticket_line_push, volatile
void __addr32 __mem* address, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Helper function for initializing the ticket release functionality in 32-bit addressed MEM.

This variation enables the ability to push the result elsewhere.

Table 3.419. mem_ticket_line_push_init_ptr32 parameters

Type	Name	Description
mem_ticket_line_push_t*	<i>mem_ticket_line_push</i>	Pre-populated mem_ticket_line_push_t struct to initialize MEM with
volatile void __addr32 __mem*	<i>address</i>	MEM address at which the 128-bit ticket structure should start
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.16.3.8 mem_ticket_line_push_init_ptr40

Prototype:

```
void mem_ticket_line_push_init_ptr40(mem_ticket_line_push_t* mem_ticket_line_push, volatile
void __addr40 __mem* address, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Helper function for initializing the ticket release functionality in 40-bit addressed MEM.

This variation enables the ability to push the result elsewhere.

Table 3.420. mem_ticket_line_push_init_ptr40 parameters

Type	Name	Description
mem_ticket_line_push_t *	<i>mem_ticket_line_push</i>	Pre-populated mem_ticket_line_push_t struct to initialize MEM with
volatile void __addr40 __mem*	<i>address</i>	40-bit MEM address at which the 128-bit ticket structure should start
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.17 MEM Queue Lock Intrinsics

This section discusses the enumerations, structs, unions, types and functions related to MEM Queue Lock (LockQ) operations.

Lock queues maintain a queue of lock requests. A lock is acquired with mem_lockq128/256_lock(). Subsequent calls by other threads/MEs to mem_lockq128/256_lock() will block (not get signalled) until the lock is released with mem_lockq128/256_unlock() by the lock holder.

On queue overflow an odd signal is returned by mem_lockq128/256_lock().



Note

Please refer to the section on MEM QueueLock Commands in the Databook for more detail.

3.17.1 MU Queue Lock Structs

3.17.1.1 mem_lockq128_t

128 bit lock queue container type.

The first word is the queue descriptor

Table 3.421. struct mem_lockq128_t

Type	Name	Description
mem_lockq_desc_t	desc	Lock queue descriptor.
mem_lockq_entry_t	q[7]	3 words for queue contents

See Also:

- `mem_lockq_desc_t`.

3.17.1.2 `mem_lockq256_t`

256 bit lock queue container type.

The first word is the queue descriptor

Table 3.422. struct `mem_lockq256_t`

Type	Name	Description
<code>mem_lockq_desc_t</code>	<code>desc</code>	Lock queue descriptor.
<code>mem_lockq_entry_t</code>	<code>q[15]</code>	7 words for queue contents

See Also:

- `mem_lockq_desc_t`.

3.17.2 MU Queue Lock Unions

3.17.2.1 `mem_lockq_desc_t`

Lock queue descriptor.

Table 3.423. union `mem_lockq_desc_t`

Type	Name	Description
unsigned int	<code>reserved_1:4</code>	Reserved.
unsigned int	<code>event_source:12</code>	Event source number to include in event.
unsigned int	<code>reserved_2:6</code>	Reserved.
unsigned int	<code>generate_event:1</code>	Generate event on under or overflow.
unsigned int	<code>entry_size:1</code>	Entry size - must be 0 to be 16 bits.
unsigned int	<code>overflow:1</code>	Set on overflow.
unsigned int	<code>reserved_3:2</code>	Reserved.
unsigned int	<code>locked:1</code>	Set when the lock has been given out.
unsigned int	<code>num_elements:4</code>	0-6 or 0-14 depending on if it is a 128 or 256 bit queue.
unsigned int	<code>value</code>	Accessor to entire structure.

3.17.2.2 `mem_lockq_entry_t`

Lock queue entries.

This is an internal structure and no user manipulation is needed for the use of lock queues.

Table 3.424. union mem_lockq_entry_t

Type	Name	Description
unsigned short	signal_ref:7	Signal Reference.
unsigned short	reserved:1	Reserved.
unsigned short	signal_master:8	Signal Master.
unsigned short	value	Accessor to entire structure.

3.17.3 MU Queue Lock Typedefs

3.17.3.1 mem_lockq_desc_t

Lock queue descriptor.

Table 3.425. typedef mem_lockq_desc_t

Type	Definition
mem_lockq_desc_t	union mem_lockq_desc_t

3.17.3.2 mem_lockq_entry_t

Lock queue entries.

This is an internal structure and no user manipulation is needed for the use of lock queues.

Table 3.426. typedef mem_lockq_entry_t

Type	Definition
mem_lockq_entry_t	union mem_lockq_entry_t

3.17.3.3 mem_lockq128_t

128 bit lock queue container type.

The first word is the queue descriptor

Table 3.427. typedef mem_lockq128_t

Type	Definition
mem_lockq128_t	struct mem_lockq128_t

See Also:

- `mem_lockq_desc_t`.

3.17.3.4 mem_lockq256_t

256 bit lock queue container type.

The first word is the queue descriptor

Table 3.428. typedef mem_lockq256_t

Type	Definition
mem_lockq256_t	struct mem_lockq256_t

See Also:

- mem_lockq_desc_t.

3.17.3.5 mem_lockq128_in_mem_t

128 bit lock queue aligned in 32-bit addressed MEM.

Table 3.429. typedef mem_lockq128_in_mem_t

Type	Definition
mem_lockq128_in_mem_t	volatile __addr32 __align16 mem_lockq128_t

3.17.3.6 mem_lockq128_ptr40_t

128 bit lock queue pointer aligned in 40-bit addressed MEM.

Table 3.430. typedef mem_lockq128_ptr40_t

Type	Definition
mem_lockq128_ptr40_t	volatile __mem __addr40 __align16 mem_lockq128_t*

3.17.3.7 mem_lockq256_in_mem_t

256 bit lock queue aligned in 32-bit addressed MEM.

Table 3.431. typedef mem_lockq256_in_mem_t

Type	Definition
mem_lockq256_in_mem_t	volatile __addr32 __align16 mem_lockq256_t

3.17.3.8 mem_lockq256_ptr40_t

256 bit lock queue pointer aligned in 40-bit addressed MEM.

Table 3.432. `typedef mem_lockq256_ptr40_t`

Type	Definition
<code>mem_lockq256_ptr40_t</code>	<code>volatile __mem __addr40 __align16 mem_lockq256_t*</code>

3.17.4 MU Queue Lock Functions

3.17.4.1 `mem_lockq128_lock_ptr32`

Prototype:

```
void mem_lockq128_lock_ptr32(__mem mem_lockq128_in_mem_t* lockq, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Claim a lock in a 128-bit lock queue in 32-bit addressed MEM.

Table 3.433. `mem_lockq128_lock_ptr32` parameters

Type	Name	Description
<code>__mem mem_lockq128_in_mem_t*</code>	<code>lockq</code>	Pointer to lock queue in MEM
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use. Must be <code>sig_done</code> .
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Even signal is raised upon completion. Odd signal is raised on overflow.

3.17.4.2 `mem_lockq128_lock_ind_ptr32`

Prototype:

```
void mem_lockq128_lock_ind_ptr32(__mem mem_lockq128_in_mem_t* lockq, generic_ind_t ind, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Claim a lock in a 128-bit lock queue in 32-bit addressed MEM with indirect word.

Table 3.434. `mem_lockq128_lock_ind_ptr32` parameters

Type	Name	Description
<code>__mem mem_lockq128_in_mem_t*</code>	<code>lockq</code>	Pointer to lock queue in MEM
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use. Must be <code>sig_done</code> .

Type	Name	Description
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Even signal is raised upon completion. Odd signal is raised on overflow.

3.17.4.3 mem_lockq128_lock_ptr40

Prototype:

```
void mem_lockq128_lock_ptr40(mem_lockq128_ptr40_t lockq, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Claim a lock in a 128-bit lock queue in 40-bit addressed MEM.

Table 3.435. mem_lockq128_lock_ptr40 parameters

Type	Name	Description
mem_lockq128_ptr40_t	<i>lockq</i>	Pointer to lock queue in MEM
sync_t	<i>sync</i>	Type of synchronization to use. Must be sig_done.
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Even signal is raised upon completion. Odd signal is raised on overflow.

3.17.4.4 mem_lockq128_lock_ind_ptr40

Prototype:

```
void mem_lockq128_lock_ind_ptr40(mem_lockq128_ptr40_t lockq, generic_ind_t ind, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Claim a lock in a 128-bit lock queue in 40-bit addressed MEM with indirect word.

Table 3.436. mem_lockq128_lock_ind_ptr40 parameters

Type	Name	Description
mem_lockq128_ptr40_t	<i>lockq</i>	Pointer to lock queue in MEM
generic_ind_t	<i>ind</i>	Indirect word
sync_t	<i>sync</i>	Type of synchronization to use. Must be sig_done.
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Even signal is raised upon completion. Odd signal is raised on overflow.

3.17.4.5 mem_lockq256_lock_ptr32

Prototype:

```
void mem_lockq256_lock_ptr32(__mem mem_lockq256_in_mem_t* lockq, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Claim a lock in a 256-bit lock queue in 32-bit addressed MEM.

Table 3.437. mem_lockq256_lock_ptr32 parameters

Type	Name	Description
__mem mem_lockq256_in_mem_t*	<i>lockq</i>	Pointer to lock queue in MEM
sync_t	<i>sync</i>	Type of synchronization to use. Must be sig_done.
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Even signal is raised upon completion. Odd signal is raised on overflow.

3.17.4.6 mem_lockq256_lock_ind_ptr32

Prototype:

```
void mem_lockq256_lock_ind_ptr32(__mem mem_lockq256_in_mem_t* lockq, generic_ind_t ind,  
sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Claim a lock in a 256-bit lock queue in 32-bit addressed MEM with indirect word.

Table 3.438. mem_lockq256_lock_ind_ptr32 parameters

Type	Name	Description
__mem mem_lockq256_in_mem_t*	<i>lockq</i>	Pointer to lock queue in MEM
generic_ind_t	<i>ind</i>	Indirect word
sync_t	<i>sync</i>	Type of synchronization to use. Must be sig_done.
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Even signal is raised upon completion. Odd signal is raised on overflow.

3.17.4.7 mem_lockq256_lock_ptr40

Prototype:

```
void mem_lockq256_lock_ptr40(mem_lockq256_ptr40_t lockq, sync_t sync, SIGNAL_PAIR*  
sig_pair_ptr)
```

Description:

Claim a lock in a 256-bit lock queue in 40-bit addressed MEM.

Table 3.439. mem_lockq256_lock_ptr40 parameters

Type	Name	Description
mem_lockq256_ptr40_t	<i>lockq</i>	Pointer to lock queue in 40-bit addressed MEM
sync_t	<i>sync</i>	Type of synchronization to use. Must be sig_done.
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Even signal is raised upon completion. Odd signal is raised on overflow.

3.17.4.8 mem_lockq256_lock_ind_ptr40

Prototype:

```
void mem_lockq256_lock_ind_ptr40(mem_lockq256_ptr40_t lockq, generic_ind_t ind, sync_t
sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Claim a lock in a 256-bit lock queue in 40-bit addressed MEM with indirect word.

Table 3.440. mem_lockq256_lock_ind_ptr40 parameters

Type	Name	Description
mem_lockq256_ptr40_t	<i>lockq</i>	Pointer to lock queue in 40-bit addressed MEM with indirect word.
generic_ind_t	<i>ind</i>	Indirect word
sync_t	<i>sync</i>	Type of synchronization to use. Must be sig_done.
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Even signal is raised upon completion. Odd signal is raised on overflow.

3.17.4.9 mem_lockq128_unlock_ptr32

Prototype:

```
void mem_lockq128_unlock_ptr32(__mem mem_lockq128_in_mem_t* lockq)
```

Description:

Release a lock in a 128-bit lock queue in 32-bit addressed MEM.

Table 3.441. mem_lockq128_unlock_ptr32 parameters

Type	Name	Description
<u>__mem</u> mem_lockq128_in_mem_t*	<i>lockq</i>	Pointer to lock queue in MEM

3.17.4.10 mem_lockq128_unlock_ind_ptr32

Prototype:

```
void mem_lockq128_unlock_ind_ptr32(__mem mem_lockq128_in_mem_t* lockq, generic_ind_t ind)
```

Description:

Release a lock in a 128-bit lock queue in 32-bit addressed MEM with indirect word.

Table 3.442. mem_lockq128_unlock_ind_ptr32 parameters

Type	Name	Description
__mem mem_lockq128_in_mem_t*	<i>lockq</i>	Pointer to lock queue in MEM
generic_ind_t	<i>ind</i>	Indirect word

3.17.4.11 mem_lockq128_unlock_ptr40

Prototype:

```
void mem_lockq128_unlock_ptr40(mem_lockq128_ptr40_t lockq)
```

Description:

Release a lock in a 128-bit lock queue in 40-bit addressed MEM.

Table 3.443. mem_lockq128_unlock_ptr40 parameters

Type	Name	Description
mem_lockq128_ptr40_t	<i>lockq</i>	Pointer to lock queue in MEM

3.17.4.12 mem_lockq128_unlock_ind_ptr40

Prototype:

```
void mem_lockq128_unlock_ind_ptr40(mem_lockq128_ptr40_t lockq, generic_ind_t ind)
```

Description:

Release a lock in a 128-bit lock queue in 40-bit addressed MEM with indirect word.

Table 3.444. mem_lockq128_unlock_ind_ptr40 parameters

Type	Name	Description
mem_lockq128_ptr40_t	<i>lockq</i>	Pointer to lock queue in MEM
generic_ind_t	<i>ind</i>	Indirect word

3.17.4.13 mem_lockq256_unlock_ptr32

Prototype:

```
void mem_lockq256_unlock_ptr32(__mem mem_lockq256_in_mem_t* lockq)
```

Description:

Release a lock in a 256-bit lock queue in 32-bit addressed MEM.

Table 3.445. mem_lockq256_unlock_ptr32 parameters

Type	Name	Description
<code>__mem mem_lockq256_in_mem_t*</code>	<code>lockq</code>	Pointer to lock queue in MEM

3.17.4.14 mem_lockq256_unlock_ind_ptr32

Prototype:

```
void mem_lockq256_unlock_ind_ptr32(__mem mem_lockq256_in_mem_t* lockq, generic_ind_t ind)
```

Description:

Release a lock in a 256-bit lock queue in 32-bit addressed MEM with indirect word.

Table 3.446. mem_lockq256_unlock_ind_ptr32 parameters

Type	Name	Description
<code>__mem mem_lockq256_in_mem_t*</code>	<code>lockq</code>	Pointer to lock queue in MEM
<code>generic_ind_t</code>	<code>ind</code>	Indirect word

3.17.4.15 mem_lockq256_unlock_ptr40

Prototype:

```
void mem_lockq256_unlock_ptr40(mem_lockq256_ptr40_t lockq)
```

Description:

Release a lock in a 256-bit lock queue in 40-bit addressed MEM.

Table 3.447. mem_lockq256_unlock_ptr40 parameters

Type	Name	Description
<code>mem_lockq256_ptr40_t</code>	<code>lockq</code>	Pointer to lock queue in MEM

3.17.4.16 mem_lockq256_unlock_ind_ptr40

Prototype:

```
void mem_lockq256_unlock_ind_ptr40(mem_lockq256_ptr40_t lockq, generic_ind_t ind)
```

Description:

Release a lock in a 256-bit lock queue in 40-bit addressed MEM with indirect word.

Table 3.448. mem_lockq256_unlock_ind_ptr40 parameters

Type	Name	Description
mem_lockq256_ptr40_t	<i>lockq</i>	Pointer to lock queue in MEM
generic_ind_t	<i>ind</i>	Indirect word

3.17.4.17 mem_lockq128_init_ptr32

Prototype:

```
void mem_lockq128_init_ptr32(__mem mem_lockq128_in_mem_t* lockq, unsigned int
generate_event, unsigned int event_source)
```

Description:

Initialize a 128-bit lock queue in 32-bit addressed MEM.

The header of the lock queue is written and then read back to ensure the lock queue is configured when this function returns.

Table 3.449. mem_lockq128_init_ptr32 parameters

Type	Name	Description
__mem mem_lockq128_in_mem_t*	<i>lockq</i>	Pointer to lock queue in MEM
unsigned int	<i>generate_event</i>	Enable event on overflow or underflow
unsigned int	<i>event_source</i>	Event to present on event bus if generate_event is 1

3.17.4.18 mem_lockq128_init_ptr40

Prototype:

```
void mem_lockq128_init_ptr40(mem_lockq128_ptr40_t lockq, unsigned int generate_event,
unsigned int event_source)
```

Description:

Initialize a 128-bit lock queue in 40-bit addressed MEM.

The header of the lock queue is written and then read back to ensure the lock queue is configured when this function returns.

Table 3.450. mem_lockq128_init_ptr40 parameters

Type	Name	Description
mem_lockq128_ptr40_t	<i>lockq</i>	Pointer to lock queue in MEM

Type	Name	Description
unsigned int	<i>generate_event</i>	Enable event on overflow or underflow
unsigned int	<i>event_source</i>	Event to present on event bus if generate_event is 1

3.17.4.19 mem_lockq256_init_ptr32

Prototype:

```
void mem_lockq256_init_ptr32(__mem mem_lockq256_in_mem_t* lockq, unsigned int
generate_event, unsigned int event_source)
```

Description:

Initialize a 256-bit lock queue in 32-bit addressed MEM.

The header of the lock queue is written and then read back to ensure the lock queue is configured when this function returns.

Table 3.451. mem_lockq256_init_ptr32 parameters

Type	Name	Description
<i>lockq</i>	<i>lockq</i>	Pointer to lock queue in MEM
unsigned int	<i>generate_event</i>	Enable event on overflow or underflow
unsigned int	<i>event_source</i>	Event to present on event bus if generate_event is 1

3.17.4.20 mem_lockq256_init_ptr40

Prototype:

```
void mem_lockq256_init_ptr40(mem_lockq256_ptr40_t lockq, unsigned int generate_event,
unsigned int event_source)
```

Description:

Initialize a 256-bit lock queue in 40-bit addressed MEM.

The header of the lock queue is written and then read back to ensure the lock queue is configured when this function returns.

Table 3.452. mem_lockq256_init_ptr40 parameters

Type	Name	Description
<i>lockq</i>	<i>lockq</i>	Pointer to lock queue in MEM
unsigned int	<i>generate_event</i>	Enable event on overflow or underflow
unsigned int	<i>event_source</i>	Event to present on event bus if generate_event is 1

3.18 MEM Lock Intrinsics

This section discusses the enumerations, structs, unions, types and functions related to MEM Lock operations.

Memory lock commands support locking of specific ranges of memory as opposed to semantic locking supported by queue locking commands. It allows the user to request a lock on a memory region with a specific size and base address. The size of the locked region is specified in the byte mask and its address in the write transfer register. Memory locks are implemented using 16-bit TCAM lookup with addition.

3.18.1 MU Lock Structs

3.18.1.1 mem_lock128_t

128-bit memory lock container type.

Table 3.453. struct mem_lock128_t

Type	Name	Description
unsigned int	value[4]	Storage for the memory lock entries.

3.18.1.2 mem_lock256_t

256-bit memory lock container type.

Table 3.454. struct mem_lock256_t

Type	Name	Description
unsigned int	value[8]	Storage for the memory lock entries.

3.18.1.3 mem_lock384_t

384-bit memory lock container type.

Table 3.455. struct mem_lock384_t

Type	Name	Description
unsigned int	value[12]	Storage for the memory lock entries.

3.18.1.4 mem_lock512_t

512-bit memory lock container type.

Table 3.456. struct mem_lock512_t

Type	Name	Description
unsigned int	value[16]	Storage for the memory lock entries.

3.18.2 MU Lock Unions

3.18.2.1 mem_lock_in_t

Input type for the memory lock operation.

The address field must be populated before requesting the memory lock.

Table 3.457. union mem_lock_in_t

Type	Name	Description
unsigned int	reserved:16	Unused, the regions are addressable to 64k regions.
unsigned int	address:16	Region address to lock. <div style="border: 1px solid blue; padding: 5px; margin-top: 10px;"> Note  Bit 0 is reserved and is assumed to be 0. </div>
unsigned int	value	Accessor to entire structure.

3.18.2.2 mem_lock_out_t

Output type for the memory lock operation.

Table 3.458. union mem_lock_out_t

Type	Name	Description
unsigned int	match_bitf:16	Bitfield of matching entries.
unsigned int	reserved:8	Reserved.
unsigned int	added:1	When set, this field indicates that an entry was added. <div style="border: 1px solid blue; padding: 5px; margin-top: 10px;"> Note  This field is also set when the memory lock queue is full. </div>

Type	Name	Description
unsigned int	first_match:7	First matched entry number, or on a new memory lock entry where the entry was added.
		<p>Note</p>  <p>All-bits in this field are set when the memory lock queue is full.</p>
unsigned int	value	Accessor to entire structure.

3.18.3 MU Lock Typedefs

3.18.3.1 mem_lock128_t

128-bit memory lock container type.

Table 3.459. typedef mem_lock128_t

Type	Definition
mem_lock128_t	struct mem_lock128_t

3.18.3.2 mem_lock256_t

256-bit memory lock container type.

Table 3.460. typedef mem_lock256_t

Type	Definition
mem_lock256_t	struct mem_lock256_t

3.18.3.3 mem_lock384_t

384-bit memory lock container type.

Table 3.461. typedef mem_lock384_t

Type	Definition
mem_lock384_t	struct mem_lock384_t

3.18.3.4 mem_lock512_t

512-bit memory lock container type.

Table 3.462. `typedef mem_lock512_t`

Type	Definition
<code>mem_lock512_t</code>	<code>struct mem_lock512_t</code>

3.18.3.5 `mem_lock128_in_mem_t`

Table 3.463. `typedef mem_lock128_in_mem_t`

Type	Definition
<code>mem_lock128_in_mem_t</code>	<code>__addr32 __align16 mem_lock128_t</code>

3.18.3.6 `mem_lock256_in_mem_t`

Table 3.464. `typedef mem_lock256_in_mem_t`

Type	Definition
<code>mem_lock256_in_mem_t</code>	<code>__addr32 __align16 mem_lock256_t</code>

3.18.3.7 `mem_lock384_in_mem_t`

Table 3.465. `typedef mem_lock384_in_mem_t`

Type	Definition
<code>mem_lock384_in_mem_t</code>	<code>__addr32 __align16 mem_lock384_t</code>

3.18.3.8 `mem_lock512_in_mem_t`

Table 3.466. `typedef mem_lock512_in_mem_t`

Type	Definition
<code>mem_lock512_in_mem_t</code>	<code>__addr32 __align16 mem_lock512_t</code>

3.18.3.9 `mem_lock128_ptr40_t`

Table 3.467. `typedef mem_lock128_ptr40_t`

Type	Definition
<code>mem_lock128_ptr40_t</code>	<code>__mem __addr40 __align16 mem_lock128_t*</code>

3.18.3.10 mem_lock256_ptr40_t

Table 3.468. typedef mem_lock256_ptr40_t

Type	Definition
mem_lock256_ptr40_t	__mem __addr40 __align16 mem_lock256_t*

3.18.3.11 mem_lock384_ptr40_t

Table 3.469. typedef mem_lock384_ptr40_t

Type	Definition
mem_lock384_ptr40_t	__mem __addr40 __align16 mem_lock384_t*

3.18.3.12 mem_lock512_ptr40_t

Table 3.470. typedef mem_lock512_ptr40_t

Type	Definition
mem_lock512_ptr40_t	__mem __addr40 __align16 mem_lock512_t*

3.18.3.13 mem_lock_in_t

Input type for the memory lock operation.

The address field must be populated before requesting the memory lock.

Table 3.471. typedef mem_lock_in_t

Type	Definition
mem_lock_in_t	union mem_lock_in_t

3.18.3.14 mem_lock_out_t

Output type for the memory lock operation.

Table 3.472. typedef mem_lock_out_t

Type	Definition
mem_lock_out_t	union mem_lock_out_t

3.18.3.15 mem_lock_out_in_read_reg_t

Type for mem_lock_out_t in read registers.

Table 3.473. `typedef mem_lock_out_in_read_reg_t`

Type	Definition
<code>mem_lock_out_in_read_reg_t</code>	<code>__xread mem_lock_out_t</code>

3.18.4 MU Lock Functions

3.18.4.1 `mem_lock128_init_ptr32`

Prototype:

```
void mem_lock128_init_ptr32(__mem mem_lock128_in_mem_t* mem_lock)
```

Description:

Initialize a 128-bit memory lock container in 32-bit addressed MEM in memory.

Table 3.474. `mem_lock128_init_ptr32` parameters

Type	Name	Description
<code>__mem mem_lock128_in_mem_t*</code>	<code>mem_lock</code>	Pointer to the memory lock container in MEM

3.18.4.2 `mem_lock128_init_ptr40`

Prototype:

```
void mem_lock128_init_ptr40(mem_lock128_ptr40_t mem_lock)
```

Description:

Initialize a 128-bit memory lock container in 40-bit addressed MEM.

Table 3.475. `mem_lock128_init_ptr40` parameters

Type	Name	Description
<code>mem_lock128_ptr40_t</code>	<code>mem_lock</code>	Pointer to the memory lock container in MEM

3.18.4.3 `mem_lock256_init_ptr32`

Prototype:

```
void mem_lock256_init_ptr32(__mem mem_lock256_in_mem_t* mem_lock)
```

Description:

Initialize a 256-bit memory lock container in 32-bit addressed MEM.

Table 3.476. mem_lock256_init_ptr32 parameters

Type	Name	Description
__mem mem_lock256_in_mem_t*	<i>mem_lock</i>	Pointer to the memory lock container in MEM

3.18.4.4 mem_lock256_init_ptr40

Prototype:

```
void mem_lock256_init_ptr40(mem_lock256_ptr40_t mem_lock)
```

Description:

Initialize a 256-bit lock container in 40-bit addressed MEM.

Table 3.477. mem_lock256_init_ptr40 parameters

Type	Name	Description
mem_lock256_ptr40_t	<i>mem_lock</i>	Pointer to the memory lock container in MEM

3.18.4.5 mem_lock384_init_ptr32

Prototype:

```
void mem_lock384_init_ptr32(__mem mem_lock384_in_mem_t* mem_lock)
```

Description:

Initialize a 384-bit memory lock container in 32-bit addressed MEM in memory.

Table 3.478. mem_lock384_init_ptr32 parameters

Type	Name	Description
__mem mem_lock384_in_mem_t*	<i>mem_lock</i>	Pointer to the memory lock container in MEM

3.18.4.6 mem_lock384_init_ptr40

Prototype:

```
void mem_lock384_init_ptr40(mem_lock384_ptr40_t mem_lock)
```

Description:

Initialize a 384-bit memory lock container in 40-bit addressed MEM.

Table 3.479. mem_lock384_init_ptr40 parameters

Type	Name	Description
mem_lock384_ptr40_t	<i>mem_lock</i>	Pointer to the memory lock container in MEM

3.18.4.7 mem_lock512_init_ptr32

Prototype:

```
void mem_lock512_init_ptr32(__mem mem_lock512_in_mem_t* mem_lock)
```

Description:

Initialize a 512-bit memory lock container in 32-bit addressed MEM in memory.

Table 3.480. mem_lock512_init_ptr32 parameters

Type	Name	Description
__mem mem_lock512_in_mem_t*	<i>mem_lock</i>	Pointer to the memory lock container in MEM

3.18.4.8 mem_lock512_init_ptr40

Prototype:

```
void mem_lock512_init_ptr40(mem_lock512_ptr40_t mem_lock)
```

Description:

Initialize a 512-bit memory lock container in 40-bit addressed MEM.

Table 3.481. mem_lock512_init_ptr40 parameters

Type	Name	Description
mem_lock512_ptr40_t	<i>mem_lock</i>	Pointer to the memory lock container in MEM

3.18.4.9 mem_lock128_ptr32

Prototype:

```
mem_lock_out_in_read_reg_t* mem_lock128_ptr32(__mem mem_lock128_in_mem_t* mem_lock,  
__xwrite mem_lock_in_t* xfer, unsigned int lsbzeroes, sync_t sync, SIGNAL_PAIR*  
sig_pair_ptr)
```

Description:

Attempt to acquire a lock entry in a 128-bit lock container in 32-bit addressed MEM, supporting 4 locks .



Note

The result is returned in a `mem_lock_out_t` structure.

Table 3.482. `mem_lock128_ptr32` parameters

Type	Name	Description
<code>__mem mem_lock128_in_mem_t*</code>	<code>mem_lock</code>	Pointer to the memory lock container in MEM
<code>__xwrite mem_lock_in_t*</code>	<code>xfer</code>	Pointer to write xfer register containing the region address
<code>unsigned int</code>	<code>lsbzzeroes</code>	Number of least significant-bits to set to zero in the lock mask. This represent the lock region size.
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use. (must be <code>sig_done</code>)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal pair to raise upon completion

Returns:

Pointer to the read transfer register containing the result of the lock attempt

See Also:

- `mem_lock_in_t`, `mem_lock_out_t`

3.18.4.10 `mem_lock128_ptr40`

Prototype:

```
mem_lock_out_in_read_reg_t* mem_lock128_ptr40(mem_lock128_ptr40_t mem_lock, __xwrite
mem_lock_in_t* xfer, unsigned int lsbzzeroes, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Attempt to acquire a lock entry in a 128-bit lock container in 40-bit addressed MEM, supporting 4 locks .



Note

The result is returned in a `mem_lock_out_t` structure.

Table 3.483. `mem_lock128_ptr40` parameters

Type	Name	Description
<code>mem_lock128_ptr40_t</code>	<code>mem_lock</code>	Pointer to the memory lock container in MEM
<code>__xwrite mem_lock_in_t*</code>	<code>xfer</code>	Pointer to write xfer register containing the region address

Type	Name	Description
unsigned int	<i>lsbzeroes</i>	Number of least significant-bits to set to zero in the lock mask. This represent the lock region size.
sync_t	<i>sync</i>	Type of synchronization to use. (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the read transfer register containing the result of the lock attempt

See Also:

- `mem_lock_in_t`, `mem_lock_out_t`

3.18.4.11 mem_lock256_ptr32

Prototype:

```
mem_lock_out_in_read_reg_t* mem_lock256_ptr32(__mem mem_lock256_in_mem_t* mem_lock,
__xwrite mem_lock_in_t* xfer, unsigned int lsbzeroes, sync_t sync, SIGNAL_PAIR*
sig_pair_ptr)
```

Description:

Attempt to acquire a lock entry in a 256-bit lock container in 32-bit addressed MEM, supporting 8 locks.



Note

The result is returned in a `mem_lock_out_t` structure.

Table 3.484. mem_lock256_ptr32 parameters

Type	Name	Description
<code>__mem mem_lock256_in_mem_t*</code>	<i>mem_lock</i>	Pointer to the memory lock container in MEM
<code>__xwrite mem_lock_in_t*</code>	<i>xfer</i>	Pointer to write xfer register containing the region address
unsigned int	<i>lsbzeroes</i>	Number of least significant-bits to set to zero in the lock mask. This represent the lock region size.
sync_t	<i>sync</i>	Type of synchronization to use. (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the read transfer register containing the result of the lock attempt

See Also:

- `mem_lock_in_t`, `mem_lock_out_t`

3.18.4.12 mem_lock256_ptr40

Prototype:

```
mem_lock_out_in_read_reg_t* mem_lock256_ptr40(mem_lock256_ptr40_t mem_lock, __xwrite
mem_lock_in_t* xfer, unsigned int lsbzeroes, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Attempt to acquire a lock entry in a 256-bit lock container in 40-bit addressed MEM, supporting 8 locks.



Note

The result is returned in a `mem_lock_out_t` structure.

Table 3.485. mem_lock256_ptr40 parameters

Type	Name	Description
<code>mem_lock256_ptr40_t</code>	<code>mem_lock</code>	Pointer to the memory lock container in MEM
<code>__xwrite mem_lock_in_t*</code>	<code>xfer</code>	Pointer to write xfer register containing the region address
<code>unsigned int</code>	<code>lsbzeroes</code>	Number of least significant-bits to set to zero in the lock mask. This represent the lock region size.
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use. (must be <code>sig_done</code>)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal pair to raise upon completion

Returns:

Pointer to the read transfer register containing the result of the lock attempt

See Also:

- `mem_lock_in_t`, `mem_lock_out_t`

3.18.4.13 mem_lock384_ptr32

Prototype:

```
mem_lock_out_in_read_reg_t* mem_lock384_ptr32(__mem mem_lock384_in_mem_t* mem_lock,
__xwrite mem_lock_in_t* xfer, unsigned int lsbzeroes, sync_t sync, SIGNAL_PAIR*
sig_pair_ptr)
```

Description:

Attempt to acquire a lock entry in a 384-bit lock container in 32-bit addressed MEM, supporting 12 locks.



Note

The result is returned in a `mem_lock_out_t` structure.

Table 3.486. mem_lock384_ptr32 parameters

Type	Name	Description
<code>__mem mem_lock384_in_mem_t*</code>	<code>mem_lock</code>	Pointer to the memory lock container in MEM
<code>__xwrite mem_lock_in_t*</code>	<code>xfer</code>	Pointer to write xfer register containing the region address
<code>unsigned int</code>	<code>lsbzzeroes</code>	Number of least significant-bits to set to zero in the lock mask. This represent the lock region size.
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use. (must be sig_done)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal pair to raise upon completion

Returns:

Pointer to the read transfer register containing the result of the lock attempt

See Also:

- `mem_lock_in_t`, `mem_lock_out_t`

3.18.4.14 mem_lock384_ptr40

Prototype:

```
mem_lock_out_in_read_reg_t* mem_lock384_ptr40(mem_lock384_ptr40_t mem_lock, __xwrite
mem_lock_in_t* xfer, unsigned int lsbzzeroes, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Attempt to acquire a lock entry in a 384-bit lock container in 40-bit addressed MEM, supporting 12 locks.



Note

The result is returned in a `mem_lock_out_t` structure.

Table 3.487. mem_lock384_ptr40 parameters

Type	Name	Description
<code>mem_lock384_ptr40_t</code>	<code>mem_lock</code>	Pointer to the memory lock container in MEM
<code>__xwrite mem_lock_in_t*</code>	<code>xfer</code>	Pointer to write xfer register containing the region address

Type	Name	Description
unsigned int	<i>lsbzeroes</i>	Number of least significant-bits to set to zero in the lock mask. This represent the lock region size.
sync_t	<i>sync</i>	Type of synchronization to use. (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the read transfer register containing the result of the lock attempt

See Also:

- `mem_lock_in_t`, `mem_lock_out_t`

3.18.4.15 mem_lock512_ptr32

Prototype:

```
mem_lock_out_in_read_reg_t* mem_lock512_ptr32(__mem mem_lock512_in_mem_t* mem_lock,
__xwrite mem_lock_in_t* xfer, unsigned int lsbzeroes, sync_t sync, SIGNAL_PAIR*
sig_pair_ptr)
```

Description:

Attempt to acquire a lock entry in a 512-bit lock container in 32-bit addressed MEM, supporting 16 locks.



Note

The result is returned in a `mem_lock_out_t` structure.

Table 3.488. mem_lock512_ptr32 parameters

Type	Name	Description
<code>__mem mem_lock512_in_mem_t*</code>	<i>mem_lock</i>	Pointer to the memory lock container in MEM
<code>__xwrite mem_lock_in_t*</code>	<i>xfer</i>	Pointer to write xfer register containing the region address
unsigned int	<i>lsbzeroes</i>	Number of least significant-bits to set to zero in the lock mask. This represent the lock region size.
sync_t	<i>sync</i>	Type of synchronization to use. (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the read transfer register containing the result of the lock attempt

See Also:

- `mem_lock_in_t`, `mem_lock_out_t`

3.18.4.16 mem_lock512_ptr40

Prototype:

```
mem_lock_out_in_read_reg_t* mem_lock512_ptr40(mem_lock512_ptr40_t mem_lock, __xwrite
mem_lock_in_t* xfer, unsigned int lsbzeroes, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Attempt to acquire a lock entry in a 512-bit lock container in 40-bit addressed MEM, supporting 16 locks.



Note

The result is returned in a `mem_lock_out_t` structure.

Table 3.489. mem_lock512_ptr40 parameters

Type	Name	Description
<code>mem_lock512_ptr40_t</code>	<code>mem_lock</code>	Pointer to the memory lock container in MEM
<code>__xwrite mem_lock_in_t*</code>	<code>xfer</code>	Pointer to write xfer register containing the region address
<code>unsigned int</code>	<code>lsbzeroes</code>	Number of least significant-bits to set to zero in the lock mask. This represent the lock region size.
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use. (must be <code>sig_done</code>)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal pair to raise upon completion

Returns:

Pointer to the read transfer register containing the result of the lock attempt

See Also:

- `mem_lock_in_t`, `mem_lock_out_t`

3.18.4.17 mem_unlock128_ptr32

Prototype:

```
void mem_unlock128_ptr32(__mem mem_lock128_in_mem_t* mem_lock, unsigned int index, sync_t
sync, SIGNAL* sig_ptr)
```

Description:

Unlock a memory lock entry in a 128-bit lock container in 32-bit addressed MEM.



Note

The maximum index is 3.

Table 3.490. mem_unlock128_ptr32 parameters

Type	Name	Description
<code>__mem mem_lock128_in_mem_t*</code>	<code>mem_lock</code>	Pointer to the memory lock container in MEM
<code>unsigned int</code>	<code>index</code>	Memory lock entry to unlock (0 is the first entry)
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use. (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.18.4.18 mem_unlock128_ptr40

Prototype:

```
void mem_unlock128_ptr40(mem_lock128_ptr40_t mem_lock, unsigned int index, sync_t sync,
SIGNAL* sig_ptr)
```

Description:

Unlock a memory lock entry in a 128-bit lock container in 40-bit addressed MEM.



Note

The maximum index is 3.

Table 3.491. mem_unlock128_ptr40 parameters

Type	Name	Description
<code>mem_lock128_ptr40_t</code>	<code>mem_lock</code>	Pointer to the memory lock container in MEM
<code>unsigned int</code>	<code>index</code>	Memory lock entry to unlock (0 is the first entry)
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use. (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.18.4.19 mem_unlock256_ptr32

Prototype:

```
void mem_unlock256_ptr32(__mem mem_lock256_in_mem_t* mem_lock, unsigned int index, sync_t
sync, SIGNAL* sig_ptr)
```

Description:

Unlock a memory lock entry in a 256-bit lock container in 32-bit addressed MEM.



Note

The maximum index is 7.

Table 3.492. mem_unlock256_ptr32 parameters

Type	Name	Description
__mem mem_lock256_in_mem_t*	<i>mem_lock</i>	Pointer to the memory lock container in MEM
unsigned int	<i>index</i>	Memory lock entry to unlock (0 is the first entry)
sync_t	<i>sync</i>	Type of synchronization to use. (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.18.4.20 mem_unlock256_ptr40

Prototype:

```
void mem_unlock256_ptr40(mem_lock256_ptr40_t mem_lock, unsigned int index, sync_t sync,
SIGNAL* sig_ptr)
```

Description:

Unlock a memory lock entry in a 256-bit lock container in 40-bit addressed MEM.



Note

The maximum index is 7.

Table 3.493. mem_unlock256_ptr40 parameters

Type	Name	Description
mem_lock256_ptr40_t	<i>mem_lock</i>	Pointer to the memory lock container in MEM
unsigned int	<i>index</i>	Memory lock entry to unlock (0 is the first entry)
sync_t	<i>sync</i>	Type of synchronization to use. (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.18.4.21 mem_unlock384_ptr32

Prototype:

```
void mem_unlock384_ptr32(__mem mem_lock384_in_mem_t* mem_lock, unsigned int index, sync_t
sync, SIGNAL* sig_ptr)
```

Description:

Unlock a memory lock entry in a 384-bit lock container in 32-bit addressed MEM.



Note

The maximum index is 11.

Table 3.494. mem_unlock384_ptr32 parameters

Type	Name	Description
<code>__mem mem_lock384_in_mem_t*</code>	<code>mem_lock</code>	Pointer to the memory lock container in MEM
<code>unsigned int</code>	<code>index</code>	Memory lock entry to unlock (0 is the first entry)
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use. (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.18.4.22 mem_unlock384_ptr40

Prototype:

```
void mem_unlock384_ptr40(mem_lock384_ptr40_t mem_lock, unsigned int index, sync_t sync,
SIGNAL* sig_ptr)
```

Description:

Unlock a memory lock entry in a 384-bit lock container in 40-bit addressed MEM.



Note

The maximum index is 11.

Table 3.495. mem_unlock384_ptr40 parameters

Type	Name	Description
<code>mem_lock384_ptr40_t</code>	<code>mem_lock</code>	Pointer to the memory lock container in MEM
<code>unsigned int</code>	<code>index</code>	Memory lock entry to unlock (0 is the first entry)
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use. (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.18.4.23 mem_unlock512_ptr32

Prototype:

```
void mem_unlock512_ptr32(__mem mem_lock512_in_mem_t* mem_lock, unsigned int index, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Unlock a memory lock entry in a 512-bit lock container in 32-bit addressed MEM.



Note

The maximum index is 15.

Table 3.496. mem_unlock512_ptr32 parameters

Type	Name	Description
<code>__mem mem_lock512_in_mem_t*</code>	<code>mem_lock</code>	Pointer to the memory lock container in MEM
<code>unsigned int</code>	<code>index</code>	Memory lock entry to unlock (0 is the first entry)
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use. (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.18.4.24 mem_unlock512_ptr40

Prototype:

```
void mem_unlock512_ptr40(mem_lock512_ptr40_t mem_lock, unsigned int index, sync_t sync,
SIGNAL* sig_ptr)
```

Description:

Unlock a memory lock entry in a 512-bit lock container in 40-bit addressed MEM.



Note

The maximum index is 15.

Table 3.497. mem_unlock512_ptr40 parameters

Type	Name	Description
<code>mem_lock512_ptr40_t</code>	<code>mem_lock</code>	Pointer to the memory lock container in MEM
<code>unsigned int</code>	<code>index</code>	Memory lock entry to unlock (0 is the first entry)
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use. (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.19 MEM List Intrinsics

This section discusses the enumerations, structs, unions, types and functions related to MEM List operations.

3.19.1 MU List Unions

3.19.1.1 mem_list_desc_t0

List type 0 descriptor type.

Table 3.498. union mem_list_desc_t0

Type	Name	Description
unsigned int	seg_cnt:6	Segment count.
unsigned int	head_ptr:24	Word address of first buffer.
unsigned int	eop:1	End Of Packet indication.
unsigned int	sop:1	Start Of Packet indication.
unsigned int	tail_ptr:30	Word address of last(tail) buffer of the list.
unsigned int	q_type:2	Set to TYPE_0.
unsigned int	q_loc:2	Locality specification of list.
unsigned int	reserved_1:4	Reserved.
unsigned int	q_page:2	Page of list storage.
unsigned int	q_count:24	Number of entries in list.
unsigned int	reserved_2	Reserved.
unsigned int	value[4]	Accessor to entire structure.

3.19.1.2 mem_list_desc_t1

List type 1 descriptor type.

Buffer-based linked list counting packets: each link includes SOP, EOP and a 24-bit word address.

Table 3.499. union mem_list_desc_t1

Type	Name	Description
unsigned int	user_data:6	6 - bits of user data.
unsigned int	head_ptr:24	Word address of first buffer.
unsigned int	eop:1	End Of Packet indication.
unsigned int	sop:1	Start Of Packet indication.

Type	Name	Description
unsigned int	tail_ptr:30	Word address of last(tail) buffer of the list.
unsigned int	q_type:2	Set to TYPE_1.
unsigned int	q_loc:2	Locality specification of list.
unsigned int	reserved_1:4	Reserved.
unsigned int	q_page:2	Page of list storage.
unsigned int	q_count:24	Number of entries in list.
unsigned int	reserved_2	Reserved.
unsigned int	value[4]	Accessor to entire descriptor structure.

3.19.1.3 mem_list_desc_t2

List type 2 descriptor type.

32-bit buffer based linked list counting buffers.

Table 3.500. union mem_list_desc_t2

Type	Name	Description
unsigned int	head_ptr:30	Word address of first buffer.
unsigned int	eop:1	End Of Packet indication.
unsigned int	sop:1	Start Of Packet indication.
unsigned int	tail_ptr:30	Word address of last(tail) buffer of the list.
unsigned int	q_type:2	Set to TYPE_2.
unsigned int	q_loc:2	Locality specification of list.
unsigned int	reserved_1:4	Reserved.
unsigned int	q_page:2	Page of list storage.
unsigned int	q_count:24	Number of entries in list.
unsigned int	reserved_2	Reserved.
unsigned int	value[4]	Accessor to entire descriptor structure.

3.19.1.4 mem_list_desc_t3

List Type 3 descriptor type.

24-bit buffer-based linked list counting buffers.

Table 3.501. union mem_list_desc_t3

Type	Name	Description
unsigned int	seg_cnt:6	Segment count.

Type	Name	Description
unsigned int	head_ptr:24	Word address of first buffer.
unsigned int	eop:1	End Of Packet indication.
unsigned int	sop:1	Start Of Packet indication.
unsigned int	tail_ptr:30	Word address of last(tail) buffer of the list.
unsigned int	q_type:2	Set to TYPE_3.
unsigned int	q_loc:2	Locality specification of list.
unsigned int	reserved_1:4	Reserved.
unsigned int	q_page:2	Page of list storage.
unsigned int	q_count:24	Number of entries in list.
unsigned int	reserved_2	Reserved.
unsigned int	value[4]	Accessor to entire descriptor structure.

3.19.1.5 mem_list_link_mem_t0

List type 0 link descriptor format in memory.

Table 3.502. union mem_list_link_mem_t0

Type	Name	Description
unsigned int	eop:1	End Of Packet indication.
unsigned int	sop:1	Start Of Packet indication.
unsigned int	seg_cnt:6	Segment count.
unsigned int	next_ptr:24	Pointer to next list entry.
unsigned int	value	Accessor to entire structure.

3.19.1.6 mem_list_link_mem_t1

List type 1 link descriptor format in memory.

This type is typically used when a linked list is construct in memory using a ME.

Table 3.503. union mem_list_link_mem_t1

Type	Name	Description
unsigned int	eop:1	End Of Packet indication.
unsigned int	sop:1	Start Of Packet indication.
unsigned int	user_data:6	6 - bits of user data.
unsigned int	next_ptr:24	Word address of next list entry.
unsigned int	value	Accessor to entire structure.

3.19.1.7 mem_list_link_mem_t2

List type 2 link descriptor format in memory.

This type is typically used when a linked list is construct in memory using a ME.

Table 3.504. union mem_list_link_mem_t2

Type	Name	Description
unsigned int	next_ptr:30	Word address of next list entry.
unsigned int	eop:1	End Of Packet indication.
unsigned int	sop:1	Start Of Packet indication.
unsigned int	value	Accessor to entire structure.

3.19.1.8 mem_list_link_mem_t3

List type 3 link descriptor format in memory.

This type is typically used when a linked list is construct in memory using a ME.

Table 3.505. union mem_list_link_mem_t3

Type	Name	Description
unsigned int	eop:1	End Of Packet indication.
unsigned int	sop:1	Start Of Packet indication.
unsigned int	seg_cnt:6	Segment count.
unsigned int	next_ptr:24	Word address of next list entry.
unsigned int	value	Accessor to entire structure.

3.19.1.9 mem_list_link_t0

List type 0 link type.

Table 3.506. union mem_list_link_t0

Type	Name	Description
unsigned int	seg_cnt:6	Segment count.
unsigned int	next_ptr:24	Pointer to next list entry.
unsigned int	eop:1	End Of Packet indication.
unsigned int	sop:1	Start Of Packet indication.
unsigned int	value	Accessor to entire structure.

3.19.1.10 mem_list_link_t1

List type 1 link type.

Table 3.507. union mem_list_link_t1

Type	Name	Description
unsigned int	user_data:6	6 - bits of user data.
unsigned int	next_ptr:24	Word address of next list entry.
unsigned int	eop:1	End Of Packet indication.
unsigned int	sop:1	Start Of Packet indication.
unsigned int	value	Accessor to entire structure.

3.19.1.11 mem_list_link_t2

List type 2 link type.

Table 3.508. union mem_list_link_t2

Type	Name	Description
unsigned int	next_ptr:30	Word address of next list entry.
unsigned int	eop:1	End Of Packet indication.
unsigned int	sop:1	Start Of Packet indication.
unsigned int	value	Accessor to entire structure.

3.19.1.12 mem_list_link_t3

List type 3 link type.

Table 3.509. union mem_list_link_t3

Type	Name	Description
unsigned int	seg_cnt:6	Segment count.
unsigned int	next_ptr:24	Word address of next list entry.
unsigned int	eop:1	End Of Packet indication.
unsigned int	sop:1	Start Of Packet indication.
unsigned int	value	Accessor to entire structure.

3.19.2 MU List Typedefs

3.19.2.1 MEM_LIST_TYPE

Available list types.

Table 3.510. typedef MEM_LIST_TYPE

Type	Definition
MEM_LIST_TYPE	enum MEM_LIST_TYPE

3.19.2.2 mem_list_desc_t0

List type 0 descriptor type.

Table 3.511. typedef mem_list_desc_t0

Type	Definition
mem_list_desc_t0	union mem_list_desc_t0

3.19.2.3 mem_list_link_t0

List type 0 link type.

Table 3.512. typedef mem_list_link_t0

Type	Definition
mem_list_link_t0	union mem_list_link_t0

3.19.2.4 mem_list_link_mem_t0

List type 0 link descriptor format in memory.

Table 3.513. typedef mem_list_link_mem_t0

Type	Definition
mem_list_link_mem_t0	union mem_list_link_mem_t0

3.19.2.5 mem_list_desc_in_mem_t0

List type 0 descriptor aligned in memory.

Table 3.514. `typedef mem_list_desc_in_mem_t0`

Type	Definition
<code>mem_list_desc_in_mem_t0</code>	<code>__addr40 __align16 mem_list_desc_t0</code>

3.19.2.6 `mem_list_link_in_mem_t0`

List type 0 link descriptor aligned in memory.

Table 3.515. `typedef mem_list_link_in_mem_t0`

Type	Definition
<code>mem_list_link_in_mem_t0</code>	<code>__addr40 __align16 mem_list_link_mem_t0</code>

3.19.2.7 `mem_list_desc_t1`

List type 1 descriptor type.

Buffer-based linked list counting packets: each link includes SOP, EOP and a 24-bit word address.

Table 3.516. `typedef mem_list_desc_t1`

Type	Definition
<code>mem_list_desc_t1</code>	<code>union mem_list_desc_t1</code>

3.19.2.8 `mem_list_link_t1`

List type 1 link type.

Table 3.517. `typedef mem_list_link_t1`

Type	Definition
<code>mem_list_link_t1</code>	<code>union mem_list_link_t1</code>

3.19.2.9 `mem_list_link_mem_t1`

List type 1 link descriptor format in memory.

This type is typically used when a linked list is construct in memory using a ME.

Table 3.518. `typedef mem_list_link_mem_t1`

Type	Definition
<code>mem_list_link_mem_t1</code>	<code>union mem_list_link_mem_t1</code>

3.19.2.10 mem_list_desc_in_mem_t1

List type 1 descriptor aligned in memory.

Table 3.519. typedef mem_list_desc_in_mem_t1

Type	Definition
mem_list_desc_in_mem_t1	__addr40 __align16 mem_list_desc_t1

3.19.2.11 mem_list_link_in_mem_t1

List type 1 link aligned in memory.

Table 3.520. typedef mem_list_link_in_mem_t1

Type	Definition
mem_list_link_in_mem_t1	__addr40 __align16 mem_list_link_mem_t1

3.19.2.12 mem_list_desc_t2

List type 2 descriptor type.

32-bit buffer based linked list counting buffers.

Table 3.521. typedef mem_list_desc_t2

Type	Definition
mem_list_desc_t2	union mem_list_desc_t2

3.19.2.13 mem_list_link_t2

List type 2 link type.

Table 3.522. typedef mem_list_link_t2

Type	Definition
mem_list_link_t2	union mem_list_link_t2

3.19.2.14 mem_list_link_mem_t2

List type 2 link descriptor format in memory.

This type is typically used when a linked list is construct in memory using a ME.

Table 3.523. `typedef mem_list_link_mem_t2`

Type	Definition
<code>mem_list_link_mem_t2</code>	<code>union mem_list_link_mem_t2</code>

3.19.2.15 `mem_list_desc_in_mem_t2`

List type 2 descriptor aligned in memory.

Table 3.524. `typedef mem_list_desc_in_mem_t2`

Type	Definition
<code>mem_list_desc_in_mem_t2</code>	<code>__addr40 __align16 mem_list_desc_t2</code>

3.19.2.16 `mem_list_link_in_mem_t2`

List type 2 link aligned in memory.

This type is typically used when a linked list is construct in memory using a ME

Table 3.525. `typedef mem_list_link_in_mem_t2`

Type	Definition
<code>mem_list_link_in_mem_t2</code>	<code>__addr40 __align16 mem_list_link_mem_t2</code>

3.19.2.17 `mem_list_desc_t3`

List Type 3 descriptor type.

24-bit buffer-based linked list counting buffers.

Table 3.526. `typedef mem_list_desc_t3`

Type	Definition
<code>mem_list_desc_t3</code>	<code>union mem_list_desc_t3</code>

3.19.2.18 `mem_list_link_t3`

List type 3 link type.

Table 3.527. `typedef mem_list_link_t3`

Type	Definition
<code>mem_list_link_t3</code>	<code>union mem_list_link_t3</code>

3.19.2.19 mem_list_link_mem_t3

List type 3 link descriptor format in memory.

This type is typically used when a linked list is construct in memory using a ME.

Table 3.528. typedef mem_list_link_mem_t3

Type	Definition
mem_list_link_mem_t3	union mem_list_link_mem_t3

3.19.2.20 mem_list_desc_in_mem_t3

List type 3 descriptor aligned in memory.

Table 3.529. typedef mem_list_desc_in_mem_t3

Type	Definition
mem_list_desc_in_mem_t3	__addr40 __align16 mem_list_desc_t3

3.19.2.21 mem_list_link_in_mem_t3

List type 3 link aligned in memory.

Table 3.530. typedef mem_list_link_in_mem_t3

Type	Definition
mem_list_link_in_mem_t3	__addr40 __align16 mem_list_link_mem_t3

3.19.3 MU List Functions

3.19.3.1 mem_list_read_desc_t0

Prototype:

```
void mem_list_read_desc_t0(unsigned int q_array_number, mem_list_desc_in_mem_t0
*mem_list_desc_in_mem)
```

Description:

Read MEM linked list queue descriptor for type 0 lists from MEM into queue array.

Table 3.531. mem_list_read_desc_t0 parameters

Type	Name	Description
unsigned int	<i>q_array_number</i>	queue array number in which to load. The range of the array number identifies the external memory to use. <ul style="list-style-type: none"> • 0 - 1023: emem0 or island 24, • 1024 - 2047: emem1 or island 25, • 2048 - 3071: emem2 or island 26
mem_list_desc_in_mem_t0*	<i>mem_list_desc_in_mem</i>	40 bit pointer to list descriptor in external memory including the MU island. The pointer must be 16 byte aligned.

3.19.3.2 mem_list_read_desc_t1

Prototype:

```
void mem_list_read_desc_t1(unsigned int q_array_number, mem_list_desc_in_mem_t1
*mem_list_desc_in_mem)
```

Description:

Read MEM linked list queue descriptor for type 1 lists from MEM into queue array.

Table 3.532. mem_list_read_desc_t1 parameters

Type	Name	Description
unsigned int	<i>q_array_number</i>	queue array number in which to load. The range of the array number identifies the external memory to use. <ul style="list-style-type: none"> • 0 - 1023: emem0 or island 24, • 1024 - 2047: emem1 or island 25, • 2048 - 3071: emem2 or island 26
mem_list_desc_in_mem_t1*	<i>mem_list_desc_in_mem</i>	40 bit pointer to list descriptor in external memory including the MU island. The pointer must be 16 byte aligned.

3.19.3.3 mem_list_read_desc_t2

Prototype:

```
void mem_list_read_desc_t2(unsigned int q_array_number, mem_list_desc_in_mem_t2
*mem_list_desc_in_mem)
```

Description:

Read MEM linked list queue descriptor for type 2 lists from MEM into queue array.

Table 3.533. mem_list_read_desc_t2 parameters

Type	Name	Description
unsigned int	<i>q_array_number</i>	queue array number in which to load. The range of the array number identifies the external memory to use. <ul style="list-style-type: none"> • 0 - 1023: emem0 or island 24, • 1024 - 2047: emem1 or island 25, • 2048 - 3071: emem2 or island 26
mem_list_desc_in_mem_t2*	<i>mem_list_desc_in_mem</i>	40 bit pointer to list descriptor in external memory including the MU island. The pointer must be 16 byte aligned.

3.19.3.4 mem_list_read_desc_t3

Prototype:

```
void mem_list_read_desc_t3(unsigned int q_array_number, mem_list_desc_in_mem_t3
*mem_list_desc_in_mem)
```

Description:

Read MEM linked list queue descriptor for type 3 lists from MEM into queue array.

Table 3.534. mem_list_read_desc_t3 parameters

Type	Name	Description
unsigned int	<i>q_array_number</i>	queue array number in which to load. The range of the array number identifies the external memory to use. <ul style="list-style-type: none"> • 0 - 1023: emem0 or island 24, • 1024 - 2047: emem1 or island 25, • 2048 - 3071: emem2 or island 26
mem_list_desc_in_mem_t3*	<i>mem_list_desc_in_mem</i>	40 bit pointer to list descriptor in external memory including the MU island. The pointer must be 16 byte aligned.

3.19.3.5 mem_list_push_desc_t0

Prototype:

```
void mem_list_push_desc_t0(unsigned int q_array_number, __xread mem_list_desc_t0 *xfer,
sync_t sync, SIGNAL* sig_ptr)
```

Description:

Push MEM queue descriptor from queue array into `mem_list_desc_t0` residing in transfer registers.

Table 3.535. mem_list_push_desc_t0 parameters

Type	Name	Description
unsigned int	<code>q_array_number</code>	queue array number in which to push. The range of the array number identifies the external memory to use. <ul style="list-style-type: none"> • 0 - 1023: emem0 or island 24, • 1024 - 2047: emem1 or island 25, • 2048 - 3071: emem2 or island 26
<code>__xread mem_list_desc_t0*</code>	<code>xfer</code>	Pointer to read xfer registers where the descriptor will be written
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.19.3.6 mem_list_push_desc_t1

Prototype:

```
void mem_list_push_desc_t1(unsigned int q_array_number, __xread mem_list_desc_t1 *xfer,
                           sync_t sync, SIGNAL* sig_ptr)
```

Description:

Push MEM queue descriptor from queue array into `mem_list_desc_t1` residing in transfer registers.

Table 3.536. mem_list_push_desc_t1 parameters

Type	Name	Description
unsigned int	<code>q_array_number</code>	queue array number in which to push. The range of the array number identifies the external memory to use. <ul style="list-style-type: none"> • 0 - 1023: emem0 or island 24, • 1024 - 2047: emem1 or island 25, • 2048 - 3071: emem2 or island 26
<code>__xread mem_list_desc_t1*</code>	<code>xfer</code>	Pointer to read xfer registers where the descriptor will be written
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.19.3.7 mem_list_push_desc_t2

Prototype:

```
void mem_list_push_desc_t2(unsigned int q_array_number, __xread mem_list_desc_t2 *xfer,
sync_t sync, SIGNAL* sig_ptr)
```

Description:

Push MEM queue descriptor from queue array into `mem_list_desc_t2` residing in transfer registers.

Table 3.537. mem_list_push_desc_t2 parameters

Type	Name	Description
unsigned int	<code>q_array_number</code>	queue array number in which to push. The range of the array number identifies the external memory to use. <ul style="list-style-type: none"> • 0 - 1023: emem0 or island 24, • 1024 - 2047: emem1 or island 25, • 2048 - 3071: emem2 or island 26
<code>__xread mem_list_desc_t2*</code>	<code>xfer</code>	Pointer to read xfer registers where the descriptor will be written
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.19.3.8 mem_list_push_desc_t3

Prototype:

```
void mem_list_push_desc_t3(unsigned int q_array_number, __xread mem_list_desc_t3 *xfer,
sync_t sync, SIGNAL* sig_ptr)
```

Description:

Push MEM queue descriptor from queue array into `mem_list_desc_t3` residing in transfer registers.

Table 3.538. mem_list_push_desc_t3 parameters

Type	Name	Description
unsigned int	<code>q_array_number</code>	queue array number in which to push. The range of the array number identifies the external memory to use. <ul style="list-style-type: none"> • 0 - 1023: emem0 or island 24, • 1024 - 2047: emem1 or island 25, • 2048 - 3071: emem2 or island 26

Type	Name	Description
<code>__xread mem_list_desc_t3*</code>	<code>xfer</code>	Pointer to read xfer registers where the descriptor will be written
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.19.3.9 mem_list_write_desc_t0

Prototype:

```
void mem_list_write_desc_t0(unsigned int q_array_number, mem_list_desc_in_mem_t0
*mem_list_desc_in_mem)
```

Description:

Write MEM queue descriptor from queue array into MEM for a type 0 list.

Table 3.539. mem_list_write_desc_t0 parameters

Type	Name	Description
<code>unsigned int</code>	<code>q_array_number</code>	queue array number to write to memory. The range of the array number identifies the external memory to use. <ul style="list-style-type: none"> • 0 - 1023: emem0 or island 24, • 1024 - 2047: emem1 or island 25, • 2048 - 3071: emem2 or island 26
<code>mem_list_desc_in_mem_t0*</code>	<code>mem_list_desc_in_mem</code>	40 bit pointer to descriptor in external memory including the MU island where the descriptor will be written

3.19.3.10 mem_list_write_desc_t1

Prototype:

```
void mem_list_write_desc_t1(unsigned int q_array_number, mem_list_desc_in_mem_t1
*mem_list_desc_in_mem)
```

Description:

Write MEM queue descriptor from queue array into MEM for a type 1 list.

Table 3.540. mem_list_write_desc_t1 parameters

Type	Name	Description
unsigned int	<i>q_array_number</i>	queue array number to write to memory. The range of the array number identifies the external memory to use. <ul style="list-style-type: none">• 0 - 1023: emem0 or island 24,• 1024 - 2047: emem1 or island 25,• 2048 - 3071: emem2 or island 26
mem_list_desc_in_mem_t1*	<i>mem_list_desc_in_mem</i>	40 bit pointer to descriptor in external memory including the MU island where the descriptor will be written

3.19.3.11 mem_list_write_desc_t2

Prototype:

```
void mem_list_write_desc_t2(unsigned int q_array_number, mem_list_desc_in_mem_t2
*mem_list_desc_in_mem)
```

Description:

Write MEM queue descriptor from queue array into MEM for a type 2 list.

Table 3.541. mem_list_write_desc_t2 parameters

Type	Name	Description
unsigned int	<i>q_array_number</i>	queue array number in to write to memory. The range of the array number identifies the external memory to use. <ul style="list-style-type: none">• 0 - 1023: emem0 or island 24,• 1024 - 2047: emem1 or island 25,• 2048 - 3071: emem2 or island 26
mem_list_desc_in_mem_t2*	<i>mem_list_desc_in_mem</i>	40 bit pointer to descriptor in external memory including the MU island where the descriptor will be written.

3.19.3.12 mem_list_write_desc_t3

Prototype:

```
void mem_list_write_desc_t3(unsigned int q_array_number, mem_list_desc_in_mem_t3
*mem_list_desc_in_mem)
```

Description:

Write MEM queue descriptor from queue array into MEM for a type 3 list.

Table 3.542. mem_list_write_desc_t3 parameters

Type	Name	Description
unsigned int	<i>q_array_number</i>	queue array number to write to memory. The range of the array number identifies the external memory to use. <ul style="list-style-type: none"> • 0 - 1023: emem0 or island 24, • 1024 - 2047: emem1 or island 25, • 2048 - 3071: emem2 or island 26
mem_list_desc_in_mem_t3*	<i>mem_list_desc_in_mem</i>	40 bit pointer to descriptor in external memory including the MU island where the descriptor will be written.

3.19.3.13 mem_list_enqueue_t0

Prototype:

```
void mem_list_enqueue_t0(unsigned int q_array_number, unsigned int sop, unsigned int eop,
                        unsigned int seg_cnt, mem_list_link_in_mem_t0 *next_ptr)
```

Description:

Add an entry to the tail of type 0 list.

If the list is empty the head and tail pointers are set to point to the next_ptr parameter and the q_count in the Q array to 1.

If there is an entry on the list then the new entry is linked to the tail before the tail is set to the next parameter. The q_count in the Q array is incremented.

Table 3.543. mem_list_enqueue_t0 parameters

Type	Name	Description
unsigned int	<i>q_array_number</i>	queue array number in which to enqueue entry. The range of the array number identifies the external memory to use. <ul style="list-style-type: none"> • 0 - 1023: emem0 or island 24, • 1024 - 2047: emem1 or island 25, • 2048 - 3071: emem2 or island 26
unsigned int	<i>sop</i>	start of packet indication
unsigned int	<i>eop</i>	end of packet indication

Type	Name	Description
unsigned int	<i>seg_cnt</i>	segment count
mem_list_link_in_mem_t0*	<i>next_ptr</i>	40 bit pointer to next list descriptor in external memory including the MU island. The pointer must be 16 byte aligned.

3.19.3.14 mem_list_enqueue_t1

Prototype:

```
void mem_list_enqueue_t1(unsigned int q_array_number, unsigned int sop, unsigned int eop,
                        unsigned int user_data, mem_list_link_in_mem_t1 *next_ptr)
```

Description:

Add an entry to the tail of a type 1 list.

If the list is empty the head and tail pointers are set to point to the *next_ptr* parameter and the *q_count* in the Q array to 1. The *sop*, *eop* and *user_data* fields are set from the call.

If there is already an entry on the list then the new entry is linked to the tail before the tail is set to the *next* parameter. The *q_count* in the Q array is incremented.

Table 3.544. mem_list_enqueue_t1 parameters

Type	Name	Description
unsigned int	<i>q_array_number</i>	queue array number in which to enqueue entry. The range of the array number identifies the external memory to use. <ul style="list-style-type: none"> • 0 - 1023: emem0 or island 24, • 1024 - 2047: emem1 or island 25, • 2048 - 3071: emem2 or island 26
unsigned int	<i>sop</i>	start of packet indication
unsigned int	<i>eop</i>	end of packet indication
unsigned int	<i>user_data</i>	User Data (6 bits)
mem_list_link_in_mem_t1*	<i>next_ptr</i>	40 bit pointer to next list descriptor in external memory including the MU island. The pointer must be 16 byte aligned.

3.19.3.15 mem_list_enqueue_t2

Prototype:

```
void mem_list_enqueue_t2(unsigned int q_array_number, unsigned int sop, unsigned int eop,
                        mem_list_link_in_mem_t2 *next_ptr)
```

Description:

Add an entry to the tail of a type 2 list.

If the list is empty the head and tail pointers are set to point to the next_ptr parameter and the q_count in the Q array to 1. sop and eop is set from the call.

If there is already an entry on the list then the new entry is linked to the tail before the tail is set to the next parameter. The q_count in the Q array is incremented.

Table 3.545. mem_list_enqueue_t2 parameters

Type	Name	Description
unsigned int	<i>q_array_number</i>	queue array number in which to enqueue entry. The range of the array number identifies the external memory to use. <ul style="list-style-type: none"> • 0 - 1023: emem0 or island 24, • 1024 - 2047: emem1 or island 25, • 2048 - 3071: emem2 or island 26
unsigned int	<i>sop</i>	start of packet indication
unsigned int	<i>eop</i>	end of packet indication
mem_list_link_in_mem_t2*	<i>next_ptr</i>	40 bit pointer to next list descriptor in external memory including the MU island. The pointer must be 16 byte aligned.

3.19.3.16 mem_list_enqueue_t3

Prototype:

```
void mem_list_enqueue_t3(unsigned int q_array_number, unsigned int sop, unsigned int eop,
                        unsigned int seg_cnt, mem_list_link_in_mem_t3 *next_ptr)
```

Description:

Add an entry to the tail of a type 3 list.

If the list is empty the head and tail pointers are set to point to the next_ptr parameter and the q_count in the Q array to 1.

If there is an entry on the list then the new entry is linked to the tail before the tail is set to the next parameter. The q_count is set from the seg_cnt which must have a value between 1 and 63.

Table 3.546. mem_list_enqueue_t3 parameters

Type	Name	Description
unsigned int	<i>q_array_number</i>	queue array number in which to enqueue entry. The range of the array number identifies the external memory to use. <ul style="list-style-type: none"> • 0 - 1023: emem0 or island 24, • 1024 - 2047: emem1 or island 25,

Type	Name	Description
		• 2048 - 3071: emem2 or island 26
unsigned int	sop	start of packet indication
unsigned int	eop	end of packet indication
unsigned int	seg_cnt	segment count
mem_list_link_in_mem_t3*	next_ptr	40 bit pointer to next list descriptor in external memory including the MU island. The pointer must be 16 byte aligned.

3.19.3.17 mem_list_enqueue_tail_t0

Prototype:

```
void mem_list_enqueue_tail_t0(unsigned int q_array_number, mem_list_link_in_mem_t0 *tail)
```

Description:

Update the tail pointer of a type 0 list.

This function is typically used when a list of packets is constructed in memory and after being enqueued the tail pointer in the q_array needs to be updated.

Table 3.547. mem_list_enqueue_tail_t0 parameters

Type	Name	Description
unsigned int	q_array_number	queue array number in which to enqueue entry. The range of the array number identifies the external memory to use. <ul style="list-style-type: none">• 0 - 1023: emem0 or island 24,• 1024 - 2047: emem1 or island 25,• 2048 - 3071: emem2 or island 26
mem_list_link_in_mem_t0*	tail	40 bit pointer to the tail of the list in external memory including the MU island. The pointer must be 16 byte aligned.

3.19.3.18 mem_list_enqueue_tail_t1

Prototype:

```
void mem_list_enqueue_tail_t1(unsigned int q_array_number, mem_list_link_in_mem_t1 *tail)
```

Description:

Update the tail pointer of a type 1 list.

This function is typically used when a list of packets is constructed in memory and after being enqueued the tail pointer in the q_array needs to be updated.

Table 3.548. mem_list_enqueue_tail_t1 parameters

Type	Name	Description
unsigned int	<i>q_array_number</i>	queue array number in which to enqueue entry. The range of the array number identifies the external memory to use. <ul style="list-style-type: none"> • 0 - 1023: emem0 or island 24, • 1024 - 2047: emem1 or island 25, • 2048 - 3071: emem2 or island 26
mem_list_link_in_mem_t1*	<i>tail</i>	40 bit pointer to the tail of the list in external memory including the MU island. The pointer must be 16 byte aligned.

3.19.3.19 mem_list_enqueue_tail_t2

Prototype:

```
void mem_list_enqueue_tail_t2(unsigned int q_array_number, mem_list_link_in_mem_t2 *tail)
```

Description:

Update the tail pointer of a type 2 list.

This function is used to set the tail pointer in the q_array.

Table 3.549. mem_list_enqueue_tail_t2 parameters

Type	Name	Description
unsigned int	<i>q_array_number</i>	queue array number in which to enqueue entry. The range of the array number identifies the external memory to use. <ul style="list-style-type: none"> • 0 - 1023: emem0 or island 24, • 1024 - 2047: emem1 or island 25, • 2048 - 3071: emem2 or island 26
mem_list_link_in_mem_t2*	<i>tail</i>	40 bit pointer to the tail of the list in external memory including the MU island. The pointer must be 16 byte aligned.

3.19.3.20 mem_list_enqueue_tail_t3

Prototype:

```
void mem_list_enqueue_tail_t3(unsigned int q_array_number, mem_list_link_in_mem_t3 *tail)
```

Description:

Update the tail pointer of a type 3 list.

This function is typically used when a list of packets is constructed in memory and after being enqueued the tail pointer in the q_array needs to be updated.

Table 3.550. mem_list_enqueue_tail_t3 parameters

Type	Name	Description
unsigned int	<i>q_array_number</i>	queue array number in which to enqueue entry. The range of the array number identifies the external memory to use. <ul style="list-style-type: none"> • 0 - 1023: emem0 or island 24, • 1024 - 2047: emem1 or island 25, • 2048 - 3071: emem2 or island 26
mem_list_link_in_mem_t3*	<i>tail</i>	40 bit pointer to the tail of the list in external memory including the MU island. The pointer must be 16 byte aligned.

3.19.3.21 mem_list_dequeue_t0

Prototype:

```
void mem_list_dequeue_t0(__xread mem_list_link_t0 *mem_list_link, unsigned int
q_array_number, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Remove an entry from the head of a type 0 list.

This function decrements the segment count value and only removes the entry from the list when the segment count is zero. The queue count is only decremented when the EOP bit is set.

Table 3.551. mem_list_dequeue_t0 parameters

Type	Name	Description
__xread mem_list_link_t0*	<i>mem_list_link</i>	Pointer to read xfer registers where dequeued entry will be written.
unsigned int	<i>q_array_number</i>	queue array number to dequeue from. The range of the array number identifies the external memory to use. <ul style="list-style-type: none"> • 0 - 1023: emem0 or island 24, • 1024 - 2047: emem1 or island 25, • 2048 - 3071: emem2 or island 26
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.19.3.22 mem_list_dequeue_t1

Prototype:

```
void mem_list_dequeue_t1(__xread mem_list_link_t1 *mem_list_link, unsigned int
q_array_number, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Remove an entry from the head of a type 1 list.

This function always removes the entry from the head of the list, but only decrements the queue count when the EOP bit is set.

Table 3.552. mem_list_dequeue_t1 parameters

Type	Name	Description
<code>__xread mem_list_link_t1*</code>	<code>mem_list_link</code>	Pointer to read xfer registers where dequeued entry will be written.
<code>unsigned int</code>	<code>q_array_number</code>	queue array number to dequeue from. The range of the array number identifies the external memory to use. <ul style="list-style-type: none"> • 0 - 1023: emem0 or island 24, • 1024 - 2047: emem1 or island 25, • 2048 - 3071: emem2 or island 26
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.19.3.23 mem_list_dequeue_t2

Prototype:

```
void mem_list_dequeue_t2(__xread mem_list_link_t2 *mem_list_link, unsigned int
q_array_number, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Remove an entry from the head of a type 2 list.

The dequeue command always removes the entry from the head of the list, but only decrements the queue count when the EOP bit for the entry was set on the enqueue command. The values for SOP and EOP in the link returned from the dequeue command, are inverted from the values used for the enqueue command.

Table 3.553. mem_list_dequeue_t2 parameters

Type	Name	Description
<code>__xread mem_list_link_t2*</code>	<code>mem_list_link</code>	Pointer to read xfer registers where dequeued entry will be written.
<code>unsigned int</code>	<code>q_array_number</code>	queue array number to dequeue from. The range of the array number identifies the external memory to use. <ul style="list-style-type: none"> • 0 - 1023: emem0 or island 24, • 1024 - 2047: emem1 or island 25, • 2048 - 3071: emem2 or island 26
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.19.3.24 mem_list_dequeue_t3

Prototype:

```
void mem_list_dequeue_t3(__xread mem_list_link_t3 *mem_list_link, unsigned int
q_array_number, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Remove an entry from the head of a type 3 list.

This function always removes the entry from the head of the list and decrements the queue count by one. The values for SOP and EOP are the same as used for the enqueue command.

Table 3.554. mem_list_dequeue_t3 parameters

Type	Name	Description
<code>__xread mem_list_link_t3*</code>	<code>mem_list_link</code>	Pointer to read xfer registers where dequeued entry will be written.
<code>unsigned int</code>	<code>q_array_number</code>	queue array number to dequeue from. The range of the array number identifies the external memory to use. <ul style="list-style-type: none"> • 0 - 1023: emem0 or island 24, • 1024 - 2047: emem1 or island 25, • 2048 - 3071: emem2 or island 26
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.19.3.25 mem_list_init

Prototype:

```
void mem_list_init(unsigned int q_array_number, enum MEM_LIST_TYPE type)
```

Description:

Initialize a memory list.

The following example shows how to create a list of type 3 in emem0 (i24). Note the setup of the q_array_number to indicate that the list is in emem0.

```
// buf1 is on emem0 (i24)
__emem_n(0) __addr40 mem_list_link_in_mem_t3  buf1;
SIGNAL                                     sig;
__xread  mem_list_desc_t3                  descriptor;

// list is on emem0 (i24)
unsigned int                                q_array_number = 1023;

mem_list_init(q_array_number,TYPE_3);

// enqueue the entry and verify list count and list type
{
    unsigned int      sop = 0;
    unsigned int      eop = 1;
    unsigned int      cell_count = 1;

    mem_list_enqueue_t3(q_array_number, sop, eop, cell_count, &buf1);
    mem_list_push_desc_t3(q_array_number, &descriptor, ctx_swap, &sig);

    if (descriptor.q_count != 1)
    {
        return 0;           // We have an error
    }
    if (descriptor.q_type != TYPE_3)
    {
        return 0;           // We have an error
    }
}

// dequeue the entry and verify sop and eop
{
    __xread mem_list_link_t3 result;

    mem_list_dequeue_t3(&result,q_array_number,sig_done,&sig);
    wait_for_all(&sig);

    if (result.sop != 0 || result.eop != 1)
    {
        return 0;           // We have an error
    }

    mem_list_push_desc_t3(q_array_number, &descriptor, ctx_swap, &sig);

    if (descriptor.q_count != 0)
    {
        return 0;           // We have an error
    }
}
```

```
}
```

```
return 1;
```

The following code can be used to create a list on any emem island:

```
unsigned int emem_island = 1; // 0 = emem0, 1 = emem1, 2 = emem2
unsigned int q_array_number = emem_island << 10 | 1023; // list with q_array_nr 1023 on emem1
```

Table 3.555. mem_list_init parameters

Type	Name	Description
unsigned int	<i>q_array_number</i>	queue array number to use for list. The range of the array number identifies the external memory to use. <ul style="list-style-type: none"> • 0 - 1023: emem0 or island 24, • 1024 - 2047: emem1 or island 25, • 2048 - 3071: emem2 or island 26
enum MEM_LIST_TYPE	<i>type</i>	List type

3.19.3.26 mem_list_init_with_loc_and_page

Prototype:

```
void mem_list_init_with_loc_and_page(unsigned int q_array_number, enum MEM_LIST_TYPE
type, unsigned int loc, unsigned int page)
```

Description:

Initialize a memory list specifying the page and locality of the list.

Table 3.556. mem_list_init_with_loc_and_page parameters

Type	Name	Description
unsigned int	<i>q_array_number</i>	queue array number to use for list. The range of the array number identifies the external memory to use. <ul style="list-style-type: none"> • 0 - 1023: emem0 or island 24, • 1024 - 2047: emem1 or island 25, • 2048 - 3071: emem2 or island 26
enum MEM_LIST_TYPE	<i>type</i>	List type
unsigned int	<i>loc</i>	Locality specification of the list. Please refer to the PRM for detail
unsigned int	<i>page</i>	4G page in which the list needs to reside.

3.20 MEM MicroQ Intrinsics

This section discusses the enumerations, structs, unions, types and functions related to MEM MicroQ operations.

The micro queue intrinsic functions make provision for 128 or 256 bit queues.

Micro queues can be configured to store 16 or 32-bit words. A 128 bit micro queue can queue 6, 16 bit words or 3, 32-bit words and a 256 bit micro queue can contain 14, 16 bit words and 7, 32-bit words.

Data is queued (at the tail) using the put function. Data can be read from head of the queue using the get function or from the tail using the pop function.

Initialization functions are available to set up micro queues before first use.

3.20.1 MU MicroQ Enumerations

3.20.1.1 MICROQ_ENTRY_SIZE

Micro queue entry sizes.

0 - 16 bit 1 - 32 bit

Table 3.557. enum MICROQ_ENTRY_SIZE

Name	Description
MICROQ_ENTRY_SIZE_16	16 bit
MICROQ_ENTRY_SIZE_32	32 bit

3.20.2 MU MicroQ Structs

3.20.2.1 mem_microq128_t

128 bit micro queue container type.

The first word is the queue descriptor

Table 3.558. struct mem_microq128_t

Type	Name	Description
mem_microq_desc_t	desc	Micro queue descriptor.
unsigned int	q[3]	3 words for queue contents

See Also:

- `mem_microq_desc_t`

3.20.2.2 `mem_microq256_t`

256 bit micro queue container type.

The first word is the queue descriptor

Table 3.559. struct `mem_microq256_t`

Type	Name	Description
<code>mem_microq_desc_t</code>	<code>desc</code>	Micro queue descriptor.
<code>unsigned int</code>	<code>q[7]</code>	7 words for queue contents

See Also:

- `mem_microq_desc_t`

3.20.3 MU MicroQ Unions

3.20.3.1 `mem_microq_desc_t`

Micro queue descriptor.

The descriptor is the first word in a micro queue structure and contains the configuration and state of the micro queue.

Table 3.560. union `mem_microq_desc_t`

Type	Name	Description
<code>unsigned int</code>	<code>reserved_1:4</code>	Reserved.
<code>unsigned int</code>	<code>event_source:12</code>	Event source number to include in event.
<code>unsigned int</code>	<code>reserved_2:6</code>	Reserved.
<code>unsigned int</code>	<code>generate_event:1</code>	Generate event on under or overflow.
<code>unsigned int</code>	<code>entry_size:1</code>	One of <code>MICROQ_ENTRY_SIZE</code> .
<code>unsigned int</code>	<code>overflow:1</code>	Set on overflow.
<code>unsigned int</code>	<code>underflow:1</code>	Set on underflow.
<code>unsigned int</code>	<code>reserved_3:1</code>	Reserved.
<code>unsigned int</code>	<code>locked:1</code>	Used for queue lock queues. When this bit is set, it indicates that the lock has been given out.

Type	Name	Description
unsigned int	num_elements:4	0-3 or 0-6 or 0-14 depending on entry_size and if it is quarter or half cache line.
unsigned int	value	Accessor to entire structure.

3.20.4 MU MicroQ Typedefs

3.20.4.1 MICROQ_ENTRY_SIZE

Micro queue entry sizes.

0 - 16 bit 1 - 32 bit

Table 3.561. typedef MICROQ_ENTRY_SIZE

Type	Definition
MICROQ_ENTRY_SIZE	enum MICROQ_ENTRY_SIZE

3.20.4.2 mem_microq_desc_t

Micro queue descriptor.

The descriptor is the first word in a micro queue structure and contains the configuration and state of the micro queue.

Table 3.562. typedef mem_microq_desc_t

Type	Definition
mem_microq_desc_t	union mem_microq_desc_t

3.20.4.3 mem_microq128_t

128 bit micro queue container type.

The first word is the queue descriptor

Table 3.563. typedef mem_microq128_t

Type	Definition
mem_microq128_t	struct mem_microq128_t

See Also:

- mem_microq_desc_t

3.20.4.4 mem_microq256_t

256 bit micro queue container type.

The first word is the queue descriptor

Table 3.564. typedef mem_microq256_t

Type	Definition
mem_microq256_t	struct mem_microq256_t

See Also:

- mem_microq_desc_t

3.20.4.5 mem_microq128_in_mem_t

128-bit microqueue aligned in 32-bit addressed memory.

Table 3.565. typedef mem_microq128_in_mem_t

Type	Definition
mem_microq128_in_mem_t	volatile __addr32 __align16 mem_microq128_t

3.20.4.6 mem_microq128_ptr40_t

128-bit microqueue aligned in 40-bit addressed memory.

Table 3.566. typedef mem_microq128_ptr40_t

Type	Definition
mem_microq128_ptr40_t	volatile __mem __addr40 __align16 mem_microq128_t*

3.20.4.7 mem_microq256_in_mem_t

256-bit microqueue aligned in 32-bit addressed memory.

Table 3.567. typedef mem_microq256_in_mem_t

Type	Definition
mem_microq256_in_mem_t	volatile __addr32 __align32 mem_microq256_t

3.20.4.8 mem_microq256_ptr40_t

256-bit microqueue aligned in memory 40-bit addressed memory.

Table 3.568. `typedef mem_microq256_ptr40_t`

Type	Definition
<code>mem_microq256_ptr40_t</code>	<code>volatile __mem __addr40 __align32 mem_microq256_t*</code>

3.20.5 MU MicroQ Functions

3.20.5.1 `mem_microq128_put_ptr32`

Prototype:

```
void mem_microq128_put_ptr32(__mem mem_microq128_in_mem_t* microq, __xwrite void* xfer,
sync_t sync, SIGNAL* sig_ptr)
```

Description:

Add data to the tail of a 128-bit micro queue in 32-bit addressed MEM.

If there are not enough space, the data is discarded and depending on the micro queue configuration an event is presented on the event bus. 16 bits or 32 bits of the transfer register is written to the micro queue based on the configuration of the micro queue.

Table 3.569. `mem_microq128_put_ptr32` parameters

Type	Name	Description
<code>__mem mem_microq128_in_mem_t*</code>	<code>microq</code>	Pointer to micro queue in MEM
<code>__xwrite void*</code>	<code>xfer</code>	Transfer register containing the data
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.20.5.2 `mem_microq128_put_ptr40`

Prototype:

```
void mem_microq128_put_ptr40(mem_microq128_ptr40_t microq, __xwrite void* xfer, sync_t
sync, SIGNAL* sig_ptr)
```

Description:

Add data to the tail of a 128-bit micro queue in 40-bit addressed MEM.

If there are not enough space, the data is discarded and depending on the micro queue configuration an event is presented on the event bus. 16 bits or 32 bits of the transfer register is written to the micro queue based on the configuration of the micro queue.

Table 3.570. mem_microq128_put_ptr40 parameters

Type	Name	Description
mem_microq128_ptr40_t	<i>microq</i>	Pointer to micro queue in MEM
__xwrite void*	<i>xfer</i>	Transfer register containing the data
sync_t	<i>sync</i>	Type of synchronization to use
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.20.5.3 mem_microq256_put_ptr32

Prototype:

```
void mem_microq256_put_ptr32(__mem mem_microq256_in_mem_t* microq, __xwrite void* xfer,
sync_t sync, SIGNAL* sig_ptr)
```

Description:

Add data to the tail of a 256-bit micro queue in 32-bit addressed MEM.

If there are not enough space, the data is discarded and depending on the micro queue configuration an event is presented on the event bus. 16 bits or 32 bits of the transfer register is written to the micro queue based on the configuration of the micro queue.

Table 3.571. mem_microq256_put_ptr32 parameters

Type	Name	Description
__mem mem_microq256_in_mem_t*	<i>microq</i>	Pointer to micro queue in MEM
__xwrite void*	<i>xfer</i>	Transfer register containing the data
sync_t	<i>sync</i>	Type of synchronization to use
SIGNAL*	<i>sig_ptr</i>	Signal to raised upon completion

3.20.5.4 mem_microq256_put_ptr40

Prototype:

```
void mem_microq256_put_ptr40(mem_microq256_ptr40_t microq, __xwrite void* xfer, sync_t
sync, SIGNAL* sig_ptr)
```

Description:

Add data to the tail of a 256-bit micro queue in 40-bit addressed MEM .

If there are not enough space, the data is discarded and depending on the micro queue configuration an event is presented on the event bus. 16 bits or 32 bits of the transfer register is written to the micro queue based on the configuration of the micro queue.

Table 3.572. mem_microq256_put_ptr40 parameters

Type	Name	Description
mem_microq256_ptr40_t	<i>microq</i>	Pointer to micro queue in MEM
__xwrite void*	<i>xfer</i>	Transfer register containing the data
sync_t	<i>sync</i>	Type of synchronization to use
SIGNAL*	<i>sig_ptr</i>	Signal to raised upon completion

3.20.5.5 mem_microq128_get_ptr32

Prototype:

```
void mem_microq128_get_ptr32(__mem mem_microq128_in_mem_t* microq, __xread void* xfer,
sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Remove data from the head of a 128-bit micro queue in 32-bit addressed MEM and place data in transfer register.

If there are not enough entries in the micro queue, a zero long word is returned and an odd signal will also be delivered.

Table 3.573. mem_microq128_get_ptr32 parameters

Type	Name	Description
__mem mem_microq128_in_mem_t*	<i>microq</i>	Pointer to micro queue in MEM
__xread void*	<i>xfer</i>	Transfer register receiving the data
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Even signal is raised upon completion. Odd signal is raised on underflow.

3.20.5.6 mem_microq128_get_ptr40

Prototype:

```
void mem_microq128_get_ptr40(mem_microq128_ptr40_t microq, __xread void* xfer, sync_t
sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Remove data from the head of a 128-bit micro queue in 40-bit addressed MEM and place data in transfer register.

If there are not enough entries in the micro queue, a zero long word is returned and an odd signal will also be delivered.

Table 3.574. mem_microq128_get_ptr40 parameters

Type	Name	Description
mem_microq128_ptr40_t	<i>microq</i>	Pointer to micro queue in MEM
__xread void*	<i>xfer</i>	Transfer register receiving the data
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Even signal is raised upon completion. Odd signal is raised on underflow.

3.20.5.7 mem_microq256_get_ptr32

Prototype:

```
void mem_microq256_get_ptr32(__mem mem_microq256_in_mem_t* microq, __xread void* xfer,
sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Remove data from the head of a 128-bit micro queue in 32-bit addressed MEM and place data in transfer register.

If there are not enough entries in the micro queue, a zero long word is returned and an odd signal will also be delivered.

Table 3.575. mem_microq256_get_ptr32 parameters

Type	Name	Description
__mem mem_microq256_in_mem_t*	<i>microq</i>	Pointer to micro queue in MEM
__xread void*	<i>xfer</i>	Transfer register receiving the data
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Even signal is raised upon completion. Odd signal is raised on underflow.

3.20.5.8 mem_microq256_get_ptr40

Prototype:

```
void mem_microq256_get_ptr40(mem_microq256_ptr40_t microq, __xread void* xfer, sync_t
sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Remove data from the head of a 128-bit micro queue in 40-bit addressed MEM and place data in transfer register.

If there are not enough entries in the micro queue, a zero long word is returned and an odd signal will also be delivered.

Table 3.576. mem_microq256_get_ptr40 parameters

Type	Name	Description
mem_microq256_ptr40_t	<i>microq</i>	Pointer to micro queue in MEM
__xread void*	<i>xfer</i>	Transfer register receiving the data
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Even signal is raised upon completion. Odd signal is raised on underflow.

3.20.5.9 mem_microq128_pop_ptr32

Prototype:

```
void mem_microq128_pop_ptr32(__mem mem_microq128_in_mem_t* microq, __xread void* xfer,
sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Remove data from the tail of a 128-bit micro queue in 32-bit addressed MEM and place data in transfer register.

If there are not enough entries in the micro queue, a zero long word is returned and an odd signal will also be delivered.

Table 3.577. mem_microq128_pop_ptr32 parameters

Type	Name	Description
__mem mem_microq128_in_mem_t*	<i>microq</i>	Pointer to micro queue in MEM
__xread void*	<i>xfer</i>	Transfer register receiving the data
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Even signal is raised upon completion. Odd signal is raised on underflow.

3.20.5.10 mem_microq128_pop_ptr40

Prototype:

```
void mem_microq128_pop_ptr40(mem_microq128_ptr40_t microq, __xread void* xfer, sync_t
sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Remove data from the tail of a 128-bit micro queue in 40-bit addressed MEM and place data in transfer register.

If there are not enough entries in the micro queue, a zero long word is returned and an odd signal will also be delivered.

Table 3.578. mem_microq128_pop_ptr40 parameters

Type	Name	Description
mem_microq128_ptr40_t	<i>microq</i>	Pointer to micro queue in MEM
__xread void*	<i>xfer</i>	Transfer register receiving the data
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Even signal is raised upon completion. Odd signal is raised on underflow.

3.20.5.11 mem_microq256_pop_ptr32

Prototype:

```
void mem_microq256_pop_ptr32(__mem mem_microq256_in_mem_t* microq, __xread void* xfer,
sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Remove data from the tail of a 256-bit micro queue in 32-bit addressed MEM and place data in transfer register.

If there are not enough entries in the micro queue, a zero long word is returned and an odd signal will also be delivered.

Table 3.579. mem_microq256_pop_ptr32 parameters

Type	Name	Description
__mem mem_microq256_in_mem_t*	<i>microq</i>	Pointer to micro queue in MEM
__xread void*	<i>xfer</i>	Transfer register receiving the data
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Even signal is raised upon completion. Odd signal is raised on underflow.

3.20.5.12 mem_microq256_pop_ptr40

Prototype:

```
void mem_microq256_pop_ptr40(mem_microq256_ptr40_t microq, __xread void* xfer, sync_t
sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Remove data from the tail of a 256-bit micro queue in 40-bit addressed MEM and place data in transfer register.

If there are not enough entries in the micro queue, a zero long word is returned and an odd signal will also be delivered.

Table 3.580. mem_microq256_pop_ptr40 parameters

Type	Name	Description
mem_microq256_ptr40_t	<i>microq</i>	Pointer to micro queue in MEM
__xread void*	<i>xfer</i>	Transfer register receiving the data
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Even signal is raised upon completion. Odd signal is raised on underflow.

3.20.5.13 mem_microq128_init_ptr32

Prototype:

```
void mem_microq128_init_ptr32(__mem mem_microq128_in_mem_t* microq, enum MICROQ_ENTRY_SIZE entry_size, unsigned int generate_event, unsigned int event_source)
```

Description:

Initialize a 128-bit micro queue in 32-bit addressed MEM.

The header of the micro queue is written and then read back to ensure the micro queue is configured when this function returns.

Table 3.581. mem_microq128_init_ptr32 parameters

Type	Name	Description
__mem mem_microq128_in_mem_t*	<i>microq</i>	Pointer to micro queue in MEM
enum MICROQ_ENTRY_SIZE	<i>entry_size</i>	Size of entries in the microqueue, 1 = 32-bit 0 = 16-bit
unsigned int	<i>generate_event</i>	Enable event on overflow or underflow
unsigned int	<i>event_source</i>	Event to present on event bus if generate_event is 1

3.20.5.14 mem_microq128_init_ptr40

Prototype:

```
void mem_microq128_init_ptr40(mem_microq128_ptr40_t microq, enum MICROQ_ENTRY_SIZE entry_size, unsigned int generate_event, unsigned int event_source)
```

Description:

Initialize a 128-bit micro queue in 40-bit addressed MEM.

The header of the micro queue is written and then read back to ensure the micro queue is configured when this function returns.

The following example shows how to create a 128-bit microq in emem0 (i24).

```

volatile __emem_n(0) __addr40 __align16 mem_microq128_t mq128_ptr40;
__xread unsigned int read_data;
__xwrite unsigned int write_data;
unsigned int counter;
int i;
SIGNAL sig;
SIGNAL_PAIR sig_pair;

mem_microq128_init_ptr40(&mq128_ptr40, MICROQ_ENTRY_SIZE_16, 0, 0);

write_data = 0x10001000;
mem_microq128_put_ptr40(&mq128_ptr40, &write_data, ctx_swap, &sig);

write_data = 0x20002000;
mem_microq128_put_ptr40(&mq128_ptr40, &write_data, ctx_swap, &sig);

// Verify the number of elements
if (mq128_ptr40.desc.num_elements != 2)
{
    return 0;           // We have an error
}

mem_microq128_pop_ptr40(&mq128_ptr40, &read_data, sig_done, &sig_pair);
wait_for_all_single(&sig_pair.event);

if (signal_test(&sig_pair.odd))
{
    return 0;           // We have an error
}

if (read_data != (0x20002000 & 0xffff))
{
    return 0;           // We have an error
}

return 1;

```

Table 3.582. mem_microq128_init_ptr40 parameters

Type	Name	Description
mem_microq128_ptr40_t	<i>microq</i>	Pointer to micro queue in MEM
enum MICROQ_ENTRY_SIZE	<i>entry_size</i>	Size of entries in the microqueue, 1 = 32-bit 0 = 16-bit
unsigned int	<i>generate_event</i>	Enable event on overflow or underflow
unsigned int	<i>event_source</i>	Event to present on event bus if generate_event is 1

3.20.5.15 mem_microq256_init_ptr32

Prototype:

```
void mem_microq256_init_ptr32(__mem mem_microq256_in_mem_t* microq, enum MICROQ_ENTRY_SIZE entry_size, unsigned int generate_event, unsigned int event_source)
```

Description:

Initialize a 256-bit micro queue in 32-bit addressed MEM.

The header of the micro queue is written and then read back to ensure the micro queue is configured when this function returns.

Table 3.583. mem_microq256_init_ptr32 parameters

Type	Name	Description
<code>__mem mem_microq256_in_mem_t*</code>	<code>microq</code>	Pointer to micro queue in MEM
<code>enum MICROQ_ENTRY_SIZE</code>	<code>entry_size</code>	Size of entries in the microqueue, 1 = 32-bit 0 = 16-bit
<code>unsigned int</code>	<code>generate_event</code>	Enable event on overflow or underflow
<code>unsigned int</code>	<code>event_source</code>	Event to present on event bus if generate_event is 1

3.20.5.16 mem_microq256_init_ptr40

Prototype:

```
void mem_microq256_init_ptr40(mem_microq256_ptr40_t microq, enum MICROQ_ENTRY_SIZE
entry_size, unsigned int generate_event, unsigned int event_source)
```

Description:

Initialize a 256-bit micro queue in 40-bit addressed MEM.

The header of the micro queue is written and then read back to ensure the micro queue is configured when this function returns.

Table 3.584. mem_microq256_init_ptr40 parameters

Type	Name	Description
<code>mem_microq256_ptr40_t</code>	<code>microq</code>	Pointer to micro queue in MEM
<code>enum MICROQ_ENTRY_SIZE</code>	<code>entry_size</code>	Size of entries in the microqueue, 1 = 32-bit 0 = 16-bit
<code>unsigned int</code>	<code>generate_event</code>	Enable event on overflow or underflow
<code>unsigned int</code>	<code>event_source</code>	Event to present on event bus if generate_event is 1

3.21 MEM CAM Intrinsics

This section discusses the enumerations, structs, unions, types and functions related to MEM CAM operations.

3.21.1 MU CAM Structs

3.21.1.1 mem_cam128_t

128-bit CAM container type.

Table 3.585. struct mem_cam128_t

Type	Name	Description
unsigned int	value[4]	Storage for the CAM data.

3.21.1.2 mem_cam256_t

256-bit CAM container type.

Table 3.586. struct mem_cam256_t

Type	Name	Description
unsigned int	value[8]	Storage for the CAM data.

3.21.1.3 mem_cam384_t

384-bit CAM container type.

Table 3.587. struct mem_cam384_t

Type	Name	Description
unsigned int	value[12]	Storage for the CAM data.

3.21.1.4 mem_cam512_t

512-bit CAM container type.

Table 3.588. struct mem_cam512_t

Type	Name	Description
unsigned int	value[16]	Storage for the CAM data.

3.21.1.5 mem_cam_lookup32_in_t

Input type for 32-bit CAM lookups.

This search field of this structure needs to be populated before performing a CAM lookup.

Table 3.589. struct mem_cam_lookup32_in_t

Type	Name	Description
unsigned int	search	Value to search for.

3.21.2 MU CAM Unions

3.21.2.1 mem_cam_lookup16_add_out_t

Output type of 16-bit CAM lookup and add operations.

The match-bit fields indicate which CAM entries matched the lookup.

Table 3.590. union mem_cam_lookup16_add_out_t

Type	Name	Description
unsigned int	reserved:24	Reserved.
unsigned int	added:1	When set, this field indicates that an entry was added. <div style="border: 1px solid blue; padding: 5px; margin-top: 10px;">  Note This field is also set on a CAM miss when it is full. </div>
unsigned int	first_match:7	First matched entry number, or on a CAM miss where the entry was added. <div style="border: 1px solid blue; padding: 5px; margin-top: 10px;">  Note All-bits in this field are set on a CAM miss when it is full. </div>
unsigned int	match_bitf	Bitfield of matching entries.
unsigned int	value[2]	Accessor to entire structure.

3.21.2.2 mem_cam_lookup16_in_t

Input type for 16-bit CAM lookups.

This search field of this structure needs to be populated before performing a CAM lookup.

Table 3.591. union mem_cam_lookup16_in_t

Type	Name	Description
unsigned int	reserved1:16	Reserved.

Type	Name	Description
unsigned int	search:16	16-bit value to search for in the CAM
unsigned int	padding	Additional word to ensure read data is same size as write data.
unsigned int	value[2]	Accessor to entire structure.

3.21.2.3 mem_cam_lookup16_out_t

Output type of 16-bit CAM lookups.

The match-bit fields indicate which CAM entries matched the lookup.

Table 3.592. union mem_cam_lookup16_out_t

Type	Name	Description
unsigned int	reserved:24	Reserved.
unsigned int	first_match:8	0xff - when not found
unsigned int	match_bitf	Bitfield of matching entries.
unsigned int	value[2]	Accessor to entire structure.

3.21.2.4 mem_cam_lookup24_add_inc_out_t

Output type of 24-bit CAM lookup, add and increment operations.

The match-bit fields indicate which CAM entries matched the lookup.

Table 3.593. union mem_cam_lookup24_add_inc_out_t

Type	Name	Description
unsigned int	match_bitf:16	Bits are set on matching entries.
unsigned int	count:8	Upper 8-bits of the matched CAM entry is returned in this field. The upper 8-bits is incremented in memory after the CAM operation.
unsigned int	added:1	When set, this field indicates that an entry was added.
 Note This field is also set on a CAM miss when it is full.		
unsigned int	first_match:7	First matched entry number, or on a CAM miss where the entry was added.

Type	Name	Description
		 Note All-bits in this field are set on a CAM miss when it is full.
unsigned int	value	Accessor to entire structure.

3.21.2.5 mem_cam_lookup24_add_in_t

Input type for 24-bit CAM lookup and add operations.

This search field of this structure needs to be populated before performing a CAM lookup. The upper 8-bits of the CAM entry is set from the high_byte field in this structure when a new entry is added on a CAM miss.



Note

24-bit CAM entries occupy 32-bits in MEM with 8-bits being available for user data or a count which is typically used with the 24-bit CAM add_inc functions.

Table 3.594. union mem_cam_lookup24_add_in_t

Type	Name	Description
unsigned int	high_byte:8	Value to set upper 8-bits in CAM to when entry is added.
unsigned int	search:24	24-bit value to search for in the CAM
unsigned int	value	Accessor to entire structure.

3.21.2.6 mem_cam_lookup24_add_out_t

Output type of 24-bit CAM lookup and add operations.

The match-bit fields indicate which CAM entries matched the lookup.

Table 3.595. union mem_cam_lookup24_add_out_t

Type	Name	Description
unsigned int	match_bitf:16	Bits are set on matching entries.
unsigned int	match_high_byte:8	Upper 8-bits of matched CAM entry is returned in this field.
unsigned int	added:1	When set, this field indicates that an entry was added.

Note

This field is also set on a CAM miss when it is full.

Type	Name	Description
unsigned int	first_match:7	First matched entry number, or on a CAM miss where the entry was added.
		<div style="border: 1px solid blue; padding: 5px;">  Note All-bits in this field are set on a CAM miss when it is full. </div>
unsigned int	value	Accessor to entire structure.

3.21.2.7 mem_cam_lookup24_in_t

Input type for 24-bit CAM lookups.



Note

24-bit CAM entries occupy 32-bits in MEM with 8-bits being available for user data or a count which is typically used with the 24-bit CAM add_inc functions.

This search field of this structure needs to be populated before performing a CAM lookup.

Table 3.596. union mem_cam_lookup24_in_t

Type	Name	Description
unsigned int	reserved1:8	Reserved.
unsigned int	search:24	24-bit value to search for in the CAM
unsigned int	value	Accessor to entire structure.

3.21.2.8 mem_cam_lookup24_out_t

Output type of 24-bit CAM lookups.

The match-bit fields indicate which CAM entries matched the lookup.

Table 3.597. union mem_cam_lookup24_out_t

Type	Name	Description
unsigned int	match_bitf:16	Bits are set on CAM entries matching.
unsigned int	match_high_byte:8	Upper 8-bits of matched CAM entry.
unsigned int	first_match:8	Entry number of first match lookup - 0xff when not found.
unsigned int	value	Accessor to entire structure.

3.21.2.9 mem_cam_lookup32_add_out_t

Output type of 32-bit CAM lookup and add operations.

The match-bit fields indicate which CAM entries matched the lookup.

Table 3.598. union mem_cam_lookup32_add_out_t

Type	Name	Description
unsigned int	match_bitf:16	Bits are set on matching entries.
unsigned int	match_high_byte:8	Upper 8-bits of the matched CAM entry is returned in this field.
unsigned int	added:1	When set, this field indicates that an entry was added. <div style="border: 1px solid blue; padding: 5px; margin-top: 10px;">  Note This field is also set on a CAM miss when it is full. </div>
unsigned int	first_match:7	First matched entry number, or on a CAM miss where the entry was added. <div style="border: 1px solid blue; padding: 5px; margin-top: 10px;">  Note All-bits in this field are set on a CAM miss when it is full. </div>
unsigned int	value	Accessor to entire structure.

3.21.2.10 mem_cam_lookup32_out_t

Output type of 32-bit CAM lookups.

The match-bit fields indicate which CAM entries matched the lookup.

Table 3.599. union mem_cam_lookup32_out_t

Type	Name	Description
unsigned int	match_bitf:16	Bits are set on matching entries.
unsigned int	match_high_byte:8	Upper 8-bits of the matched CAM entry is returned in this field.
unsigned int	first_match:8	First CAM entry that matched or 0xff - when not found.
unsigned int	value	Accessor to entire structure.

3.21.2.11 mem_cam_lookup8_add_out_t

Output type of 8-bit CAM lookup and add operations.

The match-bit fields indicate which CAM entries matched the lookup.

Table 3.600. union mem_cam_lookup8_add_out_t

Type	Name	Description
unsigned int	reserved:24	Reserved.
unsigned int	added:1	When set, this field indicates that an entry was added. <div style="border: 1px solid blue; padding: 5px; margin-top: 10px;"> Note  This field is also set on a CAM miss when it is full. </div>
unsigned int	first_match:7	First matched entry number, or on a CAM miss where the entry was added. <div style="border: 1px solid blue; padding: 5px; margin-top: 10px;"> Note  All-bits in this field are set on a CAM miss when it is full. </div>
unsigned int	match_bitf	Bitfield of matching entries.
unsigned int	value[2]	Accessor to entire structure.

3.21.2.12 mem_cam_lookup8_in_t

Input type for 8-bit CAM lookups.

This search field of this structure needs to be populated before performing a CAM lookup.

Table 3.601. union mem_cam_lookup8_in_t

Type	Name	Description
unsigned int	reserved1:24	Reserved.
unsigned int	search:8	8-bit value to search for in the CAM
unsigned int	value[2]	Accessor to entire structure.

3.21.2.13 mem_cam_lookup8_out_t

Output type of 8-bit CAM lookups.

The match-bit fields indicate which CAM entries matched the lookup.

Table 3.602. union mem_cam_lookup8_out_t

Type	Name	Description
unsigned int	match_bitf_low	Lower 32-bits of 64-bit field with ones indicating that a corresponding entry matched.
unsigned int	match_bitf_high	Upper 32-bits of 64-bit field with ones indicating that a corresponding entry matched.
unsigned int	value[2]	Accessor to entire structure.

3.21.3 MU CAM Typedefs

3.21.3.1 mem_cam128_t

128-bit CAM container type.

Table 3.603. typedef mem_cam128_t

Type	Definition
mem_cam128_t	struct mem_cam128_t

3.21.3.2 mem_cam256_t

256-bit CAM container type.

Table 3.604. typedef mem_cam256_t

Type	Definition
mem_cam256_t	struct mem_cam256_t

3.21.3.3 mem_cam384_t

384-bit CAM container type.

Table 3.605. typedef mem_cam384_t

Type	Definition
mem_cam384_t	struct mem_cam384_t

3.21.3.4 mem_cam512_t

512-bit CAM container type.

Table 3.606. `typedef mem_cam512_t`

Type	Definition
<code>mem_cam512_t</code>	<code>struct mem_cam512_t</code>

3.21.3.5 `mem_cam128_in_mem_t`

Table 3.607. `typedef mem_cam128_in_mem_t`

Type	Definition
<code>mem_cam128_in_mem_t</code>	<code>__addr32 __align16 mem_cam128_t</code>

3.21.3.6 `mem_cam256_in_mem_t`

Table 3.608. `typedef mem_cam256_in_mem_t`

Type	Definition
<code>mem_cam256_in_mem_t</code>	<code>__addr32 __align32 mem_cam256_t</code>

3.21.3.7 `mem_cam384_in_mem_t`

Table 3.609. `typedef mem_cam384_in_mem_t`

Type	Definition
<code>mem_cam384_in_mem_t</code>	<code>__addr32 __align64 mem_cam384_t</code>

3.21.3.8 `mem_cam512_in_mem_t`

Table 3.610. `typedef mem_cam512_in_mem_t`

Type	Definition
<code>mem_cam512_in_mem_t</code>	<code>__addr32 __align64 mem_cam512_t</code>

3.21.3.9 `mem_cam128_ptr40_t`

Table 3.611. `typedef mem_cam128_ptr40_t`

Type	Definition
<code>mem_cam128_ptr40_t</code>	<code>__mem __addr40 __align16 mem_cam128_t*</code>

3.21.3.10 mem_cam256_ptr40_t

Table 3.612. typedef mem_cam256_ptr40_t

Type	Definition
mem_cam256_ptr40_t	__mem __addr40 __align32 mem_cam256_t *

3.21.3.11 mem_cam384_ptr40_t

Table 3.613. typedef mem_cam384_ptr40_t

Type	Definition
mem_cam384_ptr40_t	__mem __addr40 __align64 mem_cam384_t *

3.21.3.12 mem_cam512_ptr40_t

Table 3.614. typedef mem_cam512_ptr40_t

Type	Definition
mem_cam512_ptr40_t	__mem __addr40 __align64 mem_cam512_t *

3.21.3.13 mem_cam_lookup8_in_t

Input type for 8-bit CAM lookups.

This search field of this structure needs to be populated before performing a CAM lookup.

Table 3.615. typedef mem_cam_lookup8_in_t

Type	Definition
mem_cam_lookup8_in_t	union mem_cam_lookup8_in_t

3.21.3.14 mem_cam_lookup8_out_t

Output type of 8-bit CAM lookups.

The match-bit fields indicate which CAM entries matched the lookup.

Table 3.616. typedef mem_cam_lookup8_out_t

Type	Definition
mem_cam_lookup8_out_t	union mem_cam_lookup8_out_t

3.21.3.15 mem_cam_lookup8_add_out_t

Output type of 8-bit CAM lookup and add operations.

The match-bit fields indicate which CAM entries matched the lookup.

Table 3.617. typedef mem_cam_lookup8_add_out_t

Type	Definition
mem_cam_lookup8_add_out_t	union mem_cam_lookup8_add_out_t

3.21.3.16 mem_cam_lookup8_out_in_read_reg_t

Type for mem_cam_lookup8_out_t in read registers.

Table 3.618. typedef mem_cam_lookup8_out_in_read_reg_t

Type	Definition
mem_cam_lookup8_out_in_read_reg_t	__xread mem_cam_lookup8_out_t

3.21.3.17 mem_cam_lookup8_add_out_in_read_reg_t

Type for mem_cam_lookup8_add_out_t in read registers.

Table 3.619. typedef mem_cam_lookup8_add_out_in_read_reg_t

Type	Definition
mem_cam_lookup8_add_out_in_read_reg_t	__xread mem_cam_lookup8_add_out_t

3.21.3.18 mem_cam_lookup16_in_t

Input type for 16-bit CAM lookups.

This search field of this structure needs to be populated before performing a CAM lookup.

Table 3.620. typedef mem_cam_lookup16_in_t

Type	Definition
mem_cam_lookup16_in_t	union mem_cam_lookup16_in_t

3.21.3.19 mem_cam_lookup16_out_t

Output type of 16-bit CAM lookups.

The match-bit fields indicate which CAM entries matched the lookup.

Table 3.621. `typedef mem_cam_lookup16_out_t`

Type	Definition
<code>mem_cam_lookup16_out_t</code>	<code>union mem_cam_lookup16_out_t</code>

3.21.3.20 `mem_cam_lookup16_add_out_t`

Output type of 16-bit CAM lookup and add operations.

The match-bit fields indicate which CAM entries matched the lookup.

Table 3.622. `typedef mem_cam_lookup16_add_out_t`

Type	Definition
<code>mem_cam_lookup16_add_out_t</code>	<code>union mem_cam_lookup16_add_out_t</code>

3.21.3.21 `mem_cam_lookup16_out_in_read_reg_t`

Type for `mem_cam_lookup16_out_t` in read registers.

Table 3.623. `typedef mem_cam_lookup16_out_in_read_reg_t`

Type	Definition
<code>mem_cam_lookup16_out_in_read_reg_t</code>	<code>__xread mem_cam_lookup16_out_t</code>

3.21.3.22 `mem_cam_lookup16_add_out_in_read_reg_t`

Type for `mem_cam_lookup16_out_t` in read registers.

Table 3.624. `typedef mem_cam_lookup16_add_out_in_read_reg_t`

Type	Definition
<code>mem_cam_lookup16_add_out_in_read_reg_t</code>	<code>__xread mem_cam_lookup16_add_out_t</code>

3.21.3.23 `mem_cam_lookup24_in_t`

Input type for 24-bit CAM lookups.



Note

24-bit CAM entries occupy 32-bits in MEM with 8-bits being available for user data or a count which is typically used with the 24-bit CAM add_inc functions.

This search field of this structure needs to be populated before performing a CAM lookup.

Table 3.625. `typedef mem_cam_lookup24_in_t`

Type	Definition
<code>mem_cam_lookup24_in_t</code>	<code>union mem_cam_lookup24_in_t</code>

3.21.3.24 `mem_cam_lookup24_add_in_t`

Input type for 24-bit CAM lookup and add operations.

This search field of this structure needs to be populated before performing a CAM lookup. The upper 8-bits of the CAM entry is set from the high_byte field in this structure when a new entry is added on a CAM miss.



Note

24-bit CAM entries occupy 32-bits in MEM with 8-bits being available for user data or a count which is typically used with the 24-bit CAM add_inc functions.

Table 3.626. `typedef mem_cam_lookup24_add_in_t`

Type	Definition
<code>mem_cam_lookup24_add_in_t</code>	<code>union mem_cam_lookup24_add_in_t</code>

3.21.3.25 `mem_cam_lookup24_add_inc_in_t`

Input type for 24-bit CAM lookup, add and inc operations.

Table 3.627. `typedef mem_cam_lookup24_add_inc_in_t`

Type	Definition
<code>mem_cam_lookup24_add_inc_in_t</code>	<code>mem_cam_lookup24_add_in_t</code>

See Also:

- `mem_cam_lookup24_add_in_t`

3.21.3.26 `mem_cam_lookup24_out_t`

Output type of 24-bit CAM lookups.

The match-bit fields indicate which CAM entries matched the lookup.

Table 3.628. `typedef mem_cam_lookup24_out_t`

Type	Definition
<code>mem_cam_lookup24_out_t</code>	<code>union mem_cam_lookup24_out_t</code>

3.21.3.27 mem_cam_lookup24_add_out_t

Output type of 24-bit CAM lookup and add operations.

The match-bit fields indicate which CAM entries matched the lookup.

Table 3.629. typedef mem_cam_lookup24_add_out_t

Type	Definition
mem_cam_lookup24_add_out_t	union mem_cam_lookup24_add_out_t

3.21.3.28 mem_cam_lookup24_add_inc_out_t

Output type of 24-bit CAM lookup, add and increment operations.

The match-bit fields indicate which CAM entries matched the lookup.

Table 3.630. typedef mem_cam_lookup24_add_inc_out_t

Type	Definition
mem_cam_lookup24_add_inc_out_t	union mem_cam_lookup24_add_inc_out_t

3.21.3.29 mem_cam_lookup24_out_in_read_reg_t

Type for mem_cam_lookup24_out_t in read registers.

Table 3.631. typedef mem_cam_lookup24_out_in_read_reg_t

Type	Definition
mem_cam_lookup24_out_in_read_reg_t	<code>__xread mem_cam_lookup24_out_t</code>

3.21.3.30 mem_cam_lookup24_add_out_in_read_reg_t

Type for mem_cam_lookup24_add_out_t in read registers.

Table 3.632. typedef mem_cam_lookup24_add_out_in_read_reg_t

Type	Definition
mem_cam_lookup24_add_out_in_read_reg_t	<code>__xread mem_cam_lookup24_add_out_t</code>

3.21.3.31 mem_cam_lookup24_add_inc_out_in_read_reg_t

Type for mem_cam_lookup24_add_inc_out_t in read registers.

Table 3.633. `typedef mem_cam_lookup24_add_inc_out_in_read_reg_t`

Type	Definition
<code>mem_cam_lookup24_add_inc_out_in_read_reg_t</code>	<code>__xread mem_cam_lookup24_add_inc_out_t</code>

3.21.3.32 `mem_cam_lookup32_in_t`

Input type for 32-bit CAM lookups.

This search field of this structure needs to be populated before performing a CAM lookup.

Table 3.634. `typedef mem_cam_lookup32_in_t`

Type	Definition
<code>mem_cam_lookup32_in_t</code>	<code>struct mem_cam_lookup32_in_t</code>

3.21.3.33 `mem_cam_lookup32_out_t`

Output type of 32-bit CAM lookups.

The match-bit fields indicate which CAM entries matched the lookup.

Table 3.635. `typedef mem_cam_lookup32_out_t`

Type	Definition
<code>mem_cam_lookup32_out_t</code>	<code>union mem_cam_lookup32_out_t</code>

3.21.3.34 `mem_cam_lookup32_add_out_t`

Output type of 32-bit CAM lookup and add operations.

The match-bit fields indicate which CAM entries matched the lookup.

Table 3.636. `typedef mem_cam_lookup32_add_out_t`

Type	Definition
<code>mem_cam_lookup32_add_out_t</code>	<code>union mem_cam_lookup32_add_out_t</code>

3.21.3.35 `mem_cam_lookup32_out_in_read_reg_t`

Type for `mem_cam_lookup32_out_t` in read registers.

Table 3.637. `typedef mem_cam_lookup32_out_in_read_reg_t`

Type	Definition
<code>mem_cam_lookup32_out_in_read_reg_t</code>	<code>__xread mem_cam_lookup32_out_t</code>

3.21.3.36 mem_cam_lookup32_add_out_in_read_reg_t

Type for mem_cam_lookup32_add_out_t in read registers.

Table 3.638. typedef mem_cam_lookup32_add_out_in_read_reg_t

Type	Definition
mem_cam_lookup32_add_out_in_read_reg_t	<code>__xread mem_cam_lookup32_add_out_t</code>

3.21.4 MU CAM Functions

3.21.4.1 mem_cam128_init_ptr32

Prototype:

```
void mem_cam128_init_ptr32(__mem mem_cam128_in_mem_t* cam)
```

Description:

Initialize a 128-bit CAM in 32-bit addressed memory.

Table 3.639. mem_cam128_init_ptr32 parameters

Type	Name	Description
<code>__mem mem_cam128_in_mem_t*</code>	<code>cam</code>	Pointer to the CAM in MEM

3.21.4.2 mem_cam128_init_ptr40

Prototype:

```
void mem_cam128_init_ptr40(mem_cam128_ptr40_t cam)
```

Description:

Initialize a 128-bit CAM in 40-bit addressed MEM.

Table 3.640. mem_cam128_init_ptr40 parameters

Type	Name	Description
<code>mem_cam128_ptr40_t</code>	<code>cam</code>	Pointer to the CAM in MEM

3.21.4.3 mem_cam256_init_ptr32

Prototype:

```
void mem_cam256_init_ptr32(__mem mem_cam256_in_mem_t* cam)
```

Description:

Initialize a 256-bit CAM in in 32-bit addressed memory.

Table 3.641. mem_cam256_init_ptr32 parameters

Type	Name	Description
<code>__mem mem_cam256_in_mem_t*</code>	<code>cam</code>	Pointer to the CAM in MEM

3.21.4.4 mem_cam256_init_ptr40

Prototype:

```
void mem_cam256_init_ptr40(mem_cam256_ptr40_t cam)
```

Description:

Initialize a 256-bit CAM in 40-bit addressed MEM.

Table 3.642. mem_cam256_init_ptr40 parameters

Type	Name	Description
<code>mem_cam256_ptr40_t</code>	<code>cam</code>	Pointer to the CAM in MEM

3.21.4.5 mem_cam384_init_ptr32

Prototype:

```
void mem_cam384_init_ptr32(__mem mem_cam384_in_mem_t* cam)
```

Description:

Initialize a 384-bit CAM in in 32-bit addressed memory.

Table 3.643. mem_cam384_init_ptr32 parameters

Type	Name	Description
<code>__mem mem_cam384_in_mem_t*</code>	<code>cam</code>	Pointer to the CAM in MEM

3.21.4.6 mem_cam384_init_ptr40

Prototype:

```
void mem_cam384_init_ptr40(mem_cam384_ptr40_t cam)
```

Description:

Initialize a 384-bit CAM in 40-bit addressed MEM.

Table 3.644. mem_cam384_init_ptr40 parameters

Type	Name	Description
mem_cam384_ptr40_t	<i>cam</i>	Pointer to the CAM in MEM

3.21.4.7 mem_cam512_init_ptr32

Prototype:

```
void mem_cam512_init_ptr32(__mem mem_cam512_in_mem_t* cam)
```

Description:

Initialize a 512-bit CAM in 32-bit addressed memory.

Table 3.645. mem_cam512_init_ptr32 parameters

Type	Name	Description
__mem mem_cam512_in_mem_t*	<i>cam</i>	Pointer to the CAM in MEM

3.21.4.8 mem_cam512_init_ptr40

Prototype:

```
void mem_cam512_init_ptr40(mem_cam512_ptr40_t cam)
```

Description:

Initialize a 512-bit CAM in 40-bit addressed memory.

Table 3.646. mem_cam512_init_ptr40 parameters

Type	Name	Description
mem_cam512_ptr40_t	<i>cam</i>	Pointer to the CAM in MEM

3.21.4.9 mem_cam128_set8_ptr32

Prototype:

```
void mem_cam128_set8_ptr32(__mem mem_cam128_in_mem_t* cam, unsigned int index, unsigned
int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 8-bit entry in a 128-bit CAM in 32-bit addressed MEM.



Note

The maximum index is 15

Table 3.647. mem_cam128_set8_ptr32 parameters

Type	Name	Description
<code>__mem mem_cam128_in_mem_t*</code>	<code>cam</code>	Pointer to the CAM structure in MEM
<code>unsigned int</code>	<code>index</code>	CAM entry to set (0 is the first entry)
<code>unsigned int</code>	<code>value</code>	Value of the CAM entry
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.21.4.10 mem_cam128_set8_ptr40

Prototype:

```
void mem_cam128_set8_ptr40(mem_cam128_ptr40_t cam, unsigned int index, unsigned int value,
                           sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 8-bit entry in a 128-bit CAM in 40-bit addressed MEM.



Note

The maximum index is 15

Table 3.648. mem_cam128_set8_ptr40 parameters

Type	Name	Description
<code>mem_cam128_ptr40_t</code>	<code>cam</code>	Pointer to the CAM structure in MEM
<code>unsigned int</code>	<code>index</code>	CAM entry to set (0 is the first entry)
<code>unsigned int</code>	<code>value</code>	Value of the CAM entry
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.21.4.11 mem_cam256_set8_ptr32

Prototype:

```
void mem_cam256_set8_ptr32(__mem mem_cam256_in_mem_t* cam, unsigned int index, unsigned
                           int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 8-bit entry in a 256-bit CAM in 32-bit addressed MEM.



Note

The maximum index is 31.

Table 3.649. mem_cam256_set8_ptr32 parameters

Type	Name	Description
<code>__mem</code> <code>mem_cam256_in_mem_t*</code>	<code>cam</code>	Pointer to the CAM structure in MEM
<code>unsigned int</code>	<code>index</code>	CAM entry to set (0 is the first entry)
<code>unsigned int</code>	<code>value</code>	Value of the CAM entry
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.21.4.12 mem_cam256_set8_ptr40

Prototype:

```
void mem_cam256_set8_ptr40(mem_cam256_ptr40_t cam, unsigned int index, unsigned int value,
sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 8-bit entry in a 256-bit CAM in 40-bit addressed MEM.



Note

The maximum index is 31.

Table 3.650. mem_cam256_set8_ptr40 parameters

Type	Name	Description
<code>mem_cam256_ptr40_t</code>	<code>cam</code>	Pointer to the CAM structure in MEM
<code>unsigned int</code>	<code>index</code>	CAM entry to set (0 is the first entry)
<code>unsigned int</code>	<code>value</code>	Value of the CAM entry
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.21.4.13 mem_cam384_set8_ptr32

Prototype:

```
void mem_cam384_set8_ptr32(__mem mem_cam384_in_mem_t* cam, unsigned int index, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 8-bit entry in a 384-bit CAM in 32-bit addressed MEM.

Note

The maximum index is 47.

Table 3.651. mem_cam384_set8_ptr32 parameters

Type	Name	Description
<code>__mem mem_cam384_in_mem_t*</code>	<code>cam</code>	Pointer to the CAM structure in MEM
<code>unsigned int</code>	<code>index</code>	CAM entry to set (0 is the first entry)
<code>unsigned int</code>	<code>value</code>	Value of the CAM entry
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.21.4.14 mem_cam384_set8_ptr40

Prototype:

```
void mem_cam384_set8_ptr40(mem_cam384_ptr40_t cam, unsigned int index, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 8-bit entry in a 384-bit CAM in 40-bit addressed MEM.

Note

The maximum index is 47.

Table 3.652. mem_cam384_set8_ptr40 parameters

Type	Name	Description
<code>mem_cam384_ptr40_t</code>	<code>cam</code>	Pointer to the CAM structure in MEM
<code>unsigned int</code>	<code>index</code>	CAM entry to set (0 is the first entry)

Type	Name	Description
unsigned int	<i>value</i>	Value of the CAM entry
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.21.4.15 mem_cam512_set8_ptr32

Prototype:

```
void mem_cam512_set8_ptr32(__mem mem_cam512_in_mem_t* cam, unsigned int index, unsigned
int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 8-bit entry in a 512-bit CAM in 32-bit addressed MEM.



Note

The maximum index is 63.

Table 3.653. mem_cam512_set8_ptr32 parameters

Type	Name	Description
__mem mem_cam512_in_mem_t*	<i>cam</i>	Pointer to the CAM structure in MEM
unsigned int	<i>index</i>	CAM entry to set (0 is the first entry)
unsigned int	<i>value</i>	Value of the CAM entry
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.21.4.16 mem_cam512_set8_ptr40

Prototype:

```
void mem_cam512_set8_ptr40(mem_cam512_ptr40_t cam, unsigned int index, unsigned int value,
sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 8-bit entry in a 512-bit CAM in 40-bit addressed MEM.



Note

The maximum index is 63.

Table 3.654. mem_cam512_set8_ptr40 parameters

Type	Name	Description
mem_cam512_ptr40_t	<i>cam</i>	Pointer to the CAM structure in MEM
unsigned int	<i>index</i>	CAM entry to set (0 is the first entry)
unsigned int	<i>value</i>	Value of the CAM entry
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.21.4.17 mem_cam128_set16_ptr32

Prototype:

```
void mem_cam128_set16_ptr32(__mem mem_cam128_in_mem_t* cam, unsigned int index, unsigned
int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 16-bit entry in a 128-bit CAM in 32-bit addressed MEM.



Note

The maximum index is 7.

Table 3.655. mem_cam128_set16_ptr32 parameters

Type	Name	Description
__mem mem_cam128_in_mem_t*	<i>cam</i>	Pointer to the CAM structure in MEM
unsigned int	<i>index</i>	CAM entry to set (0 is the first entry)
unsigned int	<i>value</i>	Value of the CAM entry
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.21.4.18 mem_cam128_set16_ptr40

Prototype:

```
void mem_cam128_set16_ptr40(mem_cam128_ptr40_t cam, unsigned int index, unsigned int
value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 16-bit entry in a 128-bit CAM in 40-bit addressed MEM.



Note

The maximum index is 7.

Table 3.656. mem_cam128_set16_ptr40 parameters

Type	Name	Description
mem_cam128_ptr40_t	<i>cam</i>	Pointer to the CAM structure in MEM
unsigned int	<i>index</i>	CAM entry to set (0 is the first entry)
unsigned int	<i>value</i>	Value of the CAM entry
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.21.4.19 mem_cam256_set16_ptr32

Prototype:

```
void mem_cam256_set16_ptr32(__mem mem_cam256_in_mem_t* cam, unsigned int index, unsigned
int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 16-bit entry in a 256-bit CAM in 32-bit addressed MEM.



Note

The maximum index is 15.

Table 3.657. mem_cam256_set16_ptr32 parameters

Type	Name	Description
__mem mem_cam256_in_mem_t*	<i>cam</i>	Pointer to the CAM structure in MEM
unsigned int	<i>index</i>	CAM entry to set (0 is the first entry)
unsigned int	<i>value</i>	Value of the CAM entry
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.21.4.20 mem_cam256_set16_ptr40

Prototype:

```
void mem_cam256_set16_ptr40(mem_cam256_ptr40_t cam, unsigned int index, unsigned int
value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 16-bit entry in a 256-bit CAM in 40-bit addressed MEM.



Note

The maximum index is 15.

Table 3.658. mem_cam256_set16_ptr40 parameters

Type	Name	Description
mem_cam256_ptr40_t	<i>cam</i>	Pointer to the CAM structure in MEM
unsigned int	<i>index</i>	CAM entry to set (0 is the first entry)
unsigned int	<i>value</i>	Value of the CAM entry
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.21.4.21 mem_cam384_set16_ptr32

Prototype:

```
void mem_cam384_set16_ptr32(__mem mem_cam384_in_mem_t* cam, unsigned int index, unsigned
int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 16-bit entry in a 384-bit CAM in 32-bit addressed MEM.



Note

The maximum index is 23.

Table 3.659. mem_cam384_set16_ptr32 parameters

Type	Name	Description
__mem mem_cam384_in_mem_t*	<i>cam</i>	Pointer to the CAM structure in MEM
unsigned int	<i>index</i>	CAM entry to set (0 is the first entry)
unsigned int	<i>value</i>	Value of the CAM entry
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.21.4.22 mem_cam384_set16_ptr40

Prototype:

```
void mem_cam384_set16_ptr40(mem_cam384_ptr40_t cam, unsigned int index, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 16-bit entry in a 384-bit CAM in 40-bit addressed MEM.



Note

The maximum index is 23.

Table 3.660. mem_cam384_set16_ptr40 parameters

Type	Name	Description
mem_cam384_ptr40_t	<i>cam</i>	Pointer to the CAM structure in MEM
unsigned int	<i>index</i>	CAM entry to set (0 is the first entry)
unsigned int	<i>value</i>	Value of the CAM entry
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.21.4.23 mem_cam512_set16_ptr32

Prototype:

```
void mem_cam512_set16_ptr32(__mem mem_cam512_in_mem_t* cam, unsigned int index, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 16-bit entry in a 512-bit CAM in 32-bit addressed MEM.



Note

The maximum index is 31.

Table 3.661. mem_cam512_set16_ptr32 parameters

Type	Name	Description
__mem mem_cam512_in_mem_t*	<i>cam</i>	Pointer to the CAM structure in MEM
unsigned int	<i>index</i>	CAM entry to set (0 is the first entry)

Type	Name	Description
unsigned int	value	Value of the CAM entry
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

3.21.4.24 mem_cam512_set16_ptr40

Prototype:

```
void mem_cam512_set16_ptr40(mem_cam512_ptr40_t cam, unsigned int index, unsigned int
value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 16-bit entry in a 512-bit CAM in 40-bit addressed MEM.



Note

The maximum index is 31.

Table 3.662. mem_cam512_set16_ptr40 parameters

Type	Name	Description
mem_cam512_ptr40_t	cam	Pointer to the CAM structure in MEM
unsigned int	index	CAM entry to set (0 is the first entry)
unsigned int	value	Value of the CAM entry
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

3.21.4.25 mem_cam128_set24_ptr32

Prototype:

```
void mem_cam128_set24_ptr32(__mem mem_cam128_in_mem_t* cam, unsigned int index, unsigned
int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 24-bit entry in a 128-bit CAM in 32-bit addressed MEM.



Note

The maximum index is 3.

24-bit CAM entries occupy 32-bits in MEM with 8-bits being available for user data or a count which is typically used with the 24-bit CAM add_inc functions.

Table 3.663. mem_cam128_set24_ptr32 parameters

Type	Name	Description
<code>__mem mem_cam128_in_mem_t*</code>	<code>cam</code>	Pointer to the CAM structure in MEM
<code>unsigned int</code>	<code>index</code>	CAM entry to set (0 is the first entry)
<code>unsigned int</code>	<code>value</code>	Value of the CAM entry
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

See Also:

- `mem_cam128_lookup24_add_inc`

3.21.4.26 mem_cam128_set24_ptr40

Prototype:

```
void mem_cam128_set24_ptr40(mem_cam128_ptr40_t cam, unsigned int index, unsigned int  
value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 24-bit entry in a 128-bit CAM in 40-bit addressed MEM.



Note

The maximum index is 3.

24-bit CAM entries occupy 32-bits in MEM with 8-bits being available for user data or a count which is typically used with the 24-bit CAM add_inc functions.

Table 3.664. mem_cam128_set24_ptr40 parameters

Type	Name	Description
<code>mem_cam128_ptr40_t</code>	<code>cam</code>	Pointer to the CAM structure in MEM
<code>unsigned int</code>	<code>index</code>	CAM entry to set (0 is the first entry)
<code>unsigned int</code>	<code>value</code>	Value of the CAM entry
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

See Also:

- `mem_cam128_lookup24_add_inc`

3.21.4.27 mem_cam256_set24_ptr32

Prototype:

```
void mem_cam256_set24_ptr32(__mem mem_cam256_in_mem_t* cam, unsigned int index, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 24-bit entry in a 256-bit CAM in 32-bit addressed MEM.

Note

The maximum index is 7.

Table 3.665. mem_cam256_set24_ptr32 parameters

Type	Name	Description
__mem mem_cam256_in_mem_t*	cam	Pointer to the CAM structure in MEM
unsigned int	index	CAM entry to set (0 is the first entry)
unsigned int	value	Value of the CAM entry
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

3.21.4.28 mem_cam256_set24_ptr40

Prototype:

```
void mem_cam256_set24_ptr40(mem_cam256_ptr40_t cam, unsigned int index, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 24-bit entry in a 256-bit CAM in 40-bit addressed MEM.

Note

The maximum index is 7.

Table 3.666. mem_cam256_set24_ptr40 parameters

Type	Name	Description
mem_cam256_ptr40_t	cam	Pointer to the CAM structure in MEM
unsigned int	index	CAM entry to set (0 is the first entry)

Type	Name	Description
unsigned int	<i>value</i>	Value of the CAM entry
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.21.4.29 mem_cam384_set24_ptr32

Prototype:

```
void mem_cam384_set24_ptr32(__mem mem_cam384_in_mem_t* cam, unsigned int index, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 24-bit entry in a 384-bit CAM in 32-bit addressed MEM.



Note

The maximum index is 11.

Table 3.667. mem_cam384_set24_ptr32 parameters

Type	Name	Description
__mem mem_cam384_in_mem_t*	<i>cam</i>	Pointer to the CAM structure in MEM
unsigned int	<i>index</i>	CAM entry to set (0 is the first entry)
unsigned int	<i>value</i>	Value of the CAM entry
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.21.4.30 mem_cam384_set24_ptr40

Prototype:

```
void mem_cam384_set24_ptr40(mem_cam384_ptr40_t cam, unsigned int index, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 24-bit entry in a 384-bit CAM in 40-bit addressed MEM.



Note

The maximum index is 11.

Table 3.668. mem_cam384_set24_ptr40 parameters

Type	Name	Description
mem_cam384_ptr40_t	<i>cam</i>	Pointer to the CAM structure in MEM
unsigned int	<i>index</i>	CAM entry to set (0 is the first entry)
unsigned int	<i>value</i>	Value of the CAM entry
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.21.4.31 mem_cam512_set24_ptr32

Prototype:

```
void mem_cam512_set24_ptr32(__mem mem_cam512_in_mem_t* cam, unsigned int index, unsigned
int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 24-bit entry in a 512-bit CAM in 32-bit addressed MEM.



Note

The maximum index is 15.

Table 3.669. mem_cam512_set24_ptr32 parameters

Type	Name	Description
__mem mem_cam512_in_mem_t*	<i>cam</i>	Pointer to the CAM structure in MEM
unsigned int	<i>index</i>	CAM entry to set (0 is the first entry)
unsigned int	<i>value</i>	Value of the CAM entry
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.21.4.32 mem_cam512_set24_ptr40

Prototype:

```
void mem_cam512_set24_ptr40(mem_cam512_ptr40_t cam, unsigned int index, unsigned int
value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 24-bit entry in a 512-bit CAM in 40-bit addressed MEM.



Note

The maximum index is 15.

Table 3.670. mem_cam512_set24_ptr40 parameters

Type	Name	Description
mem_cam512_ptr40_t	<i>cam</i>	Pointer to the CAM structure in MEM
unsigned int	<i>index</i>	CAM entry to set (0 is the first entry)
unsigned int	<i>value</i>	Value of the CAM entry
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.21.4.33 mem_cam128_set32_ptr32

Prototype:

```
void mem_cam128_set32_ptr32(__mem mem_cam128_in_mem_t* cam, unsigned int index, unsigned
int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 32-bit entry in a 128-bit CAM in 32-bit addressed MEM.



Note

The maximum index is 3.

Table 3.671. mem_cam128_set32_ptr32 parameters

Type	Name	Description
__mem mem_cam128_in_mem_t*	<i>cam</i>	Pointer to the CAM structure in MEM
unsigned int	<i>index</i>	CAM entry to set (0 is the first entry)
unsigned int	<i>value</i>	Value of the CAM entry
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.21.4.34 mem_cam128_set32_ptr40

Prototype:

```
void mem_cam128_set32_ptr40(mem_cam128_ptr40_t cam, unsigned int index, unsigned int
value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 32-bit entry in a 128-bit CAM in 40-bit addressed MEM.



Note

The maximum index is 3.

Table 3.672. mem_cam128_set32_ptr40 parameters

Type	Name	Description
mem_cam128_ptr40_t	<i>cam</i>	Pointer to the CAM structure in MEM
unsigned int	<i>index</i>	CAM entry to set (0 is the first entry)
unsigned int	<i>value</i>	Value of the CAM entry
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.21.4.35 mem_cam256_set32_ptr32

Prototype:

```
void mem_cam256_set32_ptr32(__mem mem_cam256_in_mem_t* cam, unsigned int index, unsigned
int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 32-bit entry in a 256-bit CAM in 32-bit addressed MEM.



Note

The maximum index is 7.

Table 3.673. mem_cam256_set32_ptr32 parameters

Type	Name	Description
__mem mem_cam256_in_mem_t*	<i>cam</i>	Pointer to the CAM structure in MEM
unsigned int	<i>index</i>	CAM entry to set (0 is the first entry)
unsigned int	<i>value</i>	Value of the CAM entry
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.21.4.36 mem_cam256_set32_ptr40

Prototype:

```
void mem_cam256_set32_ptr40(mem_cam256_ptr40_t cam, unsigned int index, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 32-bit entry in a 256-bit CAM in 40-bit addressed MEM.

Note

The maximum index is 7.

Table 3.674. mem_cam256_set32_ptr40 parameters

Type	Name	Description
mem_cam256_ptr40_t	<i>cam</i>	Pointer to the CAM structure in MEM
unsigned int	<i>index</i>	CAM entry to set (0 is the first entry)
unsigned int	<i>value</i>	Value of the CAM entry
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.21.4.37 mem_cam384_set32_ptr32

Prototype:

```
void mem_cam384_set32_ptr32(__mem mem_cam384_in_mem_t* cam, unsigned int index, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 32-bit entry in a 384-bit CAM in 32-bit addressed MEM.

Note

The maximum index is 11.

Table 3.675. mem_cam384_set32_ptr32 parameters

Type	Name	Description
__mem mem_cam384_in_mem_t*	<i>cam</i>	Pointer to the CAM structure in MEM
unsigned int	<i>index</i>	CAM entry to set (0 is the first entry)

Type	Name	Description
unsigned int	value	Value of the CAM entry
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

3.21.4.38 mem_cam384_set32_ptr40

Prototype:

```
void mem_cam384_set32_ptr40(mem_cam384_ptr40_t cam, unsigned int index, unsigned int
value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 32-bit entry in a 384-bit CAM in 40-bit addressed MEM.



Note

The maximum index is 11.

Table 3.676. mem_cam384_set32_ptr40 parameters

Type	Name	Description
mem_cam384_ptr40_t	cam	Pointer to the CAM structure in MEM
unsigned int	index	CAM entry to set (0 is the first entry)
unsigned int	value	Value of the CAM entry
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

3.21.4.39 mem_cam512_set32_ptr32

Prototype:

```
void mem_cam512_set32_ptr32(__mem mem_cam512_in_mem_t* cam, unsigned int index, unsigned
int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 32-bit entry in a 512-bit CAM in 32-bit addressed MEM.



Note

The maximum index is 15.

Table 3.677. mem_cam512_set32_ptr32 parameters

Type	Name	Description
<code>__mem mem_cam512_in_mem_t*</code>	<code>cam</code>	Pointer to the CAM structure in MEM
<code>unsigned int</code>	<code>index</code>	CAM entry to set (0 is the first entry)
<code>unsigned int</code>	<code>value</code>	Value of the CAM entry
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.21.4.40 mem_cam512_set32_ptr40

Prototype:

```
void mem_cam512_set32_ptr40(mem_cam512_ptr40_t cam, unsigned int index, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 32-bit entry in a 512-bit CAM in 40-bit addressed MEM.



Note

The maximum index is 15.

Table 3.678. mem_cam512_set32_ptr40 parameters

Type	Name	Description
<code>mem_cam512_ptr40_t</code>	<code>cam</code>	Pointer to the CAM structure in MEM
<code>unsigned int</code>	<code>index</code>	CAM entry to set (0 is the first entry)
<code>unsigned int</code>	<code>value</code>	Value of the CAM entry
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.21.4.41 mem_cam128_lookup8_ptr32

Prototype:

```
mem_cam_lookup8_out_in_read_reg_t* mem_cam128_lookup8_ptr32(__mem mem_cam128_in_mem_t* cam, __xwrite mem_cam_lookup8_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 8-bit lookup in a 128-bit CAM in 32-bit addressed MEM.



Note

The result is returned in a mem_cam_lookup8_out structure occupying two read transfer registers.

Table 3.679. mem_cam128_lookup8_ptr32 parameters

Type	Name	Description
__mem mem_cam128_in_mem_t*	<i>cam</i>	Pointer to the CAM structure in MEM
__xwrite mem_cam_lookup8_in_t*	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- mem_cam_lookup8_in_t, mem_cam_lookup8_out_t

3.21.4.42 mem_cam128_lookup8_ptr40

Prototype:

```
mem_cam_lookup8_out_in_read_reg_t* mem_cam128_lookup8_ptr40(mem_cam128_ptr40_t cam,  
__xwrite mem_cam_lookup8_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 8-bit lookup in a 128-bit CAM in 40-bit addressed MEM.



Note

The result is returned in a mem_cam_lookup8_out structure occupying two read transfer registers.

Table 3.680. mem_cam128_lookup8_ptr40 parameters

Type	Name	Description
mem_cam128_ptr40_t	<i>cam</i>	Pointer to the CAM structure in MEM
__xwrite mem_cam_lookup8_in_t*	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)

Type	Name	Description
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- `mem_cam_lookup8_in_t`, `mem_cam_lookup8_out_t`

3.21.4.43 mem_cam256_lookup8_ptr32

Prototype:

```
mem_cam_lookup8_out_in_read_reg_t* mem_cam256_lookup8_ptr32(__mem mem_cam256_in_mem_t* cam, __xwrite mem_cam_lookup8_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 8-bit lookup in a 256-bit CAM in 32-bit addressed MEM.



Note

The result is returned in a `mem_cam_lookup8_out` structure occupying two read transfer registers.

Table 3.681. mem_cam256_lookup8_ptr32 parameters

Type	Name	Description
<code>__mem mem_cam256_in_mem_t*</code>	<i>cam</i>	Pointer to the CAM structure in MEM
<code>__xwrite mem_cam_lookup8_in_t*</code>	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
<code>sync_t</code>	<i>sync</i>	Type of synchronization to use (must be <code>sig_done</code>)
<code>SIGNAL_PAIR*</code>	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- `mem_cam_lookup8_in_t`, `mem_cam_lookup8_out_t`

3.21.4.44 mem_cam256_lookup8_ptr40

Prototype:

```
mem_cam_lookup8_out_in_read_reg_t* mem_cam256_lookup8_ptr40(mem_cam256_ptr40_t cam,  
__xwrite mem_cam_lookup8_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 8-bit lookup in a 256-bit CAM in 40-bit addressed MEM.



Note

The result is returned in a mem_cam_lookup8_out structure occupying two read transfer registers.

Table 3.682. mem_cam256_lookup8_ptr40 parameters

Type	Name	Description
mem_cam256_ptr40_t	cam	Pointer to the CAM structure in MEM
__xwrite mem_cam_lookup8_in_t*	xfer	Pointer to write xfer register containing the data to lookup
sync_t	sync	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	sig_pair_ptr	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- mem_cam_lookup8_in_t, mem_cam_lookup8_out_t

3.21.4.45 mem_cam384_lookup8_ptr32

Prototype:

```
mem_cam_lookup8_out_in_read_reg_t* mem_cam384_lookup8_ptr32(__mem mem_cam384_in_mem_t*  
cam, __xwrite mem_cam_lookup8_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 8-bit lookup in a 384-bit CAM in 32-bit addressed MEM.



Note

The result is returned in a mem_cam_lookup8_out structure occupying two read transfer registers.

Table 3.683. mem_cam384_lookup8_ptr32 parameters

Type	Name	Description
<code>__mem mem_cam384_in_mem_t*</code>	<code>cam</code>	Pointer to the CAM structure in MEM
<code>__xwrite mem_cam_lookup8_in_t*</code>	<code>xfer</code>	Pointer to write xfer register containing the data to lookup
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be sig_done)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- `mem_cam_lookup8_in_t`, `mem_cam_lookup8_out_t`

3.21.4.46 mem_cam384_lookup8_ptr40

Prototype:

```
mem_cam_lookup8_out_in_read_reg_t* mem_cam384_lookup8_ptr40(mem_cam384_ptr40_t cam,  
__xwrite mem_cam_lookup8_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 8-bit lookup in a 384-bit CAM in 40-bit addressed MEM.



Note

The result is returned in a `mem_cam_lookup8_out` structure occupying two read transfer registers.

Table 3.684. mem_cam384_lookup8_ptr40 parameters

Type	Name	Description
<code>mem_cam384_ptr40_t</code>	<code>cam</code>	Pointer to the CAM structure in MEM
<code>__xwrite mem_cam_lookup8_in_t*</code>	<code>xfer</code>	Pointer to write xfer register containing the data to lookup
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be sig_done)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- `mem_cam_lookup8_in_t`, `mem_cam_lookup8_out_t`

3.21.4.47 mem_cam512_lookup8_ptr32

Prototype:

```
mem_cam_lookup8_out_in_read_reg_t* mem_cam512_lookup8_ptr32(__mem mem_cam512_in_mem_t* cam, __xwrite mem_cam_lookup8_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 8-bit lookup in a 512-bit CAM in 32-bit addressed MEM.



Note

The result is returned in a `mem_cam_lookup8_out` structure occupying two read transfer registers.

Table 3.685. mem_cam512_lookup8_ptr32 parameters

Type	Name	Description
<code>__mem mem_cam512_in_mem_t*</code>	<code>cam</code>	Pointer to the CAM structure in MEM
<code>__xwrite mem_cam_lookup8_in_t*</code>	<code>xfer</code>	Pointer to write xfer register containing the data to lookup
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be <code>sig_done</code>)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- `mem_cam_lookup8_in_t`, `mem_cam_lookup8_out_t`

3.21.4.48 mem_cam512_lookup8_ptr40

Prototype:

```
mem_cam_lookup8_out_in_read_reg_t* mem_cam512_lookup8_ptr40(mem_cam512_ptr40_t cam, __xwrite mem_cam_lookup8_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 8-bit lookup in a 512-bit CAM in 40-bit addressed MEM.



Note

The result is returned in a mem_cam_lookup8_out structure occupying two read transfer registers.

Table 3.686. mem_cam512_lookup8_ptr40 parameters

Type	Name	Description
mem_cam512_ptr40_t	<i>cam</i>	Pointer to the CAM structure in MEM
__xwrite mem_cam_lookup8_in_t*	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- mem_cam_lookup8_in_t, mem_cam_lookup8_out_t

3.21.4.49 mem_cam128_lookup8_add_ptr32

Prototype:

```
mem_cam_lookup8_add_out_in_read_reg_t* mem_cam128_lookup8_add_ptr32(__mem
mem_cam128_in_mem_t* cam, __xwrite mem_cam_lookup8_in_t* xfer, sync_t sync, SIGNAL_PAIR*
sig_pair_ptr)
```

Description:

Perform a 8-bit lookup in a 128-bit CAM in 32-bit addressed MEM and if a match is not found add the entry to the first empty slot.



Note

The result is returned in a mem_cam_lookup8_add_out structure occupying two read transfer registers.

Table 3.687. mem_cam128_lookup8_add_ptr32 parameters

Type	Name	Description
__mem mem_cam128_in_mem_t*	<i>cam</i>	Pointer to the CAM in MEM

Type	Name	Description
<code>__xwrite mem_cam_lookup8_in_t*</code>	<code>xfer</code>	Pointer to write xfer register containing the data to lookup
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be sig_done)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup

See Also:

- `mem_cam_lookup8_in_t, mem_cam_lookup8_add_out_t`

3.21.4.50 mem_cam128_lookup8_add_ptr40

Prototype:

```
mem_cam_lookup8_add_out_in_read_reg_t* mem_cam128_lookup8_add_ptr40(mem_cam128_ptr40_t
cam, __xwrite mem_cam_lookup8_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 8-bit lookup in a 128-bit CAM in 40-bit addressed MEM and if a match is not found add the entry to the first empty slot.



Note

The result is returned in a `mem_cam_lookup8_add_out` structure occupying two read transfer registers.

Table 3.688. mem_cam128_lookup8_add_ptr40 parameters

Type	Name	Description
<code>mem_cam128_ptr40_t</code>	<code>cam</code>	Pointer to the CAM in MEM
<code>__xwrite mem_cam_lookup8_in_t*</code>	<code>xfer</code>	Pointer to write xfer register containing the data to lookup
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be sig_done)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup

See Also:

- `mem_cam_lookup8_in_t, mem_cam_lookup8_add_out_t`

3.21.4.51 mem_cam256_lookup8_add_ptr32

Prototype:

```
mem_cam_lookup8_add_out_in_read_reg_t* mem_cam256_lookup8_add_ptr32(__mem
mem_cam256_in_mem_t* cam, __xwrite mem_cam_lookup8_in_t* xfer, sync_t sync, SIGNAL_PAIR*
sig_pair_ptr)
```

Description:

Perform a 8-bit lookup in a 256-bit CAM in 32-bit addressed MEM and if a match is not found add the entry to the first empty slot.



Note

The result is returned in a mem_cam_lookup8_add_out structure occupying two read transfer registers.

Table 3.689. mem_cam256_lookup8_add_ptr32 parameters

Type	Name	Description
__mem mem_cam256_in_mem_t*	cam	Pointer to the CAM in MEM
__xwrite mem_cam_lookup8_in_t*	xfer	Pointer to write xfer register containing the data to lookup
sync_t	sync	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	sig_pair_ptr	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- mem_cam_lookup8_in_t, mem_cam_lookup8_add_out_t

3.21.4.52 mem_cam256_lookup8_add_ptr40

Prototype:

```
mem_cam_lookup8_add_out_in_read_reg_t* mem_cam256_lookup8_add_ptr40(mem_cam256_ptr40_t
cam, __xwrite mem_cam_lookup8_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 8-bit lookup in a 256-bit CAM in 40-bit addressed MEM and if a match is not found add the entry to the first empty slot.



Note

The result is returned in a mem_cam_lookup8_add_out structure occupying two read transfer registers.

Table 3.690. mem_cam256_lookup8_add_ptr40 parameters

Type	Name	Description
mem_cam256_ptr40_t	<i>cam</i>	Pointer to the CAM in MEM
__xwrite mem_cam_lookup8_in_t*	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- mem_cam_lookup8_in_t, mem_cam_lookup8_add_out_t

3.21.4.53 mem_cam384_lookup8_add_ptr32

Prototype:

```
mem_cam_lookup8_add_out_in_read_reg_t* mem_cam384_lookup8_add_ptr32(__mem
mem_cam384_in_mem_t* cam, __xwrite mem_cam_lookup8_in_t* xfer, sync_t sync, SIGNAL_PAIR*
sig_pair_ptr)
```

Description:

Perform a 8-bit lookup in a 384-bit CAM in 32-bit addressed MEM and if a match is not found add the entry to the first empty slot.



Note

The result is returned in a mem_cam_lookup8_add_out structure occupying two read transfer registers.

Table 3.691. mem_cam384_lookup8_add_ptr32 parameters

Type	Name	Description
__mem mem_cam384_in_mem_t*	<i>cam</i>	Pointer to the CAM in MEM

Type	Name	Description
<code>__xwrite mem_cam_lookup8_in_t*</code>	<code>xfer</code>	Pointer to write xfer register containing the data to lookup
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be sig_done)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- `mem_cam_lookup8_in_t, mem_cam_lookup8_add_out_t`

3.21.4.54 mem_cam384_lookup8_add_ptr40

Prototype:

```
mem_cam_lookup8_add_out_in_read_reg_t* mem_cam384_lookup8_add_ptr40(mem_cam384_ptr40_t cam, __xwrite mem_cam_lookup8_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 8-bit lookup in a 384-bit CAM in 40-bit addressed MEM and if a match is not found add the entry to the first empty slot.



Note

The result is returned in a `mem_cam_lookup8_add_out` structure occupying two read transfer registers.

Table 3.692. mem_cam384_lookup8_add_ptr40 parameters

Type	Name	Description
<code>mem_cam384_ptr40_t</code>	<code>cam</code>	Pointer to the CAM in MEM
<code>__xwrite mem_cam_lookup8_in_t*</code>	<code>xfer</code>	Pointer to write xfer register containing the data to lookup
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be sig_done)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- `mem_cam_lookup8_in_t, mem_cam_lookup8_add_out_t`

3.21.4.55 mem_cam512_lookup8_add_ptr32

Prototype:

```
mem_cam_lookup8_add_out_in_read_reg_t* mem_cam512_lookup8_add_ptr32(__mem
mem_cam512_in_mem_t* cam, __xwrite mem_cam_lookup8_in_t* xfer, sync_t sync, SIGNAL_PAIR*
sig_pair_ptr)
```

Description:

Perform a 8-bit lookup in a 512-bit CAM in 32-bit addressed MEM and if a match is not found add the entry to the first empty slot.



Note

The result is returned in a mem_cam_lookup8_add_out structure occupying two read transfer registers.

Table 3.693. mem_cam512_lookup8_add_ptr32 parameters

Type	Name	Description
__mem mem_cam512_in_mem_t*	cam	Pointer to the CAM in MEM
__xwrite mem_cam_lookup8_in_t*	xfer	Pointer to write xfer register containing the data to lookup
sync_t	sync	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	sig_pair_ptr	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- mem_cam_lookup8_in_t, mem_cam_lookup8_add_out_t

3.21.4.56 mem_cam512_lookup8_add_ptr40

Prototype:

```
mem_cam_lookup8_add_out_in_read_reg_t* mem_cam512_lookup8_add_ptr40(mem_cam512_ptr40_t
cam, __xwrite mem_cam_lookup8_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 8-bit lookup in a 512-bit CAM in 40-bit addressed MEM and if a match is not found add the entry to the first empty slot.



Note

The result is returned in a mem_cam_lookup8_add_out structure occupying two read transfer registers.

Table 3.694. mem_cam512_lookup8_add_ptr40 parameters

Type	Name	Description
mem_cam512_ptr40_t	<i>cam</i>	Pointer to the CAM in MEM
__xwrite mem_cam_lookup8_in_t*	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- mem_cam_lookup8_in_t, mem_cam_lookup8_add_out_t

3.21.4.57 mem_cam128_lookup16_ptr32

Prototype:

```
mem_cam_lookup16_out_in_read_reg_t* mem_cam128_lookup16_ptr32(__mem mem_cam128_in_mem_t*
cam, __xwrite mem_cam_lookup16_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 16-bit lookup in a 128-bit CAM in 32-bit addressed MEM.



Note

The result is returned in a mem_cam_lookup16_out structure occupying two read transfer registers.

Table 3.695. mem_cam128_lookup16_ptr32 parameters

Type	Name	Description
__mem mem_cam128_in_mem_t*	<i>cam</i>	Pointer to the CAM structure in MEM
__xwrite mem_cam_lookup16_in_t*	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)

Type	Name	Description
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- `mem_cam_lookup16_in_t`, `mem_cam_lookup16_out_t`

3.21.4.58 mem_cam128_lookup16_ptr40

Prototype:

```
mem_cam_lookup16_out_in_read_reg_t* mem_cam128_lookup16_ptr40(mem_cam128_ptr40_t cam,
__xwrite mem_cam_lookup16_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 16-bit lookup in a 128-bit CAM in 40-bit addressed MEM.



Note

The result is returned in a `mem_cam_lookup16_out` structure occupying two read transfer registers.

Table 3.696. mem_cam128_lookup16_ptr40 parameters

Type	Name	Description
<code>mem_cam128_ptr40_t</code>	<i>cam</i>	Pointer to the CAM structure in MEM
<code>__xwrite mem_cam_lookup16_in_t*</code>	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
<code>sync_t</code>	<i>sync</i>	Type of synchronization to use (must be <code>sig_done</code>)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- `mem_cam_lookup16_in_t`, `mem_cam_lookup16_out_t`

3.21.4.59 mem_cam256_lookup16_ptr32

Prototype:

```
mem_cam_lookup16_out_in_read_reg_t* mem_cam256_lookup16_ptr32(__mem mem_cam256_in_mem_t* cam, __xwrite mem_cam_lookup16_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 16-bit lookup in a 256-bit CAM in 32-bit addressed MEM.



Note

The result is returned in a mem_cam_lookup16_out structure occupying two read transfer registers.

Table 3.697. mem_cam256_lookup16_ptr32 parameters

Type	Name	Description
__mem mem_cam256_in_mem_t*	cam	Pointer to the CAM structure in MEM
__xwrite mem_cam_lookup16_in_t*	xfer	Pointer to write xfer register containing the data to lookup
sync_t	sync	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	sig_pair_ptr	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- mem_cam_lookup16_in_t, mem_cam_lookup16_out_t

3.21.4.60 mem_cam256_lookup16_ptr40

Prototype:

```
mem_cam_lookup16_out_in_read_reg_t* mem_cam256_lookup16_ptr40(mem_cam256_ptr40_t cam, __xwrite mem_cam_lookup16_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 16-bit lookup in a 256-bit CAM in 40-bit addressed MEM.



Note

The result is returned in a mem_cam_lookup16_out structure occupying two read transfer registers.

Table 3.698. mem_cam256_lookup16_ptr40 parameters

Type	Name	Description
mem_cam256_ptr40_t	<i>cam</i>	Pointer to the CAM structure in MEM
__xwrite mem_cam_lookup16_in_t*	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- mem_cam_lookup16_in_t, mem_cam_lookup16_out_t

3.21.4.61 mem_cam384_lookup16_ptr32

Prototype:

```
mem_cam_lookup16_out_in_read_reg_t* mem_cam384_lookup16_ptr32(__mem mem_cam384_in_mem_t*
cam, __xwrite mem_cam_lookup16_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 16-bit lookup in a 384-bit CAM in 32-bit addressed MEM.



Note

The result is returned in a mem_cam_lookup16_out structure occupying two read transfer registers.

Table 3.699. mem_cam384_lookup16_ptr32 parameters

Type	Name	Description
__mem mem_cam384_in_mem_t*	<i>cam</i>	Pointer to the CAM structure in MEM
__xwrite mem_cam_lookup16_in_t*	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- `mem_cam_lookup16_in_t`, `mem_cam_lookup16_out_t`

3.21.4.62 mem_cam384_lookup16_ptr40

Prototype:

```
mem_cam_lookup16_out_in_read_reg_t* mem_cam384_lookup16_ptr40(mem_cam384_ptr40_t cam,  
__xwrite mem_cam_lookup16_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 16-bit lookup in a 384-bit CAM in 40-bit addressed MEM.



Note

The result is returned in a `mem_cam_lookup16_out` structure occupying two read transfer registers.

Table 3.700. mem_cam384_lookup16_ptr40 parameters

Type	Name	Description
<code>mem_cam384_ptr40_t</code>	<code>cam</code>	Pointer to the CAM structure in MEM
<code>__xwrite mem_cam_lookup16_in_t*</code>	<code>xfer</code>	Pointer to write xfer register containing the data to lookup
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be <code>sig_done</code>)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- `mem_cam_lookup16_in_t`, `mem_cam_lookup16_out_t`

3.21.4.63 mem_cam512_lookup16_ptr32

Prototype:

```
mem_cam_lookup16_out_in_read_reg_t* mem_cam512_lookup16_ptr32(__mem mem_cam512_in_mem_t*  
cam, __xwrite mem_cam_lookup16_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 16-bit lookup in a 512-bit CAM in 32-bit addressed MEM.



Note

The result is returned in a mem_cam_lookup16_out structure occupying two read transfer registers.

Table 3.701. mem_cam512_lookup16_ptr32 parameters

Type	Name	Description
__mem mem_cam512_in_mem_t*	<i>cam</i>	Pointer to the CAM structure in MEM
__xwrite mem_cam_lookup16_in_t*	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- mem_cam_lookup16_in_t, mem_cam_lookup16_out_t

3.21.4.64 mem_cam512_lookup16_ptr40

Prototype:

```
mem_cam_lookup16_out_in_read_reg_t* mem_cam512_lookup16_ptr40(mem_cam512_ptr40_t cam,  
__xwrite mem_cam_lookup16_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 16-bit lookup in a 512-bit CAM in 40-bit addressed MEM.



Note

The result is returned in a mem_cam_lookup16_out structure occupying two read transfer registers.

Table 3.702. mem_cam512_lookup16_ptr40 parameters

Type	Name	Description
mem_cam512_ptr40_t	<i>cam</i>	Pointer to the CAM structure in MEM
__xwrite mem_cam_lookup16_in_t*	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)

Type	Name	Description
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- `mem_cam_lookup16_in_t`, `mem_cam_lookup16_out_t`

3.21.4.65 mem_cam128_lookup16_add_ptr32

Prototype:

```
mem_cam_lookup16_add_out_in_read_reg_t* mem_cam128_lookup16_add_ptr32(__mem
mem_cam128_in_mem_t* cam, __xwrite mem_cam_lookup16_in_t* xfer, sync_t sync, SIGNAL_PAIR*
sig_pair_ptr)
```

Description:

Perform a 16-bit lookup in a 128-bit CAM in 32-bit addressed MEM and if a match is not found add the entry to the first empty slot.



Note

The result is returned in a `mem_cam_lookup16_add_out` structure occupying two read transfer registers.

Table 3.703. mem_cam128_lookup16_add_ptr32 parameters

Type	Name	Description
<code>__mem mem_cam128_in_mem_t*</code>	<i>cam</i>	Pointer to the CAM in MEM
<code>__xwrite mem_cam_lookup16_in_t*</code>	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
<code>sync_t</code>	<i>sync</i>	Type of synchronization to use (must be <code>sig_done</code>)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- `mem_cam_lookup16_in_t`, `mem_cam_lookup16_add_out_t`

3.21.4.66 mem_cam128_lookup16_add_ptr40

Prototype:

```
mem_cam_lookup16_add_out_in_read_reg_t* mem_cam128_lookup16_add_ptr40(mem_cam128_ptr40_t
cam, __xwrite mem_cam_lookup16_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 16-bit lookup in a 128-bit CAM in 40-bit addressed MEM and if a match is not found add the entry to the first empty slot.



Note

The result is returned in a mem_cam_lookup16_add_out structure occupying two read transfer registers.

Table 3.704. mem_cam128_lookup16_add_ptr40 parameters

Type	Name	Description
mem_cam128_ptr40_t	cam	Pointer to the CAM in MEM
__xwrite mem_cam_lookup16_in_t*	xfer	Pointer to write xfer register containing the data to lookup
sync_t	sync	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	sig_pair_ptr	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- mem_cam_lookup16_in_t, mem_cam_lookup16_add_out_t

3.21.4.67 mem_cam256_lookup16_add_ptr32

Prototype:

```
mem_cam_lookup16_add_out_in_read_reg_t* mem_cam256_lookup16_add_ptr32(__mem
mem_cam256_in_mem_t* cam, __xwrite mem_cam_lookup16_in_t* xfer, sync_t sync, SIGNAL_PAIR*
sig_pair_ptr)
```

Description:

Perform a 16-bit lookup in a 256-bit CAM in 32-bit addressed MEM and if a match is not found add the entry to the first empty slot.



Note

The result is returned in a mem_cam_lookup16_add_out structure occupying two read transfer registers.

Table 3.705. mem_cam256_lookup16_add_ptr32 parameters

Type	Name	Description
__mem mem_cam256_in_mem_t*	<i>cam</i>	Pointer to the CAM in MEM
__xwrite mem_cam_lookup16_in_t*	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- mem_cam_lookup16_in_t, mem_cam_lookup16_add_out_t

3.21.4.68 mem_cam256_lookup16_add_ptr40

Prototype:

```
mem_cam_lookup16_add_out_in_read_reg_t* mem_cam256_lookup16_add_ptr40(mem_cam256_ptr40_t
cam, __xwrite mem_cam_lookup16_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 16-bit lookup in a 256-bit CAM in 40-bit addressed MEM and if a match is not found add the entry to the first empty slot.



Note

The result is returned in a mem_cam_lookup16_add_out structure occupying two read transfer registers.

Table 3.706. mem_cam256_lookup16_add_ptr40 parameters

Type	Name	Description
mem_cam256_ptr40_t	<i>cam</i>	Pointer to the CAM in MEM
__xwrite mem_cam_lookup16_in_t*	<i>xfer</i>	Pointer to write xfer register containing the data to lookup

Type	Name	Description
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- mem_cam_lookup16_in_t, mem_cam_lookup16_add_out_t

3.21.4.69 mem_cam384_lookup16_add_ptr32

Prototype:

```
mem_cam_lookup16_add_out_in_read_reg_t* mem_cam384_lookup16_add_ptr32(__mem
mem_cam384_in_mem_t* cam, __xwrite mem_cam_lookup16_in_t* xfer, sync_t sync, SIGNAL_PAIR*
sig_pair_ptr)
```

Description:

Perform a 16-bit lookup in a 384-bit CAM in 32-bit addressed MEM and if a match is not found add the entry to the first empty slot.



Note

The result is returned in a mem_cam_lookup16_add_out structure occupying two read transfer registers.

Table 3.707. mem_cam384_lookup16_add_ptr32 parameters

Type	Name	Description
__mem mem_cam384_in_mem_t*	<i>cam</i>	Pointer to the CAM in MEM
__xwrite mem_cam_lookup16_in_t*	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- mem_cam_lookup16_in_t, mem_cam_lookup16_add_out_t

3.21.4.70 mem_cam384_lookup16_add_ptr40

Prototype:

```
mem_cam_lookup16_add_out_in_read_reg_t* mem_cam384_lookup16_add_ptr40(mem_cam384_ptr40_t
cam, __xwrite mem_cam_lookup16_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 16-bit lookup in a 384-bit CAM in 40-bit addressed MEM and if a match is not found add the entry to the first empty slot.



Note

The result is returned in a mem_cam_lookup16_add_out structure occupying two read transfer registers.

Table 3.708. mem_cam384_lookup16_add_ptr40 parameters

Type	Name	Description
mem_cam384_ptr40_t	cam	Pointer to the CAM in MEM
__xwrite mem_cam_lookup16_in_t*	xfer	Pointer to write xfer register containing the data to lookup
sync_t	sync	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	sig_pair_ptr	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- mem_cam_lookup16_in_t, mem_cam_lookup16_add_out_t

3.21.4.71 mem_cam512_lookup16_add_ptr32

Prototype:

```
mem_cam_lookup16_add_out_in_read_reg_t* mem_cam512_lookup16_add_ptr32(__mem
mem_cam512_in_mem_t* cam, __xwrite mem_cam_lookup16_in_t* xfer, sync_t sync, SIGNAL_PAIR*
sig_pair_ptr)
```

Description:

Perform a 16-bit lookup in a 512-bit CAM in 32-bit addressed MEM and if a match is not found add the entry to the first empty slot.



Note

The result is returned in a mem_cam_lookup16_add_out structure occupying two read transfer registers.

Table 3.709. mem_cam512_lookup16_add_ptr32 parameters

Type	Name	Description
__mem mem_cam512_in_mem_t*	<i>cam</i>	Pointer to the CAM in MEM
__xwrite mem_cam_lookup16_in_t*	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- mem_cam_lookup16_in_t, mem_cam_lookup16_add_out_t

3.21.4.72 mem_cam512_lookup16_add_ptr40

Prototype:

```
mem_cam_lookup16_add_out_in_read_reg_t* mem_cam512_lookup16_add_ptr40(mem_cam512_ptr40_t
cam, __xwrite mem_cam_lookup16_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 16-bit lookup in a 512-bit CAM in 40-bit addressed MEM and if a match is not found add the entry to the first empty slot.



Note

The result is returned in a mem_cam_lookup16_add_out structure occupying two read transfer registers.

Table 3.710. mem_cam512_lookup16_add_ptr40 parameters

Type	Name	Description
mem_cam512_ptr40_t	<i>cam</i>	Pointer to the CAM in MEM
__xwrite mem_cam_lookup16_in_t*	<i>xfer</i>	Pointer to write xfer register containing the data to lookup

Type	Name	Description
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- mem_cam_lookup16_in_t, mem_cam_lookup16_add_out_t

3.21.4.73 mem_cam128_lookup24_ptr32

Prototype:

```
mem_cam_lookup24_out_in_read_reg_t* mem_cam128_lookup24_ptr32(__mem mem_cam128_in_mem_t* cam, __xwrite mem_cam_lookup24_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 24-bit lookup in a 128-bit CAM in 32-bit addressed MEM.



Note

The result is returned in a mem_cam_lookup24_out structure in one read transfer register.

Table 3.711. mem_cam128_lookup24_ptr32 parameters

Type	Name	Description
__mem mem_cam128_in_mem_t*	<i>cam</i>	Pointer to the CAM structure in MEM
__xwrite mem_cam_lookup24_in_t*	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the read transfer register containing the result of the lookup.

See Also:

- mem_cam_lookup24_in_t, mem_cam_lookup24_out_t

3.21.4.74 mem_cam128_lookup24_ptr40

Prototype:

```
mem_cam_lookup24_out_in_read_reg_t* mem_cam128_lookup24_ptr40(mem_cam128_ptr40_t cam,  
__xwrite mem_cam_lookup24_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 24-bit lookup in a 128-bit CAM in 40-bit addressed MEM.

Note

The result is returned in a mem_cam_lookup24_out structure in one read transfer register.

Table 3.712. mem_cam128_lookup24_ptr40 parameters

Type	Name	Description
mem_cam128_ptr40_t	cam	Pointer to the CAM structure in MEM
__xwrite mem_cam_lookup24_in_t*	xfer	Pointer to write xfer register containing the data to lookup
sync_t	sync	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	sig_pair_ptr	Signal pair to raise upon completion

Returns:

Pointer to the read transfer register containing the result of the lookup.

See Also:

- mem_cam_lookup24_in_t, mem_cam_lookup24_out_t

3.21.4.75 mem_cam256_lookup24_ptr32

Prototype:

```
mem_cam_lookup24_out_in_read_reg_t* mem_cam256_lookup24_ptr32(__mem mem_cam256_in_mem_t*  
cam, __xwrite mem_cam_lookup24_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 24-bit lookup in a 256-bit CAM in 32-bit addressed MEM.

Note

The result is returned in a mem_cam_lookup24_out structure in one read transfer register.

Table 3.713. mem_cam256_lookup24_ptr32 parameters

Type	Name	Description
<code>__mem mem_cam256_in_mem_t*</code>	<code>cam</code>	Pointer to the CAM structure in MEM
<code>__xwrite mem_cam_lookup24_in_t*</code>	<code>xfer</code>	Pointer to write xfer register containing the data to lookup
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be sig_done)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal pair to raise upon completion

Returns:

Pointer to the read transfer register containing the result of the lookup.

See Also:

- `mem_cam_lookup24_in_t`, `mem_cam_lookup24_out_t`

3.21.4.76 mem_cam256_lookup24_ptr40

Prototype:

```
mem_cam_lookup24_out_in_read_reg_t* mem_cam256_lookup24_ptr40(mem_cam256_ptr40_t cam,  
__xwrite mem_cam_lookup24_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 24-bit lookup in a 256-bit CAM in 40-bit addressed MEM.



Note

The result is returned in a `mem_cam_lookup24_out` structure in one read transfer register.

Table 3.714. mem_cam256_lookup24_ptr40 parameters

Type	Name	Description
<code>mem_cam256_ptr40_t</code>	<code>cam</code>	Pointer to the CAM structure in MEM
<code>__xwrite mem_cam_lookup24_in_t*</code>	<code>xfer</code>	Pointer to write xfer register containing the data to lookup
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be sig_done)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal pair to raise upon completion

Returns:

Pointer to the read transfer register containing the result of the lookup.

See Also:

- mem_cam_lookup24_in_t, mem_cam_lookup24_out_t

3.21.4.77 mem_cam384_lookup24_ptr32

Prototype:

```
mem_cam_lookup24_out_in_read_reg_t* mem_cam384_lookup24_ptr32(__mem mem_cam384_in_mem_t* cam, __xwrite mem_cam_lookup24_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 24-bit lookup in a 384-bit CAM in 32-bit addressed MEM.



Note

The result is returned in a mem_cam_lookup24_out structure in one read transfer register.

Table 3.715. mem_cam384_lookup24_ptr32 parameters

Type	Name	Description
__mem mem_cam384_in_mem_t*	cam	Pointer to the CAM structure in MEM
__xwrite mem_cam_lookup24_in_t*	xfer	Pointer to write xfer register containing the data to lookup
sync_t	sync	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	sig_pair_ptr	Signal pair to raise upon completion

Returns:

Pointer to the read transfer register containing the result of the lookup.

See Also:

- mem_cam_lookup24_in_t, mem_cam_lookup24_out_t

3.21.4.78 mem_cam384_lookup24_ptr40

Prototype:

```
mem_cam_lookup24_out_in_read_reg_t* mem_cam384_lookup24_ptr40(mem_cam384_ptr40_t cam, __xwrite mem_cam_lookup24_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 24-bit lookup in a 384-bit CAM in 40-bit addressed MEM.



Note

The result is returned in a mem_cam_lookup24_out structure in one read transfer register.

Table 3.716. mem_cam384_lookup24_ptr40 parameters

Type	Name	Description
mem_cam384_ptr40_t	<i>cam</i>	Pointer to the CAM structure in MEM
__xwrite	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
mem_cam_lookup24_in_t*		
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the read transfer register containing the result of the lookup.

See Also:

- mem_cam_lookup24_in_t, mem_cam_lookup24_out_t

3.21.4.79 mem_cam512_lookup24_ptr32

Prototype:

```
mem_cam_lookup24_out_in_read_reg_t* mem_cam512_lookup24_ptr32(__mem mem_cam512_in_mem_t* cam, __xwrite mem_cam_lookup24_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 24-bit lookup in a 512-bit CAM in 32-bit addressed MEM.



Note

The result is returned in a mem_cam_lookup24_out structure in one read transfer register.

Table 3.717. mem_cam512_lookup24_ptr32 parameters

Type	Name	Description
__mem		
mem_cam512_in_mem_t*	<i>cam</i>	Pointer to the CAM structure in MEM
__xwrite	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
mem_cam_lookup24_in_t*		
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the read transfer register containing the result of the lookup.

See Also:

- `mem_cam_lookup24_in_t`, `mem_cam_lookup24_out_t`

3.21.4.80 mem_cam512_lookup24_ptr40

Prototype:

```
mem_cam_lookup24_out_in_read_reg_t* mem_cam512_lookup24_ptr40(mem_cam512_ptr40_t cam,  
__xwrite mem_cam_lookup24_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 24-bit lookup in a 512-bit CAM in 40-bit addressed MEM.



Note

The result is returned in a `mem_cam_lookup24_out` structure in one read transfer register.

Table 3.718. mem_cam512_lookup24_ptr40 parameters

Type	Name	Description
<code>mem_cam512_ptr40_t</code>	<code>cam</code>	Pointer to the CAM structure in MEM
<code>__xwrite mem_cam_lookup24_in_t*</code>	<code>xfer</code>	Pointer to write xfer register containing the data to lookup
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be <code>sig_done</code>)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal pair to raise upon completion

Returns:

Pointer to the read transfer register containing the result of the lookup.

See Also:

- `mem_cam_lookup24_in_t`, `mem_cam_lookup24_out_t`

3.21.4.81 mem_cam128_lookup24_add_ptr32

Prototype:

```
mem_cam_lookup24_add_out_in_read_reg_t* mem_cam128_lookup24_add_ptr32(__mem  
mem_cam128_in_mem_t* cam, __xwrite mem_cam_lookup24_add_in_t* xfer, sync_t sync,  
SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 24-bit lookup in a 128-bit CAM in 32-bit addressed MEM and if a match is not found add the entry to the first empty slot.



Note

The result is returned in a mem_cam_lookup24_add_out structure in a read xfer register.

Table 3.719. mem_cam128_lookup24_add_ptr32 parameters

Type	Name	Description
<code>__mem mem_cam128_in_mem_t* cam</code>		Pointer to the CAM in MEM
<code>__xwrite mem_cam_lookup24_add_in_t* xfer</code>		Pointer to write xfer register containing the data to lookup
<code>sync_t sync</code>		Type of synchronization to use (must be sig_done)
<code>SIGNAL_PAIR* sig_pair_ptr</code>		Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- `mem_cam_lookup24_in_t`, `mem_cam_lookup24_add_out_t`

3.21.4.82 mem_cam128_lookup24_add_ptr40

Prototype:

```
mem_cam_lookup24_add_out_in_read_reg_t* mem_cam128_lookup24_add_ptr40(mem_cam128_ptr40_t
cam, __xwrite mem_cam_lookup24_add_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 24-bit lookup in a 128-bit CAM in 40-bit addressed MEM and if a match is not found add the entry to the first empty slot.



Note

The result is returned in a mem_cam_lookup24_add_out structure in a read xfer register.

Table 3.720. mem_cam128_lookup24_add_ptr40 parameters

Type	Name	Description
<code>mem_cam128_ptr40_t cam</code>		Pointer to the CAM in MEM

Type	Name	Description
<code>__xwrite mem_cam_lookup24_add_in_t*</code>	<code>xfer</code>	Pointer to write xfer register containing the data to lookup
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be sig_done)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- `mem_cam_lookup24_in_t, mem_cam_lookup24_add_out_t`

3.21.4.83 mem_cam256_lookup24_add_ptr32

Prototype:

```
mem_cam_lookup24_add_out_in_read_reg_t* mem_cam256_lookup24_add_ptr32(__mem
mem_cam256_in_mem_t* cam, __xwrite mem_cam_lookup24_add_in_t* xfer, sync_t sync,
SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 24-bit lookup in a 256-bit CAM in 32-bit addressed MEM and if a match is not found add the entry to the first empty slot.



Note

The result is returned in a `mem_cam_lookup24_add_out` structure in a read xfer register.

Table 3.721. mem_cam256_lookup24_add_ptr32 parameters

Type	Name	Description
<code>__mem mem_cam256_in_mem_t*</code>	<code>cam</code>	Pointer to the CAM in MEM
<code>__xwrite mem_cam_lookup24_add_in_t*</code>	<code>xfer</code>	Pointer to write xfer register containing the data to lookup
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be sig_done)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- `mem_cam_lookup24_in_t, mem_cam_lookup24_add_out_t`

3.21.4.84 mem_cam256_lookup24_add_ptr40

Prototype:

```
mem_cam_lookup24_add_out_in_read_reg_t* mem_cam256_lookup24_add_ptr40(mem_cam256_ptr40_t
cam, __xwrite mem_cam_lookup24_add_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 24-bit lookup in a 256-bit CAM in 40-bit addressed MEM and if a match is not found add the entry to the first empty slot.

Note

The result is returned in a mem_cam_lookup24_add_out structure in a read xfer register.

Table 3.722. mem_cam256_lookup24_add_ptr40 parameters

Type	Name	Description
mem_cam256_ptr40_t	cam	Pointer to the CAM in MEM
__xwrite mem_cam_lookup24_add_in_t*	xfer	Pointer to write xfer register containing the data to lookup
sync_t	sync	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	sig_pair_ptr	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- mem_cam_lookup24_in_t, mem_cam_lookup24_add_out_t

3.21.4.85 mem_cam384_lookup24_add_ptr32

Prototype:

```
mem_cam_lookup24_add_out_in_read_reg_t* mem_cam384_lookup24_add_ptr32(__mem
mem_cam384_in_mem_t* cam, __xwrite mem_cam_lookup24_add_in_t* xfer, sync_t sync,
SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 24-bit lookup in a 384-bit CAM in 32-bit addressed MEM and if a match is not found add the entry to the first empty slot.



Note

The result is returned in a mem_cam_lookup24_add_out structure in a read xfer register.

Table 3.723. mem_cam384_lookup24_add_ptr32 parameters

Type	Name	Description
<code>__mem mem_cam384_in_mem_t*</code>	<code>cam</code>	Pointer to the CAM in MEM
<code>__xwrite</code>	<code>xfer</code>	Pointer to write xfer register containing the data to lookup
<code>mem_cam_lookup24_add_in_t*</code>		
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be sig_done)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- `mem_cam_lookup24_in_t`, `mem_cam_lookup24_add_out_t`

3.21.4.86 mem_cam384_lookup24_add_ptr40

Prototype:

```
mem_cam_lookup24_add_out_in_read_reg_t* mem_cam384_lookup24_add_ptr40(mem_cam384_ptr40_t
cam, __xwrite mem_cam_lookup24_add_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 24-bit lookup in a 384-bit CAM in 40-bit addressed MEM and if a match is not found add the entry to the first empty slot.



Note

The result is returned in a mem_cam_lookup24_add_out structure in a read xfer register.

Table 3.724. mem_cam384_lookup24_add_ptr40 parameters

Type	Name	Description
<code>mem_cam384_ptr40_t</code>	<code>cam</code>	Pointer to the CAM in MEM
<code>__xwrite</code>	<code>xfer</code>	Pointer to write xfer register containing the data to lookup
<code>mem_cam_lookup24_add_in_t*</code>		
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be sig_done)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- `mem_cam_lookup24_in_t`, `mem_cam_lookup24_add_out_t`

3.21.4.87 mem_cam512_lookup24_add_ptr32

Prototype:

```
mem_cam_lookup24_add_out_in_read_reg_t* mem_cam512_lookup24_add_ptr32(__mem
mem_cam512_in_mem_t* cam, __xwrite mem_cam_lookup24_add_in_t* xfer, sync_t sync,
SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 24-bit lookup in a 512-bit CAM in 32-bit addressed MEM and if a match is not found add the entry to the first empty slot.



Note

The result is returned in a `mem_cam_lookup24_add_out` structure in a read xfer register.

Table 3.725. mem_cam512_lookup24_add_ptr32 parameters

Type	Name	Description
<code>__mem mem_cam512_in_mem_t*</code>	<code>cam</code>	Pointer to the CAM in MEM
<code>__xwrite mem_cam_lookup24_add_in_t*</code>	<code>xfer</code>	Pointer to write xfer register containing the data to lookup
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be <code>sig_done</code>)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- `mem_cam_lookup24_in_t`, `mem_cam_lookup24_add_out_t`

3.21.4.88 mem_cam512_lookup24_add_ptr40

Prototype:

```
mem_cam_lookup24_add_out_in_read_reg_t* mem_cam512_lookup24_add_ptr40(mem_cam512_ptr40_t
cam, __xwrite mem_cam_lookup24_add_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 24-bit lookup in a 512-bit CAM in 40-bit addressed MEM and if a match is not found add the entry to the first empty slot.



Note

The result is returned in a mem_cam_lookup24_add_out structure in a read xfer register.

Table 3.726. mem_cam512_lookup24_add_ptr40 parameters

Type	Name	Description
mem_cam512_ptr40_t	<i>cam</i>	Pointer to the CAM in MEM
__xwrite	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
mem_cam_lookup24_add_in_t*		
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- mem_cam_lookup24_in_t, mem_cam_lookup24_add_out_t

3.21.4.89 mem_cam128_lookup24_add_inc_ptr32

Prototype:

```
mem_cam_lookup24_add_inc_out_in_read_reg_t* mem_cam128_lookup24_add_inc_ptr32(__mem
mem_cam128_in_mem_t* cam, __xwrite mem_cam_lookup24_add_inc_in_t* xfer, sync_t sync,
SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 24-bit lookup in a 128-bit CAM in 32-bit addressed MEM and if a match is not found add the entry to the first empty slot.

The result is returned in a mem_cam_lookup24_add_out structure in a read xfer register.

When an entry is found the upper 8-bits of the entry (count) is incremented in a non saturating way. The pre-incremented value of count is returned in the count field of the mem_cam_lookup24_add_out_t structure.

Table 3.727. mem_cam128_lookup24_add_inc_ptr32 parameters

Type	Name	Description
__mem mem_cam128_in_mem_t*	<i>cam</i>	Pointer to the CAM in MEM

Type	Name	Description
<code>__xwrite</code> <code>mem_cam_lookup24_add_inc_in_t*</code>	<code>xfer</code>	Pointer to write xfer register containing the data to lookup
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be <code>sig_done</code>)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal pair to raise upon completion

Returns:

Pointer to the read transfer register containing the result of the lookup.

See Also:

- `mem_cam_lookup24_in_t`, `mem_cam_lookup24_add_out_t`

3.21.4.90 mem_cam128_lookup24_add_inc_ptr40

Prototype:

```
mem_cam_lookup24_add_inc_out_in_read_reg_t*
mem_cam128_lookup24_add_inc_ptr40(mem_cam128_ptr40_t cam, __xwrite
mem_cam_lookup24_add_inc_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 24-bit lookup in a 128-bit CAM in 40-bit addressed MEM and if a match is not found add the entry to the first empty slot.

The result is returned in a `mem_cam_lookup24_add_out` structure in a read xfer register.

When an entry is found the upper 8-bits of the entry (count) is incremented in a non saturating way. The pre-incremented value of count is returned in the count field of the `mem_cam_lookup24_add_out_t` structure.

Table 3.728. mem_cam128_lookup24_add_inc_ptr40 parameters

Type	Name	Description
<code>mem_cam128_ptr40_t</code>	<code>cam</code>	Pointer to the CAM in MEM
<code>__xwrite</code> <code>mem_cam_lookup24_add_inc_in_t*</code>	<code>xfer</code>	Pointer to write xfer register containing the data to lookup
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be <code>sig_done</code>)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal pair to raise upon completion

Returns:

Pointer to the read transfer register containing the result of the lookup.

See Also:

- `mem_cam_lookup24_in_t`, `mem_cam_lookup24_add_out_t`

3.21.4.91 mem_cam256_lookup24_add_inc_ptr32

Prototype:

```
mem_cam_lookup24_add_inc_out_in_read_reg_t* mem_cam256_lookup24_add_inc_ptr32(__mem
mem_cam256_in_mem_t* cam, __xwrite mem_cam_lookup24_add_inc_in_t* xfer, sync_t sync,
SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 24-bit lookup in a 256-bit CAM in 32-bit addressed MEM and if a match is not found add the entry to the first empty slot.

The result is returned in a mem_cam_lookup24_add_out structure in a read xfer register.

When an entry is found the upper 8-bits of the entry (count) is incremented in a non saturating way. The pre-incremented value of count is returned in the count field of the mem_cam_lookup24_add_out_t structure.

Table 3.729. mem_cam256_lookup24_add_inc_ptr32 parameters

Type	Name	Description
<code>__mem mem_cam256_in_mem_t*</code>	<code>cam</code>	Pointer to the CAM in MEM
<code>__xwrite mem_cam_lookup24_add_inc_in_t*</code>	<code>xfer</code>	Pointer to write xfer register containing the data to lookup
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be sig_done)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal pair to raise upon completion

Returns:

Pointer to the read transfer register containing the result of the lookup.

See Also:

- `mem_cam_lookup24_in_t`, `mem_cam_lookup24_add_out_t`

3.21.4.92 mem_cam256_lookup24_add_inc_ptr40

Prototype:

```
mem_cam_lookup24_add_inc_out_in_read_reg_t*
mem_cam256_lookup24_add_inc_ptr40(mem_cam256_ptr40_t cam, __xwrite
mem_cam_lookup24_add_inc_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 24-bit lookup in a 256-bit CAM in 40-bit addressed MEM and if a match is not found add the entry to the first empty slot.

The result is returned in a mem_cam_lookup24_add_out structure in a read xfer register.

When an entry is found the upper 8-bits of the entry (count) is incremented in a non saturating way. The pre-incremented value of count is returned in the count field of the `mem_cam_lookup24_add_out_t` structure.

Table 3.730. mem_cam256_lookup24_add_inc_ptr40 parameters

Type	Name	Description
<code>mem_cam256_ptr40_t</code>	<i>cam</i>	Pointer to the CAM in MEM
<code>__xwrite</code> <code>mem_cam_lookup24_add_inc_in_t*</code>	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
<code>sync_t</code>	<i>sync</i>	Type of synchronization to use (must be <code>sig_done</code>)
<code>SIGNAL_PAIR*</code>	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the read transfer register containing the result of the lookup.

See Also:

- `mem_cam_lookup24_in_t`, `mem_cam_lookup24_add_out_t`

3.21.4.93 mem_cam384_lookup24_add_inc_ptr32

Prototype:

```
mem_cam_lookup24_add_inc_out_in_read_reg_t* mem_cam384_lookup24_add_inc_ptr32(__mem
mem_cam384_in_mem_t* cam, __xwrite mem_cam_lookup24_add_inc_in_t* xfer, sync_t sync,
SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 24-bit lookup in a 384-bit CAM in 32-bit addressed MEM and if a match is not found add the entry to the first empty slot.

The result is returned in a `mem_cam_lookup24_add_out` structure in a read xfer register.

When an entry is found the upper 8-bits of the entry (count) is incremented in a non saturating way. The pre-incremented value of count is returned in the count field of the `mem_cam_lookup24_add_out_t` structure.

Table 3.731. mem_cam384_lookup24_add_inc_ptr32 parameters

Type	Name	Description
<code>__mem mem_cam384_in_mem_t*</code>	<i>cam</i>	Pointer to the CAM in MEM
<code>__xwrite</code> <code>mem_cam_lookup24_add_inc_in_t*</code>	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
<code>sync_t</code>	<i>sync</i>	Type of synchronization to use (must be <code>sig_done</code>)
<code>SIGNAL_PAIR*</code>	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the read transfer register containing the result of the lookup.

See Also:

- `mem_cam_lookup24_in_t`, `mem_cam_lookup24_add_out_t`

3.21.4.94 mem_cam384_lookup24_add_inc_ptr40

Prototype:

```
mem_cam_lookup24_add_inc_out_in_read_reg_t*
mem_cam384_lookup24_add_inc_ptr40(mem_cam384_ptr40_t cam, __xwrite
mem_cam_lookup24_add_inc_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 24-bit lookup in a 384-bit CAM in 40-bit addressed MEM and if a match is not found add the entry to the first empty slot.

The result is returned in a `mem_cam_lookup24_add_out` structure in a read xfer register.

When an entry is found the upper 8-bits of the entry (count) is incremented in a non saturating way. The pre-incremented value of count is returned in the count field of the `mem_cam_lookup24_add_out_t` structure.

Table 3.732. mem_cam384_lookup24_add_inc_ptr40 parameters

Type	Name	Description
<code>mem_cam384_ptr40_t</code>	<i>cam</i>	Pointer to the CAM in MEM
<code>__xwrite</code> <code>mem_cam_lookup24_add_inc_in_t*</code>	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
<code>sync_t</code>	<i>sync</i>	Type of synchronization to use (must be <code>sig_done</code>)
<code>SIGNAL_PAIR*</code>	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the read transfer register containing the result of the lookup.

See Also:

- `mem_cam_lookup24_in_t`, `mem_cam_lookup24_add_out_t`

3.21.4.95 mem_cam512_lookup24_add_inc_ptr32

Prototype:

```
mem_cam_lookup24_add_inc_out_in_read_reg_t* mem_cam512_lookup24_add_inc_ptr32(__mem
mem_cam512_in_mem_t* cam, __xwrite mem_cam_lookup24_add_inc_in_t* xfer, sync_t sync,
SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 24-bit lookup in a 512-bit CAM in 32-bit addressed MEM and if a match is not found add the entry to the first empty slot.

The result is returned in a mem_cam_lookup24_add_out structure in a read xfer register.

When an entry is found the upper 8-bits of the entry (count) is incremented in a non saturating way. The pre-incremented value of count is returned in the count field of the mem_cam_lookup24_add_out_t structure.

Table 3.733. mem_cam512_lookup24_add_inc_ptr32 parameters

Type	Name	Description
__mem mem_cam512_in_mem_t*	<i>cam</i>	Pointer to the CAM in MEM
__xwrite mem_cam_lookup24_add_inc_in_t*	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the read transfer register containing the result of the lookup.

See Also:

- mem_cam_lookup24_in_t, mem_cam_lookup24_add_out_t

3.21.4.96 mem_cam512_lookup24_add_inc_ptr40

Prototype:

```
mem_cam_lookup24_add_inc_out_in_read_reg_t*
mem_cam512_lookup24_add_inc_ptr40(mem_cam512_ptr40_t cam, __xwrite
mem_cam_lookup24_add_inc_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 24-bit lookup in a 512-bit CAM in 40-bit addressed MEM and if a match is not found add the entry to the first empty slot.

The result is returned in a mem_cam_lookup24_add_out structure in a read xfer register.

When an entry is found the upper 8-bits of the entry (count) is incremented in a non saturating way. The pre-incremented value of count is returned in the count field of the mem_cam_lookup24_add_out_t structure.

Table 3.734. mem_cam512_lookup24_add_inc_ptr40 parameters

Type	Name	Description
mem_cam512_ptr40_t	<i>cam</i>	Pointer to the CAM in MEM

Type	Name	Description
<code>__xwrite mem_cam_lookup24_add_inc_in_t*</code>	<code>xfer</code>	Pointer to write xfer register containing the data to lookup
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be sig_done)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal pair to raise upon completion

Returns:

Pointer to the read transfer register containing the result of the lookup.

See Also:

- `mem_cam_lookup24_in_t, mem_cam_lookup24_add_out_t`

3.21.4.97 mem_cam128_lookup32_ptr32

Prototype:

```
mem_cam_lookup32_out_in_read_reg_t* mem_cam128_lookup32_ptr32(__mem mem_cam128_in_mem_t* cam, __xwrite mem_cam_lookup32_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 32-bit lookup in a 128-bit CAM in 32-bit addressed MEM.



Note

The result is returned in a `mem_cam_lookup32_out` structure in one read transfer register.

Table 3.735. mem_cam128_lookup32_ptr32 parameters

Type	Name	Description
<code>__mem mem_cam128_in_mem_t*</code>	<code>cam</code>	Pointer to the CAM structure in MEM
<code>__xwrite mem_cam_lookup32_in_t*</code>	<code>xfer</code>	Pointer to write xfer register containing the data to lookup
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be sig_done)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal pair to raise upon completion

Returns:

Pointer to the read transfer register containing the result of the lookup.

See Also:

- `mem_cam_lookup32_in_t, mem_cam_lookup32_out_t`

3.21.4.98 mem_cam128_lookup32_ptr40

Prototype:

```
mem_cam_lookup32_out_in_read_reg_t* mem_cam128_lookup32_ptr40(mem_cam128_ptr40_t cam,  
__xwrite mem_cam_lookup32_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 32-bit lookup in a 128-bit CAM in 40-bit addressed MEM.

 **Note**

The result is returned in a mem_cam_lookup32_out structure in one read transfer register.

Table 3.736. mem_cam128_lookup32_ptr40 parameters

Type	Name	Description
mem_cam128_ptr40_t	cam	Pointer to the CAM structure in MEM
__xwrite mem_cam_lookup32_in_t*	xfer	Pointer to write xfer register containing the data to lookup
sync_t	sync	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	sig_pair_ptr	Signal pair to raise upon completion

Returns:

Pointer to the read transfer register containing the result of the lookup.

See Also:

- mem_cam_lookup32_in_t, mem_cam_lookup32_out_t

3.21.4.99 mem_cam256_lookup32_ptr32

Prototype:

```
mem_cam_lookup32_out_in_read_reg_t* mem_cam256_lookup32_ptr32(__mem mem_cam256_in_mem_t*  
cam, __xwrite mem_cam_lookup32_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 32-bit lookup in a 256-bit CAM in 32-bit addressed MEM.

 **Note**

The result is returned in a mem_cam_lookup32_out structure in one read transfer register.

Table 3.737. mem_cam256_lookup32_ptr32 parameters

Type	Name	Description
<code>__mem mem_cam256_in_mem_t*</code>	<code>cam</code>	Pointer to the CAM structure in MEM
<code>__xwrite mem_cam_lookup32_in_t*</code>	<code>xfer</code>	Pointer to write xfer register containing the data to lookup
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be sig_done)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal pair to raise upon completion

Returns:

Pointer to the read transfer register containing the result of the lookup.

See Also:

- `mem_cam_lookup32_in_t`, `mem_cam_lookup32_out_t`

3.21.4.100 mem_cam256_lookup32_ptr40

Prototype:

```
mem_cam_lookup32_out_in_read_reg_t* mem_cam256_lookup32_ptr40(mem_cam256_ptr40_t cam,  
__xwrite mem_cam_lookup32_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 32-bit lookup in a 256-bit CAM in 40-bit addressed MEM.



Note

The result is returned in a `mem_cam_lookup32_out` structure in one read transfer register.

Table 3.738. mem_cam256_lookup32_ptr40 parameters

Type	Name	Description
<code>mem_cam256_ptr40_t</code>	<code>cam</code>	Pointer to the CAM structure in MEM
<code>__xwrite mem_cam_lookup32_in_t*</code>	<code>xfer</code>	Pointer to write xfer register containing the data to lookup
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be sig_done)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal pair to raise upon completion

Returns:

Pointer to the read transfer register containing the result of the lookup.

See Also:

- mem_cam_lookup32_in_t, mem_cam_lookup32_out_t

3.21.4.101 mem_cam384_lookup32_ptr32

Prototype:

```
mem_cam_lookup32_out_in_read_reg_t* mem_cam384_lookup32_ptr32(__mem mem_cam384_in_mem_t* cam, __xwrite mem_cam_lookup32_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 32-bit lookup in a 384-bit CAM in 32-bit addressed MEM.



Note

The result is returned in a mem_cam_lookup32_out structure in one read transfer register.

Table 3.739. mem_cam384_lookup32_ptr32 parameters

Type	Name	Description
__mem mem_cam384_in_mem_t*	cam	Pointer to the CAM structure in MEM
__xwrite mem_cam_lookup32_in_t*	xfer	Pointer to write xfer register containing the data to lookup
sync_t	sync	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	sig_pair_ptr	Signal pair to raise upon completion

Returns:

Pointer to the read transfer register containing the result of the lookup.

See Also:

- mem_cam_lookup32_in_t, mem_cam_lookup32_out_t

3.21.4.102 mem_cam384_lookup32_ptr40

Prototype:

```
mem_cam_lookup32_out_in_read_reg_t* mem_cam384_lookup32_ptr40(mem_cam384_ptr40_t cam, __xwrite mem_cam_lookup32_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 32-bit lookup in a 384-bit CAM in 40-bit addressed MEM.



Note

The result is returned in a mem_cam_lookup32_out structure in one read transfer register.

Table 3.740. mem_cam384_lookup32_ptr40 parameters

Type	Name	Description
mem_cam384_ptr40_t	<i>cam</i>	Pointer to the CAM structure in MEM
__xwrite	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
mem_cam_lookup32_in_t*		
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the read transfer register containing the result of the lookup.

See Also:

- mem_cam_lookup32_in_t, mem_cam_lookup32_out_t

3.21.4.103 mem_cam512_lookup32_ptr32

Prototype:

```
mem_cam_lookup32_out_in_read_reg_t* mem_cam512_lookup32_ptr32(__mem mem_cam512_in_mem_t* cam, __xwrite mem_cam_lookup32_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 32-bit lookup in a 512-bit CAM in 32-bit addressed MEM.



Note

The result is returned in a mem_cam_lookup32_out structure in one read transfer register.

Table 3.741. mem_cam512_lookup32_ptr32 parameters

Type	Name	Description
__mem		
mem_cam512_in_mem_t*	<i>cam</i>	Pointer to the CAM structure in MEM
__xwrite	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
mem_cam_lookup32_in_t*		
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the read transfer register containing the result of the lookup.

See Also:

- `mem_cam_lookup32_in_t`, `mem_cam_lookup32_out_t`

3.21.4.104 mem_cam512_lookup32_ptr40

Prototype:

```
mem_cam_lookup32_out_in_read_reg_t* mem_cam512_lookup32_ptr40(mem_cam512_ptr40_t cam,  
__xwrite mem_cam_lookup32_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 32-bit lookup in a 512-bit CAM in 40-bit addressed MEM.



Note

The result is returned in a `mem_cam_lookup32_out` structure in one read transfer register.

Table 3.742. mem_cam512_lookup32_ptr40 parameters

Type	Name	Description
<code>mem_cam512_ptr40_t</code>	<code>cam</code>	Pointer to the CAM structure in MEM
<code>__xwrite mem_cam_lookup32_in_t*</code>	<code>xfer</code>	Pointer to write xfer register containing the data to lookup
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be <code>sig_done</code>)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal pair to raise upon completion

Returns:

Pointer to the read transfer register containing the result of the lookup.

See Also:

- `mem_cam_lookup32_in_t`, `mem_cam_lookup32_out_t`

3.21.4.105 mem_cam128_lookup32_add_ptr32

Prototype:

```
mem_cam_lookup32_add_out_in_read_reg_t* mem_cam128_lookup32_add_ptr32(__mem  
mem_cam128_in_mem_t* cam, __xwrite mem_cam_lookup32_in_t* xfer, sync_t sync, SIGNAL_PAIR*  
sig_pair_ptr)
```

Description:

Perform a 32-bit lookup in a 128-bit CAM in 32-bit addressed MEM and if a match is not found add the entry to the first empty slot.



Note

The result is returned in a mem_cam_lookup32_add_out structure in a read xfer register.

Table 3.743. mem_cam128_lookup32_add_ptr32 parameters

Type	Name	Description
<code>__mem mem_cam128_in_mem_t*</code>	<code>cam</code>	Pointer to the CAM in MEM
<code>__xwrite mem_cam_lookup32_in_t*</code>	<code>xfer</code>	Pointer to write xfer register containing the data to lookup
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be sig_done)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- `mem_cam_lookup32_in_t`, `mem_cam_lookup32_add_out_t`

3.21.4.106 mem_cam128_lookup32_add_ptr40

Prototype:

```
mem_cam_lookup32_add_out_in_read_reg_t* mem_cam128_lookup32_add_ptr40(mem_cam128_ptr40_t
cam, __xwrite mem_cam_lookup32_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 32-bit lookup in a 128-bit CAM in 40-bit addressed MEM and if a match is not found add the entry to the first empty slot.



Note

The result is returned in a mem_cam_lookup32_add_out structure in a read xfer register.

Table 3.744. mem_cam128_lookup32_add_ptr40 parameters

Type	Name	Description
<code>mem_cam128_ptr40_t</code>	<code>cam</code>	Pointer to the CAM in MEM

Type	Name	Description
<code>__xwrite mem_cam_lookup32_in_t*</code>	<code>xfer</code>	Pointer to write xfer register containing the data to lookup
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be sig_done)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- `mem_cam_lookup32_in_t, mem_cam_lookup32_add_out_t`

3.21.4.107 mem_cam256_lookup32_add_ptr32

Prototype:

```
mem_cam_lookup32_add_out_in_read_reg_t* mem_cam256_lookup32_add_ptr32(__mem
mem_cam256_in_mem_t* cam, __xwrite mem_cam_lookup32_in_t* xfer, sync_t sync, SIGNAL_PAIR*
sig_pair_ptr)
```

Description:

Perform a 32-bit lookup in a 256-bit CAM in 32-bit addressed MEM and if a match is not found add the entry to the first empty slot.



Note

The result is returned in a `mem_cam_lookup32_add_out` structure in a read xfer register.

Table 3.745. mem_cam256_lookup32_add_ptr32 parameters

Type	Name	Description
<code>__mem mem_cam256_in_mem_t*</code>	<code>cam</code>	Pointer to the CAM in MEM
<code>__xwrite mem_cam_lookup32_in_t*</code>	<code>xfer</code>	Pointer to write xfer register containing the data to lookup
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be sig_done)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- mem_cam_lookup32_in_t, mem_cam_lookup32_add_out_t

3.21.4.108 mem_cam256_lookup32_add_ptr40

Prototype:

```
mem_cam_lookup32_add_out_in_read_reg_t* mem_cam256_lookup32_add_ptr40(mem_cam256_ptr40_t
cam, __xwrite mem_cam_lookup32_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 32-bit lookup in a 256-bit CAM in 40-bit addressed MEM and if a match is not found add the entry to the first empty slot.



Note

The result is returned in a mem_cam_lookup32_add_out structure in a read xfer register.

Table 3.746. mem_cam256_lookup32_add_ptr40 parameters

Type	Name	Description
mem_cam256_ptr40_t	<i>cam</i>	Pointer to the CAM in MEM
__xwrite mem_cam_lookup32_in_t*	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- mem_cam_lookup32_in_t, mem_cam_lookup32_add_out_t

3.21.4.109 mem_cam384_lookup32_add_ptr32

Prototype:

```
mem_cam_lookup32_add_out_in_read_reg_t* mem_cam384_lookup32_add_ptr32(__mem
mem_cam384_in_mem_t* cam, __xwrite mem_cam_lookup32_in_t* xfer, sync_t sync, SIGNAL_PAIR*
sig_pair_ptr)
```

Description:

Perform a 32-bit lookup in a 384-bit CAM in 32-bit addressed MEM and if a match is not found add the entry to the first empty slot.



Note

The result is returned in a mem_cam_lookup32_add_out structure in a read xfer register.

Table 3.747. mem_cam384_lookup32_add_ptr32 parameters

Type	Name	Description
__mem mem_cam384_in_mem_t*	<i>cam</i>	Pointer to the CAM in MEM
__xwrite mem_cam_lookup32_in_t*	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- mem_cam_lookup32_in_t, mem_cam_lookup32_add_out_t

3.21.4.110 mem_cam384_lookup32_add_ptr40

Prototype:

```
mem_cam_lookup32_add_out_in_read_reg_t* mem_cam384_lookup32_add_ptr40(mem_cam384_ptr40_t
cam, __xwrite mem_cam_lookup32_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 32-bit lookup in a 384-bit CAM in 40-bit addressed MEM and if a match is not found add the entry to the first empty slot.



Note

The result is returned in a mem_cam_lookup32_add_out structure in a read xfer register.

Table 3.748. mem_cam384_lookup32_add_ptr40 parameters

Type	Name	Description
mem_cam384_ptr40_t	<i>cam</i>	Pointer to the CAM in MEM
__xwrite mem_cam_lookup32_in_t*	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- `mem_cam_lookup32_in_t`, `mem_cam_lookup32_add_out_t`

3.21.4.111 mem_cam512_lookup32_add_ptr32

Prototype:

```
mem_cam_lookup32_add_out_in_read_reg_t* mem_cam512_lookup32_add_ptr32(__mem
mem_cam512_in_mem_t* cam, __xwrite mem_cam_lookup32_in_t* xfer, sync_t sync, SIGNAL_PAIR*
sig_pair_ptr)
```

Description:

Perform a 32-bit lookup in a 512-bit CAM in 32-bit addressed MEM and if a match is not found add the entry to the first empty slot.



Note

The result is returned in a `mem_cam_lookup32_add_out` structure in a read xfer register.

Table 3.749. mem_cam512_lookup32_add_ptr32 parameters

Type	Name	Description
<code>__mem</code> <code>mem_cam512_in_mem_t*</code>	<code>cam</code>	Pointer to the CAM in MEM
<code>__xwrite</code> <code>mem_cam_lookup32_in_t*</code>	<code>xfer</code>	Pointer to write xfer register containing the data to lookup
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be <code>sig_done</code>)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- `mem_cam_lookup32_in_t`, `mem_cam_lookup32_add_out_t`

3.21.4.112 mem_cam512_lookup32_add_ptr40

Prototype:

```
mem_cam_lookup32_add_out_in_read_reg_t* mem_cam512_lookup32_add_ptr40(mem_cam512_ptr40_t
cam, __xwrite mem_cam_lookup32_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 32-bit lookup in a 512-bit CAM in 40-bit addressed MEM and if a match is not found add the entry to the first empty slot.



Note

The result is returned in a mem_cam_lookup32_add_out structure in a read xfer register.

Table 3.750. mem_cam512_lookup32_add_ptr40 parameters

Type	Name	Description
mem_cam512_ptr40_t	<i>cam</i>	Pointer to the CAM in MEM
__xwrite mem_cam_lookup32_in_t*	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup.

See Also:

- mem_cam_lookup32_in_t, mem_cam_lookup32_add_out_t

3.22 MEM TCAM Intrinsics

This section discusses the enumerations, structs, unions, types and functions related to MEM TCAM operations.

3.22.1 MU TCAM Defines

Table 3.751. MU TCAM Defines

Defined	Definition
mem_tcam128_set24_word_ptr32	mem_tcam128_set32_word_ptr32 Alias for mem_tcam128_set32_word.

Defined	Definition
	<p> Note</p> <p>A 24 bit word TCAM is the same as a 32 bit word TCAM. The difference is seen when lookups are performed. For a 24 bit word TCAM, the full 32 bit word can be set (the highest 8 bits are not masked off). When a 24 bit lookup is performed, the highest 8 bits are not compared, but will be returned for the first match.</p>
mem_tcam128_set24_entry_value_ptr32	<p>mem_tcam128_set32_entry_value_ptr32</p> <p>Alias for mem_tcam128_set32_entry_value.</p> <p> Note</p> <p>A 24 bit word TCAM is the same as a 32 bit word TCAM. The difference is seen when lookups are performed. For a 24 bit word TCAM, the full 32 bit word can be set (the highest 8 bits are not masked off). When a 24 bit lookup is performed, the highest 8 bits are not compared, but will be returned for the first match.</p>
mem_tcam128_set24_entry_value_ptr40	mem_tcam128_set32_entry_value_ptr40
mem_tcam128_set24_entry_mask_ptr32	<p>mem_tcam128_set32_entry_mask_ptr32</p> <p>Alias for mem_tcam128_set32_entry_mask.</p> <p> Note</p> <p>A 24 bit word TCAM is the same as a 32 bit word TCAM. The difference is seen when lookups are performed. For a 24 bit word TCAM, the full 32 bit word can be set (the highest 8 bits are not masked off). When a 24 bit lookup is performed, the highest 8 bits are not compared, but will be returned for the first match.</p>
mem_tcam128_set24_entry_mask_ptr40	mem_tcam128_set32_entry_mask_ptr40

Defined	Definition
mem_tcam128_set24_entry_ptr32	<p>mem_tcam128_set32_entry_ptr32</p> <p>Alias for mem_tcam128_set32_entry.</p> <div style="border: 1px solid blue; padding: 10px;">  Note A 24 bit word TCAM is the same as a 32 bit word TCAM. The difference is seen when lookups are performed. For a 24 bit word TCAM, the full 32 bit word can be set (the highest 8 bits are not masked off). When a 24 bit lookup is performed, the highest 8 bits are not compared, but will be returned for the first match. </div>
mem_tcam128_set24_entry_ptr40	mem_tcam128_set32_entry_ptr40
mem_tcam256_set24_word_ptr32	<p>mem_tcam256_set32_word_ptr32</p> <p>Alias for mem_tcam256_set32_word.</p> <div style="border: 1px solid blue; padding: 10px;">  Note A 24 bit word TCAM is the same as a 32 bit word TCAM. The difference is seen when lookups are performed. For a 24 bit word TCAM, the full 32 bit word can be set (the highest 8 bits are not masked off). When a 24 bit lookup is performed, the highest 8 bits are not compared, but will be returned for the first match. </div>
mem_tcam256_set24_word_ptr40	mem_tcam256_set32_word_ptr40
mem_tcam256_set24_entry_value_ptr32	<p>mem_tcam256_set32_entry_value</p> <p>Alias for mem_tcam256_set32_entry_value.</p> <div style="border: 1px solid blue; padding: 10px;">  Note A 24 bit word TCAM is the same as a 32 bit word TCAM. The difference is seen when lookups are performed. For a 24 bit word TCAM, the full 32 bit word can be set (the highest 8 bits are not masked off). When a 24 bit lookup is performed, the highest 8 bits are </div>

Defined	Definition
	<p>not compared, but will be returned for the first match.</p>
mem_tcam256_set24_entry_value_ptr40	mem_tcam256_set32_entry_value_ptr40
mem_tcam256_set24_entry_mask_ptr32	<p>mem_tcam256_set32_entry_mask</p> <p>Alias for mem_tcam256_set32_entry_mask.</p> <div style="border: 1px solid blue; padding: 10px;">  <p>Note</p> <p>A 24 bit word TCAM is the same as a 32 bit word TCAM. The difference is seen when lookups are performed. For a 24 bit word TCAM, the full 32 bit word can be set (the highest 8 bits are not masked off). When a 24 bit lookup is performed, the highest 8 bits are not compared, but will be returned for the first match.</p> </div>
mem_tcam256_set24_entry_mask_ptr40	mem_tcam256_set32_entry_mask_ptr40
mem_tcam256_set24_entry_ptr32	<p>mem_tcam256_set32_entry</p> <p>Alias for mem_tcam256_set32_entry.</p> <div style="border: 1px solid blue; padding: 10px;">  <p>Note</p> <p>A 24 bit word TCAM is the same as a 32 bit word TCAM. The difference is seen when lookups are performed. For a 24 bit word TCAM, the full 32 bit word can be set (the highest 8 bits are not masked off). When a 24 bit lookup is performed, the highest 8 bits are not compared, but will be returned for the first match.</p> </div>
mem_tcam256_set24_entry_ptr40	mem_tcam256_set32_entry_ptr40
mem_tcam384_set24_word_ptr32	<p>mem_tcam384_set32_word</p> <p>Alias for mem_tcam384_set32_word.</p>

Defined	Definition
	<p> Note</p> <p>A 24 bit word TCAM is the same as a 32 bit word TCAM. The difference is seen when lookups are performed. For a 24 bit word TCAM, the full 32 bit word can be set (the highest 8 bits are not masked off). When a 24 bit lookup is performed, the highest 8 bits are not compared, but will be returned for the first match.</p>
mem_tcam384_set24_word_ptr40	mem_tcam384_set32_word_ptr40
mem_tcam384_set24_entry_value_ptr32	<p>mem_tcam384_set32_entry_value</p> <p>Alias for mem_tcam384_set32_entry_value.</p> <p> Note</p> <p>A 24 bit word TCAM is the same as a 32 bit word TCAM. The difference is seen when lookups are performed. For a 24 bit word TCAM, the full 32 bit word can be set (the highest 8 bits are not masked off). When a 24 bit lookup is performed, the highest 8 bits are not compared, but will be returned for the first match.</p>
mem_tcam384_set24_entry_value_ptr40	mem_tcam384_set32_entry_value_ptr40
mem_tcam384_set24_entry_mask_ptr32	<p>mem_tcam384_set32_entry_mask</p> <p>Alias for mem_tcam384_set32_entry_mask.</p> <p> Note</p> <p>A 24 bit word TCAM is the same as a 32 bit word TCAM. The difference is seen when lookups are performed. For a 24 bit word TCAM, the full 32 bit word can be set (the highest 8 bits are not masked off). When a 24 bit lookup is performed, the highest 8 bits are not compared, but will be returned for the first match.</p>

Defined	Definition
mem_tcam384_set24_entry_mask_ptr40	mem_tcam384_set32_entry_mask_ptr40
mem_tcam384_set24_entry_ptr32	mem_tcam384_set32_entry Alias for mem_tcam384_set32_entry.
	<p> Note</p> <p>A 24 bit word TCAM is the same as a 32 bit word TCAM. The difference is seen when lookups are performed. For a 24 bit word TCAM, the full 32 bit word can be set (the highest 8 bits are not masked off). When a 24 bit lookup is performed, the highest 8 bits are not compared, but will be returned for the first match.</p>
mem_tcam384_set24_entry_ptr40	mem_tcam384_set32_entry_ptr40
mem_tcam512_set24_word_ptr32	mem_tcam512_set32_word Alias for mem_tcam512_set32_word.
	<p> Note</p> <p>A 24 bit word TCAM is the same as a 32 bit word TCAM. The difference is seen when lookups are performed. For a 24 bit word TCAM, the full 32 bit word can be set (the highest 8 bits are not masked off). When a 24 bit lookup is performed, the highest 8 bits are not compared, but will be returned for the first match.</p>
mem_tcam512_set24_word_ptr40	mem_tcam512_set32_word_ptr40
mem_tcam512_set24_entry_value_ptr32	mem_tcam512_set32_entry_value Alias for mem_tcam512_set32_entry_value.
	<p> Note</p> <p>A 24 bit word TCAM is the same as a 32 bit word TCAM. The difference is seen when lookups are performed. For a 24 bit word TCAM, the full 32 bit word can be set (the highest 8 bits are not masked off). When a 24</p>

Defined	Definition
	<p>bit lookup is performed, the highest 8 bits are not compared, but will be returned for the first match.</p>
mem_tcam512_set24_entry_value_ptr40	mem_tcam512_set32_entry_value_ptr40
mem_tcam512_set24_entry_mask_ptr32	<p>mem_tcam512_set32_entry_mask</p> <p>Alias for mem_tcam512_set32_entry_mask.</p> <p> Note</p> <p>A 24 bit word TCAM is the same as a 32 bit word TCAM. The difference is seen when lookups are performed. For a 24 bit word TCAM, the full 32 bit word can be set (the highest 8 bits are not masked off). When a 24 bit lookup is performed, the highest 8 bits are not compared, but will be returned for the first match.</p>
mem_tcam512_set24_entry_mask_ptr40	mem_tcam512_set32_entry_mask_ptr40
mem_tcam512_set24_entry_ptr32	<p>mem_tcam512_set32_entry</p> <p>Alias for mem_tcam512_set32_entry.</p> <p> Note</p> <p>A 24 bit word TCAM is the same as a 32 bit word TCAM. The difference is seen when lookups are performed. For a 24 bit word TCAM, the full 32 bit word can be set (the highest 8 bits are not masked off). When a 24 bit lookup is performed, the highest 8 bits are not compared, but will be returned for the first match.</p>
mem_tcam512_set24_entry_ptr40	mem_tcam512_set32_entry_ptr40
mem_tcam_word_to_entry_index24	<p>mem_tcam_word_to_entry_index32</p> <p>Alias for mem_tcam_word_to_entry_index32.</p> <p>mem_tcam128_set24_word</p>

Defined	Definition
mem_tcam_entry_value_to_word_index24	mem_tcam_entry_value_to_word_index32 Alias for mem_tcam_entry_value_to_word_index32. mem_tcam128_set24_word
mem_tcam_entry_mask_to_word_index24	mem_tcam_entry_mask_to_word_index32 Alias for mem_tcam_entry_mask_to_word_index32. mem_tcam128_set24_word

3.22.2 MU TCAM Structs

3.22.2.1 mem_tcam128_t

128 bit TCAM container type.

Table 3.752. struct mem_tcam128_t

Type	Name	Description
unsigned int	value[4]	Storage for the TCAM data.

3.22.2.2 mem_tcam256_t

256 bit TCAM container type.

Table 3.753. struct mem_tcam256_t

Type	Name	Description
unsigned int	value[8]	Storage for the TCAM data.

3.22.2.3 mem_tcam384_t

384 bit TCAM container type.

Table 3.754. struct mem_tcam384_t

Type	Name	Description
unsigned int	value[12]	Storage for the TCAM data.

3.22.2.4 mem_tcam512_t

512 bit TCAM container type.

Table 3.755. struct mem_tcam512_t

Type	Name	Description
unsigned int	value[16]	Storage for the TCAM data.

3.22.2.5 mem_tcam_lookup32_in_t

Input type for 32 bit TCAM lookups.

The search field of this structure needs to be populated before performing a TCAM lookup.

Table 3.756. struct mem_tcam_lookup32_in_t

Type	Name	Description
unsigned int	search	32 bit value to search for in the TCAM

3.22.3 MU TCAM Unions

3.22.3.1 mem_tcam_lookup16_in_t

Input type for 16 bit TCAM lookups.

The search field of this structure needs to be populated before performing a TCAM lookup.

Table 3.757. union mem_tcam_lookup16_in_t

Type	Name	Description
unsigned int	reserved1:16	Reserved.
unsigned int	search:16	16 bit value to search for in the TCAM.
unsigned int	value	Accessor to entire structure.

3.22.3.2 mem_tcam_lookup16_out_t

Output type of 16 bit TCAM lookups.

The match_bit field indicates which TCAM entries matched the lookup.

Table 3.758. union mem_tcam_lookup16_out_t

Type	Name	Description
unsigned int	match_bitf:16	16 bit field with ones indicating that a corresponding entry matched.
unsigned int	reserved:8	Reserved.
unsigned int	first_match:8	First matched 16 bit word index.

Type	Name	Description
		Equal to 0xFF when no match was found
unsigned int	value	Accessor to entire structure.

3.22.3.3 mem_tcam_lookup24_in_t

Input type for 24 bit TCAM lookups.

The search field of this structure needs to be populated before performing a TCAM lookup.

Table 3.759. union mem_tcam_lookup24_in_t

Type	Name	Description
unsigned int	reserved1:8	Reserved.
unsigned int	search:24	24 bit value to search for in the TCAM
unsigned int	value	Accessor to entire structure.

3.22.3.4 mem_tcam_lookup24_out_t

Output type of 24 bit TCAM lookup and add operations.

These match bit fields indicate which TCAM entries matched the lookup.

Table 3.760. union mem_tcam_lookup24_out_t

Type	Name	Description
unsigned int	reserved:8	Reserved.
unsigned int	match_bitf:8	8 bit field with ones indicating that a corresponding entry matched
unsigned int	match_high_byte:8	Upper 8 bits of the matched TCAM entry is returned in this field.
unsigned int	first_match:8	First matched 32 bit word index. Equal to 0xFF when no match was found
unsigned int	value	

3.22.3.5 mem_tcam_lookup32_out_t

Output type of 32 bit TCAM lookup and add operations.

The match bit field indicates which TCAM entries matched the lookup.

Table 3.761. union mem_tcam_lookup32_out_t

Type	Name	Description
unsigned int	reserved:8	Reserved.

Type	Name	Description
unsigned int	match_bitf:8	8 bit field with ones indicating that a corresponding entry matched
unsigned int	match_high_byte:8	Upper 8 bits of the matched TCAM entry is returned in this field.
unsigned int	first_match:8	First matched 32 bit word index. Equal to 0xFF when no match was found
unsigned int	value	

3.22.3.6 mem_tcam_lookup8_in_t

Input type for 8 bit TCAM lookups.

The search field of this structure needs to be populated before performing a TCAM lookup.

Table 3.762. union mem_tcam_lookup8_in_t

Type	Name	Description
unsigned int	reserved1:24	Reserved.
unsigned int	search:8	8 bit value to search for in the TCAM
unsigned int	padding	Additional word to ensure read data is same size as write data.
unsigned int	value[2]	Accessor to entire structure.

3.22.3.7 mem_tcam_lookup8_out_t

Output type of 8 bit TCAM lookups.

The match bit fields indicate which TCAM entries matched the lookup.

Table 3.763. union mem_tcam_lookup8_out_t

Type	Name	Description
unsigned int	reserved:24	Reserved.
unsigned int	first_match:8	First matched 8 bit word index. 0xFF when no match was found
unsigned int	match_bitf	32 bit field with ones indicating that a corresponding entry matched
unsigned int	value[2]	Accessor to entire structure.

3.22.4 MU TCAM Typedefs

3.22.4.1 mem_tcam128_t

128 bit TCAM container type.

Table 3.764. typedef mem_tcam128_t

Type	Definition
mem_tcam128_t	struct mem_tcam128_t

3.22.4.2 mem_tcam256_t

256 bit TCAM container type.

Table 3.765. typedef mem_tcam256_t

Type	Definition
mem_tcam256_t	struct mem_tcam256_t

3.22.4.3 mem_tcam384_t

384 bit TCAM container type.

Table 3.766. typedef mem_tcam384_t

Type	Definition
mem_tcam384_t	struct mem_tcam384_t

3.22.4.4 mem_tcam512_t

512 bit TCAM container type.

Table 3.767. typedef mem_tcam512_t

Type	Definition
mem_tcam512_t	struct mem_tcam512_t

3.22.4.5 mem_tcam128_in_mem_t

Table 3.768. typedef mem_tcam128_in_mem_t

Type	Definition
mem_tcam128_in_mem_t	__addr32 __align16 mem_tcam128_t

3.22.4.6 mem_tcam256_in_mem_t

Table 3.769. typedef mem_tcam256_in_mem_t

Type	Definition
mem_tcam256_in_mem_t	<code>__addr32 __align32 mem_tcam256_t</code>

3.22.4.7 mem_tcam384_in_mem_t

Table 3.770. typedef mem_tcam384_in_mem_t

Type	Definition
mem_tcam384_in_mem_t	<code>__addr32 __align64 mem_tcam384_t</code>

3.22.4.8 mem_tcam512_in_mem_t

Table 3.771. typedef mem_tcam512_in_mem_t

Type	Definition
mem_tcam512_in_mem_t	<code>__addr32 __align64 mem_tcam512_t</code>

3.22.4.9 mem_tcam128_ptr40_t

Table 3.772. typedef mem_tcam128_ptr40_t

Type	Definition
mem_tcam128_ptr40_t	<code>__mem __addr40 __align16 mem_tcam128_t*</code>

3.22.4.10 mem_tcam256_ptr40_t

Table 3.773. typedef mem_tcam256_ptr40_t

Type	Definition
mem_tcam256_ptr40_t	<code>__mem __addr40 __align32 mem_tcam256_t*</code>

3.22.4.11 mem_tcam384_ptr40_t

Table 3.774. typedef mem_tcam384_ptr40_t

Type	Definition
mem_tcam384_ptr40_t	<code>__mem __addr40 __align64 mem_tcam384_t*</code>

3.22.4.12 mem_tcam512_ptr40_t

Table 3.775. typedef mem_tcam512_ptr40_t

Type	Definition
mem_tcam512_ptr40_t	__mem __addr40 __align64 mem_tcam512_t*

3.22.4.13 mem_tcam_lookup8_in_t

Input type for 8 bit TCAM lookups.

The search field of this structure needs to be populated before performing a TCAM lookup.

Table 3.776. typedef mem_tcam_lookup8_in_t

Type	Definition
mem_tcam_lookup8_in_t	union mem_tcam_lookup8_in_t

3.22.4.14 mem_tcam_lookup8_out_t

Output type of 8 bit TCAM lookups.

The match bit fields indicate which TCAM entries matched the lookup.

Table 3.777. typedef mem_tcam_lookup8_out_t

Type	Definition
mem_tcam_lookup8_out_t	union mem_tcam_lookup8_out_t

3.22.4.15 mem_tcam_lookup8_out_in_read_reg_t

Type for mem_tcam_lookup8_out_t in read registers.

Table 3.778. typedef mem_tcam_lookup8_out_in_read_reg_t

Type	Definition
mem_tcam_lookup8_out_in_read_reg_t	__xread mem_tcam_lookup8_out_t

3.22.4.16 mem_tcam_lookup16_in_t

Input type for 16 bit TCAM lookups.

The search field of this structure needs to be populated before performing a TCAM lookup.

Table 3.779. `typedef mem_tcam_lookup16_in_t`

Type	Definition
<code>mem_tcam_lookup16_in_t</code>	<code>union mem_tcam_lookup16_in_t</code>

3.22.4.17 `mem_tcam_lookup16_out_t`

Output type of 16 bit TCAM lookups.

The match bit field indicates which TCAM entries matched the lookup.

Table 3.780. `typedef mem_tcam_lookup16_out_t`

Type	Definition
<code>mem_tcam_lookup16_out_t</code>	<code>union mem_tcam_lookup16_out_t</code>

3.22.4.18 `mem_tcam_lookup16_out_in_read_reg_t`

Type for `mem_tcam_lookup16_out_t` in read registers.

Table 3.781. `typedef mem_tcam_lookup16_out_in_read_reg_t`

Type	Definition
<code>mem_tcam_lookup16_out_in_read_reg_t</code>	<code>__xread mem_tcam_lookup16_out_t</code>

3.22.4.19 `mem_tcam_lookup24_in_t`

Input type for 24 bit TCAM lookups.

The search field of this structure needs to be populated before performing a TCAM lookup.

Table 3.782. `typedef mem_tcam_lookup24_in_t`

Type	Definition
<code>mem_tcam_lookup24_in_t</code>	<code>union mem_tcam_lookup24_in_t</code>

3.22.4.20 `mem_tcam_lookup24_out_t`

Output type of 24 bit TCAM lookup and add operations.

These match bit fields indicate which TCAM entries matched the lookup.

Table 3.783. `typedef mem_tcam_lookup24_out_t`

Type	Definition
<code>mem_tcam_lookup24_out_t</code>	<code>union mem_tcam_lookup24_out_t</code>

3.22.4.21 mem_tcam_lookup24_out_in_read_reg_t

Type for `mem_tcam_lookup24_out_t` in read registers.

Table 3.784. typedef mem_tcam_lookup24_out_in_read_reg_t

Type	Definition
<code>mem_tcam_lookup24_out_in_read_reg_t</code>	<code>__xread mem_tcam_lookup24_out_t</code>

3.22.4.22 mem_tcam_lookup32_in_t

Input type for 32 bit TCAM lookups.

The search field of this structure needs to be populated before performing a TCAM lookup.

Table 3.785. typedef mem_tcam_lookup32_in_t

Type	Definition
<code>mem_tcam_lookup32_in_t</code>	<code>struct mem_tcam_lookup32_in_t</code>

3.22.4.23 mem_tcam_lookup32_out_t

Output type of 32 bit TCAM lookup and add operations.

The match bit field indicates which TCAM entries matched the lookup.

Table 3.786. typedef mem_tcam_lookup32_out_t

Type	Definition
<code>mem_tcam_lookup32_out_t</code>	<code>union mem_tcam_lookup32_out_t</code>

3.22.4.24 mem_tcam_lookup32_out_in_read_reg_t

Type for `mem_tcam_lookup32_out_t` in read registers.

Table 3.787. typedef mem_tcam_lookup32_out_in_read_reg_t

Type	Definition
<code>mem_tcam_lookup32_out_in_read_reg_t</code>	<code>__xread mem_tcam_lookup32_out_t</code>

3.22.5 MU TCAM Functions

3.22.5.1 mem_tcam128_init_ptr32

Prototype:

```
void mem_tcam128_init_ptr32(__mem mem_tcam128_in_mem_t* tcam)
```

Description:

Initialize a 128 bit TCAM in memory.

All words are initialized to 0. This means a TCAM lookup on a newly initialized TCAM will match all entries, because the masks are all 0.

Table 3.788. mem_tcam128_init_ptr32 parameters

Type	Name	Description
<code>__mem mem_tcam128_in_mem_t*</code>	<code>tcam</code>	Pointer to the TCAM in MEM

3.22.5.2 mem_tcam128_init_ptr40

Prototype:

```
void mem_tcam128_init_ptr40(mem_tcam128_ptr40_t tcam)
```

Description:

Initialize a 128 bit TCAM in 40-bit addressed MEM.

All words are initialized to 0. This means a TCAM lookup on a newly initialized TCAM will match all entries, because the masks are all 0.

Table 3.789. mem_tcam128_init_ptr40 parameters

Type	Name	Description
<code>mem_tcam128_ptr40_t</code>	<code>tcam</code>	Pointer to the TCAM in MEM

3.22.5.3 mem_tcam256_init_ptr32

Prototype:

```
void mem_tcam256_init_ptr32(__mem mem_tcam256_in_mem_t* tcam)
```

Description:

Initialize a 256 bit TCAM in memory.

All words are initialized to 0. This means a TCAM lookup on a newly initialized TCAM will match all entries, because the masks are all 0.

Table 3.790. mem_tcam256_init_ptr32 parameters

Type	Name	Description
<code>__mem mem_tcam256_in_mem_t*</code>	<code>tcam</code>	Pointer to the TCAM in MEM

3.22.5.4 mem_tcam256_init_ptr40

Prototype:

```
void mem_tcam256_init_ptr40(mem_tcam256_ptr40_t tcam)
```

Description:

Initialize a 256 bit TCAM in 40-bit addressed MEM.

All words are initialized to 0. This means a TCAM lookup on a newly initialized TCAM will match all entries, because the masks are all 0.

Table 3.791. mem_tcam256_init_ptr40 parameters

Type	Name	Description
<code>mem_tcam256_ptr40_t</code>	<code>tcam</code>	Pointer to the TCAM in MEM

3.22.5.5 mem_tcam384_init_ptr32

Prototype:

```
void mem_tcam384_init_ptr32(__mem mem_tcam384_in_mem_t* tcam)
```

Description:

Initialize a 384 bit TCAM in memory.

All words are initialized to 0. This means a TCAM lookup on a newly initialized TCAM will match all entries, because the masks are all 0.

Table 3.792. mem_tcam384_init_ptr32 parameters

Type	Name	Description
<code>__mem mem_tcam384_in_mem_t*</code>	<code>tcam</code>	Pointer to the TCAM in MEM

3.22.5.6 mem_tcam384_init_ptr40

Prototype:

```
void mem_tcam384_init_ptr40(mem_tcam384_ptr40_t tcam)
```

Description:

Initialize a 384 bit TCAM in 40-bit addressed MEM.

All words are initialized to 0. This means a TCAM lookup on a newly initialized TCAM will match all entries, because the masks are all 0.

Table 3.793. mem_tcam384_init_ptr40 parameters

Type	Name	Description
mem_tcam384_ptr40_t	<i>tcam</i>	Pointer to the TCAM in MEM

3.22.5.7 mem_tcam512_init_ptr32

Prototype:

```
void mem_tcam512_init_ptr32(__mem mem_tcam512_in_mem_t* tcam)
```

Description:

Initialize a 512 bit TCAM in memory.

All words are initialized to 0. This means a TCAM lookup on a newly initialized TCAM will match all entries, because the masks are all 0.

Table 3.794. mem_tcam512_init_ptr32 parameters

Type	Name	Description
<u>__mem</u> mem_tcam512_in_mem_t*	<i>tcam</i>	Pointer to the TCAM in MEM

3.22.5.8 mem_tcam512_init_ptr40

Prototype:

```
void mem_tcam512_init_ptr40(mem_tcam512_ptr40_t tcam)
```

Description:

Initialize a 512 bit TCAM in 40-bit addressed MEM.

All words are initialized to 0. This means a TCAM lookup on a newly initialized TCAM will match all entries, because the masks are all 0.

Table 3.795. mem_tcam512_init_ptr40 parameters

Type	Name	Description
mem_tcam512_ptr40_t	<i>tcam</i>	Pointer to the TCAM in MEM

3.22.5.9 mem_tcam128_set8_word_ptr32

Prototype:

```
void mem_tcam128_set8_word_ptr32(__mem mem_tcam128_in_mem_t* tcam, unsigned int index,
unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set an 8 bit word in a 128 bit TCAM.

Note

The maximum index is 15.

Table 3.796. mem_tcam128_set8_word_ptr32 parameters

Type	Name	Description
__mem mem_tcam128_in_mem_t*	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>index</i>	TCAM word to set (0 is the first entry value, 3 is the first entry mask, 1 is the second entry value)
unsigned int	<i>value</i>	Value to set the TCAM word to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.10 mem_tcam128_set8_word_ptr40

Prototype:

```
void mem_tcam128_set8_word_ptr40(mem_tcam128_ptr40_t tcam, unsigned int index, unsigned
int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set an 8 bit word in a 128 bit TCAM in 40-bit addressed MEM.

Note

The maximum index is 15.

Table 3.797. mem_tcam128_set8_word_ptr40 parameters

Type	Name	Description
mem_tcam128_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>index</i>	TCAM word to set (0 is the first entry value, 3 is the first entry mask, 1 is the second entry value)
unsigned int	<i>value</i>	Value to set the TCAM word to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.11 mem_tcam128_set8_entry_value_ptr32

Prototype:

```
void mem_tcam128_set8_entry_value_ptr32(__mem mem_tcam128_in_mem_t* tcam, unsigned int
entry_index, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set an 8 bit entry value in a 128 bit TCAM.



Note

The maximum index is 7.

Table 3.798. mem_tcam128_set8_entry_value_ptr32 parameters

Type	Name	Description
__mem mem_tcam128_in_mem_t*	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
unsigned int	<i>value</i>	Value to set the TCAM entry value to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.12 mem_tcam128_set8_entry_value_ptr40

Prototype:

```
void mem_tcam128_set8_entry_value_ptr40(mem_tcam128_ptr40_t tcam, unsigned int entry_index,
unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set an 8 bit entry value in a 128 bit TCAM in 40-bit addressed MEM.



Note

The maximum index is 7.

Table 3.799. mem_tcam128_set8_entry_value_ptr40 parameters

Type	Name	Description
mem_tcam128_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
unsigned int	<i>value</i>	Value to set the TCAM entry value to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.13 mem_tcam128_set8_entry_mask_ptr32

Prototype:

```
void mem_tcam128_set8_entry_mask_ptr32(__mem mem_tcam128_in_mem_t* tcam, unsigned int entry_index, unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set an 8 bit entry mask in a 128 bit TCAM.



Note

The maximum index is 7.

Table 3.800. mem_tcam128_set8_entry_mask_ptr32 parameters

Type	Name	Description
__mem mem_tcam128_in_mem_t*	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
unsigned int	<i>mask</i>	Value to set the TCAM entry mask to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.14 mem_tcam128_set8_entry_mask_ptr40

Prototype:

```
void mem_tcam128_set8_entry_mask_ptr40(mem_tcam128_ptr40_t tcam, unsigned int entry_index,  
unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set an 8 bit entry mask in a 128 bit TCAM in 40-bit addressed MEM.

Note

The maximum index is 7.

Table 3.801. mem_tcam128_set8_entry_mask_ptr40 parameters

Type	Name	Description
mem_tcam128_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
unsigned int	<i>mask</i>	Value to set the TCAM entry mask to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.15 mem_tcam128_set8_entry_ptr32

Prototype:

```
void mem_tcam128_set8_entry_ptr32(__mem mem_tcam128_in_mem_t* tcam, unsigned int  
entry_index, unsigned int value, unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set an 8 bit entry value and mask in a 128 bit TCAM.

Note

The maximum index is 7.

Table 3.802. mem_tcam128_set8_entry_ptr32 parameters

Type	Name	Description
__mem mem_tcam128_in_mem_t*	<i>tcam</i>	Pointer to the TCAM structure in MEM

Type	Name	Description
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
unsigned int	<i>value</i>	Value to set the TCAM entry value to
unsigned int	<i>mask</i>	Value to set the TCAM entry mask to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.16 mem_tcam128_set8_entry_ptr40

Prototype:

```
void mem_tcam128_set8_entry_ptr40(mem_tcam128_ptr40_t tcam, unsigned int entry_index,
unsigned int value, unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set an 8 bit entry value and mask in a 128 bit TCAM in 40-bit addressed MEM.



Note

The maximum index is 7.

Table 3.803. mem_tcam128_set8_entry_ptr40 parameters

Type	Name	Description
mem_tcam128_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
unsigned int	<i>value</i>	Value to set the TCAM entry value to
unsigned int	<i>mask</i>	Value to set the TCAM entry mask to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.17 mem_tcam256_set8_word_ptr32

Prototype:

```
void mem_tcam256_set8_word_ptr32(__mem mem_tcam256_in_mem_t* tcam, unsigned int index,
unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set an 8 bit word in a 256 bit TCAM.



Note

The maximum index is 31.

Table 3.804. mem_tcam256_set8_word_ptr32 parameters

Type	Name	Description
<code>__mem mem_tcam256_in_mem_t*</code>	<code>tcam</code>	Pointer to the TCAM structure in MEM
<code>unsigned int</code>	<code>index</code>	TCAM word to set (0 is the first entry value, 3 is the first entry mask, 1 is the second entry value)
<code>unsigned int</code>	<code>value</code>	Value to set the TCAM word to
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.22.5.18 mem_tcam256_set8_word_ptr40

Prototype:

```
void mem_tcam256_set8_word_ptr40(mem_tcam256_ptr40_t tcam, unsigned int index, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set an 8 bit word in a 256 bit TCAM in 40-bit addressed MEM.



Note

The maximum index is 31.

Table 3.805. mem_tcam256_set8_word_ptr40 parameters

Type	Name	Description
<code>mem_tcam256_ptr40_t</code>	<code>tcam</code>	Pointer to the TCAM structure in MEM
<code>unsigned int</code>	<code>index</code>	TCAM word to set (0 is the first entry value, 3 is the first entry mask, 1 is the second entry value)
<code>unsigned int</code>	<code>value</code>	Value to set the TCAM word to
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.22.5.19 mem_tcam256_set8_entry_value_ptr32

Prototype:

```
void mem_tcam256_set8_entry_value_ptr32(__mem mem_tcam256_in_mem_t* tcam, unsigned int entry_index, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set an 8 bit entry value in a 256 bit TCAM.



Note

The maximum index is 15.

Table 3.806. mem_tcam256_set8_entry_value_ptr32 parameters

Type	Name	Description
<code>__mem mem_tcam256_in_mem_t*</code>	<code>tcam</code>	Pointer to the TCAM structure in MEM
<code>unsigned int</code>	<code>entry_index</code>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
<code>unsigned int</code>	<code>value</code>	Value to set the TCAM entry value to
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.22.5.20 mem_tcam256_set8_entry_value_ptr40

Prototype:

```
void mem_tcam256_set8_entry_value_ptr40(mem_tcam256_ptr40_t tcam, unsigned int entry_index, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set an 8 bit entry value in a 256 bit TCAM in 40-bit addressed MEM.



Note

The maximum index is 15.

Table 3.807. mem_tcam256_set8_entry_value_ptr40 parameters

Type	Name	Description
<code>mem_tcam256_ptr40_t</code>	<code>tcam</code>	Pointer to the TCAM structure in MEM
<code>unsigned int</code>	<code>entry_index</code>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
<code>unsigned int</code>	<code>value</code>	Value to set the TCAM entry value to
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)

Type	Name	Description
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.21 mem_tcam256_set8_entry_mask_ptr32

Prototype:

```
void mem_tcam256_set8_entry_mask_ptr32(__mem mem_tcam256_in_mem_t* tcam, unsigned int entry_index, unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set an 8 bit entry mask in a 256 bit TCAM.



Note

The maximum index is 15.

Table 3.808. mem_tcam256_set8_entry_mask_ptr32 parameters

Type	Name	Description
<u>__mem</u> mem_tcam256_in_mem_t*	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
unsigned int	<i>mask</i>	Value to set the TCAM entry mask to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.22 mem_tcam256_set8_entry_mask_ptr40

Prototype:

```
void mem_tcam256_set8_entry_mask_ptr40(mem_tcam256_ptr40_t tcam, unsigned int entry_index, unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set an 8 bit entry mask in a 256 bit TCAM in 40-bit addressed MEM.



Note

The maximum index is 15.

Table 3.809. mem_tcam256_set8_entry_mask_ptr40 parameters

Type	Name	Description
mem_tcam256_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
unsigned int	<i>mask</i>	Value to set the TCAM entry mask to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.23 mem_tcam256_set8_entry_ptr32

Prototype:

```
void mem_tcam256_set8_entry_ptr32(__mem mem_tcam256_in_mem_t* tcam, unsigned int
entry_index, unsigned int value, unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set an 8 bit entry value and mask in a 256 bit TCAM.



Note

The maximum index is 15.

Table 3.810. mem_tcam256_set8_entry_ptr32 parameters

Type	Name	Description
__mem mem_tcam256_in_mem_t*	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
unsigned int	<i>value</i>	Value to set the TCAM entry value to
unsigned int	<i>mask</i>	Value to set the TCAM entry mask to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.24 mem_tcam256_set8_entry_ptr40

Prototype:

```
void mem_tcam256_set8_entry_ptr40(mem_tcam256_ptr40_t tcam, unsigned int entry_index,
unsigned int value, unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set an 8 bit entry value and mask in a 256 bit TCAM in 40-bit addressed MEM.



Note

The maximum index is 15.

Table 3.811. mem_tcam256_set8_entry_ptr40 parameters

Type	Name	Description
mem_tcam256_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
unsigned int	<i>value</i>	Value to set the TCAM entry value to
unsigned int	<i>mask</i>	Value to set the TCAM entry mask to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.25 mem_tcam384_set8_word_ptr32

Prototype:

```
void mem_tcam384_set8_word_ptr32(__mem mem_tcam384_in_mem_t* tcam, unsigned int index,
                                  unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set an 8 bit word in a 384 bit TCAM.



Note

The maximum index is 47.

Table 3.812. mem_tcam384_set8_word_ptr32 parameters

Type	Name	Description
__mem mem_tcam384_in_mem_t*	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>index</i>	TCAM word to set (0 is the first entry value, 3 is the first entry mask, 1 is the second entry value)
unsigned int	<i>value</i>	Value to set the TCAM word to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.26 mem_tcam384_set8_word_ptr40

Prototype:

```
void mem_tcam384_set8_word_ptr40(mem_tcam384_ptr40_t tcam, unsigned int index, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set an 8 bit word in a 384 bit TCAM in 40-bit addressed MEM.

Note

The maximum index is 47.

Table 3.813. mem_tcam384_set8_word_ptr40 parameters

Type	Name	Description
mem_tcam384_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>index</i>	TCAM word to set (0 is the first entry value, 3 is the first entry mask, 1 is the second entry value)
unsigned int	<i>value</i>	Value to set the TCAM word to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.27 mem_tcam384_set8_entry_value_ptr32

Prototype:

```
void mem_tcam384_set8_entry_value_ptr32(__mem mem_tcam384_in_mem_t* tcam, unsigned int entry_index, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set an 8 bit entry value in a 384 bit TCAM.

Note

The maximum index is 23.

Table 3.814. mem_tcam384_set8_entry_value_ptr32 parameters

Type	Name	Description
__mem mem_tcam384_in_mem_t*	<i>tcam</i>	Pointer to the TCAM structure in MEM

Type	Name	Description
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
unsigned int	<i>value</i>	Value to set the TCAM entry value to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.28 mem_tcam384_set8_entry_value_ptr40

Prototype:

```
void mem_tcam384_set8_entry_value_ptr40(mem_tcam384_ptr40_t tcam, unsigned int entry_index,
                                         unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set an 8 bit entry value in a 384 bit TCAM in 40-bit addressed MEM.



Note

The maximum index is 23.

Table 3.815. mem_tcam384_set8_entry_value_ptr40 parameters

Type	Name	Description
mem_tcam384_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
unsigned int	<i>value</i>	Value to set the TCAM entry value to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.29 mem_tcam384_set8_entry_mask_ptr32

Prototype:

```
void mem_tcam384_set8_entry_mask_ptr32(__mem mem_tcam384_in_mem_t* tcam, unsigned int entry_index,
                                         unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set an 8 bit entry mask in a 384 bit TCAM.



Note

The maximum index is 23.

Table 3.816. mem_tcam384_set8_entry_mask_ptr32 parameters

Type	Name	Description
<code>__mem mem_tcam384_in_mem_t*</code>	<code>tcam</code>	Pointer to the TCAM structure in MEM
<code>unsigned int</code>	<code>entry_index</code>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
<code>unsigned int</code>	<code>mask</code>	Value to set the TCAM entry mask to
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.22.5.30 mem_tcam384_set8_entry_mask_ptr40

Prototype:

```
void mem_tcam384_set8_entry_mask_ptr40(mem_tcam384_ptr40_t tcam, unsigned int entry_index,
unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set an 8 bit entry mask in a 384 bit TCAM in 40-bit addressed MEM.



Note

The maximum index is 23.

Table 3.817. mem_tcam384_set8_entry_mask_ptr40 parameters

Type	Name	Description
<code>mem_tcam384_ptr40_t</code>	<code>tcam</code>	Pointer to the TCAM structure in MEM
<code>unsigned int</code>	<code>entry_index</code>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
<code>unsigned int</code>	<code>mask</code>	Value to set the TCAM entry mask to
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.22.5.31 mem_tcam384_set8_entry_ptr32

Prototype:

```
void mem_tcam384_set8_entry_ptr32(__mem mem_tcam384_in_mem_t* tcam, unsigned int entry_index, unsigned int value, unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set an 8 bit entry value and mask in a 384 bit TCAM.



Note

The maximum index is 23.

Table 3.818. mem_tcam384_set8_entry_ptr32 parameters

Type	Name	Description
<code>__mem mem_tcam384_in_mem_t*</code>	<code>tcam</code>	Pointer to the TCAM structure in MEM
<code>unsigned int</code>	<code>entry_index</code>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
<code>unsigned int</code>	<code>value</code>	Value to set the TCAM entry value to
<code>unsigned int</code>	<code>mask</code>	Value to set the TCAM entry mask to
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.22.5.32 mem_tcam384_set8_entry_ptr40

Prototype:

```
void mem_tcam384_set8_entry_ptr40(mem_tcam384_ptr40_t tcam, unsigned int entry_index, unsigned int value, unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set an 8 bit entry value and mask in a 384 bit TCAM in 40-bit addressed MEM.



Note

The maximum index is 23.

Table 3.819. mem_tcam384_set8_entry_ptr40 parameters

Type	Name	Description
<code>mem_tcam384_ptr40_t</code>	<code>tcam</code>	Pointer to the TCAM structure in MEM
<code>unsigned int</code>	<code>entry_index</code>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
<code>unsigned int</code>	<code>value</code>	Value to set the TCAM entry value to

Type	Name	Description
unsigned int	<i>mask</i>	Value to set the TCAM entry mask to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.33 mem_tcam512_set8_word_ptr32

Prototype:

```
void mem_tcam512_set8_word_ptr32(__mem mem_tcam512_in_mem_t* tcam, unsigned int index,
unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set an 8 bit word in a 512 bit TCAM.



Note

The maximum index is 63.

Table 3.820. mem_tcam512_set8_word_ptr32 parameters

Type	Name	Description
__mem mem_tcam512_in_mem_t*	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>index</i>	TCAM word to set (0 is the first entry value, 3 is the first entry mask, 1 is the second entry value)
unsigned int	<i>value</i>	Value to set the TCAM word to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.34 mem_tcam512_set8_word_ptr40

Prototype:

```
void mem_tcam512_set8_word_ptr40(mem_tcam512_ptr40_t tcam, unsigned int index, unsigned
int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set an 8 bit word in a 512 bit TCAM in 40-bit addressed MEM.



Note

The maximum index is 63.

Table 3.821. mem_tcam512_set8_word_ptr40 parameters

Type	Name	Description
mem_tcam512_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>index</i>	TCAM word to set (0 is the first entry value, 3 is the first entry mask, 1 is the second entry value)
unsigned int	<i>value</i>	Value to set the TCAM word to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.35 mem_tcam512_set8_entry_value_ptr32

Prototype:

```
void mem_tcam512_set8_entry_value_ptr32(__mem mem_tcam512_in_mem_t* tcam, unsigned int
entry_index, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set an 8 bit entry value in a 512 bit TCAM.



Note

The maximum index is 31.

Table 3.822. mem_tcam512_set8_entry_value_ptr32 parameters

Type	Name	Description
__mem mem_tcam512_in_mem_t*	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
unsigned int	<i>value</i>	Value to set the TCAM entry value to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.36 mem_tcam512_set8_entry_value_ptr40

Prototype:

```
void mem_tcam512_set8_entry_value_ptr40(mem_tcam512_ptr40_t tcam, unsigned int entry_index,
unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set an 8 bit entry value in a 512 bit TCAM in 40-bit addressed MEM.



Note

The maximum index is 31.

Table 3.823. mem_tcam512_set8_entry_value_ptr40 parameters

Type	Name	Description
mem_tcam512_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
unsigned int	<i>value</i>	Value to set the TCAM entry value to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.37 mem_tcam512_set8_entry_mask_ptr32

Prototype:

```
void mem_tcam512_set8_entry_mask_ptr32(__mem mem_tcam512_in_mem_t* tcam, unsigned int
entry_index, unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set an 8 bit entry mask in a 512 bit TCAM.



Note

The maximum index is 31.

Table 3.824. mem_tcam512_set8_entry_mask_ptr32 parameters

Type	Name	Description
__mem mem_tcam512_in_mem_t*	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
unsigned int	<i>mask</i>	Value to set the TCAM entry mask to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)

Type	Name	Description
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.38 mem_tcam512_set8_entry_mask_ptr40

Prototype:

```
void mem_tcam512_set8_entry_mask_ptr40(mem_tcam512_ptr40_t tcam, unsigned int entry_index,
unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set an 8 bit entry mask in a 512 bit TCAM in 40-bit addressed MEM.

Note

The maximum index is 31.

Table 3.825. mem_tcam512_set8_entry_mask_ptr40 parameters

Type	Name	Description
mem_tcam512_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
unsigned int	<i>mask</i>	Value to set the TCAM entry mask to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.39 mem_tcam512_set8_entry_ptr32

Prototype:

```
void mem_tcam512_set8_entry_ptr32(__mem mem_tcam512_in_mem_t* tcam, unsigned int entry_index,
unsigned int value, unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set an 8 bit entry value and mask in a 512 bit TCAM.

Note

The maximum index is 31.

Table 3.826. mem_tcam512_set8_entry_ptr32 parameters

Type	Name	Description
<code>__mem mem_tcam512_in_mem_t*</code>	<code>tcam</code>	Pointer to the TCAM structure in MEM
<code>unsigned int</code>	<code>entry_index</code>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
<code>unsigned int</code>	<code>value</code>	Value to set the TCAM entry value to
<code>unsigned int</code>	<code>mask</code>	Value to set the TCAM entry mask to
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.22.5.40 mem_tcam512_set8_entry_ptr40

Prototype:

```
void mem_tcam512_set8_entry_ptr40(mem_tcam512_ptr40_t tcam, unsigned int entry_index,
unsigned int value, unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set an 8 bit entry value and mask in a 512 bit TCAM in 40-bit addressed MEM.



Note

The maximum index is 31.

Table 3.827. mem_tcam512_set8_entry_ptr40 parameters

Type	Name	Description
<code>mem_tcam512_ptr40_t</code>	<code>tcam</code>	Pointer to the TCAM structure in MEM
<code>unsigned int</code>	<code>entry_index</code>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
<code>unsigned int</code>	<code>value</code>	Value to set the TCAM entry value to
<code>unsigned int</code>	<code>mask</code>	Value to set the TCAM entry mask to
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.22.5.41 mem_tcam128_set16_word_ptr32

Prototype:

```
void mem_tcam128_set16_word_ptr32(__mem mem_tcam128_in_mem_t* tcam, unsigned int index,
unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 16 bit word in a 128 bit TCAM.



Note

The maximum index is 7.

Table 3.828. mem_tcam128_set16_word_ptr32 parameters

Type	Name	Description
<code>__mem mem_tcam128_in_mem_t*</code>	<code>tcam</code>	Pointer to the TCAM structure in MEM
<code>unsigned int</code>	<code>index</code>	TCAM word to set (0 is the first entry value, 2 is the first entry mask, 1 is the second entry value)
<code>unsigned int</code>	<code>value</code>	Value to set the TCAM word to
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.22.5.42 mem_tcam128_set16_word_ptr40

Prototype:

```
void mem_tcam128_set16_word_ptr40(mem_tcam128_ptr40_t tcam, unsigned int index, unsigned
int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 16 bit word in a 128 bit TCAM in 40-bit addressed MEM.



Note

The maximum index is 7.

Table 3.829. mem_tcam128_set16_word_ptr40 parameters

Type	Name	Description
<code>mem_tcam128_ptr40_t</code>	<code>tcam</code>	Pointer to the TCAM structure in MEM
<code>unsigned int</code>	<code>index</code>	TCAM word to set (0 is the first entry value, 2 is the first entry mask, 1 is the second entry value)
<code>unsigned int</code>	<code>value</code>	Value to set the TCAM word to
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.22.5.43 mem_tcam128_set16_entry_value_ptr32

Prototype:

```
void mem_tcam128_set16_entry_value_ptr32(__mem mem_tcam128_in_mem_t* tcam, unsigned int entry_index, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 16 bit entry value in a 128 bit TCAM.

Note

The maximum index is 3.

Table 3.830. mem_tcam128_set16_entry_value_ptr32 parameters

Type	Name	Description
__mem mem_tcam128_in_mem_t*	tcam	Pointer to the TCAM structure in MEM
unsigned int	entry_index	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
unsigned int	value	Value to set the TCAM entry value to
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

3.22.5.44 mem_tcam128_set16_entry_value_ptr40

Prototype:

```
void mem_tcam128_set16_entry_value_ptr40(mem_tcam128_ptr40_t tcam, unsigned int entry_index, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 16 bit entry value in a 128 bit TCAM in 40-bit addressed MEM.

Note

The maximum index is 3.

Table 3.831. mem_tcam128_set16_entry_value_ptr40 parameters

Type	Name	Description
mem_tcam128_ptr40_t	tcam	Pointer to the TCAM structure in MEM

Type	Name	Description
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
unsigned int	<i>value</i>	Value to set the TCAM entry value to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.45 mem_tcam128_set16_entry_mask_ptr32

Prototype:

```
void mem_tcam128_set16_entry_mask_ptr32(__mem mem_tcam128_in_mem_t* tcam, unsigned int entry_index, unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 16 bit entry mask in a 128 bit TCAM.



Note

The maximum index is 3.

Table 3.832. mem_tcam128_set16_entry_mask_ptr32 parameters

Type	Name	Description
__mem mem_tcam128_in_mem_t*	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
unsigned int	<i>mask</i>	Value to set the TCAM entry mask to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.46 mem_tcam128_set16_entry_mask_ptr40

Prototype:

```
void mem_tcam128_set16_entry_mask_ptr40(mem_tcam128_ptr40_t tcam, unsigned int entry_index, unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 16 bit entry mask in a 128 bit TCAM in 40-bit addressed MEM.



Note

The maximum index is 3.

Table 3.833. mem_tcam128_set16_entry_mask_ptr40 parameters

Type	Name	Description
mem_tcam128_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
unsigned int	<i>mask</i>	Value to set the TCAM entry mask to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.47 mem_tcam128_set16_entry_ptr32

Prototype:

```
void mem_tcam128_set16_entry_ptr32(__mem mem_tcam128_in_mem_t* tcam, unsigned int
entry_index, unsigned int value, unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 16 bit entry value and mask in a 128 bit TCAM.



Note

The maximum index is 3.

Table 3.834. mem_tcam128_set16_entry_ptr32 parameters

Type	Name	Description
__mem mem_tcam128_in_mem_t*	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
unsigned int	<i>value</i>	Value to set the TCAM entry value to
unsigned int	<i>mask</i>	Value to set the TCAM entry mask to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.48 mem_tcam128_set16_entry_ptr40

Prototype:

```
void mem_tcam128_set16_entry_ptr40(mem_tcam128_ptr40_t tcam, unsigned int entry_index,  
unsigned int value, unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 16 bit entry value and mask in a 128 bit TCAM in 40-bit addressed MEM.

Note

The maximum index is 3.

Table 3.835. mem_tcam128_set16_entry_ptr40 parameters

Type	Name	Description
mem_tcam128_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
unsigned int	<i>value</i>	Value to set the TCAM entry value to
unsigned int	<i>mask</i>	Value to set the TCAM entry mask to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.49 mem_tcam256_set16_word_ptr32

Prototype:

```
void mem_tcam256_set16_word_ptr32(__mem mem_tcam256_in_mem_t* tcam, unsigned int index,  
unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 16 bit word in a 256 bit TCAM.

Note

The maximum index is 15.

Table 3.836. mem_tcam256_set16_word_ptr32 parameters

Type	Name	Description
<code>__mem mem_tcam256_in_mem_t*</code>	<code>tcam</code>	Pointer to the TCAM structure in MEM
<code>unsigned int</code>	<code>index</code>	TCAM word to set (0 is the first entry value, 2 is the first entry mask, 1 is the second entry value)
<code>unsigned int</code>	<code>value</code>	Value to set the TCAM word to
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.22.5.50 mem_tcam256_set16_word_ptr40

Prototype:

```
void mem_tcam256_set16_word_ptr40(mem_tcam256_ptr40_t tcam, unsigned int index, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 16 bit word in a 256 bit TCAM in 40-bit addressed MEM.



Note

The maximum index is 15.

Table 3.837. mem_tcam256_set16_word_ptr40 parameters

Type	Name	Description
<code>mem_tcam256_ptr40_t</code>	<code>tcam</code>	Pointer to the TCAM structure in MEM
<code>unsigned int</code>	<code>index</code>	TCAM word to set (0 is the first entry value, 2 is the first entry mask, 1 is the second entry value)
<code>unsigned int</code>	<code>value</code>	Value to set the TCAM word to
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.22.5.51 mem_tcam256_set16_entry_value_ptr32

Prototype:

```
void mem_tcam256_set16_entry_value_ptr32(__mem mem_tcam256_in_mem_t* tcam, unsigned int entry_index, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 16 bit entry value in a 256 bit TCAM.



Note

The maximum index is 7.

Table 3.838. mem_tcam256_set16_entry_value_ptr32 parameters

Type	Name	Description
__mem mem_tcam256_in_mem_t*	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
unsigned int	<i>value</i>	Value to set the TCAM entry value to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.52 mem_tcam256_set16_entry_value_ptr40

Prototype:

```
void mem_tcam256_set16_entry_value_ptr40(mem_tcam256_ptr40_t tcam, unsigned int
entry_index, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 16 bit entry value in a 256 bit TCAM in 40-bit addressed MEM.



Note

The maximum index is 7.

Table 3.839. mem_tcam256_set16_entry_value_ptr40 parameters

Type	Name	Description
mem_tcam256_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
unsigned int	<i>value</i>	Value to set the TCAM entry value to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.53 mem_tcam256_set16_entry_mask_ptr32

Prototype:

```
void mem_tcam256_set16_entry_mask_ptr32(__mem mem_tcam256_in_mem_t* tcam, unsigned int
entry_index, unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 16 bit entry mask in a 256 bit TCAM.

Note

The maximum index is 7.

Table 3.840. mem_tcam256_set16_entry_mask_ptr32 parameters

Type	Name	Description
<code>__mem mem_tcam256_in_mem_t*</code>	<code>tcam</code>	Pointer to the TCAM structure in MEM
<code>unsigned int</code>	<code>entry_index</code>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
<code>unsigned int</code>	<code>mask</code>	Value to set the TCAM entry mask to
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.22.5.54 mem_tcam256_set16_entry_mask_ptr40

Prototype:

```
void mem_tcam256_set16_entry_mask_ptr40(mem_tcam256_ptr40_t tcam, unsigned int entry_index,
unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 16 bit entry mask in a 256 bit TCAM in 40-bit addressed MEM.

Note

The maximum index is 7.

Table 3.841. mem_tcam256_set16_entry_mask_ptr40 parameters

Type	Name	Description
<code>mem_tcam256_ptr40_t</code>	<code>tcam</code>	Pointer to the TCAM structure in MEM

Type	Name	Description
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
unsigned int	<i>mask</i>	Value to set the TCAM entry mask to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.55 mem_tcam256_set16_entry_ptr32

Prototype:

```
void mem_tcam256_set16_entry_ptr32(__mem mem_tcam256_in_mem_t* tcam, unsigned int entry_index, unsigned int value, unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 16 bit entry value and mask in a 256 bit TCAM.



Note

The maximum index is 7.

Table 3.842. mem_tcam256_set16_entry_ptr32 parameters

Type	Name	Description
__mem mem_tcam256_in_mem_t*	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
unsigned int	<i>value</i>	Value to set the TCAM entry value to
unsigned int	<i>mask</i>	Value to set the TCAM entry mask to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.56 mem_tcam256_set16_entry_ptr40

Prototype:

```
void mem_tcam256_set16_entry_ptr40(mem_tcam256_ptr40_t tcam, unsigned int entry_index, unsigned int value, unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 16 bit entry value and mask in a 256 bit TCAM in 40-bit addressed MEM.



Note

The maximum index is 7.

Table 3.843. mem_tcam256_set16_entry_ptr40 parameters

Type	Name	Description
mem_tcam256_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
unsigned int	<i>value</i>	Value to set the TCAM entry value to
unsigned int	<i>mask</i>	Value to set the TCAM entry mask to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.57 mem_tcam384_set16_word_ptr32

Prototype:

```
void mem_tcam384_set16_word_ptr32(__mem mem_tcam384_in_mem_t* tcam, unsigned int index,
unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 16 bit word in a 384 bit TCAM.



Note

The maximum index is 23.

Table 3.844. mem_tcam384_set16_word_ptr32 parameters

Type	Name	Description
__mem mem_tcam384_in_mem_t*	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>index</i>	TCAM word to set (0 is the first entry value, 2 is the first entry mask, 1 is the second entry value)
unsigned int	<i>value</i>	Value to set the TCAM word to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.58 mem_tcam384_set16_word_ptr40

Prototype:

```
void mem_tcam384_set16_word_ptr40(mem_tcam384_ptr40_t tcam, unsigned int index, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 16 bit word in a 384 bit TCAM in 40-bit addressed MEM.

Note

The maximum index is 23.

Table 3.845. mem_tcam384_set16_word_ptr40 parameters

Type	Name	Description
mem_tcam384_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>index</i>	TCAM word to set (0 is the first entry value, 2 is the first entry mask, 1 is the second entry value)
unsigned int	<i>value</i>	Value to set the TCAM word to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.59 mem_tcam384_set16_entry_value_ptr32

Prototype:

```
void mem_tcam384_set16_entry_value_ptr32(__mem mem_tcam384_in_mem_t* tcam, unsigned int entry_index, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 16 bit entry value in a 384 bit TCAM.

Note

The maximum index is 11.

Table 3.846. mem_tcam384_set16_entry_value_ptr32 parameters

Type	Name	Description
__mem mem_tcam384_in_mem_t*	<i>tcam</i>	Pointer to the TCAM structure in MEM

Type	Name	Description
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
unsigned int	<i>value</i>	Value to set the TCAM entry value to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.60 mem_tcam384_set16_entry_value_ptr40

Prototype:

```
void mem_tcam384_set16_entry_value_ptr40(mem_tcam384_ptr40_t tcam, unsigned int
entry_index, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 16 bit entry value in a 384 bit TCAM in 40-bit addressed MEM.



Note

The maximum index is 11.

Table 3.847. mem_tcam384_set16_entry_value_ptr40 parameters

Type	Name	Description
mem_tcam384_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
unsigned int	<i>value</i>	Value to set the TCAM entry value to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.61 mem_tcam384_set16_entry_mask_ptr32

Prototype:

```
void mem_tcam384_set16_entry_mask_ptr32(__mem mem_tcam384_in_mem_t* tcam, unsigned int
entry_index, unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 16 bit entry mask in a 384 bit TCAM.



Note

The maximum index is 11.

Table 3.848. mem_tcam384_set16_entry_mask_ptr32 parameters

Type	Name	Description
<code>__mem mem_tcam384_in_mem_t*</code>	<code>tcam</code>	Pointer to the TCAM structure in MEM
<code>unsigned int</code>	<code>entry_index</code>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
<code>unsigned int</code>	<code>mask</code>	Value to set the TCAM entry mask to
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.22.5.62 mem_tcam384_set16_entry_mask_ptr40

Prototype:

```
void mem_tcam384_set16_entry_mask_ptr40(mem_tcam384_ptr40_t tcam, unsigned int entry_index,
                                         unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 16 bit entry mask in a 384 bit TCAM in 40-bit addressed MEM.



Note

The maximum index is 11.

Table 3.849. mem_tcam384_set16_entry_mask_ptr40 parameters

Type	Name	Description
<code>mem_tcam384_ptr40_t</code>	<code>tcam</code>	Pointer to the TCAM structure in MEM
<code>unsigned int</code>	<code>entry_index</code>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
<code>unsigned int</code>	<code>mask</code>	Value to set the TCAM entry mask to
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.22.5.63 mem_tcam384_set16_entry_ptr32

Prototype:

```
void mem_tcam384_set16_entry_ptr32(__mem mem_tcam384_in_mem_t* tcam, unsigned int entry_index, unsigned int value, unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 16 bit entry value and mask in a 384 bit TCAM.



Note

The maximum index is 11.

Table 3.850. mem_tcam384_set16_entry_ptr32 parameters

Type	Name	Description
<code>__mem mem_tcam384_in_mem_t*</code>	<code>tcam</code>	Pointer to the TCAM structure in MEM
<code>unsigned int</code>	<code>entry_index</code>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
<code>unsigned int</code>	<code>value</code>	Value to set the TCAM entry value to
<code>unsigned int</code>	<code>mask</code>	Value to set the TCAM entry mask to
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.22.5.64 mem_tcam384_set16_entry_ptr40

Prototype:

```
void mem_tcam384_set16_entry_ptr40(mem_tcam384_ptr40_t tcam, unsigned int entry_index, unsigned int value, unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 16 bit entry value and mask in a 384 bit TCAM in 40-bit addressed MEM.



Note

The maximum index is 11.

Table 3.851. mem_tcam384_set16_entry_ptr40 parameters

Type	Name	Description
<code>mem_tcam384_ptr40_t</code>	<code>tcam</code>	Pointer to the TCAM structure in MEM
<code>unsigned int</code>	<code>entry_index</code>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
<code>unsigned int</code>	<code>value</code>	Value to set the TCAM entry value to

Type	Name	Description
unsigned int	<i>mask</i>	Value to set the TCAM entry mask to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.65 mem_tcam512_set16_word_ptr32

Prototype:

```
void mem_tcam512_set16_word_ptr32(__mem mem_tcam512_in_mem_t* tcam, unsigned int index,
unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 16 bit word in a 512 bit TCAM.



Note

The maximum index is 31.

Table 3.852. mem_tcam512_set16_word_ptr32 parameters

Type	Name	Description
__mem mem_tcam512_in_mem_t*	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>index</i>	TCAM word to set (0 is the first entry value, 2 is the first entry mask, 1 is the second entry value)
unsigned int	<i>value</i>	Value to set the TCAM word to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.66 mem_tcam512_set16_word_ptr40

Prototype:

```
void mem_tcam512_set16_word_ptr40(mem_tcam512_ptr40_t tcam, unsigned int index, unsigned
int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 16 bit word in a 512 bit TCAM in 40-bit addressed MEM.



Note

The maximum index is 31.

Table 3.853. mem_tcam512_set16_word_ptr40 parameters

Type	Name	Description
mem_tcam512_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>index</i>	TCAM word to set (0 is the first entry value, 2 is the first entry mask, 1 is the second entry value)
unsigned int	<i>value</i>	Value to set the TCAM word to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.67 mem_tcam512_set16_entry_value_ptr32

Prototype:

```
void mem_tcam512_set16_entry_value_ptr32(__mem mem_tcam512_in_mem_t* tcam, unsigned int
entry_index, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 16 bit entry value in a 512 bit TCAM.



Note

The maximum index is 15.

Table 3.854. mem_tcam512_set16_entry_value_ptr32 parameters

Type	Name	Description
__mem mem_tcam512_in_mem_t*	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
unsigned int	<i>value</i>	Value to set the TCAM entry value to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.68 mem_tcam512_set16_entry_value_ptr40

Prototype:

```
void mem_tcam512_set16_entry_value_ptr40(mem_tcam512_ptr40_t tcam, unsigned int
entry_index, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 16 bit entry value in a 512 bit TCAM in 40-bit addressed MEM.



Note

The maximum index is 15.

Table 3.855. mem_tcam512_set16_entry_value_ptr40 parameters

Type	Name	Description
mem_tcam512_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
unsigned int	<i>value</i>	Value to set the TCAM entry value to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.69 mem_tcam512_set16_entry_mask_ptr32

Prototype:

```
void mem_tcam512_set16_entry_mask_ptr32(__mem mem_tcam512_in_mem_t* tcam, unsigned int
entry_index, unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 16 bit entry mask in a 512 bit TCAM.



Note

The maximum index is 15.

Table 3.856. mem_tcam512_set16_entry_mask_ptr32 parameters

Type	Name	Description
__mem mem_tcam512_in_mem_t*	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
unsigned int	<i>mask</i>	Value to set the TCAM entry mask to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)

Type	Name	Description
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.70 mem_tcam512_set16_entry_mask_ptr40

Prototype:

```
void mem_tcam512_set16_entry_mask_ptr40(mem_tcam512_ptr40_t tcam, unsigned int entry_index,
unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 16 bit entry mask in a 512 bit TCAM in 40-bit addressed MEM.



Note

The maximum index is 15.

Table 3.857. mem_tcam512_set16_entry_mask_ptr40 parameters

Type	Name	Description
mem_tcam512_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
unsigned int	<i>mask</i>	Value to set the TCAM entry mask to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.71 mem_tcam512_set16_entry_ptr32

Prototype:

```
void mem_tcam512_set16_entry_ptr32(__mem mem_tcam512_in_mem_t* tcam, unsigned int entry_index,
unsigned int value, unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 16 bit entry value and mask in a 512 bit TCAM.



Note

The maximum index is 15.

Table 3.858. mem_tcam512_set16_entry_ptr32 parameters

Type	Name	Description
<code>__mem mem_tcam512_in_mem_t*</code>	<code>tcam</code>	Pointer to the TCAM structure in MEM
<code>unsigned int</code>	<code>entry_index</code>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
<code>unsigned int</code>	<code>value</code>	Value to set the TCAM entry value to
<code>unsigned int</code>	<code>mask</code>	Value to set the TCAM entry mask to
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.22.5.72 mem_tcam512_set16_entry_ptr40

Prototype:

```
void mem_tcam512_set16_entry_ptr40(mem_tcam512_ptr40_t tcam, unsigned int entry_index,
                                    unsigned int value, unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 16 bit entry value and mask in a 512 bit TCAM in 40-bit addressed MEM.



Note

The maximum index is 15.

Table 3.859. mem_tcam512_set16_entry_ptr40 parameters

Type	Name	Description
<code>mem_tcam512_ptr40_t</code>	<code>tcam</code>	Pointer to the TCAM structure in MEM
<code>unsigned int</code>	<code>entry_index</code>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
<code>unsigned int</code>	<code>value</code>	Value to set the TCAM entry value to
<code>unsigned int</code>	<code>mask</code>	Value to set the TCAM entry mask to
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.22.5.73 mem_tcam128_set32_word_ptr32

Prototype:

```
void mem_tcam128_set32_word_ptr32(__mem mem_tcam128_in_mem_t* tcam, unsigned int index,
                                    unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 32 bit word in a 128 bit TCAM.



Note

The maximum index is 3.

Table 3.860. mem_tcam128_set32_word_ptr32 parameters

Type	Name	Description
<code>_mem mem_tcam128_in_mem_t*</code>	<code>tcam</code>	Pointer to the TCAM structure in MEM
<code>unsigned int</code>	<code>index</code>	TCAM word to set (0 is the first entry value, 1 is the first entry mask, 2 is the second entry value)
<code>unsigned int</code>	<code>value</code>	Value to set the TCAM word to
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.22.5.74 mem_tcam128_set32_word_ptr40

Prototype:

```
void mem_tcam128_set32_word_ptr40(mem_tcam128_ptr40_t tcam, unsigned int index, unsigned
int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 32 bit word in a 128 bit TCAM in 40-bit addressed MEM.



Note

The maximum index is 3.

Table 3.861. mem_tcam128_set32_word_ptr40 parameters

Type	Name	Description
<code>mem_tcam128_ptr40_t</code>	<code>tcam</code>	Pointer to the TCAM structure in MEM
<code>unsigned int</code>	<code>index</code>	TCAM word to set (0 is the first entry value, 1 is the first entry mask, 2 is the second entry value)
<code>unsigned int</code>	<code>value</code>	Value to set the TCAM word to
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.22.5.75 mem_tcam128_set32_entry_value_ptr32

Prototype:

```
void mem_tcam128_set32_entry_value_ptr32(__mem mem_tcam128_in_mem_t* tcam, unsigned int entry_index, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 32 bit entry value in a 128 bit TCAM.

Note

The maximum index is 1.

Table 3.862. mem_tcam128_set32_entry_value_ptr32 parameters

Type	Name	Description
__mem mem_tcam128_in_mem_t*	tcam	Pointer to the TCAM structure in MEM
unsigned int	entry_index	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
unsigned int	value	Value to set the TCAM entry value to
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

3.22.5.76 mem_tcam128_set32_entry_value_ptr40

Prototype:

```
void mem_tcam128_set32_entry_value_ptr40(mem_tcam128_ptr40_t tcam, unsigned int entry_index, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 32 bit entry value in a 128 bit TCAM in 40-bit addressed MEM.

Note

The maximum index is 1.

Table 3.863. mem_tcam128_set32_entry_value_ptr40 parameters

Type	Name	Description
mem_tcam128_ptr40_t	tcam	Pointer to the TCAM structure in MEM

Type	Name	Description
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
unsigned int	<i>value</i>	Value to set the TCAM entry value to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.77 mem_tcam128_set32_entry_mask_ptr32

Prototype:

```
void mem_tcam128_set32_entry_mask_ptr32(__mem mem_tcam128_in_mem_t* tcam, unsigned int entry_index, unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 32 bit entry mask in a 128 bit TCAM.



Note

The maximum index is 1.

Table 3.864. mem_tcam128_set32_entry_mask_ptr32 parameters

Type	Name	Description
__mem mem_tcam128_in_mem_t*	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
unsigned int	<i>mask</i>	Value to set the TCAM entry mask to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.78 mem_tcam128_set32_entry_mask_ptr40

Prototype:

```
void mem_tcam128_set32_entry_mask_ptr40(mem_tcam128_ptr40_t tcam, unsigned int entry_index, unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 32 bit entry mask in a 128 bit TCAM in 40-bit addressed MEM.



Note

The maximum index is 1.

Table 3.865. mem_tcam128_set32_entry_mask_ptr40 parameters

Type	Name	Description
mem_tcam128_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
unsigned int	<i>mask</i>	Value to set the TCAM entry mask to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.79 mem_tcam128_set32_entry_ptr32

Prototype:

```
void mem_tcam128_set32_entry_ptr32(__mem mem_tcam128_in_mem_t* tcam, unsigned int
entry_index, unsigned int value, unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 32 bit entry value and mask in a 128 bit TCAM.



Note

The maximum index is 1.

Table 3.866. mem_tcam128_set32_entry_ptr32 parameters

Type	Name	Description
__mem mem_tcam128_in_mem_t*	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
unsigned int	<i>value</i>	Value to set the TCAM entry value to
unsigned int	<i>mask</i>	Value to set the TCAM entry mask to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.80 mem_tcam128_set32_entry_ptr40

Prototype:

```
void mem_tcam128_set32_entry_ptr40(mem_tcam128_ptr40_t tcam, unsigned int entry_index,
unsigned int value, unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 32 bit entry value and mask in a 128 bit TCAM in 40-bit addressed MEM.



Note

The maximum index is 1.

Table 3.867. mem_tcam128_set32_entry_ptr40 parameters

Type	Name	Description
mem_tcam128_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
unsigned int	<i>value</i>	Value to set the TCAM entry value to
unsigned int	<i>mask</i>	Value to set the TCAM entry mask to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.81 mem_tcam256_set32_word_ptr32

Prototype:

```
void mem_tcam256_set32_word_ptr32(__mem mem_tcam256_in_mem_t* tcam, unsigned int index,
unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 32 bit word in a 256 bit TCAM.



Note

The maximum index is 7.

Table 3.868. mem_tcam256_set32_word_ptr32 parameters

Type	Name	Description
__mem mem_tcam256_in_mem_t*	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>index</i>	TCAM word to set (0 is the first entry value, 1 is the first entry mask, 2 is the second entry value)
unsigned int	<i>value</i>	Value to set the TCAM word to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)

Type	Name	Description
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.82 mem_tcam256_set32_word_ptr40

Prototype:

```
void mem_tcam256_set32_word_ptr40(mem_tcam256_ptr40_t tcam, unsigned int index, unsigned
int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 32 bit word in a 256 bit TCAM in 40-bit addressed MEM.



Note

The maximum index is 7.

Table 3.869. mem_tcam256_set32_word_ptr40 parameters

Type	Name	Description
mem_tcam256_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>index</i>	TCAM word to set (0 is the first entry value, 1 is the first entry mask, 2 is the second entry value)
unsigned int	<i>value</i>	Value to set the TCAM word to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.83 mem_tcam256_set32_entry_value_ptr32

Prototype:

```
void mem_tcam256_set32_entry_value_ptr32(__mem mem_tcam256_in_mem_t* tcam, unsigned int
entry_index, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 32 bit entry value in a 256 bit TCAM.



Note

The maximum index is 3.

Table 3.870. mem_tcam256_set32_entry_value_ptr32 parameters

Type	Name	Description
<code>__mem mem_tcam256_in_mem_t*</code>	<code>tcam</code>	Pointer to the TCAM structure in MEM
<code>unsigned int</code>	<code>entry_index</code>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
<code>unsigned int</code>	<code>value</code>	Value to set the TCAM entry value to
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.22.5.84 mem_tcam256_set32_entry_value_ptr40

Prototype:

```
void mem_tcam256_set32_entry_value_ptr40(mem_tcam256_ptr40_t tcam, unsigned int
entry_index, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 32 bit entry value in a 256 bit TCAM in 40-bit addressed MEM.



Note

The maximum index is 3.

Table 3.871. mem_tcam256_set32_entry_value_ptr40 parameters

Type	Name	Description
<code>mem_tcam256_ptr40_t</code>	<code>tcam</code>	Pointer to the TCAM structure in MEM
<code>unsigned int</code>	<code>entry_index</code>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
<code>unsigned int</code>	<code>value</code>	Value to set the TCAM entry value to
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.22.5.85 mem_tcam256_set32_entry_mask_ptr32

Prototype:

```
void mem_tcam256_set32_entry_mask_ptr32(__mem mem_tcam256_in_mem_t* tcam, unsigned int
entry_index, unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 32 bit entry mask in a 256 bit TCAM.



Note

The maximum index is 3.

Table 3.872. mem_tcam256_set32_entry_mask_ptr32 parameters

Type	Name	Description
<code>__mem mem_tcam256_in_mem_t*</code>	<code>tcam</code>	Pointer to the TCAM structure in MEM
<code>unsigned int</code>	<code>entry_index</code>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
<code>unsigned int</code>	<code>mask</code>	Value to set the TCAM entry mask to
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.22.5.86 mem_tcam256_set32_entry_mask_ptr40

Prototype:

```
void mem_tcam256_set32_entry_mask_ptr40(mem_tcam256_ptr40_t tcam, unsigned int entry_index,
                                         unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 32 bit entry mask in a 256 bit TCAM in 40-bit addressed MEM.



Note

The maximum index is 3.

Table 3.873. mem_tcam256_set32_entry_mask_ptr40 parameters

Type	Name	Description
<code>mem_tcam256_ptr40_t</code>	<code>tcam</code>	Pointer to the TCAM structure in MEM
<code>unsigned int</code>	<code>entry_index</code>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
<code>unsigned int</code>	<code>mask</code>	Value to set the TCAM entry mask to
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.22.5.87 mem_tcam256_set32_entry_ptr32

Prototype:

```
void mem_tcam256_set32_entry_ptr32(__mem mem_tcam256_in_mem_t* tcam, unsigned int entry_index, unsigned int value, unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 32 bit entry value and mask in a 256 bit TCAM.



Note

The maximum index is 3.

Table 3.874. mem_tcam256_set32_entry_ptr32 parameters

Type	Name	Description
<code>__mem mem_tcam256_in_mem_t*</code>	<code>tcam</code>	Pointer to the TCAM structure in MEM
<code>unsigned int</code>	<code>entry_index</code>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
<code>unsigned int</code>	<code>value</code>	Value to set the TCAM entry value to
<code>unsigned int</code>	<code>mask</code>	Value to set the TCAM entry mask to
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.22.5.88 mem_tcam256_set32_entry_ptr40

Prototype:

```
void mem_tcam256_set32_entry_ptr40(mem_tcam256_ptr40_t tcam, unsigned int entry_index, unsigned int value, unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 32 bit entry value and mask in a 256 bit TCAM in 40-bit addressed MEM.



Note

The maximum index is 3.

Table 3.875. mem_tcam256_set32_entry_ptr40 parameters

Type	Name	Description
<code>mem_tcam256_ptr40_t</code>	<code>tcam</code>	Pointer to the TCAM structure in MEM
<code>unsigned int</code>	<code>entry_index</code>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
<code>unsigned int</code>	<code>value</code>	Value to set the TCAM entry value to
<code>unsigned int</code>	<code>mask</code>	Value to set the TCAM entry mask to

Type	Name	Description
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.89 mem_tcam384_set32_word_ptr32

Prototype:

```
void mem_tcam384_set32_word_ptr32(__mem mem_tcam384_in_mem_t* tcam, unsigned int index,
unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 32 bit word in a 384 bit TCAM.



Note

The maximum index is 11.

Table 3.876. mem_tcam384_set32_word_ptr32 parameters

Type	Name	Description
__mem mem_tcam384_in_mem_t*	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>index</i>	TCAM word to set (0 is the first entry value, 1 is the first entry mask, 2 is the second entry value)
unsigned int	<i>value</i>	Value to set the TCAM word to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.90 mem_tcam384_set32_word_ptr40

Prototype:

```
void mem_tcam384_set32_word_ptr40(mem_tcam384_ptr40_t tcam, unsigned int index, unsigned
int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 32 bit word in a 384 bit TCAM in 40-bit addressed MEM.



Note

The maximum index is 11.

Table 3.877. mem_tcam384_set32_word_ptr40 parameters

Type	Name	Description
mem_tcam384_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>index</i>	TCAM word to set (0 is the first entry value, 1 is the first entry mask, 2 is the second entry value)
unsigned int	<i>value</i>	Value to set the TCAM word to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.91 mem_tcam384_set32_entry_value_ptr32

Prototype:

```
void mem_tcam384_set32_entry_value_ptr32(__mem mem_tcam384_in_mem_t* tcam, unsigned int
entry_index, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 32 bit entry value in a 384 bit TCAM.



Note

The maximum index is 5.

Table 3.878. mem_tcam384_set32_entry_value_ptr32 parameters

Type	Name	Description
__mem mem_tcam384_in_mem_t*	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
unsigned int	<i>value</i>	Value to set the TCAM entry value to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.92 mem_tcam384_set32_entry_value_ptr40

Prototype:

```
void mem_tcam384_set32_entry_value_ptr40(mem_tcam384_ptr40_t tcam, unsigned int
entry_index, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 32 bit entry value in a 384 bit TCAM in 40-bit addressed MEM.



Note

The maximum index is 5.

Table 3.879. mem_tcam384_set32_entry_value_ptr40 parameters

Type	Name	Description
mem_tcam384_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
unsigned int	<i>value</i>	Value to set the TCAM entry value to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.93 mem_tcam384_set32_entry_mask_ptr32

Prototype:

```
void mem_tcam384_set32_entry_mask_ptr32(__mem mem_tcam384_in_mem_t* tcam, unsigned int
entry_index, unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 32 bit entry mask in a 384 bit TCAM.



Note

The maximum index is 5.

Table 3.880. mem_tcam384_set32_entry_mask_ptr32 parameters

Type	Name	Description
__mem mem_tcam384_in_mem_t*	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
unsigned int	<i>mask</i>	Value to set the TCAM entry mask to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.94 mem_tcam384_set32_entry_mask_ptr40

Prototype:

```
void mem_tcam384_set32_entry_mask_ptr40(mem_tcam384_ptr40_t tcam, unsigned int entry_index,
unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 32 bit entry mask in a 384 bit TCAM in 40-bit addressed MEM.



Note

The maximum index is 5.

Table 3.881. mem_tcam384_set32_entry_mask_ptr40 parameters

Type	Name	Description
mem_tcam384_ptr40_t	tcam	Pointer to the TCAM structure in MEM
unsigned int	entry_index	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
unsigned int	mask	Value to set the TCAM entry mask to
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

3.22.5.95 mem_tcam384_set32_entry_ptr32

Prototype:

```
void mem_tcam384_set32_entry_ptr32(__mem mem_tcam384_in_mem_t* tcam, unsigned int
entry_index, unsigned int value, unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 32 bit entry value and mask in a 384 bit TCAM.



Note

The maximum index is 5.

Table 3.882. mem_tcam384_set32_entry_ptr32 parameters

Type	Name	Description
__mem mem_tcam384_in_mem_t*	tcam	Pointer to the TCAM structure in MEM
unsigned int	entry_index	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
unsigned int	value	Value to set the TCAM entry value to
unsigned int	mask	Value to set the TCAM entry mask to
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)

Type	Name	Description
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.96 mem_tcam384_set32_entry_ptr40

Prototype:

```
void mem_tcam384_set32_entry_ptr40(mem_tcam384_ptr40_t tcam, unsigned int entry_index,
unsigned int value, unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 32 bit entry value and mask in a 384 bit TCAM in 40-bit addressed MEM.



Note

The maximum index is 5.

Table 3.883. mem_tcam384_set32_entry_ptr40 parameters

Type	Name	Description
mem_tcam384_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
unsigned int	<i>value</i>	Value to set the TCAM entry value to
unsigned int	<i>mask</i>	Value to set the TCAM entry mask to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.97 mem_tcam512_set32_word_ptr32

Prototype:

```
void mem_tcam512_set32_word_ptr32(__mem mem_tcam512_in_mem_t* tcam, unsigned int index,
unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 32 bit word in a 512 bit TCAM.



Note

The maximum index is 15.

Table 3.884. mem_tcam512_set32_word_ptr32 parameters

Type	Name	Description
<code>__mem mem_tcam512_in_mem_t*</code>	<code>tcam</code>	Pointer to the TCAM structure in MEM
<code>unsigned int</code>	<code>index</code>	TCAM word to set (0 is the first entry value, 1 is the first entry mask, 2 is the second entry value)
<code>unsigned int</code>	<code>value</code>	Value to set the TCAM word to
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.22.5.98 mem_tcam512_set32_word_ptr40

Prototype:

```
void mem_tcam512_set32_word_ptr40(mem_tcam512_ptr40_t tcam, unsigned int index, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 32 bit word in a 512 bit TCAM in 40-bit addressed MEM.



Note

The maximum index is 15.

Table 3.885. mem_tcam512_set32_word_ptr40 parameters

Type	Name	Description
<code>mem_tcam512_ptr40_t</code>	<code>tcam</code>	Pointer to the TCAM structure in MEM
<code>unsigned int</code>	<code>index</code>	TCAM word to set (0 is the first entry value, 1 is the first entry mask, 2 is the second entry value)
<code>unsigned int</code>	<code>value</code>	Value to set the TCAM word to
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.22.5.99 mem_tcam512_set32_entry_value_ptr32

Prototype:

```
void mem_tcam512_set32_entry_value_ptr32(__mem mem_tcam512_in_mem_t* tcam, unsigned int entry_index, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 32 bit entry value in a 512 bit TCAM.



Note

The maximum index is 7.

Table 3.886. mem_tcam512_set32_entry_value_ptr32 parameters

Type	Name	Description
__mem mem_tcam512_in_mem_t*	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
unsigned int	<i>value</i>	Value to set the TCAM entry value to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.100 mem_tcam512_set32_entry_value_ptr40

Prototype:

```
void mem_tcam512_set32_entry_value_ptr40(mem_tcam512_ptr40_t tcam, unsigned int
entry_index, unsigned int value, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 32 bit entry value in a 512 bit TCAM in 40-bit addressed MEM.



Note

The maximum index is 7.

Table 3.887. mem_tcam512_set32_entry_value_ptr40 parameters

Type	Name	Description
mem_tcam512_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
unsigned int	<i>value</i>	Value to set the TCAM entry value to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.101 mem_tcam512_set32_entry_mask_ptr32

Prototype:

```
void mem_tcam512_set32_entry_mask_ptr32(__mem mem_tcam512_in_mem_t* tcam, unsigned int entry_index, unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 32 bit entry mask in a 512 bit TCAM.



Note

The maximum index is 7.

Table 3.888. mem_tcam512_set32_entry_mask_ptr32 parameters

Type	Name	Description
<code>__mem mem_tcam512_in_mem_t*</code>	<code>tcam</code>	Pointer to the TCAM structure in MEM
<code>unsigned int</code>	<code>entry_index</code>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
<code>unsigned int</code>	<code>mask</code>	Value to set the TCAM entry mask to
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.22.5.102 mem_tcam512_set32_entry_mask_ptr40

Prototype:

```
void mem_tcam512_set32_entry_mask_ptr40(mem_tcam512_ptr40_t tcam, unsigned int entry_index, unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 32 bit entry mask in a 512 bit TCAM in 40-bit addressed MEM.



Note

The maximum index is 7.

Table 3.889. mem_tcam512_set32_entry_mask_ptr40 parameters

Type	Name	Description
<code>mem_tcam512_ptr40_t</code>	<code>tcam</code>	Pointer to the TCAM structure in MEM
<code>unsigned int</code>	<code>entry_index</code>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
<code>unsigned int</code>	<code>mask</code>	Value to set the TCAM entry mask to
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.22.5.103 mem_tcam512_set32_entry_ptr32

Prototype:

```
void mem_tcam512_set32_entry_ptr32(__mem mem_tcam512_in_mem_t* tcam, unsigned int entry_index, unsigned int value, unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 32 bit entry value and mask in a 512 bit TCAM.

Note

The maximum index is 7.

Table 3.890. mem_tcam512_set32_entry_ptr32 parameters

Type	Name	Description
__mem mem_tcam512_in_mem_t*	tcam	Pointer to the TCAM structure in MEM
unsigned int	entry_index	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
unsigned int	value	Value to set the TCAM entry value to
unsigned int	mask	Value to set the TCAM entry mask to
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

3.22.5.104 mem_tcam512_set32_entry_ptr40

Prototype:

```
void mem_tcam512_set32_entry_ptr40(mem_tcam512_ptr40_t tcam, unsigned int entry_index, unsigned int value, unsigned int mask, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Set a 32 bit entry value and mask in a 512 bit TCAM in 40-bit addressed MEM.

Note

The maximum index is 7.

Table 3.891. mem_tcam512_set32_entry_ptr40 parameters

Type	Name	Description
mem_tcam512_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
unsigned int	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
unsigned int	<i>value</i>	Value to set the TCAM entry value to
unsigned int	<i>mask</i>	Value to set the TCAM entry mask to
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.22.5.105 mem_tcam128_lookup8_ptr32

Prototype:

```
mem_tcam_lookup8_out_in_read_reg_t* mem_tcam128_lookup8_ptr32(__mem mem_tcam128_in_mem_t* tcam, __xwrite mem_tcam_lookup8_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform an 8 bit lookup in a 128 bit TCAM.



Note

The result is returned in a mem_tcam_lookup8_out structure occupying two read transfer registers.

Table 3.892. mem_tcam128_lookup8_ptr32 parameters

Type	Name	Description
__mem mem_tcam128_in_mem_t*	<i>tcam</i>	Pointer to the TCAM structure in MEM
__xwrite mem_tcam_lookup8_in_t*	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup

See Also:

- mem_tcam_lookup8_in_t, mem_tcam_lookup8_out_t

3.22.5.106 mem_tcam128_lookup8_ptr40

Prototype:

```
mem_tcam_lookup8_out_in_read_reg_t* mem_tcam128_lookup8_ptr40(mem_tcam128_ptr40_t tcam,  
__xwrite mem_tcam_lookup8_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform an 8 bit lookup in a 128 bit TCAM in 40-bit addressed MEM.



Note

The result is returned in a mem_tcam_lookup8_out structure occupying two read transfer registers.

Table 3.893. mem_tcam128_lookup8_ptr40 parameters

Type	Name	Description
mem_tcam128_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
__xwrite mem_tcam_lookup8_in_t*	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup

See Also:

- mem_tcam_lookup8_in_t, mem_tcam_lookup8_out_t

3.22.5.107 mem_tcam256_lookup8_ptr32

Prototype:

```
mem_tcam_lookup8_out_in_read_reg_t* mem_tcam256_lookup8_ptr32(__mem mem_tcam256_in_mem_t*  
tcam, __xwrite mem_tcam_lookup8_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform an 8 bit lookup in a 256 bit TCAM.



Note

The result is returned in a mem_tcam_lookup8_out structure occupying two read transfer registers.

Table 3.894. mem_tcam256_lookup8_ptr32 parameters

Type	Name	Description
__mem mem_tcam256_in_mem_t*	<i>tcam</i>	Pointer to the TCAM structure in MEM
__xwrite mem_tcam_lookup8_in_t*	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup

See Also:

- mem_tcam_lookup8_in_t, mem_tcam_lookup8_out_t

3.22.5.108 mem_tcam256_lookup8_ptr40

Prototype:

```
mem_tcam_lookup8_out_in_read_reg_t* mem_tcam256_lookup8_ptr40(mem_tcam256_ptr40_t tcam,  
__xwrite mem_tcam_lookup8_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform an 8 bit lookup in a 256 bit TCAM in 40-bit addressed MEM.



Note

The result is returned in a mem_tcam_lookup8_out structure occupying two read transfer registers.

Table 3.895. mem_tcam256_lookup8_ptr40 parameters

Type	Name	Description
mem_tcam256_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
__xwrite mem_tcam_lookup8_in_t*	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)

Type	Name	Description
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup

See Also:

- mem_tcam_lookup8_in_t, mem_tcam_lookup8_out_t

3.22.5.109 mem_tcam384_lookup8_ptr32

Prototype:

```
mem_tcam_lookup8_out_in_read_reg_t* mem_tcam384_lookup8_ptr32(__mem mem_tcam384_in_mem_t* tcam, __xwrite mem_tcam_lookup8_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform an 8 bit lookup in a 384 bit TCAM.



Note

The result is returned in a mem_tcam_lookup8_out structure occupying two read transfer registers.

Table 3.896. mem_tcam384_lookup8_ptr32 parameters

Type	Name	Description
__mem mem_tcam384_in_mem_t*	<i>tcam</i>	Pointer to the TCAM structure in MEM
__xwrite mem_tcam_lookup8_in_t*	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup

See Also:

- mem_tcam_lookup8_in_t, mem_tcam_lookup8_out_t

3.22.5.110 mem_tcam384_lookup8_ptr40

Prototype:

```
mem_tcam_lookup8_out_in_read_reg_t* mem_tcam384_lookup8_ptr40(mem_tcam384_ptr40_t tcam,  
__xwrite mem_tcam_lookup8_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform an 8 bit lookup in a 384 bit TCAM in 40-bit addressed MEM.



Note

The result is returned in a mem_tcam_lookup8_out structure occupying two read transfer registers.

Table 3.897. mem_tcam384_lookup8_ptr40 parameters

Type	Name	Description
mem_tcam384_ptr40_t	tcam	Pointer to the TCAM structure in MEM
__xwrite mem_tcam_lookup8_in_t*	xfer	Pointer to write xfer register containing the data to lookup
sync_t	sync	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	sig_pair_ptr	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup

See Also:

- mem_tcam_lookup8_in_t, mem_tcam_lookup8_out_t

3.22.5.111 mem_tcam512_lookup8_ptr32

Prototype:

```
mem_tcam_lookup8_out_in_read_reg_t* mem_tcam512_lookup8_ptr32(__mem mem_tcam512_in_mem_t*  
tcam, __xwrite mem_tcam_lookup8_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform an 8 bit lookup in a 512 bit TCAM.



Note

The result is returned in a mem_tcam_lookup8_out structure occupying two read transfer registers.

Table 3.898. mem_tcam512_lookup8_ptr32 parameters

Type	Name	Description
<code>__mem mem_tcam512_in_mem_t*</code>	<code>tcam</code>	Pointer to the TCAM structure in MEM
<code>__xwrite mem_tcam_lookup8_in_t*</code>	<code>xfer</code>	Pointer to write xfer register containing the data to lookup
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be sig_done)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup

See Also:

- `mem_tcam_lookup8_in_t`, `mem_tcam_lookup8_out_t`

3.22.5.112 mem_tcam512_lookup8_ptr40

Prototype:

```
mem_tcam_lookup8_out_in_read_reg_t* mem_tcam512_lookup8_ptr40(mem_tcam512_ptr40_t tcam,  
__xwrite mem_tcam_lookup8_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform an 8 bit lookup in a 512 bit TCAM in 40-bit addressed MEM.



Note

The result is returned in a `mem_tcam_lookup8_out` structure occupying two read transfer registers.

Table 3.899. mem_tcam512_lookup8_ptr40 parameters

Type	Name	Description
<code>mem_tcam512_ptr40_t</code>	<code>tcam</code>	Pointer to the TCAM structure in MEM
<code>__xwrite mem_tcam_lookup8_in_t*</code>	<code>xfer</code>	Pointer to write xfer register containing the data to lookup
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be sig_done)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup

See Also:

- `mem_tcam_lookup8_in_t`, `mem_tcam_lookup8_out_t`

3.22.5.113 mem_tcam128_lookup16_ptr32

Prototype:

```
mem_tcam_lookup16_out_in_read_reg_t* mem_tcam128_lookup16_ptr32(__mem mem_tcam128_in_mem_t* tcam, __xwrite mem_tcam_lookup16_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 16 bit lookup in a 128 bit TCAM.



Note

The result is returned in a `mem_tcam_lookup16_out` structure in one read transfer register.

Table 3.900. mem_tcam128_lookup16_ptr32 parameters

Type	Name	Description
<code>__mem mem_tcam128_in_mem_t*</code>	<code>tcam</code>	Pointer to the TCAM structure in MEM
<code>__xwrite mem_tcam_lookup16_in_t*</code>	<code>xfer</code>	Pointer to write xfer register containing the data to lookup
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be <code>sig_done</code>)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup

See Also:

- `mem_tcam_lookup16_in_t`, `mem_tcam_lookup16_out_t`

3.22.5.114 mem_tcam128_lookup16_ptr40

Prototype:

```
mem_tcam_lookup16_out_in_read_reg_t* mem_tcam128_lookup16_ptr40(mem_tcam128_ptr40_t tcam, __xwrite mem_tcam_lookup16_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 16 bit lookup in a 128 bit TCAM in 40-bit addressed MEM.



Note

The result is returned in a mem_tcam_lookup16_out structure in one read transfer register.

Table 3.901. mem_tcam128_lookup16_ptr40 parameters

Type	Name	Description
mem_tcam128_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
__xwrite mem_tcam_lookup16_in_t*	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup

See Also:

- mem_tcam_lookup16_in_t, mem_tcam_lookup16_out_t

3.22.5.115 mem_tcam256_lookup16_ptr32

Prototype:

```
mem_tcam_lookup16_out_in_read_reg_t* mem_tcam256_lookup16_ptr32(__mem mem_tcam256_in_mem_t*
tcam, __xwrite mem_tcam_lookup16_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 16 bit lookup in a 256 bit TCAM.



Note

The result is returned in a mem_tcam_lookup16_out structure in one read transfer register.

Table 3.902. mem_tcam256_lookup16_ptr32 parameters

Type	Name	Description
__mem mem_tcam256_in_mem_t*	<i>tcam</i>	Pointer to the TCAM structure in MEM
__xwrite mem_tcam_lookup16_in_t*	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup

See Also:

- `mem_tcam_lookup16_in_t`, `mem_tcam_lookup16_out_t`

3.22.5.116 mem_tcam256_lookup16_ptr40

Prototype:

```
mem_tcam_lookup16_out_in_read_reg_t* mem_tcam256_lookup16_ptr40(mem_tcam256_ptr40_t tcam,  
__xwrite mem_tcam_lookup16_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 16 bit lookup in a 256 bit TCAM in 40-bit addressed MEM.



Note

The result is returned in a `mem_tcam_lookup16_out` structure in one read transfer register.

Table 3.903. mem_tcam256_lookup16_ptr40 parameters

Type	Name	Description
<code>mem_tcam256_ptr40_t</code>	<code>tcam</code>	Pointer to the TCAM structure in MEM
<code>__xwrite mem_tcam_lookup16_in_t*</code>	<code>xfer</code>	Pointer to write xfer register containing the data to lookup
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be <code>sig_done</code>)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup

See Also:

- `mem_tcam_lookup16_in_t`, `mem_tcam_lookup16_out_t`

3.22.5.117 mem_tcam384_lookup16_ptr32

Prototype:

```
mem_tcam_lookup16_out_in_read_reg_t* mem_tcam384_lookup16_ptr32(__mem mem_tcam384_in_mem_t*  
tcam, __xwrite mem_tcam_lookup16_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 16 bit lookup in a 384 bit TCAM.



Note

The result is returned in a mem_tcam_lookup16_out structure in one read transfer register.

Table 3.904. mem_tcam384_lookup16_ptr32 parameters

Type	Name	Description
__mem mem_tcam384_in_mem_t*	<i>tcam</i>	Pointer to the TCAM structure in MEM
__xwrite mem_tcam_lookup16_in_t*	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup

See Also:

- mem_tcam_lookup16_in_t, mem_tcam_lookup16_out_t

3.22.5.118 mem_tcam384_lookup16_ptr40

Prototype:

```
mem_tcam_lookup16_out_in_read_reg_t* mem_tcam384_lookup16_ptr40(mem_tcam384_ptr40_t tcam,  
__xwrite mem_tcam_lookup16_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 16 bit lookup in a 384 bit TCAM in 40-bit addressed MEM.



Note

The result is returned in a mem_tcam_lookup16_out structure in one read transfer register.

Table 3.905. mem_tcam384_lookup16_ptr40 parameters

Type	Name	Description
mem_tcam384_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
__xwrite mem_tcam_lookup16_in_t*	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)

Type	Name	Description
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup

See Also:

- `mem_tcam_lookup16_in_t`, `mem_tcam_lookup16_out_t`

3.22.5.119 mem_tcam512_lookup16_ptr32

Prototype:

```
mem_tcam_lookup16_out_in_read_reg_t* mem_tcam512_lookup16_ptr32(__mem mem_tcam512_in_mem_t*
tcam, __xwrite mem_tcam_lookup16_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 16 bit lookup in a 512 bit TCAM.



Note

The result is returned in a `mem_tcam_lookup16_out` structure in one read transfer register.

Table 3.906. mem_tcam512_lookup16_ptr32 parameters

Type	Name	Description
<code>__mem mem_tcam512_in_mem_t*</code>	<i>tcam</i>	Pointer to the TCAM structure in MEM
<code>__xwrite mem_tcam_lookup16_in_t*</code>	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
<code>sync_t</code>	<i>sync</i>	Type of synchronization to use (must be <code>sig_done</code>)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup

See Also:

- `mem_tcam_lookup16_in_t`, `mem_tcam_lookup16_out_t`

3.22.5.120 mem_tcam512_lookup16_ptr40

Prototype:

```
mem_tcam_lookup16_out_in_read_reg_t* mem_tcam512_lookup16_ptr40(mem_tcam512_ptr40_t tcam,  
__xwrite mem_tcam_lookup16_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 16 bit lookup in a 512 bit TCAM in 40-bit addressed MEM.



Note

The result is returned in a mem_tcam_lookup16_out structure in one read transfer register.

Table 3.907. mem_tcam512_lookup16_ptr40 parameters

Type	Name	Description
mem_tcam512_ptr40_t	tcam	Pointer to the TCAM structure in MEM
__xwrite mem_tcam_lookup16_in_t*	xfer	Pointer to write xfer register containing the data to lookup
sync_t	sync	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	sig_pair_ptr	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup

See Also:

- mem_tcam_lookup16_in_t, mem_tcam_lookup16_out_t

3.22.5.121 mem_tcam128_lookup24_ptr32

Prototype:

```
mem_tcam_lookup24_out_in_read_reg_t* mem_tcam128_lookup24_ptr32(__mem mem_tcam128_in_mem_t*  
tcam, __xwrite mem_tcam_lookup24_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 24 bit lookup in a 128 bit TCAM.



Note

The result is returned in a mem_tcam_lookup24_out structure in one read transfer register.

Table 3.908. mem_tcam128_lookup24_ptr32 parameters

Type	Name	Description
__mem mem_tcam128_in_mem_t*	<i>tcam</i>	Pointer to the TCAM structure in MEM
__xwrite mem_tcam_lookup24_in_t*	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup

See Also:

- mem_tcam_lookup24_in_t, mem_tcam_lookup24_out_t

3.22.5.122 mem_tcam128_lookup24_ptr40

Prototype:

```
mem_tcam_lookup24_out_in_read_reg_t* mem_tcam128_lookup24_ptr40(mem_tcam128_ptr40_t tcam,  
__xwrite mem_tcam_lookup24_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 24 bit lookup in a 128 bit TCAM in 40-bit addressed MEM.



Note

The result is returned in a mem_tcam_lookup24_out structure in one read transfer register.

Table 3.909. mem_tcam128_lookup24_ptr40 parameters

Type	Name	Description
mem_tcam128_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
__xwrite mem_tcam_lookup24_in_t*	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup

See Also:

- mem_tcam_lookup24_in_t, mem_tcam_lookup24_out_t

3.22.5.123 mem_tcam256_lookup24_ptr32

Prototype:

```
mem_tcam_lookup24_out_in_read_reg_t* mem_tcam256_lookup24_ptr32(__mem mem_tcam256_in_mem_t*  
tcam, __xwrite mem_tcam_lookup24_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 24 bit lookup in a 256 bit TCAM.



Note

The result is returned in a mem_tcam_lookup24_out structure in one read transfer register.

Table 3.910. mem_tcam256_lookup24_ptr32 parameters

Type	Name	Description
__mem mem_tcam256_in_mem_t*	tcam	Pointer to the TCAM structure in MEM
__xwrite mem_tcam_lookup24_in_t*	xfer	Pointer to write xfer register containing the data to lookup
sync_t	sync	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	sig_pair_ptr	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup

See Also:

- mem_tcam_lookup24_in_t, mem_tcam_lookup24_out_t

3.22.5.124 mem_tcam256_lookup24_ptr40

Prototype:

```
mem_tcam_lookup24_out_in_read_reg_t* mem_tcam256_lookup24_ptr40(mem_tcam256_ptr40_t tcam,  
__xwrite mem_tcam_lookup24_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 24 bit lookup in a 256 bit TCAM in 40-bit addressed MEM.



Note

The result is returned in a mem_tcam_lookup24_out structure in one read transfer register.

Table 3.911. mem_tcam256_lookup24_ptr40 parameters

Type	Name	Description
mem_tcam256_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
__xwrite mem_tcam_lookup24_in_t*	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup

See Also:

- [mem_tcam_lookup24_in_t, mem_tcam_lookup24_out_t](#)

3.22.5.125 mem_tcam384_lookup24_ptr32

Prototype:

```
mem_tcam_lookup24_out_in_read_reg_t* mem_tcam384_lookup24_ptr32(__mem mem_tcam384_in_mem_t*
tcam, __xwrite mem_tcam_lookup24_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 24 bit lookup in a 384 bit TCAM.



Note

The result is returned in a mem_tcam_lookup24_out structure in one read transfer register.

Table 3.912. mem_tcam384_lookup24_ptr32 parameters

Type	Name	Description
__mem mem_tcam384_in_mem_t*	<i>tcam</i>	Pointer to the TCAM structure in MEM
__xwrite mem_tcam_lookup24_in_t*	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup

See Also:

- `mem_tcam_lookup24_in_t`, `mem_tcam_lookup24_out_t`

3.22.5.126 mem_tcam384_lookup24_ptr40

Prototype:

```
mem_tcam_lookup24_out_in_read_reg_t* mem_tcam384_lookup24_ptr40(mem_tcam384_ptr40_t tcam,  
__xwrite mem_tcam_lookup24_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 24 bit lookup in a 384 bit TCAM in 40-bit addressed MEM.



Note

The result is returned in a `mem_tcam_lookup24_out` structure in one read transfer register.

Table 3.913. mem_tcam384_lookup24_ptr40 parameters

Type	Name	Description
<code>mem_tcam384_ptr40_t</code>	<code>tcam</code>	Pointer to the TCAM structure in MEM
<code>__xwrite mem_tcam_lookup24_in_t*</code>	<code>xfer</code>	Pointer to write xfer register containing the data to lookup
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be <code>sig_done</code>)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup

See Also:

- `mem_tcam_lookup24_in_t`, `mem_tcam_lookup24_out_t`

3.22.5.127 mem_tcam512_lookup24_ptr32

Prototype:

```
mem_tcam_lookup24_out_in_read_reg_t* mem_tcam512_lookup24_ptr32(__mem mem_tcam512_in_mem_t*  
tcam, __xwrite mem_tcam_lookup24_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 24 bit lookup in a 512 bit TCAM.



Note

The result is returned in a mem_tcam_lookup24_out structure in one read transfer register.

Table 3.914. mem_tcam512_lookup24_ptr32 parameters

Type	Name	Description
__mem mem_tcam512_in_mem_t*	<i>tcam</i>	Pointer to the TCAM structure in MEM
__xwrite mem_tcam_lookup24_in_t*	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup

See Also:

- mem_tcam_lookup24_in_t, mem_tcam_lookup24_out_t

3.22.5.128 mem_tcam512_lookup24_ptr40

Prototype:

```
mem_tcam_lookup24_out_in_read_reg_t* mem_tcam512_lookup24_ptr40(mem_tcam512_ptr40_t tcam,  
__xwrite mem_tcam_lookup24_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 24 bit lookup in a 512 bit TCAM in 40-bit addressed MEM.



Note

The result is returned in a mem_tcam_lookup24_out structure in one read transfer register.

Table 3.915. mem_tcam512_lookup24_ptr40 parameters

Type	Name	Description
mem_tcam512_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
__xwrite mem_tcam_lookup24_in_t*	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)

Type	Name	Description
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup

See Also:

- mem_tcam_lookup24_in_t, mem_tcam_lookup24_out_t

3.22.5.129 mem_tcam128_lookup32_ptr32

Prototype:

```
mem_tcam_lookup32_out_in_read_reg_t* mem_tcam128_lookup32_ptr32(__mem mem_tcam128_in_mem_t*
tcam, __xwrite mem_tcam_lookup32_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 32 bit lookup in a 128 bit TCAM.



Note

The result is returned in a mem_tcam_lookup32_out structure in one read transfer register.

Table 3.916. mem_tcam128_lookup32_ptr32 parameters

Type	Name	Description
__mem mem_tcam128_in_mem_t*	<i>tcam</i>	Pointer to the TCAM structure in MEM
__xwrite mem_tcam_lookup32_in_t*	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup

See Also:

- mem_tcam_lookup32_in_t, mem_tcam_lookup32_out_t

3.22.5.130 mem_tcam128_lookup32_ptr40

Prototype:

```
mem_tcam_lookup32_out_in_read_reg_t* mem_tcam128_lookup32_ptr40(mem_tcam128_ptr40_t tcam,  
__xwrite mem_tcam_lookup32_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 32 bit lookup in a 128 bit TCAM in 40-bit addressed MEM.



Note

The result is returned in a mem_tcam_lookup32_out structure in one read transfer register.

Table 3.917. mem_tcam128_lookup32_ptr40 parameters

Type	Name	Description
mem_tcam128_ptr40_t	tcam	Pointer to the TCAM structure in MEM
__xwrite mem_tcam_lookup32_in_t*	xfer	Pointer to write xfer register containing the data to lookup
sync_t	sync	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	sig_pair_ptr	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup

See Also:

- mem_tcam_lookup32_in_t, mem_tcam_lookup32_out_t

3.22.5.131 mem_tcam256_lookup32_ptr32

Prototype:

```
mem_tcam_lookup32_out_in_read_reg_t* mem_tcam256_lookup32_ptr32(__mem mem_tcam256_in_mem_t*  
tcam, __xwrite mem_tcam_lookup32_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 32 bit lookup in a 256 bit TCAM.



Note

The result is returned in a mem_tcam_lookup32_out structure in one read transfer register.

Table 3.918. mem_tcam256_lookup32_ptr32 parameters

Type	Name	Description
<code>__mem mem_tcam256_in_mem_t*</code>	<code>tcam</code>	Pointer to the TCAM structure in MEM
<code>__xwrite mem_tcam_lookup32_in_t*</code>	<code>xfer</code>	Pointer to write xfer register containing the data to lookup
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be sig_done)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup

See Also:

- `mem_tcam_lookup32_in_t`, `mem_tcam_lookup32_out_t`

3.22.5.132 mem_tcam256_lookup32_ptr40

Prototype:

```
mem_tcam_lookup32_out_in_read_reg_t* mem_tcam256_lookup32_ptr40(mem_tcam256_ptr40_t tcam,  
__xwrite mem_tcam_lookup32_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 32 bit lookup in a 256 bit TCAM in 40-bit addressed MEM.



Note

The result is returned in a `mem_tcam_lookup32_out` structure in one read transfer register.

Table 3.919. mem_tcam256_lookup32_ptr40 parameters

Type	Name	Description
<code>mem_tcam256_ptr40_t</code>	<code>tcam</code>	Pointer to the TCAM structure in MEM
<code>__xwrite mem_tcam_lookup32_in_t*</code>	<code>xfer</code>	Pointer to write xfer register containing the data to lookup
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be sig_done)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup

See Also:

- mem_tcam_lookup32_in_t, mem_tcam_lookup32_out_t

3.22.5.133 mem_tcam384_lookup32_ptr32

Prototype:

```
mem_tcam_lookup32_out_in_read_reg_t* mem_tcam384_lookup32_ptr32(__mem mem_tcam384_in_mem_t*  
tcam, __xwrite mem_tcam_lookup32_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 32 bit lookup in a 384 bit TCAM.



Note

The result is returned in a mem_tcam_lookup32_out structure in one read transfer register.

Table 3.920. mem_tcam384_lookup32_ptr32 parameters

Type	Name	Description
__mem mem_tcam384_in_mem_t*	tcam	Pointer to the TCAM structure in MEM
__xwrite mem_tcam_lookup32_in_t*	xfer	Pointer to write xfer register containing the data to lookup
sync_t	sync	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	sig_pair_ptr	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup

See Also:

- mem_tcam_lookup32_in_t, mem_tcam_lookup32_out_t

3.22.5.134 mem_tcam384_lookup32_ptr40

Prototype:

```
mem_tcam_lookup32_out_in_read_reg_t* mem_tcam384_lookup32_ptr40(mem_tcam384_ptr40_t tcam,  
__xwrite mem_tcam_lookup32_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 32 bit lookup in a 384 bit TCAM in 40-bit addressed MEM.



Note

The result is returned in a mem_tcam_lookup32_out structure in one read transfer register.

Table 3.921. mem_tcam384_lookup32_ptr40 parameters

Type	Name	Description
mem_tcam384_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
__xwrite mem_tcam_lookup32_in_t*	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup

See Also:

- [mem_tcam_lookup32_in_t, mem_tcam_lookup32_out_t](#)

3.22.5.135 mem_tcam512_lookup32_ptr32

Prototype:

```
mem_tcam_lookup32_out_in_read_reg_t* mem_tcam512_lookup32_ptr32(__mem mem_tcam512_in_mem_t*
tcam, __xwrite mem_tcam_lookup32_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 32 bit lookup in a 512 bit TCAM.



Note

The result is returned in a mem_tcam_lookup32_out structure in one read transfer register.

Table 3.922. mem_tcam512_lookup32_ptr32 parameters

Type	Name	Description
__mem mem_tcam512_in_mem_t*	<i>tcam</i>	Pointer to the TCAM structure in MEM
__xwrite mem_tcam_lookup32_in_t*	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
sync_t	<i>sync</i>	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup

See Also:

- `mem_tcam_lookup32_in_t`, `mem_tcam_lookup32_out_t`

3.22.5.136 mem_tcam512_lookup32_ptr40

Prototype:

```
mem_tcam_lookup32_out_in_read_reg_t* mem_tcam512_lookup32_ptr40(mem_tcam512_ptr40_t tcam,  
__xwrite mem_tcam_lookup32_in_t* xfer, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Perform a 32 bit lookup in a 512 bit TCAM in 40-bit addressed MEM.



Note

The result is returned in a `mem_tcam_lookup32_out` structure in one read transfer register.

Table 3.923. mem_tcam512_lookup32_ptr40 parameters

Type	Name	Description
<code>mem_tcam512_ptr40_t</code>	<code>tcam</code>	Pointer to the TCAM structure in MEM
<code>__xwrite mem_tcam_lookup32_in_t*</code>	<code>xfer</code>	Pointer to write xfer register containing the data to lookup
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (must be <code>sig_done</code>)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup

See Also:

- `mem_tcam_lookup32_in_t`, `mem_tcam_lookup32_out_t`

3.22.5.137 mem_tcam_word_to_entry_index8

Prototype:

```
unsigned int mem_tcam_word_to_entry_index8(unsigned int word_idx)
```

Description:

Convert a TCAM 8 bit word index to an entry index.



Note

This function does not perform range checking.

Table 3.924. mem_tcam_word_to_entry_index8 parameters

Type	Name	Description
unsigned int	<i>word_idx</i>	A word index, as typically returned in a lookup result's first_match field

Returns:

An entry index which can be used with other entry indexed TCAM functions

See Also:

- `mem_tcam_lookup8_out_t`

3.22.5.138 mem_tcam_entry_value_to_word_index8

Prototype:

```
unsigned int mem_tcam_entry_value_to_word_index8(unsigned int entry_idx)
```

Description:

Convert a TCAM 8 bit entry value index to a word index.



Note

This function does not perform range checking.

Table 3.925. mem_tcam_entry_value_to_word_index8 parameters

Type	Name	Description
unsigned int	<i>entry_idx</i>	A entry index

Returns:

A word index to the word containing the value of entry number *entry_idx*.

3.22.5.139 mem_tcam_entry_mask_to_word_index8

Prototype:

```
unsigned int mem_tcam_entry_mask_to_word_index8(unsigned int entry_idx)
```

Description:

Convert a TCAM 8 bit entry mask index to a word index.



Note

This function does not perform range checking.

Table 3.926. mem_tcam_entry_mask_to_word_index8 parameters

Type	Name	Description
unsigned int	<i>entry_idx</i>	A entry index

Returns:

A word index to the word containing the mask of entry number *entry_idx*.

3.22.5.140 mem_tcam_word_to_entry_index16

Prototype:

```
unsigned int mem_tcam_word_to_entry_index16(unsigned int word_idx)
```

Description:

Convert a TCAM 16 bit word index to an entry index.



Note

This function does not perform range checking.

Table 3.927. mem_tcam_word_to_entry_index16 parameters

Type	Name	Description
unsigned int	<i>word_idx</i>	A word index, as typically returned in a lookup result's first_match field

Returns:

An entry index which can be used with other entry indexed TCAM functions

See Also:

- `mem_tcam_lookup16_out_t`

3.22.5.141 mem_tcam_entry_value_to_word_index16

Prototype:

```
unsigned int mem_tcam_entry_value_to_word_index16(unsigned int entry_idx)
```

Description:

Convert a TCAM 16 bit entry value index to a word index.



Note

This function does not perform range checking.

Table 3.928. mem_tcam_entry_value_to_word_index16 parameters

Type	Name	Description
unsigned int	<i>entry_idx</i>	A entry index

Returns:

A word index to the word containing the value of entry number entry_idx.

3.22.5.142 mem_tcam_entry_mask_to_word_index16

Prototype:

```
unsigned int mem_tcam_entry_mask_to_word_index16(unsigned int entry_idx)
```

Description:

Convert a TCAM 16 bit entry mask index to a word index.



Note

This function does not perform range checking.

Table 3.929. mem_tcam_entry_mask_to_word_index16 parameters

Type	Name	Description
unsigned int	<i>entry_idx</i>	A entry index

Returns:

A word index to the word containing the mask of entry number entry_idx.

3.22.5.143 mem_tcam_word_to_entry_index32

Prototype:

```
unsigned int mem_tcam_word_to_entry_index32(unsigned int word_idx)
```

Description:

Convert a TCAM 32 bit word index to an entry index.



Note

This function does not perform range checking.

Table 3.930. mem_tcam_word_to_entry_index32 parameters

Type	Name	Description
unsigned int	<i>word_idx</i>	A word index, as typically returned in a lookup result's first_match field

Returns:

An entry index which can be used with other entry indexed TCAM functions

See Also:

- `mem_tcam_lookup32_out_t`

3.22.5.144 mem_tcam_entry_value_to_word_index32

Prototype:

```
unsigned int mem_tcam_entry_value_to_word_index32(unsigned int entry_idx)
```

Description:

Convert a TCAM 32 bit entry value index to a word index.



Note

This function does not perform range checking.

Table 3.931. mem_tcam_entry_value_to_word_index32 parameters

Type	Name	Description
unsigned int	<i>entry_idx</i>	A entry index

Returns:

A word index to the word containing the value of entry number *entry_idx*.

3.22.5.145 mem_tcam_entry_mask_to_word_index32

Prototype:

```
unsigned int mem_tcam_entry_mask_to_word_index32(unsigned int entry_idx)
```

Description:

Convert a TCAM 32 bit entry mask index to a word index.



Note

This function does not perform range checking.

Table 3.932. mem_tcam_entry_mask_to_word_index32 parameters

Type	Name	Description
unsigned int	<i>entry_idx</i>	A entry index

Returns:

A word index to the word containing the mask of entry number *entry_idx*.

3.23 MEM Packet Engine Intrinsics

This section discusses the enumerations, structs, unions, types and functions related to memory unit packet engine operations.

3.23.1 MU PE Enumerations

3.23.1.1 MEM_PACKET_REWRITE_SCRIPT_OFFSET

Packet engine rewrite script offset.

Table 3.933. enum MEM_PACKET_REWRITE_SCRIPT_OFFSET

Name	Description
PACKET_ENGINE_SCRIPT_OFFSET_8	0 = 8.
PACKET_ENGINE_SCRIPT_OFFSET_16	1 = 16.
PACKET_ENGINE_SCRIPT_OFFSET_24	2 = 24.
PACKET_ENGINE_SCRIPT_OFFSET_32	3 = 32.
PACKET_ENGINE_SCRIPT_OFFSET_40	4 = 40.
PACKET_ENGINE_SCRIPT_OFFSET_48	5 = 48.
PACKET_ENGINE_SCRIPT_OFFSET_56	6 = 56.

3.23.1.2 MEM_PACKET_MASTER_BUCKET

Indicate the credit bucket of the packet engine for each master.

The ME bucket is maintained for all ME masters.

Table 3.934. enum MEM_PACKET_MASTER_BUCKET

Name	Description
PACKET_ENGINE_MASTER_ME	Credits of all MEs.
PACKET_ENGINE_MASTER_NBI_0	Credits of NBI-0.
PACKET_ENGINE_MASTER_NBI_1	Credits of NBI-1.

3.23.1.3 MEM_PACKET_LENGTH

Packet length for packet allocation.

Table 3.935. enum MEM_PACKET_LENGTH

Name	Description
PACKET_LENGTH_256B	256B packets are allocated at a 256B offset.
PACKET_LENGTH_512B	512B packets are allocated at a 512B offset.
PACKET_LENGTH_1KB	1KB packets are allocated at a 1KB offset.
PACKET_LENGTH_2KB	2KB packets are allocated at a 2KB offset.

3.23.1.4 MEM_PACKET_TRANSFER_TYPE

The transfer type for packets.

Table 3.936. enum MEM_PACKET_TRANSFER_TYPE

Name	Description
PACKET_TRANSFER_ADDRESS_MODE	Transfer type is address mode.
PACKET_TRANSER_PACKET_MODE	Transfer type is packet mode.

3.23.2 MU PE Structs

3.23.2.1 mem_packet_header_t

Packet header as Initial CTM transfer format.

Two 32 bit words.

Table 3.937. struct mem_packet_header_t

Type	Name	Description
unsigned int	ctm_number:6	CTM that contains the packet.
unsigned int	packet_number:10	Packet number used to address the packet in CTM.
unsigned int	buffer_list_queue:2	Buffer list queue associated with the MU pointer.
unsigned int	packet_length:14	Packet length in bytes.
unsigned int	value	Accessor to word_1 structure. Accessor to word_2 structure.
union mem_packet_header_t::@126	word_1	First word.
unsigned int	split:1	Indicates if the packet is split between CTM and a memory buffer.
unsigned int	reserved:2	Reserved.
unsigned int	mu_pointer:29	MU pointer to the memory buffer at 2KB boundaries.
union mem_packet_header_t::@127	word_2	Second word.

3.23.3 MU PE Unions

3.23.3.1 mem_packet_alloc_response_t

Type for response of mem_packet_alloc.

Table 3.938. union mem_packet_alloc_response_t

Type	Name	Description
unsigned int	reserved:2	Reserved.
unsigned int	packet_number:10	Last segment received.
unsigned int	packet_credit:11	Number of packets to be returned to the master.
unsigned int	buffer_credit:9	Number of buffers to be returned to the master.
unsigned int	value	Accessor to entire structure.

3.23.3.2 mem_packet_complete_request_t

Packet complete request.

Table 3.939. union mem_packet_complete_request_t

Type	Name	Description
unsigned int	reserved:4	Not used.
unsigned int	nbi_number:2	NBI number.
unsigned int	total_packet_length:14	Total packet length.
MEM_PACKET_REWRITE_SCRIPT_OFFSET	script_offset:14	Script offset.
unsigned int	sequencer_number:5	Sequencer number.
unsigned int	sequence_number:12	Sequence number.
unsigned int	tx_queue_number:10	Tx queue number.
unsigned int	retry_bit:1	Retry bit.
unsigned long long	value	Accessor to entire structure.

3.23.3.3 mem_packet_read_status_response_t

Response of mem_packet_read_packet_status.

Table 3.940. union mem_packet_read_status_response_t

Type	Name	Description
unsigned int	error:1	Error.
unsigned int	last_segment_received:1	Last segment received.
unsigned int	first_segment_received:1	First segment received.
unsigned int	sent_to_ME:1	Packet sent to ME.
unsigned int	packet_not_valid:1	Packet not valid, error.
unsigned int	packet_owned_by_ME:1	Packet owner = 0 or packet owner = 1 is packet owned by ME.
MEM_PACKET_MASTER_BUCKET	packet_owner:2	Packet owner .
unsigned int	reserved2:6	Reserved.
MEM_PACKET_LENGTH	packet_length:2	Packet size.
unsigned int	reserved:6	Reserved.
unsigned int	ctm_dcache_address_256B:10	CTM DCACHE address. Multiply by 256 to get actual CTM address.
unsigned int	value	Accessor to entire structure.

3.23.4 MU PE Typedefs

3.23.4.1 MEM_PACKET_REWRITE_SCRIPT_OFFSET

Packet engine rewrite script offset.

Table 3.941. typedef MEM_PACKET_REWRITE_SCRIPT_OFFSET

Type	Definition
MEM_PACKET_REWRITE_SCRIPT_OFFSET	enum MEM_PACKET_REWRITE_SCRIPT_OFFSET

3.23.4.2 MEM_PACKET_MASTER_BUCKET

Indicate the credit bucket of the packet engine for each master.

The ME bucket is maintained for all ME masters.

Table 3.942. typedef MEM_PACKET_MASTER_BUCKET

Type	Definition
MEM_PACKET_MASTER_BUCKET	enum MEM_PACKET_MASTER_BUCKET

3.23.4.3 MEM_PACKET_LENGTH

Packet length for packet allocation.

Table 3.943. typedef MEM_PACKET_LENGTH

Type	Definition
MEM_PACKET_LENGTH	enum MEM_PACKET_LENGTH

3.23.4.4 MEM_PACKET_TRANSFER_TYPE

The transfer type for packets.

Table 3.944. typedef MEM_PACKET_TRANSFER_TYPE

Type	Definition
MEM_PACKET_TRANSFER_TYPE	enum MEM_PACKET_TRANSFER_TYPE

3.23.4.5 mem_packet_read_status_response_t

Response of mem_packet_read_packet_status.

Table 3.945. `typedef mem_packet_read_status_response_t`

Type	Definition
<code>mem_packet_read_status_response_t</code>	<code>union mem_packet_read_status_response_t</code>

3.23.4.6 `mem_packet_read_status_response_in_read_reg_t`

Type for response of `mem_packet_read_packet_status` in read registers.

Table 3.946. `typedef mem_packet_read_status_response_in_read_reg_t`

Type	Definition
<code>mem_packet_read_status_response_in_read_reg_t</code>	<code>__xread mem_packet_read_status_response_t</code>

3.23.4.7 `mem_packet_alloc_response_t`

Type for response of `mem_packet_alloc`.

Table 3.947. `typedef mem_packet_alloc_response_t`

Type	Definition
<code>mem_packet_alloc_response_t</code>	<code>union mem_packet_alloc_response_t</code>

3.23.4.8 `mem_packet_alloc_response_in_read_reg_t`

Type for response of `mem_packet_alloc` in read registers.

Table 3.948. `typedef mem_packet_alloc_response_in_read_reg_t`

Type	Definition
<code>mem_packet_alloc_response_in_read_reg_t</code>	<code>__xread mem_packet_alloc_response_t</code>

3.23.4.9 `mem_packet_complete_request_t`

Packet complete request.

Table 3.949. `typedef mem_packet_complete_request_t`

Type	Definition
<code>mem_packet_complete_request_t</code>	<code>union mem_packet_complete_request_t</code>

3.23.4.10 `mem_packet_header_t`

Packet header as Initial CTM transfer format.

Two 32 bit words.

Table 3.950. `typedef mem_packet_header_t`

Type	Definition
<code>mem_packet_header_t</code>	<code>struct mem_packet_header_t</code>

3.23.5 MU PE Functions

3.23.5.1 `mem_packet_credit_get`

Prototype:

```
void mem_packet_credit_get(__xread void* data, enum MEM_PACKET_MASTER_BUCKET credit_bucket,
                           sync_t sync, SIGNAL* sig_ptr)
```

Description:

Return packet and buffer credits.

The response is sent irrespective of the availability of packets/memory or packet/buffer credits. The ref_count

Table 3.951. `mem_packet_credit_get` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Transfer register for packet credit response
<code>enum MEM_PACKET_MASTER_BUCKET</code>	<code>credit_bucket</code>	<ul style="list-style-type: none"> • 0 - ME • 1 - NBI0 • 2 - NBI1
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.23.5.2 `mem_packet_alloc`

Prototype:

```
void mem_packet_alloc(__xread mem_packet_alloc_response_t* data, enum
                      MEM_PACKET_MASTER_BUCKET credit_bucket, enum MEM_PACKET_LENGTH packet_length, sync_t
                      sync, SIGNAL* sig_ptr)
```

Description:

Packet allocation request.

If the packet cannot be allocated, the allocation request waits for a free packet.

Table 3.952. mem_packet_alloc parameters

Type	Name	Description
<code>__xread mem_packet_alloc_response_t*</code>	<code>data</code>	Transfer register for packet allocation response
<code>enum MEM_PACKET_MASTER_BUCKET</code>	<code>credit_bucket</code>	<ul style="list-style-type: none"> • 0 - ME • 1 - NBI0 • 2 - NBI1
<code>enum MEM_PACKET_LENGTH</code>	<code>packet_length</code>	<ul style="list-style-type: none"> • 0 - 256B • 1 - 512B • 2 - 1KB • 3 - 2KB
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.23.5.3 mem_packet_alloc_poll

Prototype:

```
void mem_packet_alloc_poll(__xread mem_packet_alloc_response_t* data, enum
MEM_PACKET_MASTER_BUCKET credit_bucket, enum MEM_PACKET_LENGTH packet_length, sync_t
sync, SIGNAL* sig_ptr)
```

Description:

Packet allocation poll request.

If the packet cannot be allocated, the response is sent immediately indicating that the packet cannot be allocated. The response contains all 1's in the data field.

The following example shows `mem_packet_alloc_poll()` request with `mem_packet_read_packet_status()`.

```
mem_packet_alloc_response_in_read_reg_t alloc_response;
MEM_PACKET_MASTER_BUCKET bucket_master = PACKET_ENGINE_MASTER_NBI_1;
unsigned int pkt_number;
SIGNAL sig;

// allocate two packets
mem_packet_alloc_poll(&alloc_response, bucket_master, PACKET_LENGTH_1KB, ctx_swap, &sig);
mem_packet_alloc_poll(&alloc_response, bucket_master, PACKET_LENGTH_1KB, ctx_swap, &sig);

pkt_number = alloc_response.packet_number; // packet number of second allocated packet

// read and verify the packet status of second allocated packet
{
    mem_packet_read_status_response_in_read_reg_t packet_status;
```

```

mem_packet_read_packet_status(&packet_status, pkt_number, ctx_swap, &sig);

if (packet_status.packet_length != PACKET_LENGTH_1KB)
{
    return 0;           // We have an error
}
if (packet_status.packet_owner != bucket_master)
{
    return 0;           // We have an error
}
if (packet_status.error != 0)
{
    return 0;           // We have an error
}
if (packet_status.packet_not_valid != 0)
{
    return 0;           // We have an error
}
}

return 1;

```

Table 3.953. mem_packet_alloc_poll parameters

Type	Name	Description
<code>__xread mem_packet_alloc_response_t*</code>	<code>data</code>	Transfer register for packet allocation response
<code>enum MEM_PACKET_MASTER_BUCKET</code>	<code>credit_bucket</code>	<ul style="list-style-type: none"> • 0 - ME • 1 - NBI0 • 2 - NBI1
<code>enum MEM_PACKET_LENGTH</code>	<code>packet_length</code>	<ul style="list-style-type: none"> • 0 - 256B • 1 - 512B • 2 - 1KB • 3 - 2KB
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.23.5.4 mem_packet_free

Prototype:

```
void mem_packet_free(unsigned int packet_number)
```

Description:

Packet free request.

Free the packet

Table 3.954. mem_packet_free parameters

Type	Name	Description
unsigned int	<i>packet_number</i>	Packet number to free

3.23.5.5 mem_packet_free_and_return_pointer

Prototype:

```
void mem_packet_free_and_return_pointer(__xread void* data, unsigned int packet_number,
sync_t sync, SIGNAL* sig_ptr)
```

Description:

Packet free and return pointer request.

Free Packet and push MU Pointer of the freed packet.

Table 3.955. mem_packet_free_and_return_pointer parameters

Type	Name	Description
__xread void*	<i>data</i>	Transfer register for packet free response
unsigned int	<i>packet_number</i>	Packet number to be freed
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.23.5.6 mem_packet_free_and_signal

Prototype:

```
void mem_packet_free_and_signal(unsigned int packet_number, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Packet free and signal request.

Free Packet and signal when packet is freed.

Table 3.956. mem_packet_free_and_signal parameters

Type	Name	Description
unsigned int	<i>packet_number</i>	Packet number to be freed
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.23.5.7 mem_packet_return_pointer

Prototype:

```
void mem_packet_return_pointer(__xread void* data, unsigned int packet_number, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Packet return pointer request.

Return the MU Pointer in the packet header

Table 3.957. mem_packet_return_pointer parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Transfer register for packet MU pointer
<code>unsigned int</code>	<code>packet_number</code>	Packet number to be returned
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.23.5.8 mem_packet_complete_drop

Prototype:

```
void mem_packet_complete_drop(mem_packet_complete_request_t packet_complete, unsigned int packet_number)
```

Description:

Packet processing complete.,

Used for sequence number only.

Table 3.958. mem_packet_complete_drop parameters

Type	Name	Description
<code>mem_packet_complete_request_t</code>	<code>packet_complete</code>	Data for the packet complete command.
<code>unsigned int</code>	<code>packet_number</code>	Packet number of packet complete.

Availability:

NFP-6xxx Indirect Reference Mode

3.23.5.9 mem_packet_complete_unicast

Prototype:

```
void mem_packet_complete_unicast(mem_packet_complete_request_t packet_complete, unsigned
int packet_number)
```

Description:

Packet processing complete for unicast packet.

Packet processing complete and free packet on last transfer.

Table 3.959. mem_packet_complete_unicast parameters

Type	Name	Description
mem_packet_complete_request_t	<i>packet_complete</i>	Data for the packet complete command.
unsigned int	<i>packet_number</i>	Packet number of packet complete.

Availability:

NFP-6xxx Indirect Reference Mode

3.23.5.10 mem_packet_complete_multicast

Prototype:

```
void mem_packet_complete_multicast(mem_packet_complete_request_t packet_complete, unsigned
int packet_number)
```

Description:

Packet processing complete for multicast packet.

Packet processing complete but do not free packet on last transfer.

Table 3.960. mem_packet_complete_multicast parameters

Type	Name	Description
mem_packet_complete_request_t	<i>packet_complete</i>	Data for the packet complete command.
unsigned int	<i>packet_number</i>	Packet number of packet complete.

Availability:

NFP-6xxx Indirect Reference Mode

3.23.5.11 mem_packet_complete_multicast_free

Prototype:

```
void mem_packet_complete_multicast_free(mem_packet_complete_request_t packet_complete,
unsigned int packet_number)
```

Description:

Packet processing complete for multicast packet.

Packet processing complete and free packet on last transfer.

Table 3.961. mem_packet_complete_multicast_free parameters

Type	Name	Description
mem_packet_complete_request_t	<i>packet_complete</i>	Data for the packet complete command.
unsigned int	<i>packet_number</i>	Packet number of packet complete.

Availability:

NFP-6xxx Indirect Reference Mode

3.23.5.12 mem_packet_read_packet_status

Prototype:

```
void mem_packet_read_packet_status(__xread mem_packet_read_status_response_t* data,
unsigned int packet_number, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Packet read packet status.

Read the status of a packet. The status is returned regardless of the status of the packet. If the read packet status FIFO is full; the response contains all f's.

Table 3.962. mem_packet_read_packet_status parameters

Type	Name	Description
__xread mem_packet_read_status_response_t*	<i>data</i>	Transfer register for packet read status response
unsigned int	<i>packet_number</i>	Packet number to read status of
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.23.5.13 mem_packet_wait_packet_status

Prototype:

```
void mem_packet_wait_packet_status(__xread mem_packet_read_status_response_t* data,
unsigned int packet_number, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Packet wait packet status.

Read the status of a packet. The status is returned after the last segment of the packet has arrived. If the read packet status FIFO is full; the response contains all f's.

Table 3.963. mem_packet_wait_packet_status parameters

Type	Name	Description
<code>__xread mem_packet_read_status_response_t*</code>	<code>data</code>	Transfer register for packet read status response
<code>unsigned int</code>	<code>packet_number</code>	Packet number to read status of
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.23.5.14 mem_packet_add_thread

Prototype:

```
void mem_packet_add_thread(__xread void* data, unsigned int packet_offset, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Add ME thread to Packet engine work queue.

Add an ME thread to the Packet Engine work queue for incoming packets. If the ME adds itself to the Work Queue with a length '7' and offset '3', it receives the first 6 32-bit words in 3 Push cycles. Next it receives 2 32-bit words starting at offset 3, at the dataRef it specified.

Table 3.964. mem_packet_add_thread parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Transfer register for packet allocation response
<code>unsigned int</code>	<code>packet_offset</code>	Offset to receive from after the first 6 32-bit words at 4 byte granularity.
<code>unsigned int</code>	<code>count</code>	Length in words of data to receive (valid values 6 -31)
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.23.5.15 mem_pe_dma_to_memory_packet

Prototype:

```
void mem_pe_dma_to_memory_packet(volatile void __addr40 __mem* address, unsigned int packet_offset, unsigned int packet_number, unsigned int ctm_island, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Packet engine DMA to memory using packet mode.

Move CTM data to main memory using packet mode. The packet number is specified as the source. LoadTimeConstant() can be used to populate ctm_island address correctly, i.e. LoadTimeConstant("__ADDR_I34_CTM") for CTM on i34.

The following shows an example of using mem_pe_dma_to_memory_packet():

```

__emem_n(1) __addr40 unsigned long long          emem_address[6];
mem_packet_alloc_response_in_read_reg_t        alloc_response;
unsigned int                                     pkt_number;
SIGNAL                                           sig;
unsigned int                                     packet_offset = 0;

// allocate a packet
mem_packet_alloc_poll
(
    &alloc_response,
    PACKET_ENGINE_MASTER_NBI_1,
    PACKET_LENGTH_1KB,
    ctx_swap,
    &sig
);
pkt_number = alloc_response.packet_number;

// write to CTM in packet mode
{
    __ctm unsigned int                  ctm_packet;
    __xwrite unsigned long long        write_data[2];

    write_data[0] = 0x1122334455667788;
    write_data[1] = 0x9900aabbccddeeff;
    ctm_packet = (1 << 31) | (unsigned int)(pkt_number << 16) | (packet_offset);
    mem_write64
    (
        (void*)write_data,
        (volatile void __ctm*)ctm_packet,
        2,
        ctx_swap,
        &sig
    );
}
mem_pe_dma_to_memory_packet(emem_address, packet_offset, pkt_number, 0, ctx_swap, &sig);

// verify data in emem1
{
    __xread unsigned long long        read_data[2];

    mem_read64_ptr40((void*)read_data, emem_address, 2, ctx_swap, &sig);

    if (read_data[0] != 0x1122334455667788)
    {
        return 0;                // We have an error
    }
    if (read_data[1] != 0x9900aabbccddeeff)
}

```

```

    {
        return 0;           // We have an error
    }
}

return 1;

```

Table 3.965. mem_pe_dma_to_memory_packet parameters

Type	Name	Description
volatile void __addr40 __mem*	address	40 bit address of external memory including the MU island. Must be 8B aligned.
unsigned int	packet_offset	8B aligned offset with packet data to start DMA from.
unsigned int	packet_number	Packet number of packet to write.
unsigned int	ctm_island	Address of the CTM to execute instruction. Can be 0 for local island. See 6xxx databook for recommended addressing mode.
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

Availability:

NFP-6xxx Indirect Reference Mode

3.23.5.16 mem_pe_dma_to_memory_packet_free

Prototype:

```
void mem_pe_dma_to_memory_packet_free(volatile void __addr40 __mem* address, unsigned
int packet_offset, unsigned int packet_number, unsigned int ctm_island, sync_t sync,
SIGNAL* sig_ptr)
```

Description:

Packet engine DMA to memory using packet mode.

Free packet.

Move CTM data to external memory using packet mode. The packet number is specified as the source. Free packet after DMA completes.



Note

LoadTimeConstant() can be used to populate ctm_island address correctly, i.e.
LoadTimeConstant("__ADDR_I34_CTM") for CTM on i34.

Table 3.966. mem_pe_dma_to_memory_packet_free parameters

Type	Name	Description
volatile void __addr40 __mem*	<i>address</i>	40 bit address of external memory including the MU island. Must be 8B aligned.
unsigned int	<i>packet_offset</i>	8B aligned offset with packet data to start DMA from.
unsigned int	<i>packet_number</i>	Packet number of packet to write.
unsigned int	<i>ctm_island</i>	Address of the CTM to execute instruction. Can be 0 for local island. See 6xxx databook for recommended addressing mode.
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

Availability:

NFP-6xxx Indirect Reference Mode

3.23.5.17 mem_pe_dma_to_memory_packet_swap

Prototype:

```
void mem_pe_dma_to_memory_packet_swap(volatile void __addr40 __mem* address, unsigned int packet_offset, unsigned int packet_number, unsigned int ctm_island, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Packet engine DMA to memory using packet mode.

Byte swapped transferred data.

Move CTM data to external memory using packet mode. The packet number is specified as the source. Byte swap the transferred data.



Note

LoadTimeConstant() can be used to populate ctm_island address correctly, i.e.
LoadTimeConstant("__ADDR_I34_CTM") for CTM on i34.

Table 3.967. mem_pe_dma_to_memory_packet_swap parameters

Type	Name	Description
volatile void __addr40 __mem*	<i>address</i>	40 bit address of external memory including the MU island. Must be 8B aligned.
unsigned int	<i>packet_offset</i>	8B aligned offset with packet data to start DMA from.
unsigned int	<i>packet_number</i>	Packet number of packet to write.

Type	Name	Description
unsigned int	<i>ctm_island</i>	Address of the CTM to execute instruction. Can be 0 for local island. See 6xxx databook for recommended addressing mode.
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

Availability:

NFP-6xxx Indirect Reference Mode

3.23.5.18 mem_pe_dma_to_memory_packet_free_swap

Prototype:

```
void mem_pe_dma_to_memory_packet_free_swap(volatile void __addr40 __mem* address, unsigned int packet_offset, unsigned int packet_number, unsigned int ctm_island, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Packet engine DMA to memory using packet mode.

Byte swap the transferred data. Free packet.

Move CTM data to external memory using packet mode. The packet number is specified as the source. Byte swap the transferred data. Free packet after DMA completes.



Note

LoadTimeConstant() can be used to populate *ctm_island* address correctly, i.e.
LoadTimeConstant("__ADDR_I34_CTM") for CTM on i34.

Table 3.968. mem_pe_dma_to_memory_packet_free_swap parameters

Type	Name	Description
volatile void __addr40 __mem*	<i>address</i>	40 bit address of external memory including the MU island. Must be 8B aligned.
unsigned int	<i>packet_offset</i>	8B aligned offset with packet data to start DMA from.
unsigned int	<i>packet_number</i>	Packet number of packet to write.
unsigned int	<i>ctm_island</i>	Address of the CTM to execute instruction. Can be 0 for local island. See 6xxx databook for recommended addressing mode.
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

Availability:

NFP-6xxx Indirect Reference Mode

3.23.5.19 mem_pe_dma_to_memory_buffer

Prototype:

```
void mem_pe_dma_to_memory_buffer(volatile void __ctm* source_address, volatile void
__addr40 __mem* destination_address, unsigned int ctm_island, unsigned int count, sync_t
sync, SIGNAL* sig_ptr)
```

Description:

Packet engine DMA to memory using address mode.

Move CTM data to external memory using address mode. The CTM address is specified as the source.



Note

LoadTimeConstant() can be used to populate ctm_island address correctly, i.e.
LoadTimeConstant("__ADDR_I34_CTM") for CTM on i34.

Table 3.969. mem_pe_dma_to_memory_buffer parameters

Type	Name	Description
volatile void __ctm*	source_address	32 bit source address of CTM (must be 8B aligned)
volatile void __addr40 __mem*	destination_address	40 bit address of external memory including island and locality for the DMA command.
unsigned int	ctm_island	Address of the CTM to execute instruction. Can be 0 for local island. See 6xxx databook for recommended addressing mode.
unsigned int	count	64B to 2048B where valid values (1-32)
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

Availability:

NFP-6xxx Indirect Reference Mode

3.23.5.20 mem_pe_dma_to_memory_buffer_swap

Prototype:

```
void mem_pe_dma_to_memory_buffer_swap(volatile void __ctm* source_address, volatile void
__addr40 __mem* destination_address, unsigned int ctm_island, unsigned int count, sync_t
sync, SIGNAL* sig_ptr)
```

Description:

Packet engine DMA to memory using address mode with byte swap.

Move CTM data to external memory using address mode with byte swap. The CTM address is specified as the source.



Note

LoadTimeConstant() can be used to populate ctm_island address correctly, i.e.
LoadTimeConstant("__ADDR_I34_CTM") for CTM on i34.

Table 3.970. mem_pe_dma_to_memory_buffer_swap parameters

Type	Name	Description
volatile void __ctm*	source_address	32 bit source address of CTM (must be 8B aligned)
volatile void __addr40 __mem*	destination_address	40 bit address of external memory including island and locality for the DMA command.
unsigned int	ctm_island	Address of the CTM to execute instruction. Can be 0 for local island. See 6xxx databook for recommended addressing mode.
unsigned int	count	64B to 2048B where valid values (1-32)
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

Availability:

NFP-6xxx Indirect Reference Mode

3.23.5.21 mem_pe_dma_to_memory_buffer_le

Prototype:

```
void mem_pe_dma_to_memory_buffer_le(volatile void __ctm* source_address, volatile void __addr40 __mem* destination_address, unsigned int ctm_island, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Packet engine DMA to memory using address mode.

Use little_endian addressing.

Move CTM data to external memory using address mode with little endian addressing. The CTM address is specified as the source.



Note

LoadTimeConstant() can be used to populate ctm_island address correctly, i.e.
LoadTimeConstant("__ADDR_I34_CTM") for CTM on i34.

Table 3.971. mem_pe_dma_to_memory_buffer_le parameters

Type	Name	Description
volatile void __ctm*	source_address	32 bit source address of CTM (must be 8B aligned)
volatile void __addr40 __mem*	destination_address	40 bit address of external memory including island and locality for the DMA command.
unsigned int	ctm_island	Address of the CTM to execute instruction. Can be 0 for local island. See 6xxx databook for recommended addressing mode.
unsigned int	count	64B to 2048B where valid values (1-32)
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

Availability:

NFP-6xxx Indirect Reference Mode

3.23.5.22 mem_pe_dma_to_memory_buffer_le_swap

Prototype:

```
void mem_pe_dma_to_memory_buffer_le_swap(volatile void __ctm* source_address, volatile
void __addr40 __mem* destination_address, unsigned int ctm_island, unsigned int count,
sync_t sync, SIGNAL* sig_ptr)
```

Description:

Packet engine DMA to memory using address mode with byte swap.

Use little_endian addressing.

Move CTM data to external memory using address mode with byte swap and little endian addressing. The CTM address is specified as the source.



Note

LoadTimeConstant() can be used to populate ctm_island address correctly, i.e.
LoadTimeConstant("__ADDR_I34_CTM") for CTM on i34.

Table 3.972. mem_pe_dma_to_memory_buffer_le_swap parameters

Type	Name	Description
volatile void __ctm*	<i>source_address</i>	32 bit source address of CTM (must be 8B aligned)
volatile void __addr40 __mem*	<i>destination_address</i>	40 bit address of external memory including island and locality for the DMA command.
unsigned int	<i>ctm_island</i>	Address of the CTM to execute instruction. Can be 0 for local island. See 6xxx databook for recommended addressing mode.
unsigned int	<i>count</i>	64B to 2048B where valid values (1-32)
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

Availability:

NFP-6xxx Indirect Reference Mode

3.23.5.23 mem_pe_dma_from_memory_buffer

Prototype:

```
void mem_pe_dma_from_memory_buffer(volatile void __addr40 __mem* source_address, volatile
void __ctm* destination_address, unsigned int ctm_island, unsigned int count, sync_t
sync, SIGNAL* sig_ptr)
```

Description:

Packet engine DMA from memory using address mode.

Move data from external memory to CTM. The CTM address is specified as the destination address.



Note

LoadTimeConstant() can be used to populate *ctm_island* address correctly, i.e.
LoadTimeConstant("__ADDR_I34_CTM") for CTM on i34.

Table 3.973. mem_pe_dma_from_memory_buffer parameters

Type	Name	Description
volatile void __addr40 __mem*	<i>source_address</i>	40 bit address of external memory including island and locality for the DMA command.
volatile void __ctm*	<i>destination_address</i>	32 bit destination address of CTM (must be 8B aligned)
unsigned int	<i>ctm_island</i>	Address of the CTM to execute instruction. Can be 0 for local island. See 6xxx databook for recommended addressing mode.

Type	Name	Description
unsigned int	<i>count</i>	64B to 2048B where valid values (1-32)
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

Availability:

NFP-6xxx Indirect Reference Mode

3.23.5.24 mem_pe_dma_from_memory_buffer_swap

Prototype:

```
void mem_pe_dma_from_memory_buffer_swap(volatile void __addr40 __mem* source_address,
volatile void __ctm* destination_address, unsigned int ctm_island, unsigned int count,
sync_t sync, SIGNAL* sig_ptr)
```

Description:

Packet engine DMA from memory using address mode with byte swap.

Move data from external memory to CTM. The CTM address is specified as the destination address.



Note

LoadTimeConstant() can be used to populate ctm_island address correctly, i.e.
LoadTimeConstant("__ADDR_I34_CTM") for CTM on i34.

Table 3.974. mem_pe_dma_from_memory_buffer_swap parameters

Type	Name	Description
volatile void __addr40 __mem*	<i>source_address</i>	40 bit address of external memory including island and locality for the DMA command.
volatile void __ctm*	<i>destination_address</i>	32 bit destination address of CTM (must be 8B aligned)
unsigned int	<i>ctm_island</i>	Address of the CTM to execute instruction. Can be 0 for local island. See 6xxx databook for recommended addressing mode.
unsigned int	<i>count</i>	64B to 2048B where valid values (1-32)
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

Availability:

NFP-6xxx Indirect Reference Mode

3.23.5.25 mem_pe_dma_from_memory_buffer_le

Prototype:

```
void mem_pe_dma_from_memory_buffer_le(volatile void __addr40 __mem* source_address,  
volatile void __ctm* destination_address, unsigned int ctm_island, unsigned int count,  
sync_t sync, SIGNAL* sig_ptr)
```

Description:

Packet engine DMA from memory using address mode.

Use little endian addressing.

Move data from external memory to CTM. The CTM address is specified as the destination address.



Note

LoadTimeConstant() can be used to populate ctm_island address correctly, i.e.
LoadTimeConstant("__ADDR_I34_CTM") for CTM on i34.

Table 3.975. mem_pe_dma_from_memory_buffer_le parameters

Type	Name	Description
volatile void __addr40 __mem*	source_address	40 bit address of external memory including island and locality for the DMA command.
volatile void __ctm*	destination_address	32 bit destination address of CTM (must be 8B aligned)
unsigned int	ctm_island	Address of the CTM to execute instruction. Can be 0 for local island. See 6xxx databook for recommended addressing mode.
unsigned int	count	64B to 2048B where valid values (1-32)
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

Availability:

NFP-6xxx Indirect Reference Mode

3.23.5.26 mem_pe_dma_from_memory_buffer_le_swap

Prototype:

```
void mem_pe_dma_from_memory_buffer_le_swap(volatile void __addr40 __mem* source_address,  
volatile void __ctm* destination_address, unsigned int ctm_island, unsigned int count,  
sync_t sync, SIGNAL* sig_ptr)
```

Description:

Packet engine DMA from memory using address mode with byte swap.

Use little endian addressing.

Move data from external memory to CTM. The CTM address is specified as the destination address.



Note

LoadTimeConstant() can be used to populate ctm_island address correctly, i.e.
LoadTimeConstant("__ADDR_I34_CTM") for CTM on i34.

Table 3.976. mem_pe_dma_from_memory_buffer_le_swap parameters

Type	Name	Description
volatile void __addr40 __mem*	<i>source_address</i>	40 bit address of external memory including island and locality for the DMA command.
volatile void __ctm*	<i>destination_address</i>	32 bit destination address of CTM (must be 8B aligned)
unsigned int	<i>ctm_island</i>	Address of the CTM to execute instruction. Can be 0 for local island. See 6xxx databook for recommended addressing mode.
unsigned int	<i>count</i>	64B to 2048B where valid values (1-32)
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

Availability:

NFP-6xxx Indirect Reference Mode

3.23.5.27 mem_pe_dma_to_memory_indirect

Prototype:

```
void mem_pe_dma_to_memory_indirect(__xread unsigned int* data, unsigned int packet_number,
sync_t sync, SIGNAL* sig_ptr)
```

Description:

Packet engine DMA to memory in packet mode.

Write data from CTM packet to memory buffer. The packet should be setup with a header as per initial CTM transfer format. The packet length and memory address are used from this header to copy data from the CTM packet to memory.



Note

See union `mem_packet_header_t` to setup the header of ME allocated packet.

Table 3.977. mem_pe_dma_to_memory_indirect parameters

Type	Name	Description
<code>_xread unsigned int*</code>	<code>data</code>	Transfer register with read result
<code>unsigned int</code>	<code>packet_number</code>	Packet number of packet to write
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.23.5.28 mem_pe_dma_to_memory_indirect_swap

Prototype:

```
void mem_pe_dma_to_memory_indirect_swap(_xread unsigned int* data, unsigned int
packet_number, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Packet engine DMA to memory in packet mode with byte swap.

Write data from CTM packet to memory buffer with byte swap. The packet should be setup with a header as per initial CTM transfer format. The packet length and memory address are used from this header to copy data from the CTM packet to memory.



Note

See union `mem_packet_header_t` to setup the header of ME allocated packet.

Table 3.978. mem_pe_dma_to_memory_indirect_swap parameters

Type	Name	Description
<code>_xread unsigned int*</code>	<code>data</code>	Transfer register with read result
<code>unsigned int</code>	<code>packet_number</code>	Packet number of packet to write
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.23.5.29 mem_pe_dma_to_memory_indirect_free

Prototype:

```
void mem_pe_dma_to_memory_indirect_free(_xread unsigned int* data, unsigned int
packet_number, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Packet engine DMA to memory in packet mode.

Free the packet after DMA completes.

Write data from CTM packet to memory buffer and free packet after DMA completes. The packet should be setup with a header as per initial CTM transfer format. The packet length and memory address are used from this header to copy data from the CTM packet to memory.



Note

See union `mem_packet_header_t` to setup the header of ME allocated packet.

Table 3.979. mem_pe_dma_to_memory_indirect_free parameters

Type	Name	Description
<code>_xread unsigned int*</code>	<code>data</code>	Transfer register with read result
<code>unsigned int</code>	<code>packet_number</code>	Packet number of packet to write
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.23.5.30 mem_pe_dma_to_memory_indirect_free_swap

Prototype:

```
void mem_pe_dma_to_memory_indirect_free_swap(_xread unsigned int* data, unsigned int
packet_number, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Packet engine DMA to memory in packet mode with byte swap.

Free the packet after DMA completes.

Write data from CTM packet to memory buffer with byte swap and free packet after DMA completes. The packet should have a header as per initial CTM transfer format. The packet length and memory address are used from this header to copy data from the CTM packet to memory.



Note

See union `mem_packet_header_t` to setup the header of ME allocated packet.

Table 3.980. mem_pe_dma_to_memory_indirect_free_swap parameters

Type	Name	Description
<code>_xread unsigned int*</code>	<code>data</code>	Transfer register with read result

Type	Name	Description
unsigned int	<i>packet_number</i>	Packet number of packet to write
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.24 MEM Statistics Intrinsics

This section discusses the enumerations, structs, unions, types and functions related to MEM Statistics operations.

Statistics engine functions are available on internal memory, island 28 or island 29.

Statistics are cleared with mem_stats_read_and_clear() and read with mem_stats_read(). Statistics are logged with mem_stats_log(), mem_stats_log_event(), mem_stats_log_saturate() or mem_stats_log_event_saturate(). The statistics can be in packed or unpacked format (mem_stats_unpacked_address_detail_t) or (mem_stats_packed_address_detail_t).

Use helper macros MEM_STATS_EXTRACT_WRAP_PKT_AND_BYTE_COUNT() to extract the packet and byte count or MEM_STATS_EXTRACT_SATURATE_PKT_AND_BYTE_COUNT() to also extract the saturation flag for packet and byte count.

Below is a complete example with statistics on island 28 (imem0) and the statistics (unpacked) are cleared, logged and read to verify values.

```

mem_stats_read_command_address_format_t      read_stats;
SIGNAL                           sig;
unsigned int                      count = 4;
unsigned int                      i;
unsigned long long    island = (unsigned long long)LoadTimeConstant("___ADDR_I28_IMEM");

read_stats.value = 0;
read_stats.base_address_select = BASE_ADDRESS_1;
read_stats.statistic_address = 0x00;                                // address 0x00

// read and clear stats
{
    // address for reading stats is made up from:
    //   - island id ([39:32]) and
    //   - mem_stats_read_command_address_format_t for lower 32-bits
    unsigned long long  stats_address = island | read_stats.value;
    __declspec(read_reg) unsigned int read_register[8];

    mem_stats_read_and_clear(
        (void *)read_register,
        (__declspec(mem, addr40) void *)stats_address,
        count,
        ctx_swap,
        &sig
    );
}

```

```
// log stats
{
    mem_stats_log_command_address_format_t log_stats;

    log_stats.value = 0;
    log_stats.add_byte_count_value = 0x11;
    log_stats.address_packed_config = STATS_ALL_32_BIT_UNPACKED;

    {
        volatile __declspec(local_mem) mem_stats_unpacked_address_detail_t stats_addr;

        // address for logging stats is made up from:
        //   - island id ([39:32]) and
        //   - mem_stats_log_command_address_format_t for lower 32-bits
        unsigned long long stat_index_address = island | log_stats.value;
        __declspec(write_reg) unsigned int wr_reg[4];

        stats_addr.value = 0;
        stats_addr.base_address_select = BASE_ADDRESS_1;

        stats_addr.statistic_address = 0;
        wr_reg[0] = stats_addr.value;

        stats_addr.statistic_address = 1;
        wr_reg[1] = stats_addr.value;

        stats_addr.statistic_address = 2;
        wr_reg[2] = stats_addr.value;

        stats_addr.statistic_address = 3;
        wr_reg[3] = stats_addr.value;

        // increment statistics for all 4 addresses
        mem_stats_log(
            (void *)wr_reg,
            (__declspec(mem, addr40) void *) (stat_index_address),
            count*2,
            ctx_swap,
            &sig
        );
    }

    // increment statistics only for first address
    mem_stats_log(
        (void *)&wr_reg,
        (__declspec(mem, addr40) void *) (stat_index_address),
        2,
        ctx_swap,
        &sig
    );
}

/* read statistics and verify content */
{
    unsigned long long stats_address = island | read_stats.value;
    __declspec(read_reg) unsigned int rd_reg[8];

    mem_stats_read(
        (void *)rd_reg,
```

```
(__declspec(mem, addr40) void *) stats_address,  
3,  
ctx_swap,  
&sig  
  
// first statistics should be two packets and 0x11*2 byte count  
{  
    unsigned long long stats0 = ((unsigned long long) rd_reg[1] << 32) | rd_reg[0];  
    unsigned long long byte_cnt;  
    unsigned int      pkt_cnt;  
  
    MEM_STATS_EXTRACT_WRAP_PKT_AND_BYTE_COUNT(stats0, pkt_cnt, byte_cnt);  
  
    if (byte_cnt != 0x22)  
    {  
        return 0;          // We have an error  
    }  
  
    if (pkt_cnt != 0x2)  
    {  
        return 0;          // We have an error  
    }  
}  
  
// second statistics should be one packet and 0x11 byte count  
{  
    unsigned long long stats1 = ((unsigned long long) rd_reg[3] << 32) | rd_reg[2];  
    unsigned long long byte_cnt;  
    unsigned int      pkt_cnt;  
  
    MEM_STATS_EXTRACT_WRAP_PKT_AND_BYTE_COUNT(stats1, pkt_cnt, byte_cnt);  
  
    if (byte_cnt != 0x11)  
    {  
        return 0;          // We have an error  
    }  
  
    if (pkt_cnt != 0x1)  
    {  
        return 0;          // We have an error  
    }  
}  
}  
  
return 1;
```

3.24.1 MU Statistics Defines

Table 3.981. MU Statistics Defines

Defined	Definition	
MEM_STATS_WRAP_PACKET_COUNT(value)	(value >> 35) Helper macro to extract only the packet count (29 bits) from 64-bit value after reading statistics in wrapping statistic format.	
MEM_STATS_WRAP_BYTE_COUNT(value)	(value & 0x7fffffff)	Helper macro to extract only the byte count (35 bits) from 64-bit value after reading statistics in wrapping statistic format.
MEM_STATS_EXTRACT_WRAP_PKT_AND_BYTE_COUNT	(value, pkt_cnt, byte_cnt) \ { \ byte_cnt = MEM_STATS_WRAP_BYTE_COUNT(value); \ pkt_cnt = MEM_STATS_WRAP_PACKET_COUNT(value); \ }	Helper macro to extract the packet count (29 bits) and byte count (35 bits) from 64-bit value after reading statistics in wrapping statistic format.
MEM_STATS_SATURATE_PACKET_COUNT(value)	((value >> 35) & 0x7fffffff)	Helper macro to extract only the packet count (28 bits) from 64-bit value after reading statistics in saturating statistic format.
MEM_STATS_SATURATE_BYTE_COUNT(value)	(value & 0x3fffffff)	Helper macro to extract only the byte count (34 bits) from 64-bit value after reading statistics in saturating statistic format.
MEM_STATS_SATURATE_PACKET_COUNT_SATURATE_IND(value)	(value >> 63)	Helper macro to extract only the saturate packet count indicator (1 bit) from 64-bit value after reading statistics in saturating statistic format.

Defined	Definition
<code>MEM_STATS_SATURATE_BYTE_COUNT_SATURATE_IND(value)</code>	<pre>((value >> 34) & 0x01)</pre> <p>Helper macro to extract only the saturate byte count indicator (1 bit) from 64-bit value after reading statistics in saturating statistic format.</p>
<code>MEM_STATS_EXTRACT_SATURATE_PKT_AND_BYTE_COUNT</code>	<pre>(value, pkt_cnt, byte_cnt, pkt_sat_flag, byte_sat_flag) \ { byte_cnt = MEM_STATS_SATURATE_BYTE_COUNT(value); \ pkt_cnt = MEM_STATS_SATURATE_PACKET_COUNT(value); \ pkt_sat_flag = MEM_STATS_SATURATE_PACKET_COUNT_SATURATE_IND(value); \ byte_sat_flag = MEM_STATS_SATURATE_BYTE_COUNT_SATURATE_IND(value); \ }</pre> <p>Helper macro to extract the packet count (28 bits) and byte count (34 bits) as well as saturation indicators from the 64-bit value after reading statistics in saturate statistic format.</p>

3.24.2 MU Statistics Enumerations

3.24.2.1 MEM_STATS_BASE_ADDRESS_SELECT

Base address CSR select (MUSEBaseAddr).

Table 3.982. enum MEM_STATS_BASE_ADDRESS_SELECT

Name	Description
<code>BASE_ADDRESS_0</code>	Base address 0.
<code>BASE_ADDRESS_1</code>	Base address 1.
<code>BASE_ADDRESS_2</code>	Base address 2.
<code>BASE_ADDRESS_3</code>	Base address 3.

3.24.2.2 MEM_STATS_ADDRESS_PACK_CONFIG

Address packing configuration.

Table 3.983. enum MEM_STATS_ADDRESS_PACK_CONFIG

Name	Description
STATS_ALL_16_BIT_PACKED	All addresses are 16bit packed addresses.
STATS_ALL_32_BIT_UNPACKED	All addresses are 32bit unpacked addresses.
STATS_FIRST_32_BIT_UNPACKED_ONLY	First address is 32bit unpacked and remaining addresses are 16bit packed.

3.24.3 MU Statistics Unions

3.24.3.1 mem_stats_log_command_address_format_t

Log statistic command address field.

Table 3.984. union mem_stats_log_command_address_format_t

Type	Name	Description
unsigned int	reserved:14	Reserved.
MEM_STATS_ADDRESS_PACK_CONFIG	address_packed_config:2	How the pulled addresses are packed into the transfer registers.
unsigned int	add_byte_count_value:16	Value to add to the byte count statistics field.
unsigned int	value	Accessor to entire descriptor structure.

3.24.3.2 mem_stats_packed_address_detail_t

Statistic packed address detail (16 bit).

Table 3.985. union mem_stats_packed_address_detail_t

Type	Name	Description
MEM_STATS_BASE_ADDRESS_SELECT	base_address_select:2	Select base address for statistics address.
unsigned int	statistic_address:14	Statistic address (64b aligned).
unsigned short	value	Accessor to entire descriptor structure.

3.24.3.3 mem_stats_read_command_address_format_t

Read (push) statistic command address field.

Table 3.986. union mem_stats_read_command_address_format_t

Type	Name	Description
MEM_STATS_BASE_ADDRESS_SELECT	base_address_select:2	Select base address for statistics address.

Type	Name	Description
unsigned int	reserved_2:8	Reserved.
unsigned int	statistic_address:19	Statistic address (64b aligned).
unsigned int	reserved_1:3	Reserved.
unsigned int	value	Accessor to entire descriptor structure.

3.24.3.4 mem_stats_unpacked_address_detail_t

Statistic unpacked address detail (32 bit).

Table 3.987. union mem_stats_unpacked_address_detail_t

Type	Name	Description
MEM_STATS_BASE_ADDRESS_SELECT	base_address_select:2	Select base address for statistics address.
unsigned int	reserved:11	Reserved.
unsigned int	statistic_address:19	Statistic address (64b aligned).
unsigned int	value	Accessor to entire descriptor structure.

3.24.4 MU Statistics Typedefs

3.24.4.1 MEM_STATS_BASE_ADDRESS_SELECT

Base address CSR select (MUSEBaseAddr).

Table 3.988. typedef MEM_STATS_BASE_ADDRESS_SELECT

Type	Definition
MEM_STATS_BASE_ADDRESS_SELECT	enum MEM_STATS_BASE_ADDRESS_SELECT

3.24.4.2 MEM_STATS_ADDRESS_PACK_CONFIG

Address packing configuration.

Table 3.989. typedef MEM_STATS_ADDRESS_PACK_CONFIG

Type	Definition
MEM_STATS_ADDRESS_PACK_CONFIG	enum MEM_STATS_ADDRESS_PACK_CONFIG

3.24.4.3 mem_stats_log_command_address_format_t

Log statistic command address field.

Table 3.990. `typedef mem_stats_log_command_address_format_t`

Type	Definition
<code>mem_stats_log_command_address_format_t</code>	<code>union mem_stats_log_command_address_format_t</code>

3.24.4.4 `mem_stats_read_command_address_format_t`

Read (push) statistic command address field.

Table 3.991. `typedef mem_stats_read_command_address_format_t`

Type	Definition
<code>mem_stats_read_command_address_format_t</code>	<code>union mem_stats_read_command_address_format_t</code>

3.24.4.5 `mem_stats_packed_address_detail_t`

Statistic packed address detail (16 bit).

Table 3.992. `typedef mem_stats_packed_address_detail_t`

Type	Definition
<code>mem_stats_packed_address_detail_t</code>	<code>union mem_stats_packed_address_detail_t</code>

3.24.4.6 `mem_stats_unpacked_address_detail_t`

Statistic unpacked address detail (32 bit).

Table 3.993. `typedef mem_stats_unpacked_address_detail_t`

Type	Definition
<code>mem_stats_unpacked_address_detail_t</code>	<code>union mem_stats_unpacked_address_detail_t</code>

3.24.5 MU Statistics Functions

3.24.5.1 `mem_stats_read`

Prototype:

```
void mem_stats_read(__xread void* xfer, volatile void __addr40 __mem* address, unsigned
int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Retrieve the statistic data from the data cache.

Table 3.994. mem_stats_read parameters

Type	Name	Description
<code>__xread void*</code>	<code>xfer</code>	Transfer registers containing statistics data
<code>volatile void __addr40 __mem*</code>	<code>address</code>	40 bit pointer containing mu island in upper 8 bits (see 6xxx databook for recommended addressing mode) and <code>mem_stats_read_command_address_format_t</code> for the lower 32 bits.
<code>unsigned int</code>	<code>count</code>	Length in 64-bits to read (valid values 1-16).
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (<code>sig_done</code> or <code>ctx_swap</code>).
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion.

3.24.5.2 mem_stats_read_and_clear

Prototype:

```
void mem_stats_read_and_clear(__xread void* xfer, volatile void __addr40 __mem* address,
                           unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Retrieve the statistic data from the data cache and also clear the the statistics data on read.

Table 3.995. mem_stats_read_and_clear parameters

Type	Name	Description
<code>__xread void*</code>	<code>xfer</code>	Transfer registers containing statistics data.
<code>volatile void __addr40 __mem*</code>	<code>address</code>	40 bit pointer containing mu island in upper 8 bits (see 6xxx databook for recommended addressing mode) and <code>mem_stats_read_command_address_format_t</code> for the lower 32 bits.
<code>unsigned int</code>	<code>count</code>	Length in 64-bits to read (valid values 1-16).
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (<code>sig_done</code> or <code>ctx_swap</code>).
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion.

3.24.5.3 mem_stats_log

Prototype:

```
void mem_stats_log(__xwrite void* xfer, volatile void __addr40 __mem* data, unsigned int
                   count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Log statistics data by adding byte count and incrementing packet count to specified statistics.

A single command can support a maximum of sixteen statistic updates, where each statistic data is updated with the same byte count add value. Stats format is wrapping format.

Table 3.996. mem_stats_log parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>xfer</code>	Transfer registers containing statistic addresses. Use either <code>mem_stats_unpacked_address_detail_t</code> or <code>mem_stats_packed_address_detail_t</code>
<code>volatile void __addr40 __mem*</code>	<code>data</code>	40 bit pointer containing the mu island in the 8 upper bits (see 6xxx databook for recommended addressing mode) and <code>mem_stats_log_command_address_format_t</code> for the lower 32 bits
<code>unsigned int</code>	<code>count</code>	Number of statistic updates in 16-bit statistic address pointers to pull (valid values 1-16)
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.24.5.4 mem_stats_log_saturate

Prototype:

```
void mem_stats_log_saturate(__xwrite void* xfer, volatile void __addr40 __mem* data,
                           unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Log statistics data with saturation by adding byte count and incrementing packet count to specified statistics.

A single command can support a maximum of sixteen statistic updates, where each statistic data is updated with the same byte count add value. Saturating statistics will continue to update after the saturation point is reached and the saturate indicator will indicate the counter reached saturation point. Saturation indicator will remain set until command `stats_push_clear` is performed to the same address.

Table 3.997. mem_stats_log_saturate parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>xfer</code>	Transfer registers containing statistic addresses. Use either <code>mem_stats_unpacked_address_detail_t</code> or <code>mem_stats_packed_address_detail_t</code>
<code>volatile void __addr40 __mem*</code>	<code>data</code>	40 bit pointer containing the mu island in the 8 upper bits (see 6xxx databook for recommended addressing mode) and <code>mem_stats_log_command_address_format_t</code> for the lower 32 bits
<code>unsigned int</code>	<code>count</code>	Number of statistic updates in 16-bit statistic address pointers to pull (valid values 1-16)
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.24.5.5 mem_stats_log_event

Prototype:

```
void mem_stats_log_event(__xwrite void* xfer, volatile void __addr40 __mem* data, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Log statistics data and also supports packet and byte counter half-wrap (event type = 3) and full-wrap (event type = 2) thresholds.

Same as:

Table 3.998. mem_stats_log_event parameters

Type	Name	Description
__xwrite void*	xfer	Transfer registers containing statistic addresses. Use either mem_stats_unpacke_address_detail_t or mem_stats_packed_address_detail_t
volatile void __addr40 __mem*	data	40 bit pointer containing the mu island in the 8 upper bits (see 6xxx databook for recommended addressing mode) and mem_stats_log_command_address_format_t for the lower 32 bits
unsigned int	count	Number of statistic updates in 16-bit statistic address pointers to pull (valid values 1- 16)
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

See Also:

- mem_stats_log but also raises an event on half wrap or full wrap levels. Note that the continued counting will re-trigger half-wrap and full-wrap events when reached if the stat has not been cleared.

3.24.5.6 mem_stats_log_event_saturate

Prototype:

```
void mem_stats_log_event_saturate(__xwrite void* xfer, volatile void __addr40 __mem* data, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Log statistics data with saturation and also supports packet and byte counter half-wrap (event type = 3) and full-wrap (event type = 2) thresholds.

Same as:

Table 3.999. mem_stats_log_event_saturate parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>xfer</code>	Transfer registers containing statistic addresses. Use either <code>mem_stats_unpacked_address_detail_t</code> or <code>mem_stats_packed_address_detail_t</code>
<code>volatile void __addr40 __mem*</code>	<code>data</code>	40 bit pointer containing the mu island in the 8 upper bits (see 6xxx databook for recommended addressing mode) and <code>mem_stats_log_command_address_format_t</code> for the lower 32 bits
<code>unsigned int</code>	<code>count</code>	Number of statistic updates in 16-bit statistic address pointers to pull (valid values 1 - 16)
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

See Also:

- `mem_stats_log_saturate` but also raises an event on half wrap or full wrap levels. Note that the continued counting will re-trigger half-wrap and full-wrap events when reached if the stat has not been cleared.

3.25 MEM Load Balancing Intrinsics

This section discusses the enumerations, structs, unions, types and functions related to MEM Load Balancing operations.

3.25.1 MU LB Defines

Table 3.1000. MU LB Defines

Defined	Definition
<code>MEM_LB_HASH_SEED_HIGH(seed)</code>	<code>((seed >> 13) & 0x7ff);</code> Helper macro to extract the hash_seed_high (higher 11 bits) from 24-bit hash seed as needed for load balance descriptor data (<code>mem_lb_desc_t</code>).
<code>MEM_LB_HASH_SEED_LOW(seed)</code>	<code>(seed & 0x1fff);</code> Helper macro to extract the hash_seed_low (lower 13 bits) from 24-bit hash seed as needed for load balance descriptor data (<code>mem_lb_desc_t</code>).

Defined	Definition
MEM_LB_HASH_BASE_ADDR_HIGH(addr)	((addr >> 13) & 0x1f); Helper macro to extract the hash_base_addr_high (higher 5 bits) from 18-bit hash base address as needed for load balance descriptor data (mem_lb_desc_t).
MEM_LB_HASH_BASE_ADDR_LOW(addr)	(addr & 0xffff); Helper macro to extract the hash_base_addr_low (lower 13 bits) from 18-bit hash base address as needed for load balance descriptor data (mem_lb_desc_t).
MEM_LB_HASH_BASE_ADDR_FROM_DESC(desc)	(desc.hash_base_addr_high << 13) desc.hash_base_addr_low; Helper macro to concatenate hash base address from load balance descriptor data (mem_lb_desc_t). The hash_base_addr_high and hash_base_addr_low fields are concatenated for the hash base address.
MEM_LB_HASH_SEED_FROM_DESC(desc)	(desc.hash_seed_high << 13) desc.hash_seed_low; Helper macro to concatenate hash seed from load balance descriptor data (mem_lb_desc_t). The hash_seed_high and hash_seed_low fields are concatenated for the hash seed.

3.25.2 MU LB Unions

3.25.2.1 mem_lb_bucket_dcache_address_format_t

Lower 32 bit of Dcache Hash Bucket address detail.

Table 3.1001. union mem_lb_bucket_dcache_address_format_t

Type	Name	Description
unsigned int	reserved_2:10	Reserved.
unsigned int	bucket_address:18	Dcache hash bucket address in hash table array.
unsigned int	reserved_1:4	Reserved.
unsigned int	value	Accessor to entire bucket address structure.

3.25.2.2 mem_lb_bucket_local_address_format_t

Lower 32 bit of Local Hash Bucket address detail.

Table 3.1002. union mem_lb_bucket_local_address_format_t

Type	Name	Description
unsigned int	reserved_2:19	Reserved.
unsigned int	bucket_address:9	Local hash bucket address in hash table array.
unsigned int	reserved_1:4	Reserved.
unsigned int	value	Accessor to entire bucket address structure.

3.25.2.3 mem_lb_dcache_stats_address_format_t

Lower 32 bit of read statistic command address of dcache stat array.

Table 3.1003. union mem_lb_dcache_stats_address_format_t

Type	Name	Description
unsigned int	reserved_2:10	Reserved.
unsigned int	statistic_address:19	Statistic address (64b address).
unsigned int	reserved_1:3	Reserved.
unsigned int	value	Accessor to entire descriptor structure.

3.25.2.4 mem_lb_descriptor_address_format_t

Lower 32 bit of local descriptor address detail.

Table 3.1004. union mem_lb_descriptor_address_format_t

Type	Name	Description
unsigned int	reserved:26	Reserved.
unsigned int	array_entry:6	Descriptor array entry location.
unsigned int	value	Accessor to entire descriptor structure.

3.25.2.5 mem_lb_desc_t

Load balance descriptor detail.

Table 3.1005. union mem_lb_desc_t

Type	Name	Description
unsigned int	hash_base_addr_low:13	Upper 13 bits of 18 bit hash base address.
unsigned int	stat_addr_ref:1	Location reference for StatAddr (0-local mem,1-DCache).
unsigned int	stat_base_addr:18	Statistics base address.
unsigned int	hash_seed_low:13	Lower 13 bits of 24 bit hash seed value.
unsigned int	boundaries:3	Number of boundaries in the hash bucket.

Type	Name	Description
unsigned int	index_bits:5	Number of bits of hash to use as address offset.
unsigned int	compare_bits:5	Number of bits of hash to use for compare.
unsigned int	hash_addr_ref:1	Location reference for HashAddr (0-local mem,1-DCache).
unsigned int	hash_base_addr_high:5	Upper 5 bits of 18 bit hash base address.
unsigned int	reserved_1:21	Reserved.
unsigned int	hash_seed_high:11	Upper 11 bits of 24 bit hash seed value.
unsigned int	reserved_2:32	Reserved.
unsigned int	user_data_1:32	User data first 32-bit word.
unsigned int	user_data_2:32	User data second 32-bit word.
unsigned int	value[6]	Accessor to entire descriptor structure.

3.25.2.6 mem_lb_id_table_address_format_t

Lower 32 bit of Local ID table address detail.

This is used when writing to the local id table.

Table 3.1006. union mem_lb_id_table_address_format_t

Type	Name	Description
unsigned int	reserved_2:21	Reserved.
unsigned int	id_table_address:8	Local ID table array address.
unsigned int	reserved_1:3	Reserved.
unsigned int	value	Accessor to entire id table address structure.

3.25.2.7 mem_lb_local_stats_address_format_t

Lower 32 bit of read statistic command address of local stat array.

Table 3.1007. union mem_lb_local_stats_address_format_t

Type	Name	Description
unsigned int	reserved_2:19	Reserved.
unsigned int	statistic_address:10	Statistic address (64b address).
unsigned int	reserved_1:3	Reserved.
unsigned int	value	Accessor to entire descriptor structure.

3.25.2.8 mem_lb_lookup_bundle_id_result_t

Lookup data bundle_id result detail.

The bundle id and user data are returned.

Table 3.1008. union mem_lb_lookup_bundle_id_result_t

Type	Name	Description
unsigned int	reserved_1:24	Reserved.
unsigned int	bundle_id:8	Bundle id.
unsigned int	reserved_2:32	Reserved.
unsigned int	user_data_1:32	User_data_1.
unsigned int	user_data_2:32	User data_2.
unsigned int	value[4]	Accessor to entire lookup detail structure.

3.25.2.9 mem_lb_lookup_command_format_t

Lookup data detail.

Table 3.1009. union mem_lb_lookup_command_format_t

Type	Name	Description
unsigned int	reserved_1:8	Reserved.
unsigned int	hash_input:24	Initial hash input.
unsigned int	reserved_2:16	Reserved.
unsigned int	stat_add_value:16	Value to add to statistic.
unsigned int	padding[2]	Padding required for xfer to be used together with result lookup.
unsigned int	value[4]	Accessor to entire lookup detail structure.

3.25.2.10 mem_lb_lookup_direct_result_t

Lookup direct result detail.

The lookup result and user data are returned.

Table 3.1010. union mem_lb_lookup_direct_result_t

Type	Name	Description
unsigned int	lo_result:32	Bottom 32 bits of id table lookup result.
unsigned int	hi_result:32	Top 32 bits of id table lookup result.
unsigned int	user_data_1:32	User_data_1.
unsigned int	user_data_2:32	User data_2.
unsigned int	value[4]	Accessor to entire lookup detail structure.

3.25.3 MU LB Typedefs

3.25.3.1 MEM_LB_LOCATION

Location reference for HashAddr or StatAddr.

Table 3.1011. typedef MEM_LB_LOCATION

Type	Definition
MEM_LB_LOCATION	enum MEM_LB_LOCATION

3.25.3.2 mem_lb_desc_t

Load balance descriptor detail.

Table 3.1012. typedef mem_lb_desc_t

Type	Definition
mem_lb_desc_t	union mem_lb_desc_t

3.25.3.3 mem_lb_desc_in_write_reg_t

descriptor in write_reg

Table 3.1013. typedef mem_lb_desc_in_write_reg_t

Type	Definition
mem_lb_desc_in_write_reg_t	<code>__xwrite mem_lb_desc_t</code>

3.25.3.4 mem_lb_desc_in_read_reg_t

descriptor in write_reg

Table 3.1014. typedef mem_lb_desc_in_read_reg_t

Type	Definition
mem_lb_desc_in_read_reg_t	<code>__xread mem_lb_desc_t</code>

3.25.3.5 mem_lb_local_stats_address_format_t

Lower 32 bit of read statistic command address of local stat array.

Table 3.1015. `typedef mem_lb_local_stats_address_format_t`

Type	Definition
<code>mem_lb_local_stats_address_format_t</code>	union <code>mem_lb_local_stats_address_format_t</code>

3.25.3.6 `mem_lb_dcache_stats_address_format_t`

Lower 32 bit of read statistic command address of dcache stat array.

Table 3.1016. `typedef mem_lb_dcache_stats_address_format_t`

Type	Definition
<code>mem_lb_dcache_stats_address_format_t</code>	union <code>mem_lb_dcache_stats_address_format_t</code>

3.25.3.7 `mem_lb_descriptor_address_format_t`

Lower 32 bit of local descriptor address detail.

Table 3.1017. `typedef mem_lb_descriptor_address_format_t`

Type	Definition
<code>mem_lb_descriptor_address_format_t</code>	union <code>mem_lb_descriptor_address_format_t</code>

3.25.3.8 `mem_lb_id_table_address_format_t`

Lower 32 bit of Local ID table address detail.

This is used when writing to the local id table.

Table 3.1018. `typedef mem_lb_id_table_address_format_t`

Type	Definition
<code>mem_lb_id_table_address_format_t</code>	union <code>mem_lb_id_table_address_format_t</code>

3.25.3.9 `mem_lb_bucket_local_address_format_t`

Lower 32 bit of Local Hash Bucket address detail.

Table 3.1019. `typedef mem_lb_bucket_local_address_format_t`

Type	Definition
<code>mem_lb_bucket_local_address_format_t</code>	union <code>mem_lb_bucket_local_address_format_t</code>

3.25.3.10 `mem_lb_bucket_dcache_address_format_t`

Lower 32 bit of Dcache Hash Bucket address detail.

Table 3.1020. `typedef mem_lb_bucket_dcache_address_format_t`

Type	Definition
<code>mem_lb_bucket_dcache_address_format_t</code>	<code>union mem_lb_bucket_dcache_address_format_t</code>

3.25.3.11 `mem_lb_lookup_command_format_t`

Lookup data detail.

Table 3.1021. `typedef mem_lb_lookup_command_format_t`

Type	Definition
<code>mem_lb_lookup_command_format_t</code>	<code>union mem_lb_lookup_command_format_t</code>

3.25.3.12 `mem_lb_lookup_command_format_in_write_reg_t`

lookup command in write_reg

Table 3.1022. `typedef mem_lb_lookup_command_format_in_write_reg_t`

Type	Definition
<code>mem_lb_lookup_command_format_in_write_reg_t</code>	<code>__xwrite mem_lb_lookup_command_format_t</code>

3.25.3.13 `mem_lb_lookup_bundle_id_result_t`

Lookup data bundle_id result detail.

The bundle id and user data are returned.

Table 3.1023. `typedef mem_lb_lookup_bundle_id_result_t`

Type	Definition
<code>mem_lb_lookup_bundle_id_result_t</code>	<code>union mem_lb_lookup_bundle_id_result_t</code>

3.25.3.14 `mem_lb_lookup_bundle_id_result_in_read_reg_t`

lookup result in read_reg

Table 3.1024. `typedef mem_lb_lookup_bundle_id_result_in_read_reg_t`

Type	Definition
<code>mem_lb_lookup_bundle_id_result_in_read_reg_t</code>	<code>__xread mem_lb_lookup_bundle_id_result_t</code>

3.25.3.15 `mem_lb_lookup_direct_result_t`

Lookup direct result detail.

The lookup result and user data are returned.

Table 3.1025. `typedef mem_lb_lookup_direct_result_t`

Type	Definition
<code>mem_lb_lookup_direct_result_t</code>	<code>union mem_lb_lookup_direct_result_t</code>

3.25.3.16 `mem_lb_lookup_direct_result_in_read_reg_t`

lookup result in read_reg

Table 3.1026. `typedef mem_lb_lookup_direct_result_in_read_reg_t`

Type	Definition
<code>mem_lb_lookup_direct_result_in_read_reg_t</code>	<code>__xread mem_lb_lookup_direct_result_t</code>

3.25.4 MU LB Functions

3.25.4.1 `mem_lb_write_desc`

Prototype:

```
void mem_lb_write_desc(mem_lb_desc_in_write_reg_t* xfer, volatile void __addr40 __mem*
address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write descriptor data into the Local Descriptor Array.

Table 3.1027. `mem_lb_write_desc` parameters

Type	Name	Description
<code>mem_lb_desc_in_write_reg_t*</code>	<code>xfer</code>	Transfer registers containing descriptor data.
<code>volatile void __addr40 __mem*</code>	<code>address</code>	40 bit pointer with array entry location (<code>mem_lb_descriptor_address_format_t</code>) in lower 32 bits See 6xxx databook for recommended addressing mode for higher 8 bits of address parameter.
<code>unsigned int</code>	<code>count</code>	Length in 64 bit words to write, calculated as <code>count * 3</code> (valid values 1- 4).
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (<code>sig_done</code> or <code>ctx_swap</code>).
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion. * Load descriptor data into the Local Descriptor Array. The array supports up to 64 Descriptors. Descriptors must be loaded into the Local Descriptor Array prior to doing lookups.

3.25.4.2 mem_lb_read_desc

Prototype:

```
void mem_lb_read_desc(mem_lb_desc_in_read_reg_t* xfer, volatile void __addr40 __mem* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read descriptor data from the Local Descriptor Array.

Read descriptor data from the Local Descriptor Array. The array supports up to 64 Descriptors. Descriptors must be loaded into the Local Descriptor Array prior to doing lookups.

Table 3.1028. mem_lb_read_desc parameters

Type	Name	Description
mem_lb_desc_in_read_reg_t*	xfer	Transfer registers containing descriptor data
volatile void __addr40 __mem*	address	40 bit pointer with array entry location (mem_lb_descriptor_address_format_t) in lower 32 bits See 6xxx databook for recommended addressing mode for higher 8 bits of address parameter.
unsigned int	count	Length in 64 bit words to read, calculated as count * 3 (valid values 1- 4).
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

3.25.4.3 mem_lb_write_id_table

Prototype:

```
void mem_lb_write_id_table(__xwrite_void* xfer, volatile void __addr40 __mem* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write entries to the Local ID Table Array.

Write entries to the Local ID Table Array. The array supports up to 256 entries.

Table 3.1029. mem_lb_write_id_table parameters

Type	Name	Description
__xwrite_void*	xfer	Transfer registers containing entries to write.
volatile void __addr40 __mem*	address	40 bit pointer with array entry location (mem_lb_id_table_address_format_t) in lower 32 bits. See 6xxx

Type	Name	Description
		databook for recommended addressing mode for higher 8 bits of address parameter.
unsigned int	count	Length in 64 bit words to write (valid values 1 - 16).
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

3.25.4.4 mem_lb_read_id_table

Prototype:

```
void mem_lb_read_id_table(__xread void* xfer, volatile void __addr40 __mem* address,
unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read entries from the Local ID Table Array.

Read entries from the Local ID Table Array. The array supports up to 256 entries.

Table 3.1030. mem_lb_read_id_table parameters

Type	Name	Description
__xread void*	xfer	Transfer registers containing entries read.
volatile void __addr40 __mem*	address	40 bit pointer with array entry location (mem_lb_id_table_address_format_t) in lower 32 bits. See 6xxx databook for recommended addressing mode for higher 8 bits of address parameter.
unsigned int	count	Length in 64 bit words to read (valid values 1 - 16).
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap).
SIGNAL*	sig_ptr	Signal to raise upon completion.

3.25.4.5 mem_lb_bucket_write_local

Prototype:

```
void mem_lb_bucket_write_local(__xwrite void* xfer, volatile void __addr40 __mem* address,
unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write entries to the Local Hash Table Bucket Array.

Write entries to the Local Hash Table Bucket Array. The array supports up to 512 entries with each entry 128 bits in width.

Table 3.1031. mem_lb_bucket_write_local parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>xfer</code>	Transfer registers containing entries to write.
<code>volatile void __addr40 __mem*</code>	<code>address</code>	40 bit pointer with bucket address (<code>mem_lb_bucket_local_address_format_t</code>) in lower 32 bits. See 6xxx databook for recommended addressing mode for higher 8 bits of address parameter.
<code>unsigned int</code>	<code>count</code>	Length in 64 bit words to write, calculated as count * 2 (valid values 1 - 8).
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (<code>sig_done</code> or <code>ctx_swap</code>).
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion.

3.25.4.6 mem_lb_bucket_read_local

Prototype:

```
void mem_lb_bucket_read_local(__xread void* xfer, volatile void __addr40 __mem* address,
unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read entries from the Local Hash Table Array.

Read entries from the Local Hash Table Bucket Array. The array supports up to 512 entries with each entry 128 bits in width.

Table 3.1032. mem_lb_bucket_read_local parameters

Type	Name	Description
<code>__xread void*</code>	<code>xfer</code>	Transfer registers containing descriptor data.
<code>volatile void __addr40 __mem*</code>	<code>address</code>	40 bit pointer with bucket address (<code>mem_lb_bucket_local_address_format_t</code>) in lower 32 bits. See 6xxx databook for recommended addressing mode for higher 8 bits of address parameter.
<code>unsigned int</code>	<code>count</code>	Length in 64 bit words to read calculated as count * 2 (valid values 1 - 8).
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (<code>sig_done</code> or <code>ctx_swap</code>).
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion.

3.25.4.7 mem_lb_bucket_write_dcache

Prototype:

```
void mem_lb_bucket_write_dcache(__xwrite void* xfer, volatile void __addr40 __mem* address,
unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write entries to the DCache Hash Table Bucket Array.

Write entries to the DCache Hash Table Bucket Array. The array supports up to 512 entries with each entry 128 bits in width.

Table 3.1033. mem_lb_bucket_write_dcache parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>xfer</code>	Transfer registers containing entries to write.
<code>volatile void __addr40 __mem*</code>	<code>address</code>	40 bit pointer with bucket address (<code>mem_lb_bucket_dcache_address_format_t</code>) in lower 32 bits. See 6xxx databook for recommended addressing mode for higher 8 bits of address parameter.
<code>unsigned int</code>	<code>count</code>	Length in 64 bit words to read calculated as count * 2 (valid values 1 - 8).
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (<code>sig_done</code> or <code>ctx_swap</code>).
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion.

3.25.4.8 mem_lb_bucket_read_dcache

Prototype:

```
void mem_lb_bucket_read_dcache(__xread void* xfer, volatile void __addr40 __mem* address,
unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read entries from the DCache Hash Table Bucket Array.

Read entries from the DCache Hash Table Bucket Array. The array supports up to 512 entries with each entry 128 bits in width.

Table 3.1034. mem_lb_bucket_read_dcache parameters

Type	Name	Description
<code>__xread void*</code>	<code>xfer</code>	Transfer registers containing descriptor data.
<code>volatile void __addr40 __mem*</code>	<code>address</code>	40 bit pointer with bucket address (<code>mem_lb_bucket_dcache_address_format_t</code>) in lower 32 bits. See 6xxx databook for recommended addressing mode for higher 8 bits of address parameter.
<code>unsigned int</code>	<code>count</code>	Length in 64 bit words to read calculated as count * 2 (valid values 1 - 8).
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (<code>sig_done</code> or <code>ctx_swap</code>).
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion.

3.25.4.9 mem_lb_stats_read_and_clear_local

Prototype:

```
void mem_lb_stats_read_and_clear_local(__xread void* xfer, volatile void __addr40 __mem* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read statistics data from the Local Stats Table Array.

Also clear the statistics after read.

Read and return stats data from the Local Stats Table Array. Write zeros into same location to clear stats data.

Table 3.1035. mem_lb_stats_read_and_clear_local parameters

Type	Name	Description
__xread void*	xfer	Transfer registers containing statistics data.
volatile void __addr40 __mem*	address	40 bit pointer with bucket address (mem_lb_bucket_local_address_format_t) in lower 32 bits. See 6xxx databook for recommended addressing mode for higher 8 bits of address parameter.
unsigned int	count	Length in 64 bit words to read (valid values 1 - 16).
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

3.25.4.10 mem_lb_stats_read_local

Prototype:

```
void mem_lb_stats_read_local(__xread void* xfer, volatile void __addr40 __mem* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read statistics data from the Local Stats Table Array.

Read and return stats data from the Local Stats Table Array.

Table 3.1036. mem_lb_stats_read_local parameters

Type	Name	Description
__xread void*	xfer	Transfer registers containing statistics data.
volatile void __addr40 __mem*	address	40 bit pointer with bucket address (mem_lb_bucket_local_address_format_t) in lower 32 bits. See 6xxx databook for recommended addressing mode for higher 8 bits of address parameter.
unsigned int	count	Length in 64 bit words to read (valid values 1 - 16).
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion.

3.25.4.11 mem_lb_stats_read_and_clear_dcache

Prototype:

```
void mem_lb_stats_read_and_clear_dcache(__xread void* xfer, volatile void __addr40 __mem*
address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read statistics data from the designated location in DCache.

Also clear the statistics after read.

Read and return stats data from DCache. Write zeros into same location to clear stats data.

Table 3.1037. mem_lb_stats_read_and_clear_dcache parameters

Type	Name	Description
__xread void*	xfer	Transfer registers containing statistics data (mem_lb_read_stats_address_format_t).
volatile void __addr40 __mem*	address	40 bit pointer with bucket address (mem_lb_dcache_stats_address_format_t) in lower 32 bits. See 6xxx databook for recommended addressing mode for higher 8 bits of address parameter.
unsigned int	count	Length in 64 bit words to read (valid values 1 - 16).
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap).
SIGNAL*	sig_ptr	Signal to raise upon completion.

3.25.4.12 mem_lb_stats_read_dcache

Prototype:

```
void mem_lb_stats_read_dcache(__xread void* xfer, volatile void __addr40 __mem* address,
unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read statistics data from the designated location in DCache.

Read and return stats data from DCache.

Table 3.1038. mem_lb_stats_read_dcache parameters

Type	Name	Description
__xread void*	xfer	Transfer registers containing statistics data (mem_lb_read_stats_address_format_t).

Type	Name	Description
volatile void __addr40 __mem*	address	40 bit pointer with bucket address (mem_lb_dcache_stats_address_format_t) in lower 32 bits. See 6xxx databook for recommended addressing mode for higher 8 bits of address parameter.
unsigned int	count	Length in 64 bit words to read (valid values 1 - 16).
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion.

3.25.4.13 mem_lb_lookup_bundle_id

Prototype:

```
mem_lb_lookup_bundle_id_result_in_read_reg_t*
mem_lb_lookup_bundle_id(mem_lb_lookup_command_format_in_write_reg_t* xfer, volatile void
__addr40 __mem* address, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Lookup bundle id using has input and update statistics.

Perform a comparison lookup on 24-bit initial hash input and return bundleId. Add 16-bit stat input data to the statistics counter. Pull one 64-bit word of Hash and Stat, push two 64-bit words as result. This is also called indirect lookup.

Below is an example of bundle id lookup on MU island 29 (imem1). Steps: 1. Initialise 0x0f to load balancing hash multiply CSR. Hash SBox is left at default. 2. Initialise descriptor array (3) to Boundaries=3, CompareBits=24. 3. Write to the hash bucket table. 4. Do lookup with hash input.

```

SIGNAL          sig;
unsigned long long mu_island = (unsigned long long)LoadTimeConstant("___ADDR_I29_IMEM");
unsigned int     descriptor_array = 3;
unsigned int     count = 1;
unsigned int     write_stat_base_addr = 0x100;
unsigned int     write_hash_base_addr = 0x00;
unsigned int     write_user_data_1 = 0x9abcdeff;
unsigned int     write_user_data_2 = 0x12345678;
unsigned int     write_hash_seed = 0xaaaaaaaa;

// Write 0x0f to load balancing hash multiply CSR
{
    cluster_target_xpb_address_format_t      xpb_command;
    __xwrite unsigned int      wr_xfer;
    xpb_command.value = 0;
    xpb_command.device = 0x20;
    xpb_command.target_island = 29;
    xpb_command.global_xpb = 1;

    wr_xfer = 0x0f;
    cluster_target_xpb_write((void *)&wr_xfer, &xpb_command, 1, ctx_swap, &sig);
}
```

```

// Initialise load balancing descriptor array 3
mem_lb_init
(
    descriptor_array,
    mu_island >> 32,
    write_stat_base_addr,
    LB_LOCATION_LOCAL_MEM,
    write_hash_base_addr,
    LB_LOCATION_LOCAL_MEM,
    3,           // boundaries
    0,           // index_bits
    24,          // compare_bits
    write_hash_seed,
    ((unsigned long long)write_user_data_1 << 32) | write_user_data_2,
    count
);

// Write hash bucket
// Refer to Table 9.51 "HashBucket - Boundaries=3, CompareBits=24" in 6xxx databook
{
    mem_lb_bucket_local_address_format_t      hb_addr;
    volatile __xwrite unsigned int write_xfer[4];
    __xread unsigned int          read_xfer[4];
    unsigned int                  hb_entry = 0;
    unsigned int                  bnd0 = 0x900000,
                                  bnd1 = 0xa00000,
                                  bnd2 = 0xb00000;

    write_xfer[3] = bnd2 >> 16;
    write_xfer[2] = (bnd2 << 16) | (bnd1 >> 8);
    write_xfer[1] = (bnd1 << 24) | bnd0;
    write_xfer[0] = (3 << 24) | (2 << 16) | (1 << 8) | 0; // regions

    hb_addr.value = 0;
    hb_addr.bucket_address = hb_entry;

    {
        unsigned long long addr = mu_island |
            (unsigned long long)(write_hash_base_addr | hb_addr.value);

        mem_lb_bucket_write_local(
            (void *)write_xfer,
            (__addr40 __mem void *) addr,
            1,
            ctx_swap,
            &sig
        );
        mem_lb_bucket_read_local(
            (void *)read_xfer,
            (__addr40 __mem void *) addr,
            1,
            ctx_swap,
            &sig
        );
    }
}

// Do lookup

```

```

{
    SIGNAL_PAIR                      sig_pair;
    mem_lb_lookup_bundle_id_result_in_read_reg_t *result;
    mem_lb_lookup_command_format_in_write_reg_t  write_data;
    unsigned long long                 addr = (mu_island | descriptor_array);

    write_data.value[0] = 0;
    write_data.value[1] = 0;
    write_data.stat_add_value = 0x2233;
    write_data.hash_input = 0x200000;

    result = mem_lb_lookup_bundle_id(
        &write_data,
        (__addr40 __mem void *)addr,
        sig_done,
        &sig_pair
    );
    wait_for_all_single(&sig_pair.odd);

    if (result->user_data_1 != write_user_data_1)
    {
        return 0;          // error;
    }

    if (result->user_data_2 != write_user_data_2)
    {
        return 0;          // error
    }

    // bundle id must be 0
    if (result->bundle_id != 0)
    {
        return 0;          // error
    }
}

return 1;
}

```

Table 3.1039. mem_lb_lookup_bundle_id parameters

Type	Name	Description
mem_lb_lookup_command_format_in_write_reg_t*	xfer	Transfer registers containing one 64 bit word of hash input and statistics
volatile void __addr40 __mem*	address	40 bit pointer with bucket address (<code>mem_lb_descriptor_address_format_t</code>) in lower 32 bits. See 6xxx databook for recommended addressing mode for higher 8 bits of address parameter.
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL_PAIR*	sig_pair_ptr	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup (two 64-bit words).

3.25.4.14 mem_lb_lookup_dcache

Prototype:

```
mem_lb_lookup_direct_result_in_read_reg_t*
mem_lb_lookup_dcache(mem_lb_lookup_command_format_in_write_reg_t* xfer, volatile void
__addr40 __mem* address, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Lookup DCache using has input and update statistics.

Perform a comparison lookup on 24-bit initial hash input, use the bundleId to address the Id table and use the the Id table result to address the designated location in DCache that becomes the DCache lookup result. Add 16-bit stat input data to the statistics counter. Pull one 64-bit word of Hash and Stat, push two 64-bit words as result. This is also called direct DCache lookup.

Table 3.1040. mem_lb_lookup_dcache parameters

Type	Name	Description
mem_lb_lookup_command_format_in_write_reg_t*	xfer	Transfer registers containing one 64 bit word of hash and statistics
volatile void __addr40 __mem*	address	40 bit pointer with bucket address (mem_lb_descriptor_address_format_t) in lower 32 bits. See 6xxx databook for recommended addressing mode for higher 8 bits of address parameter.
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL_PAIR*	sig_pair_ptr	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup (two 64-bit words).

3.25.4.15 mem_lb_lookup_id_table

Prototype:

```
mem_lb_lookup_direct_result_in_read_reg_t*
mem_lb_lookup_id_table(mem_lb_lookup_command_format_in_write_reg_t* xfer, volatile void
__addr40 __mem* address, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Lookup IdTable using has input and update statistics.

Perform a comparison lookup on 24-bit initial hash input, use the bundleId to address the Id table and return the Id table result. Add 16-bit stat input data to the statistics counter. Pull one 64-bit word of Hash and Stat, push two 64-bit words as result. This is also called direct Id table lookup.

Table 3.1041. mem_lb_lookup_id_table parameters

Type	Name	Description
mem_lb_lookup_command_format_in_write_reg_t*	xfer	Transfer registers containing one 64 bit word of hash and statistics
volatile void __addr40 __mem*	address	40 bit pointer with bucket address (mem_lb_descriptor_address_format_t) in lower 32 bits. See 6xxx databook for recommended addressing mode for higher 8 bits of address parameter.
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL_PAIR*	sig_pair_ptr	Signal pair to raise upon completion

Returns:

Pointer to the first read transfer register containing the result of the lookup (two 64-bit words).

3.25.4.16 mem_lb_init

Prototype:

```
void mem_lb_init(unsigned int descriptor_array_number, unsigned int mu_island, unsigned
int stat_base_addr, enum MEM_LB_LOCATION stat_addr_ref, unsigned int hash_base_addr, enum
MEM_LB_LOCATION hash_addr_ref, unsigned int boundaries, unsigned int index_bits, unsigned
int compare_bits, unsigned int hash_seed, unsigned long long user_data, unsigned int
count)
```

Description:

Initialise the descriptor array prior to doing lookups.

A valid descriptor must be loaded to the Descriptor Array prior to issuing any descriptor- dependent commands (eg. LBLookup). `mem_lb_write_desc()` is used to initialise a location in the descriptor array. The descriptor array supports 64 descriptors. Boundaries, index_bits and compare_bits have only a few supported configurations. See Supported Descriptor Parameters Table in 6xxx databook.

An example of `mem_lb_init()` is given below. The descriptor is stored in array number 4. `mem_lb_read_desc()` is used to read back the descriptor and verify the stored values.

```
unsigned long long island = (unsigned long long)LoadTimeConstant("___ADDR_I28_IMEM");
mem_lb_descriptor_address_format_t descriptor;
unsigned int descriptor_array = 5;
unsigned int count = 1;
```

```
unsigned int write_stat_base_addr = 0x100;
unsigned int write_hash_base_addr = 0x00;
unsigned int write_user_data_1 = 0x9abcdef;
unsigned int write_user_data_2 = 0x12345678;
unsigned int write_hash_seed = 0aaaaaaaa;

descriptor.value = 0;
descriptor.array_entry = descriptor_array;

mem_lb_init
(
    descriptor.array_entry,
    island >> 32,
    write_stat_base_addr,
    0,
    write_hash_base_addr,
    0,
    3,      // boundaries
    0,      // index_bits
    24,     // compare_bits
    write_hash_seed,
    ((unsigned long long)write_user_data_1 << 32) | write_user_data_2,
    count
);

// Verify by reading back the descriptor
{
    unsigned long long address = ((unsigned long long) island | descriptor.value);
    SIGNAL           sig;
    mem_lb_desc_in_read_reg_t      lb_read_xfer_desc;

    mem_lb_read_desc(
        (void *)&lb_read_xfer_desc.value[0],
        (__addr40 __mem void *)address,
        count,
        ctx_swap,
        &sig
    );

    // Verify user data
    if (lb_read_xfer_desc.user_data_2 != write_user_data_2)
    {
        return 0;          // We have an error
    }

    if (lb_read_xfer_desc.user_data_1 != write_user_data_1)
    {
        return 0;          // We have an error
    }

    // Verify hash seed
    {
        unsigned int read_hash_seed = MEM_LB_HASH_SEED_FROM_DESC(lb_read_xfer_desc);

        if (read_hash_seed != write_hash_seed)
        {
            return 0;          // We have an error
        }
    }
}
```

```

// Verify statistics address
{
    unsigned int read_stat_base_addr= lb_read_xfer_desc.stat_base_addr;
    unsigned int read_stat_addr_ref = lb_read_xfer_desc.stat_addr_ref;

    if (read_stat_base_addr != write_stat_base_addr)
    {
        return 0;          // We have an error
    }

    if (read_stat_addr_ref != 0)
    {
        return 0;          // We have an error
    }
}

// Verify hash address
{
    unsigned int read_hash_addr_ref = lb_read_xfer_desc.hash_addr_ref;
    unsigned int read_hash_base_addr = MEM_LB_HASH_BASE_ADDR_FROM_DESC(lb_read_xfer_desc);

    if (read_hash_base_addr != write_hash_base_addr)
    {
        return 0;          // We have an error
    }

    if (read_hash_addr_ref != 0)
    {
        return 0;          // We have an error
    }
}

return 1;

```

Table 3.1042. mem_lb_init parameters

Type	Name	Description
unsigned int	<i>descriptor_array_number</i>	Descriptor array entry location (0 - 63).
unsigned int	<i>mu_island</i>	8 high bits of 40-bit address indicating the mu_island. See 6xxx databook for recommended addressing mode for higher 8 bits of address parameter.
unsigned int	<i>stat_base_addr</i>	Statistics base address
enum MEM_LB_LOCATION	<i>stat_addr_ref</i>	Location reference for statsics address (LB_LOCATION_LOCAL_MEM or LB_LOCATION_DCACHE)
unsigned int	<i>hash_base_addr</i>	Hash base address
enum MEM_LB_LOCATION	<i>hash_addr_ref</i>	Location reference for hash address (LB_LOCATION_LOCAL_MEM or LB_LOCATION_DCACHE)
unsigned int	<i>boundaries</i>	Boundary value that is used to divide the hash into regions. The number of boundaries in a hash bucket is determined by

Type	Name	Description
		the descriptor value. Hash is compared to boundary value in order to determine region indirection index
unsigned int	<i>index_bits</i>	Number of bits of hash to use as address offset
unsigned int	<i>compare_bits</i>	Number of bits of hash to use for compare
unsigned int	<i>hash_seed</i>	Hash seed value
unsigned long long	<i>user_data</i>	User specific data
unsigned int	<i>count</i>	Length in 64 bit words to write, calculated as count * 3 (valid values 1- 4)

See Also:

- `mem_lb_write_desc()`

3.26 MEM Lookup Intrinsics

This section discusses the enumerations, structs, unions, types and functions related to MEM Lookup operations.

Lookup engine functions are available on internal memory or external memory. Different lookups are available, refer to NFP-6xxx Programmer Reference Manual for a list.

Below is a complete example with 32-bit direct lookup on island 28 (imem0). The DLUT table is populated and then a lookup is performed of a specific index.

```
// i28 imem is used for island and base address
unsigned long long                      island = (unsigned long long)LoadTimeConstant("__ADDR__");
__declspec(i28.mem, addr40) unsigned int   base_address[16];

// Write to 32-bit DLUT table. See Table 9.66 "DLUT32 Memory Contents" in 6xxx databook
{
    volatile __declspec(write_reg) unsigned int wr_reg[8];
    unsigned int                           i;
    SIGNAL                                sig;

    // write in chunks of 128 bits for each data line
    // (result 0 to result 3/result 4 - result 7/..)
    for (i = 0; i < 4; i++)
    {
        wr_reg[i] = 0x11110000 + i;
    }

    mem_write64_ptr40(
        (void *)wr_reg,
        (__declspec(mem, addr40) void *)(&base_address[0]),
        2, ctx_swap,
        &sig
    );
}
```

```
for (i = 0; i < 4; i++)
{
    wr_reg[i] = 0x22222222 + i;
}

mem_write64_ptr40(
    (void *)wr_reg,
    (__declspec(mem, addr40) void *)(&base_address[4]),
    2,
    ctx_swap,
    &sig
);

for (i = 0; i < 4; i++)
{
    wr_reg[i] = 0x33333333 + i;
}

mem_write64_ptr40(
    (void *)wr_reg,
    (__declspec(mem, addr40) void *)(&base_address[8]),
    2,
    ctx_swap,
    &sig
);

for (i = 0; i < 4; i++)
{
    wr_reg[i] = 0x44444444 + i;
}

mem_write64_ptr40(
    (void *)wr_reg,
    (__declspec(mem, addr40) void *)(&base_address[12]),
    2,
    ctx_swap,
    &sig
);

// verify that all information is written before we do lookup.
{
    __declspec(read_reg) unsigned int rd_reg[2];
    mem_read64_ptr40(
        (void *)rd_reg,
        (__declspec(mem, addr40) void *)(&base_address[12]),
        1,
        ctx_swap,
        &sig
    );
}

// Do the lookup on index 0x0a
{
    volatile mem_lookup_dlut_small_t           small_table_desc;
    volatile __declspec(write_reg) unsigned int lookup_index[4];
    mem_lookup_result_in_read_reg_t          *read_result;
    SIGNAL_PAIR                             sig_pair;
```

```

lookup_index[0] = 0x0a;
lookup_index[1] = 0x00;

small_table_desc.value = 0;
small_table_desc.start_bit_position = 0;
small_table_desc.table_type = LOOKUP_DLUT_SMALL;
small_table_desc.table_size = LOOKUP_DLUT_SMALL_SIZE_1K;
small_table_desc.result = LOOKUP_32BIT_RESULT;
small_table_desc.direct_lookup = 1;
small_table_desc.internal_memory = 1;

// Refer to Table 9.64 "Small Direct Lookup Table (word address) -DLUT32" in 6xxx databook
small_table_desc.base_address = (unsigned int)base_address >> 12;

{
    unsigned long long lookup_addr = (island | small_table_desc.value);

    read_result = mem_lookup(
        (void *)&lookup_index[0],
        (__declspec(mem, addr40) void *)(&lookup_addr),
        1,
        sig_done,
        &sig_pair
    );
    wait_for_all(&sig_pair);
}
if (read_result->result != 0x33333335)
{
    return 0;          // We have an error
}
}
return 1;

```

3.26.1 MU Lookup Structs

3.26.1.1 mem_lookup_result_t

Table 3.1043. struct mem_lookup_result_t

Type	Name	Description
unsigned int	result	

3.26.2 MU Lookup Enumerations

3.26.2.1 MEM_LOOKUP_DLUT_SMALL_SIZE

Direct lookup small table sizes defined by the number of entries.

The actual size of the table is dependent on 24-bit or 32-bit result (see MEM_LOOKUP_RESULT) and the number of entries

- 24 bit result has table size of 3 * number of entries.
- 32 bit result has table size of 4 * number of entries. Small tables must be 4K byte aligned.

Table 3.1044. enum MEM_LOOKUP_DLUT_SMALL_SIZE

Name	Description
LOOKUP_DLUT_SMALL_SIZE_1K	1K table entries.
LOOKUP_DLUT_SMALL_SIZE_2K	2K table entries.
LOOKUP_DLUT_SMALL_SIZE_4K	4K table entries.
LOOKUP_DLUT_SMALL_SIZE_8K	8K table entries.
LOOKUP_DLUT_SMALL_SIZE_16K	16K table entries.
LOOKUP_DLUT_SMALL_SIZE_32K	32K table entries.
LOOKUP_DLUT_SMALL_SIZE_64K	64K table entries.
LOOKUP_DLUT_SMALL_SIZE_128K	128K table entries.

3.26.2.2 MEM_LOOKUP_DLUT_LARGE_SIZE

Direct lookup large table sizes defined by the number of entries.

The actual size of the table is dependent on 24-bit or 32-bit result (see MEM_LOOKUP_RESULT) and the number of entries

- 24 bit result has table size of 3 * number of entries.
- 32 bit result has table size of 4 * number of entries. Large tables must be 256K byte aligned.

Table 3.1045. enum MEM_LOOKUP_DLUT_LARGE_SIZE

Name	Description
LOOKUP_DLUT_LARGE_SIZE_64K	64K table entries.
LOOKUP_DLUT_LARGE_SIZE_128K	128K table entries.
LOOKUP_DLUT_LARGE_SIZE_256K	256K table entries.
LOOKUP_DLUT_LARGE_SIZE_512K	512K table entries.
LOOKUP_DLUT_LARGE_SIZE_1M	1M table entries.
LOOKUP_DLUT_LARGE_SIZE_2M	2M table entries.
LOOKUP_DLUT_LARGE_SIZE_4M	4M table entries.
LOOKUP_DLUT_LARGE_SIZE_8M	8M table entries.

3.26.2.3 MEM_LOOKUP_RESULT

Lookup result bits.

Table 3.1046. enum MEM_LOOKUP_RESULT

Name	Description
LOOKUP_24BIT_RESULT	24 bit result.
LOOKUP_32BIT_RESULT	32 bit result.

3.26.2.4 MEM_LOOKUP_DLUT_TYPE

Lookup result bits.

Table 3.1047. enum MEM_LOOKUP_DLUT_TYPE

Name	Description
LOOKUP_DLUT_SMALL	Small table - 1K to 128K.
LOOKUP_DLUT_LARGE	Large table - 64K to 8M.

3.26.2.5 MEM_LOOKUP_ALUT_SIZE

Algorithmic lookup table sizes in cache lines where 16 bytes (128-bit) per cache line.

Table 3.1048. enum MEM_LOOKUP_ALUT_SIZE

Name	Description
LOOKUP_ALUT_SIZE_1	1 128-bit cache line.
LOOKUP_ALUT_SIZE_2	2 128-bit cache lines.
LOOKUP_ALUT_SIZE_3	3 128-bit cache lines.
LOOKUP_ALUT_SIZE_4	4 128-bit cache lines.

3.26.2.6 MEM_LOOKUP_ALUT_COMMANDS

Algorithmic lookup table sizes in cache lines where 16 bytes (128-bit) per cache line.

algorithmic lookup tables include: ALUT, PMM (prefix match with mask), TCAM (ternary content addressable memory table), SPLIT and multi-bit tables.

Table 3.1049. enum MEM_LOOKUP_ALUT_COMMANDS

Name	Description
LOOKUP_ALUT_COMMAND_24_3_4	ALUT with index of 3-bits and a result of 24-bits with 4 possible results.
LOOKUP_ALUT_COMMAND_24_4_7	ALUT with index of 4-bits and a result of 24-bits with 7 possible results.
LOOKUP_ALUT_COMMAND_24_5_4	ALUT with index of 5-bits and a result of 24-bits with 4 possible results.
LOOKUP_ALUT_COMMAND_24_5_8	ALUT with index of 5-bits and a result of 24-bits with 8 possible results.
LOOKUP_ALUT_COMMAND_24_5_14	ALUT with index of 5-bits and a result of 24-bits with 14 possible results.
LOOKUP_ALUT_COMMAND_24_6_2	ALUT with index of 6-bits and a result of 24-bits with 2 possible results.

Name	Description
LOOKUP_ALUT_COMMAND_24_6_4	ALUT with index of 6-bits and a result of 24-bits with 4 possible results.
LOOKUP_ALUT_COMMAND_32_5_2	ALUT with index of 5-bits and a result of 32-bits with 2 possible results.
LOOKUP_ALUT_COMMAND_32_5_4	ALUT with index of 5-bits and a result of 32-bits with 4 possible results.
LOOKUP_ALUT_COMMAND_32_5_8	ALUT with index of 5-bits and a result of 32-bits with 8 possible results.
LOOKUP_ALUT_COMMAND_32_5_11	ALUT with index of 5-bits and a result of 32-bits with 11 possible results.
LOOKUP_ALUT_COMMAND_32_7_2	ALUT with index of 7-bits and a result of 32-bits with 2 possible results.
LOOKUP_ALUT_COMMAND_32_7_4	ALUT with index of 7-bits and a result of 32-bits with 4 possible results.
LOOKUP_PMM_COMMAND_32_12_4	PMM with index of 12-bits and a result of 32 bits with 4 possible results.
LOOKUP_PMM_COMMAND_32_12_7	PMM with index of 12-bits and a result of 32 bits with 7 possible results.
LOOKUP_PMM_COMMAND_32_12_9	PMM with index of 12-bits and a result of 32 bits with 9 possible results.
LOOKUP_TCAM_COMMAND_32_8_4	TCAM with index of 8-bits and a result of 32 bits with 4 possible results.
LOOKUP_TCAM_COMMAND_32_8_8	TCAM with index of 8-bits and a result of 32 bits with 8 possible results.
LOOKUP_TCAM_COMMAND_32_8_10	TCAM with index of 8-bits and a result of 32 bits with 10 possible results.
LOOKUP_TCAM_COMMAND_32_12_4	TCAM with index of 12-bits and a result of 32 bits with 4 possible results.
LOOKUP_TCAM_COMMAND_32_12_8	TCAM with index of 12-bits and a result of 32 bits with 8 possible results.
LOOKUP_SPLIT_COMMAND_32_8_6	SPLIT with index of 8-bits and a result of 32 bits with 6 possible results.
LOOKUP_SPLIT_COMMAND_32_16_5	SPLIT with index of 16-bits and a result of 32 bits with 5 possible results.
LOOKUP_SPLIT_COMMAND_32_24_9	SPLIT with index of 24-bits and a result of 32 bits with 9 possible results.
LOOKUP_SPLIT_COMMAND_32_32_8	SPLIT with index of 32-bits and a result of 32 bits with 8 possible results.
LOOKUP_MULTIBIT_COMMAND	Multi-bit command.

3.26.2.7 MEM_LOOKUP_HASH_TABLE_SIZE

Hash lookup table sizes.

Table 3.1050. enum MEM_LOOKUP_HASH_TABLE_SIZE

Name	Description
LOOKUP_HASH_TABLE_SIZE_1	1K table size.
LOOKUP_HASH_TABLE_SIZE_2	2K table size.
LOOKUP_HASH_TABLE_SIZE_3	4K table size.
LOOKUP_HASH_TABLE_SIZE_4	8K table size.
LOOKUP_HASH_TABLE_SIZE_5	16K table size.
LOOKUP_HASH_TABLE_SIZE_6	32K table size.
LOOKUP_HASH_TABLE_SIZE_7	64K table size.
LOOKUP_HASH_TABLE_SIZE_8	128K table size.

3.26.2.8 MEM_LOOKUP_HASH_STARTING_BIT

Starting bit position for hash table lookup.

Table 3.1051. enum MEM_LOOKUP_HASH_STARTING_BIT

Name	Description
LOOKUP_HASH_BIT_0	Starting bit position at bit 0.
LOOKUP_HASH_BIT_32	Starting bit position at bit 32.
LOOKUP_HASH_BIT_64	Starting bit position at bit 64.
LOOKUP_HASH_BIT_96	Starting bit position at bit 96.

3.26.2.9 MEM_LOOKUP_HASH_COMMANDS

Hash command to use for lookup.

Table 3.1052. enum MEM_LOOKUP_HASH_COMMANDS

Name	Description
LOOKUP_HASH_COMMAND_CAMR_32_16	16 bytes operation.
LOOKUP_HASH_COMMAND_CAMR_32_64	64 bytes operation.
LOOKUP_HASH_COMMAND_CAMR_48_64	64 bytes operation.
LOOKUP_HASH_COMMAND_CAMR_64_16	16 bytes operation.
LOOKUP_HASH_COMMAND_CAMR_64_64	64 bytes operation.
LOOKUP_HASH_COMMAND_CAM_32_16	16 bytes operation.
LOOKUP_HASH_COMMAND_CAM_32_64	64 bytes operation.
LOOKUP_HASH_COMMAND_CAM_48_16	16 bytes operation.
LOOKUP_HASH_COMMAND_CAM_48_64	64 bytes operation.
LOOKUP_HASH_COMMAND_CAM_64_16	16 bytes operation.
LOOKUP_HASH_COMMAND_CAM_64_64	64 bytes operation.
LOOKUP_HASH_COMMAND_CAM_128_16	16 bytes operation.
LOOKUP_HASH_COMMAND_CAM_128_64	64 bytes operation.
LOOKUP_HASH_COMMAND_LHASHR_16_28_16_2	16 bytes bucket size with 2 bucket search.
LOOKUP_HASH_COMMAND_LHASHR_16_28_16_4	16 bytes bucket size with 4 bucket search.
LOOKUP_HASH_COMMAND_LHASHR_16_28_16_7	16 bytes bucket size with 7 bucket search.
LOOKUP_HASH_COMMAND_LHASHR_16_28_64_2	64 bytes bucket size with 2 bucket search.
LOOKUP_HASH_COMMAND_LHASHR_16_28_64_4	64 bytes bucket size with 4 bucket search.
LOOKUP_HASH_COMMAND_LHASHR_16_28_64_7	64 bytes bucket size with 7 bucket search.
LOOKUP_HASH_COMMAND_LHASHR_48_60_16_2	16 bytes bucket size with 2 bucket search.
LOOKUP_HASH_COMMAND_LHASHR_48_60_16_4	16 bytes bucket size with 4 bucket search.

Name	Description
LOOKUP_HASH_COMMAND_LHASHR_48_60_16_7	16 bytes bucket size with 7 bucket search.
LOOKUP_HASH_COMMAND_LHASHR_48_60_64_2	64 bytes bucket size with 2 bucket search.
LOOKUP_HASH_COMMAND_LHASHR_48_60_64_4	64 bytes bucket size with 4 bucket search.
LOOKUP_HASH_COMMAND_LHASHR_48_60_64_7	64 bytes bucket size with 7 bucket search.
LOOKUP_HASH_COMMAND_LHASH_16_28_16_2	16 bytes bucket size with 2 bucket search.
LOOKUP_HASH_COMMAND_LHASH_16_28_16_4	16 bytes bucket size with 4 bucket search.
LOOKUP_HASH_COMMAND_LHASH_16_28_16_7	16 bytes bucket size with 7 bucket search.
LOOKUP_HASH_COMMAND_LHASH_16_28_64_2	64 bytes bucket size with 2 bucket search.
LOOKUP_HASH_COMMAND_LHASH_16_28_64_4	64 bytes bucket size with 4 bucket search.
LOOKUP_HASH_COMMAND_LHASH_16_28_64_7	64 bytes bucket size with 7 bucket search.
LOOKUP_HASH_COMMAND_LHASH_48_60_16_2	16 bytes bucket size with 2 bucket search.
LOOKUP_HASH_COMMAND_LHASH_48_60_16_4	16 bytes bucket size with 4 bucket search.
LOOKUP_HASH_COMMAND_LHASH_48_60_16_7	16 bytes bucket size with 7 bucket search.
LOOKUP_HASH_COMMAND_LHASH_48_60_64_2	64 bytes bucket size with 2 bucket search.
LOOKUP_HASH_COMMAND_LHASH_48_60_64_4	64 bytes bucket size with 4 bucket search.
LOOKUP_HASH_COMMAND_LHASH_48_60_64_7	64 bytes bucket size with 7 bucket search.

3.26.3 MU Lookup Unions

3.26.3.1 mem_lookup_alut_t

Command for algorithmic lookup table.

Table 3.1053. union mem_lookup_alut_t

Type	Name	Description
unsigned int	internal_memory:1	If internal memory is used.
unsigned int	direct_lookup:1	0 = not a direct lookup.
unsigned int	lookup_type:1	0 = algorithmic lookup.
MEM_LOOKUP_ALUT_SIZE	table_size:2	Specified table size.
unsigned int	table_number:3	Table number 0-7 (algorithmic table location CSR) contains address bits 32:28 and IMEM or EMEM select.
unsigned int	address_bits:24	Address bits 27:4.
unsigned int	value	Accessor to entire descriptor structure.

3.26.3.2 mem_lookup_dlut_large_recursive_t

Command for recursive 32-bit direct lookup large table.

Table 3.1054. union mem_lookup_dlut_large_recursive_t

Type	Name	Description
unsigned int	further_lookup_required:1	If further lookup is required.
unsigned int	direct_lookup:1	1 = direct lookup.
unsigned int	base_address:15	Base address at which the first result is located.
unsigned int	reserved_1:2	Reserved.
MEM_LOOKUP_RESULT	result:1	0 = 24 bits, 1 = 32 bits.
unsigned int	mem_location:1	Location where 0 = IMEM and 1 = EMEM.
MEM_LOOKUP_DLUT_LARGE_SIZE	table_size:3	Specified table size.
unsigned int	table_type:1	1 = large table size.
unsigned int	start_bit_position:7	Starting bit position, bit 0 to bit 127.
unsigned int	value	Accessor to entire descriptor structure.

3.26.3.3 mem_lookup_dlut_large_t

Command for direct lookup large table.

Table 3.1055. union mem_lookup_dlut_large_t

Type	Name	Description
unsigned int	internal_memory:1	If internal memory is used.
unsigned int	direct_lookup:1	1 = direct lookup.
unsigned int	base_address:15	Base address at which the first result is located.
unsigned int	reserved_2:2	Reserved.
MEM_LOOKUP_RESULT	result:1	0 = 24 bits, 1 = 32 bits.
unsigned int	reserved_1:1	Reserved.
MEM_LOOKUP_DLUT_LARGE_SIZE	table_size:3	Specified table size.
unsigned int	table_type:1	1 = large table size.
unsigned int	start_bit_position:7	Starting bit position, bit 0 to bit 127.
unsigned int	value	Accessor to entire descriptor structure.

3.26.3.4 mem_lookup_dlut_small_recursive_t

Command for recursive 32-bit direct lookup small table.

Table 3.1056. union mem_lookup_dlut_small_recursive_t

Type	Name	Description
unsigned int	further_lookup_required:1	If further lookup is required.
unsigned int	direct_lookup:1	1 = direct lookup.
unsigned int	base_address:16	Base address at which the first result is located.
unsigned int	reserved_1:1	Reserved.
MEM_LOOKUP_RESULT	result:1	0 = 24 bits, 1 = 32 bits.
unsigned int	mem_location:1	Location where 0 = IMEM and 1 = EMEM.
MEM_LOOKUP_DLUT_SMALL_SIZE	table_size:3	Specified table size.
unsigned int	table_type:1	0 = small table type: 1K to 128K.
unsigned int	start_bit_position:7	Starting bit position, bit 0 to bit 127.
unsigned int	value	Accessor to entire descriptor structure.

3.26.3.5 mem_lookup_dlut_small_t

Command for direct lookup small table.

Table 3.1057. union mem_lookup_dlut_small_t

Type	Name	Description
unsigned int	internal_memory:1	If internal memory is used.
unsigned int	direct_lookup:1	1 = direct lookup.
unsigned int	base_address:16	Base address at which the first result is located.
unsigned int	reserved_2:1	Reserved.
MEM_LOOKUP_RESULT	result:1	0 = 24 bits, 1 = 32 bit.
unsigned int	reserved_1:1	Reserved.
MEM_LOOKUP_DLUT_SMALL_SIZE	table_size:3	Specified table entry size.
unsigned int	table_type:1	0 = small table type: 1K to 128K.
unsigned int	start_bit_position:7	Starting bit position, bit 0 to bit 127.
unsigned int	value	Accessor to entire descriptor structure.

3.26.3.6 mem_lookup_hash_table_t

Command for hash lookup table.

Table 3.1058. union mem_lookup_hash_table_t

Type	Name	Description
unsigned int	internal_memory:1	If internal memory is used: 1 = IMEM 0=EMEM.

Type	Name	Description
unsigned int	direct_lookup:1	0 = not a direct lookup.
unsigned int	hash_lookup:1	1 = hash lookup.
unsigned int	base_address:17	17-bit base address.
unsigned int	reserved_1:1	Reserved.
MEM_LOOKUP_HASH_TABLE_SIZE	table_size:3	Specified table size.
unsigned int	hash_command:6	6-bit hash command.
MEM_LOOKUP_HASH_STARTING_BIT	start_bit_position:2	Starting bit position.
unsigned int	value	Accessor to entire descriptor structure.

3.26.3.7 mem_lookup_recursive_hash_table_t

Command for recursive hash lookup table.

Table 3.1059. union mem_lookup_recursive_hash_table_t

Type	Name	Description
unsigned int	further_lookup_required:1	1 = further lookup required.
unsigned int	direct_lookup:1	0 = not a direct lookup.
unsigned int	hash_lookup:1	1 = hash lookup.
unsigned int	base_address:17	17-bit base address.
unsigned int	mem_location:1	Location where 0 = IMEM and 1 = EMEM.
MEM_LOOKUP_HASH_TABLE_SIZE	table_size:3	Specified table size.
unsigned int	hash_command:6	6-bit hash command.
MEM_LOOKUP_HASH_STARTING_BIT	start_bit_position:2	Starting bit position.
unsigned int	value	Accessor to entire descriptor structure.

3.26.4 MU Lookup Typedefs

3.26.4.1 MEM_LOOKUP_DLUT_SMALL_SIZE

Direct lookup small table sizes defined by the number of entries.

The actual size of the table is dependent on 24-bit or 32-bit result (see MEM_LOOKUP_RESULT) and the number of entries

- 24 bit result has table size of 3 * number of entries.
- 32 bit result has table size of 4 * number of entries. Small tables must be 4K byte aligned.

Table 3.1060. `typedef MEM_LOOKUP_DLUT_SMALL_SIZE`

Type	Definition
<code>MEM_LOOKUP_DLUT_SMALL_SIZE</code>	<code>enum MEM_LOOKUP_DLUT_SMALL_SIZE</code>

3.26.4.2 `MEM_LOOKUP_DLUT_LARGE_SIZE`

Direct lookup large table sizes defined by the number of entries.

The actual size of the table is dependent on 24-bit or 32-bit result (see `MEM_LOOKUP_RESULT`) and the number of entries

- 24 bit result has table size of 3 * number of entries.
- 32 bit result has table size of 4 * number of entries. Large tables must be 256K byte aligned.

Table 3.1061. `typedef MEM_LOOKUP_DLUT_LARGE_SIZE`

Type	Definition
<code>MEM_LOOKUP_DLUT_LARGE_SIZE</code>	<code>enum MEM_LOOKUP_DLUT_LARGE_SIZE</code>

3.26.4.3 `MEM_LOOKUP_RESULT`

Lookup result bits.

Table 3.1062. `typedef MEM_LOOKUP_RESULT`

Type	Definition
<code>MEM_LOOKUP_RESULT</code>	<code>enum MEM_LOOKUP_RESULT</code>

3.26.4.4 `MEM_LOOKUP_DLUT_TYPE`

Lookup result bits.

Table 3.1063. `typedef MEM_LOOKUP_DLUT_TYPE`

Type	Definition
<code>MEM_LOOKUP_DLUT_TYPE</code>	<code>enum MEM_LOOKUP_DLUT_TYPE</code>

3.26.4.5 `mem_lookup_dlut_small_t`

Command for direct lookup small table.

Table 3.1064. `typedef mem_lookup_dlut_small_t`

Type	Definition
<code>mem_lookup_dlut_small_t</code>	<code>union mem_lookup_dlut_small_t</code>

3.26.4.6 mem_lookup_dlut_large_t

Command for direct lookup large table.

Table 3.1065. typedef mem_lookup_dlut_large_t

Type	Definition
mem_lookup_dlut_large_t	union mem_lookup_dlut_large_t

3.26.4.7 mem_lookup_dlut_small_recursive_t

Command for recursive 32-bit direct lookup small table.

Table 3.1066. typedef mem_lookup_dlut_small_recursive_t

Type	Definition
mem_lookup_dlut_small_recursive_t	union mem_lookup_dlut_small_recursive_t

3.26.4.8 mem_lookup_dlut_large_recursive_t

Command for recursive 32-bit direct lookup large table.

Table 3.1067. typedef mem_lookup_dlut_large_recursive_t

Type	Definition
mem_lookup_dlut_large_recursive_t	union mem_lookup_dlut_large_recursive_t

3.26.4.9 mem_lookup_result_t

Table 3.1068. typedef mem_lookup_result_t

Type	Definition
mem_lookup_result_t	struct mem_lookup_result_t

3.26.4.10 mem_lookup_result_in_read_reg_t

Table 3.1069. typedef mem_lookup_result_in_read_reg_t

Type	Definition
mem_lookup_result_in_read_reg_t	__xread mem_lookup_result_t

3.26.4.11 MEM_LOOKUP_ALUT_SIZE

Algorithmic lookup table sizes in cache lines where 16 bytes (128-bit) per cache line.

Table 3.1070. `typedef MEM_LOOKUP_ALUT_SIZE`

Type	Definition
<code>MEM_LOOKUP_ALUT_SIZE</code>	<code>enum MEM_LOOKUP_ALUT_SIZE</code>

3.26.4.12 `MEM_LOOKUP_ALUT_COMMANDS`

Algorithmic lookup table sizes in cache lines where 16 bytes (128-bit) per cache line.

algorithmic lookup tables include: ALUT, PMM (prefix match with mask), TCAM (ternary content addressable memory table), SPLIT and multi-bit tables.

Table 3.1071. `typedef MEM_LOOKUP_ALUT_COMMANDS`

Type	Definition
<code>MEM_LOOKUP_ALUT_COMMANDS</code>	<code>enum MEM_LOOKUP_ALUT_COMMANDS</code>

3.26.4.13 `mem_lookup_alut_t`

Command for algorithmic lookup table.

Table 3.1072. `typedef mem_lookup_alut_t`

Type	Definition
<code>mem_lookup_alut_t</code>	<code>union mem_lookup_alut_t</code>

3.26.4.14 `MEM_LOOKUP_HASH_TABLE_SIZE`

Hash lookup table sizes.

Table 3.1073. `typedef MEM_LOOKUP_HASH_TABLE_SIZE`

Type	Definition
<code>MEM_LOOKUP_HASH_TABLE_SIZE</code>	<code>enum MEM_LOOKUP_HASH_TABLE_SIZE</code>

3.26.4.15 `MEM_LOOKUP_HASH_STARTING_BIT`

Starting bit position for hash table lookup.

Table 3.1074. `typedef MEM_LOOKUP_HASH_STARTING_BIT`

Type	Definition
<code>MEM_LOOKUP_HASH_STARTING_BIT</code>	<code>enum MEM_LOOKUP_HASH_STARTING_BIT</code>

3.26.4.16 MEM_LOOKUP_HASH_COMMANDS

Hash command to use for lookup.

Table 3.1075. typedef MEM_LOOKUP_HASH_COMMANDS

Type	Definition
MEM_LOOKUP_HASH_COMMANDS	enum MEM_LOOKUP_HASH_COMMANDS

3.26.4.17 mem_lookup_hash_table_t

Command for hash lookup table.

Table 3.1076. typedef mem_lookup_hash_table_t

Type	Definition
mem_lookup_hash_table_t	union mem_lookup_hash_table_t

3.26.4.18 mem_lookup_recursive_hash_table_t

Command for recursive hash lookup table.

Table 3.1077. typedef mem_lookup_recursive_hash_table_t

Type	Definition
mem_lookup_recursive_hash_table_t	union mem_lookup_recursive_hash_table_t

3.26.5 MU Lookup Functions

3.26.5.1 mem_lookup

Prototype:

```
mem_lookup_result_in_read_reg_t* mem_lookup(__xwrite void* xfer, volatile void __addr40
__mem* address, unsigned int count, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Do lookup.

Load descriptor data into the Local Descriptor Array. The array supports up to 64 Descriptors. Descriptors must be loaded into the Local Descriptor Array prior to doing lookups.

Table 3.1078. mem_lookup parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>xfer</code>	Transfer registers containing index to lookup.
<code>volatile void __addr40 __mem*</code>	<code>address</code>	40 bit pointer with lower 32 bits setup as specific command address for table, <ul style="list-style-type: none"> • <code>mem_lookup_dlut_small_t</code> or <code>mem_lookup_dlut_large_t</code> for DLUT tables, • <code>mem_lookup_alut_t</code> for ALUT tables, • <code>mem_lookup_recursive_hash_table_t</code> for HASH tables.
<code>unsigned int</code>	<code>count</code>	Length in 64-bit words to read, (valid values 1 - 2).
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (can only be <code>sig_done</code>).
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal to raise upon completion.

3.27 PCI Express Intrinsics

This section describes functions for PCI Express operations.

3.27.1 PCI Express Functions

3.27.1.1 pcie_read_ptr40

Prototype:

```
void pcie_read_ptr40(__xread void* data, volatile void __addr40 *address, unsigned int
count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

read from PCIe into transfer registers, 40-bit addr

Table 3.1079. pcie_read_ptr40 parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Address to read data into
<code>volatile void __addr40*</code>	<code>address</code>	Address to read data from (aligned on 4-byte boundary)
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to read (1-32).
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (<code>sig_done</code> or <code>ctx_swap</code>)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.27.1.2 pcie_write_ptr40

Prototype:

```
void pcie_write_ptr40(__xwrite void* data, volatile void __addr40 *address, unsigned int
count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

write to PCIe from transfer register, 40-bit addr

Table 3.1080. pcie_write_ptr40 parameters

Type	Name	Description
__xwrite void*	data	Data to write to PCIe
volatile void __addr40*	address	Address to write data to (aligned on 4-byte boundary)
unsigned int	count	Number of 32-bit words to write (1-32).
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

3.27.1.3 pcie_read_pci_ptr40

Prototype:

```
void pcie_read_pci_ptr40(__xread void* data, volatile void __addr40 *address, unsigned
int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

read from PCIe internal target into transfer registers, 40-bit addr

Table 3.1081. pcie_read_pci_ptr40 parameters

Type	Name	Description
__xread void*	data	Xfer reg to read data into
volatile void __addr40*	address	Address, [9:16] target number, [15:0] target addr
unsigned int	count	Number of 32-bit words to read (1-32).
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

3.27.1.4 pcie_write_pci_ptr40

Prototype:

```
void pcie_write_pci_ptr40(__xwrite void* data, volatile void __addr40 *address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

write to PCIe internal target from transfer register, 40-bit addr

Table 3.1082. pcie_write_pci_ptr40 parameters

Type	Name	Description
__xwrite void*	data	Xfer reg to write to PCIe
volatile void __addr40*	address	Address, [9:16] target number, [15:0] target addr
unsigned int	count	Number of 32-bit words to write (1-32).
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

3.27.1.5 pcie_read_rid_ptr40

Prototype:

```
void pcie_read_rid_ptr40(__xread void* data, unsigned int requester_id, volatile void __addr40 *address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read data from PCIe into transfer registers; overriding the PCIe Requester ID to use for the PCIe transaction, 40-bit addr.

Table 3.1083. pcie_read_rid_ptr40 parameters

Type	Name	Description
__xread void*	data	Address to read data into
unsigned int	requester_id	PCIe requester ID to override
volatile void __addr40*	address	40-bit address
unsigned int	count	Number of 32-bit words to read (1-32).
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

3.27.1.6 pcie_write_rid_ptr40

Prototype:

```
void pcie_write_rid_ptr40(__xwrite void* data, unsigned int requester_id, volatile void __addr40 *address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write data to PCIe from transfer register; overriding the PCIe Requester ID to use for the PCIe transaction, 40-bit addr.

Table 3.1084. pcie_write_rid_ptr40 parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Data to write to PCIe
<code>unsigned int</code>	<code>requester_id</code>	PCIe requester ID to override
<code>volatile void __addr40*</code>	<code>address</code>	Address to write data to (aligned on 4-byte boundary)
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to write (1-32).
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.27.1.7 pcie_read_ptr32

Prototype:

```
void pcie_read_ptr32(__xread void* data, volatile void* address, unsigned int count,
sync_t sync, SIGNAL* sig_ptr)
```

Description:

32-bit read from PCIe into transfer registers, 32-bit addr

Table 3.1085. pcie_read_ptr32 parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Address to read data into
<code>volatile void*</code>	<code>address</code>	Address to read data from (aligned on 4-byte boundary)
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to read (1-32).
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.27.1.8 pcie_write_ptr32

Prototype:

```
void pcie_write_ptr32(__xwrite void* data, volatile void* address, unsigned int count,
sync_t sync, SIGNAL* sig_ptr)
```

Description:

32-bit write to PCIe from transfer register, 32-bit addr

Table 3.1086. pcie_write_ptr32 parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Data to write to PCIe
<code>volatile void*</code>	<code>address</code>	Address to write data to (aligned on 4-byte boundary)
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to write (1-32).
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.27.1.9 pcie_read_ind_ptr32

Prototype:

```
void pcie_read_ind_ptr32(__xread void* data, volatile void* address, unsigned int max_nn,
generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read data from PCIe into transfer register indirect mode, 32-bit addr.

Table 3.1087. pcie_read_ind_ptr32 parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Address to read data into
<code>volatile void*</code>	<code>address</code>	Address to read data from (aligned on 4-byte boundary)
<code>unsigned int</code>	<code>max_nn</code>	Maximum number of 32-bit words to read (1-32)
<code>generic_ind_t</code>	<code>ind</code>	Indirect word
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

3.27.1.10 pcie_write_ind_ptr32

Prototype:

```
void pcie_write_ind_ptr32(__xwrite void* data, volatile void* address, unsigned int max_nn,
generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write data to PCIe from transfer register indirect mode, 32-bit addr.

Table 3.1088. pcie_write_ind_ptr32 parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Data to write to PCIe

Type	Name	Description
volatile void*	address	Address to write data to (aligned on 4-byte boundary)
unsigned int	max_nn	Maximum number of 32-bit words to write (1-32)
generic_ind_t	ind	Indirect word
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

3.27.1.11 pcie_read_pci_ptr32

Prototype:

```
void pcie_read_pci_ptr32(__xread void* data, volatile void* address, unsigned int count,
sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read 4-byte words from PCIe internal into transfer register, 32-bit addr.

Table 3.1089. pcie_read_pci_ptr32 parameters

Type	Name	Description
__xread void*	data	Address to read data into
volatile void*	address	Address to read from
unsigned int	count	Number of 32-bit words to read (1-16).
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

3.27.1.12 pcie_read_pci_ind_ptr32

Prototype:

```
void pcie_read_pci_ind_ptr32(__xread void* data, volatile void* address, unsigned int
max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read data from PCIe internal to transfer register indirect mode, 32-bit addr.

Table 3.1090. pcie_read_pci_ind_ptr32 parameters

Type	Name	Description
__xread void*	data	Address to read data into
volatile void*	address	Address to read data from
unsigned int	max_nn	Maximum number of 32-bit words to read (1-16)

Type	Name	Description
generic_ind_t	<i>ind</i>	Indirect word
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.27.1.13 pcie_write_pci_ptr32

Prototype:

```
void pcie_write_pci_ptr32(__xwrite void* data, volatile void* address, unsigned int count,
sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write 4-byte words to PCI internal from transfer register,32-bit addr.

Table 3.1091. pcie_write_pci_ptr32 parameters

Type	Name	Description
__xwrite void*	<i>data</i>	Data to write to PCIe internal
volatile void*	<i>address</i>	Address to write data to (aligned on 4-byte boundary)
unsigned int	<i>count</i>	Number of 32-bit words to write (1-16).
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.27.1.14 pcie_write_pci_ind_ptr32

Prototype:

```
void pcie_write_pci_ind_ptr32(__xwrite void* data, volatile void* address, unsigned int
max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write data to PCI internal from transfer register indirect mode, 32-bit addr.

Table 3.1092. pcie_write_pci_ind_ptr32 parameters

Type	Name	Description
__xwrite void*	<i>data</i>	Data to write to PCIe internal
volatile void*	<i>address</i>	Address to write data to
unsigned int	<i>max_nn</i>	Maximum number of 32-bit words to write (1-16)
generic_ind_t	<i>ind</i>	Indirect word
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)

Type	Name	Description
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.27.1.15 pcie_read_rid_ptr32

Prototype:

```
void pcie_read_rid_ptr32(__xread void* data, unsigned int requester_id, volatile void* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read data from PCIe into transfer registers; overriding the PCIe Requester ID to use for the PCIe transaction, 32-bit addr.

Table 3.1093. pcie_read_rid_ptr32 parameters

Type	Name	Description
__xread void*	<i>data</i>	Address to read data into
unsigned int	<i>requester_id</i>	PCIe requester ID to override
volatile void*	<i>address</i>	Address to read data from (aligned on 4-byte boundary)
unsigned int	<i>count</i>	Number of 32-bit words to read (1-32).
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.27.1.16 pcie_read_rid_ind_ptr32

Prototype:

```
void pcie_read_rid_ind_ptr32(__xread void* data, volatile void* address, unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read 4-byte words from PCIe internal into transfer register, indirect mode, 32-bit addr.

Table 3.1094. pcie_read_rid_ind_ptr32 parameters

Type	Name	Description
__xread void*	<i>data</i>	Address to read data into
volatile void*	<i>address</i>	Address to read from (aligned on 4-byte boundary)
unsigned int	<i>max_nn</i>	Maximum number of 32-bit words to read (1-32)
generic_ind_t	<i>ind</i>	Indirect word
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)

Type	Name	Description
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.27.1.17 pcie_write_rid_ptr32

Prototype:

```
void pcie_write_rid_ptr32(__xwrite void* data, unsigned int requester_id, volatile void* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write data to PCIe from transfer register; overriding the PCIe Requester ID to use for the PCIe transaction, 32-bit addr.

Table 3.1095. pcie_write_rid_ptr32 parameters

Type	Name	Description
__xwrite void*	<i>data</i>	Data to write to PCIe
unsigned int	<i>requester_id</i>	PCIe requester ID to override
volatile void*	<i>address</i>	Address to write data to (aligned on 4-byte boundary)
unsigned int	<i>count</i>	Number of 32-bit words to write (1-32).
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.27.1.18 pcie_write_rid_ind_ptr32

Prototype:

```
void pcie_write_rid_ind_ptr32(__xwrite void* data, volatile void* address, unsigned int max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write data to PCIe from transfer register; overriding the PCIe Requester ID to use for the PCIe transaction, indirect mode, 32-bit addr.

Table 3.1096. pcie_write_rid_ind_ptr32 parameters

Type	Name	Description
__xwrite void*	<i>data</i>	Data to write to PCIe
volatile void*	<i>address</i>	Address to write data to (aligned on 4-byte boundary)
unsigned int	<i>max_nn</i>	Maximum number of 32-bit words to write (1-32).
generic_ind_t	<i>ind</i>	Indirect word
sync_t	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)

Type	Name	Description
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

3.28 ILA Intrinsics

This section describes functions for Interlaken Look-Aside operations.

3.28.1 ILA Enumerations

3.28.1.1 ILA_OPERATION_MODE

Enum for ILA operation mode.

Table 3.1097. enum ILA_OPERATION_MODE

Name	Description
ILA_OPERATION_TCAM	TCAM operations.
ILA_OPERATION_ACCELERATOR	ACCELERATOR operations.

3.28.1.2 ILA_LOGIC_RESET

Enum for reset logic for ILA_OPERATION_MODE setting.

Table 3.1098. enum ILA_LOGIC_RESET

Name	Description
ILA_LOGIC_OUT_OF_RESET	Out of reset.
ILA_LOGIC_IN_RESET	In reset.

3.28.1.3 ILA_CHANNEL

Enum for ILA channels.

Table 3.1099. enum ILA_CHANNEL

Name	Description
ILA_CHANNEL_0	ILA channel 0.
ILA_CHANNEL_1	ILA channel 1.

3.28.2 ILA Unions

3.28.2.1 ila_config_control_register_format_t

Layout of config control register for ILA.

This is used when configuring ILA for TCAM or ACCELERATOR using cluster_target_xpb_write.

Table 3.1100. union ila_config_control_register_format_t

Type	Name	Description
unsigned int	reserved_2:11	Reserved.
unsigned int	rx_fifo_pack:1	ILA RX FIFO packing (0=packing enabled / 1=packing disabled).
unsigned int	tx_idle_count:10	ILA tx idle count.
unsigned int	rx_fifo_size:1	ILA Rx FIFO size (0=4KByte / 1=8KByte).
unsigned int	rx_tx_credit_count:1	ILA Tx credit count enabled (0=disabled / 1=enabled).
unsigned int	tcam_vendor_select:1	TCAM vendor select (0=select NetLogic TCAM / 1=reserved).
unsigned int	reserved_1:1	Reserved.
unsigned int	error_poison_flag:1	Enables poisoning of data on receipt of ILA Rx error (0=disabled/1=enabled).
unsigned int	stats_channel_select:1	Select channel statistics (0=channel 0 / 1=channel 1).
unsigned int	burst_max:1	Maximum size of burst on ILA bus (0=256 bytes / 1=512 bytes).
ILA_LOGIC_RESET	logic_reset_flag:1	Reset to the logic affected by ILA mode setting (0=out of reset / 1=in reset).
ILA_OPERATION_MODE	operation_mode:1	Mode of ILA block (0=TCAM operation / 1= Accelerator operation).
unsigned int	cpp_addr_mode:1	C PP address bit mode (0=40-bits / 1=32-bits).
unsigned int	value	Accessor to entire structure.

3.28.2.2 ila_cpp_to_ilabar_register_format_t

Layout of 32-bit ILA to CPP BAR registers (CPP To ILA BAR Registers).

Table 3.1101. union ila_cpp_to_ilabar_register_format_t

Type	Name	Description
unsigned int	reserved:10	Reserved.
ILA_CHANNEL	channel_select:1	Selects which ILA channel to send the command on.

Type	Name	Description
unsigned int	address:21	In 32-bit mode, ILA address bits [47:27]. In 40-bit mode, ILA address bits [47:35].
unsigned int	value	Accessor to entire structure.

3.28.2.3 ila_dma_command_register_format_t

Layout of 128-bit ILA DMA Command Register.

This breaks down into 4 transfer registers, where xfer[0] == 127:96, xfer[1] == 95:64 etc..

Table 3.1102. union ila_dma_command_register_format_t

Type	Name	Description
unsigned int	xfer_length:12	Length of transfer in bytes. This should be length - 1.
unsigned int	reserved:4	Reserved.
unsigned int	ila_address_hi:16	Upper 16-bits of 48-bit ILA address.
unsigned int	ila_address_lo:30	Lower 30-bits of 48-bit ILA address.
unsigned int	complete_indication_hi:2	Action on completed command (no signal/generate event/send signal to CPP master).
unsigned int	complete_indication_lo:16	Action on completed command (no signal/generate event/send signal to CPP master).
unsigned int	token:2	CPP token bits for CPP command.
unsigned int	signal_both_on_error:1	On ILA Rx Error signal both odd or even (0=disabled / 1=enabled).
unsigned int	cpp_is_target_64:1	Is CPP target 64 bit (0=32-bit / 1=64-bit).
unsigned int	cpp_target:4	Target id for CPP transaction.
unsigned int	cpp_address_hi:8	CPP bus address - higher 8-bit.
unsigned int	cpp_address_lo:32	CPP bus address - lower 32-bit.
unsigned int	value[4]	Accessor to entire structure.

3.28.2.4 ila_internal_address_format_t

Layout of 32-bit address used with ILA internal commands (ila_read_internal_ptr40 and ila_write_internal_ptr40).

Table 3.1103. union ila_internal_address_format_t

Type	Name	Description
unsigned int	reserved:12	
unsigned int	target:4	Internal target number.

Type	Name	Description
unsigned int	address:16	Internal target address.
unsigned int	value	Accessor to entire structure.

3.28.2.5 ila_to_cpp_bar_register_format_t

Layout of 32-bit ILA to CPP BAR registers (ILA To CPP BAR Registers).

Table 3.1104. union ila_to_cpp_bar_register_format_t

Type	Name	Description
unsigned int	reserved_2:2	Reserved.
unsigned int	target_id:5	CPP target id.
unsigned int	token:2	CPP token field.
unsigned int	length_select:1	CPP length field (0=32-bit increments/1=64-bit increments.
unsigned int	reserved_1:22	Reserved.
unsigned int	value	Accessor to entire structure.

3.28.3 ILA Typedefs

3.28.3.1 ILA_OPERATION_MODE

Enum for ILA operation mode.

Table 3.1105. typedef ILA_OPERATION_MODE

Type	Definition
ILA_OPERATION_MODE	enum ILA_OPERATION_MODE

3.28.3.2 ILA_LOGIC_RESET

Enum for reset logic for ILA_OPERATION_MODE setting.

Table 3.1106. typedef ILA_LOGIC_RESET

Type	Definition
ILA_LOGIC_RESET	enum ILA_LOGIC_RESET

3.28.3.3 ILA_CHANNEL

Enum for ILA channels.

Table 3.1107. `typedef ILA_CHANNEL`

Type	Definition
<code>ILA_CHANNEL</code>	<code>enum ILA_CHANNEL</code>

3.28.3.4 `ila_config_control_register_format_t`

Layout of config control register for ILA.

This is used when configuring ILA for TCAM or ACCELERATOR using `cluster_target_xpb_write`.

Table 3.1108. `typedef ila_config_control_register_format_t`

Type	Definition
<code>ila_config_control_register_format_t</code>	<code>union ila_config_control_register_format_t</code>

3.28.3.5 `ila_internal_address_format_t`

Layout of 32-bit address used with ILA internal commands (`ila_read_internal_ptr40` and `ila_write_internal_ptr40`).

Table 3.1109. `typedef ila_internal_address_format_t`

Type	Definition
<code>ila_internal_address_format_t</code>	<code>union ila_internal_address_format_t</code>

3.28.3.6 `ila_to_cpp_bar_register_format_t`

Layout of 32-bit ILA to CPP BAR registers (ILA To CPP BAR Registers).

Table 3.1110. `typedef ila_to_cpp_bar_register_format_t`

Type	Definition
<code>ila_to_cpp_bar_register_format_t</code>	<code>union ila_to_cpp_bar_register_format_t</code>

3.28.3.7 `ila_cpp_to_ilabar_register_format_t`

Layout of 32-bit ILA to CPP BAR registers (CPP To ILA BAR Registers).

Table 3.1111. `typedef ila_cpp_to_ilabar_register_format_t`

Type	Definition
<code>ila_cpp_to_ilabar_register_format_t</code>	<code>union ila_cpp_to_ilabar_register_format_t</code>

3.28.3.8 `ila_dma_command_register_format_t`

Layout of 128-bit ILA DMA Command Register.

This breaks down into 4 transfer registers, where xfer[0] == 127:96, xfer[1] == 95:64 etc..

Table 3.1112. `typedef ila_dma_command_register_format_t`

Type	Definition
<code>ila_dma_command_register_format_t</code>	union <code>ila_dma_command_register_format_t</code>

3.28.4 ILA Functions

3.28.4.1 `ila_write_ptr40`

Prototype:

```
void ila_write_ptr40(__xwrite void* data, volatile void __addr40 __mem* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write data to ILA attached device.

Table 3.1113. `ila_write_ptr40` parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Xfer register(s) containing data to write to memory.
<code>volatile void __addr40 __mem*</code>	<code>address</code>	40 bit address in ILA SRAM to write data to.
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to write (1 - 32).
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (sig_done or ctx_swap).
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion.

3.28.4.2 `ila_read_ptr40`

Prototype:

```
void ila_read_ptr40(__xread void* data, volatile void __addr40 __mem* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read data from ILA attached device.

Table 3.1114. `ila_read_ptr40` parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Xfer register(s) containing data read from memory.

Type	Name	Description
volatile void __addr40 __mem*	address	40 bit address in ILA SRAM to read data from.
unsigned int	count	Number of 32-bit words to read (1 - 32).
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap).
SIGNAL*	sig_ptr	Signal to raise upon completion.

3.28.4.3 ila_write_check_error_ptr40

Prototype:

```
void ila_write_check_error_ptr40(__xwrite void* data, volatile void __addr40 __mem*
address, unsigned int count, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Write data to ILA attached device.

Command will use even signal when write response is received from ILA attached device. If an ILA Rx error is encountered, 4 bytes data is returned with both even and odd signals.

Table 3.1115. ila_write_check_error_ptr40 parameters

Type	Name	Description
__xwrite void*	data	Xfer register(s) containing data to write to memory.
volatile void __addr40 __mem*	address	40 bit address in ILA SRAM to write data to.
unsigned int	count	Number of 32-bit words to write (1 - 32).
sync_t	sync	Type of synchronization to use (can only be sig_done).
SIGNAL_PAIR*	sig_pair_ptr	Signal pair to raise upon completion.

3.28.4.4 ila_read_check_error_ptr40

Prototype:

```
void ila_read_check_error_ptr40(__xread void* data, volatile void __addr40 __mem* address,
unsigned int count, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

Description:

Read data from attached ILA device.

Command will signal back immediate with read response from ILA Target. If an ILA Rx error is encountered, data is returned with both even and odd signal.

Table 3.1116. ila_read_check_error_ptr40 parameters

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Xfer register(s) containing data read from memory.
<code>volatile void __addr40 __mem*</code>	<code>address</code>	40 bit address in ILA SRAM to read data from.
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to read (1 - 32).
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (can only be <code>sig_done</code>).
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal pair to raise upon completion.

3.28.4.5 ila_write_internal_ptr40

Prototype:

```
void ila_write_internal_ptr40(__xwrite void* data, volatile void __addr40 __mem* address,
unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write data to ILA internal target.

Table 3.1117. ila_write_internal_ptr40 parameters

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Xfer register(s) containing data to write to memory.
<code>volatile void __addr40 __mem*</code>	<code>address</code>	40 bit address in ILA where the lower 32 bits are of format <code>ila_internal_address_format_t</code> . See 6xxx databook for recommended addressing mode for address bits.
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to read (1 - 16).
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (<code>sig_done</code> or <code>ctx_swap</code>).
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion.

See Also:

- `ila_internal_address_format_t` for format of 32 bit of the address parameter.

3.28.4.6 ila_read_internal_ptr40

Prototype:

```
void ila_read_internal_ptr40(__xread void* data, volatile void __addr40 __mem* address,
unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read data from ILA internal target.

Table 3.1118. ila_read_internal_ptr40 parameters

Type	Name	Description	
<code>__xread void*</code>	<code>data</code>	Xfer register(s) containing data read from memory.	
<code>volatile void __addr40 __mem*</code>	<code>address</code>	40 bit address in ILA where the lower 32 bits are of format <code>ila_internal_address_format_t</code> . See 6xxx databook for recommended addressing mode for address bits.	32 bit address in ILA of type <code>ila_internal_address_format_t</code> .
<code>unsigned int</code>	<code>count</code>	Number of 32-bit words to read (1 - 16).	
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use (<code>sig_done</code> or <code>ctx_swap</code>).	
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion.	

See Also:

- `ila_internal_address_format_t` for format of 32 bit of the address parameter.

3.29 NBI Intrinsics

This section describes functions for Network Block Interface operations.

3.29.1 NBI Functions

3.29.1.1 nbi_write

Prototype:

```
void nbi_write(__xwrite void* data, volatile void __addr40 __mem* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Write data to NBI.

Table 3.1119. nbi_write parameters

Type	Name	Description	
<code>__xwrite void*</code>	<code>data</code>	Xfer register(s) containing data to write to memory.	

Type	Name	Description
volatile void __addr40 __mem*	address	40 bit address in NBI. See 6xxx databook for recommended addressing mode for address bits.
unsigned int	count	Number of 64-bit words to read (1 - 16).
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap).
SIGNAL*	sig_ptr	Signal to raise upon completion.

3.29.1.2 nbi_read

Prototype:

```
void nbi_read(__xread void* data, volatile void __addr40 __mem* address, unsigned int count, sync_t sync, SIGNAL* sig_ptr)
```

Description:

Read data from ILA internal target.

Table 3.1120. nbi_read parameters

Type	Name	Description
__xread void*	data	Xfer register(s) containing data read from memory.
volatile void __addr40 __mem*	address	40 bit address in NBI. See 6xxx databook for recommended addressing mode for address bits.
unsigned int	count	Number of 64-bit words to read (1 - 16).
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap).
SIGNAL*	sig_ptr	Signal to raise upon completion.

3.30 SRAM Intrinsics

This section describes functions for SRAM. SRAM is deprecated in NFP-6000 but is still supported for backwards compatibility. SRAM is mapped to EMEM0.

FIX ME doxy_SRAM_functions.xml

3.31 Synchronization Intrinsics

This section describes the functions used for synchronization.

3.31.1 Synchronization Enumerations

3.31.1.1 sync_t

Sync types for I/O operations.

Table 3.1121. enum sync_t

Name	Description
sig_done	
ctx_swap	
sig_done	Continue execution and set signal when operation is done.
ctx_swap	Swap thread out and wait until operation is complete.

3.31.1.2 signal_t

Signal types for ctx_wait other than mask.

Table 3.1122. enum signal_t

Name	Description
kill	
voluntary	
bpt	
kill	Suspend the context. The kill signal puts the context into the Sleep state and does not return to the Ready state.
voluntary	Put the context in the sleep state. The voluntary signal puts the context into the Sleep state. The context is put back into the Ready state in one cycle since the Voluntary Event Signal is always set.
bpt	The bpt (breakpoint) stops all contexts, interrupts the ARM processor, and puts the current context into the Sleep state. It also sets the CTX_Enable[Breakpoint] bit. This value is typically used for debugging purposes. For more information the use of this value, refer to the ctx_arb instruction in the <i>Netronome Network Flow Processor 6000 Programmers Reference</i>

3.31.2 Synchronization Structs

3.31.2.1 SIGNAL_PAIR

SIGNAL_PAIR container type for even/odd signal pair.

Table 3.1123. struct SIGNAL_PAIR

Type	Name	Description
int	even	Even signal number.
int	odd	Odd signal number.

3.31.3 Synchronization Typedefs

3.31.3.1 SIGNAL_MASK

SIGNAL_MASK data type is used for masks specifying one or more signal registers.

Table 3.1124. typedef SIGNAL_MASK

Type	Definition
SIGNAL_MASK	int

3.31.3.2 SIGNAL

SIGNAL data type is used to declare signal variables.

Table 3.1125. typedef SIGNAL

Type	Definition
SIGNAL	<code>__declspec(signal) int</code>

3.31.3.3 SIGNAL_PAIR

SIGNAL_PAIR container type for even/odd signal pair.

Table 3.1126. typedef SIGNAL_PAIR

Type	Definition
SIGNAL_PAIR	<code>__declspec(signal_pair) struct SIGNAL_PAIR</code>

3.31.4 Synchronization Functions

3.31.4.1 __signal_number

Prototype:

```
int __signal_number(volatile void* sig,...)
```

Description:

Take address of a signal variable and return the number of the physical signal register allocated to that variable.

This is useful for operations such as setting up of the RX_THREAD_FREELIST CSRs. For signals declared as remote, the `__signal_number()` takes two arguments, the first being the address of the signal and the second being the microengine number in which the remote signal resides.



Note

If the signal is declared as remote, the second argument should be an unsigned integer of the microengine on which it resides.

Table 3.1127. __signal_number parameters

Type	Name	Description
volatile void*	<i>sig</i>	Pointer to a signal variable
...	...	Optional microengine number

3.31.4.2 __wait_for_any

Prototype:

```
void __wait_for_any(...)
```

Description:

Generate a ctx_arb on the set of signals specified by the arguments.

This function take a variable length list of signal pointers, and/or signal masks as arguments and generates a ctx_arb on their union.

The OR form of the ctx_arb is generated will return when any signal is set.



Note

For signal pairs, both even and odd signals are included in the mask used for the ctx_arb. On return, signals also remain set and `signal_test()` is typically used to ensure they are cleared before next use.

Table 3.1128. __wait_for_any parameters

Type	Name	Description
...	...	List of addresses of SIGNAL or SIGNAL_PAIR variables

3.31.4.3 __wait_for_any_single

Prototype:

```
void __wait_for_any_single(...)
```

Description:

Generate a ctx_arb on the set of signals specified by the arguments.

This function take a variable length list of signal pointers, and/or signal masks as arguments and generates a ctx_arb on their union.

The OR form of the ctx_arb is generated will return when any signal is set.



Note

For signal pairs, only the even signals are included in the mask used for the ctx_arb. On return, signals also remain set and `signal_test()` is typically used to ensure they are cleared before next use.

Table 3.1129. __wait_for_any_single parameters

Type	Name	Description
...	...	List of addresses of SIGNAL or SIGNAL_PAIR variables

3.31.4.4 __wait_for_all

Prototype:

```
void __wait_for_all(...)
```

Description:

Generate a ctx_arb on the set of signals specified by the arguments.

This function take a variable length list of signal pointers, and/or signal masks as arguments and generates a ctx_arb on their union.

The AND form of the ctx_arb is generated which will only return when all signals are set.



Note

For signal pairs, both even and odd signals are included in the mask used for the ctx_arb.

Table 3.1130. __wait_for_all parameters

Type	Name	Description
...	...	List of addresses of SIGNAL or SIGNAL_PAIR variables

3.31.4.5 __wait_for_all_single

Prototype:

```
void __wait_for_all_single(...)
```

Description:

Generate a ctx_arb on the set of signals specified by the arguments.

This function take a variable length list of signal pointers, and/or signal masks as arguments and generates a ctx_arb on their union.

The AND form of the ctx_arb is generated which will only return when all signals are set.



Note

For signal pairs, only even signals are included in the mask used for the ctx_arb.

Table 3.1131. __wait_for_all_single parameters

Type	Name	Description
...	...	List of addresses of SIGNAL or SIGNAL_PAIR variables

3.31.4.6 __signals

Prototype:

```
SIGNAL_MASK __signals(...)
```

Description:

This intrinsic creates a signal mask out of a variable length argument list of signal pointers, and/or other signal masks.

Each argument must be one of the following:

- An address of a SIGNAL or SIGNAL_PAIR variable. For signal pairs, both even and odd signals are included in the mask.
- A signal mask.

The returned mask represents the hardware signals allocated to your signal variables. Any number of signals of type SIGNAL between 1 and 15 can be specified.

This function is useful for generating signal masks for use with ctx_arb.



Note

When a user passes in a signal mask to a function as an int parameter, the compiler cannot always figure out what bits are set in the mask and hence does not know the members of the signal set that is represented by the mask. The compiler does not know the life range of these signals represented in such a mask. To get register allocation right, you have to insert calls to implicit_read().

Table 3.1132. __signals parameters

Type	Name	Description
...	...	List of addresses of SIGNAL, SIGNAL_PAIR or signal mask variables

Returns:

A 16 bit signal mask as an int that can be used to generate a ctx_arb either through the __wait_for_all() / __wait_for_all_single() / __wait_for_any() / __wait_for_any_single() intrinsics, or through inline assembly.

3.31.4.7 ctx_wait

Prototype:

```
void ctx_wait(signal_t sig)
```

Description:

Wait for signal.

This function swaps out the current context and waits for the specified signal, bpt, voluntary, kill, or no_load (where the signal mask is specified earlier in one of the local csr CTX_WAKEUP_EVENTS). Please use the __wait_for_all(), __wait_for_all_single(), __wait_for_any(), __wait_for_any_single() or signal_test() intrinsics to wait for signal variables.

Table 3.1133. ctx_wait parameters

Type	Name	Description
signal_t	<i>sig</i>	One of kill, voluntary or bpt.

3.31.4.8 signal_test

Prototype:

```
int signal_test(volatile SIGNAL* sig_ptr)
```

Description:

Test if a signal is set.

This function tests whether or not a signal is set. Signal is a user signal variable.



Note

If the signal is set, it gets cleared as a side effect of testing it.

Table 3.1134. signal_test parameters

Type	Name	Description
volatile SIGNAL*	<i>sig_ptr</i>	Pointer to the signal

Returns:

When it is set a value of 1 is returned otherwise, 0 is returned.

3.31.4.9 signal_same_ME

Prototype:

```
void signal_same_ME(unsigned int sig_no, unsigned int ctx)
```

Description:

Signal the same Microengine, same context.

Raise the specified signal number in the specified context of the same Microengine. The context "ctx" must be running on the same Microengine.

Table 3.1135. signal_same_ME parameters

Type	Name	Description
unsigned int	<i>sig_no</i>	Signal number to raise
unsigned int	<i>ctx</i>	Context in which the signal is to be raised

3.31.4.10 signal_same_ME_next_ctx

Prototype:

```
void signal_same_ME_next_ctx(unsigned int sig_no)
```

Description:

Signal the same Microengine, next context.

Raise the specified signal number in the next context number of the same Microengine.

Table 3.1136. signal_same_ME_next_ctx parameters

Type	Name	Description
unsigned int	<i>sig_no</i>	Signal number to raise

3.31.4.11 signal_prev_ME

Prototype:

```
void signal_prev_ME(unsigned int sig_no, unsigned int ctx)
```

Description:

Signal the previous Microengine.

Raise the specified signal number in the specified context of the previous Microengine.

Table 3.1137. signal_prev_ME parameters

Type	Name	Description
unsigned int	<i>sig_no</i>	Signal number to raise
unsigned int	<i>ctx</i>	Context in which to raise the signal

3.31.4.12 signal_prev_ME_this_ctx

Prototype:

```
void signal_prev_ME_this_ctx(unsigned int sig_no)
```

Description:

Signal the previous Microengine, same context.

Raise the specified signal number in the same context number of the previous Microengine.

Table 3.1138. signal_prev_ME_this_ctx parameters

Type	Name	Description
unsigned int	<i>sig_no</i>	Signal number to raise

3.31.4.13 signal_next_ME

Prototype:

```
void signal_next_ME(unsigned int sig_no, unsigned int ctx)
```

Description:

Signal the next Microengine.

Raise the specified signal number in the specified context of the next Microengine.

Table 3.1139. signal_next_ME parameters

Type	Name	Description
unsigned int	<i>sig_no</i>	Signal number to raise
unsigned int	<i>ctx</i>	Context in which to raise the signal

3.31.4.14 signal_next_ME_this_ctx

Prototype:

```
void signal_next_ME_this_ctx(unsigned int sig_no)
```

Description:

Signal the next Microengine, same context.

Raise the specified signal number in the same context number of the next Microengine.

Table 3.1140. signal_next_ME_this_ctx parameters

Type	Name	Description
unsigned int	<i>sig_no</i>	Signal number to raise

3.32 Math Intrinsics

3.32.1 MATH Functions

3.32.1.1 log2

Prototype:

```
unsigned int log2(uint32_t value)
```

Description:

Calculate the log2 of a 32-bit value.

Table 3.1141. log2 parameters

Type	Name	Description
uint32_t	value	The unsigned 32-bit word to do the log2 on.

Returns:

The result of the log2 value.

3.33 Unaligned Data Access

The functions described in the following sections allow access to data that is not aligned on natural boundaries (32 or 64 bits). They also allow access to data types smaller than those supported by the hardware (8, 16, 32 and 64 bits) at any byte address.

These functions take a two-part address:

- A pointer, aligned or unaligned
- An integer byte offset from the pointer.

For functions that do not specify a particular memory region in its name or argument list, the pointer may point to any memory region. For example, the following function gets a signed 8-bit integer addressed by the void pointer “ptr” and the integer “offset”:

```
int ua_get_s8(void *ptr, unsigned int offset);
```

This function resolves the void pointer to the appropriate memory region (SRAM, MEM, LOCAL_MEM or Cluster Local Scratch). In almost all situations, you should use these general functions to access unaligned data. The function names are shorter and you do not have to specify the memory region in which the data resides.

However, in rare cases, the compiler will be unable to resolve the pointer to a memory region. In these cases, you should use one of the memory-region-qualified unaligned functions. For example:

```
int ua_get_s8_mem(MEM_VOID *p, unsigned int offset);
```

This function resolves the pointer to the MEM memory region and returns a signed 8-bit integer from MEM.

The following sections summarize all unaligned get, set, and memcpy intrinsic functions.

3.33.1 Unaligned Functions

3.33.1.1 ua_get_s8

Prototype:

```
int ua_get_s8(void* p, unsigned int offset)
```

Description:

Get a signed 8-bit integer addressed by the pointer and offset from the appropriate memory region.

Table 3.1142. ua_get_s8 parameters

Type	Name	Description
void*	<i>p</i>	Pointer to location
unsigned int	<i>offset</i>	Offset from pointer location

Returns:

```
*(signed char *)((int)p+offset);
```

3.33.1.2 ua_get_s16

Prototype:

```
int ua_get_s16(void* p, unsigned int offset)
```

Description:

Get a signed 16-bit integer addressed by the pointer and offset from the appropriate memory region.

Table 3.1143. ua_get_s16 parameters

Type	Name	Description
void*	<i>p</i>	Pointer to location
unsigned int	<i>offset</i>	Offset from pointer location

Returns:

```
*(signed short *)((int)p+offset);
```

3.33.1.3 ua_get_s32

Prototype:

```
int ua_get_s32(void* p, unsigned int offset)
```

Description:

Get a signed 32-bit integer addressed by the pointer and offset from the appropriate memory region.

Table 3.1144. ua_get_s32 parameters

Type	Name	Description
void*	<i>p</i>	Pointer to location
unsigned int	<i>offset</i>	Offset from pointer location

Returns:

```
*(int *)((int)p+offset);
```

3.33.1.4 ua_get_s64

Prototype:

```
long long ua_get_s64(void* p, unsigned int offset)
```

Description:

Get a signed 64-bit integer addressed by the pointer and offset from the appropriate memory region.

Table 3.1145. ua_get_s64 parameters

Type	Name	Description
void*	<i>p</i>	Pointer to location
unsigned int	<i>offset</i>	Offset from pointer location

Returns:

```
*(long long *)((int)p+offset);
```

3.33.1.5 ua_get_u8

Prototype:

```
unsigned int ua_get_u8(void* p, unsigned int offset)
```

Description:

Get an unsigned 8-bit integer addressed by the pointer and offset from the appropriate memory region.

Table 3.1146. ua_get_u8 parameters

Type	Name	Description
void*	<i>p</i>	Pointer to location
unsigned int	<i>offset</i>	Offset from pointer location

Returns:

```
*(unsigned char *)((int)p+offset);
```

3.33.1.6 ua_get_u16

Prototype:

```
unsigned int ua_get_u16(void* p, unsigned int offset)
```

Description:

Get an unsigned 16-bit integer addressed by the pointer and offset from the appropriate memory region.

Table 3.1147. ua_get_u16 parameters

Type	Name	Description
void*	<i>p</i>	Pointer to location
unsigned int	<i>offset</i>	Offset from pointer location

Returns:

```
*(unsigned short *)((int)p+offset);
```

3.33.1.7 ua_get_u32

Prototype:

```
unsigned int ua_get_u32(void* p, unsigned int offset)
```

Description:

Get an unsigned 32-bit integer addressed by the pointer and offset from the appropriate memory region.

Table 3.1148. ua_get_u32 parameters

Type	Name	Description
void*	<i>p</i>	Pointer to location
unsigned int	<i>offset</i>	Offset from pointer location

Returns:

```
*(unsigned int *)((int)p+offset);
```

3.33.1.8 ua_get_u64

Prototype:

```
unsigned long long ua_get_u64(void* p, unsigned int offset)
```

Description:

Get an unsigned 64-bit integer addressed by the pointer and offset from the appropriate memory region.

Table 3.1149. ua_get_u64 parameters

Type	Name	Description
void*	<i>p</i>	Pointer to location
unsigned int	<i>offset</i>	Offset from pointer location

Returns:

```
*(unsigned long long *)((int)p+offset);
```

3.33.1.9 ua_set_8

Prototype:

```
void ua_set_8(void* p, unsigned int offset, unsigned int value)
```

Description:

Set an unsigned 8-bit integer addressed by the pointer and offset in the appropriate memory region.

Table 3.1150. ua_set_8 parameters

Type	Name	Description
void*	<i>p</i>	Pointer to location
unsigned int	<i>offset</i>	Offset from pointer location
unsigned int	<i>value</i>	Value to set

3.33.1.10 ua_set_16

Prototype:

```
void ua_set_16(void* p, unsigned int offset, unsigned int value)
```

Description:

Set an unsigned 16-bit integer addressed by the pointer and offset in the appropriate memory region.

Table 3.1151. ua_set_16 parameters

Type	Name	Description
void*	<i>p</i>	Pointer to location
unsigned int	<i>offset</i>	Offset from pointer location
unsigned int	<i>value</i>	Value to set

3.33.1.11 ua_set_32

Prototype:

```
void ua_set_32(void* p, unsigned int offset, unsigned int value)
```

Description:

Set an unsigned 32-bit integer addressed by the pointer and offset in the appropriate memory region.

Table 3.1152. ua_set_32 parameters

Type	Name	Description
void*	<i>p</i>	Pointer to location
unsigned int	<i>offset</i>	Offset from pointer location
unsigned int	<i>value</i>	Value to set

3.33.1.12 ua_set_64

Prototype:

```
void ua_set_64(void* p, unsigned int offset, unsigned long long value)
```

Description:

Set an unsigned 64-bit integer addressed by the pointer and offset in the appropriate memory region.

Table 3.1153. ua_set_64 parameters

Type	Name	Description
void*	<i>p</i>	Pointer to location
unsigned int	<i>offset</i>	Offset from pointer location
unsigned long long	<i>value</i>	Value to set

3.33.1.13 ua_memcpy

Prototype:

```
void ua_memcpy(void* dst, unsigned int dst_off, void* src, unsigned int src_off, unsigned
int length)
```

Description:

Unaligned memory copy.

Copies the number of bytes specified in the length argument from the source to the destination. Both the source and the destination can reside in any memory region.

Table 3.1154. ua_memcpy parameters

Type	Name	Description
void*	<i>dst</i>	Pointer to destination
unsigned int	<i>dst_off</i>	Offset from destination pointer
void*	<i>src</i>	Pointer to source
unsigned int	<i>src_off</i>	Offset from source pointer
unsigned int	<i>length</i>	Number of bytes to copy

3.33.1.14 ua_get_s64_mem

Prototype:

```
long long ua_get_s64_mem(__mem void* p, unsigned int offset)
```

Description:

Get a signed 64-bit integer addressed by the pointer and offset from MEM.

Table 3.1155. ua_get_s64_mem parameters

Type	Name	Description
__mem void*	<i>p</i>	Pointer to MEM location
unsigned int	<i>offset</i>	Offset from pointer location

Returns:

```
*(long long *)((int)p+offset);
```

3.33.1.15 ua_get_u64_cls

Prototype:

```
unsigned long long ua_get_u64_cls(__cls void* p, unsigned int offset)
```

Description:

Get an unsigned 64-bit integer addressed by the pointer and offset from Cluster Local Scratch.

Table 3.1156. ua_get_u64_cls parameters

Type	Name	Description
<code>__cls void*</code>	<code>p</code>	Pointer to Cluster Local Scratch location
<code>unsigned int</code>	<code>offset</code>	Offset from pointer location

Returns:

```
*(unsigned long long *)((int)p+offset);
```

3.33.1.16 ua_memcpy_lmem0_7_sxfer_w_clr

Prototype:

```
void ua_memcpy_lmem0_7_sxfer_w_clr(__lmem void* dst, unsigned int dst_off, unsigned int
ctx, unsigned int sxfer_reg_number, unsigned int src_off, unsigned int length)
```

Description:

Perform an unaligned copy from S-xfer register(s) to an aligned local memory location with dst_off from 0 to 7.

The unwritten trailing and leading bytes written 32-bit words will be zeroed out. All leading unwritten 32-bit words will be zeroed out as well.



Note

Known Issues & limitations (but not enforced):

- This implementation only supports big-endian
- dst_off must be 0..7
- dst must be properly aligned according to dst_off + length You can't safely pass middle of LM array to workaround the range limit of dst_off
- dst_off + length > 64 may not work
- src_off + length can't exceed xfer array

`__implicit_read(&sxfer)` should follow this call.

Table 3.1157. ua_memcpy_lmem0_7_sxfer_w_clr parameters

Type	Name	Description
<code>__lmem void*</code>	<code>dst</code>	Destination in local memory aligned (>=4) (depends on length)
<code>unsigned int</code>	<code>dst_off</code>	0..7 and must be a constant
<code>unsigned int</code>	<code>ctx</code>	0..7 and must be a constant
<code>unsigned int</code>	<code>sxfer_reg_number</code>	Destination xfer register
<code>unsigned int</code>	<code>src_off</code>	Source offset in range of dxfer[]

Type	Name	Description
unsigned int	<i>length</i>	Byte length

3.33.1.17 ua_memcpy_lmem0_7_dxfer_w_clr

Prototype:

```
void ua_memcpy_lmem0_7_dxfer_w_clr(__lmem void* dst, unsigned int dst_off, unsigned int
ctx, unsigned int dxfer_reg_number, unsigned int src_off, unsigned int length)
```

Description:

Perform an unaligned copy from D-xfer register(s) to an aligned local memory location with dst_off from 0 to 7.

The unwritten trailing and leading bytes written 32-bit words will be zeroed out. All leading unwritten 32-bit words will be zeroed out as well.



Note

`_implicit_read(&dxfer)` should follow this call.

Table 3.1158. ua_memcpy_lmem0_7_dxfer_w_clr parameters

Type	Name	Description
<code>__lmem void*</code>	<i>dst</i>	Destination in local memory aligned (>=4) (depends on length)
unsigned int	<i>dst_off</i>	0..7 and must be a constant
unsigned int	<i>ctx</i>	0..7 and must be a constant
unsigned int	<i>dxfer_reg_number</i>	Destination xfer register
unsigned int	<i>src_off</i>	Source offset in range of dxfer[]
unsigned int	<i>length</i>	Byte length

See Also:

- `ua_memcpy_lmem0_7_sxfer_w_clr()`

3.34 Restrictions On Intrinsics

3.34.1 Intrinsic Function Arguments that Map to Transfer Registers in Microcode

The memory and csr intrinsic functions each take an argument that points to a buffer of memory. This buffer is mapped to transfer registers and due to the read-only/write-only restrictions on transfer registers, and due to their

asynchronous update, the compiler restricts their usage. In the following, *xfer buffer address* refers to the buffer pointer that is passed to these intrinsics as an argument whereas *xfer buffer* refers to the memory locations referred to by the *xfer buffer address*. For example, in the following code:

```
{
    __declspec(i24.emem) long long x;
    __declspec(read_reg) long long rd, buf[1];
    SIGNAL sig;

    mem_read64(&rd, &x, 2, ctx_swap, &sig);
    mem_read64(buf, &x, 2, ctx_swap, &sig);
}
```

`rd`, and `buf[]` are the *xfer buffers* and `&rd`, and `buf` are the *xfer buffer addresses*.

`__declspec(read_reg)` and other transfer register data types are used to make you aware of the restrictions listed below.

The following are the restrictions imposed and are illustrated with examples below:

1. The *xfer buffer* argument of a read intrinsic cannot be redefined by any other instruction. However, it is possible to re-use it in another read intrinsic to the compatible memory region if the reads are not asynchronous, or if their lifetimes do not overlap.
2. The *xfer buffer* argument of a write intrinsic cannot be read by any other instruction. However, it is possible to re-use it in another write intrinsic to the compatible memory region if the writes are not asynchronous, or if their lifetimes do not overlap.
3. The *xfer buffer* argument to a read intrinsic cannot be used as the *xfer buffer* argument to a write intrinsic and vice versa.
4. It is in general not permitted to assign or take the address of an xfer buffer except when passing it to the intrinsic. See "Things to Remember When Writing Code" in the *Netronome Network Flow C Compiler User's Guide* for more guidelines.
5. The *xfer buffer address* argument to an intrinsic must be supplied by a direct reference to a buffer variable, and not through a pointer variable.
6. An *xfer buffer* cannot be part of a larger declared aggregate in memory
7. For intrinsics that do an asynchronous write (i.e. not `ctx_swap`), you must mark the spot in the code where the operation has been tested and is known to be complete. This is done by calling the intrinsic function `free_write_buffer(&x)` where `x` is the *xfer buffer* used in the write.
8. Intrinsics that perform asynchronous reads may require the use of `__implicit_read()`. See "Things to Remember When Writing Code" in the *Netronome Network Flow C Compiler User's Guide* for details.
9. Parameter count to intrinsics are preferred to be constant. The compiler may generate an `indirect_ref` with performance loss if it cannot be resolved to a constant at compile-time, or it will error if it's not possible. Parameters `sync` and `csr` must be resolved to be constant at compile-time.

Example

The following example shows violations of these restrictions:

```
{
    __declspec (i24.emem) long long x;
    __declspec (i24.emem) int y;
    int *rdp;
    SIGNAL s;
    __declspec(read_reg) int rd;
    __declspec(write_reg) int wr;
    __declspec(read_reg) int b[2];
    int buf[100];
    int mask;
    ...
    mem_read64(b, &x, 2, sig_done, &s);
    ...
    y=b[i];           // Violates restr 4. Address (b) implicitly
                      // taken since i is not a constant.
    rdp = buf;
    mem_read64(&rd, &y, 1, sig_done, &s);
    mem_read64(rdp, &x, 2, sig_done, &s);           // Violates restr 5.
    mem_write64(&wr, &y, 1,sig_done, &s);
    mem_write64(buf, &x, 2,sig_done, &s);           // Violates restr 6.
    ...
    rd += 1;           // Violates restr 1.
    if (wr)           // Violates restr 2.
    {
        rdp = &rd;
    }
    mem_write64(&rd, &y, 1, sig_done, &s);           // Violates restr 3.
}
}
```

3.35 Miscellaneous Intrinsics

3.35.1 Miscellaneous Enumerations

3.35.1.1 `inp_state_t`

This enumeration defines the state values that can be tested with the `inp_state_test()` intrinsic.

Table 3.1159. enum `inp_state_t`

Name	Description
<code>inp_state_nn_empty</code>	NN_Ring in the neighbor microengine is empty.
<code>inp_state_nn_full</code>	NN_Ring in the neighbor microengine is full.
<code>inp_state_ctm_ring0_status</code>	Indicates if CTM Ring 0 is full or empty. The Full Flag can be configured as an Empty Flag by the RING_STATUS_FLAG bit in the CTM_RING_BASE_0 register.
<code>inp_state_ctm_ring1_status</code>	Indicates if CTM Ring 1 is full or empty.

Name	Description
	The Full Flag can be configured as an Empty Flag by the RING_STATUS_FLAG bit in the CTMRING_BASE_1 register.
inp_state_ctm_ring2_status	Indicates if CTM Ring 2 is full or empty. The Full Flag can be configured as an Empty Flag by the RING_STATUS_FLAG bit in the CTM_RING_BASE_2 register.
inp_state_ctm_ring3_status	Indicates if CTM Ring 3 is full or empty. The Full Flag can be configured as an Empty Flag by the RING_STATUS_FLAG bit in the CTM_RING_BASE_3 register.
inp_state_ctm_ring4_status	Indicates if CTM Ring 4 is full or empty. The Full Flag can be configured as an Empty Flag by the RING_STATUS_FLAG bit in the CTM_RING_BASE_4 register.
inp_state_ctm_ring5_status	Indicates if CTM Ring 5 is full or empty. The Full Flag can be configured as an Empty Flag by the RING_STATUS_FLAG bit in the CTM_RING_BASE_5 register.
inp_state_ctm_ring6_status	Indicates if CTM Ring 6 is full or empty. The Full Flag can be configured as an Empty Flag by the RING_STATUS_FLAG bit in the CTM_RING_BASE_6 register.
inp_state_ctm_ring7_status	Indicates if CTM Ring 7 is full or empty. The Full Flag can be configured as an Empty Flag by the RING_STATUS_FLAG bit in the CTM_RING_BASE_7 register.
inp_state_ctm_ring8_status	Indicates if CTM Ring 8 is full or empty. The Full Flag can be configured as an Empty Flag by the RING_STATUS_FLAG bit in the CTM_RING_BASE_8 register.
inp_state_ctm_ring9_status	Indicates if CTM Ring 9 is full or empty. The Full Flag can be configured as an Empty Flag by the RING_STATUS_FLAG bit in the CTM_RING_BASE_9 register.
inp_state_ctm_ring10_status	Indicates if CTM Ring 10 is full or empty. The Full Flag can be configured as an Empty Flag by the RING_STATUS_FLAG bit in the CTM_RING_BASE_10 register.
inp_state_ctm_ring11_status	Indicates if CTM Ring 11 is full or empty. The Full Flag can be configured as an Empty Flag by the RING_STATUS_FLAG bit in the CTM_RING_BASE_11 register.
inp_state_ctm_ring12_status	Indicates if CTM Ring 12 is full or empty.

Name	Description
	The Full Flag can be configured as an Empty Flag by the RING_STATUS_FLAG bit in the CTM_RING_BASE_12 register.
inp_state_ctm_ring13_status	Indicates if CTM Ring 13 is full or empty. The Full Flag can be configured as an Empty Flag by the RING_STATUS_FLAG bit in the CTM_RING_BASE_13 register.
inp_state_ctm_ring0_full	Indicates if CTM Ring 0 is full. This assumes CTM_Ring_Base_0[Ring_Status_Flag] is configured to set the flag when the ring is full.
inp_state_ctm_ring1_full	Indicates if CTM Ring 1 is full. This assumes CTM_Ring_Base_1[Ring_Status_Flag] is configured to set the flag when the ring is full.
inp_state_ctm_ring2_full	Indicates if CTM Ring 2 is full. This assumes CTM_Ring_Base_2[Ring_Status_Flag] is configured to set the flag when the ring is full.
inp_state_ctm_ring3_full	Indicates if CTM Ring 3 is full. This assumes CTM_Ring_Base_3[Ring_Status_Flag] is configured to set the flag when the ring is full.
inp_state_ctm_ring4_full	Indicates if CTM Ring 4 is full. This assumes CTM_Ring_Base_4[Ring_Status_Flag] is configured to set the flag when the ring is full.
inp_state_ctm_ring5_full	Indicates if CTM Ring 5 is full. This assumes CTM_Ring_Base_5[Ring_Status_Flag] is configured to set the flag when the ring is full.
inp_state_ctm_ring6_full	Indicates if CTM Ring 6 is full. This assumes CTM_Ring_Base_6[Ring_Status_Flag] is configured to set the flag when the ring is full.
inp_state_ctm_ring7_full	Indicates if CTM Ring 7 is full. This assumes CTM_Ring_Base_7[Ring_Status_Flag] is configured to set the flag when the ring is full.
inp_state_ctm_ring8_full	Indicates if CTM Ring 8 is full. This assumes CTM_Ring_Base_8[Ring_Status_Flag] is configured to set the flag when the ring is full.
inp_state_ctm_ring9_full	Indicates if CTM Ring 9 is full.

Name	Description
	This assumes CTM_Ring_Base_9[Ring_Status_Flag] is configured to set the flag when the ring is full.
inp_state_ctm_ring10_full	Indicates if CTM Ring 10 is full. This assumes CTM_Ring_Base_10[Ring_Status_Flag] is configured to set the flag when the ring is full.
inp_state_ctm_ring11_full	Indicates if CTM Ring 11 is full. This assumes CTM_Ring_Base_11[Ring_Status_Flag] is configured to set the flag when the ring is full.
inp_state_ctm_ring12_full	Indicates if CTM Ring 12 is full. This assumes Scratch_Ring_Base_12[Ring_Status_Flag] is configured to set the flag when the ring is full.
inp_state_ctm_ring13_full	Indicates if CTM Ring 13 is full. This assumes CTM_Ring_Base_13[Ring_Status_Flag] is configured to set the flag when the ring is full.

3.35.1.2 swpack_t

Switch annotation.



Note

"Pack" refers to pack case body to the same size so jump[] src can be calculated by zero-based index multiply max-size.

Table 3.1160. enum swpack_t

Name	Description
swpack_none	
swpack_lmem	
swpack_auto	
swpack_1	
swpack_2	
swpack_3	
swpack_4	
swpack_5	
swpack_6	
swpack_7	
swpack_8	

Name	Description
swpack_none	No pack, jump[] to a sequence of branch.
swpack_lmem	No pack, use local memory to hold branch table.
swpack_auto	Auto pack when appropriate.
swpack_1	Pack it, using up to 1 extra instruction to compute new src.
swpack_2	Pack it, using up to 2 extra instructions to compute new src.
swpack_3	Pack it, using up to 3 extra instructions to compute new src.
swpack_4	Pack it, using up to 4 extra instructions to compute new src.
swpack_5	Pack it, using up to 5 extra instructions to compute new src.
swpack_6	Pack it, using up to 6 extra instructions to compute new src.
swpack_7	Pack it, using up to 7 extra instructions to compute new src.
swpack_8	Pack it, using up to 8 extra instructions to compute new src.

3.35.2 Miscellaneous Defines

Table 3.1161. Miscellaneous Defines

Defined	Definition
free_write_buffer(a)	__free_write_buffer(a)
global_label(a)	__global_label(a)
LoadTimeConstant(a)	__LoadTimeConstant(a)
ctx	__ctx()
n_ctx	__n_ctx()
nctx_mode	__nctx_mode()
timer_start(a)	__timer_start(a)
timer_stop(a)	__timer_stop(a)
timer_strobe(a)	__timer_strobe(a)
timer_reset(a)	__timer_reset(a)
timer_report(a)	__timer_report(a)
timer_report_all_ctx(a)	__timer_report_all_ctx(a)
sleep(a)	__sleep(a)
set_timestamp(a)	__set_timestamp(a)
timestamp_start	__timestamp_start()
timestamp_stop(a)	__timestamp_stop(a)
set_profile_count(a)	__set_profile_count(a)
profile_count_start	__profile_count_start()

Defined	Definition
profile_count_stop(a)	<code>__profile_count_stop(a)</code>
wait_for_any	<code>__wait_for_any</code>
wait_for_any_single	<code>__wait_for_any_single</code>
wait_for_all	<code>__wait_for_all</code>
wait_for_all_single	<code>__wait_for_all_single</code>
lmpt_reserve	<code>__lmpt_reserve</code>
i24_emem_base	" <code>__ADDR_I24_EMEM</code> "
i25_emem_base	" <code>__ADDR_I25_EMEM</code> "
i26_emem_base	" <code>__ADDR_I26_EMEM</code> "
offsetof(s,m)	<code>(size_t)&(((s *)0)->m)</code>
ASM_HAS_JUMP	0x0001 Statement attribute.
LITERAL_ASM	0x0002 Statement attribute.
LITERAL_ASM_HAS_JUMP	0x0003 Statement attribute.
NBI0_TARG_ID	1 Two NBI channels.
QDR_TARG_ID	2 Address selects channel which is configurable at boot time.
ILA_TAR_ID	6 Two ILAs.
MEM_TARG_ID	7 Memory units for external DDR3, internal memory and Cluster Target Memory.
PCIE_TARG_ID	8 Two PCIe blocks.
ARM11_TARG_ID	10 ARM target id.
CRYPTO_TARG_ID	12 CRYPTO target id.

Defined	Definition
CLUSTER_TARG_ID	14 Access specific cluster by address.
LOCAL_SCRATCH_TARG_ID	15 Local within cluster.
assert(exp)	(void)((exp) (_assert(#exp, __FILE__, __LINE__), 0))
assert_fast(exp)	(void)((exp) (_assert(0, 0, 0), 0))
NFCC_VERSION(a,b,c)	((a*10000)+(b*100)+c)

3.35.3 Miscellaneous Unions

3.35.3.1 cam_lookup_t

This structure is used to capture the results of a CAM lookup.

`cam_lookup()`, `cam_read_state()`.

Table 3.1162. union cam_lookup_t

Type	Name	Description
unsigned int	zeros1:20	All zeros.
unsigned int	state:4	CAM entry state.
unsigned int	hit:1	hit (1) or miss (0).
unsigned int	entry_num:4	CAM entry number.
unsigned int	zeros2:3	All zeros.
unsigned int	value	Accessor to all fields simultaneously.

See Also:

- `cam_lookup()`, `cam_read_state()`.

3.35.4 Miscellaneous Functions

3.35.4.1 assert_range

Prototype:

```
void assert_range(unsigned int value, unsigned int min_value, unsigned int max_value)
```

Description:

Assert if unsigned value is not between min_value and max_value.

If all of these are constants, a compile time assert is shown otherwise a runtime assert.

Table 3.1163. assert_range parameters

Type	Name	Description
unsigned int	value	Value to verify if within range
unsigned int	min_value	The minimum value of the range
unsigned int	max_value	The maximum value of the range

3.35.4.2 rotr

Prototype:

```
unsigned int rotr(unsigned int value, int shift)
```

Description:

Rotate right a 32-bit word by specified number of bits.

Table 3.1164. rotr parameters

Type	Name	Description
unsigned int	value	The 32-bit value to rotate to the right
int	shift	The number of bits to rotate to the right (1 - 31).

Returns:

The 32-bit result of the value shifted to the right by "shift" bits.

3.35.4.3 rotl

Prototype:

```
unsigned int rotl(unsigned int value, int shift)
```

Description:

Rotate left a 32-bit word by specified number of bits.

Table 3.1165. rotl parameters

Type	Name	Description
unsigned int	value	The 32-bit value to rotate to the left
int	shift	The number of bits to rotate to the left (1 - 31).

Returns:

The 32-bit result of the value shifted to the left by "shift" bits.

3.35.4.4 byte_align_block_be

Prototype:

```
void byte_align_block_be(unsigned int n_byte_align_oper, void* dest, void* src, unsigned shift_cnt)
```

Description:

Byte align block in Big-endian format.

This function sets local_csr BYTE_INDEX to shift_cnt, then performs n_byte_align_oper times of consecutive byte_align_le operations on a pair of 32-bit elements in dest and src. Arguments dest and src are addresses of Xfer/GPR 32-bit variables (or aggregates of 32-bit elements) that must be enregisterized.

Table 3.1166. byte_align_block_be parameters

Type	Name	Description
unsigned int	<i>n_byte_align_oper</i>	Number of byte_align operations to perform on dest and src
void*	<i>dest</i>	Address that stores the results of the alignment shift
void*	<i>src</i>	Address that contains the pair of 32-bit elements to shift
unsigned	<i>shift_cnt</i>	Number of bytes to shift

3.35.4.5 byte_align_block_le

Prototype:

```
void byte_align_block_le(unsigned int n_byte_align_oper, void* dest, void* src, unsigned shift_cnt)
```

Description:

Byte align block in Little-endian format.

This function sets local_csr BYTE_INDEX to shift_cnt, then performs n_byte_align_oper times of consecutive byte_align_le operations on a pair of 32-bit elements in dest and src. Arguments dest and src are addresses of Xfer/GPR 32-bit variables (or aggregates of 32-bit elements) that must be enregisterized.

Table 3.1167. byte_align_block_le parameters

Type	Name	Description
unsigned int	<i>n_byte_align_oper</i>	Number of byte_align operations to perform on dest and src
void*	<i>dest</i>	Address that stores the results of the alignment shift

Type	Name	Description
void*	<i>src</i>	Address that contains the pair of 32-bit elements to shift
unsigned	<i>shift_cnt</i>	Number of bytes to shift

3.35.4.6 halt

Prototype:

```
void halt()
```

Description:

Halt the ME.

3.35.4.7 ffs

Prototype:

```
unsigned int ffs(unsigned int data)
```

Description:

Find the first (least significant) bit set in data.

This function finds the first (least significant) bit set in data and returns its bit position. If there are no bits set (i.e., the data argument is 0) then the return value is undefined. Otherwise, the return value is in the range 0 through 31.

Table 3.1168. ffs parameters

Type	Name	Description
unsigned int	<i>data</i>	Data to examine

3.35.4.8 pop_count

Prototype:

```
unsigned int pop_count(unsigned int data)
```

Description:

Pop the number of "1" bits.

This function returns the number of "1" bits in the given value "data".

Table 3.1169. pop_count parameters

Type	Name	Description
unsigned int	<i>data</i>	Value on which to perform the operation

3.35.4.9 bswap

Prototype:

```
unsigned int bswap(unsigned int lw)
```

Description:

Swap bytes in a long word.

Table 3.1170. bswap parameters

Type	Name	Description
unsigned int	<i>lw</i>	Long word to swap bytes in

3.35.4.10 bitswap

Prototype:

```
unsigned int bitswap(unsigned int lw)
```

Description:

Swap bits in each byte of a long word.

Table 3.1171. bitswap parameters

Type	Name	Description
unsigned int	<i>lw</i>	Long word where bits in bytes are swapped

3.35.4.11 cam_write

Prototype:

```
void cam_write(unsigned int entry_num, unsigned int tag, unsigned int cam_state)
```

Description:

Write an entry in the CAM.

Writes an entry in the CAM specified by the argument index with the value specified by tag, and sets the state to the value specified in the argument state. Argument state must be a constant literal specified directly in the intrinsics argument list. Otherwise, the compiler may have to generate runtime checks for the possible 16 values, since the microcode only accepts a constant literal for the state.

Table 3.1172. cam_write parameters

Type	Name	Description
unsigned int	<i>entry_num</i>	CAM entry number to write
unsigned int	<i>tag</i>	Value to set for this CAM entry
unsigned int	<i>cam_state</i>	State to set for this CAM entry

3.35.4.12 cam_lookup

Prototype:

```
cam_lookup_t cam_lookup(unsigned int tag)
```

Description:

Perform a CAM lookup.

Perform a CAM lookup and return the hit/miss status, state, and entry number as bitfields in the return value. In the event of a miss, the entry value is the LRU (least recently used) entry (which is the suggested entry to replace) and state bits are 0. On a CAM hit, this function has the side effect of marking the CAM entry as MRU (most recently used).

Table 3.1173. cam_lookup parameters

Type	Name	Description
unsigned int	<i>tag</i>	The value to lookup in the CAM

3.35.4.13 cam_write_state

Prototype:

```
void cam_write_state(unsigned int entry_num, unsigned int cam_state)
```

Description:

Set the state for the entry in the CAM.

Set the state for the entry in the CAM specified by the *entry_num* argument to the value specified in the argument *state*. Argument *state* must be a constant literal specified directly in the intrinsics argument list. Otherwise, the compiler may have to generate runtime checks for the possible 16 values, since the microcode only accepts a constant literal for the state.

Table 3.1174. cam_write_state parameters

Type	Name	Description
unsigned int	<i>entry_num</i>	CAM entry whose state is set
unsigned int	<i>cam_state</i>	State to set for the CAM entry

3.35.4.14 cam_read_tag

Prototype:

```
unsigned int cam_read_tag(unsigned int entry_num)
```

Description:

Read the tag associated with the CAM entry.

Read out the tag associated with the CAM entry specified by the entry_num argument.

Table 3.1175. cam_read_tag parameters

Type	Name	Description
unsigned int	entry_num	CAM entry whose tag is returned

3.35.4.15 cam_read_state

Prototype:

```
cam_lookup_t cam_read_state(unsigned int entry_num)
```

Description:

Read the state associated with the CAM entry.

Read out the state associated with the CAM entry specified by the entry_num argument and returns it in the state bitfield of the return value. All other fields of the return value structure are set to 0.

Table 3.1176. cam_read_state parameters

Type	Name	Description
unsigned int	entry_num	CAM entry whose state is returned

3.35.4.16 cam_clear

Prototype:

```
void cam_clear(void)
```

Description:

Clear all entries in the CAM.

3.35.4.17 multiply_24x8

Prototype:

```
unsigned int multiply_24x8(unsigned int x, unsigned int y)
```

Description:

Multiply 24-bit by 8-bit.

Returns the result of 24-bit x multiplied by 8-bit y.

Table 3.1177. multiply_24x8 parameters

Type	Name	Description
unsigned int	x	24-bit int to multiply
unsigned int	y	8-bit int to multiply

3.35.4.18 multiply_16x16

Prototype:

```
unsigned int multiply_16x16(unsigned int x, unsigned int y)
```

Description:

Multiply 16-bit by 16-bit.

Returns the result of 16-bit x multiplied by 16-bit y.

Table 3.1178. multiply_16x16 parameters

Type	Name	Description
unsigned int	x	16-bit int to multiply
unsigned int	y	16-bit int to multiply

3.35.4.19 multiply_32x32_lo

Prototype:

```
unsigned int multiply_32x32_lo(unsigned int x, unsigned int y)
```

Description:

Multiply 32-bit by 32-bit.

Returns the lower 32-bit result of 32-bit x multiplied by 32-bit y.

Table 3.1179. multiply_32x32_lo parameters

Type	Name	Description
unsigned int	x	32-bit int to multiply

Type	Name	Description
unsigned int	y	32-bit int to multiply

3.35.4.20 multiply_32x32_hi

Prototype:

```
unsigned int multiply_32x32_hi(unsigned int x, unsigned int y)
```

Description:

Multiply 32-bit by 32-bit.

Returns the higher 32-bit result of 32-bit x multiplied by 32-bit y.

Table 3.1180. multiply_32x32_hi parameters

Type	Name	Description
unsigned int	x	32-bit int to multiply
unsigned int	y	32-bit int to multiply

3.35.4.21 multiply_32x32

Prototype:

```
unsigned long long multiply_32x32(unsigned int x, unsigned int y)
```

Description:

Multiply 32-bit by 32-bit.

Returns the higher 32-bit result of 32-bit x multiplied by 32-bit y.

Table 3.1181. multiply_32x32 parameters

Type	Name	Description
unsigned int	x	32-bit int to multiply
unsigned int	y	32-bit int to multiply

3.35.4.22 __set_timestamp

Prototype:

```
void __set_timestamp(__int64 timestamp)
```

Description:

Sets the timestamp for local CSR enum.

Sets both the local_csr_timestamp_low and local_csr_timestamp_high fields of the local_csr_t enum.

Table 3.1182. __set_timestamp parameters

Type	Name	Description
__int64	<i>timestamp</i>	Timestamp to set

3.35.4.23 __timestamp_start

Prototype:

```
__int64 __timestamp_start(void)
```

Description:

Start timestamp.

This function uses local_csr_timestamp_low and timestamp_high to measure time elapse in 16-cycle intervals between start and stop. The __timestamp_start() function returns a handle that is used by the __timestamp_stop() function.



Note

This function returns a handle that is used by the __timestamp_stop() function.

3.35.4.24 __timestamp_stop

Prototype:

```
__int64 __timestamp_stop(__int64 handle)
```

Description:

Stop timestamp.

This function returns the time elapse in 16-cycle intervals between start and stop.

Table 3.1183. __timestamp_stop parameters

Type	Name	Description
__int64	<i>handle</i>	Timestamp handle returned by the __timestamp_start() function

3.35.4.25 __sleep

Prototype:

```
void __sleep(unsigned int cycles)
```

Description:

Sleep for a number of cycles.



Note

The ME timers must be enabled before using the sleep function.

Table 3.1184. __sleep parameters

Type	Name	Description
unsigned int	<i>cycles</i>	Approximate number of cycles to sleep, resolution is 16 cycles and should be less than $(1 \ll 20)$.

3.35.4.26 __island

Prototype:

```
unsigned int __island(void)
```

Description:

Get the current island/cluster id of the current Microengine.

Also available for use is LoadTimeConstant("__island");

Returns:

The island/cluster id of the currently executing microengine

See Also:

- __LoadTimeConstant

3.35.4.27 __ME

Prototype:

```
unsigned int __ME(void)
```

Description:

Get the current Microengine number and island id.

Also available for use LoadTimeConstant("__meid");

Returns:

The value of the currently executing microengine. (`island_id << 4`) | (`me_num + 4`)

See Also:

- `__LoadTimeConstant`

3.35.4.28 __ctx

Prototype:

```
unsigned int __ctx(void)
```

Description:

Get the current context.

Returns:

The value of the currently executing context in the range of 0 through 7.

3.35.4.29 __n_ctx

Prototype:

```
unsigned int __n_ctx(void)
```

Description:

Get the number of contexts compiled.

This is specified by the nfcc option "-Qnctx=<1, 2, 3, 4, 5, 6, 7, 8>"

Returns:

The number of contexts compiled to run, the range is 1 to 8.

3.35.4.30 __nctx_mode

Prototype:

```
unsigned int __nctx_mode(void)
```

Description:

Get the context mode.

This is specified by the nfcc option "-Qnctx_mode=<4, 8>"

Returns:

Depending on context mode, either 4 or 8.

3.35.4.31 nn_ring_dequeue

Prototype:

```
unsigned int nn_ring_dequeue(void)
```

Description:

Dequeue from next neighbor ring.

This function returns the next neighbor ring indexed by NN_GET without post-incrementing NN_GET.

Returns:

Next neighbor ring

3.35.4.32 nn_ring_dequeue_incr

Prototype:

```
unsigned int nn_ring_dequeue_incr(void)
```

Description:

Dequeue from and post-increment next neighbor ring.

This function returns the next neighbor ring indexed by NN_GET, then post-increments NN_GET.

Returns:

Next neighbor ring

3.35.4.33 nn_ring_enqueue_incr

Prototype:

```
void nn_ring_enqueue_incr(unsigned int val)
```

Description:

Enqueue to the next neighbor ring.

This function sets the next neighbor ring indexed by NN_PUT with val and post-increments NN_PUT.

Table 3.1185. nn_ring_enqueue_incr parameters

Type	Name	Description
unsigned int	val	Value to set next neighbor ring indexed by NN_PUT

3.35.4.34 __assert

Prototype:

```
__noinline void __assert(char* s, char* f, unsigned l)
```

Description:

Assert.

3.35.4.35 __xfer_reg_number

Prototype:

```
int __xfer_reg_number(volatile void* xfer, ...)
```

Description:

Take the address of a transfer register variable and return the number of the transfer register allocated to that variable.

This is useful for operations such as setting up of the RX_THREAD_FREELIST CSRs. For transfer registers declared as remote, the `__xfer_reg_number()` intrinsic takes two arguments, the first being the address of the transfer register and the second being the microengine number in which the transfer register resides.

Table 3.1186. __xfer_reg_number parameters

Type	Name	Description
volatile void*	xfer	Pointer to a transfer register
...	...	If the signal is declared as remote, the second argument should be an unsigned integer of the microengine on which it resides.

3.35.4.36 __implicit_read

Prototype:

```
void __implicit_read(void* sig_or_xfer, ...)
```

Description:

Take the address of a signal or transfer register variable and indicates that the signal or transfer register is being read asynchronously or implicitly by the hardware.

This function takes as argument the address of a register variable and indicates that the signal or transfer register is being read asynchronously or implicitly by the hardware. It is necessary to use this intrinsic to mark all definitions and use points of signal/transfer registers that are not directly visible to the compiler. They ensure that the compiler does correct lifetime analysis and hence correct register allocation to such variables. This intrinsic must be used, for example, when the RX_THREAD_FREELIST CSR is written with the register number allocated to a variable,

when a signal is requested by writing into a CSR, or a signal is tested by doing a ctx_arb with a signal mask generated with `__signals()`.



Note

Performs the same task as `__free_write_buffer()`.

Table 3.1187. __implicit_read parameters

Type	Name	Description
<code>void*</code>	<code>sig_or_xfer</code>	Pointer to a signal or transfer register address
...	...	Integer argument that specifies the size in bytes of the read. This can be used to target specific elements of a structure or array.

3.35.4.37 __implicit_write

Prototype:

```
void __implicit_write(void* sig_or_xfer,...)
```

Description:

Take the address of a signal or transfer register variable and indicate that the signal or transfer register is being written asynchronously or implicitly by the hardware.

This function takes as argument the address of a signal or transfer register variable and indicates that the signal or transfer register is being written asynchronously or implicitly by the hardware. It is necessary to use this intrinsic to mark all definitions and use points of signal/transfer registers that are not directly visible to the compiler. They ensure that the compiler does correct lifetime analysis and hence correct register allocation to such variables. This intrinsic must be used, for example, when the RX_THREAD_FREELIST CSR is written with the register number allocated to a variable, when a signal is requested by writing into a CSR, or a signal is tested by doing a ctx_arb with a signal mask generated with `__signals()`. See "Things to Remember When Writing Code" in the *Netronome Network Flow C Compiler User's Guide* for a more complete explanation with examples.

Table 3.1188. __implicit_write parameters

Type	Name	Description
<code>void*</code>	<code>sig_or_xfer</code>	Pointer to a signal or transfer register address
...	...	Integer argument that specifies the size in bytes of the write. This can be used to target specific elements of a structure or array.

3.35.4.38 __implicit_undef

Prototype:

```
void __implicit_undef(void* sig_or_xfer,...)
```

Description:

Indicate that the given variable should become undefined after this program point and its register(s) can be released.

This function takes as an argument the address of a register variable, and indicates to the compiler that the variable can be considered undefined until the next time it is written to, even if the variable is read. Undefined variables do not require registers. `__implicit_undef()` and `__implicit_write()` are similar in that they can be used to prevent the compiler from storing a prior value of the variable. Unlike `__implicit_write()`, however, `__implicit_undef()` will not cause the compiler to reserve registers for the variable after the call.

Example:

```
while (1) {
    __implicit_undef(&x);
    ...
    if (!error) {
        x = 1;
    }
    sram_write(&x,...);
}
```

In this example the `__implicit_undef()` will guarantee that registers will not be reserved for "x" in the area marked "Other code". Without it, the compiler has to assume that the value of "x" from a previous iteration of the loop might be used in the current iteration, since "error" might be true. Registers would then have to be reserved for "x" throughout the entire loop. With the `__implicit_undef()` call, the compiler will see that "x" starts with an undefined value in every loop iteration, and the previous value will not be needed. `__implicit_write()` could also be used, but it would have to be placed directly before the "if (!error)" statement, since it is equivalent to writing a new value to "x" which has to be preserved by the compiler in a register. `__implicit_undef()`, in contrast, "overwrites" any previous value of the variable but does not "write" a new value.

Table 3.1189. `__implicit_undef` parameters

Type	Name	Description
void*	<code>sig_or_xfer</code>	Pointer to signal or transfer register variables
...	...	Optional argument to specify the size of the area in bytes to be marked undefined

3.35.4.39 `__assign_relative_register`

Prototype:

```
void __assign_relative_register(void* sig_or_reg, int reg_num)
```

Description:

Take the address of a signal variable or transfer register variable and binds it to a physical register number.

This function takes the address of a signal variable or register variable, and binds it to a physical register number. If the address of a structure or array is passed, the first element of the structure or array will be assigned the physical register number, and every successive element will be assigned a consecutive register number.

Table 3.1190. __assign_relative_register parameters

Type	Name	Description
void*	<i>sig_or_reg</i>	Address of a signal variable or transfer register variable
int	<i>reg_num</i>	Physical register number to bind variable to

3.35.4.40 __no_spill_begin

Prototype:

```
void __no_spill_begin(void)
```

Description:

Mark the beginning of a "no-spill" program region, where the compiler attempts to keep all the used variables in registers unless they have been explicitly allocated to memory or have had their address taken.

This is done at the expense of other program regions, which may incur extra spills. If the compiler cannot allocate all the variables to registers, compilation will halt with an error message.

3.35.4.41 __no_spill_end

Prototype:

```
void __no_spill_end(void)
```

Description:

Mark the end of a "no-spill" program region.

See Also:

- `__no_spill_begin()`

3.35.4.42 __no_swap_begin

Prototype:

```
void __no_swap_begin(void)
```

Description:

Mark the start of a no swap region.

Create a region where the compiler will not generate any instructions that will incur a context swap.

3.35.4.43 __no_swap_end

Prototype:

```
void __no_swap_end(void)
```

Description:

Mark the end of a no swap region.

End of a no swap region.

See Also:

- `__no_swap_begin()`

3.35.4.44 __set_profile_count

Prototype:

```
void __set_profile_count(unsigned int profile_count)
```

Description:

Sets the local_csr_profile_count field of the local_csr_t enum.

Table 3.1191. __set_profile_count parameters

Type	Name	Description
unsigned int	<i>profile_count</i>	Profile count to set.

3.35.4.45 __profile_count_start

Prototype:

```
unsigned int __profile_count_start(void)
```

Description:

Uses local_csr_profile_count to measure time elapse in cycle between start and stop.

The `__profile_count_start()` function returns a handle that is used by the `__profile_count_stop()` function.

See Also:

- `__profile_count_stop`

3.35.4.46 __profile_count_stop

Prototype:

```
unsigned int __profile_count_stop(unsigned int handle)
```

Description:

This function use local_csr_profile_count to measure time elapse in cycle between start and stop.

The __profile_count_start() function returns a handle that is used by the __profile_count_stop() function.

Table 3.1192. __profile_count_stop parameters

Type	Name	Description
unsigned int	<i>handle</i>	Handle returned by the __profile_count_start() function that is passed to the __profile_count_stop() function.

See Also:

- __profile_count_start

3.35.4.47 bit_test

Prototype:

```
int bit_test(unsigned int data, unsigned int bit_pos)
```



Note

No description!

3.35.4.48 inp_state_test

Prototype:

```
int inp_state_test(inp_state_t state)
```

Description:

Tests the value of the specified input state name.

This function tests the value of the specified state name and returns a 1 if the state is set or 0 if clear. The argument state must be a constant literal as required by the microcode assembler; otherwise, the compiler generates a runtime check, if possible, with loss of performance.

Table 3.1193. inp_state_test parameters

Type	Name	Description
inp_state_t	<i>state</i>	State to test

3.35.4.49 __free_write_buffer

Prototype:

```
void __free_write_buffer(void* data)
```

Description:

Indicates that any pending write that writes out the data buffer can be considered as having completed.

The transfer registers allocated for the write can now be used for other write operations.

This intrinsic is called after an asynchronous memory write operation has been issued. It indicates that all pending writes that require the given data buffer have completed. The transfer registers allocated for the write can then be reused for other write operations. Without a call to this intrinsic, the compiler will assume that transfer registers involved in an asynchronous memory write can be reused immediately after the operation has issued, which may cause invalid data to be written out to memory. See "Things to Remember When Writing Code" in the *Netronome Network Flow C Compiler User's Guide* for a more complete explanation with examples.



Note

Performs the same task as `__implicit_read()`.

Table 3.1194. `__free_write_buffer` parameters

Type	Name	Description
<code>void*</code>	<code>data</code>	Data buffer to write

3.35.4.50 `__global_label`

Prototype:

```
void __global_label(char* label)
```

Description:

Creates a named global label at the point the intrinsic is defined.

This function creates a named global label at the point the intrinsic is defined.

Examples:

1. This example creates a label named `ixp_start_packet_count` at the intrinsic invocation point. The name of the label is exactly as specified (i.e., no `_` is prepended to the name). The name of the label must be unique. The label does not interact with other C labels since C labels are renamed.

```
global_label("ixp_start_packet_count");
```

2. This fragment creates two different labels -- one global LABEL# and some other option/ code specific label, such as `1_345#`.

```
global_label(LABEL);
...
LABEL:
```

Table 3.1195. __global_label parameters

Type	Name	Description
char*	<i>label</i>	Name of the label to create

3.35.4.51 __LoadTimeConstant

Prototype:

```
long long __LoadTimeConstant(char* const_name)
```

Description:

Causes a string constant (name) to be associated with an integer value at load time (using nfld) and returns the integer.

Only 48-bit values are supported with this function.

This function associates a constant name with an integer value at load time using nfld and returns the integer. This provides equivalent functionality as the .import_var using assembler. There is no register or memory allocated to this constant. The name argument uniquely identifies the constant.

A number of predefined import variables are supported by the linker. These are fully resolved at link time. These are also documented in the assembler user guide.

- __meid, __uengine_id : A microengine ID, format depends on chip family,
 1. For NFP-32xx: (cluster_num << 4) | (me_num) == __nfp_meid(cluster_num, menum)
 2. For NFP-6xxx: (island_id << 4) | (me_num + 4) == __nfp_meid(island_id, menum)
- __island: Island/Cluster number
- __menum: Zero-based microengine number within island.
- __addr_<id> : A series of predefined import variables that resolve to the base address of the <id> memory resource or target from the island the list file is assigned to. Values for <id> are the same as memory types used in __declspec, with dots replaced by underscores, and some additions for non-memory targets. For example:
 - __addr_emem0, __addr_i24_emem : Both base of emem0.
 - __addr_imem0, __addr_i28_imem : Both base of imem0.
 - __addr_iX_ctm : Base of CTM in island iX.
 - __addr_ctm : Base of CTM in local island. This is 0 for the default SDK IMB CPP Address Translation Tables.
 - __addr_iX_cls : Same as iX_ctm, but for CLS.
 - __addr_cls : Same as ctm, but for CLS.
 - __addr_i4_pc当地 : Both resolve to address 0 of PCIE island 4, to be used with pcie[] instructions.
 - __addr_i8_nbi, __addr_nbi0 : Same as above, but for nbi[] instructions targeting NBI islands.
 - __addr_i48_il当地 , __addr_il0 : Same as above, for ila[] instructions and ILA islands.
 - __addr_arm : For arm[] instructions targeting the ARM island.

Example:

In the following example, A1 and A2 have the same constant, which may or may not be the same as the independent constant for B.

```
A1 = __LoadTimeConstant( "CONST_A" );
A2 = __LoadTimeConstant( "CONST_A" );
B = __LoadTimeConstant( "CONST_B" );
```

Table 3.1196. __LoadTimeConstant parameters

Type	Name	Description
char*	<i>const_name</i>	Name of a constant to bound at load time

3.35.4.52 __link_sym

Prototype:

```
long long __link_sym(const char* id)
```

Description:

This function provides generic access to linker symbols.

Table 3.1197. __link_sym parameters

Type	Name	Description
const char*	<i>id</i>	Link-time symbol

3.35.4.53 __alloc_resource

Prototype:

```
unsigned long long __alloc_resource(const char* argstr)
```

Description:

This function provides equivalent functionality to .alloc_resource in the assembler.

Table 3.1198. __alloc_resource parameters

Type	Name	Description
const char*	<i>argstr</i>	Argument string

3.35.4.54 __critical_path

Prototype:

```
void __critical_path(...)
```

Description:

Marks a section of code as being on the critical path of the application.

Takes an optional integer argument (0-100) that specifies the priority of overlapping critical paths. Higher numbered paths receive higher priority for code layout. The default is 100 if no argument is specified.

Table 3.1199. __critical_path parameters

Type	Name	Description
...	...	Optional argument indicates priority in 0-100. Default is 100.

3.35.4.55 __impossible_path

Prototype:

```
void __impossible_path(void)
```

Description:

Ask the compiler to perform the default case removal optimization.

This function should only be placed in the "default" handler of a switch() statement.

3.35.4.56 __switch_pack

Prototype:

```
void __switch_pack(swpack_t swpack, ...)
```

Description:

Ask the compiler to perform the switch block packing optimization.

This function should only be placed in the "default" handler of a switch().

Table 3.1200. __switch_pack parameters

Type	Name	Description
swpack_t	<i>swpack</i>	Type of the switch packing optimization desired.
...	...	Integer max_n_nop which defaults to -1 (auto, up to compiler)

4. Technical Support

To obtain additional information, or to provide feedback, please email <support@netronome.com> or contact the nearest **Netronome** technical support representative.