



# **Netronome Network Flow Processor 6xxx**

**NFP SDK version 6.1 Preview**

## **Micro-C Standard Library Reference Manual**

**- Proprietary and Confidential -**

**b3286.dr7089**

**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

<b>1. Introduction .....</b>	31
1.1. About This Guide .....	31
1.2. Related Documentation .....	31
1.3. Conventions Used in this Manual .....	31
<b>2. LibC Library Functions .....</b>	33
2.1. String Literals .....	33
2.2. stdlib.h .....	34
2.3. nfptypes.h .....	34
2.3.1. Basic Types .....	34
2.3.2. Memory Region Types .....	35
2.4. memory.h .....	35
2.4.1. memcpy() .....	35
2.4.2. memmove() .....	38
2.4.3. memset() .....	39
2.4.4. memcmp() .....	40
2.4.5. memchr() .....	42
2.5. string.h .....	43
2.5.1. strlen() .....	43
2.5.2. strcmp() .....	43
2.5.3. strncmp() .....	44
2.5.4. strcpy() .....	45
2.5.5. strncpy() .....	46
2.5.6. strcat() .....	46
2.5.7. strncat() .....	47
2.5.8. strchr() .....	48
2.5.9. strrchr() .....	49
2.5.10. strstr() .....	49
2.5.11. strspn() .....	50
2.5.12. strcspn() .....	51
2.5.13. strpbrk() .....	51
2.5.14. strtok() .....	52
2.5.15. strtol() .....	53
2.5.16. strtoul() .....	54
<b>3. Intrinsic Functions .....</b>	55
3.1. Intrinsic Syntax Conventions .....	56
3.1.1. Transfer Register Modifiers .....	56
3.1.2. Limitations on Some I/O Functions .....	57
3.2. NFP-32xx Indirect Reference Formats .....	58
3.3. NFP-6xxx Indirect Reference Formats .....	58
3.3.1. Indirect Reference 6xxx Unions .....	58
3.4. NFP Indirect Reference in General .....	59
3.4.1. Indirect Reference General Enumerations .....	59
3.4.2. Indirect Reference 6xxx Structs .....	60
3.4.3. Indirect Reference General Functions .....	62
3.5. ARM Intrinsics .....	64
3.5.1. ARM Functions .....	64

3.6. Cluster Local Scratch Intrinsics .....	66
3.6.1. Cluster Local Scratch Enumerations .....	66
3.6.2. Cluster Local Scratch Typedefs .....	67
3.6.3. Cluster Local Scratch Functions .....	67
3.7. Cluster Local Scratch Ring Intrinsics .....	189
3.7.1. CLS Ring Enumerations .....	190
3.7.2. CLS Ring Typedefs .....	192
3.7.3. CLS Ring Functions .....	192
3.8. Cluster Local Scratch Reflect Intrinsics .....	215
3.8.1. CLS Reflect Functions .....	215
3.9. Cluster Target Intrinsics .....	231
3.9.1. Cluster Target Enumerations .....	231
3.9.2. Cluster Target Unions .....	232
3.9.3. Cluster Target Typedefs .....	234
3.9.4. Cluster Target Functions .....	236
3.10. Control and Status Register Access Intrinsics .....	250
3.10.1. Control and Status Register Access Enumerations .....	251
3.10.2. Control and Status Register Access Unions .....	253
3.11. CRC Intrinsics .....	254
3.11.1. CRC Enumerations .....	254
3.11.2. CRC Functions .....	255
3.12. Crypto Intrinsics .....	269
3.12.1. Crypto Functions .....	269
3.13. MEM Intrinsics .....	272
3.14. MEM WorkQ Intrinsics .....	272
3.14.1. MU WorkQ Functions .....	273
3.15. MEM Ring Intrinsics .....	275
3.15.1. MU Ring Defines .....	275
3.15.2. MU Ring Enumerations .....	275
3.15.3. MU Ring Structs .....	276
3.15.4. MU Ring Unions .....	276
3.15.5. MU Ring Typedefs .....	277
3.15.6. MU Ring Functions .....	278
3.16. MEM Ticket Intrinsics .....	292
3.16.1. MU Ticket Structs .....	292
3.16.2. MU Ticket Typedefs .....	293
3.16.3. MU Ticket Functions .....	294
3.17. MEM Queue Lock Intrinsics .....	298
3.17.1. MU Queue Lock Structs .....	298
3.17.2. MU Queue Lock Unions .....	299
3.17.3. MU Queue Lock Typedefs .....	300
3.17.4. MU Queue Lock Functions .....	302
3.18. MEM Lock Intrinsics .....	310
3.18.1. MU Lock Structs .....	310
3.18.2. MU Lock Unions .....	311
3.18.3. MU Lock Typedefs .....	312
3.18.4. MU Lock Functions .....	314
3.19. MEM List Intrinsics .....	327
3.19.1. MU List Unions .....	327
3.19.2. MU List Typedefs .....	332
3.19.3. MU List Functions .....	337
3.20. MEM MicroQ Intrinsics .....	353
3.20.1. MU MicroQ Enumerations .....	353

3.20.2. MU MicroQ Structs .....	354
3.20.3. MU MicroQ Unions .....	354
3.20.4. MU MicroQ Typedefs .....	355
3.20.5. MU MicroQ Functions .....	357
3.21. MEM CAM IntrinsicS .....	366
3.21.1. MU CAM Structs .....	366
3.21.2. MU CAM Unions .....	367
3.21.3. MU CAM Typedefs .....	373
3.21.4. MU CAM Functions .....	381
3.22. MEM TCAM IntrinsicS .....	453
3.22.1. MU TCAM Defines .....	453
3.22.2. MU TCAM Structs .....	460
3.22.3. MU TCAM Unions .....	461
3.22.4. MU TCAM Typedefs .....	464
3.22.5. MU TCAM Functions .....	469
3.23. MEM Packet Engine IntrinsicS .....	555
3.23.1. MU PE Enumerations .....	555
3.23.2. MU PE Structs .....	556
3.23.3. MU PE Unions .....	557
3.23.4. MU PE Typedefs .....	559
3.23.5. MU PE Functions .....	561
3.24. MEM Statistics IntrinsicS .....	582
3.24.1. MU Statistics Defines .....	585
3.24.2. MU Statistics Enumerations .....	587
3.24.3. MU Statistics Unions .....	587
3.24.4. MU Statistics Typedefs .....	588
3.24.5. MU Statistics Functions .....	590
3.25. MEM Load Balancing IntrinsicS .....	593
3.25.1. MU LB Defines .....	594
3.25.2. MU LB Unions .....	595
3.25.3. MU LB Typedefs .....	598
3.25.4. MU LB Functions .....	601
3.26. MEM Lookup IntrinsicS .....	615
3.26.1. MU Lookup Structs .....	618
3.26.2. MU Lookup Enumerations .....	618
3.26.3. MU Lookup Unions .....	623
3.26.4. MU Lookup Typedefs .....	626
3.26.5. MU Lookup Functions .....	630
3.27. PCI Express IntrinsicS .....	631
3.27.1. PCI Express Functions .....	631
3.28. ILA IntrinsicS .....	639
3.28.1. ILA Enumerations .....	640
3.28.2. ILA Unions .....	640
3.28.3. ILA Typedefs .....	643
3.28.4. ILA Functions .....	645
3.29. NBI IntrinsicS .....	648
3.29.1. NBI Functions .....	648
3.30. SRAM IntrinsicS .....	649
3.31. Synchronization IntrinsicS .....	649
3.31.1. Synchronization Enumerations .....	650
3.31.2. Synchronization Structs .....	650
3.31.3. Synchronization Typedefs .....	651
3.31.4. Synchronization Functions .....	651

3.32. Math Intrinsics .....	654
3.33. Unaligned Data Access .....	654
3.34. Restrictions On Intrinsics .....	655
3.34.1. Intrinsic Function Arguments that Map to Transfer Registers in Microcode .....	655
3.35. Miscellaneous Intrinsics .....	657
3.35.1. Miscellaneous Enumerations .....	657
3.35.2. Miscellaneous Defines .....	660
3.35.3. Miscellaneous Unions .....	661
3.35.4. Miscellaneous Functions .....	661
<b>4. Technical Support .....</b>	<b>674</b>

## List of Tables

1.1. Contents of this Guide .....	31
1.2. Conventions .....	32
2.1. memcpy() Prototypes .....	36
2.2. memmove() Prototypes .....	38
2.3. memset() Prototypes .....	40
2.4. memcmp() Prototypes .....	40
2.5. memchr() Prototypes .....	42
2.6. strlen() Prototypes .....	43
2.7. strcmp() Prototypes .....	44
2.8. strncmp() Prototypes .....	44
2.9. strcpy() Prototypes .....	45
2.10. strncpy() Prototypes .....	46
2.11. strcat() Prototypes .....	47
2.12. strncat() Prototypes .....	47
2.13. strchr() Prototypes .....	48
2.14. strrchr() Prototypes .....	49
2.15. strstr() Prototypes .....	50
2.16. strspn() Prototypes .....	50
2.17. strcspn() Prototypes .....	51
2.18. strpbrk() Prototypes .....	52
2.19. strtok() Prototypes .....	52
2.20. strtol() Prototypes .....	53
2.21. strtoul() Prototypes .....	54
3.1. union override_alu_ind_t .....	58
3.2. union override_csr_ind_t .....	59
3.3. enum ovr_field_t .....	60
3.4. struct generic_ind_t .....	62
3.5. ovr_init parameters .....	63
3.6. ovr_set parameters .....	64
3.7. cmd_arm_read parameters .....	64
3.8. cmd_arm_read_ind parameters .....	65
3.9. cmd_arm_write parameters .....	65
3.10. cmd_arm_write_ind parameters .....	66
3.11. enum CLS_METER_COLOR .....	66
3.12. enum CLS_METER_RFC .....	66
3.13. typedef CLS_METER_COLOR .....	67
3.14. typedef CLS_METER_RFC .....	67
3.15. cmd_cls_read_be_ptr32 parameters .....	67
3.16. cmd_cls_read_be_ptr40 parameters .....	68
3.17. cmd_cls_read_le_ptr32 parameters .....	68
3.18. cmd_cls_read_le_ptr40 parameters .....	69
3.19. cmd_cls_read_ptr32 parameters .....	70
3.20. cmd_cls_read_ptr40 parameters .....	70
3.21. cmd_cls_write_be_ptr32 parameters .....	71
3.22. cmd_cls_write_be_ptr40 parameters .....	71
3.23. cmd_cls_write_le_ptr32 parameters .....	72
3.24. cmd_cls_write_le_ptr40 parameters .....	73
3.25. cmd_cls_write8_be_ind_ptr40 parameters .....	73
3.26. cmd_cls_write8_be_ind_ptr32 parameters .....	74
3.27. cmd_cls_write8_le_ind_ptr40 parameters .....	74
3.28. cmd_cls_write8_le_ind_ptr32 parameters .....	75

3.29. cmd_cls_write8_ind_ptr40 parameters .....	75
3.30. cmd_cls_write8_ind_ptr32 parameters .....	76
3.31. cmd_cls_write_ptr32 parameters .....	77
3.32. cmd_cls_write_ptr40 parameters .....	77
3.33. cmd_cls_write8_be_ptr32 parameters .....	78
3.34. cmd_cls_write8_be_ptr40 parameters .....	78
3.35. cmd_cls_write8_le_ptr32 parameters .....	79
3.36. cmd_cls_write8_le_ptr40 parameters .....	79
3.37. cmd_cls_write8_ptr32 parameters .....	80
3.38. cmd_cls_write8_ptr40 parameters .....	80
3.39. cmd_cls_clear_bits_ind_ptr40 parameters .....	81
3.40. cmd_cls_clear_bits_ind_ptr32 parameters .....	82
3.41. cmd_cls_set_bits_ind_ptr40 parameters .....	82
3.42. cmd_cls_set_bits_ind_ptr32 parameters .....	83
3.43. cmd_cls_clear_bits_ptr32 parameters .....	84
3.44. cmd_cls_clear_bits_ptr40 parameters .....	84
3.45. cmd_cls_set_bits_ptr32 parameters .....	85
3.46. cmd_cls_set_bits_ptr40 parameters .....	86
3.47. cmd_cls_set_bits_imm_ptr32 parameters .....	86
3.48. cmd_cls_set_bits_imm_ptr40 parameters .....	87
3.49. cmd_cls_clear_bits_imm_ptr32 parameters .....	88
3.50. cmd_cls_clear_bits_imm_ptr40 parameters .....	88
3.51. cmd_cls_xor_bits_ind_ptr40 parameters .....	89
3.52. cmd_cls_xor_bits_ind_ptr32 parameters .....	89
3.53. cmd_cls_xor_bits_ptr32 parameters .....	90
3.54. cmd_cls_xor_bits_ptr40 parameters .....	91
3.55. cmd_cls_swap_ptr32 parameters .....	91
3.56. cmd_cls_swap_ptr40 parameters .....	92
3.57. cmd_cls_meter_ptr32 parameters .....	92
3.58. cmd_cls_meter_ptr40 parameters .....	93
3.59. cmd_cls_statistic_ptr32 parameters .....	93
3.60. cmd_cls_statistic_ptr40 parameters .....	94
3.61. cmd_cls_statistic_imm_ptr32 parameters .....	94
3.62. cmd_cls_statistic_imm_ptr40 parameters .....	95
3.63. cmd_cls_add_ptr32 parameters .....	95
3.64. cmd_cls_add_ptr40 parameters .....	96
3.65. cmd_cls_test_and_add_ptr32 parameters .....	97
3.66. cmd_cls_test_and_add_ptr40 parameters .....	97
3.67. cmd_cls_add64_ptr32 parameters .....	98
3.68. cmd_cls_add64_ptr40 parameters .....	99
3.69. cmd_cls_test_and_add64_ptr32 parameters .....	99
3.70. cmd_cls_test_and_add64_ptr40 parameters .....	100
3.71. cmd_cls_add_sat_ptr32 parameters .....	101
3.72. cmd_cls_add_ind_ptr40 parameters .....	101
3.73. cmd_cls_add_ind_ptr32 parameters .....	102
3.74. cmd_cls_add64_ind_ptr40 parameters .....	103
3.75. cmd_cls_add64_ind_ptr32 parameters .....	103
3.76. cmd_cls_add_sat_ind_ptr40 parameters .....	104
3.77. cmd_cls_add_sat_ind_ptr32 parameters .....	105
3.78. cmd_cls_sub_ind_ptr40 parameters .....	106
3.79. cmd_cls_sub_ind_ptr32 parameters .....	106
3.80. cmd_cls_sub64_ind_ptr40 parameters .....	107
3.81. cmd_cls_sub64_ind_ptr32 parameters .....	108

3.82. cmd_cls_sub_sat_ind_ptr40 parameters .....	108
3.83. cmd_cls_sub_sat_ind_ptr32 parameters .....	109
3.84. cmd_cls_add_sat_ptr40 parameters .....	110
3.85. cmd_cls_add_imm_ptr32 parameters .....	110
3.86. cmd_cls_add_imm_ptr40 parameters .....	111
3.87. cmd_cls_add_imm_sat_ptr32 parameters .....	111
3.88. cmd_cls_add_imm_sat_ptr40 parameters .....	112
3.89. cmd_cls_add64_imm_ptr32 parameters .....	112
3.90. cmd_cls_add64_imm_ptr40 parameters .....	113
3.91. cmd_cls_test_and_add_imm_ptr32 parameters .....	113
3.92. cmd_cls_test_and_add_imm_ptr40 parameters .....	114
3.93. cmd_cls_test_and_add64_imm_ptr32 parameters .....	114
3.94. cmd_cls_test_and_add64_imm_ptr40 parameters .....	115
3.95. cmd_cls_sub_ptr32 parameters .....	116
3.96. cmd_cls_sub_ptr40 parameters .....	116
3.97. cmd_cls_read_le_ind_ptr40 parameters .....	117
3.98. cmd_cls_read_le_ind_ptr32 parameters .....	117
3.99. cmd_cls_read_be_ind_ptr40 parameters .....	118
3.100. cmd_cls_read_be_ind_ptr32 parameters .....	118
3.101. cmd_cls_read_ind_ptr40 parameters .....	119
3.102. cmd_cls_read_ind_ptr32 parameters .....	119
3.103. cmd_cls_write_le_ind_ptr40 parameters .....	120
3.104. cmd_cls_write_le_ind_ptr32 parameters .....	121
3.105. cmd_cls_write_be_ind_ptr40 parameters .....	121
3.106. cmd_cls_write_be_ind_ptr32 parameters .....	122
3.107. cmd_cls_write_ind_ptr40 parameters .....	122
3.108. cmd_cls_write_ind_ptr32 parameters .....	123
3.109. cmd_cls_test_and_sub_ptr32 parameters .....	124
3.110. cmd_cls_test_and_sub_ptr40 parameters .....	124
3.111. cmd_cls_sub64_ptr32 parameters .....	125
3.112. cmd_cls_sub64_ptr40 parameters .....	126
3.113. cmd_cls_test_and_sub64_ptr32 parameters .....	127
3.114. cmd_cls_test_and_sub64_ptr40 parameters .....	127
3.115. cmd_cls_sub_sat_ptr32 parameters .....	128
3.116. cmd_cls_sub_sat_ptr40 parameters .....	129
3.117. cmd_cls_sub_imm_ptr32 parameters .....	130
3.118. cmd_cls_sub_imm_ptr40 parameters .....	130
3.119. cmd_cls_sub_imm_sat_ptr32 parameters .....	131
3.120. cmd_cls_sub_imm_sat_ptr40 parameters .....	131
3.121. cmd_cls_sub64_imm_ptr32 parameters .....	132
3.122. cmd_cls_sub64_imm_ptr40 parameters .....	132
3.123. cmd_cls_test_and_sub_imm_ptr32 parameters .....	133
3.124. cmd_cls_test_and_sub_imm_ptr40 parameters .....	133
3.125. cmd_cls_test_and_sub64_imm_ptr32 parameters .....	134
3.126. cmd_cls_test_and_sub64_imm_ptr40 parameters .....	134
3.127. cmd_cls_test_and_clear_bits_ptr32 parameters .....	135
3.128. cmd_cls_test_and_clear_bits_ptr40 parameters .....	135
3.129. cmd_cls_test_and_clear_bits_ind_ptr40 parameters .....	136
3.130. cmd_cls_test_and_clear_bits_ind_ptr32 parameters .....	136
3.131. cmd_cls_test_and_set_bits_ind_ptr40 parameters .....	137
3.132. cmd_cls_test_and_set_bits_ind_ptr32 parameters .....	137
3.133. cmd_cls_test_and_clear_bits_imm_ptr32 parameters .....	138
3.134. cmd_cls_test_and_clear_bits_imm_ptr40 parameters .....	139

3.135. cmd_cls_test_and_set_bits_ptr32 parameters .....	139
3.136. cmd_cls_test_and_set_bits_ptr40 parameters .....	140
3.137. cmd_cls_test_and_set_bits_imm_ptr32 parameters .....	140
3.138. cmd_cls_test_and_set_bits_imm_ptr40 parameters .....	141
3.139. cmd_cls_test_and_add_sat_ptr32 parameters .....	141
3.140. cmd_cls_test_and_add_sat_ptr40 parameters .....	142
3.141. cmd_cls_test_and_add_sat_ind_ptr40 parameters .....	142
3.142. cmd_cls_test_and_add_sat_ind_ptr32 parameters .....	143
3.143. cmd_cls_test_and_sub_sat_ind_ptr40 parameters .....	143
3.144. cmd_cls_test_and_sub_sat_ind_ptr32 parameters .....	144
3.145. cmd_cls_test_and_add_imm_sat_ptr32 parameters .....	144
3.146. cmd_cls_test_and_add_imm_sat_ptr40 parameters .....	145
3.147. cmd_cls_test_and_sub_sat_ptr32 parameters .....	145
3.148. cmd_cls_test_and_sub_sat_ptr40 parameters .....	146
3.149. cmd_cls_test_and_sub_imm_sat_ptr32 parameters .....	146
3.150. cmd_cls_test_and_sub_imm_sat_ptr40 parameters .....	147
3.151. cmd_cls_incr_ptr32 parameters .....	148
3.152. cmd_cls_incr_ptr40 parameters .....	148
3.153. cmd_cls_decr_ptr32 parameters .....	148
3.154. cmd_cls_decr_ptr40 parameters .....	149
3.155. cmd_cls_incr64_ptr32 parameters .....	149
3.156. cmd_cls_incr64_ptr40 parameters .....	149
3.157. cmd_cls_decr64_ptr32 parameters .....	150
3.158. cmd_cls_decr64_ptr40 parameters .....	150
3.159. cmd_cls_cmp_read_write_ptr32 parameters .....	151
3.160. cmd_cls_cmp_read_write_ptr40 parameters .....	151
3.161. cmd_cls_cmp_read_write_ind_ptr40 parameters .....	152
3.162. cmd_cls_cmp_read_write_ind_ptr32 parameters .....	152
3.163. cmd_cls_queue_lock_ind_ptr40 parameters .....	153
3.164. cmd_cls_queue_lock_ind_ptr32 parameters .....	153
3.165. cmd_cls_queue_lock_ptr32 parameters .....	154
3.166. cmd_cls_queue_lock_ptr40 parameters .....	154
3.167. cmd_cls_queue_unlock_ptr32 parameters .....	155
3.168. cmd_cls_queue_unlock_ptr40 parameters .....	155
3.169. cmd_cls_hash_mask_ptr32 parameters .....	156
3.170. cmd_cls_hash_mask_ptr40 parameters .....	156
3.171. cmd_cls_hash_mask_ind_ptr40 parameters .....	157
3.172. cmd_cls_hash_mask_ind_ptr32 parameters .....	157
3.173. cmd_cls_hash_mask_clear_ptr32 parameters .....	158
3.174. cmd_cls_hash_mask_clear_ptr40 parameters .....	158
3.175. cmd_cls_cam_lookup_ind_ptr40 parameters .....	159
3.176. cmd_cls_cam_lookup_ind_ptr32 parameters .....	159
3.177. cmd_cls_cam_lookup_add_ind_ptr40 parameters .....	160
3.178. cmd_cls_cam_lookup_add_ind_ptr32 parameters .....	160
3.179. cmd_cls_cam_lookup24_ind_ptr40 parameters .....	161
3.180. cmd_cls_cam_lookup24_ind_ptr32 parameters .....	162
3.181. cmd_cls_cam_lookup24_add_ind_ptr40 parameters .....	162
3.182. cmd_cls_cam_lookup24_add_ind_ptr32 parameters .....	163
3.183. cmd_cls_cam_lookup24_add_inc_ind_ptr40 parameters .....	163
3.184. cmd_cls_cam_lookup24_add_inc_ind_ptr32 parameters .....	164
3.185. cmd_cls_cam_lookup16_ind_ptr40 parameters .....	164
3.186. cmd_cls_cam_lookup16_ind_ptr32 parameters .....	165
3.187. cmd_cls_cam_lookup16_add_ind_ptr40 parameters .....	165

3.188. cmd_cls_cam_lookup16_add_ind_ptr32 parameters .....	166
3.189. cmd_cls_cam_lookup8_ind_ptr40 parameters .....	167
3.190. cmd_cls_cam_lookup8_ind_ptr32 parameters .....	167
3.191. cmd_cls_cam_lookup8_add_ind_ptr40 parameters .....	168
3.192. cmd_cls_cam_lookup8_add_ind_ptr32 parameters .....	168
3.193. cmd_cls_tcam_lookup_ind_ptr40 parameters .....	169
3.194. cmd_cls_tcam_lookup_ind_ptr32 parameters .....	169
3.195. cmd_cls_tcam_lookup24_ind_ptr40 parameters .....	170
3.196. cmd_cls_tcam_lookup24_ind_ptr32 parameters .....	170
3.197. cmd_cls_tcam_lookup16_ind_ptr40 parameters .....	171
3.198. cmd_cls_tcam_lookup16_ind_ptr32 parameters .....	172
3.199. cmd_cls_tcam_lookup8_ind_ptr40 parameters .....	172
3.200. cmd_cls_tcam_lookup8_ind_ptr32 parameters .....	173
3.201. cmd_cls_cam_lookup_ptr32 parameters .....	173
3.202. cmd_cls_cam_lookup_ptr40 parameters .....	174
3.203. cmd_cls_cam_lookup_add_ptr32 parameters .....	174
3.204. cmd_cls_cam_lookup_add_ptr40 parameters .....	175
3.205. cmd_cls_cam_lookup24_ptr32 parameters .....	175
3.206. cmd_cls_cam_lookup24_ptr40 parameters .....	176
3.207. cmd_cls_cam_lookup24_add_ptr32 parameters .....	176
3.208. cmd_cls_cam_lookup24_add_ptr40 parameters .....	177
3.209. cmd_cls_cam_lookup24_add_inc_ptr32 parameters .....	178
3.210. cmd_cls_cam_lookup24_add_inc_ptr40 parameters .....	178
3.211. cmd_cls_cam_lookup24_add_lock_ptr32 parameters .....	179
3.212. cmd_cls_cam_lookup24_add_lock_ptr40 parameters .....	179
3.213. cmd_cls_cam_lookup24_add_extend_ptr32 parameters .....	180
3.214. cmd_cls_cam_lookup24_add_extend_ptr40 parameters .....	181
3.215. cmd_cls_cam_lookup16_ptr32 parameters .....	181
3.216. cmd_cls_cam_lookup16_ptr40 parameters .....	182
3.217. cmd_cls_cam_lookup16_add_ptr32 parameters .....	182
3.218. cmd_cls_cam_lookup16_add_ptr40 parameters .....	183
3.219. cmd_cls_cam_lookup8_ptr32 parameters .....	183
3.220. cmd_cls_cam_lookup8_ptr40 parameters .....	184
3.221. cmd_cls_cam_lookup8_add_ptr32 parameters .....	184
3.222. cmd_cls_cam_lookup8_add_ptr40 parameters .....	185
3.223. cmd_cls_tcam_lookup_ptr32 parameters .....	185
3.224. cmd_cls_tcam_lookup_ptr40 parameters .....	186
3.225. cmd_cls_tcam_lookup24_ptr32 parameters .....	186
3.226. cmd_cls_tcam_lookup24_ptr40 parameters .....	187
3.227. cmd_cls_tcam_lookup16_ptr32 parameters .....	188
3.228. cmd_cls_tcam_lookup16_ptr40 parameters .....	188
3.229. cmd_cls_tcam_lookup8_ptr32 parameters .....	189
3.230. cmd_cls_tcam_lookup8_ptr40 parameters .....	189
3.231. enum cls_state_t .....	190
3.232. enum CLS_RING_SIZE .....	190
3.233. enum CLS_RING_EVENT_REPORTS .....	191
3.234. typedef CLS_RING_SIZE .....	192
3.235. typedef CLS_RING_EVENT_REPORTS .....	192
3.236. cmd_cls_state_test parameters .....	192
3.237. cmd_cls_ring_put_ptr32 parameters .....	193
3.238. cmd_cls_ring_put_ptr40 parameters .....	193
3.239. cmd_cls_ring_journal_ptr32 parameters .....	194
3.240. cmd_cls_ring_journal_ptr40 parameters .....	195

3.241. cmd_cls_ring_get_ptr32 parameters .....	195
3.242. cmd_cls_ring_get_ptr40 parameters .....	196
3.243. cmd_cls_ring_pop_ptr32 parameters .....	196
3.244. cmd_cls_ring_pop_ptr40 parameters .....	197
3.245. cmd_cls_ring_get_safe_ptr32 parameters .....	198
3.246. cmd_cls_ring_get_safe_ptr40 parameters .....	198
3.247. cmd_cls_ring_pop_safe_ptr32 parameters .....	199
3.248. cmd_cls_ring_pop_safe_ptr40 parameters .....	199
3.249. cmd_cls_ring_init_ptr32 parameters .....	200
3.250. cmd_cls_ring_init_ptr40 parameters .....	202
3.251. cmd_cls_ring_write_ptr32 parameters .....	202
3.252. cmd_cls_ring_write_ptr40 parameters .....	203
3.253. cmd_cls_ring_read_ptr32 parameters .....	204
3.254. cmd_cls_ring_read_ptr40 parameters .....	204
3.255. cmd_cls_ring_ordered_lock_ptr32 parameters .....	205
3.256. cmd_cls_ring_ordered_lock_ptr40 parameters .....	205
3.257. cmd_cls_ring_ordered_unlock_ptr32 parameters .....	206
3.258. cmd_cls_ring_ordered_unlock_ptr40 parameters .....	206
3.259. cmd_cls_ring_workq_add_thread_ptr32 parameters .....	207
3.260. cmd_cls_ring_workq_add_thread_ptr40 parameters .....	207
3.261. cmd_cls_ring_workq_add_work_ptr32 parameters .....	208
3.262. cmd_cls_ring_workq_add_work_ptr40 parameters .....	208
3.263. cmd_cls_ring_put_ind_ptr40 parameters .....	209
3.264. cmd_cls_ring_put_ind_ptr32 parameters .....	209
3.265. cmd_cls_ring_journal_ind_ptr40 parameters .....	210
3.266. cmd_cls_ring_journal_ind_ptr32 parameters .....	210
3.267. cmd_cls_ring_get_ind_ptr40 parameters .....	211
3.268. cmd_cls_ring_get_ind_ptr32 parameters .....	211
3.269. cmd_cls_ring_pop_ind_ptr40 parameters .....	212
3.270. cmd_cls_ring_pop_ind_ptr32 parameters .....	212
3.271. cmd_cls_ring_get_safe_ind_ptr40 parameters .....	213
3.272. cmd_cls_ring_get_safe_ind_ptr32 parameters .....	213
3.273. cmd_cls_ring_pop_safe_ind_ptr40 parameters .....	214
3.274. cmd_cls_ring_pop_safe_ind_ptr32 parameters .....	214
3.275. cmd_cls_reflect_write_sig_local parameters .....	215
3.276. cmd_cls_reflect_write_sig_local_ptr40 parameters .....	216
3.277. cmd_cls_reflect_write_sig_remote parameters .....	217
3.278. cmd_cls_reflect_write_sig_remote_ptr40 parameters .....	217
3.279. cmd_cls_reflect_write_sig_both parameters .....	218
3.280. cmd_cls_reflect_write_sig_both_ptr40 parameters .....	218
3.281. cmd_cls_reflect_read_sig_remote parameters .....	219
3.282. cmd_cls_reflect_read_sig_remote_ptr40 parameters .....	220
3.283. cmd_cls_reflect_read_sig_local parameters .....	220
3.284. cmd_cls_reflect_read_sig_local_ptr40 parameters .....	221
3.285. cmd_cls_reflect_read_sig_both parameters .....	222
3.286. cmd_cls_reflect_read_sig_both_ptr40 parameters .....	222
3.287. cmd_cls_reflect_write_sig_local_ind_ptr40 parameters .....	223
3.288. cmd_cls_reflect_write_sig_local_ind parameters .....	224
3.289. cmd_cls_reflect_write_sig_remote_ind_ptr40 parameters .....	224
3.290. cmd_cls_reflect_write_sig_remote_ind parameters .....	225
3.291. cmd_cls_reflect_write_sig_both_ind_ptr40 parameters .....	226
3.292. cmd_cls_reflect_write_sig_both_ind parameters .....	226
3.293. cmd_cls_reflect_read_sig_remote_ind_ptr40 parameters .....	227

3.294. cmd_cls_reflect_read_sig_remote_ind parameters .....	228
3.295. cmd_cls_reflect_read_sig_local_ind_ptr40 parameters .....	228
3.296. cmd_cls_reflect_read_sig_local_ind parameters .....	229
3.297. cmd_cls_reflect_read_sig_both_ind_ptr40 parameters .....	230
3.298. cmd_cls_reflect_read_sig_both_ind parameters .....	230
3.299. enum CLUSTER_TARGET_REGISTER_TYPE .....	231
3.300. enum CLUSTER_TARGET_ADDRESS_MODE .....	231
3.301. enum CT_RING_SIZE .....	231
3.302. enum CT_RING_STATUS .....	232
3.303. union cluster_target_next_neighbour_write_address_format_t .....	233
3.304. union cluster_target_reflect_address_format_t .....	233
3.305. union cluster_target_signal_me_address_format_t .....	234
3.306. union cluster_target_xpb_address_format_t .....	234
3.307. typedef CLUSTER_TARGET_REGISTER_TYPE .....	235
3.308. typedef CLUSTER_TARGET_ADDRESS_MODE .....	235
3.309. typedef cluster_target_xpb_address_format_t .....	235
3.310. typedef cluster_target_reflect_address_format_t .....	235
3.311. typedef cluster_target_signal_me_address_format_t .....	235
3.312. typedef cluster_target_next_neighbour_write_address_format_t .....	236
3.313. typedef CT_RING_SIZE .....	236
3.314. typedef CT_RING_STATUS .....	236
3.315. cmd_cluster_target_xpb_write parameters .....	237
3.316. cmd_cluster_target_xpb_read parameters .....	237
3.317. cmd_cluster_target_reflect_write_sig_none parameters .....	238
3.318. cmd_cluster_target_reflect_read_sig_none parameters .....	238
3.319. cmd_cluster_target_reflect_write_sig_both parameters .....	239
3.320. cmd_cluster_target_reflect_read_sig_both parameters .....	240
3.321. cmd_cluster_target_reflect_write_sig_init parameters .....	240
3.322. cmd_cluster_target_reflect_read_sig_init parameters .....	241
3.323. cmd_cluster_target_reflect_write_sig_remote parameters .....	242
3.324. cmd_cluster_target_reflect_read_sig_remote parameters .....	243
3.325. cmd_cluster_target_sig_me_ctx parameters .....	244
3.326. cmd_cluster_target_next_neighbour_write parameters .....	244
3.327. cmd_cluster_target_next_neighbour_write_ind parameters .....	246
3.328. cmd_cluster_target_ring_put parameters .....	246
3.329. cmd_cluster_target_ring_get parameters .....	247
3.330. cmd_cluster_target_ring_init_ptr32 parameters .....	248
3.331. cmd_cluster_target_ring_init_ptr40 parameters .....	250
3.332. cmd_cluster_target_ring_init parameters .....	250
3.333. enum local_csr_t .....	251
3.334. union ACTIVE_CTX_STS_t .....	253
3.335. enum bytesSpecifier_t .....	254
3.336. crc_iscsi_le parameters .....	255
3.337. crc_iscsi_be parameters .....	255
3.338. crc_10_le parameters .....	256
3.339. crc_10_be parameters .....	256
3.340. crc_5_le parameters .....	257
3.341. crc_5_be parameters .....	257
3.342. crc_iscsi_le_bit_swap parameters .....	258
3.343. crc_iscsi_be_bit_swap parameters .....	258
3.344. crc_10_le_bit_swap parameters .....	259
3.345. crc_10_be_bit_swap parameters .....	260
3.346. crc_5_le_bit_swap parameters .....	260

3.347. crc_5_be_bit_swap parameters .....	261
3.348. pop_count parameters .....	261
3.349. rotr parameters .....	261
3.350. rotl parameters .....	262
3.351. crc_16_le parameters .....	262
3.352. crc_16_be parameters .....	263
3.353. crc_32_le parameters .....	263
3.354. crc_32_be parameters .....	264
3.355. crc_ccitt_le parameters .....	264
3.356. crc_ccitt_be parameters .....	265
3.357. crc_16_le_bit_swap parameters .....	265
3.358. crc_16_be_bit_swap parameters .....	266
3.359. crc_32_le_bit_swap parameters .....	266
3.360. crc_32_be_bit_swap parameters .....	267
3.361. crc_ccitt_le_bit_swap parameters .....	267
3.362. crc_ccitt_be_bit_swap parameters .....	268
3.363. crc_write parameters .....	269
3.364. cmd_crypto_read parameters .....	269
3.365. cmd_crypto_read_ind parameters .....	270
3.366. cmd_crypto_write parameters .....	270
3.367. cmd_crypto_write_ind parameters .....	270
3.368. cmd_crypto_write_fifo parameters .....	271
3.369. cmd_crypto_write_fifo_ind parameters .....	271
3.370. cmd_mem_workq_add_work parameters .....	274
3.371. cmd_mem_workq_add_work_imm parameters .....	274
3.372. cmd_mem_workq_add_thread parameters .....	275
3.373. MU Ring Defines .....	275
3.374. enum MEM_RING_SIZE .....	275
3.375. struct mem_ring_put_status_t .....	276
3.376. union mem_ring_desc_t .....	276
3.377. typedef MEM_RING_SIZE .....	277
3.378. typedef mem_ring_desc_t .....	277
3.379. typedef mem_ring_put_status_t .....	278
3.380. typedef mem_ring_desc_in_mem_t .....	278
3.381. typedef mem_ring_put_status_in_read_reg_t .....	278
3.382. cmd_mem_ring_read_desc parameters .....	279
3.383. cmd_mem_ring_write_desc parameters .....	279
3.384. cmd_mem_ring_push_desc parameters .....	280
3.385. cmd_mem_ring_read_buffer_unbounded parameters .....	280
3.386. cmd_mem_ring_read_buffer parameters .....	281
3.387. cmd_mem_ring_write_buffer_unbounded parameters .....	281
3.388. cmd_mem_ring_write_buffer parameters .....	282
3.389. cmd_mem_ring_add_tail parameters .....	283
3.390. cmd_mem_ring_put_buffer parameters .....	283
3.391. cmd_mem_ring_journal_buffer parameters .....	284
3.392. cmd_mem_ring_get_buffer parameters .....	284
3.393. cmd_mem_ring_get_buffer_eop parameters .....	285
3.394. cmd_mem_ring_get_buffer_freely parameters .....	287
3.395. cmd_mem_ring_pop_buffer parameters .....	287
3.396. cmd_mem_ring_pop_buffer_freely parameters .....	288
3.397. cmd_mem_ring_pop_buffer_eop parameters .....	288
3.398. cmd_mem_ring_fastjournal_imm parameters .....	289
3.399. cmd_mem_ring_init parameters .....	291

3.400. cmd_mem_ring_init_with_loc_and_page parameters .....	292
3.401. struct mem_ticket_line_push_t .....	292
3.402. struct mem_ticket_line_t .....	293
3.403. typedef mem_ticket_line_t .....	293
3.404. typedef mem_ticket_line_push_t .....	294
3.405. cmd_mem_ticket_release_ptr32 parameters .....	294
3.406. cmd_mem_ticket_release_ptr40 parameters .....	295
3.407. cmd_mem_ticket_release_push_ptr32 parameters .....	295
3.408. cmd_mem_ticket_release_push_ptr40 parameters .....	296
3.409. cmd_mem_ticket_line_init_ptr32 parameters .....	296
3.410. cmd_mem_ticket_line_init_ptr40 parameters .....	297
3.411. cmd_mem_ticket_line_push_init_ptr32 parameters .....	297
3.412. cmd_mem_ticket_line_push_init_ptr40 parameters .....	298
3.413. struct mem_lockq128_t .....	299
3.414. struct mem_lockq256_t .....	299
3.415. union mem_lockq_desc_t .....	299
3.416. union mem_lockq_entry_t .....	300
3.417. typedef mem_lockq_desc_t .....	300
3.418. typedef mem_lockq_entry_t .....	300
3.419. typedef mem_lockq128_t .....	301
3.420. typedef mem_lockq256_t .....	301
3.421. typedef mem_lockq128_in_mem_t .....	301
3.422. typedef mem_lockq128_ptr40_t .....	301
3.423. typedef mem_lockq256_in_mem_t .....	302
3.424. typedef mem_lockq256_ptr40_t .....	302
3.425. cmd_mem_lockq128_lock_ptr32 parameters .....	302
3.426. cmd_mem_lockq128_lock_ind_ptr32 parameters .....	303
3.427. cmd_mem_lockq128_lock_ptr40 parameters .....	303
3.428. cmd_mem_lockq128_lock_ind_ptr40 parameters .....	303
3.429. cmd_mem_lockq256_lock_ptr32 parameters .....	304
3.430. cmd_mem_lockq256_lock_ind_ptr32 parameters .....	304
3.431. cmd_mem_lockq256_lock_ptr40 parameters .....	305
3.432. cmd_mem_lockq256_lock_ind_ptr40 parameters .....	305
3.433. cmd_mem_lockq128_unlock_ptr32 parameters .....	306
3.434. cmd_mem_lockq128_unlock_ind_ptr32 parameters .....	306
3.435. cmd_mem_lockq128_unlock_ptr40 parameters .....	306
3.436. cmd_mem_lockq128_unlock_ind_ptr40 parameters .....	307
3.437. cmd_mem_lockq256_unlock_ptr32 parameters .....	307
3.438. cmd_mem_lockq256_unlock_ind_ptr32 parameters .....	307
3.439. cmd_mem_lockq256_unlock_ptr40 parameters .....	308
3.440. cmd_mem_lockq256_unlock_ind_ptr40 parameters .....	308
3.441. cmd_mem_lockq128_init_ptr32 parameters .....	308
3.442. cmd_mem_lockq128_init_ptr40 parameters .....	309
3.443. cmd_mem_lockq256_init_ptr32 parameters .....	309
3.444. cmd_mem_lockq256_init_ptr40 parameters .....	310
3.445. struct mem_lock128_t .....	310
3.446. struct mem_lock256_t .....	311
3.447. struct mem_lock384_t .....	311
3.448. struct mem_lock512_t .....	311
3.449. union mem_lock_in_t .....	311
3.450. union mem_lock_out_t .....	312
3.451. typedef mem_lock128_t .....	312
3.452. typedef mem_lock256_t .....	313

3.453. <code>typedef mem_lock384_t</code> .....	313
3.454. <code>typedef mem_lock512_t</code> .....	313
3.455. <code>typedef mem_lock_in_t</code> .....	313
3.456. <code>typedef mem_lock_out_t</code> .....	313
3.457. <code>cmd_mem_lock128_init_ptr32</code> parameters .....	314
3.458. <code>cmd_mem_lock128_init_ptr40</code> parameters .....	314
3.459. <code>cmd_mem_lock256_init_ptr32</code> parameters .....	315
3.460. <code>cmd_mem_lock256_init_ptr40</code> parameters .....	315
3.461. <code>cmd_mem_lock384_init_ptr32</code> parameters .....	315
3.462. <code>cmd_mem_lock384_init_ptr40</code> parameters .....	316
3.463. <code>cmd_mem_lock512_init_ptr32</code> parameters .....	316
3.464. <code>cmd_mem_lock512_init_ptr40</code> parameters .....	316
3.465. <code>cmd_mem_lock128_ptr32</code> parameters .....	317
3.466. <code>cmd_mem_lock128_ptr40</code> parameters .....	318
3.467. <code>mem_lock256_ptr32</code> parameters .....	318
3.468. <code>mem_lock256_ptr40</code> parameters .....	319
3.469. <code>mem_lock384_ptr32</code> parameters .....	320
3.470. <code>mem_lock384_ptr40</code> parameters .....	321
3.471. <code>mem_lock512_ptr32</code> parameters .....	321
3.472. <code>mem_lock512_ptr40</code> parameters .....	322
3.473. <code>cmd_mem_unlock128_ptr32</code> parameters .....	323
3.474. <code>cmd_mem_unlock128_ptr40</code> parameters .....	324
3.475. <code>mem_unlock256_ptr32</code> parameters .....	324
3.476. <code>mem_unlock256_ptr40</code> parameters .....	325
3.477. <code>mem_unlock384_ptr32</code> parameters .....	325
3.478. <code>mem_unlock384_ptr40</code> parameters .....	326
3.479. <code>mem_unlock512_ptr32</code> parameters .....	326
3.480. <code>mem_unlock512_ptr40</code> parameters .....	327
3.481. <code>union mem_list_desc_t0</code> .....	327
3.482. <code>union mem_list_desc_t1</code> .....	328
3.483. <code>union mem_list_desc_t2</code> .....	329
3.484. <code>union mem_list_desc_t3</code> .....	329
3.485. <code>union mem_list_link_mem_t0</code> .....	330
3.486. <code>union mem_list_link_mem_t1</code> .....	330
3.487. <code>union mem_list_link_mem_t2</code> .....	330
3.488. <code>union mem_list_link_mem_t3</code> .....	331
3.489. <code>union mem_list_link_t0</code> .....	331
3.490. <code>union mem_list_link_t1</code> .....	331
3.491. <code>union mem_list_link_t2</code> .....	331
3.492. <code>union mem_list_link_t3</code> .....	332
3.493. <code>typedef MEM_LIST_TYPE</code> .....	332
3.494. <code>typedef mem_list_desc_t0</code> .....	332
3.495. <code>typedef mem_list_link_t0</code> .....	333
3.496. <code>typedef mem_list_link_mem_t0</code> .....	333
3.497. <code>typedef mem_list_desc_in_mem_t0</code> .....	333
3.498. <code>typedef mem_list_link_in_mem_t0</code> .....	333
3.499. <code>typedef mem_list_desc_t1</code> .....	333
3.500. <code>typedef mem_list_link_t1</code> .....	334
3.501. <code>typedef mem_list_link_mem_t1</code> .....	334
3.502. <code>typedef mem_list_desc_in_mem_t1</code> .....	334
3.503. <code>typedef mem_list_link_in_mem_t1</code> .....	334
3.504. <code>typedef mem_list_desc_t2</code> .....	334
3.505. <code>typedef mem_list_link_t2</code> .....	335

3.506. typedef mem_list_link_mem_t2 .....	335
3.507. typedef mem_list_desc_in_mem_t2 .....	335
3.508. typedef mem_list_link_in_mem_t2 .....	335
3.509. typedef mem_list_desc_t3 .....	336
3.510. typedef mem_list_link_t3 .....	336
3.511. typedef mem_list_link_mem_t3 .....	336
3.512. typedef mem_list_desc_in_mem_t3 .....	336
3.513. typedef mem_list_link_in_mem_t3 .....	336
3.514. cmd_mem_list_read_desc_t0 parameters .....	337
3.515. cmd_mem_list_read_desc_t1 parameters .....	337
3.516. cmd_mem_list_read_desc_t2 parameters .....	338
3.517. cmd_mem_list_read_desc_t3 parameters .....	339
3.518. cmd_mem_list_push_desc_t0 parameters .....	339
3.519. cmd_mem_list_push_desc_t1 parameters .....	340
3.520. cmd_mem_list_push_desc_t2 parameters .....	340
3.521. cmd_mem_list_push_desc_t3 parameters .....	341
3.522. cmd_mem_list_write_desc_t0 parameters .....	341
3.523. cmd_mem_list_write_desc_t1 parameters .....	342
3.524. cmd_mem_list_write_desc_t2 parameters .....	343
3.525. cmd_mem_list_write_desc_t3 parameters .....	343
3.526. cmd_mem_list_enqueue_t0 parameters .....	344
3.527. cmd_mem_list_enqueue_t1 parameters .....	345
3.528. cmd_mem_list_enqueue_t2 parameters .....	345
3.529. cmd_mem_list_enqueue_t3 parameters .....	346
3.530. cmd_mem_list_enqueue_tail_t0 parameters .....	347
3.531. cmd_mem_list_enqueue_tail_t1 parameters .....	347
3.532. cmd_mem_list_enqueue_tail_t2 parameters .....	348
3.533. cmd_mem_list_enqueue_tail_t3 parameters .....	348
3.534. cmd_mem_list_dequeue_t0 parameters .....	349
3.535. cmd_mem_list_dequeue_t1 parameters .....	349
3.536. cmd_mem_list_dequeue_t2 parameters .....	350
3.537. cmd_mem_list_dequeue_t3 parameters .....	351
3.538. cmd_mem_list_init parameters .....	352
3.539. cmd_mem_list_init_with_loc_and_page parameters .....	353
3.540. enum MICROQ_ENTRY_SIZE .....	353
3.541. struct mem_microq128_t .....	354
3.542. struct mem_microq256_t .....	354
3.543. union mem_microq_desc_t .....	355
3.544. typedef MICROQ_ENTRY_SIZE .....	355
3.545. typedef mem_microq_desc_t .....	355
3.546. typedef mem_microq128_t .....	356
3.547. typedef mem_microq256_t .....	356
3.548. typedef mem_microq128_in_mem_t .....	356
3.549. typedef mem_microq128_ptr40_t .....	356
3.550. typedef mem_microq256_in_mem_t .....	357
3.551. typedef mem_microq256_ptr40_t .....	357
3.552. cmd_mem_microq128_put_ptr32 parameters .....	357
3.553. cmd_mem_microq128_put_ptr40 parameters .....	358
3.554. cmd_mem_microq256_put_ptr32 parameters .....	358
3.555. cmd_mem_microq256_put_ptr40 parameters .....	359
3.556. cmd_mem_microq128_get_ptr32 parameters .....	359
3.557. cmd_mem_microq128_get_ptr40 parameters .....	360
3.558. cmd_mem_microq256_get_ptr32 parameters .....	360

3.559. cmd_mem_microq256_get_ptr40 parameters .....	361
3.560. cmd_mem_microq128_pop_ptr32 parameters .....	361
3.561. cmd_mem_microq128_pop_ptr40 parameters .....	362
3.562. cmd_mem_microq256_pop_ptr32 parameters .....	362
3.563. cmd_mem_microq256_pop_ptr40 parameters .....	363
3.564. cmd_mem_microq128_init_ptr32 parameters .....	363
3.565. cmd_mem_microq128_init_ptr40 parameters .....	365
3.566. cmd_mem_microq256_init_ptr32 parameters .....	365
3.567. cmd_mem_microq256_init_ptr40 parameters .....	366
3.568. struct mem_cam128_t .....	366
3.569. struct mem_cam256_t .....	366
3.570. struct mem_cam384_t .....	366
3.571. struct mem_cam512_t .....	367
3.572. struct mem_cam_lookup32_in_t .....	367
3.573. union mem_cam_lookup16_add_out_t .....	367
3.574. union mem_cam_lookup16_in_t .....	368
3.575. union mem_cam_lookup16_out_t .....	368
3.576. union mem_cam_lookup24_add_inc_out_t .....	368
3.577. union mem_cam_lookup24_add_in_t .....	369
3.578. union mem_cam_lookup24_add_out_t .....	370
3.579. union mem_cam_lookup24_in_t .....	370
3.580. union mem_cam_lookup24_out_t .....	371
3.581. union mem_cam_lookup32_add_out_t .....	371
3.582. union mem_cam_lookup32_out_t .....	371
3.583. union mem_cam_lookup8_add_out_t .....	372
3.584. union mem_cam_lookup8_in_t .....	372
3.585. union mem_cam_lookup8_out_t .....	373
3.586. typedef mem_cam128_t .....	373
3.587. typedef mem_cam256_t .....	373
3.588. typedef mem_cam384_t .....	373
3.589. typedef mem_cam512_t .....	374
3.590. typedef mem_cam128_in_mem_t .....	374
3.591. typedef mem_cam256_in_mem_t .....	374
3.592. typedef mem_cam384_in_mem_t .....	374
3.593. typedef mem_cam512_in_mem_t .....	374
3.594. typedef mem_cam128_ptr40_t .....	374
3.595. typedef mem_cam256_ptr40_t .....	375
3.596. typedef mem_cam384_ptr40_t .....	375
3.597. typedef mem_cam512_ptr40_t .....	375
3.598. typedef mem_cam_lookup8_in_t .....	375
3.599. typedef mem_cam_lookup8_out_t .....	375
3.600. typedef mem_cam_lookup8_add_out_t .....	376
3.601. typedef mem_cam_lookup8_out_in_read_reg_t .....	376
3.602. typedef mem_cam_lookup8_add_out_in_read_reg_t .....	376
3.603. typedef mem_cam_lookup16_in_t .....	376
3.604. typedef mem_cam_lookup16_out_t .....	377
3.605. typedef mem_cam_lookup16_add_out_t .....	377
3.606. typedef mem_cam_lookup16_out_in_read_reg_t .....	377
3.607. typedef mem_cam_lookup16_add_out_in_read_reg_t .....	377
3.608. typedef mem_cam_lookup24_in_t .....	378
3.609. typedef mem_cam_lookup24_add_in_t .....	378
3.610. typedef mem_cam_lookup24_add_inc_in_t .....	378
3.611. typedef mem_cam_lookup24_out_t .....	378

3.612. typedef mem_cam_lookup24_add_out_t .....	379
3.613. typedef mem_cam_lookup24_add_inc_out_t .....	379
3.614. typedef mem_cam_lookup24_out_in_read_reg_t .....	379
3.615. typedef mem_cam_lookup24_add_out_in_read_reg_t .....	379
3.616. typedef mem_cam_lookup24_add_inc_out_in_read_reg_t .....	380
3.617. typedef mem_cam_lookup32_in_t .....	380
3.618. typedef mem_cam_lookup32_out_t .....	380
3.619. typedef mem_cam_lookup32_add_out_t .....	380
3.620. typedef mem_cam_lookup32_out_in_read_reg_t .....	380
3.621. typedef mem_cam_lookup32_add_out_in_read_reg_t .....	381
3.622. cmd_mem_cam128_init_ptr32 parameters .....	381
3.623. cmd_mem_cam128_init_ptr40 parameters .....	381
3.624. cmd_mem_cam256_init_ptr32 parameters .....	382
3.625. cmd_mem_cam256_init_ptr40 parameters .....	382
3.626. cmd_mem_cam384_init_ptr32 parameters .....	382
3.627. cmd_mem_cam384_init_ptr40 parameters .....	383
3.628. cmd_mem_cam512_init_ptr32 parameters .....	383
3.629. cmd_mem_cam512_init_ptr40 parameters .....	383
3.630. mem_cam128_set8_ptr32 parameters .....	384
3.631. mem_cam128_set8_ptr40 parameters .....	384
3.632. mem_cam256_set8_ptr32 parameters .....	385
3.633. mem_cam256_set8_ptr40 parameters .....	385
3.634. mem_cam384_set8_ptr32 parameters .....	386
3.635. mem_cam384_set8_ptr40 parameters .....	386
3.636. mem_cam512_set8_ptr32 parameters .....	387
3.637. mem_cam512_set8_ptr40 parameters .....	388
3.638. mem_cam128_set16_ptr32 parameters .....	388
3.639. mem_cam128_set16_ptr40 parameters .....	389
3.640. mem_cam256_set16_ptr32 parameters .....	389
3.641. mem_cam256_set16_ptr40 parameters .....	390
3.642. mem_cam384_set16_ptr32 parameters .....	390
3.643. mem_cam384_set16_ptr40 parameters .....	391
3.644. mem_cam512_set16_ptr32 parameters .....	391
3.645. mem_cam512_set16_ptr40 parameters .....	392
3.646. mem_cam128_set24_ptr32 parameters .....	393
3.647. mem_cam128_set24_ptr40 parameters .....	393
3.648. mem_cam256_set24_ptr32 parameters .....	394
3.649. mem_cam256_set24_ptr40 parameters .....	394
3.650. mem_cam384_set24_ptr32 parameters .....	395
3.651. mem_cam384_set24_ptr40 parameters .....	396
3.652. mem_cam512_set24_ptr32 parameters .....	396
3.653. mem_cam512_set24_ptr40 parameters .....	397
3.654. mem_cam128_set32_ptr32 parameters .....	397
3.655. mem_cam128_set32_ptr40 parameters .....	398
3.656. mem_cam256_set32_ptr32 parameters .....	398
3.657. mem_cam256_set32_ptr40 parameters .....	399
3.658. mem_cam384_set32_ptr32 parameters .....	399
3.659. mem_cam384_set32_ptr40 parameters .....	400
3.660. mem_cam512_set32_ptr32 parameters .....	401
3.661. mem_cam512_set32_ptr40 parameters .....	401
3.662. mem_cam128_lookup8_ptr32 parameters .....	402
3.663. mem_cam128_lookup8_ptr40 parameters .....	402
3.664. mem_cam256_lookup8_ptr32 parameters .....	403

3.665. mem_cam256_lookup8_ptr40 parameters .....	404
3.666. mem_cam384_lookup8_ptr32 parameters .....	405
3.667. mem_cam384_lookup8_ptr40 parameters .....	405
3.668. mem_cam512_lookup8_ptr32 parameters .....	406
3.669. mem_cam512_lookup8_ptr40 parameters .....	407
3.670. mem_cam128_lookup8_add_ptr32 parameters .....	407
3.671. mem_cam128_lookup8_add_ptr40 parameters .....	408
3.672. mem_cam256_lookup8_add_ptr32 parameters .....	409
3.673. mem_cam256_lookup8_add_ptr40 parameters .....	410
3.674. mem_cam384_lookup8_add_ptr32 parameters .....	410
3.675. mem_cam384_lookup8_add_ptr40 parameters .....	411
3.676. mem_cam512_lookup8_add_ptr32 parameters .....	412
3.677. mem_cam512_lookup8_add_ptr40 parameters .....	413
3.678. mem_cam128_lookup16_ptr32 parameters .....	413
3.679. mem_cam128_lookup16_ptr40 parameters .....	414
3.680. mem_cam256_lookup16_ptr32 parameters .....	415
3.681. mem_cam256_lookup16_ptr40 parameters .....	416
3.682. mem_cam384_lookup16_ptr32 parameters .....	416
3.683. mem_cam384_lookup16_ptr40 parameters .....	417
3.684. mem_cam512_lookup16_ptr32 parameters .....	418
3.685. mem_cam512_lookup16_ptr40 parameters .....	418
3.686. mem_cam128_lookup16_add_ptr32 parameters .....	419
3.687. mem_cam128_lookup16_add_ptr40 parameters .....	420
3.688. mem_cam256_lookup16_add_ptr32 parameters .....	421
3.689. mem_cam256_lookup16_add_ptr40 parameters .....	421
3.690. mem_cam384_lookup16_add_ptr32 parameters .....	422
3.691. mem_cam384_lookup16_add_ptr40 parameters .....	423
3.692. mem_cam512_lookup16_add_ptr32 parameters .....	424
3.693. mem_cam512_lookup16_add_ptr40 parameters .....	424
3.694. mem_cam128_lookup24_ptr32 parameters .....	425
3.695. mem_cam128_lookup24_ptr40 parameters .....	426
3.696. mem_cam256_lookup24_ptr32 parameters .....	427
3.697. mem_cam256_lookup24_ptr40 parameters .....	427
3.698. mem_cam384_lookup24_ptr32 parameters .....	428
3.699. mem_cam384_lookup24_ptr40 parameters .....	429
3.700. mem_cam512_lookup24_ptr32 parameters .....	429
3.701. mem_cam512_lookup24_ptr40 parameters .....	430
3.702. mem_cam128_lookup24_add_ptr32 parameters .....	431
3.703. mem_cam128_lookup24_add_ptr40 parameters .....	431
3.704. mem_cam256_lookup24_add_ptr32 parameters .....	432
3.705. mem_cam256_lookup24_add_ptr40 parameters .....	433
3.706. mem_cam384_lookup24_add_ptr32 parameters .....	434
3.707. mem_cam384_lookup24_add_ptr40 parameters .....	434
3.708. mem_cam512_lookup24_add_ptr32 parameters .....	435
3.709. mem_cam512_lookup24_add_ptr40 parameters .....	436
3.710. mem_cam128_lookup24_add_inc_ptr32 parameters .....	436
3.711. mem_cam128_lookup24_add_inc_ptr40 parameters .....	437
3.712. mem_cam256_lookup24_add_inc_ptr32 parameters .....	438
3.713. mem_cam256_lookup24_add_inc_ptr40 parameters .....	439
3.714. mem_cam384_lookup24_add_inc_ptr32 parameters .....	439
3.715. mem_cam384_lookup24_add_inc_ptr40 parameters .....	440
3.716. mem_cam512_lookup24_add_inc_ptr32 parameters .....	441
3.717. mem_cam512_lookup24_add_inc_ptr40 parameters .....	441

3.718. mem_cam128_lookup32_ptr32 parameters .....	442
3.719. mem_cam128_lookup32_ptr40 parameters .....	443
3.720. mem_cam256_lookup32_ptr32 parameters .....	444
3.721. mem_cam256_lookup32_ptr40 parameters .....	444
3.722. mem_cam384_lookup32_ptr32 parameters .....	445
3.723. mem_cam384_lookup32_ptr40 parameters .....	446
3.724. mem_cam512_lookup32_ptr32 parameters .....	446
3.725. mem_cam512_lookup32_ptr40 parameters .....	447
3.726. mem_cam128_lookup32_add_ptr32 parameters .....	448
3.727. mem_cam128_lookup32_add_ptr40 parameters .....	448
3.728. mem_cam256_lookup32_add_ptr32 parameters .....	449
3.729. mem_cam256_lookup32_add_ptr40 parameters .....	450
3.730. mem_cam384_lookup32_add_ptr32 parameters .....	451
3.731. mem_cam384_lookup32_add_ptr40 parameters .....	451
3.732. mem_cam512_lookup32_add_ptr32 parameters .....	452
3.733. mem_cam512_lookup32_add_ptr40 parameters .....	453
3.734. MU TCAM Defines .....	453
3.735. struct mem_tcam128_t .....	460
3.736. struct mem_tcam256_t .....	460
3.737. struct mem_tcam384_t .....	460
3.738. struct mem_tcam512_t .....	461
3.739. struct mem_tcam_lookup32_in_t .....	461
3.740. union mem_tcam_lookup16_in_t .....	461
3.741. union mem_tcam_lookup16_out_t .....	461
3.742. union mem_tcam_lookup24_in_t .....	462
3.743. union mem_tcam_lookup24_out_t .....	462
3.744. union mem_tcam_lookup32_out_t .....	462
3.745. union mem_tcam_lookup8_in_t .....	463
3.746. union mem_tcam_lookup8_out_t .....	463
3.747. typedef mem_tcam128_t .....	464
3.748. typedef mem_tcam256_t .....	464
3.749. typedef mem_tcam384_t .....	464
3.750. typedef mem_tcam512_t .....	464
3.751. typedef mem_tcam128_in_mem_t .....	464
3.752. typedef mem_tcam256_in_mem_t .....	465
3.753. typedef mem_tcam384_in_mem_t .....	465
3.754. typedef mem_tcam512_in_mem_t .....	465
3.755. typedef mem_tcam128_ptr40_t .....	465
3.756. typedef mem_tcam256_ptr40_t .....	465
3.757. typedef mem_tcam384_ptr40_t .....	465
3.758. typedef mem_tcam512_ptr40_t .....	466
3.759. typedef mem_tcam_lookup8_in_t .....	466
3.760. typedef mem_tcam_lookup8_out_t .....	466
3.761. typedef mem_tcam_lookup8_out_in_read_reg_t .....	466
3.762. typedef mem_tcam_lookup16_in_t .....	467
3.763. typedef mem_tcam_lookup16_out_t .....	467
3.764. typedef mem_tcam_lookup16_out_in_read_reg_t .....	467
3.765. typedef mem_tcam_lookup24_in_t .....	467
3.766. typedef mem_tcam_lookup24_out_t .....	467
3.767. typedef mem_tcam_lookup24_out_in_read_reg_t .....	468
3.768. typedef mem_tcam_lookup32_in_t .....	468
3.769. typedef mem_tcam_lookup32_out_t .....	468
3.770. typedef mem_tcam_lookup32_out_in_read_reg_t .....	468

3.771. cmd_mem_tcam128_init_ptr32 parameters .....	469
3.772. cmd_mem_tcam128_init_ptr40 parameters .....	469
3.773. cmd_mem_tcam256_init_ptr32 parameters .....	470
3.774. cmd_mem_tcam256_init_ptr40 parameters .....	470
3.775. cmd_mem_tcam384_init_ptr32 parameters .....	470
3.776. cmd_mem_tcam384_init_ptr40 parameters .....	471
3.777. cmd_mem_tcam512_init_ptr32 parameters .....	471
3.778. cmd_mem_tcam512_init_ptr40 parameters .....	472
3.779. mem_tcam128_set8_word_ptr32 parameters .....	472
3.780. mem_tcam128_set8_word_ptr40 parameters .....	473
3.781. mem_tcam128_set8_entry_value_ptr32 parameters .....	473
3.782. mem_tcam128_set8_entry_value_ptr40 parameters .....	474
3.783. mem_tcam128_set8_entry_mask_ptr32 parameters .....	474
3.784. mem_tcam128_set8_entry_mask_ptr40 parameters .....	475
3.785. mem_tcam128_set8_entry_ptr32 parameters .....	475
3.786. mem_tcam128_set8_entry_ptr40 parameters .....	476
3.787. mem_tcam256_set8_word_ptr32 parameters .....	477
3.788. mem_tcam256_set8_word_ptr40 parameters .....	477
3.789. mem_tcam256_set8_entry_value_ptr32 parameters .....	478
3.790. mem_tcam256_set8_entry_value_ptr40 parameters .....	478
3.791. mem_tcam256_set8_entry_mask_ptr32 parameters .....	479
3.792. mem_tcam256_set8_entry_mask_ptr40 parameters .....	480
3.793. mem_tcam256_set8_entry_ptr32 parameters .....	480
3.794. mem_tcam256_set8_entry_ptr40 parameters .....	481
3.795. mem_tcam384_set8_word_ptr32 parameters .....	481
3.796. mem_tcam384_set8_word_ptr40 parameters .....	482
3.797. mem_tcam384_set8_entry_value_ptr32 parameters .....	482
3.798. mem_tcam384_set8_entry_value_ptr40 parameters .....	483
3.799. mem_tcam384_set8_entry_mask_ptr32 parameters .....	484
3.800. mem_tcam384_set8_entry_mask_ptr40 parameters .....	484
3.801. mem_tcam384_set8_entry_ptr32 parameters .....	485
3.802. mem_tcam384_set8_entry_ptr40 parameters .....	485
3.803. mem_tcam512_set8_word_ptr32 parameters .....	486
3.804. mem_tcam512_set8_word_ptr40 parameters .....	487
3.805. mem_tcam512_set8_entry_value_ptr32 parameters .....	487
3.806. mem_tcam512_set8_entry_value_ptr40 parameters .....	488
3.807. mem_tcam512_set8_entry_mask_ptr32 parameters .....	488
3.808. mem_tcam512_set8_entry_mask_ptr40 parameters .....	489
3.809. mem_tcam512_set8_entry_ptr32 parameters .....	490
3.810. mem_tcam512_set8_entry_ptr40 parameters .....	490
3.811. mem_tcam128_set16_word_ptr32 parameters .....	491
3.812. mem_tcam128_set16_word_ptr40 parameters .....	491
3.813. mem_tcam128_set16_entry_value_ptr32 parameters .....	492
3.814. mem_tcam128_set16_entry_value_ptr40 parameters .....	492
3.815. mem_tcam128_set16_entry_mask_ptr32 parameters .....	493
3.816. mem_tcam128_set16_entry_mask_ptr40 parameters .....	494
3.817. mem_tcam128_set16_entry_ptr32 parameters .....	494
3.818. mem_tcam128_set16_entry_ptr40 parameters .....	495
3.819. mem_tcam256_set16_word_ptr32 parameters .....	496
3.820. mem_tcam256_set16_word_ptr40 parameters .....	496
3.821. mem_tcam256_set16_entry_value_ptr32 parameters .....	497
3.822. mem_tcam256_set16_entry_value_ptr40 parameters .....	497
3.823. mem_tcam256_set16_entry_mask_ptr32 parameters .....	498

3.824. mem_tcam256_set16_entry_mask_ptr40 parameters .....	498
3.825. mem_tcam256_set16_entry_ptr32 parameters .....	499
3.826. mem_tcam256_set16_entry_ptr40 parameters .....	500
3.827. mem_tcam384_set16_word_ptr32 parameters .....	500
3.828. mem_tcam384_set16_word_ptr40 parameters .....	501
3.829. mem_tcam384_set16_entry_value_ptr32 parameters .....	501
3.830. mem_tcam384_set16_entry_value_ptr40 parameters .....	502
3.831. mem_tcam384_set16_entry_mask_ptr32 parameters .....	503
3.832. mem_tcam384_set16_entry_mask_ptr40 parameters .....	503
3.833. mem_tcam384_set16_entry_ptr32 parameters .....	504
3.834. mem_tcam384_set16_entry_ptr40 parameters .....	504
3.835. mem_tcam512_set16_word_ptr32 parameters .....	505
3.836. mem_tcam512_set16_word_ptr40 parameters .....	506
3.837. mem_tcam512_set16_entry_value_ptr32 parameters .....	506
3.838. mem_tcam512_set16_entry_value_ptr40 parameters .....	507
3.839. mem_tcam512_set16_entry_mask_ptr32 parameters .....	507
3.840. mem_tcam512_set16_entry_mask_ptr40 parameters .....	508
3.841. mem_tcam512_set16_entry_ptr32 parameters .....	509
3.842. mem_tcam512_set16_entry_ptr40 parameters .....	509
3.843. mem_tcam128_set32_word_ptr32 parameters .....	510
3.844. mem_tcam128_set32_word_ptr40 parameters .....	510
3.845. mem_tcam128_set32_entry_value_ptr32 parameters .....	511
3.846. mem_tcam128_set32_entry_value_ptr40 parameters .....	511
3.847. mem_tcam128_set32_entry_mask_ptr32 parameters .....	512
3.848. mem_tcam128_set32_entry_mask_ptr40 parameters .....	513
3.849. mem_tcam128_set32_entry_ptr32 parameters .....	513
3.850. mem_tcam128_set32_entry_ptr40 parameters .....	514
3.851. mem_tcam256_set32_word_ptr32 parameters .....	514
3.852. mem_tcam256_set32_word_ptr40 parameters .....	515
3.853. mem_tcam256_set32_entry_value_ptr32 parameters .....	516
3.854. mem_tcam256_set32_entry_value_ptr40 parameters .....	516
3.855. mem_tcam256_set32_entry_mask_ptr32 parameters .....	517
3.856. mem_tcam256_set32_entry_mask_ptr40 parameters .....	517
3.857. mem_tcam256_set32_entry_ptr32 parameters .....	518
3.858. mem_tcam256_set32_entry_ptr40 parameters .....	518
3.859. mem_tcam384_set32_word_ptr32 parameters .....	519
3.860. mem_tcam384_set32_word_ptr40 parameters .....	520
3.861. mem_tcam384_set32_entry_value_ptr32 parameters .....	520
3.862. mem_tcam384_set32_entry_value_ptr40 parameters .....	521
3.863. mem_tcam384_set32_entry_mask_ptr32 parameters .....	521
3.864. mem_tcam384_set32_entry_mask_ptr40 parameters .....	522
3.865. mem_tcam384_set32_entry_ptr32 parameters .....	522
3.866. mem_tcam384_set32_entry_ptr40 parameters .....	523
3.867. mem_tcam512_set32_word_ptr32 parameters .....	524
3.868. mem_tcam512_set32_word_ptr40 parameters .....	524
3.869. mem_tcam512_set32_entry_value_ptr32 parameters .....	525
3.870. mem_tcam512_set32_entry_value_ptr40 parameters .....	525
3.871. mem_tcam512_set32_entry_mask_ptr32 parameters .....	526
3.872. mem_tcam512_set32_entry_mask_ptr40 parameters .....	526
3.873. mem_tcam512_set32_entry_ptr32 parameters .....	527
3.874. mem_tcam512_set32_entry_ptr40 parameters .....	528
3.875. mem_tcam128_lookup8_ptr32 parameters .....	528
3.876. mem_tcam128_lookup8_ptr40 parameters .....	529

3.877. mem_tcam256_lookup8_ptr32 parameters .....	530
3.878. mem_tcam256_lookup8_ptr40 parameters .....	530
3.879. mem_tcam384_lookup8_ptr32 parameters .....	531
3.880. mem_tcam384_lookup8_ptr40 parameters .....	532
3.881. mem_tcam512_lookup8_ptr32 parameters .....	533
3.882. mem_tcam512_lookup8_ptr40 parameters .....	533
3.883. mem_tcam128_lookup16_ptr32 parameters .....	534
3.884. mem_tcam128_lookup16_ptr40 parameters .....	535
3.885. mem_tcam256_lookup16_ptr32 parameters .....	535
3.886. mem_tcam256_lookup16_ptr40 parameters .....	536
3.887. mem_tcam384_lookup16_ptr32 parameters .....	537
3.888. mem_tcam384_lookup16_ptr40 parameters .....	537
3.889. mem_tcam512_lookup16_ptr32 parameters .....	538
3.890. mem_tcam512_lookup16_ptr40 parameters .....	539
3.891. mem_tcam128_lookup24_ptr32 parameters .....	540
3.892. mem_tcam128_lookup24_ptr40 parameters .....	540
3.893. mem_tcam256_lookup24_ptr32 parameters .....	541
3.894. mem_tcam256_lookup24_ptr40 parameters .....	542
3.895. mem_tcam384_lookup24_ptr32 parameters .....	542
3.896. mem_tcam384_lookup24_ptr40 parameters .....	543
3.897. mem_tcam512_lookup24_ptr32 parameters .....	544
3.898. mem_tcam512_lookup24_ptr40 parameters .....	544
3.899. mem_tcam128_lookup32_ptr32 parameters .....	545
3.900. mem_tcam128_lookup32_ptr40 parameters .....	546
3.901. mem_tcam256_lookup32_ptr32 parameters .....	547
3.902. mem_tcam256_lookup32_ptr40 parameters .....	547
3.903. mem_tcam384_lookup32_ptr32 parameters .....	548
3.904. mem_tcam384_lookup32_ptr40 parameters .....	549
3.905. mem_tcam512_lookup32_ptr32 parameters .....	549
3.906. mem_tcam512_lookup32_ptr40 parameters .....	550
3.907. mem_tcam_word_to_entry_index8 parameters .....	551
3.908. mem_tcam_entry_value_to_word_index8 parameters .....	551
3.909. mem_tcam_entry_mask_to_word_index8 parameters .....	552
3.910. mem_tcam_word_to_entry_index16 parameters .....	552
3.911. mem_tcam_entry_value_to_word_index16 parameters .....	553
3.912. mem_tcam_entry_mask_to_word_index16 parameters .....	553
3.913. mem_tcam_word_to_entry_index32 parameters .....	554
3.914. mem_tcam_entry_value_to_word_index32 parameters .....	554
3.915. mem_tcam_entry_mask_to_word_index32 parameters .....	555
3.916. enum MEM_PACKET_REWRITE_SCRIPT_OFFSET .....	555
3.917. enum MEM_PACKET_MASTER_BUCKET .....	556
3.918. enum MEM_PACKET_LENGTH .....	556
3.919. enum MEM_PACKET_TRANSFER_TYPE .....	556
3.920. struct mem_packet_header_t .....	557
3.921. union mem_packet_alloc_response_t .....	557
3.922. union mem_packet_complete_request_t .....	558
3.923. union mem_packet_read_status_response_t .....	558
3.924. typedef MEM_PACKET_REWRITE_SCRIPT_OFFSET .....	559
3.925. typedef MEM_PACKET_MASTER_BUCKET .....	559
3.926. typedef MEM_PACKET_LENGTH .....	559
3.927. typedef MEM_PACKET_TRANSFER_TYPE .....	559
3.928. typedef mem_packet_read_status_response_t .....	560
3.929. typedef mem_packet_read_status_response_in_read_reg_t .....	560

3.930. typedef mem_packet_alloc_response_t .....	560
3.931. typedef mem_packet_alloc_response_in_read_reg_t .....	560
3.932. typedef mem_packet_complete_request_t .....	560
3.933. typedef mem_packet_header_t .....	561
3.934. cmd_mem_packet_credit_get parameters .....	561
3.935. cmd_mem_packet_alloc parameters .....	562
3.936. cmd_mem_packet_alloc_poll parameters .....	563
3.937. cmd_mem_packet_free parameters .....	564
3.938. cmd_mem_packet_free_and_return_pointer parameters .....	565
3.939. cmd_mem_packet_free_and_signal parameters .....	565
3.940. cmd_mem_packet_return_pointer parameters .....	565
3.941. cmd_mem_packet_complete_drop parameters .....	566
3.942. cmd_mem_packet_complete_unicast parameters .....	566
3.943. cmd_mem_packet_complete_multicast parameters .....	567
3.944. cmd_mem_packet_complete_multicast_free parameters .....	567
3.945. cmd_mem_packet_read_packet_status parameters .....	568
3.946. cmd_mem_packet_wait_packet_status parameters .....	568
3.947. cmd_mem_packet_add_thread parameters .....	569
3.948. cmd_mem_pe_dma_to_memory_packet parameters .....	570
3.949. cmd_mem_pe_dma_to_memory_packet_free parameters .....	571
3.950. cmd_mem_pe_dma_to_memory_packet_swap parameters .....	572
3.951. cmd_mem_pe_dma_to_memory_packet_free_swap parameters .....	573
3.952. cmd_mem_pe_dma_to_memory_buffer parameters .....	574
3.953. cmd_mem_pe_dma_to_memory_buffer_swap parameters .....	574
3.954. cmd_mem_pe_dma_to_memory_buffer_le parameters .....	575
3.955. cmd_mem_pe_dma_to_memory_buffer_le_swap parameters .....	576
3.956. cmd_mem_pe_dma_from_memory_buffer parameters .....	577
3.957. cmd_mem_pe_dma_from_memory_buffer_swap parameters .....	578
3.958. cmd_mem_pe_dma_from_memory_buffer_le parameters .....	579
3.959. cmd_mem_pe_dma_from_memory_buffer_le_swap parameters .....	579
3.960. cmd_mem_pe_dma_to_memory_indirect parameters .....	580
3.961. cmd_mem_pe_dma_to_memory_indirect_swap parameters .....	581
3.962. cmd_mem_pe_dma_to_memory_indirect_free parameters .....	581
3.963. cmd_mem_pe_dma_to_memory_indirect_free_swap parameters .....	582
3.964. MU Statistics Defines .....	585
3.965. enum MEM_STATS_BASE_ADDRESS_SELECT .....	587
3.966. enum MEM_STATS_ADDRESS_PACK_CONFIG .....	587
3.967. union cmd_mem_stats_log_command_address_format_t .....	587
3.968. union cmd_mem_stats_read_command_address_format_t .....	588
3.969. union mem_stats_packed_address_detail_t .....	588
3.970. union mem_stats_unpacked_address_detail_t .....	588
3.971. typedef MEM_STATS_BASE_ADDRESS_SELECT .....	589
3.972. typedef MEM_STATS_ADDRESS_PACK_CONFIG .....	589
3.973. typedef cmd_mem_stats_log_command_address_format_t .....	589
3.974. typedef cmd_mem_stats_read_command_address_format_t .....	589
3.975. typedef mem_stats_packed_address_detail_t .....	589
3.976. typedef mem_stats_unpacked_address_detail_t .....	590
3.977. cmd_mem_stats_read parameters .....	590
3.978. cmd_mem_stats_read_and_clear parameters .....	590
3.979. cmd_mem_stats_log parameters .....	591
3.980. cmd_mem_stats_log_saturate parameters .....	592
3.981. cmd_mem_stats_log_event parameters .....	592
3.982. cmd_mem_stats_log_event_saturate parameters .....	593

3.983. MU LB Defines .....	594
3.984. union mem_lb_bucket_dcache_address_format_t .....	595
3.985. union mem_lb_bucket_local_address_format_t .....	595
3.986. union mem_lb_dcache_stats_address_format_t .....	595
3.987. union mem_lb_descriptor_address_format_t .....	596
3.988. union mem_lb_desc_t .....	596
3.989. union mem_lb_id_table_address_format_t .....	596
3.990. union mem_lb_local_stats_address_format_t .....	597
3.991. union mem_lb_lookup_bundle_id_result_t .....	597
3.992. union mem_lb_lookup_command_format_t .....	597
3.993. union mem_lb_lookup_direct_result_t .....	598
3.994. typedef MEM_LB_LOCATION .....	598
3.995. typedef mem_lb_desc_t .....	598
3.996. typedef mem_lb_desc_in_write_reg_t .....	599
3.997. typedef mem_lb_desc_in_read_reg_t .....	599
3.998. typedef mem_lb_local_stats_address_format_t .....	599
3.999. typedef mem_lb_dcache_stats_address_format_t .....	599
3.1000. typedef mem_lb_descriptor_address_format_t .....	599
3.1001. typedef mem_lb_id_table_address_format_t .....	600
3.1002. typedef mem_lb_bucket_local_address_format_t .....	600
3.1003. typedef mem_lb_bucket_dcache_address_format_t .....	600
3.1004. typedef mem_lb_lookup_command_format_t .....	600
3.1005. typedef mem_lb_lookup_command_format_in_write_reg_t .....	600
3.1006. typedef mem_lb_lookup_bundle_id_result_t .....	601
3.1007. typedef mem_lb_lookup_bundle_id_result_in_read_reg_t .....	601
3.1008. typedef mem_lb_lookup_direct_result_t .....	601
3.1009. typedef mem_lb_lookup_direct_result_in_read_reg_t .....	601
3.1010. cmd_mem_lb_write_desc parameters .....	602
3.1011. cmd_mem_lb_read_desc parameters .....	602
3.1012. cmd_mem_lb_write_id_table parameters .....	603
3.1013. cmd_mem_lb_read_id_table parameters .....	603
3.1014. cmd_mem_lb_bucket_write_local parameters .....	604
3.1015. cmd_mem_lb_bucket_read_local parameters .....	605
3.1016. cmd_mem_lb_bucket_write_dcache parameters .....	605
3.1017. cmd_mem_lb_bucket_read_dcache parameters .....	606
3.1018. cmd_mem_lb_stats_read_and_clear_local parameters .....	606
3.1019. cmd_mem_lb_stats_read_local parameters .....	607
3.1020. cmd_mem_lb_stats_read_and_clear_dcache parameters .....	607
3.1021. cmd_mem_lb_stats_read_dcache parameters .....	608
3.1022. cmd_mem_lb_lookup_bundle_id parameters .....	611
3.1023. cmd_mem_lb_lookup_dcache parameters .....	612
3.1024. cmd_mem_lb_lookup_id_table parameters .....	612
3.1025. cmd_mem_lb_init parameters .....	615
3.1026. struct mem_lookup_result_t .....	618
3.1027. enum MEM_LOOKUP_DLUT_SMALL_SIZE .....	618
3.1028. enum MEM_LOOKUP_DLUT_LARGE_SIZE .....	619
3.1029. enum MEM_LOOKUP_RESULT .....	619
3.1030. enum MEM_LOOKUP_DLUT_TYPE .....	619
3.1031. enum MEM_LOOKUP_ALUT_SIZE .....	620
3.1032. enum MEM_LOOKUP_ALUT_COMMANDS .....	620
3.1033. enum MEM_LOOKUP_HASH_TABLE_SIZE .....	621
3.1034. enum MEM_LOOKUP_HASH_STARTING_BIT .....	621
3.1035. enum MEM_LOOKUP_HASH_COMMANDS .....	622

3.1036. union mem_lookup_alut_t .....	623
3.1037. union mem_lookup_dlut_large_recursive_t .....	623
3.1038. union mem_lookup_dlut_large_t .....	624
3.1039. union mem_lookup_dlut_small_recursive_t .....	624
3.1040. union mem_lookup_dlut_small_t .....	625
3.1041. union mem_lookup_hash_table_t .....	625
3.1042. union mem_lookup_recursive_hash_table_t .....	626
3.1043. typedef MEM_LOOKUP_DLUT_SMALL_SIZE .....	626
3.1044. typedef MEM_LOOKUP_DLUT_LARGE_SIZE .....	627
3.1045. typedef MEM_LOOKUP_RESULT .....	627
3.1046. typedef MEM_LOOKUP_DLUT_TYPE .....	627
3.1047. typedef mem_lookup_dlut_small_t .....	627
3.1048. typedef mem_lookup_dlut_large_t .....	627
3.1049. typedef mem_lookup_dlut_small_recursive_t .....	628
3.1050. typedef mem_lookup_dlut_large_recursive_t .....	628
3.1051. typedef mem_lookup_result_t .....	628
3.1052. typedef mem_lookup_result_in_read_reg_t .....	628
3.1053. typedef MEM_LOOKUP_ALUT_SIZE .....	628
3.1054. typedef MEM_LOOKUP_ALUT_COMMANDS .....	629
3.1055. typedef mem_lookup_alut_t .....	629
3.1056. typedef MEM_LOOKUP_HASH_TABLE_SIZE .....	629
3.1057. typedef MEM_LOOKUP_HASH_STARTING_BIT .....	629
3.1058. typedef MEM_LOOKUP_HASH_COMMANDS .....	629
3.1059. typedef mem_lookup_hash_table_t .....	630
3.1060. typedef mem_lookup_recursive_hash_table_t .....	630
3.1061. cmd_mem_lookup parameters .....	630
3.1062. cmd_PCIE_read_ptr32 parameters .....	631
3.1063. cmd_PCIE_read_ind_ptr32 parameters .....	631
3.1064. cmd_PCIE_write_ptr32 parameters .....	632
3.1065. cmd_PCIE_write_ind_ptr32 parameters .....	632
3.1066. cmd_PCIE_read_pci_ptr32 parameters .....	633
3.1067. cmd_PCIE_read_pci_ind_ptr32 parameters .....	633
3.1068. cmd_PCIE_write_pci_ptr32 parameters .....	634
3.1069. cmd_PCIE_write_pci_ind_ptr32 parameters .....	634
3.1070. cmd_PCIE_read_rid_ptr32 parameters .....	635
3.1071. cmd_PCIE_read_rid_ind_ptr32 parameters .....	635
3.1072. cmd_PCIE_write_rid_ptr32 parameters .....	636
3.1073. cmd_PCIE_write_rid_ind_ptr32 parameters .....	636
3.1074. cmd_PCIE_read_ptr40 parameters .....	637
3.1075. cmd_PCIE_write_ptr40 parameters .....	637
3.1076. cmd_PCIE_read_pci_ptr40 parameters .....	638
3.1077. cmd_PCIE_write_pci_ptr40 parameters .....	638
3.1078. cmd_PCIE_read_rid_ptr40 parameters .....	639
3.1079. cmd_PCIE_write_rid_ptr40 parameters .....	639
3.1080. enum ILA_OPERATION_MODE .....	640
3.1081. enum ILA_LOGIC_RESET .....	640
3.1082. enum ILA_CHANNEL .....	640
3.1083. union ila_config_control_register_format_t .....	641
3.1084. union ila_cpp_to_ila_bar_register_format_t .....	641
3.1085. union ila_dma_command_register_format_t .....	642
3.1086. union ila_internal_address_format_t .....	642
3.1087. union ila_to_cpp_bar_register_format_t .....	643
3.1088. typedef ILA_OPERATION_MODE .....	643

3.1089. <code>typedef ILA_LOGIC_RESET</code> .....	643
3.1090. <code>typedef ILA_CHANNEL</code> .....	643
3.1091. <code>typedef ila_config_control_register_format_t</code> .....	644
3.1092. <code>typedef ila_internal_address_format_t</code> .....	644
3.1093. <code>typedef ila_to_cpp_bar_register_format_t</code> .....	644
3.1094. <code>typedef ila_cpp_to_ilabar_register_format_t</code> .....	644
3.1095. <code>typedef ila_dma_command_register_format_t</code> .....	644
3.1096. <code>cmd_ila_write_ptr40</code> parameters .....	645
3.1097. <code>cmd_ila_read_ptr40</code> parameters .....	645
3.1098. <code>cmd_ila_write_check_error_ptr40</code> parameters .....	646
3.1099. <code>cmd_ila_read_check_error_ptr40</code> parameters .....	646
3.1100. <code>cmd_ila_write_internal_ptr40</code> parameters .....	647
3.1101. <code>cmd_ila_read_internal_ptr40</code> parameters .....	648
3.1102. <code>cmd_nbi_write</code> parameters .....	648
3.1103. <code>cmd_nbi_read</code> parameters .....	649
3.1104. <code>enum sync_t</code> .....	650
3.1105. <code>enum signal_t</code> .....	650
3.1106. <code>struct SIGNAL_PAIR</code> .....	651
3.1107. <code>typedef SIGNAL_MASK</code> .....	651
3.1108. <code>typedef SIGNAL</code> .....	651
3.1109. <code>typedef SIGNAL_PAIR</code> .....	651
3.1110. <code>signal_same_ME</code> parameters .....	652
3.1111. <code>signal_same_ME_next_ctx</code> parameters .....	652
3.1112. <code>signal_prev_ME</code> parameters .....	652
3.1113. <code>signal_prev_ME_this_ctx</code> parameters .....	653
3.1114. <code>signal_next_ME</code> parameters .....	653
3.1115. <code>signal_next_ME_this_ctx</code> parameters .....	654
3.1116. <code>enum swpack_t</code> .....	657
3.1117. <code>enum inp_state_t</code> .....	657
3.1118. Miscellaneous Defines .....	660
3.1119. <code>union cam_lookup_t</code> .....	661
3.1120. <code>nn_ring_enqueue_incr</code> parameters .....	662
3.1121. <code>byte_align_block_be</code> parameters .....	662
3.1122. <code>byte_align_block_le</code> parameters .....	663
3.1123. <code>ffs</code> parameters .....	663
3.1124. <code>bswap</code> parameters .....	664
3.1125. <code>bitswap</code> parameters .....	664
3.1126. <code>cam_write</code> parameters .....	665
3.1127. <code>cam_lookup</code> parameters .....	665
3.1128. <code>cam_write_state</code> parameters .....	666
3.1129. <code>cam_read_tag</code> parameters .....	666
3.1130. <code>cam_read_state</code> parameters .....	666
3.1131. <code>multiply_24x8</code> parameters .....	667
3.1132. <code>multiply_16x16</code> parameters .....	667
3.1133. <code>multiply_32x32_lo</code> parameters .....	668
3.1134. <code>multiply_32x32_hi</code> parameters .....	668
3.1135. <code>multiply_32x32</code> parameters .....	668
3.1136. <code>_set_timestamp</code> parameters .....	669
3.1137. <code>_timestamp_stop</code> parameters .....	669
3.1138. <code>_sleep</code> parameters .....	670
3.1139. <code>_set_profile_count</code> parameters .....	671
3.1140. <code>_profile_count_stop</code> parameters .....	672
3.1141. <code>assert_range</code> parameters .....	672

3.1142. bit_test parameters .....	672
3.1143. inp_state_test parameters .....	673

# 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.
<b>Command1 Command2 Command3</b>	(For Windows*) Selecting cascading options. Indicates that you should follow these steps: <ul style="list-style-type: none"><li>• Click on <b>Command1</b>, which offers options including Command2</li><li>• Click on <b>Command2</b>, which offers options including Command3</li><li>• Click on <b>Command3</b>.</li></ul> For example: <b>Start Programs Accessories Command Prompt</b> .
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 <b>n</b> .
<code>long labs(long n);</code>	Returns absolute value of <b>n</b> (same as <code>abs()</code> ).
<code>int atoi(__declspec(sram) CHAR *s); __int64 atoi64(__declspec(sram) CHAR *s)</code>	Converts string <b>s</b> 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 <b>c</b> .
<code>int toupper(int c);</code>	Returns uppercase of character <b>c</b> .
<code>int islower(int c);</code>	Returns 1 if <b>c</b> is a lower-case character, and 0 otherwise.
<code>int isupper(int c);</code>	Returns 1 if <b>c</b> 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

```
typedef int                      bool;
typedef void                     VOID;
typedef unsigned char            U8;
typedef char                      I8;
```

```
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 <b>src</b> in MEM (IMEM/EMEM/CTM) to <b>dest</b> 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 <b>src</b> in LMEM to <b>dest</b> 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 <b>src</b> in SRAM to <b>dest</b> 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 <b>src</b> in MEM (IMEM/EMEM/CTM) to <b>dest</b> 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 <b>src</b> in LMEM to <b>dest</b> 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 <b>src</b> in SRAM to <b>dest</b> in LMEM.
<code>__declspec(sram) void* memcpy_sram_mem(         __declspec(sram) const void *src,         unsigned int count);</code>	Copies from <b>src</b> in MEM (IMEM/EMEM/CTM) to <b>dest</b> 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 <b>src</b> in local memory (LMEM) to <b>dest</b> in SRAM.
<code>__declspec(sram) void* memcpy_sram_sram(               __declspec(sram) void *dest,               __declspec(sram) const void *src,               unsigned int count);</code>	Copies from <b>src</b> in SRAM to <b>dest</b> 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 <b>src</b> in CLUSTER LOCAL SCRATCH to <b>dest</b> in SRAM.
<code>__declspec(mem) void* memcpy_mem_cls(               __declspec(mem) void *dest,               __declspec(cls) const void *src,               unsigned int count);</code>	Copies from <b>src</b> in CLUSTER LOCAL SCRATCH to <b>dest</b> 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 <b>src</b> in SRAM to <b>dest</b> 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 <b>src</b> in MEM (IMEM/EMEM/CTM) to <b>dest</b> 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 <b>src</b> in CLUSTER LOCAL SCRATCH to <b>dest</b> 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 <b>src</b> in LOCAL MEMORY to <b>dest</b> 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 <b>src</b> in CLUSTER LOCAL SCRATCH to <b>dest</b> 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 <b>src</b> in SRAM to <b>dest</b> 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 <b>src</b> in local memory (LMEM) to <b>dest</b> 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 <b>src</b> in MEM (IMEM/EMEM/CTM) to <b>dest</b> 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 <b>src</b> in SRAM to <b>dest</b> 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 <b>src</b> in LMEM to <b>dest</b> 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 <b>src</b> in MEM (IMEM/EMEM/CTM) to <b>dest</b> in LMEM.
<code>__declspec(mem) void * memmove_mem_sram(                   __declspec(mem) void *dest,                   __declspec(sram) const void *src,                   unsigned int count);</code>	Copies from <b>src</b> in SRAM to <b>dest</b> 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 <b>src</b> in LMEM to <b>dest</b> 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 <b>src</b> in MEM (IMEM/EMEM/CTM) to <b>dest</b> 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 * memset_sram(__declspec(sram) void *dest, CHAR c, unsigned int count);</code>	Fills the <b>dest</b> buffer in SRAM with <b>count</b> number of 8-bit characters specified by <b>c</b> . This is identical to the <code>memset()</code> function.
<code>__declspec(local_mem) void * memset_lmem(__declspec(local_mem) void *dest, CHAR c, unsigned int count);</code>	Fills the <b>dest</b> buffer in local memory with <b>count</b> number of 8-bit characters specified by <b>c</b> .
<code>__declspec(mem) void * memset_mem(__declspec(mem) void *dest, CHAR c, unsigned int count);</code>	Fills the <b>dest</b> buffer in MEM (IMEM/EMEM/CTM) with <b>count</b> number of 8-bit characters specified by <b>c</b> .
<code>__declspec(cls) void * memset_cls(__declspec(cls) void *dest, CHAR c, unsigned int count);</code>	Fills the <b>dest</b> buffer in CLUSTER LOCAL SCRATCH with <b>count</b> number of 8-bit characters specified by <b>c</b> .

## 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 <b>count</b> number of bytes in <b>buf1</b> and <b>buf2</b> , 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 <b>buf1</b> in SRAM and <b>buf2</b> in LMEM.
<code>int memcmp_sram_mem( __declspec(sram) const void *buf1,                          __declspec(mem) const void *buf2,                          unsigned int count);</code>	Compares <b>count</b> number of bytes between <b>buf1</b> in SRAM and <b>buf2</b> 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 <b>count</b> 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 <b>count</b> 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 <b>count</b> 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 <b>count</b> 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 <b>count</b> 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 <b>count</b> 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 <b>count</b> 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 <b>count</b> 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 <b>count</b> 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 <b>count</b> 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 <b>count</b> 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 <b>count</b> 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 <b>count</b> 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 <b>c</b> 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 <b>c</b> 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 <b>c</b> 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 <b>c</b> 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 <b>s</b> in SRAM.
unsigned int strlen_lmem(__declspec(local_mem) const CHAR *s);	Returns the length of string <b>s</b> in LMEM.
unsigned int strlen_mem(__declspec(mem) const CHAR *s);	Returns the length of string <b>s</b> in MEM (IMEM/EMEM/CTM).
unsigned int strlen_cls(__declspec(cls) const CHAR *s);	Returns the length of string <b>s</b> 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 <b>s1</b> with string <b>s2</b> 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 <b>s1</b> with string <b>s2</b> in LMEM
unsigned int strcmp_mem( __declspec(mem) const CHAR *s1, __declspec(mem) const CHAR *s2);	Compares string <b>s1</b> with string <b>s2</b> in MEM (IMEM/EMEM/CTM)
unsigned int strcmp_cls( __declspec(cls) const CHAR *s1, __declspec(cls) const CHAR *s2);	Compares string <b>s1</b> with string <b>s2</b> 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 <b>count</b> number of bytes in string <b>s1</b> with string <b>s2</b> in SRAM. This is identical to the strncmp() function.
int strncmp_lmem( __declspec(local_mem) const CHAR *s1,	Compares <b>count</b> number of bytes in string <b>s1</b> with string <b>s2</b> 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 <b>count</b> number of bytes in string <b>s1</b> with string <b>s2</b> 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 <b>count</b> number of bytes in string <b>s1</b> with string <b>s2</b> 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 <b>src</b> to string <b>dest</b> 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 <b>src</b> to string <b>dest</b> in LMEM.
<pre style="margin: 0;">__declspec(mem) CHAR * strcpy_mem( __declspec(mem) CHAR *dest, __declspec(mem) const CHAR *src);</pre>	Copies string <b>src</b> to string <b>dest</b> 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 <b>src</b> to string <b>dest</b> 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 <b>count</b> number of bytes from string <b>src</b> to string <b>dest</b> 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 <b>count</b> number of bytes from string <b>src</b> to string <b>dest</b> in LMEM.
<code>__declspec(mem) CHAR * strncpy_mem(                   __declspec(mem) CHAR *dest,                   __declspec(mem) const CHAR *src,                   unsigned int count);</code>	Copies <b>count</b> number of bytes from string <b>src</b> to string <b>dest</b> 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 <b>count</b> number of bytes from string <b>src</b> to string <b>dest</b> 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 <b>src</b> to string <b>dest</b> 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 <b>src</b> to string <b>dest</b> in LMEM.
<code>__declspec(mem) CHAR * strcat_mem(                   __declspec(mem) CHAR *dest,                   __declspec(mem) const CHAR *src);</code>	Appends string <b>src</b> to string <b>dest</b> in MEM (IMEM/EMEM/CTM).
<code>__declspec(cls) CHAR * strcat_cls(                   __declspec(cls) CHAR *dest,                   __declspec(cls) const CHAR *src);</code>	Appends string <b>src</b> to string <b>dest</b> 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 <b>count</b> number of bytes from string <b>src</b> to string <b>dest</b> 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 <b>count</b> number of bytes from string <b>src</b> to string <b>dest</b> in LMEM.

Prototype	Description
<code>__declspec(local_mem) const CHAR *src, unsigned int count);</code>	
<code>__declspec(mem) CHAR * strncat_mem(     __declspec(mem) CHAR *dest,     __declspec(mem) const CHAR *src,     unsigned int count);</code>	Appends <b>count</b> number of bytes from string <b>src</b> to string <b>dest</b> in MEM (IMEM/EMEM/CTM).
<code>__declspec(cls) CHAR * strncat_cls(     __declspec(cls) CHAR *dest,     __declspec(cls) const CHAR *src,     unsigned int count);</code>	Appends <b>count</b> number of bytes from string <b>src</b> to string <b>dest</b> 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>__declspec(sram) CHAR * strchr_sram(                                   __declspec(sram) CHAR *s,                                   CHAR *c)</code>	Returns the address of the first occurrence of <b>c</b> in string <b>s</b> in SRAM. This is identical to the strchr() function.
<code>__declspec(local_mem) CHAR * strcat_lmem(                                   __declspec(local_mem) CHAR *s,                                   CHAR *c);</code>	Returns the address of the first occurrence of <b>c</b> in string <b>s</b> in LMEM.
<code>__declspec(mem) CHAR * strchr_mem(                                   __declspec(mem) CHAR *s,                                   CHAR *c);</code>	Returns the address of the first occurrence of <b>c</b> in string <b>s</b> in MEM (IMEM/EMEM/CTM).
<code>__declspec(cls) CHAR * strchr_cls(                                   __declspec(cls) CHAR *s,                                   CHAR *c);</code>	Returns the address of the first occurrence of <b>c</b> in string <b>s</b> 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 <b>c</b> in string <b>s</b> 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 <b>c</b> in string <b>s</b> in LMEM.
<code>__declspec(mem) CHAR * strrchr_mem(           __declspec(mem) CHAR *s,           CHAR *c);</code>	Returns the address of the last occurrence of <b>c</b> in string <b>s</b> 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 <b>c</b> in string <b>s</b> 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 <b>s2</b> in string <b>s1</b> in SRAM. This is identical to the strstr() 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 <b>s2</b> in string <b>s1</b> 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 <b>s2</b> in string <b>s1</b> 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 <b>s2</b> in string <b>s1</b> 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 <b>s</b> that consists entirely of characters from <b>chset</b> in SRAM. This is identical to the strspn() 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 <b>s</b> that consists entirely of characters from <b>chset</b> 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 <b>s</b> that consists entirely of characters from <b>chset</b> 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 <b>s</b> that consists entirely of characters from <b>chset</b> 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 <b>s</b> that consists entirely of characters not contained in <b>chset</b> 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 <b>s</b> that consists entirely of characters not contained in <b>chset</b> in LMEM.
unsigned int strcspn_mem( __declspec(mem) const CHAR *chset; __declspec(sram) const CHAR *chset);	Returns the length of the substring in string <b>s</b> that consists entirely of characters not contained in <b>chset</b> 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 <b>s</b> that consists entirely of characters not contained in <b>chset</b> 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 <b>s1</b> that comes from <b>s2</b> 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 <b>s</b> that comes from <b>s2</b> 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 <b>s</b> that comes from <b>s2</b> 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 <b>s</b> that comes from <b>s2</b> 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 <b>s</b> delimited by

Prototype	Description
<code>__declspec(sram) CHAR *s,</code> <code>__declspec(sram) const CHAR *delimit);</code>	a character from <b>delimit</b> 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 <b>s</b> delimited by a character from <b>delimit</b> 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 <b>s</b> delimited by a character from <b>delimit</b> 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 <b>s</b> delimited by a character from <b>delimit</b> 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 <b>s</b> to a long integer using the specified <b>base</b> 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 <b>s</b> to a long integer using the specified <b>base</b> 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 <b>s</b> to a long integer using the specified <b>base</b> in MEM (IMEM/EMEM/CTM).

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

## 3. Intrinsic Functions

---

Generally speaking, intrinsic functions support features of the Netronome Network Flow Processors that are not easily accessible using standard C. Two distinct sets of intrinsic functions exist: the compiler intrinsics, and the user intrinsics. The compiler intrinsics are built into the C compiler, whereas the user intrinsics are implemented in Micro-C and use the keyword `_intrinsic` which makes available special features of Micro-C.

The compiler intrinsic functions are documented in the *Network Flow C Compiler User's Guide* and this chapter covers the user intrinsic functions as well as the data types and structures needed to use them.

The descriptions here are not intended to be complete descriptions of the intrinsic functions—they must be supplemented with the information in the relevant *Netronome Network Flow Processor 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.

For documentation purposes, the user intrinsic functions have been divided 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 Intrinsics
- NBI Intrinsics
- SRAM Intrinsics
- Synchronization Intrinsics
- Unaligned Data Access Intrinsics
- Miscellaneous Intrinsics
- Math Intrinsics

Intrinsic functions do not generate a function call, being expanded as inline code by the compiler.

## 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`.

FIX ME doxy\_Indirect\_Reference\_32xx\_unions.xml

## 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 3.1. union `override_alu_ind_t`**

Type	Name	Description
uint32_t	data16:16	Override data reference or byte mask if data reference is not used.
uint32_t	reserved:1	Reserved.
uint32_t	override_signal_ctx:1	Override signal context.
uint32_t	override_signal_num:1	Override signal number.
uint32_t	ref_count:5	Number of bytes, long/quad-words to access.
uint32_t	override_length:1	Override length.
uint32_t	override_bytemask_csr:1	Override byte mask in IND_CSR.
uint32_t	override_data:3	Override data.

Type	Name	Description
		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.
uint32_t	override_master:2	Override master.  0x01: Island and master. 0x02: Island, data master, signal master. 0x03: Island.
uint32_t	override_signal_master:1	Override signal master.
uint32_t	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 3.2. union override\_csr\_ind\_t**

Type	Name	Description
uint32_t	reserved_1:2	Reserved.
uint32_t	island:6	Override island when field is set in override_alu_ind_t.
uint32_t	master:4	Override data master when field is set in override_alu_ind_t.
uint32_t	signal_master:4	Override signal master when field is set in override_alu_ind_t.
uint32_t	signal_ctx:3	Override signal context when field is set in override_alu_ind_t.
uint32_t	signal_num:4	Override signal number when field is set in override_alu_ind_t.
uint32_t	reserved_2:1	Reserved.
uint32_t	byte_mask:8	Override byte mask when field is set in override_alu_ind_t.
uint32_t	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 3.3. enum ovr\_field\_t**

Name	Description
ovr_byte_mask	Override bytemask. In NFP-6xxx indirect ref mode the bytemask is in Indirect CSR
ovr_length	Override the count/length.
ovr_signal_ctx	Override signal context.
ovr_signal_master	Override signal master.
ovr_signal_number	Override signal number.
ovr_data_full_ref	Override the full data reference. NFP-6xxx indirect ref mode only
ovr_data_16bit_imm	Override 16-bit immediate data. NFP-6xxx indirect ref mode only
ovr_data_ctx	Override data context only. NFP-6xxx indirect ref mode only
ovr_data_and_signal_ctx	Override data and signal contexts from DATA16. NFP-6xxx indirect ref mode only
ovr_data_ctx_offset	Override per-context offset of the data reference. NFP-6xxx indirect ref mode only
ovr_data_byte_mask	Override byte mask from DATA16. NFP-6xxx indirect ref mode only
ovr_island_and_data_master	Override island and data master. NFP-6xxx indirect ref mode only
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 `uint32_t`.

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;

    uint32_t write_data = 0x12;
    uint32_t 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;

    uint32_t write_data = 0x12;
    uint32_t 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 3.4. struct generic\_ind\_t**

Type	Name	Description
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, uint32_t 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 uint32_t xfer_wr[16];
    SIGNAL      sig;
    uint32_t    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 uint32_t address[2];
    volatile __xrw uint32_t      xfer_read_write[2];
    uint32_t                   override_island = 35; // i25
    uint32_t                   override_me = 4;        // ME 4

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

    cmd_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 );

    cmd_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;
    uint32_t                        length = 12;
    volatile __xread uint32_t       xfer_read[12];
    volatile __xwrite uint32_t      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));

    cmd_cls_test_and_add_sat_ind_ptr40(xfer_read, xfer_write, address, length, add_sat_ind, sig_done,
    wait_for_all(&override_signal));
}
```

**Table 3.5. 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.
uint32_t	<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, int32_t value)
```

**Description:**

Set the value of an override field.

**Table 3.6. 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.
int32_t	<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 cmd\_arm\_read

**Prototype:**

```
void cmd_arm_read(__xread void* data, volatile void __sram* address, uint32_t count,
sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

Read data from ARM into transfer registers.

**Table 3.7. cmd\_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
<code>uint32_t</code>	<i>count</i>	Number of 32-bit words to read
<code>sync_t</code>	<i>sync</i>	Type of synchronization to use (sig_done or ctx_swap)
<code>SIGNAL*</code>	<i>sig_ptr</i>	Signal to raise upon completion

#### 3.5.1.2 cmd\_arm\_read\_ind

**Prototype:**

```
void cmd_arm_read_ind(__xread void* data, volatile void __sram* address, uint32_t max_nn,
generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

Read data from ARM into transfer register indirect mode.

**Table 3.8. cmd\_arm\_read\_ind parameters**

Type	Name	Description
__xread void*	data	Address to read data into
volatile void __sram*	address	Address to read data from
uint32_t	max_nn	Maximum number of 32-bit words to read
generic_ind_t	ind	Indirect word
sync_t	sync	Type off synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

### 3.5.1.3 cmd\_arm\_write

**Prototype:**

```
void cmd_arm_write(__xwrite void* data, volatile void __sram* address, uint32_t count,
sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

Write data to ARM from transfer register.

**Table 3.9. cmd\_arm\_write parameters**

Type	Name	Description
__xwrite void*	data	Data to write
volatile void __sram*	address	Address to write data to
uint32_t	count	Number of 32-bit words to write
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

### 3.5.1.4 cmd\_arm\_write\_ind

**Prototype:**

```
void cmd_arm_write_ind(__xwrite void* data, volatile void __sram* address, uint32_t max_nn,
generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

Write data to ARM from transfer register indirect mode.

**Table 3.10. cmd\_arm\_write\_ind parameters**

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Data to write
<code>volatile void __sram*</code>	<code>address</code>	Address to write data to
<code>uint32_t</code>	<code>max_nn</code>	Maximum number of 32-bit words to 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 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 3.11. enum CLS\_METER\_COLOR**

Name	Description
<code>CLS_METER_COLOR_GREEN</code>	Packet is green.
<code>CLS_METER_COLOR_YELLOW</code>	Packet is yellow.
<code>CLS_METER_COLOR_RED</code>	Packet is red.

#### 3.6.1.2 CLS\_METER\_RFC

RFC type to use for CLS Metering operations.

**Table 3.12. enum CLS\_METER\_RFC**

Name	Description
<code>CLS_METER_RFC2698</code>	RFC 2698 defines one method for two rate three color metering.
<code>CLS_METER_RFC2697</code>	RFC 2697 defines one method for two rate three color metering.
<code>CLS_METER_RFC4115</code>	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 3.13. `typedef CLS_METER_COLOR`**

Type	Definition
<code>CLS_METER_COLOR</code>	<code>enum CLS_METER_COLOR</code>

### 3.6.2.2 CLS\_METER\_RFC

RFC type to use for CLS Metering operations.

**Table 3.14. `typedef CLS_METER_RFC`**

Type	Definition
<code>CLS_METER_RFC</code>	<code>enum CLS_METER_RFC</code>

## 3.6.3 Cluster Local Scratch Functions

### 3.6.3.1 cmd\_cls\_read\_be\_ptr32

**Prototype:**

```
void cmd_cls_read_be_ptr32(__xread void* data, volatile void __cls* address, uint32_t
count, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

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

**Table 3.15. `cmd_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>uint32_t</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:**

- cmd\_cls\_read\_ptr32()

### 3.6.3.2 cmd\_cls\_read\_be\_ptr40

**Prototype:**

```
void cmd_cls_read_be_ptr40(__xread void* data, volatile void __addr40 __cls* address,
uint32_t count, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

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

**Table 3.16. cmd\_cls\_read\_be\_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 where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
uint32_t	count	Number of 32-bit words to read
sync_t	sync	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to be raised on completion

**See Also:**

- cmd\_cls\_read\_ptr40()

### 3.6.3.3 cmd\_cls\_read\_le\_ptr32

**Prototype:**

```
void cmd_cls_read_le_ptr32(__xread void* data, volatile void __cls* address, uint32_t
count, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

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

**Table 3.17. cmd\_cls\_read\_le\_ptr32 parameters**

Type	Name	Description
__xread void*	data	Read Transfer Registers the read data will be placed in
volatile void __cls*	address	Address to read from
uint32_t	count	Number of 32-bit words to read
sync_t	sync	Type of synchronization (sig_done or ctx_swap)

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

**See Also:**

- cmd\_cls\_read\_ptr32()

### 3.6.3.4 cmd\_cls\_read\_le\_ptr40

**Prototype:**

```
void cmd_cls_read_le_ptr40(__xread void* data, volatile void __addr40 __cls* address,
uint32_t count, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

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

**Table 3.18. cmd\_cls\_read\_le\_ptr40 parameters**

Type	Name	Description
<u>xread</u> void*	<i>data</i>	Read Transfer Registers the read data will be placed in
volatile void <u>addr40</u> __cls*	<i>address</i>	40 bit address to read from where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
uint32_t	<i>count</i>	Number of 32-bit words to read
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:**

- cmd\_cls\_read\_ptr40()

### 3.6.3.5 cmd\_cls\_read\_ptr32

**Prototype:**

```
void cmd_cls_read_ptr32(__xread void* data, volatile void __cls* address, uint32_t 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 3.19. cmd\_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>uint32_t</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

### 3.6.3.6 cmd\_cls\_read\_ptr40

**Prototype:**

```
void cmd_cls_read_ptr40(__xread void* data, volatile void __addr40 __cls* address, uint32_t
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 3.20. cmd\_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.

Type	Name	Description
uint32_t	<i>count</i>	Number of 32-bit words to read
sync_t	<i>sync</i>	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to be raised on completion

### 3.6.3.7 cmd\_cls\_write\_be\_ptr32

**Prototype:**

```
void cmd_cls_write_be_ptr32(__xwrite void* data, volatile void __addr32 __cls* address,
                            uint32_t count, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

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

**Table 3.21. cmd\_cls\_write\_be\_ptr32 parameters**

Type	Name	Description
__xwrite void*	<i>data</i>	Write Transfer Registers to be written
volatile void __addr32 __cls*	<i>address</i>	Address to write to
uint32_t	<i>count</i>	Number of 32-bit words to 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:**

- [cmd\\_cls\\_write\\_ptr32\(\)](#)

### 3.6.3.8 cmd\_cls\_write\_be\_ptr40

**Prototype:**

```
void cmd_cls_write_be_ptr40(__xwrite void* data, volatile void __addr40 __cls* address,
                            uint32_t count, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

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

**Table 3.22. cmd\_cls\_write\_be\_ptr40 parameters**

Type	Name	Description
__xwrite void*	<i>data</i>	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.
uint32_t	count	Number of 32-bit words to written
sync_t	sync	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to be raised on completion

**See Also:**

- cmd\_cls\_write\_ptr32()

### 3.6.3.9 cmd\_cls\_write\_le\_ptr32

**Prototype:**

```
void cmd_cls_write_le_ptr32(__xwrite void* data, volatile void __cls* address, uint32_t
count, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

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

**Table 3.23. cmd\_cls\_write\_le\_ptr32 parameters**

Type	Name	Description
__xwrite void*	data	Write Transfer Registers to be written
volatile void __cls*	address	Address to write to
uint32_t	count	Number of 32-bit words 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:**

- cmd\_cls\_write\_ptr32()

### 3.6.3.10 cmd\_cls\_write\_le\_ptr40

**Prototype:**

```
void cmd_cls_write_le_ptr40(__xwrite void* data, volatile void __addr40 __cls* address,
uint32_t count, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

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

**Table 3.24. cmd\_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>uint32_t</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:**

- `cmd_cls_write_ptr40()`

### 3.6.3.11 cmd\_cls\_write8\_be\_ind\_ptr40

**Prototype:**

```
void cmd_cls_write8_be_ind_ptr40(__xwrite void* data, volatile void __addr40 __cls*
address, uint32_t 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 3.25. cmd\_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>uint32_t</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 cmd\_cls\_write8\_be\_ind\_ptr32

**Prototype:**

```
void cmd_cls_write8_be_ind_ptr32(__xwrite void* data, volatile void __cls* address,
uint32_t 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 3.26. cmd\_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>uint32_t</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 cmd\_cls\_write8\_le\_ind\_ptr40

**Prototype:**

```
void cmd_cls_write8_le_ind_ptr40(__xwrite void* data, volatile void __addr40 __cls*
address, uint32_t 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 3.27. cmd\_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>uint32_t</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()`

### 3.6.3.14 cmd\_cls\_write8\_le\_ind\_ptr32

**Prototype:**

```
void cmd_cls_write8_le_ind_ptr32(__xwrite void* data, volatile void __cls* address,
uint32_t 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 3.28. cmd\_cls\_write8\_le\_ind\_ptr32 parameters**

Type	Name	Description
__xwrite void*	data	Write Transfer Registers data to write to Cluster Local Scratch
volatile void __cls*	address	32 Bit address to write to
uint32_t	max_nn	Maximum number of bytes to write (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\\_write\\_ind\(\)](#)

### 3.6.3.15 cmd\_cls\_write8\_ind\_ptr40

**Prototype:**

```
void cmd_cls_write8_ind_ptr40(__xwrite void* data, volatile void __addr40 __cls* address,
uint32_t 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 3.29. cmd\_cls\_write8\_ind\_ptr40 parameters**

Type	Name	Description
__xwrite void*	data	Write Transfer Registers holds the data to be written
volatile void __addr40 __cls*	address	40 Bit address to write to
uint32_t	max_nn	Maximum number of bytes to write (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\\_write\\_ind\(\)](#)

### **3.6.3.16 cmd\_cls\_write8\_ind\_ptr32**

**Prototype:**

```
void cmd_cls_write8_ind_ptr32(__xwrite void* data, volatile void __cls* address, uint32_t 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 3.30. cmd\_cls\_write8\_ind\_ptr32 parameters**

Type	Name	Description
__xwrite void*	data	Write Transfer Registers holds the data to be written
volatile void __cls*	address	32 Bit address to write to
uint32_t	max_nn	Maximum number of bytes to write (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\\_write\\_ind\(\)](#)

### **3.6.3.17 cmd\_cls\_write\_ptr32**

**Prototype:**

```
void cmd_cls_write_ptr32(__xwrite void* data, volatile void __cls* address, uint32_t 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 3.31. cmd\_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>uint32_t</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

### 3.6.3.18 cmd\_cls\_write\_ptr40

**Prototype:**

```
void cmd_cls_write_ptr40(__xwrite void* data, volatile void __addr40 __cls* address,
                          uint32_t 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 3.32. cmd\_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>uint32_t</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

### 3.6.3.19 cmd\_cls\_write8\_be\_ptr32

**Prototype:**

```
void cmd_cls_write8_be_ptr32(__xwrite void* data, volatile void __cls* address, uint32_t
                           count, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

Write bytes to Cluster Local Scratch in Big Endian mode.

**Table 3.33. cmd\_cls\_write8\_be\_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>uint32_t</code>	<code>count</code>	Number of bytes 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:**

- `cmd_cls_write8_ptr32()`

### 3.6.3.20 cmd\_cls\_write8\_be\_ptr40

**Prototype:**

```
void cmd_cls_write8_be_ptr40(__xwrite void* data, volatile void __addr40 __cls* address,
                           uint32_t count, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

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

**Table 3.34. cmd\_cls\_write8\_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>uint32_t</code>	<code>count</code>	Number of bytes 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:**

- `cmd_cls_write8_ptr32()`

### 3.6.3.21 cmd\_cls\_write8\_le\_ptr32

**Prototype:**

```
void cmd_cls_write8_le_ptr32(__xwrite void* data, volatile void __cls* address, uint32_t
                           count, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

Write bytes to Cluster Local Scratch in Little Endian mode.

**Table 3.35. cmd\_cls\_write8\_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>uint32_t</code>	<code>count</code>	Number of bytes 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:**

- `cmd_cls_write8_ptr32()`

### 3.6.3.22 cmd\_cls\_write8\_le\_ptr40

**Prototype:**

```
void cmd_cls_write8_le_ptr40(__xwrite void* data, volatile void __addr40 __cls* address,
                           uint32_t count, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

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

**Table 3.36. cmd\_cls\_write8\_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>uint32_t</code>	<code>count</code>	Number of bytes 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:**

- `cmd_cls_write8_ptr32()`

### 3.6.3.23 cmd\_cls\_write8\_ptr32

**Prototype:**

```
void cmd_cls_write8_ptr32(__xwrite void* data, volatile void __cls* address, uint32_t
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 3.37. cmd\_cls\_write8\_ptr32 parameters**

Type	Name	Description
__xwrite void*	data	Write Transfer Registers to be written
volatile void __cls*	address	Address to write to
uint32_t	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 cmd\_cls\_write8\_ptr40

**Prototype:**

```
void cmd_cls_write8_ptr40(__xwrite void* data, volatile void __addr40 __cls* address,
uint32_t 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 3.38. cmd\_cls\_write8\_ptr40 parameters**

Type	Name	Description
__xwrite void*	data	Write Transfer Registers to be written
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.
uint32_t	count	Number of bytes to be written
sync_t	sync	Type of synchronization (sig_done or ctx_swap)

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

### 3.6.3.25 cmd\_cls\_clear\_bits\_ind\_ptr40

**Prototype:**

```
void cmd_cls_clear_bits_ind_ptr40(__xwrite uint32_t* mask, volatile void __addr40 __cls* address, uint32_t 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 3.39. cmd\_cls\_clear\_bits\_ind\_ptr40 parameters**

Type	Name	Description
<code>__xwrite uint32_t*</code>	<i>mask</i>	Mask of bits to clear
<code>volatile void __addr40 __cls*</code>	<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.
<code>uint32_t</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

### 3.6.3.26 cmd\_cls\_clear\_bits\_ind\_ptr32

**Prototype:**

```
void cmd_cls_clear_bits_ind_ptr32(__xwrite uint32_t* mask, volatile void __cls* address, uint32_t 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 3.40. cmd\_cls\_clear\_bits\_ind\_ptr32 parameters**

Type	Name	Description
<code>__xwrite uint32_t*</code>	<code>mask</code>	Mask of bits to clear
<code>volatile void __cls*</code>	<code>address</code>	Address of 32-bit word in which to clear bits
<code>uint32_t</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 (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

### 3.6.3.27 cmd\_cls\_set\_bits\_ind\_ptr40

**Prototype:**

```
void cmd_cls_set_bits_ind_ptr40(__xwrite uint32_t* mask, volatile void __addr40 __cls*
address, uint32_t 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 3.41. cmd\_cls\_set\_bits\_ind\_ptr40 parameters**

Type	Name	Description
<code>__xwrite uint32_t*</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.

Type	Name	Description
uint32_t	<i>max_nn</i>	Number of 32-bit words to set bits from
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

### 3.6.3.28 cmd\_cls\_set\_bits\_ind\_ptr32

**Prototype:**

```
void cmd_cls_set_bits_ind_ptr32(__xwrite uint32_t* mask, volatile void __cls* address,
                                uint32_t 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 3.42. cmd\_cls\_set\_bits\_ind\_ptr32 parameters**

Type	Name	Description
__xwrite uint32_t*	<i>mask</i>	Mask of bits to set
volatile void __cls*	<i>address</i>	Address of 32-bit word in which to set bits
uint32_t	<i>max_nn</i>	Number of 32-bit words to set bits from
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

### 3.6.3.29 cmd\_cls\_clear\_bits\_ptr32

**Prototype:**

```
void cmd_cls_clear_bits_ptr32(__xwrite uint32_t* mask, volatile void __cls* address,
                               uint32_t 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 3.43. cmd\_cls\_clear\_bits\_ptr32 parameters**

Type	Name	Description
<code>__xwrite uint32_t*</code>	<code>mask</code>	Mask of bits to clear
<code>volatile void __cls*</code>	<code>address</code>	Address of 32-bit word in which to clear bits
<code>uint32_t</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

### 3.6.3.30 cmd\_cls\_clear\_bits\_ptr40

**Prototype:**

```
void cmd_cls_clear_bits_ptr40(__xwrite uint32_t* mask, volatile void __addr40 __cls*
address, uint32_t 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 3.44. cmd\_cls\_clear\_bits\_ptr40 parameters**

Type	Name	Description
<code>__xwrite uint32_t*</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.

Type	Name	Description
uint32_t	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

### 3.6.3.31 cmd\_cls\_set\_bits\_ptr32

**Prototype:**

```
void cmd_cls_set_bits_ptr32(__xwrite uint32_t* mask, volatile void __cls* address, uint32_t
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 3.45. cmd\_cls\_set\_bits\_ptr32 parameters**

Type	Name	Description
__xwrite uint32_t*	mask	Mask of bits to set
volatile void __cls*	address	Address of 32-bit word in which to set bits
uint32_t	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

### 3.6.3.32 cmd\_cls\_set\_bits\_ptr40

**Prototype:**

```
void cmd_cls_set_bits_ptr40(__xwrite uint32_t* mask, volatile void __addr40 __cls* address,
uint32_t 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 3.46. cmd\_cls\_set\_bits\_ptr40 parameters**

Type	Name	Description
<code>__xwrite uint32_t*</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>uint32_t</code>	<code>count</code>	Number of 32-bit words to set 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

### 3.6.3.33 cmd\_cls\_set\_bits\_imm\_ptr32

**Prototype:**

```
void cmd_cls_set_bits_imm_ptr32(volatile void __cls* address, uint32_t 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 3.47. cmd\_cls\_set\_bits\_imm\_ptr32 parameters**

Type	Name	Description
<code>volatile void __cls*</code>	<code>address</code>	Address of 32-bit word in which to set bits
<code>uint32_t</code>	<code>value</code>	Mask of bits to set

### 3.6.3.34 cmd\_cls\_set\_bits\_imm\_ptr40

**Prototype:**

```
void cmd_cls_set_bits_imm_ptr40(volatile void __addr40 __cls* address, uint32_t 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 3.48. cmd\_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.
uint32_t	value	Mask of bits to set

### 3.6.3.35 cmd\_cls\_clear\_bits\_imm\_ptr32

**Prototype:**

```
void cmd_cls_clear_bits_imm_ptr32(volatile void __cls* address, uint32_t 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 3.49. cmd\_cls\_clear\_bits\_imm\_ptr32 parameters**

Type	Name	Description
volatile void __cls*	address	Address of 32-bit word in which to clear bits
uint32_t	value	Mask of bits to clear

### 3.6.3.36 cmd\_cls\_clear\_bits\_imm\_ptr40

**Prototype:**

```
void cmd_cls_clear_bits_imm_ptr40(volatile void __addr40 __cls* address, uint32_t 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 3.50. cmd\_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.
uint32_t	value	Mask of bits to clear

### 3.6.3.37 cmd\_cls\_xor\_bits\_ind\_ptr40

**Prototype:**

```
void cmd_cls_xor_bits_ind_ptr40(__xwrite void* mask, volatile void __addr40 __cls* address,  
uint32_t 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 3.51. cmd\_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>uint32_t</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 cmd\_cls\_xor\_bits\_ind\_ptr32

**Prototype:**

```
void cmd_cls_xor_bits_ind_ptr32(__xwrite void* mask, volatile void __cls* address, uint32_t
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 3.52. cmd\_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>uint32_t</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)

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

### 3.6.3.39 cmd\_cls\_xor\_bits\_ptr32

**Prototype:**

```
void cmd_cls_xor_bits_ptr32(__xwrite void* mask, volatile void __cls* address, uint32_t
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 3.53. cmd\_cls\_xor\_bits\_ptr32 parameters**

Type	Name	Description
__xwrite void*	<i>mask</i>	Mask of bits to XOR
volatile void __cls*	<i>address</i>	Address of 32-bit word in which to XOR bits
uint32_t	<i>count</i>	Number of 32-bit words to XOR bits from
sync_t	<i>sync</i>	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to be raised on completion

### 3.6.3.40 cmd\_cls\_xor\_bits\_ptr40

**Prototype:**

```
void cmd_cls_xor_bits_ptr40(__xwrite void* mask, volatile void __addr40 __cls* address,
uint32_t 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 3.54. cmd\_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>uint32_t</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 cmd\_cls\_swap\_ptr32

**Prototype:**

```
void cmd_cls_swap_ptr32(__xwrite void* data, volatile void __cls* address, uint32_t 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 3.55. cmd\_cls\_swap\_ptr32 parameters**

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Data to swap
<code>volatile void __cls*</code>	<code>address</code>	Address of 32-bit word in which to swap bits
<code>uint32_t</code>	<code>count</code>	Number of 32-bit words to swap (maximum is 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

### 3.6.3.42 cmd\_cls\_swap\_ptr40

**Prototype:**

```
void cmd_cls_swap_ptr40(__xwrite void* data, volatile void __addr40 __cls* address,
uint32_t 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 3.56. cmd\_cls\_swap\_ptr40 parameters**

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Data to swap
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit address in which to swap bits where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>uint32_t</code>	<code>count</code>	Number of 32-bit words to swap (maximum is 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

### 3.6.3.43 cmd\_cls\_meter\_ptr32

**Prototype:**

```
void cmd_cls_meter_ptr32(__xread uint32_t* val, __xwrite uint32_t* data, volatile void __cls* address, uint32_t rfc_type, uint32_t color, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

Perform a metering operation in Cluster Local Scratch memory.

Perform a metering operation.

**Table 3.57. cmd\_cls\_meter\_ptr32 parameters**

Type	Name	Description
<code>__xread uint32_t*</code>	<code>val</code>	Return meter command color result i.e. 0 (green), 1 (yellow), 2 (red)
<code>__xwrite uint32_t*</code>	<code>data</code>	Data to write
<code>volatile void __cls*</code>	<code>address</code>	Address to write metering information
<code>uint32_t</code>	<code>rfc_type</code>	0 (RFC2698), 1 (RFC2697/4115). See enum CLS_METER_RFC
<code>uint32_t</code>	<code>color</code>	Color of packet: 0 (green), 1 (yellow), 2 (red). See enum CLS_METER_COLOR
<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

### 3.6.3.44 cmd\_cls\_meter\_ptr40

**Prototype:**

```
void cmd_cls_meter_ptr40(__xread uint32_t* val, __xwrite uint32_t* data, volatile void
__addr40 __cls* address, uint32_t rfc_type, uint32_t color, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

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

Perform a metering operation.

**Table 3.58. cmd\_cls\_meter\_ptr40 parameters**

Type	Name	Description
<code>__xread uint32_t*</code>	<code>val</code>	Return meter command color result i.e. 0 (green), 1 (yellow), 2 (red)
<code>__xwrite uint32_t*</code>	<code>data</code>	Data to write
<code>volatile void __addr40 __cls*</code>	<code>address</code>	Address to write metering information
<code>uint32_t</code>	<code>rfc_type</code>	0 (RFC2698), 1 (RFC2697/4115). See enum CLS_METER_RFC
<code>uint32_t</code>	<code>color</code>	Color of packet: 0 (green), 1 (yellow), 2 (red). See enum CLS_METER_COLOR
<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

### 3.6.3.45 cmd\_cls\_statistic\_ptr32

**Prototype:**

```
void cmd_cls_statistic_ptr32(__xwrite void* data, volatile void __cls* address, uint32_t
count, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

Perform a statistic in Cluster Local Scratch memory.

Add statistic to CLS memory.

**Table 3.59. cmd\_cls\_statistic\_ptr32 parameters**

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Add 32-bit data to address specified
<code>volatile void __cls*</code>	<code>address</code>	Address in CLS in which to add statistic
<code>uint32_t</code>	<code>count</code>	Number of words to add (maximum is 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

### 3.6.3.46 cmd\_cls\_statistic\_ptr40

**Prototype:**

```
void cmd_cls_statistic_ptr40(__xwrite void* data, volatile void __addr40 __cls* address,
uint32_t count, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

Perform a statistic in 40 bit Cluster Local Scratch.

Add statistic to CLS memory.

**Table 3.60. cmd\_cls\_statistic\_ptr40 parameters**

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Add 32-bit data to address specified
<code>volatile void __addr40 __cls*</code>	<code>address</code>	40 bit CLS address in which to perform statistic where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>uint32_t</code>	<code>count</code>	Number of words to add (maximum is 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

### 3.6.3.47 cmd\_cls\_statistic\_imm\_ptr32

**Prototype:**

```
void cmd_cls_statistic_imm_ptr32(volatile void __cls* address, uint32_t 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 3.61. cmd\_cls\_statistic\_imm\_ptr32 parameters**

Type	Name	Description
<code>volatile void __cls*</code>	<code>address</code>	Address of 32-bit word in which to write statistic
<code>uint32_t</code>	<code>value</code>	Value to write

### 3.6.3.48 cmd\_cls\_statistic\_imm\_ptr40

**Prototype:**

```
void cmd_cls_statistic_imm_ptr40(volatile void __addr40 __cls* address, uint32_t 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 3.62. cmd\_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.
uint32_t	value	Value to write

### 3.6.3.49 cmd\_cls\_add\_ptr32

**Prototype:**

```
void cmd_cls_add_ptr32(__xwrite void* data, volatile void __cls* address, uint32_t 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 3.63. cmd\_cls\_add\_ptr32 parameters**

Type	Name	Description
__xwrite void*	data	Value(s) to add
volatile void __cls*	address	Address of 32-bit word to which to add values
uint32_t	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

### 3.6.3.50 cmd\_cls\_add\_ptr40

**Prototype:**

```
void cmd_cls_add_ptr40(__xwrite void* data, volatile void __addr40 __cls* address, uint32_t
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 3.64. cmd\_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>uint32_t</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 cmd\_cls\_test\_and\_add\_ptr32

**Prototype:**

```
void cmd_cls_test_and_add_ptr32(__xread uint32_t* val, __xwrite uint32_t* data, volatile
void __cls* address, uint32_t 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 3.65. cmd\_cls\_test\_and\_add\_ptr32 parameters**

Type	Name	Description
<code>__xread uint32_t*</code>	<code>val</code>	Returned pre-modified value
<code>__xwrite uint32_t*</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>uint32_t</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

### 3.6.3.52 cmd\_cls\_test\_and\_add\_ptr40

**Prototype:**

```
void cmd_cls_test_and_add_ptr40(__xread uint32_t* val, __xwrite uint32_t* data, volatile
void __addr40 __cls* address, uint32_t 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 3.66. cmd\_cls\_test\_and\_add\_ptr40 parameters**

Type	Name	Description
<code>__xread uint32_t*</code>	<code>val</code>	Returned pre-modified value
<code>__xwrite uint32_t*</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>uint32_t</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 cmd\_cls\_add64\_ptr32

**Prototype:**

```
void cmd_cls_add64_ptr32(__xwrite void* data, volatile void __cls* address, uint32_t
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 3.67. cmd\_cls\_add64\_ptr32 parameters**

Type	Name	Description
__xwrite void*	data	Value(s) to add
volatile void __cls*	address	Address of 64-bit word to which to add values
uint32_t	count	Number of 32-bit words to add values to and must be multiple of 2.
sync_t	sync	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to be raised on completion

### 3.6.3.54 cmd\_cls\_add64\_ptr40

**Prototype:**

```
void cmd_cls_add64_ptr40(__xwrite void* data, volatile void __addr40 __cls* address,
uint32_t 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 3.68. cmd\_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>uint32_t</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

### 3.6.3.55 cmd\_cls\_test\_and\_add64\_ptr32

**Prototype:**

```
void cmd_cls_test_and_add64_ptr32(__xread uint32_t* val, __xwrite uint32_t* data, volatile void __cls* address, uint32_t 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 3.69. cmd\_cls\_test\_and\_add64\_ptr32 parameters**

Type	Name	Description
<code>__xread uint32_t*</code>	<code>val</code>	Returned pre-modified value
<code>__xwrite uint32_t*</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>uint32_t</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 cmd\_cls\_test\_and\_add64\_ptr40

**Prototype:**

```
void cmd_cls_test_and_add64_ptr40(__xread uint32_t* val, __xwrite uint32_t* data, volatile void __addr40 __cls* address, uint32_t 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 3.70. cmd\_cls\_test\_and\_add64\_ptr40 parameters**

Type	Name	Description
__xread uint32_t*	val	Returned pre-modified value
__xwrite uint32_t*	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.
uint32_t	count	Number of 32-bit words and must be multiple of 2 up to 16
sync_t	sync	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to be raised on completion

**See Also:**

- [cls\\_add64\(\)](#)

### 3.6.3.57 cmd\_cls\_add\_sat\_ptr32

**Prototype:**

```
void cmd_cls_add_sat_ptr32(__xwrite void* data, volatile void __cls* address, uint32_t 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 3.71. cmd\_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>uint32_t</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 cmd\_cls\_add\_ind\_ptr40

**Prototype:**

```
void cmd_cls_add_ind_ptr40(__xwrite void* data, volatile void __addr40 __cls* address,
                           uint32_t 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 3.72. cmd\_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.

Type	Name	Description
uint32_t	<i>max_nn</i>	Number of 32-bit words to add values to
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 on completion

### 3.6.3.59 cmd\_cls\_add\_ind\_ptr32

**Prototype:**

```
void cmd_cls_add_ind_ptr32(__xwrite void* data, volatile void __cls* address, uint32_t
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 3.73. cmd\_cls\_add\_ind\_ptr32 parameters**

Type	Name	Description
__xwrite void*	<i>data</i>	Value(s) to add
volatile void __cls*	<i>address</i>	Address of 32-bit word to which to add values
uint32_t	<i>max_nn</i>	Number of 32-bit words to add values to
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 on completion

### 3.6.3.60 cmd\_cls\_add64\_ind\_ptr40

**Prototype:**

```
void cmd_cls_add64_ind_ptr40(__xwrite void* data, volatile void __addr40 __cls* address,
uint32_t 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 3.74. cmd\_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>uint32_t</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

### 3.6.3.61 cmd\_cls\_add64\_ind\_ptr32

**Prototype:**

```
void cmd_cls_add64_ind_ptr32(__xwrite void* data, volatile void __cls* address, uint32_t
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 3.75. cmd\_cls\_add64\_ind\_ptr32 parameters**

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Value(s) to add

Type	Name	Description
volatile void __cls*	<i>address</i>	Address of 64-bit word to which to add values
uint32_t	<i>max_nn</i>	Number of 32-bit words to add values to (2,4,6,8,10,12,14 or 16)
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 on completion

### 3.6.3.62 cmd\_cls\_add\_sat\_ind\_ptr40

**Prototype:**

```
void cmd_cls_add_sat_ind_ptr40(__xwrite void* data, volatile void __addr40 __cls* address,
                               uint32_t 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 (*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 3.76. cmd\_cls\_add\_sat\_ind\_ptr40 parameters**

Type	Name	Description
__xwrite void*	<i>data</i>	Value(s) to add
volatile void __addr40 __cls*	<i>address</i>	40 bit address in which to add values where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
uint32_t	<i>max_nn</i>	Number of 32-bit words to add values to
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 on completion

### 3.6.3.63 cmd\_cls\_add\_sat\_ind\_ptr32

**Prototype:**

```
void cmd_cls_add_sat_ind_ptr32(__xwrite void* data, volatile void __cls* address, uint32_t
                               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 3.77. cmd\_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>uint32_t</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 cmd\_cls\_sub\_ind\_ptr40

**Prototype:**

```
void cmd_cls_sub_ind_ptr40(__xwrite void* data, volatile void __addr40 __cls* address,  
                           uint32_t 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 3.78. cmd\_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>uint32_t</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 (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

### 3.6.3.65 cmd\_cls\_sub\_ind\_ptr32

**Prototype:**

```
void cmd_cls_sub_ind_ptr32(__xwrite void* data, volatile void __cls* address, uint32_t
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 3.79. cmd\_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>uint32_t</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 (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

### 3.6.3.66 cmd\_cls\_sub64\_ind\_ptr40

**Prototype:**

```
void cmd_cls_sub64_ind_ptr40(__xwrite void* data, volatile void __addr40 __cls* address,
uint32_t 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 ( $\text{max\_nn} > 1$ ) 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 3.80. cmd\_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>uint32_t</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 (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

### 3.6.3.67 cmd\_cls\_sub64\_ind\_ptr32

**Prototype:**

```
void cmd_cls_sub64_ind_ptr32(__xwrite void* data, volatile void __cls* address, uint32_t
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 ( $\text{max\_nn} > 1$ ) 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 3.81. cmd\_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>uint32_t</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 (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

### 3.6.3.68 cmd\_cls\_sub\_sat\_ind\_ptr40

**Prototype:**

```
void cmd_cls_sub_sat_ind_ptr40(__xwrite void* data, volatile void __addr40 __cls* address,
                                uint32_t 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 3.82. cmd\_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>uint32_t</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 cmd\_cls\_sub\_sat\_ind\_ptr32

**Prototype:**

```
void cmd_cls_sub_sat_ind_ptr32(__xwrite void* data, volatile void __cls* address, uint32_t
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 3.83. cmd\_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>uint32_t</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 ( <code>sig_done</code> or <code>ctx_swap</code> )
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

### 3.6.3.70 cmd\_cls\_add\_sat\_ptr40

**Prototype:**

```
void cmd_cls_add_sat_ptr40(__xwrite void* data, volatile void __addr40 __cls* address,
uint32_t 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 3.84. cmd\_cls\_add\_sat\_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>uint32_t</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

### 3.6.3.71 cmd\_cls\_add\_imm\_ptr32

**Prototype:**

```
void cmd_cls_add_imm_ptr32(volatile void __cls* address, uint32_t 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 3.85. cmd\_cls\_add\_imm\_ptr32 parameters**

Type	Name	Description
<code>volatile void __cls*</code>	<code>address</code>	Address of 32-bit word to which to add values
<code>uint32_t</code>	<code>value</code>	Value to add

### 3.6.3.72 cmd\_cls\_add\_imm\_ptr40

**Prototype:**

```
void cmd_cls_add_imm_ptr40(volatile void __addr40 __cls* address, uint32_t 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 3.86. cmd\_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.
uint32_t	value	Value to add

### 3.6.3.73 cmd\_cls\_add\_imm\_sat\_ptr32

**Prototype:**

```
void cmd_cls_add_imm_sat_ptr32(volatile void __cls* address, uint32_t 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 3.87. cmd\_cls\_add\_imm\_sat\_ptr32 parameters**

Type	Name	Description
volatile void __cls*	address	Address of 32-bit word to which to add values
uint32_t	value	Value to add

### 3.6.3.74 cmd\_cls\_add\_imm\_sat\_ptr40

**Prototype:**

```
void cmd_cls_add_imm_sat_ptr40(volatile void __addr40 __cls* address, uint32_t 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 3.88. cmd\_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.
uint32_t	value	Value to add

### 3.6.3.75 cmd\_cls\_add64\_imm\_ptr32

**Prototype:**

```
void cmd_cls_add64_imm_ptr32(volatile void __cls* address, uint32_t 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 3.89. cmd\_cls\_add64\_imm\_ptr32 parameters**

Type	Name	Description
volatile void __cls*	address	Address of 64-bit word to which to add values
uint32_t	value	Value to add

### 3.6.3.76 cmd\_cls\_add64\_imm\_ptr40

**Prototype:**

```
void cmd_cls_add64_imm_ptr40(volatile void __addr40 __cls* address, uint32_t 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 3.90. cmd\_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.
uint32_t	value	Value to add

### 3.6.3.77 cmd\_cls\_test\_and\_add\_imm\_ptr32

**Prototype:**

```
void cmd_cls_test_and_add_imm_ptr32(__xread uint32_t* val, volatile void __cls* address,
uint32_t value, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

Local scratch memory test and add immediate.

Same as cmd\_cls\_add\_imm\_ptr32 command, but also returns pre-modified memory contents.

**Table 3.91. cmd\_cls\_test\_and\_add\_imm\_ptr32 parameters**

Type	Name	Description
__xread uint32_t*	val	Returned pre-modified value
volatile void __cls*	address	Cluster Local Scratch address to add to
uint32_t	value	Immediate value to add (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:**

- cmd\_cls\_add\_imm\_ptr32()

### 3.6.3.78 cmd\_cls\_test\_and\_add\_imm\_ptr40

**Prototype:**

```
void cmd_cls_test_and_add_imm_ptr40(__xread uint32_t* val, volatile void __addr40 __cls* address, uint32_t value, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

Local scratch memory test and add immediate.

Same as cmd\_cls\_add\_imm\_ptr32 command, but also returns pre-modified memory contents.

**Table 3.92. cmd\_cls\_test\_and\_add\_imm\_ptr40 parameters**

Type	Name	Description
<code>__xread uint32_t*</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>uint32_t</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:**

- `cmd_cls_add_imm_ptr40()`

### 3.6.3.79 cmd\_cls\_test\_and\_add64\_imm\_ptr32

**Prototype:**

```
void cmd_cls_test_and_add64_imm_ptr32(__xread uint32_t* val, volatile void __cls* address, uint32_t 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 3.93. cmd\_cls\_test\_and\_add64\_imm\_ptr32 parameters**

Type	Name	Description
<code>__xread uint32_t*</code>	<code>val</code>	Returned pre-modified value
<code>volatile void __cls*</code>	<code>address</code>	Cluster Local Scratch address to add to
<code>uint32_t</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:**

- `cmd_cls_add64_imm_ptr32()`

### 3.6.3.80 cmd\_cls\_test\_and\_add64\_imm\_ptr40

**Prototype:**

```
void cmd_cls_test_and_add64_imm_ptr40(__xread uint32_t* val, volatile void __addr40 __cls*  
address, uint32_t 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 3.94. cmd\_cls\_test\_and\_add64\_imm\_ptr40 parameters**

Type	Name	Description
<code>__xread uint32_t*</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>uint32_t</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:**

- `cmd_cls_add64_imm_ptr40()`

### 3.6.3.81 cmd\_cls\_sub\_ptr32

**Prototype:**

```
void cmd_cls_sub_ptr32(__xwrite void* data, volatile void __cls* address, uint32_t 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 3.95. cmd\_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>uint32_t</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 cmd\_cls\_sub\_ptr40

**Prototype:**

```
void cmd_cls_sub_ptr40(__xwrite void* data, volatile void __addr40 __cls* address, uint32_t
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 3.96. cmd\_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>uint32_t</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 cmd\_cls\_read\_le\_ind\_ptr40

**Prototype:**

```
void cmd_cls_read_le_ind_ptr40(__xread void* data, volatile void __addr40 __cls* address,
uint32_t 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 3.97. cmd\_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>uint32_t</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 cmd\_cls\_read\_le\_ind\_ptr32

**Prototype:**

```
void cmd_cls_read_le_ind_ptr32(__xread void* data, volatile void __cls* address, uint32_t
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 3.98. cmd\_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>uint32_t</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 cmd\_cls\_read\_be\_ind\_ptr40

**Prototype:**

```
void cmd_cls_read_be_ind_ptr40(__xread void* data, volatile void __addr40 __cls* address,
uint32_t 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 3.99. cmd\_cls\_read\_be\_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
uint32_t	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\(\)](#)

### 3.6.3.86 cmd\_cls\_read\_be\_ind\_ptr32

**Prototype:**

```
void cmd_cls_read_be_ind_ptr32(__xread void* data, volatile void __cls* address, uint32_t
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 3.100. cmd\_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
uint32_t	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 cmd\_cls\_read\_ind\_ptr40

**Prototype:**

```
void cmd_cls_read_ind_ptr40(__xread void* data, volatile void __addr40 __cls* address,
uint32_t 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 3.101. cmd\_cls\_read\_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>uint32_t</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()`

### 3.6.3.88 cmd\_cls\_read\_ind\_ptr32

**Prototype:**

```
void cmd_cls_read_ind_ptr32(__xread void* data, volatile void __cls* address, uint32_t
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 3.102. cmd\_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>uint32_t</code>	<code>max_nn</code>	Maximum number of 32-bit words to read (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\\_read\\_ind\(\)](#)

### 3.6.3.89 cmd\_cls\_write\_le\_ind\_ptr40

**Prototype:**

```
void cmd_cls_write_le_ind_ptr40(_xwrite void* data, volatile void __addr40 __cls* address,
uint32_t 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 3.103. cmd\_cls\_write\_le\_ind\_ptr40 parameters**

Type	Name	Description
_xwrite void*	<i>data</i>	Write Transfer Registers data to write to Cluster Local Scratch
volatile void __addr40 __cls*	<i>address</i>	40 Bit address to write to
uint32_t	<i>max_nn</i>	Maximum number of 32-bit words 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\(\)](#)

### 3.6.3.90 cmd\_cls\_write\_le\_ind\_ptr32

**Prototype:**

```
void cmd_cls_write_le_ind_ptr32(_xwrite void* data, volatile void __cls* address, uint32_t
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 3.104. cmd\_cls\_write\_le\_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>uint32_t</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()`

### 3.6.3.91 cmd\_cls\_write\_be\_ind\_ptr40

**Prototype:**

```
void cmd_cls_write_be_ind_ptr40(__xwrite void* data, volatile void __addr40 __cls* address,
uint32_t 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 3.105. cmd\_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>uint32_t</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 cmd\_cls\_write\_be\_ind\_ptr32

**Prototype:**

```
void cmd_cls_write_be_ind_ptr32(__xwrite void* data, volatile void __cls* address, uint32_t
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 3.106. cmd\_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>uint32_t</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 cmd\_cls\_write\_ind\_ptr40

**Prototype:**

```
void cmd_cls_write_ind_ptr40(__xwrite void* data, volatile void __addr40 __cls* address,
                           uint32_t 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 3.107. cmd\_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>uint32_t</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 cmd\_cls\_write\_ind\_ptr32

**Prototype:**

```
void cmd_cls_write_ind_ptr32(__xwrite void* data, volatile void __cls* address, uint32_t 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 3.108. cmd\_cls\_write\_ind\_ptr32 parameters**

Type	Name	Description
__xwrite void*	data	Write Transfer Registers holds the data to be written
volatile void __cls*	address	32 Bit address to write to
uint32_t	max_nn	Maximum number of 32-bit words to write (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\\_write\\_ind\(\)](#)

### 3.6.3.95 cmd\_cls\_test\_and\_sub\_ptr32

**Prototype:**

```
void cmd_cls_test_and_sub_ptr32(__xread uint32_t* val, __xwrite uint32_t* data, volatile void __cls* address, uint32_t 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 cmd\_cls\_sub\_ptr32 command, but also returns pre-modified memory contents.



#### Note

Alignment is not currently enforced.

**Table 3.109. cmd\_cls\_test\_and\_sub\_ptr32 parameters**

Type	Name	Description
<code>__xread uint32_t*</code>	<code>val</code>	Returned pre-modified value
<code>__xwrite uint32_t*</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>uint32_t</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 (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised on completion

**See Also:**

- `cmd_cls_sub_ptr32()`

### 3.6.3.96 cmd\_cls\_test\_and\_sub\_ptr40

**Prototype:**

```
void cmd_cls_test_and_sub_ptr40(__xread uint32_t* val, __xwrite uint32_t* data, volatile
void __addr40 __cls* address, uint32_t 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 cmd\_cls\_sub\_ptr32 command, but also returns pre-modified memory contents.



#### Note

Alignment is not currently enforced.

**Table 3.110. cmd\_cls\_test\_and\_sub\_ptr40 parameters**

Type	Name	Description
<code>__xread uint32_t*</code>	<code>val</code>	Returned pre-modified value
<code>__xwrite uint32_t*</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>uint32_t</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 (sig_done or ctx_swap)

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

**See Also:**

- cmd\_cls\_sub\_ptr40()

### 3.6.3.97 cmd\_cls\_sub64\_ptr32

**Prototype:**

```
void cmd_cls_sub64_ptr32(__xwrite void* data, volatile void __cls* address, uint32_t
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 3.111. cmd\_cls\_sub64\_ptr32 parameters**

Type	Name	Description
__xwrite void*	<i>data</i>	Value(s) to subtract
volatile void __cls*	<i>address</i>	Address of 64-bit word from which to subtract values
uint32_t	<i>count</i>	Number of 32-bit words to subtract values from (2, 4, .. 16)
sync_t	<i>sync</i>	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to be raised on completion

### 3.6.3.98 cmd\_cls\_sub64\_ptr40

**Prototype:**

```
void cmd_cls_sub64_ptr40(__xwrite void* data, volatile void __addr40 __cls* address,
uint32_t 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 3.112. cmd\_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>uint32_t</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

### 3.6.3.99 cmd\_cls\_test\_and\_sub64\_ptr32

**Prototype:**

```
void cmd_cls_test_and_sub64_ptr32(__xread uint32_t* val, __xwrite uint32_t* data, volatile void __cls* address, uint32_t 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 cmd\_cls\_sub64\_ptr32 command, but also returns pre-modified memory contents.



## Note

Alignment is not currently enforced.

**Table 3.113. cmd\_cls\_test\_and\_sub64\_ptr32 parameters**

Type	Name	Description
<code>__xread uint32_t*</code>	<code>val</code>	Returned pre-modified value
<code>__xwrite uint32_t*</code>	<code>data</code>	Value(s) to add
<code>volatile void __cls*</code>	<code>address</code>	Address of 64-bit word from which to subtract values
<code>uint32_t</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:**

- `cmd_cls_sub64_ptr32()`

### 3.6.3.100 cmd\_cls\_test\_and\_sub64\_ptr40

**Prototype:**

```
void cmd_cls_test_and_sub64_ptr40(__xread uint32_t* val, __xwrite uint32_t* data, volatile
void __addr40 __cls* address, uint32_t 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 cmd\_cls\_sub64\_ptr40 command, but also returns pre-modified memory contents.



## Note

Alignment is not currently enforced.

**Table 3.114. cmd\_cls\_test\_and\_sub64\_ptr40 parameters**

Type	Name	Description
<code>__xread uint32_t*</code>	<code>val</code>	Returned pre-modified value
<code>__xwrite uint32_t*</code>	<code>data</code>	Value(s) to add

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.
uint32_t	count	Number of 32-bit words and must be multiple of 2 up to 16
sync_t	sync	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to be raised on completion

**See Also:**

- cmd\_cls\_sub64\_ptr40()

### 3.6.3.101 cmd\_cls\_sub\_sat\_ptr32

**Prototype:**

```
void cmd_cls_sub_sat_ptr32(__xwrite void* data, volatile void __cls* address, uint32_t
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 3.115. cmd\_cls\_sub\_sat\_ptr32 parameters**

Type	Name	Description
__xwrite void*	data	Value(s) to subtract
volatile void __cls*	address	Address of 32-bit word from which to subtract values
uint32_t	count	Number of 32-bit words to subtract values from
sync_t	sync	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to be raised on completion

### 3.6.3.102 cmd\_cls\_sub\_sat\_ptr40

**Prototype:**

```
void cmd_cls_sub_sat_ptr40(__xwrite void* data, volatile void __addr40 __cls* address,
uint32_t 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 3.116. cmd\_cls\_sub\_sat\_ptr40 parameters**

Type	Name	Description
__xwrite void*	data	Value(s) to subtract
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.
uint32_t	count	Number of 32-bit words to subtract values from
sync_t	sync	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to be raised on completion

### 3.6.3.103 cmd\_cls\_sub\_imm\_ptr32

**Prototype:**

```
void cmd_cls_sub_imm_ptr32(volatile void __cls* address, uint32_t 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 3.117. cmd\_cls\_sub\_imm\_ptr32 parameters**

Type	Name	Description
volatile void __cls*	address	Address of 32-bit word to which to add values
uint32_t	value	Value to subtract

### 3.6.3.104 cmd\_cls\_sub\_imm\_ptr40

**Prototype:**

```
void cmd_cls_sub_imm_ptr40(volatile void __addr40 __cls* address, uint32_t 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 3.118. cmd\_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.
uint32_t	value	Value to subtract

### 3.6.3.105 cmd\_cls\_sub\_imm\_sat\_ptr32

**Prototype:**

```
void cmd_cls_sub_imm_sat_ptr32(volatile void __cls* address, uint32_t 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 3.119. cmd\_cls\_sub\_imm\_sat\_ptr32 parameters**

Type	Name	Description
volatile void __cls*	address	Address of 32-bit word to which to subtract value
uint32_t	value	Value to subtract

### 3.6.3.106 cmd\_cls\_sub\_imm\_sat\_ptr40

**Prototype:**

```
void cmd_cls_sub_imm_sat_ptr40(volatile void __addr40 __cls* address, uint32_t 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 3.120. cmd\_cls\_sub\_imm\_sat\_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.
uint32_t	value	Value to subtract

### 3.6.3.107 cmd\_cls\_sub64\_imm\_ptr32

**Prototype:**

```
void cmd_cls_sub64_imm_ptr32(volatile void __cls* address, uint32_t 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 3.121. cmd\_cls\_sub64\_imm\_ptr32 parameters**

Type	Name	Description
volatile void __cls*	address	Address of 64-bit word to which to add values
uint32_t	value	Value to subtract

### 3.6.3.108 cmd\_cls\_sub64\_imm\_ptr40

**Prototype:**

```
void cmd_cls_sub64_imm_ptr40(volatile void __addr40 __cls* address, uint32_t 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 3.122. cmd\_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.
uint32_t	value	Value to subtract

### 3.6.3.109 cmd\_cls\_test\_and\_sub\_imm\_ptr32

**Prototype:**

```
void cmd_cls_test_and_sub_imm_ptr32(__xread uint32_t* val, volatile void __cls* address,  
uint32_t value, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

Local scratch memory test and subtract immediate.

Same as cmd\_cls\_sub\_imm\_ptr32 command, but also returns pre-modified memory contents.

**Table 3.123. cmd\_cls\_test\_and\_sub\_imm\_ptr32 parameters**

Type	Name	Description
<code>__xread uint32_t*</code>	<code>val</code>	Returned pre-modified value
<code>volatile void __cls*</code>	<code>address</code>	Cluster Local Scratch address to subtract from
<code>uint32_t</code>	<code>value</code>	Immediate value to subtract (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:**

- `cmd_cls_sub_imm_ptr32()`

### 3.6.3.110 cmd\_cls\_test\_and\_sub\_imm\_ptr40

**Prototype:**

```
void cmd_cls_test_and_sub_imm_ptr40(__xread uint32_t* val, volatile void __addr40 __cls* address, uint32_t value, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

40-bit Local scratch memory test and subtract immediate.

Same as `cmd_cls_sub_imm_ptr32` command, but also returns pre-modified memory contents.

**Table 3.124. cmd\_cls\_test\_and\_sub\_imm\_ptr40 parameters**

Type	Name	Description
<code>__xread uint32_t*</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>uint32_t</code>	<code>value</code>	Immediate value to subtract (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:**

- `cmd_cls_sub_imm_ptr40()`

### 3.6.3.111 cmd\_cls\_test\_and\_sub64\_imm\_ptr32

**Prototype:**

```
void cmd_cls_test_and_sub64_imm_ptr32(__xread uint32_t* val, volatile void __cls* address, uint32_t value, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

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

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

**Table 3.125. cmd\_cls\_test\_and\_sub64\_imm\_ptr32 parameters**

Type	Name	Description
<code>__xread uint32_t*</code>	<code>val</code>	Returned pre-modified value
<code>volatile void __cls*</code>	<code>address</code>	Cluster Local Scratch address to subtract from
<code>uint32_t</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:**

- `cmd_cls_sub64_imm_ptr32()`

### 3.6.3.112 cmd\_cls\_test\_and\_sub64\_imm\_ptr40

**Prototype:**

```
void cmd_cls_test_and_sub64_imm_ptr40(__xread uint32_t* val, volatile void __addr40 __cls*
address, uint32_t 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 3.126. cmd\_cls\_test\_and\_sub64\_imm\_ptr40 parameters**

Type	Name	Description
<code>__xread uint32_t*</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>uint32_t</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:**

- `cmd_cls_sub64_imm_ptr32()`

### 3.6.3.113 cmd\_cls\_test\_and\_clear\_bits\_ptr32

**Prototype:**

```
void cmd_cls_test_and_clear_bits_ptr32(__xread uint32_t* val, __xwrite uint32_t* mask,
volatile void __cls* address, uint32_t count, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

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

**Table 3.127. cmd\_cls\_test\_and\_clear\_bits\_ptr32 parameters**

Type	Name	Description
__xread uint32_t*	val	Returned pre-modified value
__xwrite uint32_t*	mask	Mask of bits to clear
volatile void __cls*	address	Cluster Local Scratch address to clear
uint32_t	count	Number of 32-bit words to clear
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

### 3.6.3.114 cmd\_cls\_test\_and\_clear\_bits\_ptr40

**Prototype:**

```
void cmd_cls_test_and_clear_bits_ptr40(__xread uint32_t* val, __xwrite uint32_t* mask,
volatile void __addr40 __cls* address, uint32_t 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 3.128. cmd\_cls\_test\_and\_clear\_bits\_ptr40 parameters**

Type	Name	Description
__xread uint32_t*	val	Returned pre-modified value
__xwrite uint32_t*	mask	Mask of bits to clear
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.
uint32_t	count	Number of 32-bit words to clear
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

### 3.6.3.115 cmd\_cls\_test\_and\_clear\_bits\_ind\_ptr40

**Prototype:**

```
void cmd_cls_test_and_clear_bits_ind_ptr40(__xread uint32_t* val, __xwrite uint32_t* mask, volatile void __addr40 __cls* address, uint32_t 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 3.129. cmd\_cls\_test\_and\_clear\_bits\_ind\_ptr40 parameters**

Type	Name	Description
__xread uint32_t*	val	Returned pre-modified value
__xwrite uint32_t*	mask	Mask of bits to clear
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.
uint32_t	max_nn	Number of 32-bit words to clear
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

### 3.6.3.116 cmd\_cls\_test\_and\_clear\_bits\_ind\_ptr32

**Prototype:**

```
void cmd_cls_test_and_clear_bits_ind_ptr32(__xread uint32_t* val, __xwrite uint32_t* mask, volatile void __cls* address, uint32_t 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 3.130. cmd\_cls\_test\_and\_clear\_bits\_ind\_ptr32 parameters**

Type	Name	Description
__xread uint32_t*	val	Returned pre-modified value
__xwrite uint32_t*	mask	Mask of bits to clear
volatile void __cls*	address	Cluster Local Scratch address to clear
uint32_t	max_nn	Number of 32-bit words to clear
generic_ind_t	ind	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

### 3.6.3.117 cmd\_cls\_test\_and\_set\_bits\_ind\_ptr40

**Prototype:**

```
void cmd_cls_test_and_set_bits_ind_ptr40(__xread uint32_t* val, __xwrite uint32_t* mask,
volatile void __addr40 __cls* address, uint32_t 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 3.131. cmd\_cls\_test\_and\_set\_bits\_ind\_ptr40 parameters**

Type	Name	Description
__xread uint32_t*	<i>val</i>	Returned pre-modified value
__xwrite uint32_t*	<i>mask</i>	Mask of bits to clear
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.
uint32_t	<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

### 3.6.3.118 cmd\_cls\_test\_and\_set\_bits\_ind\_ptr32

**Prototype:**

```
void cmd_cls_test_and_set_bits_ind_ptr32(__xread uint32_t* val, __xwrite uint32_t* mask,
volatile void __cls* address, uint32_t 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 3.132. cmd\_cls\_test\_and\_set\_bits\_ind\_ptr32 parameters**

Type	Name	Description
__xread uint32_t*	<i>val</i>	Returned pre-modified value

Type	Name	Description
<code>__xwrite uint32_t*</code>	<code>mask</code>	Mask of bits to clear
<code>volatile void __cls*</code>	<code>address</code>	Cluster Local Scratch address to clear
<code>uint32_t</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

### 3.6.3.119 cmd\_cls\_test\_and\_clear\_bits\_imm\_ptr32

**Prototype:**

```
void cmd_cls_test_and_clear_bits_imm_ptr32(__xread uint32_t* val, volatile void __cls*
address, uint32_t 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 3.133. cmd\_cls\_test\_and\_clear\_bits\_imm\_ptr32 parameters**

Type	Name	Description
<code>__xread uint32_t*</code>	<code>val</code>	Returned pre-modified value
<code>volatile void __cls*</code>	<code>address</code>	Cluster Local Scratch address to clear
<code>uint32_t</code>	<code>value</code>	Immediate value to clear (1 - 31)
<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

**See Also:**

- `cmd_cls_clear_bits_imm_ptr32()`

### 3.6.3.120 cmd\_cls\_test\_and\_clear\_bits\_imm\_ptr40

**Prototype:**

```
void cmd_cls_test_and_clear_bits_imm_ptr40(__xread uint32_t* val, volatile void __addr40
__cls* address, uint32_t 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 3.134. cmd\_cls\_test\_and\_clear\_bits\_imm\_ptr40 parameters**

Type	Name	Description
<code>__xread uint32_t*</code>	<code>val</code>	Returned pre-modified value
<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>uint32_t</code>	<code>value</code>	Immediate value to clear (1 - 31)
<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

**See Also:**

- `cmd_cls_clear_bits_imm_ptr40()`

### 3.6.3.121 cmd\_cls\_test\_and\_set\_bits\_ptr32

**Prototype:**

```
void cmd_cls_test_and_set_bits_ptr32(__xread uint32_t* val, __xwrite uint32_t* mask,
volatile void __cls* address, uint32_t count, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

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

**Table 3.135. cmd\_cls\_test\_and\_set\_bits\_ptr32 parameters**

Type	Name	Description
<code>__xread uint32_t*</code>	<code>val</code>	Returned pre-modified value
<code>__xwrite uint32_t*</code>	<code>mask</code>	Mask of bits to set
<code>volatile void __cls*</code>	<code>address</code>	Cluster Local Scratch address to set
<code>uint32_t</code>	<code>count</code>	Number of 32-bit words to set
<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

### 3.6.3.122 cmd\_cls\_test\_and\_set\_bits\_ptr40

**Prototype:**

```
void cmd_cls_test_and_set_bits_ptr40(__xread uint32_t* val, __xwrite uint32_t* mask,
volatile void __addr40 __cls* address, uint32_t 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 3.136. cmd\_cls\_test\_and\_set\_bits\_ptr40 parameters**

Type	Name	Description
<code>__xread uint32_t*</code>	<code>val</code>	Returned pre-modified value
<code>__xwrite uint32_t*</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>uint32_t</code>	<code>count</code>	Number of 32-bit words to set
<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

### 3.6.3.123 cmd\_cls\_test\_and\_set\_bits\_imm\_ptr32

**Prototype:**

```
void cmd_cls_test_and_set_bits_imm_ptr32(__xread uint32_t* val, volatile void __cls*
address, uint32_t value, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

Local scratch memory test and set bits immediate.

Same as cmd\_cls\_set\_bits\_imm\_ptr32 command, but also returns pre-modified memory contents.

**Table 3.137. cmd\_cls\_test\_and\_set\_bits\_imm\_ptr32 parameters**

Type	Name	Description
<code>__xread uint32_t*</code>	<code>val</code>	Returned pre-modified value
<code>volatile void __cls*</code>	<code>address</code>	Cluster Local Scratch address to set
<code>uint32_t</code>	<code>value</code>	Immediate value to set (1 - 31)
<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

**See Also:**

- `cmd_cls_set_bits_imm_ptr32()`

### 3.6.3.124 cmd\_cls\_test\_and\_set\_bits\_imm\_ptr40

**Prototype:**

```
void cmd_cls_test_and_set_bits_imm_ptr40(__xread uint32_t* val, volatile void __addr40
__cls* address, uint32_t value, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

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

Same as cmd\_cls\_set\_bits\_imm\_ptr32 command, but also returns pre-modified memory contents.

**Table 3.138. cmd\_cls\_test\_and\_set\_bits\_imm\_ptr40 parameters**

Type	Name	Description
<code>__xread uint32_t*</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>uint32_t</code>	<code>value</code>	Immediate value to set (1 - 31)
<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

**See Also:**

- cmd\_cls\_set\_bits\_imm\_ptr40()

### 3.6.3.125 cmd\_cls\_test\_and\_add\_sat\_ptr32

**Prototype:**

```
void cmd_cls_test_and_add_sat_ptr32(__xread uint32_t* val, __xwrite uint32_t* data,
volatile void __cls* address, uint32_t 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 3.139. cmd\_cls\_test\_and\_add\_sat\_ptr32 parameters**

Type	Name	Description
<code>__xread uint32_t*</code>	<code>val</code>	Returned pre-modified value
<code>__xwrite uint32_t*</code>	<code>data</code>	Data to add
<code>volatile void __cls*</code>	<code>address</code>	Cluster Local Scratch address to add data to
<code>uint32_t</code>	<code>count</code>	Number of 32-bit words to add
<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

### 3.6.3.126 cmd\_cls\_test\_and\_add\_sat\_ptr40

**Prototype:**

```
void cmd_cls_test_and_add_sat_ptr40(__xread uint32_t* val, __xwrite uint32_t* data,
volatile void __addr40 __cls* address, uint32_t 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 3.140. cmd\_cls\_test\_and\_add\_sat\_ptr40 parameters**

Type	Name	Description
<code>__xread uint32_t*</code>	<code>val</code>	Returned pre-modified value
<code>__xwrite uint32_t*</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>uint32_t</code>	<code>count</code>	Number of 32-bit words to add
<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

### 3.6.3.127 cmd\_cls\_test\_and\_add\_sat\_ind\_ptr40

**Prototype:**

```
void cmd_cls_test_and_add_sat_ind_ptr40(__xread uint32_t* val, __xwrite uint32_t* data,
volatile void __addr40 __cls* address, uint32_t 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 3.141. cmd\_cls\_test\_and\_add\_sat\_ind\_ptr40 parameters**

Type	Name	Description
<code>__xread uint32_t*</code>	<code>val</code>	Returned pre-modified value
<code>__xwrite uint32_t*</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>uint32_t</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 cmd\_cls\_test\_and\_add\_sat\_ind\_ptr32

**Prototype:**

```
void cmd_cls_test_and_add_sat_ind_ptr32(__xread uint32_t* val, __xwrite uint32_t* data,
volatile void __cls* address, uint32_t 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 3.142. cmd\_cls\_test\_and\_add\_sat\_ind\_ptr32 parameters**

Type	Name	Description
<code>__xread uint32_t*</code>	<code>val</code>	Returned pre-modified value
<code>__xwrite uint32_t*</code>	<code>data</code>	Data to add
<code>volatile void __cls*</code>	<code>address</code>	Cluster Local Scratch address to clear
<code>uint32_t</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 cmd\_cls\_test\_and\_sub\_sat\_ind\_ptr40

**Prototype:**

```
void cmd_cls_test_and_sub_sat_ind_ptr40(__xread uint32_t* val, __xwrite uint32_t* data,
volatile void __addr40 __cls* address, uint32_t 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 3.143. cmd\_cls\_test\_and\_sub\_sat\_ind\_ptr40 parameters**

Type	Name	Description
<code>__xread uint32_t*</code>	<code>val</code>	Returned pre-modified value
<code>__xwrite uint32_t*</code>	<code>data</code>	Data to subtract
<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>uint32_t</code>	<code>max_nn</code>	Number of 32-bit words to subtract
<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

### 3.6.3.130 cmd\_cls\_test\_and\_sub\_sat\_ind\_ptr32

**Prototype:**

```
void cmd_cls_test_and_sub_sat_ind_ptr32(__xread uint32_t* val, __xwrite uint32_t* data,
volatile void __cls* address, uint32_t 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 3.144. cmd\_cls\_test\_and\_sub\_sat\_ind\_ptr32 parameters**

Type	Name	Description
<code>__xread uint32_t*</code>	<code>val</code>	Returned pre-modified value
<code>__xwrite uint32_t*</code>	<code>data</code>	Data to subtract
<code>volatile void __cls*</code>	<code>address</code>	Cluster Local Scratch address to subtract
<code>uint32_t</code>	<code>max_nn</code>	Number of 32-bit words to subtract
<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

### 3.6.3.131 cmd\_cls\_test\_and\_add\_imm\_sat\_ptr32

**Prototype:**

```
void cmd_cls_test_and_add_imm_sat_ptr32(__xread uint32_t* val, volatile void __cls*
address, uint32_t value, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

Local scratch memory test and add immediate with saturate.

Same as cmd\_cls\_add\_imm\_sat\_ptr32 command, but also returns pre-modified memory contents.

**Table 3.145. cmd\_cls\_test\_and\_add\_imm\_sat\_ptr32 parameters**

Type	Name	Description
<code>__xread uint32_t*</code>	<code>val</code>	Returned pre-modified value
<code>volatile void __cls*</code>	<code>address</code>	Cluster Local Scratch address to add
<code>uint32_t</code>	<code>value</code>	Immediate value to add (1 - 31)
<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

**See Also:**

- cmd\_cls\_add\_imm\_sat\_ptr32()

### 3.6.3.132 cmd\_cls\_test\_and\_add\_imm\_sat\_ptr40

**Prototype:**

```
void cmd_cls_test_and_add_imm_sat_ptr40(__xread uint32_t* val, volatile void __addr40
__cls* address, uint32_t value, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

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

Same as cmd\_cls\_add\_imm\_sat\_ptr32 command, but also returns pre-modified memory contents.

**Table 3.146. cmd\_cls\_test\_and\_add\_imm\_sat\_ptr40 parameters**

Type	Name	Description
<code>__xread uint32_t*</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>uint32_t</code>	<code>value</code>	Immediate value to add (1 - 31)
<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

**See Also:**

- cmd\_cls\_add\_imm\_sat\_ptr40()

### 3.6.3.133 cmd\_cls\_test\_and\_sub\_sat\_ptr32

**Prototype:**

```
void cmd_cls_test_and_sub_sat_ptr32(__xread uint32_t* val, __xwrite uint32_t* data,
volatile void __cls* address, uint32_t 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 3.147. cmd\_cls\_test\_and\_sub\_sat\_ptr32 parameters**

Type	Name	Description
<code>__xread uint32_t*</code>	<code>val</code>	Returned pre-modified value
<code>__xwrite uint32_t*</code>	<code>data</code>	Data to subtract
<code>volatile void __cls*</code>	<code>address</code>	Cluster Local Scratch address to subtract data from

Type	Name	Description
uint32_t	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

### 3.6.3.134 cmd\_cls\_test\_and\_sub\_sat\_ptr40

**Prototype:**

```
void cmd_cls_test_and_sub_sat_ptr40(__xread uint32_t* val, __xwrite uint32_t* data,
volatile void __addr40 __cls* address, uint32_t 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 3.148. cmd\_cls\_test\_and\_sub\_sat\_ptr40 parameters**

Type	Name	Description
__xread uint32_t*	val	Returned pre-modified value
__xwrite uint32_t*	data	Data to subtract
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.
uint32_t	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 cmd\_cls\_test\_and\_sub\_imm\_sat\_ptr32

**Prototype:**

```
void cmd_cls_test_and_sub_imm_sat_ptr32(__xread uint32_t* val, volatile void __cls*
address, uint32_t value, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

Local scratch memory test and sub immediate with saturate.

Same as cmd\_cls\_sub\_imm\_ptr32 command, but also returns pre-modified memory contents.

**Table 3.149. cmd\_cls\_test\_and\_sub\_imm\_sat\_ptr32 parameters**

Type	Name	Description
__xread uint32_t*	val	Returned pre-modified value

Type	Name	Description
volatile void __cls*	address	Cluster Local Scratch address to sub
uint32_t	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:**

- cmd\_cls\_sub\_imm\_ptr32()

### 3.6.3.136 cmd\_cls\_test\_and\_sub\_imm\_sat\_ptr40

**Prototype:**

```
void cmd_cls_test_and_sub_imm_sat_ptr40(__xread uint32_t* val, volatile void __addr40
__cls* address, uint32_t value, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

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

Same as cmd\_cls\_sub\_imm\_ptr32 command, but also returns pre-modified memory contents.

**Table 3.150. cmd\_cls\_test\_and\_sub\_imm\_sat\_ptr40 parameters**

Type	Name	Description
__xread uint32_t*	val	Returned pre-modified value
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.
uint32_t	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:**

- cmd\_cls\_sub\_imm\_ptr40()

### 3.6.3.137 cmd\_cls\_incr\_ptr32

**Prototype:**

```
void cmd_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 3.151. cmd\_cls\_incr\_ptr32 parameters**

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

### 3.6.3.138 cmd\_cls\_incr\_ptr40

**Prototype:**

```
void cmd_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 3.152. cmd\_cls\_incr\_ptr40 parameters**

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

### 3.6.3.139 cmd\_cls\_decr\_ptr32

**Prototype:**

```
void cmd_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 3.153. cmd\_cls\_decr\_ptr32 parameters**

Type	Name	Description
volatile void __addr32 __cls*	address	32-bit Cluster Local Scratch address to decrement

### 3.6.3.140 cmd\_cls\_decr\_ptr40

**Prototype:**

```
void cmd_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 3.154. cmd\_cls\_decr\_ptr40 parameters**

Type	Name	Description
volatile void __addr40 __cls*	address	40-bit Cluster Local Scratch address to decrement

### 3.6.3.141 cmd\_cls\_incr64\_ptr32

**Prototype:**

```
void cmd_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 3.155. cmd\_cls\_incr64\_ptr32 parameters**

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

### 3.6.3.142 cmd\_cls\_incr64\_ptr40

**Prototype:**

```
void cmd_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 3.156. cmd\_cls\_incr64\_ptr40 parameters**

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

### 3.6.3.143 cmd\_cls\_decr64\_ptr32

**Prototype:**

```
void cmd_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 3.157. cmd\_cls\_decr64\_ptr32 parameters**

Type	Name	Description
volatile void __addr32 __cls*	address	32-bit Cluster Local Scratch address to decrement

### 3.6.3.144 cmd\_cls\_decr64\_ptr40

**Prototype:**

```
void cmd_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 3.158. cmd\_cls\_decr64\_ptr40 parameters**

Type	Name	Description
volatile void __addr40 __cls*	address	40-bit Cluster Local Scratch address to decrement

### 3.6.3.145 cmd\_cls\_cmp\_read\_write\_ptr32

**Prototype:**

```
void cmd_cls_cmp_read_write_ptr32(__xrw uint32_t* data, uint32_t mask, volatile void __cls* address, uint32_t 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 3.159. cmd\_cls\_cmp\_read\_write\_ptr32 parameters**

Type	Name	Description
<code>__xrw uint32_t*</code>	<code>data</code>	Data to read/write
<code>uint32_t</code>	<code>mask</code>	Byte mask bits value
<code>volatile void __cls*</code>	<code>address</code>	Cluster Local Scratch address to read/write from
<code>uint32_t</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

### 3.6.3.146 cmd\_cls\_cmp\_read\_write\_ptr40

**Prototype:**

```
void cmd_cls_cmp_read_write_ptr40(__xrw uint32_t* data, uint32_t mask, volatile void
__addr40 __cls* address, uint32_t 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 3.160. cmd\_cls\_cmp\_read\_write\_ptr40 parameters**

Type	Name	Description
<code>__xrw uint32_t*</code>	<code>data</code>	Data to read/write
<code>uint32_t</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>uint32_t</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 cmd\_cls\_cmp\_read\_write\_ind\_ptr40

**Prototype:**

```
void cmd_cls_cmp_read_write_ind_ptr40(__xrw void* data, volatile void __addr40 __cls* address, uint32_t 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 3.161. cmd\_cls\_cmp\_read\_write\_ind\_ptr40 parameters**

Type	Name	Description
__xrw void*	<i>data</i>	Data to read/write
volatile void __addr40 __cls*	<i>address</i>	40 bit address in which to read/write from where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
uint32_t	<i>max_nn</i>	Max number of 4-byte 32-bit words to read/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

### 3.6.3.148 cmd\_cls\_cmp\_read\_write\_ind\_ptr32

**Prototype:**

```
void cmd_cls_cmp_read_write_ind_ptr32(__xrw void* data, volatile void __cls* address, uint32_t 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 3.162. cmd\_cls\_cmp\_read\_write\_ind\_ptr32 parameters**

Type	Name	Description
__xrw void*	<i>data</i>	Data to read/write
volatile void __cls*	<i>address</i>	Cluster Local Scratch address to read/write from
uint32_t	<i>max_nn</i>	Max number of 4-byte 32-bit words to read/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

### 3.6.3.149 cmd\_cls\_queue\_lock\_ind\_ptr40

**Prototype:**

```
void cmd_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 3.163. cmd\_cls\_queue\_lock\_ind\_ptr40 parameters**

Type	Name	Description
volatile void __addr40 __cls*	address	Cluster Local Scratch 40 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

### 3.6.3.150 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 3.164. 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 cmd\_cls\_queue\_lock\_ptr32

**Prototype:**

```
void cmd_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 3.165. cmd\_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 cmd\_cls\_queue\_lock\_ptr40

**Prototype:**

```
void cmd_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 3.166. cmd\_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 cmd\_cls\_queue\_unlock\_ptr32

**Prototype:**

```
void cmd_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 3.167. cmd\_cls\_queue\_unlock\_ptr32 parameters**

Type	Name	Description
volatile void __cls*	address	Cluster Local Scratch address to read into

### 3.6.3.154 cmd\_cls\_queue\_unlock\_ptr40

**Prototype:**

```
void cmd_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 3.168. cmd\_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 cmd\_cls\_hash\_mask\_ptr32

**Prototype:**

```
void cmd_cls_hash_mask_ptr32(__xwrite void* data, volatile void __cls* address, uint32_t 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 3.169. cmd\_cls\_hash\_mask\_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>uint32_t</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

### 3.6.3.156 cmd\_cls\_hash\_mask\_ptr40

**Prototype:**

```
void cmd_cls_hash_mask_ptr40(__xwrite void* data, volatile void __addr40 __cls* address,
                           uint32_t 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 3.170. cmd\_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>uint32_t</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 cmd\_cls\_hash\_mask\_ind\_ptr40

**Prototype:**

```
void cmd_cls_hash_mask_ind_ptr40(__xwrite void* data, volatile void __addr40 __cls* address, uint32_t max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

Local scratch memory Hash in indirect mode.

**Table 3.171. cmd\_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>uint32_t</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 cmd\_cls\_hash\_mask\_ind\_ptr32

**Prototype:**

```
void cmd_cls_hash_mask_ind_ptr32(__xwrite void* data, volatile void __cls* address,
uint32_t max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

Local scratch memory Hash in indirect mode.

**Table 3.172. cmd\_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>uint32_t</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 cmd\_cls\_hash\_mask\_clear\_ptr32

**Prototype:**

```
void cmd_cls_hash_mask_clear_ptr32(__xwrite void* data, volatile void __cls* address,
uint32_t 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 3.173. cmd\_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>uint32_t</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:**

- `cmd_cls_hash_mask_ptr32()`

### 3.6.3.160 cmd\_cls\_hash\_mask\_clear\_ptr40

**Prototype:**

```
void cmd_cls_hash_mask_clear_ptr40(__xwrite void* data, volatile void __addr40 __cls* address, uint32_t 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 3.174. cmd\_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>uint32_t</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:**

- `cmd_cls_hash_mask_ptr32()`

### 3.6.3.161 cmd\_cls\_cam\_lookup\_ind\_ptr40

**Prototype:**

```
void cmd_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 3.175. cmd\_cls\_cam\_lookup\_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:**

- cmd\_cls\_cam\_lookup\_ptr32()

### 3.6.3.162 cmd\_cls\_cam\_lookup\_ind\_ptr32

**Prototype:**

```
void cmd_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 3.176. cmd\_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:**

- cmd\_cls\_cam\_lookup\_ptr32()

### 3.6.3.163 cmd\_cls\_cam\_lookup\_add\_ind\_ptr40

**Prototype:**

```
void cmd_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 3.177. cmd\_cls\_cam\_lookup\_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:**

- cmd\_cls\_cam\_lookup\_ptr32()

### 3.6.3.164 cmd\_cls\_cam\_lookup\_add\_ind\_ptr32

**Prototype:**

```
void cmd_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 3.178. cmd\_cls\_cam\_lookup\_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

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:**

- cmd\_cls\_cam\_lookup\_ptr32()

### 3.6.3.165 cmd\_cls\_cam\_lookup24\_ind\_ptr40

**Prototype:**

```
void cmd_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 3.179. cmd\_cls\_cam\_lookup24\_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:**

- cmd\_cls\_cam\_lookup\_ptr32()

### 3.6.3.166 cmd\_cls\_cam\_lookup24\_ind\_ptr32

**Prototype:**

```
void cmd_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 3.180. cmd\_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:**

- `cmd_cls_cam_lookup_ptr32()`

### 3.6.3.167 cmd\_cls\_cam\_lookup24\_add\_ind\_ptr40

**Prototype:**

```
void cmd_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 3.181. cmd\_cls\_cam\_lookup24\_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:**

- `cmd_cls_cam_lookup_ptr32()`

### 3.6.3.168 cmd\_cls\_cam\_lookup24\_add\_ind\_ptr32

**Prototype:**

```
void cmd_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 3.182. cmd\_cls\_cam\_lookup24\_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:**

- `cmd_cls_cam_lookup_ptr32()`

### 3.6.3.169 cmd\_cls\_cam\_lookup24\_add\_inc\_ind\_ptr40

**Prototype:**

```
void cmd_cls_cam_lookup24_add_inc_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 or increment usage in indirect mode.

**Table 3.183. cmd\_cls\_cam\_lookup24\_add\_inc\_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:**

- `cmd_cls_cam_lookup_ptr32()`

### 3.6.3.170 cmd\_cls\_cam\_lookup24\_add\_inc\_ind\_ptr32

**Prototype:**

```
void cmd_cls_cam_lookup24_add_inc_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 or increment usage in indirect mode.

**Table 3.184. cmd\_cls\_cam\_lookup24\_add\_inc\_ind\_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 32 bit address to lookup
generic_ind_t	ind	Indirect word
sync_t	sync	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to be raised upon completion

**See Also:**

- cmd\_cls\_cam\_lookup\_ptr32()

### 3.6.3.171 cmd\_cls\_cam\_lookup16\_ind\_ptr40

**Prototype:**

```
void cmd_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 3.185. cmd\_cls\_cam\_lookup16\_ind\_ptr40 parameters**

Type	Name	Description
__xread void*	data_out	Data looked up
__xwrite void*	data_in	Data to lookup
volatile void __addr40 __cls*	address	Local scratch CAM 40 bit address to lookup, 40 bit
generic_ind_t	ind	Indirect word
sync_t	sync	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to be raised upon completion

**See Also:**

- cmd\_cls\_cam\_lookup\_ptr32()

### 3.6.3.172 cmd\_cls\_cam\_lookup16\_ind\_ptr32

**Prototype:**

```
void cmd_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 3.186. cmd\_cls\_cam\_lookup16\_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:**

- cmd\_cls\_cam\_lookup\_ptr32()

### 3.6.3.173 cmd\_cls\_cam\_lookup16\_add\_ind\_ptr40

**Prototype:**

```
void cmd_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 3.187. cmd\_cls\_cam\_lookup16\_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

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:**

- cmd\_cls\_cam\_lookup\_ptr32()

### 3.6.3.174 cmd\_cls\_cam\_lookup16\_add\_ind\_ptr32

**Prototype:**

```
void cmd_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 3.188. cmd\_cls\_cam\_lookup16\_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:**

- cmd\_cls\_cam\_lookup\_ptr32()

### 3.6.3.175 cmd\_cls\_cam\_lookup8\_ind\_ptr40

**Prototype:**

```
void cmd_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 3.189. cmd\_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:**

- `cmd_cls_cam_lookup_ptr32()`

### 3.6.3.176 cmd\_cls\_cam\_lookup8\_ind\_ptr32

**Prototype:**

```
void cmd_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 3.190. cmd\_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:**

- `cmd_cls_cam_lookup_ptr32()`

### 3.6.3.177 cmd\_cls\_cam\_lookup8\_add\_ind\_ptr40

**Prototype:**

```
void cmd_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 3.191. cmd\_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:**

- `cmd_cls_cam_lookup_ptr32()`

### 3.6.3.178 cmd\_cls\_cam\_lookup8\_add\_ind\_ptr32

**Prototype:**

```
void cmd_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 3.192. cmd\_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:**

- `cmd_cls_cam_lookup_ptr32()`

### 3.6.3.179 cmd\_cls\_tcam\_lookup\_ind\_ptr40

**Prototype:**

```
void cmd_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 3.193. cmd\_cls\_tcam\_lookup\_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 TCAM 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:**

- cmd\_cls\_cam\_lookup\_ptr32()

### 3.6.3.180 cmd\_cls\_tcam\_lookup\_ind\_ptr32

**Prototype:**

```
void cmd_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 3.194. cmd\_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:**

- cmd\_cls\_cam\_lookup\_ptr32()

### 3.6.3.181 cmd\_cls\_tcam\_lookup24\_ind\_ptr40

**Prototype:**

```
void cmd_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 3.195. cmd\_cls\_tcam\_lookup24\_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 TCAM 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:**

- cmd\_cls\_cam\_lookup\_ptr32()

### 3.6.3.182 cmd\_cls\_tcam\_lookup24\_ind\_ptr32

**Prototype:**

```
void cmd_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 3.196. cmd\_cls\_tcam\_lookup24\_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

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:**

- cmd\_cls\_cam\_lookup\_ptr32()

### 3.6.3.183 cmd\_cls\_tcam\_lookup16\_ind\_ptr40

**Prototype:**

```
void cmd_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 3.197. cmd\_cls\_tcam\_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 TCAM 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:**

- cmd\_cls\_cam\_lookup\_ptr32()

### 3.6.3.184 cmd\_cls\_tcam\_lookup16\_ind\_ptr32

**Prototype:**

```
void cmd_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 3.198. cmd\_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:**

- `cmd_cls_cam_lookup_ptr32()`

### 3.6.3.185 cmd\_cls\_tcam\_lookup8\_ind\_ptr40

**Prototype:**

```
void cmd_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 3.199. cmd\_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:**

- `cmd_cls_cam_lookup_ptr32()`

### 3.6.3.186 cmd\_cls\_tcam\_lookup8\_ind\_ptr32

**Prototype:**

```
void cmd_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 3.200. cmd\_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:**

- `cmd_cls_cam_lookup_ptr32()`

### 3.6.3.187 cmd\_cls\_cam\_lookup\_ptr32

**Prototype:**

```
void cmd_cls_cam_lookup_ptr32(__xread void* data_out, __xwrite void* data_in, volatile void __cls* address, uint32_t 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 3.201. cmd\_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>uint32_t</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 cmd\_cls\_cam\_lookup\_ptr40

**Prototype:**

```
void cmd_cls_cam_lookup_ptr40(__xread void* data_out, __xwrite void* data_in, volatile
void __addr40 __cls* address, uint32_t 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 3.202. cmd\_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.
<code>uint32_t</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

### 3.6.3.189 cmd\_cls\_cam\_lookup\_add\_ptr32

**Prototype:**

```
void cmd_cls_cam_lookup_add_ptr32(__xread void* data_out, __xwrite void* data_in, volatile
void __cls* address, uint32_t 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 3.203. cmd\_cls\_cam\_lookup\_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>uint32_t</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

### 3.6.3.190 cmd\_cls\_cam\_lookup\_add\_ptr40

**Prototype:**

```
void cmd_cls_cam_lookup_add_ptr40(__xread void* data_out, __xwrite void* data_in, volatile
void __addr40 __cls* address, uint32_t 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 3.204. cmd\_cls\_cam\_lookup\_add\_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.
uint32_t	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

### 3.6.3.191 cmd\_cls\_cam\_lookup24\_ptr32

**Prototype:**

```
void cmd_cls_cam_lookup24_ptr32(__xread void* data_out, __xwrite void* data_in, volatile
void __cls* address, uint32_t 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 3.205. cmd\_cls\_cam\_lookup24\_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
uint32_t	count	Size of CAM in number of 64-bit words (1-32)
sync_t	sync	Type of synchronization (sig_done or ctx_swap)

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

### 3.6.3.192 cmd\_cls\_cam\_lookup24\_ptr40

**Prototype:**

```
void cmd_cls_cam_lookup24_ptr40(__xread void* data_out, __xwrite void* data_in, volatile
void __addr40 __cls* address, uint32_t 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 3.206. cmd\_cls\_cam\_lookup24\_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	<i>address</i>	40 bit CAM address to lookup where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
uint32_t	<i>count</i>	Size of CAM in number of 64-bit words (1-32)
sync_t	<i>sync</i>	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to be raised upon completion

### 3.6.3.193 cmd\_cls\_cam\_lookup24\_add\_ptr32

**Prototype:**

```
void cmd_cls_cam_lookup24_add_ptr32(__xread void* data_out, __xwrite void* data_in,
volatile void __cls* address, uint32_t 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 3.207. cmd\_cls\_cam\_lookup24\_add\_ptr32 parameters**

Type	Name	Description
__xread void*	<i>data_out</i>	Data looked up
__xwrite void*	<i>data_in</i>	Data to lookup

Type	Name	Description
volatile void __cls*	address	Local scratch CAM address to lookup
uint32_t	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

### 3.6.3.194 cmd\_cls\_cam\_lookup24\_add\_ptr40

**Prototype:**

```
void cmd_cls_cam_lookup24_add_ptr40(__xread void* data_out, __xwrite void* data_in,
volatile void __addr40 __cls* address, uint32_t 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 3.208. cmd\_cls\_cam\_lookup24\_add\_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.
uint32_t	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

### 3.6.3.195 cmd\_cls\_cam\_lookup24\_add\_inc\_ptr32

**Prototype:**

```
void cmd_cls_cam_lookup24_add_inc_ptr32(__xread void* data_out, __xwrite void* data_in,
volatile void __cls* address, uint32_t 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 3.209. cmd\_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>uint32_t</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 (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised upon completion

### 3.6.3.196 cmd\_cls\_cam\_lookup24\_add\_inc\_ptr40

**Prototype:**

```
void cmd_cls_cam_lookup24_add_inc_ptr40(__xread void* data_out, __xwrite void* data_in,
volatile void __addr40 __cls* address, uint32_t 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 3.210. cmd\_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>	40 bit CAM address to lookup where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
<code>uint32_t</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 (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to be raised upon completion

### 3.6.3.197 cmd\_cls\_cam\_lookup24\_add\_lock\_ptr32

**Prototype:**

```
void cmd_cls_cam_lookup24_add_lock_ptr32(__xread void* data_out, __xwrite void* data_in,
volatile void __cls* address, uint32_t 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 3.211. cmd\_cls\_cam\_lookup24\_add\_lock\_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>uint32_t</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

### 3.6.3.198 cmd\_cls\_cam\_lookup24\_add\_lock\_ptr40

**Prototype:**

```
void cmd_cls_cam_lookup24_add_lock_ptr40(__xread void* data_out, __xwrite void* data_in,
volatile void __addr40 __cls* address, uint32_t 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 3.212. cmd\_cls\_cam\_lookup24\_add\_lock\_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>uint32_t</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)

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

### 3.6.3.199 cmd\_cls\_cam\_lookup24\_add\_extend\_ptr32

**Prototype:**

```
void cmd_cls_cam_lookup24_add_extend_ptr32(__xread void* data_out, __xwrite void* data_in,
volatile void __cls* address, uint32_t 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 3.213. cmd\_cls\_cam\_lookup24\_add\_extend\_ptr32 parameters**

Type	Name	Description
<i>__xread void*</i>	<i>data_out</i>	Data looked up
<i>__xwrite void*</i>	<i>data_in</i>	Data to lookup
volatile void <i>__cls*</i>	<i>address</i>	Local scratch CAM address to lookup
uint32_t	<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

### 3.6.3.200 cmd\_cls\_cam\_lookup24\_add\_extend\_ptr40

**Prototype:**

```
void cmd_cls_cam_lookup24_add_extend_ptr40(__xread void* data_out, __xwrite void* data_in,
volatile void __addr40 __cls* address, uint32_t 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 3.214. cmd\_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>uint32_t</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 cmd\_cls\_cam\_lookup16\_ptr32

**Prototype:**

```
void cmd_cls_cam_lookup16_ptr32(__xread void* data_out, __xwrite void* data_in, volatile void __cls* address, uint32_t 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 3.215. cmd\_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>uint32_t</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 cmd\_cls\_cam\_lookup16\_ptr40

**Prototype:**

```
void cmd_cls_cam_lookup16_ptr40(__xread void* data_out, __xwrite void* data_in, volatile void __addr40 __cls* address, uint32_t 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 3.216. cmd\_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>uint32_t</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

### 3.6.3.203 cmd\_cls\_cam\_lookup16\_add\_ptr32

**Prototype:**

```
void cmd_cls_cam_lookup16_add_ptr32(__xread void* data_out, __xwrite void* data_in,
volatile void __cls* address, uint32_t 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 3.217. cmd\_cls\_cam\_lookup16\_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>uint32_t</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

### 3.6.3.204 cmd\_cls\_cam\_lookup16\_add\_ptr40

**Prototype:**

```
void cmd_cls_cam_lookup16_add_ptr40(__xread void* data_out, __xwrite void* data_in,
volatile void __addr40 __cls* address, uint32_t 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 3.218. cmd\_cls\_cam\_lookup16\_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>uint32_t</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

### 3.6.3.205 cmd\_cls\_cam\_lookup8\_ptr32

**Prototype:**

```
void cmd_cls_cam_lookup8_ptr32(__xread void* data_out, __xwrite void* data_in, volatile void __cls* address, uint32_t 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 3.219. cmd\_cls\_cam\_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 CAM address to lookup
<code>uint32_t</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

### 3.6.3.206 cmd\_cls\_cam\_lookup8\_ptr40

**Prototype:**

```
void cmd_cls_cam_lookup8_ptr40(__xread void* data_out, __xwrite void* data_in, volatile
void __addr40 __cls* address, uint32_t 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 3.220. cmd\_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>uint32_t</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

### 3.6.3.207 cmd\_cls\_cam\_lookup8\_add\_ptr32

**Prototype:**

```
void cmd_cls_cam_lookup8_add_ptr32(__xread void* data_out, __xwrite void* data_in, volatile
void __cls* address, uint32_t 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 3.221. cmd\_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>uint32_t</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

### **3.6.3.208 cmd\_cls\_cam\_lookup8\_add\_ptr40**

**Prototype:**

```
void cmd_cls_cam_lookup8_add_ptr40(__xread void* data_out, __xwrite void* data_in, volatile
void __addr40 __cls* address, uint32_t 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 3.222. cmd\_cls\_cam\_lookup8\_add\_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.
uint32_t	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

### **3.6.3.209 cmd\_cls\_tcam\_lookup\_ptr32**

**Prototype:**

```
void cmd_cls_tcam_lookup_ptr32(__xread void* data_out, __xwrite void* data_in, volatile
void __cls* address, uint32_t 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 3.223. cmd\_cls\_tcam\_lookup\_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 TCAM address to lookup
uint32_t	count	Number of 64-bit words
sync_t	sync	Type of synchronization (sig_done or ctx_swap)

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

### 3.6.3.210 cmd\_cls\_tcam\_lookup\_ptr40

**Prototype:**

```
void cmd_cls_tcam_lookup_ptr40(__xread void* data_out, __xwrite void* data_in, volatile
void __addr40 __cls* address, uint32_t 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 3.224. cmd\_cls\_tcam\_lookup\_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 TCAM address to lookup where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
uint32_t	<i>count</i>	Number of 64-bit words
sync_t	<i>sync</i>	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	<i>sig_ptr</i>	Signal to be raised upon completion

### 3.6.3.211 cmd\_cls\_tcam\_lookup24\_ptr32

**Prototype:**

```
void cmd_cls_tcam_lookup24_ptr32(__xread void* data_out, __xwrite void* data_in, volatile
void __cls* address, uint32_t 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 3.225. cmd\_cls\_tcam\_lookup24\_ptr32 parameters**

Type	Name	Description
__xread void*	<i>data_out</i>	Data looked up
__xwrite void*	<i>data_in</i>	Data to lookup

Type	Name	Description
volatile void __cls*	address	Local scratch TCAM address to lookup
uint32_t	count	Number of 64-bit words
sync_t	sync	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to be raised upon completion

### 3.6.3.212 cmd\_cls\_tcam\_lookup24\_ptr40

**Prototype:**

```
void cmd_cls_tcam_lookup24_ptr40(__xread void* data_out, __xwrite void* data_in, volatile void __addr40 __cls* address, uint32_t 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 3.226. cmd\_cls\_tcam\_lookup24\_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 TCAM address to lookup where upper 6 bits indicate the island. See 6xxx databook for recommended addressing mode.
uint32_t	count	Number of 64-bit words
sync_t	sync	Type of synchronization (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to be raised upon completion

### 3.6.3.213 cmd\_cls\_tcam\_lookup16\_ptr32

**Prototype:**

```
void cmd_cls_tcam_lookup16_ptr32(__xread void* data_out, __xwrite void* data_in, volatile void __cls* address, uint32_t 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 3.227. cmd\_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>uint32_t</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 cmd\_cls\_tcam\_lookup16\_ptr40

**Prototype:**

```
void cmd_cls_tcam_lookup16_ptr40(__xread void* data_out, __xwrite void* data_in, volatile
void __addr40 __cls* address, uint32_t 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 3.228. cmd\_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>uint32_t</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 cmd\_cls\_tcam\_lookup8\_ptr32

**Prototype:**

```
void cmd_cls_tcam_lookup8_ptr32(__xread void* data_out, __xwrite void* data_in, volatile
void __cls* address, uint32_t 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 3.229. cmd\_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>uint32_t</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 cmd\_cls\_tcam\_lookup8\_ptr40

**Prototype:**

```
void cmd_cls_tcam_lookup8_ptr40(__xread void* data_out, __xwrite void* data_in, volatile void __addr40 __cls* address, uint32_t 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 3.230. cmd\_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 __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>uint32_t</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

## 3.7 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 `cmd_cls_state_test()` intrinsic.

**Table 3.231. 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.
<code>cls_state_ring4_status</code>	Indicates status of CLS Ring 4.
<code>cls_state_ring5_status</code>	Indicates status of CLS Ring 5.
<code>cls_state_ring6_status</code>	Indicates status of CLS Ring 6.
<code>cls_state_ring7_status</code>	Indicates status of CLS Ring 7.
<code>cls_state_ring8_status</code>	Indicates status of CLS Ring 8.
<code>cls_state_ring9_status</code>	Indicates status of CLS Ring 9.
<code>cls_state_ring10_status</code>	Indicates status of CLS Ring 10.
<code>cls_state_ring11_status</code>	Indicates status of CLS Ring 11.
<code>cls_state_ring12_status</code>	Indicates status of CLS Ring 12.
<code>cls_state_ring13_status</code>	Indicates status of CLS Ring 13.
<code>cls_state_ring14_status</code>	Indicates status of CLS Ring 14.
<code>cls_state_ring15_status</code>	Indicates status of CLS Ring 15.

### 3.7.1.2 `CLS_RING_SIZE`

Ring word size.

**Table 3.232. enum `CLS_RING_SIZE`**

Name	Description
<code>CLS_RING_SIZE_32</code>	32 words. base address should be aligned to 128 byte boundary, full is >=24 words.
<code>CLS_RING_SIZE_64</code>	64 words. base address should be aligned to 256 byte boundary, full is >=48 words.
<code>CLS_RING_SIZE_128</code>	128 words.

Name	Description
	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 3.233. 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 3.234. 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 3.235. typedef CLS\_RING\_EVENT\_REPORTS**

Type	Definition
CLS_RING_EVENT_REPORTS	enum CLS_RING_EVENT_REPORTS

## 3.7.3 CLS Ring Functions

### 3.7.3.1 cmd\_cls\_state\_test

**Prototype:**

```
int32_t cmd_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 3.236. cmd\_cls\_state\_test parameters**

Type	Name	Description
cls_state_t	state	State to test

### 3.7.3.2 cmd\_cls\_ring\_put\_ptr32

**Prototype:**

```
void cmd_cls_ring_put_ptr32(__xwrite void* data, volatile void __addr32 __cls* address,
uint32_t 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 3.237. cmd\_cls\_ring\_put\_ptr32 parameters**

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Data to put onto ring
<code>volatile void __addr32 __cls*</code>	<code>address</code>	Ring address to put data into where the ring number is specified in address[5:2]
<code>uint32_t</code>	<code>count</code>	Number of 32-bit words to put 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

### 3.7.3.3 cmd\_cls\_ring\_put\_ptr40

**Prototype:**

```
void cmd_cls_ring_put_ptr40(__xwrite void* data, volatile void __addr40 __cls* address,
uint32_t 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 3.238. cmd\_cls\_ring\_put\_ptr40 parameters**

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

Type	Name	Description
uint32_t	<i>count</i>	Number of 32-bit words to put into ring
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

### 3.7.3.4 cmd\_cls\_ring\_journal\_ptr32

**Prototype:**

```
void cmd_cls_ring_journal_ptr32(__xwrite void* data, volatile void __addr32 __cls* address,
                                uint32_t 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 3.239. cmd\_cls\_ring\_journal\_ptr32 parameters**

Type	Name	Description
__xwrite void*	<i>data</i>	Data to add onto ring
volatile void __addr32 __cls*	<i>address</i>	Ring address to add data into where the ring number is specified in address[5:2]
uint32_t	<i>count</i>	Number of 32-bit words to add into ring
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

### 3.7.3.5 cmd\_cls\_ring\_journal\_ptr40

**Prototype:**

```
void cmd_cls_ring_journal_ptr40(__xwrite void* data, volatile void __addr40 __cls* address,
                                uint32_t 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 3.240. cmd\_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>uint32_t</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 cmd\_cls\_ring\_get\_ptr32

**Prototype:**

```
void cmd_cls_ring_get_ptr32(__xread void* data, volatile void __addr32 __cls* address,
                           uint32_t 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 3.241. cmd\_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>uint32_t</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 cmd\_cls\_ring\_get\_ptr40

**Prototype:**

```
void cmd_cls_ring_get_ptr40(__xread void* data, volatile void __addr40 __cls* address,
                           uint32_t 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 3.242. cmd\_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>uint32_t</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 cmd\_cls\_ring\_pop\_ptr32

**Prototype:**

```
void cmd_cls_ring_pop_ptr32(__xread void* data, volatile void __addr32 __cls* address,
                           uint32_t 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 3.243. cmd\_cls\_ring\_pop\_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>uint32_t</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

### 3.7.3.9 cmd\_cls\_ring\_pop\_ptr40

**Prototype:**

```
void cmd_cls_ring_pop_ptr40(__xread void* data, volatile void __addr40 __cls* address,  
uint32_t 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 3.244. cmd\_cls\_ring\_pop\_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.
uint32_t	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

### 3.7.3.10 cmd\_cls\_ring\_get\_safe\_ptr32

**Prototype:**

```
void cmd_cls_ring_get_safe_ptr32(__xread void* data, volatile void __addr32 __cls* address,  
uint32_t 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 3.245. cmd\_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>uint32_t</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 cmd\_cls\_ring\_get\_safe\_ptr40

**Prototype:**

```
void cmd_cls_ring_get_safe_ptr40(__xread void* data, volatile void __addr40 __cls* address,
                                uint32_t 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 3.246. cmd\_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>uint32_t</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 cmd\_cls\_ring\_pop\_safe\_ptr32

**Prototype:**

```
void cmd_cls_ring_pop_safe_ptr32(__xread void* data, volatile void __addr32 __cls* address,
                                 uint32_t 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 3.247. cmd\_cls\_ring\_pop\_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>uint32_t</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

### 3.7.3.13 cmd\_cls\_ring\_pop\_safe\_ptr40

**Prototype:**

```
void cmd_cls_ring_pop_safe_ptr40(__xread void* data, volatile void __addr40 __cls* address,
uint32_t 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 3.248. cmd\_cls\_ring\_pop\_safe\_ptr40 parameters**

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Data read from ring

Type	Name	Description
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.
uint32_t	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 cmd\_cls\_ring\_init\_ptr32

**Prototype:**

```
void cmd_cls_ring_init_ptr32(uint32_t ring_number, volatile void __addr32 __cls*
base_address, enum CLS_RING_SIZE size, uint32_t 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 3.249. cmd\_cls\_ring\_init\_ptr32 parameters**

Type	Name	Description
uint32_t	ring_number	Ring number to initialise (0 to 15)
volatile void __addr32 __cls*	base_address	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	size	Size of ring (0 to 5) of type enum CLS_RING_SIZE.
uint32_t	report_events	List of events of enum CLS_RING_EVENT_REPORTS which should be generated
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

### 3.7.3.15 cmd\_cls\_ring\_init\_ptr40

**Prototype:**

```
void cmd_cls_ring_init_ptr40(uint32_t ring_number, volatile void __addr40 __cls*
base_address, enum CLS_RING_SIZE size, uint32_t 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;
    uint32_t                               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(uint32_t))) uint32_t ring_base[32];

    // initialise ring
    {
        cmd_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 uint32_t write_data[4];

        write_data[0] = 0x11223344;
        write_data[1] = 0xaabbccdd;
        write_data[2] = 0x55667788;
        write_data[3] = 0x9900aabb;
        cmd_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 uint32_t read_data;

        // pop from tail
        cmd_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 uint32_t read_data;

        // get from head
        cmd_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 3.250. cmd\_cls\_ring\_init\_ptr40 parameters**

Type	Name	Description
uint32_t	<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 CLS_RING_SIZE. The top bits of 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.
uint32_t	<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 cmd\_cls\_ring\_write\_ptr32

**Prototype:**

```
void cmd_cls_ring_write_ptr32(__xwrite void* data, uint32_t ring_number, uint32_t count,
                           uint32_t 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 3.251. cmd\_cls\_ring\_write\_ptr32 parameters**

Type	Name	Description
__xwrite void*	<i>data</i>	Data to write to ring
uint32_t	<i>ring_number</i>	Ring number to write data to (0 to 15)
uint32_t	<i>count</i>	Number of 32-bit words to put into ring (1 to 32)
uint32_t	<i>offset</i>	Offset from base for ring to write; will be bounded by the ring size

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

### 3.7.3.17 cmd\_cls\_ring\_write\_ptr40

**Prototype:**

```
void cmd_cls_ring_write_ptr40(__xwrite void* data, uint32_t ring_number, uint32_t count,
uint32_t offset, uint32_t 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 3.252. cmd\_cls\_ring\_write\_ptr40 parameters**

Type	Name	Description
__xwrite void*	<i>data</i>	Data to write to ring
uint32_t	<i>ring_number</i>	Ring number to write data to (0 to 15)
uint32_t	<i>count</i>	Number of 32-bit words to put into ring (1 to 32)
uint32_t	<i>offset</i>	Offset from base for ring to write; will be bounded by the ring size
uint32_t	<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

### 3.7.3.18 cmd\_cls\_ring\_read\_ptr32

**Prototype:**

```
void cmd_cls_ring_read_ptr32(__xread void* data, uint32_t ring_number, uint32_t count,
uint32_t 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 3.253. cmd\_cls\_ring\_read\_ptr32 parameters**

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Data read from ring
<code>uint32_t</code>	<code>ring_number</code>	Ring number to read data from (0 to 15)
<code>uint32_t</code>	<code>count</code>	Number of 32-bit words to read (1 to 16)
<code>uint32_t</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 cmd\_cls\_ring\_read\_ptr40

**Prototype:**

```
void cmd_cls_ring_read_ptr40(__xread void* data, uint32_t ring_number, uint32_t count,
                            uint32_t offset, uint32_t 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 3.254. cmd\_cls\_ring\_read\_ptr40 parameters**

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Data read from ring
<code>uint32_t</code>	<code>ring_number</code>	Ring number to read data from (0 to 15)
<code>uint32_t</code>	<code>count</code>	Number of 32-bit words to read (1 to 16)
<code>uint32_t</code>	<code>offset</code>	Offset from base for ring to read; will be bounded by the ring size
<code>uint32_t</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 cmd\_cls\_ring\_ordered\_lock\_ptr32

**Prototype:**

```
void cmd_cls_ring_ordered_lock_ptr32(uint32_t ring_number, uint32_t 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 3.255. cmd\_cls\_ring\_ordered\_lock\_ptr32 parameters**

Type	Name	Description
uint32_t	<i>ring_number</i>	Ring number to read data from (0 to 15)
uint32_t	<i>sequence_number</i>	Sequence number to reorder on.
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

### 3.7.3.21 cmd\_cls\_ring\_ordered\_lock\_ptr40

**Prototype:**

```
void cmd_cls_ring_ordered_lock_ptr40(uint32_t ring_number, uint32_t sequence_number,
                                     uint32_t 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 3.256. cmd\_cls\_ring\_ordered\_lock\_ptr40 parameters**

Type	Name	Description
uint32_t	<i>ring_number</i>	Ring number to read data from (0 to 15)
uint32_t	<i>sequence_number</i>	Sequence number to reorder on.
uint32_t	<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 cmd\_cls\_ring\_ordered\_unlock\_ptr32

**Prototype:**

```
void cmd_cls_ring_ordered_unlock_ptr32(uint32_t ring_number, uint32_t 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 3.257. cmd\_cls\_ring\_ordered\_unlock\_ptr32 parameters**

Type	Name	Description
uint32_t	<i>ring_number</i>	Ring number to read data from (0 to 15)
uint32_t	<i>sequence_number</i>	Sequence number

**Availability:**

NFP-6xxx Indirect Reference Mode

### 3.7.3.23 cmd\_cls\_ring\_ordered\_unlock\_ptr40

**Prototype:**

```
void cmd_cls_ring_ordered_unlock_ptr40(uint32_t ring_number, uint32_t sequence_number,
uint32_t 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 3.258. cmd\_cls\_ring\_ordered\_unlock\_ptr40 parameters**

Type	Name	Description
uint32_t	<i>ring_number</i>	Ring number to read data from (0 to 15)
uint32_t	<i>sequence_number</i>	Sequence number
uint32_t	<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 cmd\_cls\_ring\_workq\_add\_thread\_ptr32

**Prototype:**

```
void cmd_cls_ring_workq_add_thread_ptr32(__xread void* data, uint32_t ring_number, uint32_t
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 3.259. cmd\_cls\_ring\_workq\_add\_thread\_ptr32 parameters**

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Work received from ring
<code>uint32_t</code>	<code>ring_number</code>	Ring number to read data from (0 to 15)
<code>uint32_t</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 (sig_done or ctx_swap)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

### 3.7.3.25 cmd\_cls\_ring\_workq\_add\_thread\_ptr40

**Prototype:**

```
void cmd_cls_ring_workq_add_thread_ptr40(__xread void* data, uint32_t ring_number, uint32_t count, uint32_t 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 3.260. cmd\_cls\_ring\_workq\_add\_thread\_ptr40 parameters**

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Work received from ring
<code>uint32_t</code>	<code>ring_number</code>	Ring number to read data from (0 to 15)
<code>uint32_t</code>	<code>count</code>	Number of 32-bit words to read (1 to 16)
<code>uint32_t</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

### 3.7.3.26 cmd\_cls\_ring\_workq\_add\_work\_ptr32

**Prototype:**

```
void cmd_cls_ring_workq_add_work_ptr32(__xwrite void* data, uint32_t ring_number, uint32_t 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 3.261. cmd\_cls\_ring\_workq\_add\_work\_ptr32 parameters**

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Work written to ring
<code>uint32_t</code>	<code>ring_number</code>	Ring number to read data from (0 to 15)
<code>uint32_t</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> )
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

### 3.7.3.27 cmd\_cls\_ring\_workq\_add\_work\_ptr40

**Prototype:**

```
void cmd_cls_ring_workq_add_work_ptr40(__xwrite void* data, uint32_t ring_number, uint32_t
count, uint32_t 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 3.262. cmd\_cls\_ring\_workq\_add\_work\_ptr40 parameters**

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Work written to ring
<code>uint32_t</code>	<code>ring_number</code>	Ring number to read data from (0 to 15)
<code>uint32_t</code>	<code>count</code>	Number of 32-bit words to read (1 to 16)
<code>uint32_t</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

### 3.7.3.28 cmd\_cls\_ring\_put\_ind\_ptr40

**Prototype:**

```
void cmd_cls_ring_put_ind_ptr40(__xwrite void* data, volatile void __addr40 __cls* address,
uint32_t 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 3.263. cmd\_cls\_ring\_put\_ind\_ptr40 parameters**

Type	Name	Description
__xwrite void*	data	Data to write to tail of ring
volatile void __addr40 __cls*	address	40 bit Cluster Local Scratch address See 6xxx databook for recommended addressing mode.
uint32_t	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 cmd\_cls\_ring\_put\_ind\_ptr32

**Prototype:**

```
void cmd_cls_ring_put_ind_ptr32(__xwrite void* data, volatile void __cls* address, uint32_t
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 3.264. cmd\_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
uint32_t	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 cmd\_cls\_ring\_journal\_ind\_ptr40

**Prototype:**

```
void cmd_cls_ring_journal_ind_ptr40(__xwrite void* data, volatile void __addr40 __cls*
address, uint32_t 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 3.265. cmd\_cls\_ring\_journal\_ind\_ptr40 parameters**

Type	Name	Description
__xwrite void*	<i>data</i>	Data to write to tail of ring
volatile void __addr40 __cls*	<i>address</i>	40 bit Cluster Local Scratch address See 6xxx databook for recommended addressing mode.
uint32_t	<i>max_nn</i>	Maximum number of 32-bit words to put into ring
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

### 3.7.3.31 cmd\_cls\_ring\_journal\_ind\_ptr32

**Prototype:**

```
void cmd_cls_ring_journal_ind_ptr32(__xwrite void* data, volatile void __cls* address,
uint32_t 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 3.266. cmd\_cls\_ring\_journal\_ind\_ptr32 parameters**

Type	Name	Description
__xwrite void*	<i>data</i>	Data to write to tail of ring
volatile void __cls*	<i>address</i>	32 bit Cluster Local Scratch address
uint32_t	<i>max_nn</i>	Maximum number of 32-bit words to put into ring
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

### 3.7.3.32 cmd\_cls\_ring\_get\_ind\_ptr40

**Prototype:**

```
void cmd_cls_ring_get_ind_ptr40(__xread void* data, volatile void __addr40 __cls* address,
uint32_t 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 3.267. cmd\_cls\_ring\_get\_ind\_ptr40 parameters**

Type	Name	Description
__xread void*	<i>data</i>	Data read from ring
volatile void __addr40 __cls*	<i>address</i>	40 bit address See 6xxx databook for recommended addressing mode.
uint32_t	<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

### 3.7.3.33 cmd\_cls\_ring\_get\_ind\_ptr32

**Prototype:**

```
void cmd_cls_ring_get_ind_ptr32(__xread void* data, volatile void __cls* address, uint32_t
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 3.268. cmd\_cls\_ring\_get\_ind\_ptr32 parameters**

Type	Name	Description
__xread void*	<i>data</i>	Data read from ring
volatile void __cls*	<i>address</i>	Cluster Local Scratch 32 bit address
uint32_t	<i>max_nn</i>	Maximum number of 32-bit words to read
generic_ind_t	<i>ind</i>	Indirect word

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

### 3.7.3.34 cmd\_cls\_ring\_pop\_ind\_ptr40

**Prototype:**

```
void cmd_cls_ring_pop_ind_ptr40(__xread void* data, volatile void __addr40 __cls* address,
uint32_t 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 3.269. cmd\_cls\_ring\_pop\_ind\_ptr40 parameters**

Type	Name	Description
__xread void*	<i>data</i>	Data read from ring
volatile void __addr40 __cls*	<i>address</i>	40 bit address See 6xxx databook for recommended addressing mode.
uint32_t	<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

### 3.7.3.35 cmd\_cls\_ring\_pop\_ind\_ptr32

**Prototype:**

```
void cmd_cls_ring_pop_ind_ptr32(__xread void* data, volatile void __cls* address, uint32_t
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 3.270. cmd\_cls\_ring\_pop\_ind\_ptr32 parameters**

Type	Name	Description
__xread void*	<i>data</i>	Data read from ring
volatile void __cls*	<i>address</i>	Cluster Local Scratch 32 bit address
uint32_t	<i>max_nn</i>	Maximum number of 32-bit words to read
generic_ind_t	<i>ind</i>	Indirect word

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

### 3.7.3.36 cmd\_cls\_ring\_get\_safe\_ind\_ptr40

**Prototype:**

```
void cmd_cls_ring_get_safe_ind_ptr40(__xread void* data, volatile void __addr40 __cls* address, uint32_t 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 3.271. cmd\_cls\_ring\_get\_safe\_ind\_ptr40 parameters**

Type	Name	Description
__xread void*	<i>data</i>	Data read from ring
volatile void __addr40 __cls*	<i>address</i>	40 bit address See 6xxx databook for recommended addressing mode.
uint32_t	<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

### 3.7.3.37 cmd\_cls\_ring\_get\_safe\_ind\_ptr32

**Prototype:**

```
void cmd_cls_ring_get_safe_ind_ptr32(__xread void* data, volatile void __cls* address, uint32_t 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 3.272. cmd\_cls\_ring\_get\_safe\_ind\_ptr32 parameters**

Type	Name	Description
__xread void*	<i>data</i>	Data read from ring
volatile void __cls*	<i>address</i>	Cluster Local Scratch 32 bit address
uint32_t	<i>max_nn</i>	Maximum number of 32-bit words to read

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

### 3.7.3.38 cmd\_cls\_ring\_pop\_safe\_ind\_ptr40

**Prototype:**

```
void cmd_cls_ring_pop_safe_ind_ptr40(__xread void* data, volatile void __addr40 __cls*
address, uint32_t 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 3.273. cmd\_cls\_ring\_pop\_safe\_ind\_ptr40 parameters**

Type	Name	Description
__xread void*	<i>data</i>	Data read from ring
volatile void __addr40 __cls*	<i>address</i>	40 bit address See 6xxx databook for recommended addressing mode.
uint32_t	<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

### 3.7.3.39 cmd\_cls\_ring\_pop\_safe\_ind\_ptr32

**Prototype:**

```
void cmd_cls_ring_pop_safe_ind_ptr32(__xread void* data, volatile void __cls* address,
uint32_t 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 3.274. cmd\_cls\_ring\_pop\_safe\_ind\_ptr32 parameters**

Type	Name	Description
__xread void*	<i>data</i>	Data read from ring
volatile void __cls*	<i>address</i>	32 bit address

Type	Name	Description
uint32_t	<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

## 3.8 Cluster Local Scratch Reflect Intrinsics

This section discusses Cluster Local Scratch Reflect functions.

### 3.8.1 CLS Reflect Functions

#### 3.8.1.1 cmd\_cls\_reflect\_write\_sig\_local

**Prototype:**

```
void cmd_cls_reflect_write_sig_local(__xwrite void* data, uint32_t remote_ME, uint32_t
remote_xfer, uint32_t 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 3.275. cmd\_cls\_reflect\_write\_sig\_local parameters**

Type	Name	Description
<code>__xwrite void*</code>	<i>data</i>	Data to reflect
uint32_t	<i>remote_ME</i>	Remote FPC or ME number
uint32_t	<i>remote_xfer</i>	Remote xfer register reg number
uint32_t	<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

### 3.8.1.2 cmd\_cls\_reflect\_write\_sig\_local\_ptr40

**Prototype:**

```
void cmd_cls_reflect_write_sig_local_ptr40(__xwrite void* data, uint32_t remote_island,  
uint32_t remote_ME, uint32_t remote_xfer, uint32_t 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 3.276. cmd\_cls\_reflect\_write\_sig\_local\_ptr40 parameters**

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Data to reflect
<code>uint32_t</code>	<code>remote_island</code>	Remote island number (0 for local island)
<code>uint32_t</code>	<code>remote_ME</code>	Remote FPC or ME number
<code>uint32_t</code>	<code>remote_xfer</code>	Remote xfer register reg number
<code>uint32_t</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 cmd\_cls\_reflect\_write\_sig\_remote

**Prototype:**

```
void cmd_cls_reflect_write_sig_remote(__xwrite void* data, uint32_t remote_ME, uint32_t  
remote_xfer, int32_t sig, uint32_t 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 3.277. cmd\_cls\_reflect\_write\_sig\_remote parameters**

Type	Name	Description
<code>_xwrite void*</code>	<code>data</code>	Data to reflect
<code>uint32_t</code>	<code>remote_ME</code>	Remote FPC or ME number
<code>uint32_t</code>	<code>remote_xfer</code>	Remote xfer register reg number
<code>int32_t</code>	<code>sig</code>	Triggered signal number
<code>uint32_t</code>	<code>count</code>	Number of 32-bit words to reflect

### 3.8.1.4 cmd\_cls\_reflect\_write\_sig\_remote\_ptr40

**Prototype:**

```
void cmd_cls_reflect_write_sig_remote_ptr40(_xwrite void* data, uint32_t remote_island,
uint32_t remote_ME, uint32_t remote_xfer, int32_t sig, uint32_t 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 3.278. cmd\_cls\_reflect\_write\_sig\_remote\_ptr40 parameters**

Type	Name	Description
<code>_xwrite void*</code>	<code>data</code>	Data to reflect
<code>uint32_t</code>	<code>remote_island</code>	Remote island number (0 for local island)
<code>uint32_t</code>	<code>remote_ME</code>	Remote FPC or ME number
<code>uint32_t</code>	<code>remote_xfer</code>	Remote xfer register reg number
<code>int32_t</code>	<code>sig</code>	Triggered signal number
<code>uint32_t</code>	<code>count</code>	Number of 32-bit words to reflect

### 3.8.1.5 cmd\_cls\_reflect\_write\_sig\_both

**Prototype:**

```
void cmd_cls_reflect_write_sig_both(__xwrite void* data, uint32_t remote_ME, uint32_t
remote_xfer, uint32_t 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 3.279. cmd\_cls\_reflect\_write\_sig\_both parameters**

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Data to reflect
<code>uint32_t</code>	<code>remote_ME</code>	Remote FPC or ME number
<code>uint32_t</code>	<code>remote_xfer</code>	Remote transfer register
<code>uint32_t</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 cmd\_cls\_reflect\_write\_sig\_both\_ptr40

**Prototype:**

```
void cmd_cls_reflect_write_sig_both_ptr40(__xwrite void* data, uint32_t remote_island,
uint32_t remote_ME, uint32_t remote_xfer, uint32_t 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 3.280. cmd\_cls\_reflect\_write\_sig\_both\_ptr40 parameters**

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

Type	Name	Description
uint32_t	<i>remote_island</i>	Remote island number (0 for local island)
uint32_t	<i>remote_ME</i>	Remote FPC or ME number
uint32_t	<i>remote_xfer</i>	Remote transfer register
uint32_t	<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

### 3.8.1.7 cmd\_cls\_reflect\_read\_sig\_remote

**Prototype:**

```
void cmd_cls_reflect_read_sig_remote(__xread void* data, uint32_t remote_ME, uint32_t
remote_xfer, int32_t sig, uint32_t 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 3.281. cmd\_cls\_reflect\_read\_sig\_remote parameters**

Type	Name	Description
<code>__xread void*</code>	<i>data</i>	Data from reflect
uint32_t	<i>remote_ME</i>	Remote FPC or ME number
uint32_t	<i>remote_xfer</i>	Remote xfer register reg number
int32_t	<i>sig</i>	Triggered signal number
uint32_t	<i>count</i>	Number of 32-bit words to reflect

### 3.8.1.8 cmd\_cls\_reflect\_read\_sig\_remote\_ptr40

**Prototype:**

```
void cmd_cls_reflect_read_sig_remote_ptr40(__xread void* data, uint32_t remote_island,
uint32_t remote_ME, uint32_t remote_xfer, int32_t sig, uint32_t 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 3.282. cmd\_cls\_reflect\_read\_sig\_remote\_ptr40 parameters**

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Data from reflect
<code>uint32_t</code>	<code>remote_island</code>	Remote island number (0 for local island)
<code>uint32_t</code>	<code>remote_ME</code>	Remote FPC or ME number
<code>uint32_t</code>	<code>remote_xfer</code>	Remote xfer register reg number
<code>int32_t</code>	<code>sig</code>	Triggered signal number
<code>uint32_t</code>	<code>count</code>	Number of 32-bit words to reflect

### 3.8.1.9 cmd\_cls\_reflect\_read\_sig\_local

**Prototype:**

```
void cmd_cls_reflect_read_sig_local(__xread void* data, uint32_t remote_ME, uint32_t
remote_xfer, uint32_t 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 3.283. cmd\_cls\_reflect\_read\_sig\_local parameters**

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Data from reflect
<code>uint32_t</code>	<code>remote_ME</code>	Remote FPC or ME number
<code>uint32_t</code>	<code>remote_xfer</code>	Remote xfer register reg number
<code>uint32_t</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

### 3.8.1.10 cmd\_cls\_reflect\_read\_sig\_local\_ptr40

**Prototype:**

```
void cmd_cls_reflect_read_sig_local_ptr40(__xread void* data, uint32_t remote_island,  
uint32_t remote_ME, uint32_t remote_xfer, uint32_t 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 3.284. cmd\_cls\_reflect\_read\_sig\_local\_ptr40 parameters**

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Data from reflect
<code>uint32_t</code>	<code>remote_island</code>	Remote island number (0 for local island)
<code>uint32_t</code>	<code>remote_ME</code>	Remote FPC or ME number
<code>uint32_t</code>	<code>remote_xfer</code>	Remote xfer register reg number
<code>uint32_t</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 cmd\_cls\_reflect\_read\_sig\_both

**Prototype:**

```
void cmd_cls_reflect_read_sig_both(__xread void* data, uint32_t remote_ME, uint32_t  
remote_xfer, uint32_t 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 3.285. cmd\_cls\_reflect\_read\_sig\_both parameters**

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Data from reflect
<code>uint32_t</code>	<code>remote_ME</code>	Remote FPC or ME number
<code>uint32_t</code>	<code>remote_xfer</code>	Remote xfer register number
<code>uint32_t</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

### 3.8.1.12 cmd\_cls\_reflect\_read\_sig\_both\_ptr40

**Prototype:**

```
void cmd_cls_reflect_read_sig_both_ptr40(__xread void* data, uint32_t remote_island,
uint32_t remote_ME, uint32_t remote_xfer, uint32_t 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 3.286. cmd\_cls\_reflect\_read\_sig\_both\_ptr40 parameters**

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Data from reflect
<code>uint32_t</code>	<code>remote_island</code>	Remote island number (0 for local island)
<code>uint32_t</code>	<code>remote_ME</code>	Remote FPC or ME number
<code>uint32_t</code>	<code>remote_xfer</code>	Remote xfer register number
<code>uint32_t</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

### 3.8.1.13 cmd\_cls\_reflect\_write\_sig\_local\_ind\_ptr40

**Prototype:**

```
void cmd_cls_reflect_write_sig_local_ind_ptr40(__xwrite void* data, uint32_t remote_island,  
uint32_t remote_ME, uint32_t remote_xfer_reg_number, uint32_t 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 3.287. cmd\_cls\_reflect\_write\_sig\_local\_ind\_ptr40 parameters**

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Data to reflect
<code>uint32_t</code>	<code>remote_island</code>	Remote island number (0 for local island)
<code>uint32_t</code>	<code>remote_ME</code>	Remote FPC or ME number
<code>uint32_t</code>	<code>remote_xfer_reg_number</code>	Remote xfer register reg number
<code>uint32_t</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 cmd\_cls\_reflect\_write\_sig\_local\_ind

**Prototype:**

```
void cmd_cls_reflect_write_sig_local_ind(__xwrite void* data, uint32_t remote_ME, uint32_t  
remote_xfer_reg_number, uint32_t 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 3.288. cmd\_cls\_reflect\_write\_sig\_local\_ind parameters**

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Data to reflect
<code>uint32_t</code>	<code>remote_ME</code>	Remote FPC or ME number
<code>uint32_t</code>	<code>remote_xfer_reg_number</code>	Remote xfer register reg number
<code>uint32_t</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

### 3.8.1.15 cmd\_cls\_reflect\_write\_sig\_remote\_ind\_ptr40

**Prototype:**

```
void cmd_cls_reflect_write_sig_remote_ind_ptr40(__xwrite void* data, uint32_t
remote_island, uint32_t remote_ME, uint32_t remote_xfer_reg_number, int32_t sig, uint32_t
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 3.289. cmd\_cls\_reflect\_write\_sig\_remote\_ind\_ptr40 parameters**

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Data to reflect
<code>uint32_t</code>	<code>remote_island</code>	Remote island number (0 for local island)
<code>uint32_t</code>	<code>remote_ME</code>	Remote FPC or ME number
<code>uint32_t</code>	<code>remote_xfer_reg_number</code>	Remote xfer register reg number
<code>int32_t</code>	<code>sig</code>	Triggered signal number
<code>uint32_t</code>	<code>max_nn</code>	Max number of 32-bit words to reflect
<code>generic_ind_t</code>	<code>ind</code>	Indirect word

### 3.8.1.16 cmd\_cls\_reflect\_write\_sig\_remote\_ind

**Prototype:**

```
void cmd_cls_reflect_write_sig_remote_ind(__xwrite void* data, uint32_t remote_ME, uint32_t
remote_xfer_reg_number, int32_t sig, uint32_t 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 3.290. cmd\_cls\_reflect\_write\_sig\_remote\_ind parameters**

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Data to reflect
<code>uint32_t</code>	<code>remote_ME</code>	Remote FPC or ME number
<code>uint32_t</code>	<code>remote_xfer_reg_number</code>	Remote xfer register reg number
<code>int32_t</code>	<code>sig</code>	Triggered signal number
<code>uint32_t</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 cmd\_cls\_reflect\_write\_sig\_both\_ind\_ptr40

**Prototype:**

```
void cmd_cls_reflect_write_sig_both_ind_ptr40(__xwrite void* data, uint32_t remote_island,
uint32_t remote_ME, uint32_t remote_xfer_reg_number, uint32_t 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 3.291. cmd\_cls\_reflect\_write\_sig\_both\_ind\_ptr40 parameters**

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Data to reflect
<code>uint32_t</code>	<code>remote_island</code>	Remote island number (0 for local island)
<code>uint32_t</code>	<code>remote_ME</code>	Remote FPC or ME number
<code>uint32_t</code>	<code>remote_xfer_reg_number</code>	Remote xfer register reg number
<code>uint32_t</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

### 3.8.1.18 cmd\_cls\_reflect\_write\_sig\_both\_ind

#### Prototype:

```
void cmd_cls_reflect_write_sig_both_ind(__xwrite void* data, uint32_t remote_ME, uint32_t
remote_xfer_reg_number, uint32_t 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 3.292. cmd\_cls\_reflect\_write\_sig\_both\_ind parameters**

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Data to reflect
<code>uint32_t</code>	<code>remote_ME</code>	Remote FPC or ME number
<code>uint32_t</code>	<code>remote_xfer_reg_number</code>	Remote xfer register reg number
<code>uint32_t</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

### 3.8.1.19 cmd\_cls\_reflect\_read\_sig\_remote\_ind\_ptr40

**Prototype:**

```
void cmd_cls_reflect_read_sig_remote_ind_ptr40(__xread void* data, uint32_t remote_island,
uint32_t remote_ME, uint32_t remote_xfer_reg_number, int32_t sig, uint32_t 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 3.293. cmd\_cls\_reflect\_read\_sig\_remote\_ind\_ptr40 parameters**

Type	Name	Description
<code>__xread void*</code>	<i>data</i>	Data from reflect
<code>uint32_t</code>	<i>remote_island</i>	Remote island number (0 for local island)
<code>uint32_t</code>	<i>remote_ME</i>	Remote FPC or ME number
<code>uint32_t</code>	<i>remote_xfer_reg_number</i>	Remote xfer register reg number
<code>int32_t</code>	<i>sig</i>	Triggered signal number
<code>uint32_t</code>	<i>max_nn</i>	Max number of 32-bit words to reflect
<code>generic_ind_t</code>	<i>ind</i>	Indirect word

### 3.8.1.20 cmd\_cls\_reflect\_read\_sig\_remote\_ind

**Prototype:**

```
void cmd_cls_reflect_read_sig_remote_ind(__xread void* data, uint32_t remote_ME, uint32_t
remote_xfer_reg_number, int32_t sig, uint32_t 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 3.294. cmd\_cls\_reflect\_read\_sig\_remote\_ind parameters**

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Data from reflect
<code>uint32_t</code>	<code>remote_ME</code>	Remote FPC or ME number
<code>uint32_t</code>	<code>remote_xfer_reg_number</code>	Remote xfer register reg number
<code>int32_t</code>	<code>sig</code>	Triggered signal number
<code>uint32_t</code>	<code>max_nn</code>	Max number of 32-bit words to reflect
<code>generic_ind_t</code>	<code>ind</code>	Indirect word

### 3.8.1.21 cmd\_cls\_reflect\_read\_sig\_local\_ind\_ptr40

**Prototype:**

```
void cmd_cls_reflect_read_sig_local_ind_ptr40(__xread void* data, uint32_t remote_island,
uint32_t remote_ME, uint32_t remote_xfer_reg_number, uint32_t 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 3.295. cmd\_cls\_reflect\_read\_sig\_local\_ind\_ptr40 parameters**

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Data from reflect
<code>uint32_t</code>	<code>remote_island</code>	Remote island number (0 for local island)
<code>uint32_t</code>	<code>remote_ME</code>	Remote FPC or ME number
<code>uint32_t</code>	<code>remote_xfer_reg_number</code>	Remote xfer register reg number
<code>uint32_t</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

### 3.8.1.22 cmd\_cls\_reflect\_read\_sig\_local\_ind

**Prototype:**

```
void cmd_cls_reflect_read_sig_local_ind(__xread void* data, uint32_t remote_ME, uint32_t
remote_xfer_reg_number, uint32_t 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 3.296. cmd\_cls\_reflect\_read\_sig\_local\_ind parameters**

Type	Name	Description
__xread void*	<i>data</i>	Data from reflect
uint32_t	<i>remote_ME</i>	Remote FPC or ME number
uint32_t	<i>remote_xfer_reg_number</i>	Remote xfer register reg number
uint32_t	<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

### 3.8.1.23 cmd\_cls\_reflect\_read\_sig\_both\_ind\_ptr40

**Prototype:**

```
void cmd_cls_reflect_read_sig_both_ind_ptr40(__xread void* data, uint32_t remote_island,
uint32_t remote_ME, uint32_t remote_xfer_reg_number, uint32_t 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 3.297. cmd\_cls\_reflect\_read\_sig\_both\_ind\_ptr40 parameters**

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Data from reflect
<code>uint32_t</code>	<code>remote_island</code>	Remote island number (0 for local island)
<code>uint32_t</code>	<code>remote_ME</code>	Remote FPC or ME number
<code>uint32_t</code>	<code>remote_xfer_reg_number</code>	Remote xfer register reg number
<code>uint32_t</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

### 3.8.1.24 cmd\_cls\_reflect\_read\_sig\_both\_ind

**Prototype:**

```
void cmd_cls_reflect_read_sig_both_ind(__xread void* data, uint32_t remote_ME, uint32_t
remote_xfer_reg_number, uint32_t 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 3.298. cmd\_cls\_reflect\_read\_sig\_both\_ind parameters**

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Data from reflect
<code>uint32_t</code>	<code>remote_ME</code>	Remote FPC or ME number
<code>uint32_t</code>	<code>remote_xfer_reg_number</code>	Remote xfer register reg number

Type	Name	Description
uint32_t	<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 3.299. 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 3.300. 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 3.301. enum CT\_RING\_SIZE**

Name	Description
CT_RING_SIZE_128	128 words.

Name	Description
	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.
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 3.302. 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 3.303. union cluster\_target\_next\_neighbour\_write\_address\_format\_t**

Type	Name	Description
uint32_t	reserved_3:2	Reserved.
uint32_t	remote_island:6	Island id of remote master.
uint32_t	reserved_2:3	Reserved.
uint32_t	master:4	Master within specified island.  See NFP-6xxx Pull IDs in 6xxx databook.
uint32_t	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.
uint32_t	NN_register_number:7	Next neighbour register number.
uint32_t	reserved_1:2	Reserved.
uint32_t	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 3.304. union cluster\_target\_reflect\_address\_format\_t**

Type	Name	Description
uint32_t	reserved_4:2	Reserved.
uint32_t	remote_island:6	Island Id of remote master.
uint32_t	reserved_3:7	Reserved.
CLUSTER_TARGET_REGISTER_TYPE	register_type:1	Register type where 0 = transfer registers, 1 = CSR registers.
uint32_t	reserved_2:2	Reserved.
uint32_t	master:4	Master within specified island.  See NFP-6xxx Pull IDs in 6xxx databook.
uint32_t	address:8	Transfer Register Address or Local ME CSR Address, depending on XferCsrRegSel.
uint32_t	reserved_1:2	Reserved.
uint32_t	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 3.305. union cluster\_target\_signal\_me\_address\_format\_t**

Type	Name	Description
uint32_t	reserved_3:2	Reserved.
uint32_t	remote_island:6	Island id of remote master.
uint32_t	reserved_2:11	Reserved.
uint32_t	remote_master:4	Remote master id within specified island.  See NFP-6xxx Pull IDs in 6xxx databook.
uint32_t	remote_context:3	Context number (thread number) of remote thread.
uint32_t	signal_number:4	Signal number or signal reference.
uint32_t	reserved_1:2	Reserved.
uint32_t	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 3.306. union cluster\_target\_xpb\_address\_format\_t**

Type	Name	Description
uint32_t	reserved_2:1	Reserved.
uint32_t	global_xpb:1	Global steering bit.  Local = 0, global = 1.
uint32_t	target_island:6	Target island.  0 if own island (local).
uint32_t	slave:2	Slave island number, for those islands that have slave XPB buses.
uint32_t	device:6	XPB device number with the island/slave island.
uint32_t	address:14	XPB register address within the XPB target.
uint32_t	reserved_1:2	Reserved.
uint32_t	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 3.307. `typedef CLUSTER_TARGET_REGISTER_TYPE`**

Type	Definition
<code>CLUSTER_TARGET_REGISTER_TYPE</code>	<code>enum CLUSTER_TARGET_REGISTER_TYPE</code>

### 3.9.3.2 `CLUSTER_TARGET_ADDRESS_MODE`

Enum for addressing mode.

**Table 3.308. `typedef CLUSTER_TARGET_ADDRESS_MODE`**

Type	Definition
<code>CLUSTER_TARGET_ADDRESS_MODE</code>	<code>enum CLUSTER_TARGET_ADDRESS_MODE</code>

### 3.9.3.3 `cluster_target_xpb_address_format_t`

Layout of 32-bit address used with XPB commands.

**Table 3.309. `typedef cluster_target_xpb_address_format_t`**

Type	Definition
<code>cluster_target_xpb_address_format_t</code>	<code>union cluster_target_xpb_address_format_t</code>

### 3.9.3.4 `cluster_target_reflect_address_format_t`

Layout of 32-bit address used with reflector commands.

**Table 3.310. `typedef cluster_target_reflect_address_format_t`**

Type	Definition
<code>cluster_target_reflect_address_format_t</code>	<code>union cluster_target_reflect_address_format_t</code>

### 3.9.3.5 `cluster_target_signal_me_address_format_t`

Layout of 32-bit address used with interthread signalling commands.

**Table 3.311. `typedef cluster_target_signal_me_address_format_t`**

Type	Definition
<code>cluster_target_signal_me_address_format_t</code>	<code>union cluster_target_signal_me_address_format_t</code>

### 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 3.312. 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 3.313. 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 3.314. typedef CT\_RING\_STATUS**

Type	Definition
CT_RING_STATUS	enum CT_RING_STATUS

## 3.9.4 Cluster Target Functions

### 3.9.4.1 cmd\_cluster\_target\_xpb\_write

**Prototype:**

```
void cmd_cluster_target_xpb_write(__xwrite void* xfer, volatile
cluster_target_xpb_address_format_t* address, uint32_t 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 3.315. cmd\_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>uint32_t</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 cmd\_cluster\_target\_xpb\_read

**Prototype:**

```
void cmd_cluster_target_xpb_read(__xread void* xfer, volatile
cluster_target_xpb_address_format_t* address, uint32_t 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 3.316. cmd\_cluster\_target\_xpb\_read parameters**

Type	Name	Description
<code>__xread void*</code>	<code>xfer</code>	Transfer registers containing data read.
<code>volatile cluster_target_xpb_address_format_t*</code>	<code>address</code>	32 bit pointer.
<code>uint32_t</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.

### 3.9.4.3 cmd\_cluster\_target\_reflect\_write\_sig\_none

**Prototype:**

```
void cmd_cluster_target_reflect_write_sig_none(__xwrite void* xfer, volatile
cluster_target_reflect_address_format_t* address, uint32_t 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 3.317. cmd\_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>uint32_t</code>	<code>count</code>	Length in 32-bit words to write (valid values 1 - 16).

### 3.9.4.4 cmd\_cluster\_target\_reflect\_read\_sig\_none

**Prototype:**

```
void cmd_cluster_target_reflect_read_sig_none(__xread void* xfer, volatile
cluster_target_reflect_address_format_t* address, uint32_t 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 3.318. cmd\_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>uint32_t</code>	<code>count</code>	Length in 32-bit words to write (valid values 1 - 16).

### 3.9.4.5 cmd\_cluster\_target\_reflect\_write\_sig\_both

**Prototype:**

```
void cmd_cluster_target_reflect_write_sig_both(__xwrite void* xfer, volatile
cluster_target_reflect_address_format_t* address, uint32_t 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 3.319. cmd\_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>uint32_t</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 cmd\_cluster\_target\_reflect\_read\_sig\_both

**Prototype:**

```
void cmd_cluster_target_reflect_read_sig_both(__xread void* xfer, volatile
cluster_target_reflect_address_format_t* address, uint32_t 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 3.320. cmd\_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>uint32_t</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 cmd\_cluster\_target\_reflect\_write\_sig\_init

**Prototype:**

```
void cmd_cluster_target_reflect_write_sig_init(__xwrite void* xfer, volatile
cluster_target_reflect_address_format_t* address, uint32_t 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 3.321. cmd\_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.

Type	Name	Description
uint32_t	<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 or ctx_swap).
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion.

### 3.9.4.8 cmd\_cluster\_target\_reflect\_read\_sig\_init

**Prototype:**

```
void cmd_cluster_target_reflect_read_sig_init(__xread void* xfer, volatile
cluster_target_reflect_address_format_t* address, uint32_t 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 3.322. cmd\_cluster\_target\_reflect\_read\_sig\_init parameters**

Type	Name	Description
<code>__xread void*</code>	<i>xfer</i>	Transfer registers containing data to read.
<code>volatile cluster_target_reflect_address_format_t*</code>	<i>address</i>	32 bit pointer.
uint32_t	<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 or ctx_swap).
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion.

### 3.9.4.9 cmd\_cluster\_target\_reflect\_write\_sig\_remote

**Prototype:**

```
void cmd_cluster_target_reflect_write_sig_remote(__xwrite void* xfer, volatile
cluster_target_reflect_address_format_t* address, uint32_t 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 3.323. cmd\_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>uint32_t</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 cmd\_cluster\_target\_reflect\_read\_sig\_remote

**Prototype:**

```
void cmd_cluster_target_reflect_read_sig_remote(__xread void* xfer, volatile
cluster_target_reflect_address_format_t* address, uint32_t 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 3.324. cmd\_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.
<code>uint32_t</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.

### 3.9.4.11 cmd\_cluster\_target\_sig\_me\_ctx

**Prototype:**

```
void cmd_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
    cmd_cluster_target_sig_me_ctx(&signal_command);
}
```

**Table 3.325. cmd\_cluster\_target\_sig\_me\_ctx parameters**

Type	Name	Description
volatile cluster_target_signal_me_address_format_t*	address	32 bit pointer.

### 3.9.4.12 cmd\_cluster\_target\_next\_neighbour\_write

**Prototype:**

```
void cmd_cluster_target_next_neighbour_write(__xwrite void* xfer, volatile  
cluster_target_next_neighbour_write_address_format_t* address, uint32_t 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 3.326. cmd\_cluster\_target\_next\_neighbour\_write 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.
uint32_t	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.

### 3.9.4.13 cmd\_cluster\_target\_next\_neighbour\_write\_ind

**Prototype:**

```
void cmd_cluster_target_next_neighbour_write_ind(__xwrite void* xfer, volatile  
cluster_target_next_neighbour_write_address_format_t* address, uint32_t 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;
uint32_t      master = 4;
uint32_t      signal_master = 4;
SIGNAL         sig;
uint32_t      count = 16;
generic_ind_t  ind;
uint32_t      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);

cmd_cluster_target_next_neighbour_write_ind
(
    (void *)&xfer_write[0],
    &command,
    16,
    ind,
    ctx_swap,
    &sig
);
```

**Table 3.327. cmd\_cluster\_target\_next\_neighbour\_write\_ind parameters**

Type	Name	Description
<code>__xwrite void*</code>	<code>xfcr</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>uint32_t</code>	<code>max_nn</code>	Maximum number of 32-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.

**Availability:**

NFP-6xxx Indirect Reference Mode

### 3.9.4.14 cmd\_cluster\_target\_ring\_put

**Prototype:**

```
void cmd_cluster_target_ring_put(__xwrite void* data, uint32_t island_number, uint32_t
ring_number, uint32_t 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 3.328. cmd\_cluster\_target\_ring\_put parameters**

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Data to put onto ring.
<code>uint32_t</code>	<code>island_number</code>	Island number of the target CTM.
<code>uint32_t</code>	<code>ring_number</code>	Ring number to initialise (0 to 15).

Type	Name	Description
uint32_t	<i>count</i>	Number of 32-bit words to put into ring (valid values 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.

### 3.9.4.15 cmd\_cluster\_target\_ring\_get

**Prototype:**

```
void cmd_cluster_target_ring_get(__xread void* data, uint32_t island_number, uint32_t
ring_number, uint32_t 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 3.329. cmd\_cluster\_target\_ring\_get parameters**

Type	Name	Description
__xread void*	<i>data</i>	Data read from ring.
uint32_t	<i>island_number</i>	Island number of the target CTM.
uint32_t	<i>ring_number</i>	Ring number to initialise (0 to 15).
uint32_t	<i>count</i>	Number of 32-bit words to read (valid values 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

### 3.9.4.16 cmd\_cluster\_target\_ring\_init\_ptr32

**Prototype:**

```
void cmd_cluster_target_ring_init_ptr32(uint32_t 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(int32_t)) uint32_t ring_base[128];

uint32_t          ring_index = 2;
uint32_t          island = _island();
uint32_t          count = 3;
SIGNAL           sig;

cmd_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 uint32_t wr_data[3];

    wr_data[0] = 0x12345678;
    wr_data[1] = 0x87654321;
    wr_data[2] = 0x10101010;

    cmd_cluster_target_ring_put((void *)&wr_data[0], island, ring_index, count, ctx_swap, &sig);
}

// get data from the ring
{
    __xread uint32_t rd_data[3];

    cmd_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 3.330. cmd\_cluster\_target\_ring\_init\_ptr32 parameters**

Type	Name	Description
uint32_t	<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 cmd\_cluster\_target\_ring\_init\_ptr40

**Prototype:**

```
void cmd_cluster_target_ring_init_ptr40(uint32_t 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];

uint32_t          ring_index = 2;
uint32_t          island = 33;
uint32_t          count = 3;
SIGNAL           sig;

cmd_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 uint32_t wr_data[3];

    wr_data[0] = 0x12345678;
    wr_data[1] = 0x87654321;
    wr_data[2] = 0x10101010;

    cmd_cluster_target_ring_put((void *)&wr_data[0], island, ring_index, count, ctx_swap, &sig);
}

// get data from the ring
{
    __xread  uint32_t rd_data[3];

    cmd_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 3.331. cmd\_cluster\_target\_ring\_init\_ptr40 parameters**

Type	Name	Description
uint32_t	<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 cmd\_cluster\_target\_ring\_init

**Prototype:**

```
void cmd_cluster_target_ring_init(uint32_t island_number, uint32_t ring_number, uint32_t
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 3.332. cmd\_cluster\_target\_ring\_init parameters**

Type	Name	Description
uint32_t	<i>island_number</i>	Island number of the target CTM.
uint32_t	<i>ring_number</i>	Ring number to initialise (0 to 15)
uint32_t	<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 3.333. enum local\_csr\_t**

Name	Description
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.
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_ctrl	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.

Name	Description
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.
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.

Name	Description
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.
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 3.334. 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.

Type	Name	Description
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\_Access\_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 3.335. 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\_iscsi\_le

**Prototype:**

```
uint32_t crc_iscsi_le(uint32_t data, bytes_specifier_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 3.336. crc\_iscsi\_le parameters**

Type	Name	Description
uint32_t	<i>data</i>	Data on which to perform the 32-bit iSCSI CRC computation
bytes_specifier_t	<i>bspec</i>	Specified bytes in the data argument on which to perform the computation

**See Also:**

- `crc_read()` and `crc_write()`

### 3.11.2.2 crc\_iscsi\_be

**Prototype:**

```
uint32_t crc_iscsi_be(uint32_t data, bytes_specifier_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 3.337. crc\_iscsi\_be parameters**

Type	Name	Description
uint32_t	<i>data</i>	Data on which to perform the 32-bit iSCSI CRC computation
bytes_specifier_t	<i>bspec</i>	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_le`

**Prototype:**

```
uint32_t crc_10_le(uint32_t 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 3.338. `crc_10_le` parameters**

Type	Name	Description
uint32_t	<i>data</i>	Data on which to perform the CRC 10 computation
bytesSpecifier_t	<i>bspec</i>	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_be`

**Prototype:**

```
uint32_t crc_10_be(uint32_t 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 3.339. `crc_10_be` parameters**

Type	Name	Description
uint32_t	<i>data</i>	Data on which to perform the CRC 10 computation
bytesSpecifier_t	<i>bspec</i>	Specified bytes in the data argument on which to perform the computation

**See Also:**

- `crc_read()` and `crc_write()`

### 3.11.2.5 `crc_5_le`

**Prototype:**

```
uint32_t crc_5_le(uint32_t 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 3.340. `crc_5_le` parameters**

Type	Name	Description
uint32_t	<code>data</code>	Data on which to perform the CRC 5 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()`

### 3.11.2.6 `crc_5_be`

**Prototype:**

```
uint32_t crc_5_be(uint32_t 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 3.341. `crc_5_be` parameters**

Type	Name	Description
uint32_t	<code>data</code>	Data on which to perform the CRC 5 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()`

### **3.11.2.7 `crc_iscsi_le_bit_swap`**

**Prototype:**

```
uint32_t crc_iscsi_le_bit_swap(uint32_t 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 3.342. `crc_iscsi_le_bit_swap` parameters**

Type	Name	Description
uint32_t	<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()`

### **3.11.2.8 `crc_iscsi_be_bit_swap`**

**Prototype:**

```
uint32_t crc_iscsi_be_bit_swap(uint32_t 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 3.343. `crc_iscsi_be_bit_swap` parameters**

Type	Name	Description
uint32_t	<code>data</code>	Data on which to perform the 32-bit iSCSI CRC computation

Type	Name	Description
bytesSpecifier_t	<i>bspec</i>	Specified bytes in the data argument on which to perform the computation

**See Also:**

- `crc_read()` and `crc_write()`

### 3.11.2.9 crc\_10\_le\_bit\_swap

**Prototype:**

```
uint32_t crc_10_le_bit_swap(uint32_t 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 3.344. crc\_10\_le\_bit\_swap parameters**

Type	Name	Description
uint32_t	<i>data</i>	Data on which to perform the CRC 10 computation
bytesSpecifier_t	<i>bspec</i>	Specified bytes in the data argument on which to perform the computation

**See Also:**

- `crc_read()` and `crc_write()`

### 3.11.2.10 crc\_10\_be\_bit\_swap

**Prototype:**

```
uint32_t crc_10_be_bit_swap(uint32_t 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 3.345. crc\_10\_be\_bit\_swap parameters**

Type	Name	Description
uint32_t	<i>data</i>	Data on which to perform the CRC 10 computation
bytesSpecifier_t	<i>bspec</i>	Specified bytes in the data argument on which to perform the computation

**See Also:**

- `crc_read()` and `crc_write()`

### 3.11.2.11 crc\_5\_le\_bit\_swap

**Prototype:**

```
uint32_t crc_5_le_bit_swap(uint32_t 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 3.346. crc\_5\_le\_bit\_swap parameters**

Type	Name	Description
uint32_t	<i>data</i>	Data on which to perform the CRC 5 computation
bytesSpecifier_t	<i>bspec</i>	Specified bytes in the data argument on which to perform the computation

**See Also:**

- `crc_read()` and `crc_write()`

### 3.11.2.12 crc\_5\_be\_bit\_swap

**Prototype:**

```
uint32_t crc_5_be_bit_swap(uint32_t 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 3.347. crc\_5\_be\_bit\_swap parameters**

Type	Name	Description
uint32_t	<i>data</i>	Data on which to perform the CRC 5 computation
bytesSpecifier_t	<i>bspec</i>	Specified bytes in the data argument on which to perform the computation

**See Also:**

- `crc_read()` and `crc_write()`

### 3.11.2.13 pop\_count

**Prototype:**

```
uint32_t pop_count(uint32_t data)
```

**Description:**

Pop the number of "1" bits.

This function returns the number of "1" bits in the given value "data".

**Table 3.348. pop\_count parameters**

Type	Name	Description
uint32_t	<i>data</i>	Value on which to perform the operation

### 3.11.2.14 rotr

**Prototype:**

```
uint32_t rotr(uint32_t value, int32_t shift)
```

**Description:**

Rotate right a 32-bit word by specified number of bits.

**Table 3.349. rotr parameters**

Type	Name	Description
uint32_t	<i>value</i>	The 32-bit value to rotate to the right
int32_t	<i>shift</i>	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.

### 3.11.2.15 rotl

**Prototype:**

```
uint32_t rotl(uint32_t value, int32_t shift)
```

**Description:**

Rotate left a 32-bit word by specified number of bits.

**Table 3.350. rotl parameters**

Type	Name	Description
uint32_t	value	The 32-bit value to rotate to the left
int32_t	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.

### 3.11.2.16 crc\_16\_le

**Prototype:**

```
uint32_t crc_16_le(uint32_t data, bytes_specifier_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 3.351. crc\_16\_le parameters**

Type	Name	Description
uint32_t	<i>data</i>	Data on which to perform the 16-bit CCITT CRC computation
bytes_specifier_t	<i>bspec</i>	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

**Prototype:**

```
uint32_t crc_16_be(uint32_t 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 3.352. `crc_16_be` parameters**

Type	Name	Description
uint32_t	<code>data</code>	Data on which to perform the 16-bit CCITT 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()`

### 3.11.2.18 `crc_32_le`

**Prototype:**

```
uint32_t crc_32_le(uint32_t 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 3.353. `crc_32_le` parameters**

Type	Name	Description
uint32_t	<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()`

### 3.11.2.19 `crc_32_be`

**Prototype:**

```
uint32_t crc_32_be(uint32_t 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 3.354. `crc_32_be` parameters**

Type	Name	Description
uint32_t	<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()`

### 3.11.2.20 `crc_ccitt_le`

**Prototype:**

```
uint32_t crc_ccitt_le(uint32_t 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 3.355. `crc_ccitt_le` parameters**

Type	Name	Description
uint32_t	<code>data</code>	Data on which to perform the 16-bit CCITT 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()`

### 3.11.2.21 `crc_ccitt_be`

**Prototype:**

```
uint32_t crc_ccitt_be(uint32_t 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 3.356. `crc_ccitt_be` parameters**

Type	Name	Description
uint32_t	<i>data</i>	Data on which to perform the 16-bit CCITT CRC computation
bytesSpecifier_t	<i>bspec</i>	Specified bytes in the data argument on which to perform the computation

**See Also:**

- `crc_read()` and `crc_write()`

### 3.11.2.22 `crc_16_le_bit_swap`

**Prototype:**

```
uint32_t crc_16_le_bit_swap(uint32_t 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 3.357. `crc_16_le_bit_swap` parameters**

Type	Name	Description
uint32_t	<i>data</i>	Data on which to perform the 16-bit CCITT CRC computation
bytesSpecifier_t	<i>bspec</i>	Specified bytes in the data argument on which to perform the computation

**See Also:**

- `crc_read()` and `crc_write()`

### 3.11.2.23 `crc_16_be_bit_swap`

**Prototype:**

```
uint32_t crc_16_be_bit_swap(uint32_t 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 3.358. `crc_16_be_bit_swap` parameters**

Type	Name	Description
uint32_t	<code>data</code>	Data on which to perform the 16-bit CCITT 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_32_le_bit_swap`

**Prototype:**

```
uint32_t crc_32_le_bit_swap(uint32_t 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 3.359. `crc_32_le_bit_swap` parameters**

Type	Name	Description
uint32_t	<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\_32\_be\_bit\_swap

**Prototype:**

```
uint32_t crc_32_be_bit_swap(uint32_t 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 3.360. crc\_32\_be\_bit\_swap parameters**

Type	Name	Description
uint32_t	<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()`

### 3.11.2.26 crc\_ccitt\_le\_bit\_swap

**Prototype:**

```
uint32_t crc_ccitt_le_bit_swap(uint32_t 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 3.361. crc\_ccitt\_le\_bit\_swap parameters**

Type	Name	Description
uint32_t	<code>data</code>	Data on which to perform the 16-bit CCITT 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()`

### 3.11.2.27 `crc_ccitt_be_bit_swap`

#### Prototype:

```
uint32_t crc_ccitt_be_bit_swap(uint32_t data, bytes_specifier_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 3.362. `crc_ccitt_be_bit_swap` parameters**

Type	Name	Description
uint32_t	<code>data</code>	Data on which to perform the 16-bit CCITT CRC computation
bytes_specifier_t	<code>bspec</code>	Specified bytes in the data argument on which to perform the computation

#### See Also:

- `crc_read()` and `crc_write()`

### 3.11.2.28 `crc_read`

#### Prototype:

```
uint32_t crc_read(void)
```

#### Description:

Read CRC remainder accumulated so far.

### 3.11.2.29 `crc_write`

#### Prototype:

```
void crc_write(uint32_t residue)
```

#### Description:

Write the CRC remainder.

**Table 3.363. crc\_write parameters**

Type	Name	Description
uint32_t	<i>residue</i>	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 cmd\_crypto\_read

**Prototype:**

```
void cmd_crypto_read(__xread void* data, uint32_t address, uint32_t count, sync_t sync,
                     SIGNAL* sig_ptr)
```

**Description:**

Read data from Crypto into transfer register.

**Table 3.364. cmd\_crypto\_read parameters**

Type	Name	Description
__xread void*	<i>data</i>	Xfer register(s) to store data read from memory
uint32_t	<i>address</i>	32 bit address in Crypto SRAM to read data from
uint32_t	<i>count</i>	Number of 64-bit words to read (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

#### 3.12.1.2 cmd\_crypto\_read\_ind

**Prototype:**

```
void cmd_crypto_read_ind(__xread void* data, uint32_t address, uint32_t max_nn,
                        generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

Read data from Crypto into DRAM in indirect mode.

**Table 3.365. cmd\_crypto\_read\_ind parameters**

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Xfer register(s) to store data read from memory
<code>uint32_t</code>	<code>address</code>	32 bit address in Crypto SRAM to read data from
<code>uint32_t</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 cmd\_crypto\_write

**Prototype:**

```
void cmd_crypto_write(__xwrite void* data, uint32_t address, uint32_t count, sync_t sync,
                      SIGNAL* sig_ptr)
```

**Description:**

Write data to Crypto from MEM.

**Table 3.366. cmd\_crypto\_write parameters**

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Xfer register(s) containing data to write to memory
<code>uint32_t</code>	<code>address</code>	32 bit address in Crypto SRAM to write data to
<code>uint32_t</code>	<code>count</code>	Number of 64-bit words to write (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

### 3.12.1.4 cmd\_crypto\_write\_ind

**Prototype:**

```
void cmd_crypto_write_ind(__xwrite void* data, uint32_t address, uint32_t max_nn,
                         generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

Write data to Crypto from DRAM in indirect mode.

**Table 3.367. cmd\_crypto\_write\_ind parameters**

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Xfer register(s) containing data to write to memory

Type	Name	Description
uint32_t	address	32 bit address in Crypto SRAM to write data to
uint32_t	max_nn	Maximum number of 64-bit words to write (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

### 3.12.1.5 cmd\_crypto\_write\_fifo

**Prototype:**

```
void cmd_crypto_write_fifo(__xwrite void* data, uint32_t address, uint32_t count, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

Write data to Crypto FIFO.

**Table 3.368. cmd\_crypto\_write\_fifo parameters**

Type	Name	Description
__xwrite void*	data	Xfer register(s) containing data (command) to write to FIFO
uint32_t	address	32 bit address in Crypto SRAM to write data to
uint32_t	count	Number of 64-bit words to write (1 - 16)
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

### 3.12.1.6 cmd\_crypto\_write\_fifo\_ind

**Prototype:**

```
void cmd_crypto_write_fifo_ind(__xwrite void* data, uint32_t address, uint32_t max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

Write data to Crypto FIFO from DRAM in indirect mode.

**Table 3.369. cmd\_crypto\_write\_fifo\_ind parameters**

Type	Name	Description
__xwrite void*	data	Xfer register(s) containing data (command) to write to FIFO
uint32_t	address	32 bit address in Crypto SRAM to write data to
uint32_t	max_nn	Maximum number of 64-bit words to write

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

## 3.13 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 cmd\_mem\_workq\_add\_work

**Prototype:**

```
void cmd_mem_workq_add_work(uint32_t ring_number, __xwrite void* xfer, uint32_t 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 3.370. cmd\_mem\_workq\_add\_work parameters**

Type	Name	Description
uint32_t	<i>ring_number</i>	MEM ring number to get data from (0-1023)
<u>_xwrite void*</u>	<i>xfer</i>	Transfer register(s) containing the work
uint32_t	<i>count</i>	Number of 32-bit words specifying the work (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.

### 3.14.1.2 cmd\_mem\_workq\_add\_work\_imm

**Prototype:**

```
void cmd_mem_workq_add_work_imm(uint32_t ring_number, uint32_t 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 3.371. cmd\_mem\_workq\_add\_work\_imm parameters**

Type	Name	Description
uint32_t	<i>ring_number</i>	MEM ring number to put get data from (0-1023)
uint32_t	<i>work_longword</i>	32-bit word describing the work

### 3.14.1.3 cmd\_mem\_workq\_add\_thread

**Prototype:**

```
void cmd_mem_workq_add_thread(uint32_t ring_number, __xread uint32_t* xfer, uint32_t 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 3.372. cmd\_mem\_workq\_add\_thread parameters**

Type	Name	Description
uint32_t	<i>ring_number</i>	MEM ring number to get data from (0-1023)
<u>_xread uint32_t*</u>	<i>xfer</i>	Transfer register(s) receiving the work
uint32_t	<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 3.373. MU Ring Defines**

Defined	Definition
MEM_RING_HEAD(desc)	((desc).head_ptr << 2)   (((desc).tail_ptr & ~0xFFFF) << 2)) Return the byte address of the head of a ring for a given descriptor.
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 3.374. 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

Name	Description
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 3.375. struct mem\_ring\_put\_status\_t**

Type	Name	Description
uint32_t	success:1	Bit is set when data was placed on ring.
uint32_t	reserved:7	Reserved.
uint32_t	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 3.376. union mem\_ring\_desc\_t**

Type	Name	Description
uint32_t	ring_size:4	Ring size specification and must be one of MEM_RING_SIZE.
uint32_t	reserved_1:2	Reserved.
uint32_t	head_ptr:24	24 - bit pointer - Byte address of ring buffer.

Type	Name	Description
		 <b>Note</b> Must be aligned to the byte size of the ring.
uint32_t	eop:1	This bit is set when an overflow occurred on journal command.  cmd_mem_ring_journal_buffer, cmd_mem_ring_fastjournal_imm
uint32_t	reserved_2:1	Reserved.
uint32_t	tail_ptr:30	Byte address of the tail of the ring.
uint32_t	q_type:2	Is set to 2 for ring intrinsic library functions.
uint32_t	q_loc:2	Locality specification of data.
uint32_t	reserved_3:4	Reserved.
uint32_t	q_page:2	Page of ring data.
uint32_t	q_count:24	Number of words on the ring.
uint32_t	reserved_4	
uint32_t	value[4]	Accessor to entire descriptor structure.

## 3.15.5 MU Ring Typedefs

### 3.15.5.1 MEM\_RING\_SIZE

Ring word size.

**Table 3.377. 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 3.378. 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 3.379. 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 3.380. typedef mem\_ring\_desc\_in\_mem\_t**

Type	Definition
mem_ring_desc_in_mem_t	__addr40 __align16 mem_ring_desc_t

### 3.15.5.5 mem\_ring\_put\_status\_in\_read\_reg\_t

mem\_ring\_put\_status\_t in a read register

**Table 3.381. typedef mem\_ring\_put\_status\_in\_read\_reg\_t**

Type	Definition
mem_ring_put_status_in_read_reg_t	__xread mem_ring_put_status_t

## 3.15.6 MU Ring Functions

### 3.15.6.1 cmd\_mem\_ring\_read\_desc

**Prototype:**

```
void cmd_mem_ring_read_desc(uint32_t ring_number, mem_ring_desc_in_mem_t*  
mem_ring_desc_in_mem)
```

**Description:**

Read MEM queue descriptor from MEM into queue array.

**Table 3.382. cmd\_mem\_ring\_read\_desc parameters**

Type	Name	Description
uint32_t	<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"> <li>• 0 - 1023: emem0 or island 24,</li> <li>• 1024 - 2047: emem1 or island 25,</li> <li>• 2048 - 3071: emem2 or island 26</li> </ul>
mem_ring_desc_in_mem_t*	<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 cmd\_mem\_ring\_write\_desc

**Prototype:**

```
void cmd_mem_ring_write_desc(uint32_t ring_number, mem_ring_desc_in_mem_t*
mem_ring_desc_in_mem)
```

**Description:**

Write MEM queue descriptor from queue array into MEM.

**Table 3.383. cmd\_mem\_ring\_write\_desc parameters**

Type	Name	Description
uint32_t	<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"> <li>• 0 - 1023: emem0 or island 24,</li> <li>• 1024 - 2047: emem1 or island 25,</li> <li>• 2048 - 3071: emem2 or island 26</li> </ul>
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 cmd\_mem\_ring\_push\_desc

**Prototype:**

```
void cmd_mem_ring_push_desc(uint32_t ring_number, __xread void* xfer, sync_t sync, SIGNAL*
sig_ptr)
```

**Description:**

Push MEM queue descriptor from queue array into transfer registers.

**Table 3.384. cmd\_mem\_ring\_push\_desc parameters**

Type	Name	Description
uint32_t	<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"> <li>• 0 - 1023: emem0 or island 24,</li> <li>• 1024 - 2047: emem1 or island 25,</li> <li>• 2048 - 3071: emem2 or island 26</li> </ul>
<u>xread void*</u>	<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 cmd\_mem\_ring\_read\_buffer\_unbounded

**Prototype:**

```
void cmd_mem_ring_read_buffer_unbounded(uint32_t ring_number, __xread uint32_t* xfer,
uint32_t count, uint32_t 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 3.385. cmd\_mem\_ring\_read\_buffer\_unbounded parameters**

Type	Name	Description
uint32_t	<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"> <li>• 0 - 1023: emem0 or island 24,</li> <li>• 1024 - 2047: emem1 or island 25,</li> <li>• 2048 - 3071: emem2 or island 26</li> </ul>
<u>xread uint32_t*</u>	<i>xfer</i>	Transfer register(s) receiving data
uint32_t	<i>count</i>	Number of 4-byte long words to get (1-16)
uint32_t	<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 cmd\_mem\_ring\_read\_buffer

**Prototype:**

```
void cmd_mem_ring_read_buffer(uint32_t ring_number, __xread uint32_t* xfer, uint32_t
count, uint32_t 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 3.386. cmd\_mem\_ring\_read\_buffer parameters**

Type	Name	Description
uint32_t	<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"> <li>• 0 - 1023: emem0 or island 24,</li> <li>• 1024 - 2047: emem1 or island 25,</li> <li>• 2048 - 3071: emem2 or island 26</li> </ul>
<u>__xread uint32_t*</u>	<i>xfer</i>	Transfer register(s) receiving data
uint32_t	<i>count</i>	Number of 4-byte long words to get (1-16)
uint32_t	<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

### 3.15.6.6 cmd\_mem\_ring\_write\_buffer\_unbounded

**Prototype:**

```
void cmd_mem_ring_write_buffer_unbounded(uint32_t ring_number, __xwrite void* xfer,
uint32_t count, uint32_t 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 3.387. cmd\_mem\_ring\_write\_buffer\_unbounded parameters**

Type	Name	Description
uint32_t	<i>ring_number</i>	Ring number. The range of the ring number identifies the external memory to use. <ul style="list-style-type: none"> <li>• 0 - 1023: emem0 or island 24,</li> </ul>

Type	Name	Description
		<ul style="list-style-type: none"> <li>• 1024 - 2047: emem1 or island 25,</li> <li>• 2048 - 3071: emem2 or island 26</li> </ul>
<code>__xwrite void*</code>	<code>xfer</code>	Pointer to write xfer register(s) containing the data to write
<code>uint32_t</code>	<code>count</code>	Number of 32-bit word to put on the ring (1-16)
<code>uint32_t</code>	<code>offset</code>	Long word count offset from base of 16M word ring
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raised upon completion

### 3.15.6.7 cmd\_mem\_ring\_write\_buffer

**Prototype:**

```
void cmd_mem_ring_write_buffer(uint32_t ring_number, __xwrite void* xfer, uint32_t count,
                               uint32_t offset, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

Write data at an offset into a ring.

**Table 3.388. cmd\_mem\_ring\_write\_buffer parameters**

Type	Name	Description
<code>uint32_t</code>	<code>ring_number</code>	Ring number. The range of the ring number identifies the external memory to use. <ul style="list-style-type: none"> <li>• 0 - 1023: emem0 or island 24,</li> <li>• 1024 - 2047: emem1 or island 25,</li> <li>• 2048 - 3071: emem2 or island 26</li> </ul>
<code>__xwrite void*</code>	<code>xfer</code>	Pointer to write xfer register(s) containing the data to write
<code>uint32_t</code>	<code>count</code>	Number of 32-bit word to put on the ring (1-16)
<code>uint32_t</code>	<code>offset</code>	Long word count offset from base of ring
<code>sync_t</code>	<code>sync</code>	Type of synchronization to use
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raised upon completion

### 3.15.6.8 cmd\_mem\_ring\_add\_tail

**Prototype:**

```
void cmd_mem_ring_add_tail(uint32_t ring_number, uint32_t count)
```

**Description:**

Add a value to the tail of a ring wrapping it appropriately.

**Table 3.389. cmd\_mem\_ring\_add\_tail parameters**

Type	Name	Description
uint32_t	<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"><li>• 0 - 1023: emem0 or island 24,</li><li>• 1024 - 2047: emem1 or island 25,</li><li>• 2048 - 3071: emem2 or island 26</li></ul>
uint32_t	<i>count</i>	24 bit count to add to the tail

### 3.15.6.9 cmd\_mem\_ring\_put\_buffer

**Prototype:**

```
mem_ring_put_status_in_read_reg_t* cmd_mem_ring_put_buffer(uint32_t ring_number, __xrw
void* xfer, uint32_t count, sync_t sync, SIGNAL_PAIR* sig_pair_ptr)
```

**Description:**

Add data to the tail of a ring checking for overflow.

**Table 3.390. cmd\_mem\_ring\_put\_buffer parameters**

Type	Name	Description
uint32_t	<i>ring_number</i>	Ring number. The range of the ring number identifies the external memory to use. <ul style="list-style-type: none"><li>• 0 - 1023: emem0 or island 24,</li><li>• 1024 - 2047: emem1 or island 25,</li><li>• 2048 - 3071: emem2 or island 26</li></ul>
<u>xrw</u> void*	<i>xfer</i>	Pointer to read_write xfer register(s) containing the data to put on the ring
uint32_t	<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 cmd\_mem\_ring\_journal\_buffer

**Prototype:**

```
void cmd_mem_ring_journal_buffer(uint32_t ring_number, __xwrite void* xfer, uint32_t count, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

Add data to the tail of a ring without checking for overflow.

**Table 3.391. cmd\_mem\_ring\_journal\_buffer parameters**

Type	Name	Description
uint32_t	<i>ring_number</i>	Ring number. The range of the ring number identifies the external memory to use. <ul style="list-style-type: none"> <li>• 0 - 1023: emem0 or island 24,</li> <li>• 1024 - 2047: emem1 or island 25,</li> <li>• 2048 - 3071: emem2 or island 26</li> </ul>
__xwrite void*	<i>xfer</i>	Pointer to write xfer register(s) containing the data to put on the ring
uint32_t	<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 cmd\_mem\_ring\_get\_buffer

**Prototype:**

```
void cmd_mem_ring_get_buffer(uint32_t ring_number, __xread uint32_t* xfer, uint32_t 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 3.392. cmd\_mem\_ring\_get\_buffer parameters**

Type	Name	Description
uint32_t	<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"> <li>• 0 - 1023: emem0 or island 24,</li> <li>• 1024 - 2047: emem1 or island 25,</li> <li>• 2048 - 3071: emem2 or island 26</li> </ul>
__xread uint32_t*	<i>xfer</i>	Transfer register(s) receiving data

Type	Name	Description
uint32_t	<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 cmd\_mem\_ring\_get\_buffer\_eop

**Prototype:**

```
void cmd_mem_ring_get_buffer_eop(uint32_t ring_number, __xread uint32_t* xfer, uint32_t
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 3.393. cmd\_mem\_ring\_get\_buffer\_eop parameters**

Type	Name	Description
uint32_t	<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"> <li>• 0 - 1023: emem0 or island 24,</li> <li>• 1024 - 2047: emem1 or island 25,</li> <li>• 2048 - 3071: emem2 or island 26</li> </ul>
__xread uint32_t*	<i>xfer</i>	Transfer register(s) receiving data
uint32_t	<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 cmd\_mem\_ring\_get\_buffer\_freely

**Prototype:**

```
void cmd_mem_ring_get_buffer_freely(uint32_t ring_number, __xread uint32_t* xfer, uint32_t
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(uint32_t)) uint32_t ring_mem[512];
uint32_t mu_island = 2;
uint32_t ring_number = (mu_island << 10) | 2; // placing ring on emem2
SIGNAL_PAIR sig_pair;
SIGNAL sig;

cmd_mem_ring_init(ring_number, RING_SIZE_512, ring_mem, ring_mem, 0);

// Put some words on ring
{
    __xrw uint32_t data[3];
    __xread mem_ring_put_status_t *ring_put_status;
    __xread uint32_t status;

    data[0] = 0x12345678;
    data[1] = 0x87654321;
    data[2] = 0x10101010;

    ring_put_status = cmd_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 uint32_t data_get[4];

    cmd_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 3.394. cmd\_mem\_ring\_get\_buffer\_freely parameters**

Type	Name	Description
uint32_t	<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"> <li>• 0 - 1023: emem0 or island 24,</li> <li>• 1024 - 2047: emem1 or island 25,</li> <li>• 2048 - 3071: emem2 or island 26</li> </ul>
<u>_xread uint32_t*</u>	<i>xfer</i>	Transfer register(s) receiving data
uint32_t	<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

### 3.15.6.14 cmd\_mem\_ring\_pop\_buffer

**Prototype:**

```
void cmd_mem_ring_pop_buffer(uint32_t ring_number, __xread uint32_t* xfer, uint32_t 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 3.395. cmd\_mem\_ring\_pop\_buffer parameters**

Type	Name	Description
uint32_t	<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"> <li>• 0 - 1023: emem0 or island 24,</li> <li>• 1024 - 2047: emem1 or island 25,</li> <li>• 2048 - 3071: emem2 or island 26</li> </ul>
<u>_xread uint32_t*</u>	<i>xfer</i>	Transfer register(s) receiving data
uint32_t	<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 cmd\_mem\_ring\_pop\_buffer\_freely

**Prototype:**

```
void cmd_mem_ring_pop_buffer_freely(uint32_t ring_number, __xread uint32_t* xfer, uint32_t
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 3.396. cmd\_mem\_ring\_pop\_buffer\_freely parameters**

Type	Name	Description
uint32_t	<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"> <li>• 0 - 1023: emem0 or island 24,</li> <li>• 1024 - 2047: emem1 or island 25,</li> <li>• 2048 - 3071: emem2 or island 26</li> </ul>
<u>__xread</u> uint32_t*	<i>xfer</i>	Transfer register(s) receiving data
uint32_t	<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 cmd\_mem\_ring\_pop\_buffer\_eop

**Prototype:**

```
void cmd_mem_ring_pop_buffer_eop(uint32_t ring_number, __xread uint32_t* xfer, uint32_t
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 3.397. cmd\_mem\_ring\_pop\_buffer\_eop parameters**

Type	Name	Description
uint32_t	<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"> <li>• 0 - 1023: emem0 or island 24,</li> <li>• 1024 - 2047: emem1 or island 25,</li> </ul>

Type	Name	Description
		<ul style="list-style-type: none"> <li>• 2048 - 3071: emem2 or island 26</li> </ul>
<code>__xread</code> <code>uint32_t *</code>	<code>xfer</code>	Transfer register(s) receiving data
<code>uint32_t</code>	<code>count</code>	Number of 32-bit words to get (1-16)
<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>	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 cmd\_mem\_ring\_fastjournal\_imm

**Prototype:**

```
void cmd_mem_ring_fastjournal_imm(uint32_t ring_number, uint32_t 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 3.398. cmd\_mem\_ring\_fastjournal\_imm parameters**

Type	Name	Description
<code>uint32_t</code>	<code>ring_number</code>	Ring number. The range of the ring number identifies the external memory to use. <ul style="list-style-type: none"> <li>• 0 - 1023: emem0 or island 24,</li> <li>• 1024 - 2047: emem1 or island 25,</li> <li>• 2048 - 3071: emem2 or island 26</li> </ul>
<code>uint32_t</code>	<code>longword</code>	32-bit word to put on the ring

### 3.15.6.18 cmd\_mem\_ring\_init

**Prototype:**

```
void cmd_mem_ring_init(uint32_t ring_number, enum MEM_RING_SIZE size, __addr40 __mem
uint32_t* head_ptr, __addr40 __mem uint32_t* tail_ptr, uint32_t 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(uint32_t)) uint32_t ring_mem[512];
uint32_t          mu_island = 2;
uint32_t          ring_number = (mu_island << 10) | 2;      // placing ring on emem2
SIGNAL_PAIR        sig_pair;
SIGNAL             sig;

cmd_mem_ring_init(ring_number, RING_SIZE_512, ring_mem, ring_mem, 0);

// Put some words on ring
{
    __xrw uint32_t          data[3];
    __xread mem_ring_put_status_t      *ring_put_status;
    __xread uint32_t          status;

    data[0] = 0x12345678;
    data[1] = 0x87654321;
    data[2] = 0x10101010;

    ring_put_status = cmd_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;
    cmd_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 uint32_t          data_get;

    cmd_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 uint32_t      data_pop;

        cmd_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 3.399. cmd\_mem\_ring\_init parameters**

Type	Name	Description
uint32_t	<i>ring_number</i>	Ring number. The range of the ring number identifies the external memory to use. <ul style="list-style-type: none"> <li>• 0 - 1023: emem0 or island 24,</li> <li>• 1024 - 2047: emem1 or island 25,</li> <li>• 2048 - 3071: emem2 or island 26</li> </ul>
enum MEM_RING_SIZE	<i>size</i>	The ring size specified as one of enum MEM_RING_SIZE
__addr40 __mem uint32_t*	<i>head_ptr</i>	Pointer to the head of the ring. Must be aligned to byte size of ring.
__addr40 __mem uint32_t*	<i>tail_ptr</i>	Initial pointer to the tail of the ring. For an empty ring this is set to the head.
uint32_t	<i>count</i>	Initial number of words on the ring. For an empty ring this must be zero.

### 3.15.6.19 cmd\_mem\_ring\_init\_with\_loc\_and\_page

**Prototype:**

```
void cmd_mem_ring_init_with_loc_and_page(uint32_t ring_number, enum MEM_RING_SIZE size,
                                         __addr40 __mem uint32_t* head_ptr, __addr40 __mem uint32_t* tail_ptr, uint32_t count,
                                         uint32_t q_locality, uint32_t 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 3.400. cmd\_mem\_ring\_init\_with\_loc\_and\_page parameters**

Type	Name	Description
uint32_t	<i>ring_number</i>	Ring number. The range of the ring number identifies the external memory to use. <ul style="list-style-type: none"> <li>• 0 - 1023: emem0 or island 24,</li> <li>• 1024 - 2047: emem1 or island 25,</li> <li>• 2048 - 3071: emem2 or island 26</li> </ul>
enum MEM_RING_SIZE	<i>size</i>	The ring size specified as one of enum MEM_RING_SIZE
<u>addr40</u> <u>mem</u> uint32_t*	<i>head_ptr</i>	Pointer to the head of the ring. Must be aligned to byte size of ring.
<u>addr40</u> <u>mem</u> uint32_t*	<i>tail_ptr</i>	Initial pointer to the tail of the ring. For an empty ring this is set to the head.
uint32_t	<i>count</i>	Initial number of words on the ring. For an empty ring this must be zero.
uint32_t	<i>q_locality</i>	Locality of the ring. Please refer to Ring/Circular buffer section in PRM for detail.
uint32_t	<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 3.401. struct mem\_ticket\_line\_push\_t**

Type	Name	Description
uint32_t	master_cluster:4	Cluster ID of ME that should receive the result.
uint32_t	master_is_me:1	Bit indicating if the result receiver is a ME.  This bit should be set to 1.
uint32_t	master_id:3	Microengine ID that should receive the result.

Type	Name	Description
uint32_t	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
uint32_t	dataref_address:11	Register or CSR address.
uint32_t	current_ticket:10	Current ticket number released.
uint32_t	bitmap0:32	Bitmap of tickets presented for release.  Current ticket+1 to current ticket+31.
uint32_t	bitmap1:32	Current ticket+32 to current ticket+63.
uint32_t	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 3.402. struct mem\_ticket\_line\_t**

Type	Name	Description
uint32_t	reserved:22	Reserved.
uint32_t	current_ticket:10	Current ticket number released.
uint32_t	bitmap0:32	Bitmap of tickets presented for release.  Current ticket+1 to current ticket+31.
uint32_t	bitmap1:32	Current ticket+32 to current ticket+63.
uint32_t	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 3.403. 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 3.404. typedef mem\_ticket\_line\_push\_t**

Type	Definition
mem_ticket_line_push_t	struct mem_ticket_line_push_t

## 3.16.3 MU Ticket Functions

### 3.16.3.1 cmd\_mem\_ticket\_release\_ptr32

**Prototype:**

```
void cmd_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 3.405. cmd\_mem\_ticket\_release\_ptr32 parameters**

Type	Name	Description
__xrw void*	xfer	[In] Ticket number to release [Out] Release result
volatile void __addr32 __mem*	address	MEM address where the 128-bit ticket line starts
sync_t	sync	Type of synchronization to use (must be sig_done)
SIGNAL_PAIR*	sig_pair_ptr	Signal to raise upon completion

### 3.16.3.2 cmd\_mem\_ticket\_release\_ptr40

**Prototype:**

```
void cmd_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 3.406. cmd\_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 sig_done)
<code>SIGNAL_PAIR*</code>	<code>sig_pair_ptr</code>	Signal to raise upon completion

### 3.16.3.3 cmd\_mem\_ticket\_release\_push\_ptr32

**Prototype:**

```
void cmd_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 3.407. cmd\_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 sig_done)
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise in caller upon completion

### 3.16.3.4 cmd\_mem\_ticket\_release\_push\_ptr40

**Prototype:**

```
void cmd_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 3.408. cmd\_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 <code>sig_done</code> )
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise in caller upon completion

### 3.16.3.5 cmd\_mem\_ticket\_line\_init\_ptr32

**Prototype:**

```
void cmd_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 3.409. cmd\_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 <code>sig_done</code> )
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion

### 3.16.3.6 cmd\_mem\_ticket\_line\_init\_ptr40

**Prototype:**

```
void cmd_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 3.410. cmd\_mem\_ticket\_line\_init\_ptr40 parameters**

Type	Name	Description
mem_ticket_line_t*	mem_ticket_line	Pre-populated mem_ticket_line_t struct to initialize MEM with
volatile void __addr40 __mem*	address	40-bit MEM address at which the 128-bit ticket structure should start
sync_t	sync	Type of synchronization to use (must be sig_done)
SIGNAL*	sig_ptr	Signal to raise upon completion

### 3.16.3.7 cmd\_mem\_ticket\_line\_push\_init\_ptr32

**Prototype:**

```
void cmd_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 3.411. cmd\_mem\_ticket\_line\_push\_init\_ptr32 parameters**

Type	Name	Description
mem_ticket_line_push_t*	mem_ticket_line_push	Pre-populated mem_ticket_line_push_t struct to initialize MEM with
volatile void __addr32 __mem*	address	MEM address at which the 128-bit ticket structure should start
sync_t	sync	Type of synchronization to use (must be sig_done)
SIGNAL*	sig_ptr	Signal to raise upon completion

### 3.16.3.8 cmd\_mem\_ticket\_line\_push\_init\_ptr40

**Prototype:**

```
void cmd_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 3.412. cmd\_mem\_ticket\_line\_push\_init\_ptr40 parameters**

Type	Name	Description
mem_ticket_line_push_t*	mem_ticket_line_push	Pre-populated mem_ticket_line_push_t struct to initialize MEM with
volatile void __addr40 __mem*	address	40-bit MEM address at which the 128-bit ticket structure should start
sync_t	sync	Type of synchronization to use (must be sig_done)
SIGNAL*	sig_ptr	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 3.413. 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 3.414. struct mem\_lockq256\_t**

Type	Name	Description
mem_lockq_desc_t	desc	Lock queue descriptor.
mem_lockq_entry_t	q[15]	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 3.415. union mem\_lockq\_desc\_t**

Type	Name	Description
uint32_t	reserved_1:4	Reserved.
uint32_t	event_source:12	Event source number to include in event.
uint32_t	reserved_2:6	Reserved.
uint32_t	generate_event:1	Generate event on under or overflow.
uint32_t	entry_size:1	Entry size - must be 0 to be 16 bits.
uint32_t	overflow:1	Set on overflow.
uint32_t	reserved_3:2	Reserved.
uint32_t	locked:1	Set when the lock has been given out.

Type	Name	Description
uint32_t	num_elements:4	0-6 or 0-14 depending on if it is a 128 or 256 bit queue.
uint32_t	value	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 3.416. 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 3.417. 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 3.418. 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 3.419. `typedef mem_lockq128_t`**

Type	Definition
<code>mem_lockq128_t</code>	<code>struct mem_lockq128_t</code>

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 3.420. `typedef mem_lockq256_t`**

Type	Definition
<code>mem_lockq256_t</code>	<code>struct mem_lockq256_t</code>

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 3.421. `typedef mem_lockq128_in_mem_t`**

Type	Definition
<code>mem_lockq128_in_mem_t</code>	<code>volatile __addr32 __align16 mem_lockq128_t</code>

### 3.17.3.6 `mem_lockq128_ptr40_t`

128 bit lock queue pointer aligned in 40-bit addressed MEM.

**Table 3.422. `typedef mem_lockq128_ptr40_t`**

Type	Definition
<code>mem_lockq128_ptr40_t</code>	<code>volatile __addr40 __mem __align16 mem_lockq128_t*</code>

### 3.17.3.7 `mem_lockq256_in_mem_t`

256 bit lock queue aligned in 32-bit addressed MEM.

**Table 3.423. `typedef mem_lockq256_in_mem_t`**

Type	Definition
<code>mem_lockq256_in_mem_t</code>	<code>volatile __addr32 __align16 mem_lockq256_t</code>

### 3.17.3.8 `mem_lockq256_ptr40_t`

256 bit lock queue pointer aligned in 40-bit addressed MEM.

**Table 3.424. `typedef mem_lockq256_ptr40_t`**

Type	Definition
<code>mem_lockq256_ptr40_t</code>	<code>volatile __addr40 __mem __align16 mem_lockq256_t*</code>

## 3.17.4 MU Queue Lock Functions

### 3.17.4.1 `cmd_mem_lockq128_lock_ptr32`

**Prototype:**

```
void cmd_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 3.425. `cmd_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 `cmd_mem_lockq128_lock_ind_ptr32`

**Prototype:**

```
void cmd_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 3.426. cmd\_mem\_lockq128\_lock\_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
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.

### 3.17.4.3 cmd\_mem\_lockq128\_lock\_ptr40

**Prototype:**

```
void cmd_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 3.427. cmd\_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 cmd\_mem\_lockq128\_lock\_ind\_ptr40

**Prototype:**

```
void cmd_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 3.428. cmd\_mem\_lockq128\_lock\_ind\_ptr40 parameters**

Type	Name	Description
mem_lockq128_ptr40_t	<i>lockq</i>	Pointer to lock queue in MEM

Type	Name	Description
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 cmd\_mem\_lockq256\_lock\_ptr32

**Prototype:**

```
void cmd_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 3.429. cmd\_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 cmd\_mem\_lockq256\_lock\_ind\_ptr32

**Prototype:**

```
void cmd_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 3.430. cmd\_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.

Type	Name	Description
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Even signal is raised upon completion. Odd signal is raised on overflow.

### 3.17.4.7 cmd\_mem\_lockq256\_lock\_ptr40

**Prototype:**

```
void cmd_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 3.431. cmd\_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 cmd\_mem\_lockq256\_lock\_ind\_ptr40

**Prototype:**

```
void cmd_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 3.432. cmd\_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 cmd\_mem\_lockq128\_unlock\_ptr32

**Prototype:**

```
void cmd_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 3.433. cmd\_mem\_lockq128\_unlock\_ptr32 parameters**

Type	Name	Description
<code>__mem mem_lockq128_in_mem_t*</code>	<code>lockq</code>	Pointer to lock queue in MEM

### 3.17.4.10 cmd\_mem\_lockq128\_unlock\_ind\_ptr32

**Prototype:**

```
void cmd_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 3.434. cmd\_mem\_lockq128\_unlock\_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

### 3.17.4.11 cmd\_mem\_lockq128\_unlock\_ptr40

**Prototype:**

```
void cmd_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 3.435. cmd\_mem\_lockq128\_unlock\_ptr40 parameters**

Type	Name	Description
<code>mem_lockq128_ptr40_t</code>	<code>lockq</code>	Pointer to lock queue in MEM

### 3.17.4.12 cmd\_mem\_lockq128\_unlock\_ind\_ptr40

**Prototype:**

```
void cmd_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 3.436. cmd\_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 cmd\_mem\_lockq256\_unlock\_ptr32

**Prototype:**

```
void cmd_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 3.437. cmd\_mem\_lockq256\_unlock\_ptr32 parameters**

Type	Name	Description
__mem mem_lockq256_in_mem_t*	<i>lockq</i>	Pointer to lock queue in MEM

### 3.17.4.14 cmd\_mem\_lockq256\_unlock\_ind\_ptr32

**Prototype:**

```
void cmd_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 3.438. cmd\_mem\_lockq256\_unlock\_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

### 3.17.4.15 cmd\_mem\_lockq256\_unlock\_ptr40

**Prototype:**

```
void cmd_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 3.439. cmd\_mem\_lockq256\_unlock\_ptr40 parameters**

Type	Name	Description
mem_lockq256_ptr40_t	<i>lockq</i>	Pointer to lock queue in MEM

### 3.17.4.16 cmd\_mem\_lockq256\_unlock\_ind\_ptr40

**Prototype:**

```
void cmd_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 3.440. cmd\_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 cmd\_mem\_lockq128\_init\_ptr32

**Prototype:**

```
void cmd_mem_lockq128_init_ptr32(__addr32 __mem mem_lockq128_in_mem_t* lockq, uint32_t generate_event, uint32_t 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 3.441. cmd\_mem\_lockq128\_init\_ptr32 parameters**

Type	Name	Description
__addr32 __mem mem_lockq128_in_mem_t*	<i>lockq</i>	Pointer to lock queue in MEM
uint32_t	<i>generate_event</i>	Enable event on overflow or underflow
uint32_t	<i>event_source</i>	Event to present on event bus if generate_event is 1

### 3.17.4.18 cmd\_mem\_lockq128\_init\_ptr40

**Prototype:**

```
void cmd_mem_lockq128_init_ptr40(mem_lockq128_ptr40_t lockq, uint32_t generate_event,  
uint32_t 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 3.442. cmd\_mem\_lockq128\_init\_ptr40 parameters**

Type	Name	Description
mem_lockq128_ptr40_t	<i>lockq</i>	Pointer to lock queue in MEM
uint32_t	<i>generate_event</i>	Enable event on overflow or underflow
uint32_t	<i>event_source</i>	Event to present on event bus if generate_event is 1

### 3.17.4.19 cmd\_mem\_lockq256\_init\_ptr32

**Prototype:**

```
void cmd_mem_lockq256_init_ptr32(__mem mem_lockq256_in_mem_t* lockq, uint32_t  
generate_event, uint32_t 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 3.443. cmd\_mem\_lockq256\_init\_ptr32 parameters**

Type	Name	Description
__mem mem_lockq256_in_mem_t*	<i>lockq</i>	Pointer to lock queue in MEM
uint32_t	<i>generate_event</i>	Enable event on overflow or underflow
uint32_t	<i>event_source</i>	Event to present on event bus if generate_event is 1

### 3.17.4.20 cmd\_mem\_lockq256\_init\_ptr40

**Prototype:**

```
void cmd_mem_lockq256_init_ptr40(mem_lockq256_ptr40_t lockq, uint32_t generate_event,
uint32_t 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 3.444. cmd\_mem\_lockq256\_init\_ptr40 parameters**

Type	Name	Description
mem_lockq256_ptr40_t	<i>lockq</i>	Pointer to lock queue in MEM
uint32_t	<i>generate_event</i>	Enable event on overflow or underflow
uint32_t	<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 3.445. struct mem\_lock128\_t**

Type	Name	Description
uint32_t	<i>value[4]</i>	Storage for the memory lock entries.

#### 3.18.1.2 mem\_lock256\_t

256-bit memory lock container type.

**Table 3.446. struct mem\_lock256\_t**

Type	Name	Description
uint32_t	value[8]	Storage for the memory lock entries.

### 3.18.1.3 mem\_lock384\_t

384-bit memory lock container type.

**Table 3.447. struct mem\_lock384\_t**

Type	Name	Description
uint32_t	value[12]	Storage for the memory lock entries.

### 3.18.1.4 mem\_lock512\_t

512-bit memory lock container type.

**Table 3.448. struct mem\_lock512\_t**

Type	Name	Description
uint32_t	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 3.449. union mem\_lock\_in\_t**

Type	Name	Description
uint32_t	reserved:16	Unused, the regions are addressable to 64k regions.
uint32_t	address:16	Region address to lock.  <div style="border: 1px solid blue; padding: 5px; margin-top: 10px;">  <b>Note</b>             Bit 0 is reserved and is assumed to be 0.         </div>
uint32_t	value	Accessor to entire structure.

### 3.18.2.2 mem\_lock\_out\_t

Output type for the memory lock operation.

**Table 3.450. union mem\_lock\_out\_t**

Type	Name	Description
uint32_t	match_bitf:16	Bitfield of matching entries.
uint32_t	reserved:8	Reserved.
uint32_t	added:1	When set, this field indicates that an entry was added.  <div style="border: 1px solid blue; padding: 5px; margin-top: 10px;">  <b>Note</b>            This field is also set when the memory lock queue is full.         </div>
uint32_t	first_match:7	First matched entry number, or on a new memory lock entry where the entry was added.  <div style="border: 1px solid blue; padding: 5px; margin-top: 10px;">  <b>Note</b>            All-bits in this field are set when the memory lock queue is full.         </div>
uint32_t	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 3.451. 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 3.452. `typedef mem_lock256_t`**

Type	Definition
<code>mem_lock256_t</code>	<code>struct mem_lock256_t</code>

### 3.18.3.3 `mem_lock384_t`

384-bit memory lock container type.

**Table 3.453. `typedef mem_lock384_t`**

Type	Definition
<code>mem_lock384_t</code>	<code>struct mem_lock384_t</code>

### 3.18.3.4 `mem_lock512_t`

512-bit memory lock container type.

**Table 3.454. `typedef mem_lock512_t`**

Type	Definition
<code>mem_lock512_t</code>	<code>struct mem_lock512_t</code>

### 3.18.3.5 `mem_lock_in_t`

Input type for the memory lock operation.

The address field must be populated before requesting the memory lock.

**Table 3.455. `typedef mem_lock_in_t`**

Type	Definition
<code>mem_lock_in_t</code>	<code>union mem_lock_in_t</code>

### 3.18.3.6 `mem_lock_out_t`

Output type for the memory lock operation.

**Table 3.456. `typedef mem_lock_out_t`**

Type	Definition
<code>mem_lock_out_t</code>	<code>union mem_lock_out_t</code>

## 3.18.4 MU Lock Functions

### 3.18.4.1 cmd\_mem\_lock128\_init\_ptr32

**Prototype:**

```
void cmd_mem_lock128_init_ptr32(__addr32 __mem __align16 mem_lock128_t* mem_lock)
```

**Description:**

Initialize a 128-bit memory lock container in 32-bit addressed MEM in memory.

**Table 3.457. cmd\_mem\_lock128\_init\_ptr32 parameters**

Type	Name	Description
<code>__addr32 __mem __align16 mem_lock128_t*</code>	<code>mem_lock</code>	Pointer to the memory lock container in MEM

### 3.18.4.2 cmd\_mem\_lock128\_init\_ptr40

**Prototype:**

```
void cmd_mem_lock128_init_ptr40(__addr40 __mem __align16 mem_lock128_t* mem_lock)
```

**Description:**

Initialize a 128-bit memory lock container in 40-bit addressed MEM.

**Table 3.458. cmd\_mem\_lock128\_init\_ptr40 parameters**

Type	Name	Description
<code>__addr40 __mem __align16 mem_lock128_t*</code>	<code>mem_lock</code>	Pointer to the memory lock container in MEM

### 3.18.4.3 cmd\_mem\_lock256\_init\_ptr32

**Prototype:**

```
void cmd_mem_lock256_init_ptr32(__addr32 __mem __align16 mem_lock256_t* mem_lock)
```

**Description:**

Initialize a 256-bit memory lock container in 32-bit addressed MEM.

**Table 3.459. cmd\_mem\_lock256\_init\_ptr32 parameters**

Type	Name	Description
<code>__addr32 __mem __align16 mem_lock256_t*</code>	<code>mem_lock</code>	Pointer to the memory lock container in MEM

### 3.18.4.4 cmd\_mem\_lock256\_init\_ptr40

**Prototype:**

```
void cmd_mem_lock256_init_ptr40(__addr40 __mem __align16 mem_lock256_t* mem_lock)
```

**Description:**

Initialize a 256-bit lock container in 40-bit addressed MEM.

**Table 3.460. cmd\_mem\_lock256\_init\_ptr40 parameters**

Type	Name	Description
<code>__addr40 __mem __align16 mem_lock256_t*</code>	<code>mem_lock</code>	Pointer to the memory lock container in MEM

### 3.18.4.5 cmd\_mem\_lock384\_init\_ptr32

**Prototype:**

```
void cmd_mem_lock384_init_ptr32(__addr32 __mem __align16 mem_lock384_t* mem_lock)
```

**Description:**

Initialize a 384-bit memory lock container in 32-bit addressed MEM in memory.

**Table 3.461. cmd\_mem\_lock384\_init\_ptr32 parameters**

Type	Name	Description
<code>__addr32 __mem __align16 mem_lock384_t*</code>	<code>mem_lock</code>	Pointer to the memory lock container in MEM

### 3.18.4.6 cmd\_mem\_lock384\_init\_ptr40

**Prototype:**

```
void cmd_mem_lock384_init_ptr40(__addr40 __mem __align16 mem_lock384_t* mem_lock)
```

**Description:**

Initialize a 384-bit memory lock container in 40-bit addressed MEM.

**Table 3.462. cmd\_mem\_lock384\_init\_ptr40 parameters**

Type	Name	Description
<code>__addr40 __mem __align16 mem_lock384_t*</code>	<code>mem_lock</code>	Pointer to the memory lock container in MEM

### 3.18.4.7 cmd\_mem\_lock512\_init\_ptr32

**Prototype:**

```
void cmd_mem_lock512_init_ptr32(__addr32 __mem __align16 mem_lock512_t* mem_lock)
```

**Description:**

Initialize a 512-bit memory lock container in 32-bit addressed MEM in memory.

**Table 3.463. cmd\_mem\_lock512\_init\_ptr32 parameters**

Type	Name	Description
<code>__addr32 __mem __align16 mem_lock512_t*</code>	<code>mem_lock</code>	Pointer to the memory lock container in MEM

### 3.18.4.8 cmd\_mem\_lock512\_init\_ptr40

**Prototype:**

```
void cmd_mem_lock512_init_ptr40(__addr40 __mem __align16 mem_lock512_t* mem_lock)
```

**Description:**

Initialize a 512-bit memory lock container in 40-bit addressed MEM.

**Table 3.464. cmd\_mem\_lock512\_init\_ptr40 parameters**

Type	Name	Description
<code>__addr40 __mem __align16 mem_lock512_t*</code>	<code>mem_lock</code>	Pointer to the memory lock container in MEM

### 3.18.4.9 cmd\_mem\_lock128\_ptr32

**Prototype:**

```
__xread mem_lock_out_t* cmd_mem_lock128_ptr32(__addr32 __mem __align16 mem_lock128_t*  
mem_lock, __xwrite mem_lock_in_t* xfer, uint32_t 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 3.465. cmd\_mem\_lock128\_ptr32 parameters**

Type	Name	Description
<code>__addr32 __mem_align16 mem_lock128_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>uint32_t</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`

### 3.18.4.10 cmd\_mem\_lock128\_ptr40

**Prototype:**

```
__xread mem_lock_out_t* cmd_mem_lock128_ptr40(__addr40 __mem __align16 mem_lock128_t*
mem_lock, __xwrite mem_lock_in_t* xfer, uint32_t lsbzeroes, 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 3.466. cmd\_mem\_lock128\_ptr40 parameters**

Type	Name	Description
<code>__addr40 __mem __align16 mem_lock128_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>uint32_t</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 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`

### 3.18.4.11 mem\_lock256\_ptr32

**Prototype:**

```
__xread mem_lock_out_t* mem_lock256_ptr32(__addr32 __mem __align16 mem_lock256_t* mem_lock,
__xwrite mem_lock_in_t* xfer, uint32_t 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 3.467. mem\_lock256\_ptr32 parameters**

Type	Name	Description
<code>__addr32 __mem __align16 mem_lock256_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>uint32_t</code>	<code>lsbzeroes</code>	Number of least significant-bits to set to zero in the lock mask. This represent the lock region size.

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 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:**

```
__xread mem_lock_out_t* mem_lock256_ptr40(__addr40 __mem __align16 mem_lock256_t* mem_lock,
__xwrite mem_lock_in_t* xfer, uint32_t 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 3.468. mem\_lock256\_ptr40 parameters**

Type	Name	Description
__addr40 __mem __align16 mem_lock256_t*	<i>mem_lock</i>	Pointer to the memory lock container in MEM
__xwrite mem_lock_in_t*	<i>xfer</i>	Pointer to write xfer register containing the region address
uint32_t	<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

### 3.18.4.13 mem\_lock384\_ptr32

**Prototype:**

```
__xread mem_lock_out_t* mem_lock384_ptr32(__addr32 __mem __align16 mem_lock384_t* mem_lock,  
__xwrite mem_lock_in_t* xfer, uint32_t 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 3.469. mem\_lock384\_ptr32 parameters**

Type	Name	Description
<code>__addr32 __mem __align16 mem_lock384_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>uint32_t</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`

### 3.18.4.14 mem\_lock384\_ptr40

**Prototype:**

```
__xread mem_lock_out_t* mem_lock384_ptr40(__addr40 __mem __align16 mem_lock384_t* mem_lock,  
__xwrite mem_lock_in_t* xfer, uint32_t lsbzeroes, 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 3.470. `mem_lock384_ptr40` parameters**

Type	Name	Description
<code>__addr40 __mem __align16 mem_lock384_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>uint32_t</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`

## 3.18.4.15 `mem_lock512_ptr32`

**Prototype:**

```
__xread mem_lock_out_t* mem_lock512_ptr32(__addr32 __mem __align16 mem_lock512_t* mem_lock,
__xwrite mem_lock_in_t* xfer, uint32_t 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 3.471. `mem_lock512_ptr32` parameters**

Type	Name	Description
<code>__addr32 __mem __align16 mem_lock512_t*</code>	<code>mem_lock</code>	Pointer to the memory lock container in MEM

Type	Name	Description
<code>__xwrite mem_lock_in_t*</code>	<code>xfer</code>	Pointer to write xfer register containing the region address
<code>uint32_t</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 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`

### 3.18.4.16 mem\_lock512\_ptr40

**Prototype:**

```
__xread mem_lock_out_t* mem_lock512_ptr40(__addr40 __mem __align16 mem_lock512_t* mem_lock,
__xwrite mem_lock_in_t* xfer, uint32_t 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 3.472. mem\_lock512\_ptr40 parameters**

Type	Name	Description
<code>__addr40 __mem __align16 mem_lock512_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>uint32_t</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 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`

### 3.18.4.17 cmd\_mem\_unlock128\_ptr32

**Prototype:**

```
void cmd_mem_unlock128_ptr32(__addr32 __mem __align16 mem_lock128_t* mem_lock, uint32_t
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 3.473. cmd\_mem\_unlock128\_ptr32 parameters**

Type	Name	Description
<code>__addr32 __mem __align16 mem_lock128_t*</code>	<code>mem_lock</code>	Pointer to the memory lock container in MEM
<code>uint32_t</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

### 3.18.4.18 cmd\_mem\_unlock128\_ptr40

**Prototype:**

```
void cmd_mem_unlock128_ptr40(__addr40 __mem __align16 mem_lock128_t* mem_lock, uint32_t
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 3.474. cmd\_mem\_unlock128\_ptr40 parameters**

Type	Name	Description
<code>__addr40 __mem __align16 mem_lock128_t*</code>	<code>mem_lock</code>	Pointer to the memory lock container in MEM
<code>uint32_t</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(__addr32 __mem __align16 mem_lock256_t* mem_lock, uint32_t 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 3.475. mem\_unlock256\_ptr32 parameters**

Type	Name	Description
<code>__addr32 __mem __align16 mem_lock256_t*</code>	<code>mem_lock</code>	Pointer to the memory lock container in MEM
<code>uint32_t</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

### 3.18.4.20 mem\_unlock256\_ptr40

**Prototype:**

```
void mem_unlock256_ptr40(__addr40 __mem __align16 mem_lock256_t* mem_lock, uint32_t 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 3.476. mem\_unlock256\_ptr40 parameters**

Type	Name	Description
<code>__addr40 __mem __align16 mem_lock256_t*</code>	<code>mem_lock</code>	Pointer to the memory lock container in MEM
<code>uint32_t</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

### 3.18.4.21 mem\_unlock384\_ptr32

**Prototype:**

```
void mem_unlock384_ptr32(__addr32 __mem __align16 mem_lock384_t* mem_lock, uint32_t 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 3.477. mem\_unlock384\_ptr32 parameters**

Type	Name	Description
<code>__addr32 __mem __align16 mem_lock384_t*</code>	<code>mem_lock</code>	Pointer to the memory lock container in MEM
<code>uint32_t</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

### 3.18.4.22 mem\_unlock384\_ptr40

**Prototype:**

```
void mem_unlock384_ptr40(__addr40 __mem __align16 mem_lock384_t* mem_lock, uint32_t 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 3.478. mem\_unlock384\_ptr40 parameters**

Type	Name	Description
<code>__addr40 __mem __align16 mem_lock384_t*</code>	<code>mem_lock</code>	Pointer to the memory lock container in MEM
<code>uint32_t</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

### 3.18.4.23 mem\_unlock512\_ptr32

**Prototype:**

```
void mem_unlock512_ptr32(__addr32 __mem __align16 mem_lock512_t* mem_lock, uint32_t 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 3.479. mem\_unlock512\_ptr32 parameters**

Type	Name	Description
<code>__addr32 __mem __align16 mem_lock512_t*</code>	<code>mem_lock</code>	Pointer to the memory lock container in MEM
<code>uint32_t</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

### 3.18.4.24 mem\_unlock512\_ptr40

**Prototype:**

```
void mem_unlock512_ptr40(__addr40 __mem __align16 mem_lock512_t* mem_lock, uint32_t 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 3.480. mem\_unlock512\_ptr40 parameters**

Type	Name	Description
<code>__addr40 __mem __align16 mem_lock512_t*</code>	<code>mem_lock</code>	Pointer to the memory lock container in MEM
<code>uint32_t</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 3.481. union mem\_list\_desc\_t0**

Type	Name	Description
<code>uint32_t</code>	<code>seg_cnt:6</code>	Segment count.
<code>uint32_t</code>	<code>head_ptr:24</code>	Word address of first buffer.
<code>uint32_t</code>	<code>eop:1</code>	End Of Packet indication.

Type	Name	Description
uint32_t	sop:1	Start Of Packet indication.
uint32_t	tail_ptr:30	Word address of last(tail) buffer of the list.
uint32_t	q_type:2	Set to TYPE_0.
uint32_t	q_loc:2	Locality specification of list.
uint32_t	reserved_1:4	Reserved.
uint32_t	q_page:2	Page of list storage.
uint32_t	q_count:24	Number of entries in list.
uint32_t	reserved_2	Reserved.
uint32_t	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 3.482. union mem\_list\_desc\_t1**

Type	Name	Description
uint32_t	user_data:6	6 - bits of user data.
uint32_t	head_ptr:24	Word address of first buffer.
uint32_t	eop:1	End Of Packet indication.
uint32_t	sop:1	Start Of Packet indication.
uint32_t	tail_ptr:30	Word address of last(tail) buffer of the list.
uint32_t	q_type:2	Set to TYPE_1.
uint32_t	q_loc:2	Locality specification of list.
uint32_t	reserved_1:4	Reserved.
uint32_t	q_page:2	Page of list storage.
uint32_t	q_count:24	Number of entries in list.
uint32_t	reserved_2	Reserved.
uint32_t	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 3.483. union mem\_list\_desc\_t2**

Type	Name	Description
uint32_t	head_ptr:30	Word address of first buffer.
uint32_t	eop:1	End Of Packet indication.
uint32_t	sop:1	Start Of Packet indication.
uint32_t	tail_ptr:30	Word address of last(tail) buffer of the list.
uint32_t	q_type:2	Set to TYPE_2.
uint32_t	q_loc:2	Locality specification of list.
uint32_t	reserved_1:4	Reserved.
uint32_t	q_page:2	Page of list storage.
uint32_t	q_count:24	Number of entries in list.
uint32_t	reserved_2	Reserved.
uint32_t	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 3.484. union mem\_list\_desc\_t3**

Type	Name	Description
uint32_t	seg_cnt:6	Segment count.
uint32_t	head_ptr:24	Word address of first buffer.
uint32_t	eop:1	End Of Packet indication.
uint32_t	sop:1	Start Of Packet indication.
uint32_t	tail_ptr:30	Word address of last(tail) buffer of the list.
uint32_t	q_type:2	Set to TYPE_3.
uint32_t	q_loc:2	Locality specification of list.
uint32_t	reserved_1:4	Reserved.
uint32_t	q_page:2	Page of list storage.
uint32_t	q_count:24	Number of entries in list.
uint32_t	reserved_2	Reserved.
uint32_t	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 3.485. union mem\_list\_link\_mem\_t0**

Type	Name	Description
uint32_t	eop:1	End Of Packet indication.
uint32_t	sop:1	Start Of Packet indication.
uint32_t	seg_cnt:6	Segment count.
uint32_t	next_ptr:24	Pointer to next list entry.
uint32_t	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 3.486. union mem\_list\_link\_mem\_t1**

Type	Name	Description
uint32_t	eop:1	End Of Packet indication.
uint32_t	sop:1	Start Of Packet indication.
uint32_t	user_data:6	6 - bits of user data.
uint32_t	next_ptr:24	Word address of next list entry.
uint32_t	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 3.487. union mem\_list\_link\_mem\_t2**

Type	Name	Description
uint32_t	next_ptr:30	Word address of next list entry.
uint32_t	eop:1	End Of Packet indication.
uint32_t	sop:1	Start Of Packet indication.
uint32_t	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 3.488. union mem\_list\_link\_mem\_t3**

Type	Name	Description
uint32_t	eop:1	End Of Packet indication.
uint32_t	sop:1	Start Of Packet indication.
uint32_t	seg_cnt:6	Segment count.
uint32_t	next_ptr:24	Word address of next list entry.
uint32_t	value	Accessor to entire structure.

### 3.19.1.9 mem\_list\_link\_t0

List type 0 link type.

**Table 3.489. union mem\_list\_link\_t0**

Type	Name	Description
uint32_t	seg_cnt:6	Segment count.
uint32_t	next_ptr:24	Pointer to next list entry.
uint32_t	eop:1	End Of Packet indication.
uint32_t	sop:1	Start Of Packet indication.
uint32_t	value	Accessor to entire structure.

### 3.19.1.10 mem\_list\_link\_t1

List type 1 link type.

**Table 3.490. union mem\_list\_link\_t1**

Type	Name	Description
uint32_t	user_data:6	6 - bits of user data.
uint32_t	next_ptr:24	Word address of next list entry.
uint32_t	eop:1	End Of Packet indication.
uint32_t	sop:1	Start Of Packet indication.
uint32_t	value	Accessor to entire structure.

### 3.19.1.11 mem\_list\_link\_t2

List type 2 link type.

**Table 3.491. union mem\_list\_link\_t2**

Type	Name	Description
uint32_t	next_ptr:30	Word address of next list entry.

Type	Name	Description
uint32_t	eop:1	End Of Packet indication.
uint32_t	sop:1	Start Of Packet indication.
uint32_t	value	Accessor to entire structure.

### 3.19.1.12 mem\_list\_link\_t3

List type 3 link type.

**Table 3.492. union mem\_list\_link\_t3**

Type	Name	Description
uint32_t	seg_cnt:6	Segment count.
uint32_t	next_ptr:24	Word address of next list entry.
uint32_t	eop:1	End Of Packet indication.
uint32_t	sop:1	Start Of Packet indication.
uint32_t	value	Accessor to entire structure.

## 3.19.2 MU List Typedefs

### 3.19.2.1 MEM\_LIST\_TYPE

Available list types.

**Table 3.493. 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 3.494. 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 3.495. `typedef mem_list_link_t0`**

Type	Definition
<code>mem_list_link_t0</code>	<code>union mem_list_link_t0</code>

### 3.19.2.4 `mem_list_link_mem_t0`

List type 0 link descriptor in memory.

**Table 3.496. `typedef mem_list_link_mem_t0`**

Type	Definition
<code>mem_list_link_mem_t0</code>	<code>union mem_list_link_mem_t0</code>

### 3.19.2.5 `mem_list_desc_in_mem_t0`

List type 0 descriptor aligned in memory.

**Table 3.497. `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 3.498. `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 3.499. `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 3.500. `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 3.501. `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 3.502. `typedef mem_list_desc_in_mem_t1`**

Type	Definition
<code>mem_list_desc_in_mem_t1</code>	<code>__addr40 __align16 mem_list_desc_t1</code>

### 3.19.2.11 `mem_list_link_in_mem_t1`

List type 1 link aligned in memory.

**Table 3.503. `typedef mem_list_link_in_mem_t1`**

Type	Definition
<code>mem_list_link_in_mem_t1</code>	<code>__addr40 __align16 mem_list_link_mem_t1</code>

### 3.19.2.12 `mem_list_desc_t2`

List type 2 descriptor type.

32-bit buffer based linked list counting buffers.

**Table 3.504. `typedef mem_list_desc_t2`**

Type	Definition
<code>mem_list_desc_t2</code>	<code>union mem_list_desc_t2</code>

### 3.19.2.13 mem\_list\_link\_t2

List type 2 link type.

**Table 3.505. 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 3.506. typedef mem\_list\_link\_mem\_t2**

Type	Definition
mem_list_link_mem_t2	union mem_list_link_mem_t2

### 3.19.2.15 mem\_list\_desc\_in\_mem\_t2

List type 2 descriptor aligned in memory.

**Table 3.507. typedef mem\_list\_desc\_in\_mem\_t2**

Type	Definition
mem_list_desc_in_mem_t2	__addr40 __align16 mem_list_desc_t2

### 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 3.508. typedef mem\_list\_link\_in\_mem\_t2**

Type	Definition
mem_list_link_in_mem_t2	__addr40 __align16 mem_list_link_mem_t2

### 3.19.2.17 mem\_list\_desc\_t3

List Type 3 descriptor type.

24-bit buffer-based linked list counting buffers.

**Table 3.509. `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 3.510. `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 3.511. `typedef mem_list_link_mem_t3`**

Type	Definition
<code>mem_list_link_mem_t3</code>	<code>union mem_list_link_mem_t3</code>

### 3.19.2.20 `mem_list_desc_in_mem_t3`

List type 3 descriptor aligned in memory.

**Table 3.512. `typedef mem_list_desc_in_mem_t3`**

Type	Definition
<code>mem_list_desc_in_mem_t3</code>	<code>__addr40 __align16 mem_list_desc_t3</code>

### 3.19.2.21 `mem_list_link_in_mem_t3`

List type 3 link aligned in memory.

**Table 3.513. `typedef mem_list_link_in_mem_t3`**

Type	Definition
<code>mem_list_link_in_mem_t3</code>	<code>__addr40 __align16 mem_list_link_mem_t3</code>

### 3.19.3 MU List Functions

#### 3.19.3.1 cmd\_mem\_list\_read\_desc\_t0

**Prototype:**

```
void cmd_mem_list_read_desc_t0(uint32_t 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 3.514. cmd\_mem\_list\_read\_desc\_t0 parameters**

Type	Name	Description
uint32_t	<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"><li>• 0 - 1023: emem0 or island 24,</li><li>• 1024 - 2047: emem1 or island 25,</li><li>• 2048 - 3071: emem2 or island 26</li></ul>
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 cmd\_mem\_list\_read\_desc\_t1

**Prototype:**

```
void cmd_mem_list_read_desc_t1(uint32_t 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 3.515. cmd\_mem\_list\_read\_desc\_t1 parameters**

Type	Name	Description
uint32_t	<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"><li>• 0 - 1023: emem0 or island 24,</li><li>• 1024 - 2047: emem1 or island 25,</li></ul>

Type	Name	Description
		<ul style="list-style-type: none"> <li>• 2048 - 3071: emem2 or island 26</li> </ul>
mem_list_desc_in_mem_t1*	mem_list_desc_in_mem	40 bit pointer to list descriptor in external memory including the MU island. The pointer must be 16 byte aligned.

### 3.19.3.3 cmd\_mem\_list\_read\_desc\_t2

**Prototype:**

```
void cmd_mem_list_read_desc_t2(uint32_t 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 3.516. cmd\_mem\_list\_read\_desc\_t2 parameters**

Type	Name	Description
uint32_t	q_array_number	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"> <li>• 0 - 1023: emem0 or island 24,</li> <li>• 1024 - 2047: emem1 or island 25,</li> <li>• 2048 - 3071: emem2 or island 26</li> </ul>
mem_list_desc_in_mem_t2*	mem_list_desc_in_mem	40 bit pointer to list descriptor in external memory including the MU island. The pointer must be 16 byte aligned.

### 3.19.3.4 cmd\_mem\_list\_read\_desc\_t3

**Prototype:**

```
void cmd_mem_list_read_desc_t3(uint32_t 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 3.517. cmd\_mem\_list\_read\_desc\_t3 parameters**

Type	Name	Description
uint32_t	<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"> <li>• 0 - 1023: emem0 or island 24,</li> <li>• 1024 - 2047: emem1 or island 25,</li> <li>• 2048 - 3071: emem2 or island 26</li> </ul>
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 cmd\_mem\_list\_push\_desc\_t0

**Prototype:**

```
void cmd_mem_list_push_desc_t0(uint32_t 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 3.518. cmd\_mem\_list\_push\_desc\_t0 parameters**

Type	Name	Description
uint32_t	<i>q_array_number</i>	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"> <li>• 0 - 1023: emem0 or island 24,</li> <li>• 1024 - 2047: emem1 or island 25,</li> <li>• 2048 - 3071: emem2 or island 26</li> </ul>
<code>__xread mem_list_desc_t0*</code>	<i>xfer</i>	Pointer to read xfer registers where the descriptor will be written
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

### 3.19.3.6 cmd\_mem\_list\_push\_desc\_t1

**Prototype:**

```
void cmd_mem_list_push_desc_t1(uint32_t 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 3.519. cmd\_mem\_list\_push\_desc\_t1 parameters**

Type	Name	Description
<code>uint32_t</code>	<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"> <li>• 0 - 1023: emem0 or island 24,</li> <li>• 1024 - 2047: emem1 or island 25,</li> <li>• 2048 - 3071: emem2 or island 26</li> </ul>
<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 cmd\_mem\_list\_push\_desc\_t2

**Prototype:**

```
void cmd_mem_list_push_desc_t2(uint32_t 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 3.520. cmd\_mem\_list\_push\_desc\_t2 parameters**

Type	Name	Description
<code>uint32_t</code>	<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"> <li>• 0 - 1023: emem0 or island 24,</li> <li>• 1024 - 2047: emem1 or island 25,</li> <li>• 2048 - 3071: emem2 or island 26</li> </ul>
<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 cmd\_mem\_list\_push\_desc\_t3

**Prototype:**

```
void cmd_mem_list_push_desc_t3(uint32_t 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 3.521. cmd\_mem\_list\_push\_desc\_t3 parameters**

Type	Name	Description
uint32_t	<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"> <li>• 0 - 1023: emem0 or island 24,</li> <li>• 1024 - 2047: emem1 or island 25,</li> <li>• 2048 - 3071: emem2 or island 26</li> </ul>
<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 cmd\_mem\_list\_write\_desc\_t0

**Prototype:**

```
void cmd_mem_list_write_desc_t0(uint32_t 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 3.522. cmd\_mem\_list\_write\_desc\_t0 parameters**

Type	Name	Description
uint32_t	<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"> <li>• 0 - 1023: emem0 or island 24,</li> <li>• 1024 - 2047: emem1 or island 25,</li> <li>• 2048 - 3071: emem2 or island 26</li> </ul>

Type	Name	Description
mem_list_desc_in_mem_t0*	<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

### 3.19.3.10 cmd\_mem\_list\_write\_desc\_t1

**Prototype:**

```
void cmd_mem_list_write_desc_t1(uint32_t 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 3.523. cmd\_mem\_list\_write\_desc\_t1 parameters**

Type	Name	Description
uint32_t	<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"> <li>• 0 - 1023: emem0 or island 24,</li> <li>• 1024 - 2047: emem1 or island 25,</li> <li>• 2048 - 3071: emem2 or island 26</li> </ul>
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 cmd\_mem\_list\_write\_desc\_t2

**Prototype:**

```
void cmd_mem_list_write_desc_t2(uint32_t 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 3.524. cmd\_mem\_list\_write\_desc\_t2 parameters**

Type	Name	Description
uint32_t	<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"><li>• 0 - 1023: emem0 or island 24,</li><li>• 1024 - 2047: emem1 or island 25,</li><li>• 2048 - 3071: emem2 or island 26</li></ul>
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 cmd\_mem\_list\_write\_desc\_t3

**Prototype:**

```
void cmd_mem_list_write_desc_t3(uint32_t 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 3.525. cmd\_mem\_list\_write\_desc\_t3 parameters**

Type	Name	Description
uint32_t	<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"><li>• 0 - 1023: emem0 or island 24,</li><li>• 1024 - 2047: emem1 or island 25,</li><li>• 2048 - 3071: emem2 or island 26</li></ul>
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 cmd\_mem\_list\_enqueue\_t0

**Prototype:**

```
void cmd_mem_list_enqueue_t0(uint32_t q_array_number, uint32_t sop, uint32_t eop, uint32_t
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 3.526. cmd\_mem\_list\_enqueue\_t0 parameters**

Type	Name	Description
uint32_t	<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"> <li>• 0 - 1023: emem0 or island 24,</li> <li>• 1024 - 2047: emem1 or island 25,</li> <li>• 2048 - 3071: emem2 or island 26</li> </ul>
uint32_t	<i>sop</i>	start of packet indication
uint32_t	<i>eop</i>	end of packet indication
uint32_t	<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 cmd\_mem\_list\_enqueue\_t1

**Prototype:**

```
void cmd_mem_list_enqueue_t1(uint32_t q_array_number, uint32_t sop, uint32_t eop, uint32_t 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 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 3.527. cmd\_mem\_list\_enqueue\_t1 parameters**

Type	Name	Description
uint32_t	<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"> <li>• 0 - 1023: emem0 or island 24,</li> <li>• 1024 - 2047: emem1 or island 25,</li> <li>• 2048 - 3071: emem2 or island 26</li> </ul>
uint32_t	<i>sop</i>	start of packet indication
uint32_t	<i>eop</i>	end of packet indication
uint32_t	<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 cmd\_mem\_list\_enqueue\_t2

**Prototype:**

```
void cmd_mem_list_enqueue_t2(uint32_t q_array_number, uint32_t sop, uint32_t 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 3.528. cmd\_mem\_list\_enqueue\_t2 parameters**

Type	Name	Description
uint32_t	<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"> <li>• 0 - 1023: emem0 or island 24,</li> <li>• 1024 - 2047: emem1 or island 25,</li> <li>• 2048 - 3071: emem2 or island 26</li> </ul>
uint32_t	<i>sop</i>	start of packet indication
uint32_t	<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 cmd\_mem\_list\_enqueue\_t3

**Prototype:**

```
void cmd_mem_list_enqueue_t3(uint32_t q_array_number, uint32_t sop, uint32_t eop, uint32_t
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 3.529. cmd\_mem\_list\_enqueue\_t3 parameters**

Type	Name	Description
uint32_t	<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"> <li>• 0 - 1023: emem0 or island 24,</li> <li>• 1024 - 2047: emem1 or island 25,</li> <li>• 2048 - 3071: emem2 or island 26</li> </ul>
uint32_t	<i>sop</i>	start of packet indication
uint32_t	<i>eop</i>	end of packet indication
uint32_t	<i>seg_cnt</i>	segment count
mem_list_link_in_mem_t3*	<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.

### 3.19.3.17 cmd\_mem\_list\_enqueue\_tail\_t0

**Prototype:**

```
void cmd_mem_list_enqueue_tail_t0(uint32_t 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 3.530. cmd\_mem\_list\_enqueue\_tail\_t0 parameters**

Type	Name	Description
uint32_t	<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"> <li>• 0 - 1023: emem0 or island 24,</li> <li>• 1024 - 2047: emem1 or island 25,</li> <li>• 2048 - 3071: emem2 or island 26</li> </ul>
mem_list_link_in_mem_t0*	<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.

### 3.19.3.18 cmd\_mem\_list\_enqueue\_tail\_t1

**Prototype:**

```
void cmd_mem_list_enqueue_tail_t1(uint32_t 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 3.531. cmd\_mem\_list\_enqueue\_tail\_t1 parameters**

Type	Name	Description
uint32_t	<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"> <li>• 0 - 1023: emem0 or island 24,</li> <li>• 1024 - 2047: emem1 or island 25,</li> <li>• 2048 - 3071: emem2 or island 26</li> </ul>
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 cmd\_mem\_list\_enqueue\_tail\_t2

**Prototype:**

```
void cmd_mem_list_enqueue_tail_t2(uint32_t 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 3.532. cmd\_mem\_list\_enqueue\_tail\_t2 parameters**

Type	Name	Description
uint32_t	<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"> <li>• 0 - 1023: emem0 or island 24,</li> <li>• 1024 - 2047: emem1 or island 25,</li> <li>• 2048 - 3071: emem2 or island 26</li> </ul>
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 cmd\_mem\_list\_enqueue\_tail\_t3

**Prototype:**

```
void cmd_mem_list_enqueue_tail_t3(uint32_t 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 3.533. cmd\_mem\_list\_enqueue\_tail\_t3 parameters**

Type	Name	Description
uint32_t	<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"> <li>• 0 - 1023: emem0 or island 24,</li> <li>• 1024 - 2047: emem1 or island 25,</li> <li>• 2048 - 3071: emem2 or island 26</li> </ul>
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 cmd\_mem\_list\_dequeue\_t0

**Prototype:**

```
void cmd_mem_list_dequeue_t0(__xread mem_list_link_t0 *mem_list_link, uint32_t
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 3.534. cmd\_mem\_list\_dequeue\_t0 parameters**

Type	Name	Description
<code>__xread mem_list_link_t0*</code>	<code>mem_list_link</code>	Pointer to read xfer registers where dequeued entry will be written.
<code>uint32_t</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"> <li>• 0 - 1023: emem0 or island 24,</li> <li>• 1024 - 2047: emem1 or island 25,</li> <li>• 2048 - 3071: emem2 or island 26</li> </ul>
<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

### 3.19.3.22 cmd\_mem\_list\_dequeue\_t1

**Prototype:**

```
void cmd_mem_list_dequeue_t1(__xread mem_list_link_t1 *mem_list_link, uint32_t
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 3.535. cmd\_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>uint32_t</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"> <li>• 0 - 1023: emem0 or island 24,</li> <li>• 1024 - 2047: emem1 or island 25,</li> <li>• 2048 - 3071: emem2 or island 26</li> </ul>

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

### 3.19.3.23 cmd\_mem\_list\_dequeue\_t2

**Prototype:**

```
void cmd_mem_list_dequeue_t2(__xread mem_list_link_t2 *mem_list_link, uint32_t
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 3.536. cmd\_mem\_list\_dequeue\_t2 parameters**

Type	Name	Description
__xread mem_list_link_t2*	<i>mem_list_link</i>	Pointer to read xfer registers where dequeued entry will be written.
uint32_t	<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"> <li>• 0 - 1023: emem0 or island 24,</li> <li>• 1024 - 2047: emem1 or island 25,</li> <li>• 2048 - 3071: emem2 or island 26</li> </ul>
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

### 3.19.3.24 cmd\_mem\_list\_dequeue\_t3

**Prototype:**

```
void cmd_mem_list_dequeue_t3(__xread mem_list_link_t3 *mem_list_link, uint32_t
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 3.537. cmd\_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>uint32_t</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"> <li>• 0 - 1023: emem0 or island 24,</li> <li>• 1024 - 2047: emem1 or island 25,</li> <li>• 2048 - 3071: emem2 or island 26</li> </ul>
<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 cmd\_mem\_list\_init

**Prototype:**

```
void cmd_mem_list_init(uint32_t 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)
uint32_t                                     q_array_number = 1023;

cmd_mem_list_init(q_array_number,TYPE_3);

// enqueue the entry and verify list count and list type
{
    uint32_t      sop = 0;
    uint32_t      eop = 1;
    uint32_t      cell_count = 1;

    cmd_mem_list_enqueue_t3(q_array_number, sop, eop, cell_count, &buf1);
    cmd_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;

    cmd_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
    }

    cmd_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:

```

uint32_t emem_island = 1; // 0 = emem0, 1 = emem1, 2 = emem2
uint32_t q_array_number = emem_island << 10 | 1023; // list with q_array_nr 1023 on emem1

```

**Table 3.538. cmd\_mem\_list\_init parameters**

Type	Name	Description
uint32_t	<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"> <li>• 0 - 1023: emem0 or island 24,</li> <li>• 1024 - 2047: emem1 or island 25,</li> <li>• 2048 - 3071: emem2 or island 26</li> </ul>
enum MEM_LIST_TYPE	<i>type</i>	List type

### 3.19.3.26 cmd\_mem\_list\_init\_with\_loc\_and\_page

**Prototype:**

```
void cmd_mem_list_init_with_loc_and_page(uint32_t q_array_number, enum MEM_LIST_TYPE
type, uint32_t loc, uint32_t page)
```

**Description:**

Initialize a memory list specifying the page and locality of the list.

**Table 3.539. cmd\_mem\_list\_init\_with\_loc\_and\_page parameters**

Type	Name	Description
uint32_t	<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"> <li>• 0 - 1023: emem0 or island 24,</li> <li>• 1024 - 2047: emem1 or island 25,</li> <li>• 2048 - 3071: emem2 or island 26</li> </ul>
enum MEM_LIST_TYPE	<i>type</i>	List type
uint32_t	<i>loc</i>	Locality specification of the list. Please refer to the PRM for detail
uint32_t	<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 3.540. 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 3.541. struct mem\_microq128\_t**

Type	Name	Description
mem_microq_desc_t	desc	Micro queue descriptor.
uint32_t	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 3.542. struct mem\_microq256\_t**

Type	Name	Description
mem_microq_desc_t	desc	Micro queue descriptor.
uint32_t	q[ 7 ]	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 3.543. union mem\_microq\_desc\_t**

Type	Name	Description
uint32_t	reserved_1:4	Reserved.
uint32_t	event_source:12	Event source number to include in event.
uint32_t	reserved_2:6	Reserved.
uint32_t	generate_event:1	Generate event on under or overflow.
uint32_t	entry_size:1	One of MICROQ_ENTRY_SIZE.
uint32_t	overflow:1	Set on overflow.
uint32_t	underflow:1	Set on underflow.
uint32_t	reserved_3:1	Reserved.
uint32_t	locked:1	Used for queue lock queues.  When this bit is set, it indicates that the lock has been given out.
uint32_t	num_elements:4	0-3 or 0-6 or 0-14 depending on entry_size and if it is quarter or half cache line.
uint32_t	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 3.544. 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 3.545. 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 3.546. 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 3.547. 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 3.548. 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 3.549. typedef mem\_microq128\_ptr40\_t**

Type	Definition
mem_microq128_ptr40_t	volatile __addr40 __mem __align16 mem_microq128_t*

### 3.20.4.7 mem\_microq256\_in\_mem\_t

256-bit microqueue aligned in 32-bit addressed memory.

**Table 3.550. 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 3.551. typedef mem\_microq256\_ptr40\_t**

Type	Definition
mem_microq256_ptr40_t	volatile __addr40 __mem __align32 mem_microq256_t*

## 3.20.5 MU MicroQ Functions

### 3.20.5.1 cmd\_mem\_microq128\_put\_ptr32

**Prototype:**

```
void cmd_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 3.552. cmd\_mem\_microq128\_put\_ptr32 parameters**

Type	Name	Description
__mem mem_microq128_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 raise upon completion

### 3.20.5.2 cmd\_mem\_microq128\_put\_ptr40

**Prototype:**

```
void cmd_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 3.553. cmd\_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 cmd\_mem\_microq256\_put\_ptr32

**Prototype:**

```
void cmd_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 3.554. cmd\_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 cmd\_mem\_microq256\_put\_ptr40**

**Prototype:**

```
void cmd_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 3.555. cmd\_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 cmd\_mem\_microq128\_get\_ptr32**

**Prototype:**

```
void cmd_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 3.556. cmd\_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 cmd\_mem\_microq128\_get\_ptr40

**Prototype:**

```
void cmd_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 3.557. cmd\_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 cmd\_mem\_microq256\_get\_ptr32

**Prototype:**

```
void cmd_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 3.558. cmd\_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 cmd\_mem\_microq256\_get\_ptr40

**Prototype:**

```
void cmd_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 3.559. cmd\_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 cmd\_mem\_microq128\_pop\_ptr32

**Prototype:**

```
void cmd_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 3.560. cmd\_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 cmd\_mem\_microq128\_pop\_ptr40

**Prototype:**

```
void cmd_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 3.561. cmd\_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 cmd\_mem\_microq256\_pop\_ptr32

**Prototype:**

```
void cmd_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 3.562. cmd\_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 cmd\_mem\_microq256\_pop\_ptr40

**Prototype:**

```
void cmd_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 3.563. cmd\_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 cmd\_mem\_microq128\_init\_ptr32

**Prototype:**

```
void cmd_mem_microq128_init_ptr32(__mem mem_microq128_in_mem_t* microq, enum MICROQ_ENTRY_SIZE entry_size, uint32_t generate_event, uint32_t 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 3.564. cmd\_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
uint32_t	<i>generate_event</i>	Enable event on overflow or underflow
uint32_t	<i>event_source</i>	Event to present on event bus if generate_event is 1

### 3.20.5.14 cmd\_mem\_microq128\_init\_ptr40

#### Prototype:

```
void cmd_mem_microq128_init_ptr40(mem_microq128_ptr40_t microq, enum MICROQ_ENTRY_SIZE entry_size, uint32_t generate_event, uint32_t 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 uint32_t read_data;
__xwrite uint32_t write_data;
uint32_t counter;
int32_t i;
SIGNAL sig;
SIGNAL_PAIR sig_pair;

cmd_mem_microq128_init_ptr40(&mq128_ptr40, MICROQ_ENTRY_SIZE_16, 0, 0);

write_data = 0x10001000;
cmd_mem_microq128_put_ptr40(&mq128_ptr40, &write_data, ctx_swap, &sig);

write_data = 0x20002000;
cmd_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
}

cmd_mem_microq128_pop_ptr40(&mq128_ptr40, &read_data, sig_done, &sig_pair);
wait_for_all_single(&sig_pair.even);

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 3.565. cmd\_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
uint32_t	<i>generate_event</i>	Enable event on overflow or underflow
uint32_t	<i>event_source</i>	Event to present on event bus if generate_event is 1

### 3.20.5.15 cmd\_mem\_microq256\_init\_ptr32

**Prototype:**

```
void cmd_mem_microq256_init_ptr32(__mem mem_microq256_in_mem_t* microq, enum MICROQ_ENTRY_SIZE entry_size, uint32_t generate_event, uint32_t 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 3.566. cmd\_mem\_microq256\_init\_ptr32 parameters**

Type	Name	Description
__mem mem_microq256_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
uint32_t	<i>generate_event</i>	Enable event on overflow or underflow
uint32_t	<i>event_source</i>	Event to present on event bus if generate_event is 1

### 3.20.5.16 cmd\_mem\_microq256\_init\_ptr40

**Prototype:**

```
void cmd_mem_microq256_init_ptr40(mem_microq256_ptr40_t microq, enum MICROQ_ENTRY_SIZE entry_size, uint32_t generate_event, uint32_t 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 3.567. cmd\_mem\_microq256\_init\_ptr40 parameters**

Type	Name	Description
mem_microq256_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
uint32_t	<i>generate_event</i>	Enable event on overflow or underflow
uint32_t	<i>event_source</i>	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 3.568. struct mem\_cam128\_t**

Type	Name	Description
uint32_t	<i>value[4]</i>	Storage for the CAM data.

#### 3.21.1.2 mem\_cam256\_t

256-bit CAM container type.

**Table 3.569. struct mem\_cam256\_t**

Type	Name	Description
uint32_t	<i>value[8]</i>	Storage for the CAM data.

#### 3.21.1.3 mem\_cam384\_t

384-bit CAM container type.

**Table 3.570. struct mem\_cam384\_t**

Type	Name	Description
uint32_t	<i>value[12]</i>	Storage for the CAM data.

### 3.21.1.4 mem\_cam512\_t

512-bit CAM container type.

**Table 3.571. struct mem\_cam512\_t**

Type	Name	Description
uint32_t	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 3.572. struct mem\_cam\_lookup32\_in\_t**

Type	Name	Description
uint32_t	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 3.573. union mem\_cam\_lookup16\_add\_out\_t**

Type	Name	Description
uint32_t	reserved:24	Reserved.
uint32_t	added:1	When set, this field indicates that an entry was added.  <div style="border: 1px solid blue; padding: 5px; margin-top: 10px;">  <b>Note</b>            This field is also set on a CAM miss when it is full.         </div>
uint32_t	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;">  <b>Note</b>            All-bits in this field are set on a CAM miss when it is full.         </div>

Type	Name	Description
uint32_t	match_bitf	Bitfield of matching entries.
uint32_t	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 3.574. union mem\_cam\_lookup16\_in\_t**

Type	Name	Description
uint32_t	reserved1:16	Reserved.
uint32_t	search:16	16-bit value to search for in the CAM
uint32_t	padding	Additional word to ensure read data is same size as write data.
uint32_t	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 3.575. union mem\_cam\_lookup16\_out\_t**

Type	Name	Description
uint32_t	reserved:24	Reserved.
uint32_t	first_match:8	0xff - when not found
uint32_t	match_bitf	Bitfield of matching entries.
uint32_t	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 3.576. union mem\_cam\_lookup24\_add\_inc\_out\_t**

Type	Name	Description
uint32_t	match_bitf:16	Bits are set on matching entries.
uint32_t	count:8	Upper 8-bits of the matched CAM entry is returned in this field.

Type	Name	Description
		The upper 8-bits is incremented in memory after the CAM operation.
uint32_t	added:1	When set, this field indicates that an entry was added.  <div style="border: 1px solid blue; padding: 5px; margin-top: 10px;">  <b>Note</b>            This field is also set on a CAM miss when it is full.         </div>
uint32_t	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;">  <b>Note</b>            All-bits in this field are set on a CAM miss when it is full.         </div>
uint32_t	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 3.577. union mem\_cam\_lookup24\_add\_in\_t**

Type	Name	Description
uint32_t	high_byte:8	Value to set upper 8-bits in CAM to when entry is added.
uint32_t	search:24	24-bit value to search for in the CAM
uint32_t	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 3.578. union mem\_cam\_lookup24\_add\_out\_t**

Type	Name	Description
uint32_t	match_bitf:16	Bits are set on matching entries.
uint32_t	match_high_byte:8	Upper 8-bits of matched CAM entry is returned in this field.
uint32_t	added:1	When set, this field indicates that an entry was added.  <div style="border: 1px solid blue; padding: 5px; margin-top: 10px;">  <b>Note</b>            This field is also set on a CAM miss when it is full.         </div>
uint32_t	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;">  <b>Note</b>            All-bits in this field are set on a CAM miss when it is full.         </div>
uint32_t	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 3.579. union mem\_cam\_lookup24\_in\_t**

Type	Name	Description
uint32_t	reserved1:8	Reserved.
uint32_t	search:24	24-bit value to search for in the CAM
uint32_t	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 3.580. union mem\_cam\_lookup24\_out\_t**

Type	Name	Description
uint32_t	match_bitf:16	Bits are set on CAM entries matching.
uint32_t	match_high_byte:8	Upper 8-bits of matched CAM entry.
uint32_t	first_match:8	Entry number of first match lookup - 0xff when not found.
uint32_t	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 3.581. union mem\_cam\_lookup32\_add\_out\_t**

Type	Name	Description
uint32_t	match_bitf:16	Bits are set on matching entries.
uint32_t	match_high_byte:8	Upper 8-bits of the matched CAM entry is returned in this field.
uint32_t	added:1	When set, this field indicates that an entry was added.  <div style="border: 1px solid blue; padding: 5px; margin-top: 10px;">  <b>Note</b>             This field is also set on a CAM miss when it is full.         </div>
uint32_t	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;">  <b>Note</b>             All-bits in this field are set on a CAM miss when it is full.         </div>
uint32_t	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 3.582. union mem\_cam\_lookup32\_out\_t**

Type	Name	Description
uint32_t	match_bitf:16	Bits are set on matching entries.
uint32_t	match_high_byte:8	Upper 8-bits of the matched CAM entry is returned in this field.

Type	Name	Description
uint32_t	first_match:8	First CAM entry that matched or 0xff - when not found.
uint32_t	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 3.583. union mem\_cam\_lookup8\_add\_out\_t**

Type	Name	Description
uint32_t	reserved:24	Reserved.
uint32_t	added:1	When set, this field indicates that an entry was added.
 <b>Note</b> This field is also set on a CAM miss when it is full.		
uint32_t	first_match:7	First matched entry number, or on a CAM miss where the entry was added.
 <b>Note</b> All-bits in this field are set on a CAM miss when it is full.		
uint32_t	match_bitf	Bitfield of matching entries.
uint32_t	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 3.584. union mem\_cam\_lookup8\_in\_t**

Type	Name	Description
uint32_t	reserved1:24	Reserved.
uint32_t	search:8	8-bit value to search for in the CAM
uint32_t	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 3.585. union mem\_cam\_lookup8\_out\_t**

Type	Name	Description
uint32_t	match_bitf_low	Lower 32-bits of 64-bit field with ones indicating that a corresponding entry matched.
uint32_t	match_bitf_high	Upper 32-bits of 64-bit field with ones indicating that a corresponding entry matched.
uint32_t	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 3.586. 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 3.587. 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 3.588. 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 3.589. typedef mem\_cam512\_t**

Type	Definition
mem_cam512_t	struct mem_cam512_t

### 3.21.3.5 mem\_cam128\_in\_mem\_t

**Table 3.590. typedef mem\_cam128\_in\_mem\_t**

Type	Definition
mem_cam128_in_mem_t	<code>__addr32 __align16 mem_cam128_t</code>

### 3.21.3.6 mem\_cam256\_in\_mem\_t

**Table 3.591. typedef mem\_cam256\_in\_mem\_t**

Type	Definition
mem_cam256_in_mem_t	<code>__addr32 __align32 mem_cam256_t</code>

### 3.21.3.7 mem\_cam384\_in\_mem\_t

**Table 3.592. typedef mem\_cam384\_in\_mem\_t**

Type	Definition
mem_cam384_in_mem_t	<code>__addr32 __align64 mem_cam384_t</code>

### 3.21.3.8 mem\_cam512\_in\_mem\_t

**Table 3.593. typedef mem\_cam512\_in\_mem\_t**

Type	Definition
mem_cam512_in_mem_t	<code>__addr32 __align64 mem_cam512_t</code>

### 3.21.3.9 mem\_cam128\_ptr40\_t

**Table 3.594. typedef mem\_cam128\_ptr40\_t**

Type	Definition
mem_cam128_ptr40_t	<code>__addr40 __mem __align16 mem_cam128_t*</code>

### 3.21.3.10 mem\_cam256\_ptr40\_t

**Table 3.595. typedef mem\_cam256\_ptr40\_t**

Type	Definition
mem_cam256_ptr40_t	<code>__addr40 __mem __align32 mem_cam256_t *</code>

### 3.21.3.11 mem\_cam384\_ptr40\_t

**Table 3.596. typedef mem\_cam384\_ptr40\_t**

Type	Definition
mem_cam384_ptr40_t	<code>__addr40 __mem __align64 mem_cam384_t *</code>

### 3.21.3.12 mem\_cam512\_ptr40\_t

**Table 3.597. typedef mem\_cam512\_ptr40\_t**

Type	Definition
mem_cam512_ptr40_t	<code>__addr40 __mem __align64 mem_cam512_t *</code>

### 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 3.598. typedef mem\_cam\_lookup8\_in\_t**

Type	Definition
mem_cam_lookup8_in_t	<code>union mem_cam_lookup8_in_t</code>

### 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 3.599. typedef mem\_cam\_lookup8\_out\_t**

Type	Definition
mem_cam_lookup8_out_t	<code>union mem_cam_lookup8_out_t</code>

### 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 3.600. 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 3.601. 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 3.602. 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 3.603. 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 3.604. `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 3.605. `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 3.606. `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 3.607. `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 3.608. `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 3.609. `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 3.610. `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 3.611. `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 3.612. 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 3.613. 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 3.614. 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 3.615. 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 3.616. `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 3.617. `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 3.618. `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 3.619. `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 3.620. `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 3.621. 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 cmd\_mem\_cam128\_init\_ptr32

**Prototype:**

```
void cmd_mem_cam128_init_ptr32(__mem mem_cam128_in_mem_t* cam)
```

**Description:**

Initialize a 128-bit CAM in 32-bit addressed memory.

**Table 3.622. cmd\_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 cmd\_mem\_cam128\_init\_ptr40

**Prototype:**

```
void cmd_mem_cam128_init_ptr40(mem_cam128_ptr40_t cam)
```

**Description:**

Initialize a 128-bit CAM in 40-bit addressed MEM.

**Table 3.623. cmd\_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 cmd\_mem\_cam256\_init\_ptr32

**Prototype:**

```
void cmd_mem_cam256_init_ptr32(__mem mem_cam256_in_mem_t* cam)
```

**Description:**

Initialize a 256-bit CAM in in 32-bit addressed memory.

**Table 3.624. cmd\_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 cmd\_mem\_cam256\_init\_ptr40

**Prototype:**

```
void cmd_mem_cam256_init_ptr40(mem_cam256_ptr40_t cam)
```

**Description:**

Initialize a 256-bit CAM in 40-bit addressed MEM.

**Table 3.625. cmd\_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 cmd\_mem\_cam384\_init\_ptr32

**Prototype:**

```
void cmd_mem_cam384_init_ptr32(__mem mem_cam384_in_mem_t* cam)
```

**Description:**

Initialize a 384-bit CAM in in 32-bit addressed memory.

**Table 3.626. cmd\_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 cmd\_mem\_cam384\_init\_ptr40

**Prototype:**

```
void cmd_mem_cam384_init_ptr40(mem_cam384_ptr40_t cam)
```

**Description:**

Initialize a 384-bit CAM in 40-bit addressed MEM.

**Table 3.627. cmd\_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 cmd\_mem\_cam512\_init\_ptr32**

**Prototype:**

```
void cmd_mem_cam512_init_ptr32(__mem mem_cam512_in_mem_t* cam)
```

**Description:**

Initialize a 512-bit CAM in in 32-bit addressed memory.

**Table 3.628. cmd\_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 cmd\_mem\_cam512\_init\_ptr40**

**Prototype:**

```
void cmd_mem_cam512_init_ptr40(mem_cam512_ptr40_t cam)
```

**Description:**

Initialize a 512-bit CAM in in 40-bit addressed memory.

**Table 3.629. cmd\_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, uint32_t index, uint32_t 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 3.630. 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>uint32_t</code>	<code>index</code>	CAM entry to set (0 is the first entry)
<code>uint32_t</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, uint32_t index, uint32_t 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 3.631. 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>uint32_t</code>	<code>index</code>	CAM entry to set (0 is the first entry)
<code>uint32_t</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, uint32_t index, uint32_t 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 3.632. mem\_cam256\_set8\_ptr32 parameters**

Type	Name	Description
<code>__mem mem_cam256_in_mem_t*</code>	<code>cam</code>	Pointer to the CAM structure in MEM
<code>uint32_t</code>	<code>index</code>	CAM entry to set (0 is the first entry)
<code>uint32_t</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

### 3.21.4.12 mem\_cam256\_set8\_ptr40

**Prototype:**

```
void mem_cam256_set8_ptr40(mem_cam256_ptr40_t cam, uint32_t index, uint32_t 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 3.633. 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>uint32_t</code>	<code>index</code>	CAM entry to set (0 is the first entry)
<code>uint32_t</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

### 3.21.4.13 mem\_cam384\_set8\_ptr32

**Prototype:**

```
void mem_cam384_set8_ptr32(__mem mem_cam384_in_mem_t* cam, uint32_t index, uint32_t 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 3.634. mem\_cam384\_set8\_ptr32 parameters**

Type	Name	Description
__mem mem_cam384_in_mem_t*	cam	Pointer to the CAM structure in MEM
uint32_t	index	CAM entry to set (0 is the first entry)
uint32_t	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

### 3.21.4.14 mem\_cam384\_set8\_ptr40

**Prototype:**

```
void mem_cam384_set8_ptr40(mem_cam384_ptr40_t cam, uint32_t index, uint32_t 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 3.635. mem\_cam384\_set8\_ptr40 parameters**

Type	Name	Description
mem_cam384_ptr40_t	cam	Pointer to the CAM structure in MEM
uint32_t	index	CAM entry to set (0 is the first entry)

Type	Name	Description
uint32_t	<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, uint32_t index, uint32_t 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 3.636. mem\_cam512\_set8\_ptr32 parameters**

Type	Name	Description
__mem mem_cam512_in_mem_t*	<i>cam</i>	Pointer to the CAM structure in MEM
uint32_t	<i>index</i>	CAM entry to set (0 is the first entry)
uint32_t	<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, uint32_t index, uint32_t 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 3.637. mem\_cam512\_set8\_ptr40 parameters**

Type	Name	Description
mem_cam512_ptr40_t	<i>cam</i>	Pointer to the CAM structure in MEM
uint32_t	<i>index</i>	CAM entry to set (0 is the first entry)
uint32_t	<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, uint32_t index, uint32_t
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 3.638. mem\_cam128\_set16\_ptr32 parameters**

Type	Name	Description
__mem mem_cam128_in_mem_t*	<i>cam</i>	Pointer to the CAM structure in MEM
uint32_t	<i>index</i>	CAM entry to set (0 is the first entry)
uint32_t	<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, uint32_t index, uint32_t 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 3.639. mem\_cam128\_set16\_ptr40 parameters**

Type	Name	Description
mem_cam128_ptr40_t	<i>cam</i>	Pointer to the CAM structure in MEM
uint32_t	<i>index</i>	CAM entry to set (0 is the first entry)
uint32_t	<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, uint32_t index, uint32_t
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 3.640. mem\_cam256\_set16\_ptr32 parameters**

Type	Name	Description
__mem mem_cam256_in_mem_t*	<i>cam</i>	Pointer to the CAM structure in MEM
uint32_t	<i>index</i>	CAM entry to set (0 is the first entry)
uint32_t	<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, uint32_t index, uint32_t 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 3.641. mem\_cam256\_set16\_ptr40 parameters**

Type	Name	Description
mem_cam256_ptr40_t	<i>cam</i>	Pointer to the CAM structure in MEM
uint32_t	<i>index</i>	CAM entry to set (0 is the first entry)
uint32_t	<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, uint32_t index, uint32_t
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 3.642. mem\_cam384\_set16\_ptr32 parameters**

Type	Name	Description
__mem mem_cam384_in_mem_t*	<i>cam</i>	Pointer to the CAM structure in MEM
uint32_t	<i>index</i>	CAM entry to set (0 is the first entry)
uint32_t	<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, uint32_t index, uint32_t 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 3.643. mem\_cam384\_set16\_ptr40 parameters**

Type	Name	Description
mem_cam384_ptr40_t	<i>cam</i>	Pointer to the CAM structure in MEM
uint32_t	<i>index</i>	CAM entry to set (0 is the first entry)
uint32_t	<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, uint32_t index, uint32_t 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 3.644. mem\_cam512\_set16\_ptr32 parameters**

Type	Name	Description
__mem mem_cam512_in_mem_t*	<i>cam</i>	Pointer to the CAM structure in MEM
uint32_t	<i>index</i>	CAM entry to set (0 is the first entry)

Type	Name	Description
uint32_t	<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

### 3.21.4.24 mem\_cam512\_set16\_ptr40

**Prototype:**

```
void mem_cam512_set16_ptr40(mem_cam512_ptr40_t cam, uint32_t index, uint32_t 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 3.645. mem\_cam512\_set16\_ptr40 parameters**

Type	Name	Description
mem_cam512_ptr40_t	<i>cam</i>	Pointer to the CAM structure in MEM
uint32_t	<i>index</i>	CAM entry to set (0 is the first entry)
uint32_t	<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

### 3.21.4.25 mem\_cam128\_set24\_ptr32

**Prototype:**

```
void mem_cam128_set24_ptr32(__mem mem_cam128_in_mem_t* cam, uint32_t index, uint32_t 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 3.646. 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>uint32_t</code>	<code>index</code>	CAM entry to set (0 is the first entry)
<code>uint32_t</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_ptr32`

### 3.21.4.26 mem\_cam128\_set24\_ptr40

**Prototype:**

```
void mem_cam128_set24_ptr40(mem_cam128_ptr40_t cam, uint32_t index, uint32_t 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 3.647. 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>uint32_t</code>	<code>index</code>	CAM entry to set (0 is the first entry)
<code>uint32_t</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_ptr40`

### 3.21.4.27 mem\_cam256\_set24\_ptr32

**Prototype:**

```
void mem_cam256_set24_ptr32(__mem mem_cam256_in_mem_t* cam, uint32_t index, uint32_t 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 3.648. mem\_cam256\_set24\_ptr32 parameters**

Type	Name	Description
__mem mem_cam256_in_mem_t*	cam	Pointer to the CAM structure in MEM
uint32_t	index	CAM entry to set (0 is the first entry)
uint32_t	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, uint32_t index, uint32_t 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 3.649. mem\_cam256\_set24\_ptr40 parameters**

Type	Name	Description
mem_cam256_ptr40_t	cam	Pointer to the CAM structure in MEM
uint32_t	index	CAM entry to set (0 is the first entry)

Type	Name	Description
uint32_t	<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, uint32_t index, uint32_t
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 3.650. mem\_cam384\_set24\_ptr32 parameters**

Type	Name	Description
__mem mem_cam384_in_mem_t*	<i>cam</i>	Pointer to the CAM structure in MEM
uint32_t	<i>index</i>	CAM entry to set (0 is the first entry)
uint32_t	<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, uint32_t index, uint32_t 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 3.651. mem\_cam384\_set24\_ptr40 parameters**

Type	Name	Description
mem_cam384_ptr40_t	<i>cam</i>	Pointer to the CAM structure in MEM
uint32_t	<i>index</i>	CAM entry to set (0 is the first entry)
uint32_t	<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, uint32_t index, uint32_t
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 3.652. mem\_cam512\_set24\_ptr32 parameters**

Type	Name	Description
__mem mem_cam512_in_mem_t*	<i>cam</i>	Pointer to the CAM structure in MEM
uint32_t	<i>index</i>	CAM entry to set (0 is the first entry)
uint32_t	<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, uint32_t index, uint32_t 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 3.653. mem\_cam512\_set24\_ptr40 parameters**

Type	Name	Description
mem_cam512_ptr40_t	<i>cam</i>	Pointer to the CAM structure in MEM
uint32_t	<i>index</i>	CAM entry to set (0 is the first entry)
uint32_t	<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, uint32_t index, uint32_t
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 3.654. mem\_cam128\_set32\_ptr32 parameters**

Type	Name	Description
__mem mem_cam128_in_mem_t*	<i>cam</i>	Pointer to the CAM structure in MEM
uint32_t	<i>index</i>	CAM entry to set (0 is the first entry)
uint32_t	<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, uint32_t index, uint32_t 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 3.655. mem\_cam128\_set32\_ptr40 parameters**

Type	Name	Description
mem_cam128_ptr40_t	<i>cam</i>	Pointer to the CAM structure in MEM
uint32_t	<i>index</i>	CAM entry to set (0 is the first entry)
uint32_t	<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, uint32_t index, uint32_t
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 3.656. mem\_cam256\_set32\_ptr32 parameters**

Type	Name	Description
__mem mem_cam256_in_mem_t*	<i>cam</i>	Pointer to the CAM structure in MEM
uint32_t	<i>index</i>	CAM entry to set (0 is the first entry)
uint32_t	<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, uint32_t index, uint32_t 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 3.657. mem\_cam256\_set32\_ptr40 parameters**

Type	Name	Description
mem_cam256_ptr40_t	<i>cam</i>	Pointer to the CAM structure in MEM
uint32_t	<i>index</i>	CAM entry to set (0 is the first entry)
uint32_t	<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, uint32_t index, uint32_t 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 3.658. mem\_cam384\_set32\_ptr32 parameters**

Type	Name	Description
__mem mem_cam384_in_mem_t*	<i>cam</i>	Pointer to the CAM structure in MEM
uint32_t	<i>index</i>	CAM entry to set (0 is the first entry)

Type	Name	Description
uint32_t	<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

### 3.21.4.38 mem\_cam384\_set32\_ptr40

**Prototype:**

```
void mem_cam384_set32_ptr40(mem_cam384_ptr40_t cam, uint32_t index, uint32_t 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 3.659. mem\_cam384\_set32\_ptr40 parameters**

Type	Name	Description
mem_cam384_ptr40_t	<i>cam</i>	Pointer to the CAM structure in MEM
uint32_t	<i>index</i>	CAM entry to set (0 is the first entry)
uint32_t	<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

### 3.21.4.39 mem\_cam512\_set32\_ptr32

**Prototype:**

```
void mem_cam512_set32_ptr32(__mem mem_cam512_in_mem_t* cam, uint32_t index, uint32_t 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 3.660. mem\_cam512\_set32\_ptr32 parameters**

Type	Name	Description
__mem mem_cam512_in_mem_t*	<i>cam</i>	Pointer to the CAM structure in MEM
uint32_t	<i>index</i>	CAM entry to set (0 is the first entry)
uint32_t	<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

### 3.21.4.40 mem\_cam512\_set32\_ptr40

**Prototype:**

```
void mem_cam512_set32_ptr40(mem_cam512_ptr40_t cam, uint32_t index, uint32_t 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 3.661. mem\_cam512\_set32\_ptr40 parameters**

Type	Name	Description
mem_cam512_ptr40_t	<i>cam</i>	Pointer to the CAM structure in MEM
uint32_t	<i>index</i>	CAM entry to set (0 is the first entry)
uint32_t	<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

### 3.21.4.41 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 3.662. 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 3.663. 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 3.664. mem\_cam256\_lookup8\_ptr32 parameters**

Type	Name	Description
__mem mem_cam256_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

### 3.21.4.44 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 3.665. 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 3.666. 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 3.667. 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 3.668. 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 3.669. 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 3.670. 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 3.671. 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 3.672. 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 3.673. mem\_cam256\_lookup8\_add\_ptr40 parameters**

Type	Name	Description
mem_cam256_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_lookup8_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\_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 3.674. 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 3.675. 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 3.676. 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 3.677. 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 3.678. 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 3.679. 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</code>	<i>xfer</i>	Pointer to write xfer register containing the data to lookup
<code>mem_cam_lookup16_in_t*</code>		
<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 3.680. 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 3.681. 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 3.682. 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 3.683. 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 3.684. 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 3.685. 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 3.686. 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 3.687. 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 3.688. 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 3.689. 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 3.690. 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 3.691. 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 3.692. 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 3.693. 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 3.694. 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 3.695. 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 3.696. 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 3.697. 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 3.698. 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 3.699. 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 3.700. 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 3.701. mem\_cam512\_lookup24\_ptr40 parameters**

Type	Name	Description
<code>mem_cam512_ptr40_t</code>	<i>cam</i>	Pointer to the CAM structure in MEM
<code>__xwrite mem_cam_lookup24_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_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 3.702. 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 3.703. 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 3.704. 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 3.705. 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 3.706. 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 3.707. 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 3.708. 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 3.709. 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 3.710. 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 3.711. 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 3.712. mem\_cam256\_lookup24\_add\_inc\_ptr32 parameters**

Type	Name	Description
__mem mem_cam256_in_mem_t*	cam	Pointer to the CAM in MEM
__xwrite mem_cam_lookup24_add_inc_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\_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 3.713. mem\_cam256\_lookup24\_add\_inc\_ptr40 parameters**

Type	Name	Description
<code>mem_cam256_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`

### 3.21.4.93 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 3.714. mem\_cam384\_lookup24\_add\_inc\_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>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`

### 3.21.4.94 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 3.715. mem\_cam384\_lookup24\_add\_inc\_ptr40 parameters**

Type	Name	Description
<code>mem_cam384_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`

### 3.21.4.95 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 3.716. 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 3.717. 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 3.718. 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 3.719. 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 3.720. 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 3.721. 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 3.722. 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 3.723. 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 3.724. 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 3.725. 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 3.726. 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 3.727. 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 3.728. 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 3.729. 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 3.730. mem\_cam384\_lookup32\_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 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`

### 3.21.4.110 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 3.731. mem\_cam384\_lookup32\_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_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`

### 3.21.4.111 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 3.732. 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 3.733. mem\_cam512\_lookup32\_add\_ptr40 parameters**

Type	Name	Description
mem_cam512_ptr40_t	cam	Pointer to the CAM 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 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 3.734. MU TCAM Defines**

Defined	Definition
mem_tcam128_set24_word_ptr32	mem_tcam128_set32_word_ptr32 Alias for mem_tcam128_set32_word_ptr32.

Defined	Definition
	 <b>Note</b> 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 <b>not</b> compared, but will be returned for the first match.
<code>mem_tcaml28_set24_entry_value_ptr32</code>	<code>mem_tcaml28_set32_entry_value_ptr32</code> Alias for <code>mem_tcaml28_set32_entry_value_ptr32</code> .  <b>Note</b> 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 <b>not</b> compared, but will be returned for the first match.
<code>mem_tcaml28_set24_entry_value_ptr40</code>	<code>mem_tcaml28_set32_entry_value_ptr40</code>
<code>mem_tcaml28_set24_entry_mask_ptr32</code>	<code>mem_tcaml28_set32_entry_mask_ptr32</code> Alias for <code>mem_tcaml28_set32_entry_mask_ptr32</code> .  <b>Note</b> 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 <b>not</b> compared, but will be returned for the first match.
<code>mem_tcaml28_set24_entry_mask_ptr40</code>	<code>mem_tcaml28_set32_entry_mask_ptr40</code>

Defined	Definition
mem_tcam128_set24_entry_ptr32	<p>mem_tcam128_set32_entry_ptr32</p> <p>Alias for <code>mem_tcam128_set32_entry_ptr32</code>.</p> <div style="border: 1px solid blue; padding: 10px;">  <b>Note</b>            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 <b>not</b> 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 <code>mem_tcam256_set32_word_ptr32</code>.</p> <div style="border: 1px solid blue; padding: 10px;">  <b>Note</b>            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 <b>not</b> 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_ptr32</p> <p>Alias for <code>mem_tcam256_set32_entry_value_ptr32</code>.</p> <div style="border: 1px solid blue; padding: 10px;">  <b>Note</b>            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><b>not</b> compared, but will be returned for the first match.</p>
mem_tcamb256_set24_entry_value_ptr40	mem_tcamb256_set32_entry_value_ptr40
mem_tcamb256_set24_entry_mask_ptr32	<p>mem_tcamb256_set32_entry_mask_ptr32</p> <p>Alias for <code>mem_tcamb256_set32_entry_mask_ptr32</code>.</p> <div style="border: 1px solid blue; padding: 10px;">  <p><b>Note</b></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 <b>not</b> compared, but will be returned for the first match.</p> </div>
mem_tcamb256_set24_entry_mask_ptr40	mem_tcamb256_set32_entry_mask_ptr40
mem_tcamb256_set24_entry_ptr32	<p>mem_tcamb256_set32_entry_ptr32</p> <p>Alias for <code>mem_tcamb256_set32_entry_ptr32</code>.</p> <div style="border: 1px solid blue; padding: 10px;">  <p><b>Note</b></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 <b>not</b> compared, but will be returned for the first match.</p> </div>
mem_tcamb256_set24_entry_ptr40	mem_tcamb256_set32_entry_ptr40
mem_tcamb384_set24_word_ptr32	<p>mem_tcamb384_set32_word_ptr32</p> <p>Alias for <code>mem_tcamb384_set32_word_ptr32</code>.</p>

Defined	Definition
	 <b>Note</b> 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 <b>not</b> compared, but will be returned for the first match.
<code>mem_tcaml384_set24_word_ptr40</code>	<code>mem_tcaml384_set32_word_ptr40</code>
<code>mem_tcaml384_set24_entry_value_ptr32</code>	<code>mem_tcaml384_set32_entry_value_ptr32</code> Alias for <code>mem_tcaml384_set32_entry_value_ptr32</code> .  <b>Note</b> 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 <b>not</b> compared, but will be returned for the first match.
<code>mem_tcaml384_set24_entry_value_ptr40</code>	<code>mem_tcaml384_set32_entry_value_ptr40</code>
<code>mem_tcaml384_set24_entry_mask_ptr32</code>	<code>mem_tcaml384_set32_entry_mask_ptr32</code> Alias for <code>mem_tcaml384_set32_entry_mask_ptr32</code> .  <b>Note</b> 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 <b>not</b> compared, but will be returned for the first match.

Defined	Definition
mem_tcam384_set24_entry_mask_ptr40	mem_tcam384_set32_entry_mask_ptr40
mem_tcam384_set24_entry_ptr32	mem_tcam384_set32_entry_ptr32  Alias for <code>mem_tcam384_set32_entry_ptr32</code> .
	 <b>Note</b> 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 <b>not</b> compared, but will be returned for the first match.
mem_tcam384_set24_entry_ptr40	mem_tcam384_set32_entry_ptr40
mem_tcam512_set24_word_ptr32	mem_tcam512_set32_word_ptr32  Alias for <code>mem_tcam512_set32_word_ptr32</code> .
	 <b>Note</b> 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 <b>not</b> compared, but will be returned for the first match.
mem_tcam512_set24_word_ptr40	mem_tcam512_set32_word_ptr40
mem_tcam512_set24_entry_value_ptr32	mem_tcam512_set32_entry_value_ptr32  Alias for <code>mem_tcam512_set32_entry_value_ptr32</code> .
	 <b>Note</b> 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

Defined	Definition
	<p>bit lookup is performed, the highest 8 bits are <b>not</b> compared, but will be returned for the first match.</p>
<code>mem_tcam512_set24_entry_value_ptr40</code>	<code>mem_tcam512_set32_entry_value_ptr40</code>
<code>mem_tcam512_set24_entry_mask_ptr32</code>	<code>mem_tcam512_set32_entry_mask_ptr32</code>  Alias for <code>mem_tcam512_set32_entry_mask_ptr32</code> .
	<p> <b>Note</b></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 <b>not</b> compared, but will be returned for the first match.</p>
<code>mem_tcam512_set24_entry_mask_ptr40</code>	<code>mem_tcam512_set32_entry_mask_ptr40</code>
<code>mem_tcam512_set24_entry_ptr32</code>	<code>mem_tcam512_set32_entry_ptr32</code>  Alias for <code>mem_tcam512_set32_entry_ptr32</code> .
	<p> <b>Note</b></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 <b>not</b> compared, but will be returned for the first match.</p>
<code>mem_tcam512_set24_entry_ptr40</code>	<code>mem_tcam512_set32_entry_ptr40</code>
<code>mem_tcam_word_to_entry_index24</code>	<code>mem_tcam_word_to_entry_index32</code>  Alias for <code>mem_tcam_word_to_entry_index32</code> .  <code>mem_tcam128_set24_word_ptr32</code>

Defined	Definition
<code>mem_tcam_entry_value_to_word_index24</code>	<code>mem_tcam_entry_value_to_word_index32</code>  Alias for <code>mem_tcam_entry_value_to_word_index32</code> .  <code>mem_tcam128_set24_word_ptr32</code>
<code>mem_tcam_entry_mask_to_word_index24</code>	<code>mem_tcam_entry_mask_to_word_index32</code>  Alias for <code>mem_tcam_entry_mask_to_word_index32</code> .  <code>mem_tcam128_set24_word_ptr32</code>

## 3.22.2 MU TCAM Structs

### 3.22.2.1 mem\_tcam128\_t

128 bit TCAM container type.

**Table 3.735. struct mem\_tcam128\_t**

Type	Name	Description
<code>uint32_t</code>	<code>value[4]</code>	Storage for the TCAM data.

### 3.22.2.2 mem\_tcam256\_t

256 bit TCAM container type.

**Table 3.736. struct mem\_tcam256\_t**

Type	Name	Description
<code>uint32_t</code>	<code>value[8]</code>	Storage for the TCAM data.

### 3.22.2.3 mem\_tcam384\_t

384 bit TCAM container type.

**Table 3.737. struct mem\_tcam384\_t**

Type	Name	Description
<code>uint32_t</code>	<code>value[12]</code>	Storage for the TCAM data.

### 3.22.2.4 mem\_tcam512\_t

512 bit TCAM container type.

**Table 3.738. struct mem\_tcam512\_t**

Type	Name	Description
uint32_t	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 3.739. struct mem\_tcam\_lookup32\_in\_t**

Type	Name	Description
uint32_t	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 3.740. union mem\_tcam\_lookup16\_in\_t**

Type	Name	Description
uint32_t	reserved1:16	Reserved.
uint32_t	search:16	16 bit value to search for in the TCAM.
uint32_t	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 3.741. union mem\_tcam\_lookup16\_out\_t**

Type	Name	Description
uint32_t	match_bitf:16	16 bit field with ones indicating that a corresponding entry matched.
uint32_t	reserved:8	Reserved.
uint32_t	first_match:8	First matched 16 bit word index.

Type	Name	Description
		Equal to 0xFF when no match was found
uint32_t	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 3.742. union mem\_tcam\_lookup24\_in\_t**

Type	Name	Description
uint32_t	reserved1:8	Reserved.
uint32_t	search:24	24 bit value to search for in the TCAM
uint32_t	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 3.743. union mem\_tcam\_lookup24\_out\_t**

Type	Name	Description
uint32_t	reserved:8	Reserved.
uint32_t	match_bitf:8	8 bit field with ones indicating that a corresponding entry matched
uint32_t	match_high_byte:8	Upper 8 bits of the matched TCAM entry is returned in this field.
uint32_t	first_match:8	First matched 32 bit word index.  Equal to 0xFF when no match was found
uint32_t	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 3.744. union mem\_tcam\_lookup32\_out\_t**

Type	Name	Description
uint32_t	reserved:8	Reserved.

Type	Name	Description
uint32_t	match_bitf:8	8 bit field with ones indicating that a corresponding entry matched
uint32_t	match_high_byte:8	Upper 8 bits of the matched TCAM entry is returned in this field.
uint32_t	first_match:8	First matched 32 bit word index.  Equal to 0xFF when no match was found
uint32_t	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 3.745. union mem\_tcam\_lookup8\_in\_t**

Type	Name	Description
uint32_t	reserved1:24	Reserved.
uint32_t	search:8	8 bit value to search for in the TCAM
uint32_t	padding	Additional word to ensure read data is same size as write data.
uint32_t	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 3.746. union mem\_tcam\_lookup8\_out\_t**

Type	Name	Description
uint32_t	reserved:24	Reserved.
uint32_t	first_match:8	First matched 8 bit word index.  0xFF when no match was found
uint32_t	match_bitf	32 bit field with ones indicating that a corresponding entry matched
uint32_t	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 3.747. 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 3.748. 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 3.749. 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 3.750. typedef mem\_tcam512\_t**

Type	Definition
mem_tcam512_t	struct mem_tcam512_t

### 3.22.4.5 mem\_tcam128\_in\_mem\_t

**Table 3.751. 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 3.752. **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 3.753. **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 3.754. **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 3.755. **typedef mem\_tcam128\_ptr40\_t**

Type	Definition
mem_tcam128_ptr40_t	<code>__addr40 __mem __align16 mem_tcam128_t*</code>

### 3.22.4.10 mem\_tcam256\_ptr40\_t

Table 3.756. **typedef mem\_tcam256\_ptr40\_t**

Type	Definition
mem_tcam256_ptr40_t	<code>__addr40 __mem __align32 mem_tcam256_t*</code>

### 3.22.4.11 mem\_tcam384\_ptr40\_t

Table 3.757. **typedef mem\_tcam384\_ptr40\_t**

Type	Definition
mem_tcam384_ptr40_t	<code>__addr40 __mem __align64 mem_tcam384_t*</code>

### 3.22.4.12 mem\_tcam512\_ptr40\_t

**Table 3.758. typedef mem\_tcam512\_ptr40\_t**

Type	Definition
mem_tcam512_ptr40_t	<code>__addr40 __mem __align64 mem_tcam512_t*</code>

### 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 3.759. typedef mem\_tcam\_lookup8\_in\_t**

Type	Definition
mem_tcam_lookup8_in_t	<code>union mem_tcam_lookup8_in_t</code>

### 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 3.760. typedef mem\_tcam\_lookup8\_out\_t**

Type	Definition
mem_tcam_lookup8_out_t	<code>union mem_tcam_lookup8_out_t</code>

### 3.22.4.15 mem\_tcam\_lookup8\_out\_in\_read\_reg\_t

Type for `mem_tcam_lookup8_out_t` in read registers.

**Table 3.761. typedef mem\_tcam\_lookup8\_out\_in\_read\_reg\_t**

Type	Definition
mem_tcam_lookup8_out_in_read_reg_t	<code>__xread mem_tcam_lookup8_out_t</code>

### 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 3.762. `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 3.763. `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 3.764. `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 3.765. `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 3.766. `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 3.767. 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 3.768. 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 3.769. 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 3.770. 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 cmd\_mem\_tcam128\_init\_ptr32

**Prototype:**

```
void cmd_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 3.771. cmd\_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 cmd\_mem\_tcam128\_init\_ptr40

**Prototype:**

```
void cmd_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 3.772. cmd\_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 cmd\_mem\_tcam256\_init\_ptr32

**Prototype:**

```
void cmd_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 3.773. cmd\_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 cmd\_mem\_tcam256\_init\_ptr40

**Prototype:**

```
void cmd_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 3.774. cmd\_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 cmd\_mem\_tcam384\_init\_ptr32

**Prototype:**

```
void cmd_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 3.775. cmd\_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 cmd\_mem\_tcam384\_init\_ptr40

**Prototype:**

```
void cmd_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 3.776. cmd\_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 cmd\_mem\_tcam512\_init\_ptr32

**Prototype:**

```
void cmd_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 3.777. cmd\_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 cmd\_mem\_tcam512\_init\_ptr40

**Prototype:**

```
void cmd_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 3.778. cmd\_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, uint32_t index, uint32_t
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 3.779. 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
uint32_t	<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)
uint32_t	<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, uint32_t index, uint32_t 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 3.780. mem\_tcam128\_set8\_word\_ptr40 parameters**

Type	Name	Description
mem_tcam128_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
uint32_t	<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)
uint32_t	<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, uint32_t
entry_index, uint32_t 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 3.781. 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
uint32_t	<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)
uint32_t	<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, uint32_t entry_index,
uint32_t 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 3.782. mem\_tcam128\_set8\_entry\_value\_ptr40 parameters**

Type	Name	Description
mem_tcam128_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
uint32_t	<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)
uint32_t	<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, uint32_t
entry_index, uint32_t 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 3.783. 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
uint32_t	<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)
uint32_t	<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, uint32_t entry_index,  
uint32_t 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 3.784. mem\_tcam128\_set8\_entry\_mask\_ptr40 parameters**

Type	Name	Description
mem_tcam128_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
uint32_t	<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)
uint32_t	<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, uint32_t entry_index,  
uint32_t value, uint32_t 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 3.785. 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
uint32_t	<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)
uint32_t	<i>value</i>	Value to set the TCAM entry value to
uint32_t	<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, uint32_t entry_index, uint32_t
value, uint32_t 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 3.786. mem\_tcam128\_set8\_entry\_ptr40 parameters**

Type	Name	Description
mem_tcam128_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
uint32_t	<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)
uint32_t	<i>value</i>	Value to set the TCAM entry value to
uint32_t	<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, uint32_t index, uint32_t
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 3.787. 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>uint32_t</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>uint32_t</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, uint32_t index, uint32_t 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 3.788. 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>uint32_t</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>uint32_t</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, uint32_t
entry_index, uint32_t 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 3.789. 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>uint32_t</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>uint32_t</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, uint32_t entry_index,
uint32_t 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 3.790. 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>uint32_t</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>uint32_t</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, uint32_t
entry_index, uint32_t 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 3.791. mem\_tcam256\_set8\_entry\_mask\_ptr32 parameters**

Type	Name	Description
<i>__mem mem_tcam256_in_mem_t*</i>	<i>tcam</i>	Pointer to the TCAM structure in MEM
uint32_t	<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)
uint32_t	<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, uint32_t entry_index,
uint32_t 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 3.792. mem\_tcam256\_set8\_entry\_mask\_ptr40 parameters**

Type	Name	Description
mem_tcam256_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
uint32_t	<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)
uint32_t	<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, uint32_t entry_index,
                                  uint32_t value, uint32_t 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 3.793. 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
uint32_t	<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)
uint32_t	<i>value</i>	Value to set the TCAM entry value to
uint32_t	<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, uint32_t entry_index, uint32_t
                                   value, uint32_t 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 3.794. mem\_tcam256\_set8\_entry\_ptr40 parameters**

Type	Name	Description
mem_tcam256_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
uint32_t	<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)
uint32_t	<i>value</i>	Value to set the TCAM entry value to
uint32_t	<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, uint32_t index, uint32_t
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 3.795. 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
uint32_t	<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)
uint32_t	<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, uint32_t index, uint32_t 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 3.796. mem\_tcam384\_set8\_word\_ptr40 parameters**

Type	Name	Description
mem_tcam384_ptr40_t	tcam	Pointer to the TCAM structure in MEM
uint32_t	index	TCAM word to set (0 is the first entry value, 3 is the first entry mask, 1 is the second entry value)
uint32_t	value	Value to set the TCAM word to
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

### 3.22.5.27 mem\_tcam384\_set8\_entry\_value\_ptr32

**Prototype:**

```
void mem_tcam384_set8_entry_value_ptr32(__mem mem_tcam384_in_mem_t* tcam, uint32_t  
entry_index, uint32_t 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 3.797. mem\_tcam384\_set8\_entry\_value\_ptr32 parameters**

Type	Name	Description
__mem mem_tcam384_in_mem_t*	tcam	Pointer to the TCAM structure in MEM

Type	Name	Description
uint32_t	<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)
uint32_t	<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, uint32_t entry_index,
                                         uint32_t 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 3.798. mem\_tcam384\_set8\_entry\_value\_ptr40 parameters**

Type	Name	Description
mem_tcam384_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
uint32_t	<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)
uint32_t	<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, uint32_t
                                         entry_index, uint32_t 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 3.799. 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>uint32_t</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>uint32_t</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, uint32_t entry_index,
uint32_t 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 3.800. 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>uint32_t</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>uint32_t</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, uint32_t entry_index,
uint32_t value, uint32_t 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 3.801. 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>uint32_t</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>uint32_t</code>	<code>value</code>	Value to set the TCAM entry value to
<code>uint32_t</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, uint32_t entry_index, uint32_t
value, uint32_t 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 3.802. 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>uint32_t</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>uint32_t</code>	<code>value</code>	Value to set the TCAM entry value to

Type	Name	Description
uint32_t	<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, uint32_t index, uint32_t
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 3.803. 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
uint32_t	<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)
uint32_t	<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, uint32_t index, uint32_t 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 3.804. mem\_tcam512\_set8\_word\_ptr40 parameters**

Type	Name	Description
mem_tcam512_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
uint32_t	<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)
uint32_t	<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, uint32_t
entry_index, uint32_t 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 3.805. 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
uint32_t	<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)
uint32_t	<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, uint32_t entry_index,
uint32_t 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 3.806. mem\_tcam512\_set8\_entry\_value\_ptr40 parameters**

Type	Name	Description
mem_tcam512_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
uint32_t	<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)
uint32_t	<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, uint32_t
entry_index, uint32_t 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 3.807. 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
uint32_t	<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)
uint32_t	<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, uint32_t entry_index,
uint32_t 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 3.808. mem\_tcam512\_set8\_entry\_mask\_ptr40 parameters**

Type	Name	Description
mem_tcam512_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
uint32_t	<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)
uint32_t	<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, uint32_t entry_index,
uint32_t value, uint32_t 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 3.809. 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>uint32_t</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>uint32_t</code>	<code>value</code>	Value to set the TCAM entry value to
<code>uint32_t</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, uint32_t entry_index, uint32_t
value, uint32_t 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 3.810. 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>uint32_t</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>uint32_t</code>	<code>value</code>	Value to set the TCAM entry value to
<code>uint32_t</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, uint32_t index,
uint32_t 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 3.811. 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>uint32_t</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>uint32_t</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, uint32_t index, uint32_t
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 3.812. 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>uint32_t</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>uint32_t</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, uint32_t
entry_index, uint32_t 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 3.813. mem\_tcam128\_set16\_entry\_value\_ptr32 parameters**

Type	Name	Description
__mem mem_tcam128_in_mem_t*	tcam	Pointer to the TCAM structure in MEM
uint32_t	entry_index	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
uint32_t	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, uint32_t entry_index,
uint32_t 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 3.814. 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
uint32_t	<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)
uint32_t	<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, uint32_t
entry_index, uint32_t 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 3.815. 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
uint32_t	<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)
uint32_t	<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 tc当地, uint32_t entry_index,
uint32_t 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 3.816. mem\_tcam128\_set16\_entry\_mask\_ptr40 parameters**

Type	Name	Description
mem_tcam128_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
uint32_t	<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)
uint32_t	<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, uint32_t entry_index,
uint32_t value, uint32_t 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 3.817. 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
uint32_t	<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)
uint32_t	<i>value</i>	Value to set the TCAM entry value to
uint32_t	<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, uint32_t entry_index, uint32_t value, uint32_t 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 3.818. mem\_tcam128\_set16\_entry\_ptr40 parameters**

Type	Name	Description
mem_tcam128_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
uint32_t	<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)
uint32_t	<i>value</i>	Value to set the TCAM entry value to
uint32_t	<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, uint32_t index, uint32_t 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 3.819. 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>uint32_t</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>uint32_t</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, uint32_t index, uint32_t
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 3.820. 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>uint32_t</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>uint32_t</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, uint32_t
entry_index, uint32_t 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 3.821. 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
uint32_t	<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)
uint32_t	<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, uint32_t entry_index,  
uint32_t 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 3.822. mem\_tcam256\_set16\_entry\_value\_ptr40 parameters**

Type	Name	Description
mem_tcam256_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
uint32_t	<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)
uint32_t	<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, uint32_t entry_index, uint32_t 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 3.823. mem\_tcam256\_set16\_entry\_mask\_ptr32 parameters**

Type	Name	Description
__mem mem_tcam256_in_mem_t*	tcam	Pointer to the TCAM structure in MEM
uint32_t	entry_index	Number of TCAM entry to set (0 is the first entry, 1 is the second entry, 2 is the third entry)
uint32_t	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

### 3.22.5.54 mem\_tcam256\_set16\_entry\_mask\_ptr40

**Prototype:**

```
void mem_tcam256_set16_entry_mask_ptr40(mem_tcam256_ptr40_t tcam, uint32_t entry_index, uint32_t 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 3.824. mem\_tcam256\_set16\_entry\_mask\_ptr40 parameters**

Type	Name	Description
mem_tcam256_ptr40_t	tcam	Pointer to the TCAM structure in MEM

Type	Name	Description
uint32_t	<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)
uint32_t	<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, uint32_t entry_index,
uint32_t value, uint32_t 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 3.825. 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
uint32_t	<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)
uint32_t	<i>value</i>	Value to set the TCAM entry value to
uint32_t	<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, uint32_t entry_index, uint32_t
value, uint32_t 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 3.826. mem\_tcam256\_set16\_entry\_ptr40 parameters**

Type	Name	Description
mem_tcam256_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
uint32_t	<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)
uint32_t	<i>value</i>	Value to set the TCAM entry value to
uint32_t	<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, uint32_t index,
uint32_t 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 3.827. 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
uint32_t	<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)
uint32_t	<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, uint32_t index, uint32_t 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 3.828. mem\_tcam384\_set16\_word\_ptr40 parameters**

Type	Name	Description
mem_tcam384_ptr40_t	tcam	Pointer to the TCAM structure in MEM
uint32_t	index	TCAM word to set (0 is the first entry value, 2 is the first entry mask, 1 is the second entry value)
uint32_t	value	Value to set the TCAM word to
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

### 3.22.5.59 mem\_tcam384\_set16\_entry\_value\_ptr32

**Prototype:**

```
void mem_tcam384_set16_entry_value_ptr32(__mem mem_tcam384_in_mem_t* tcam, uint32_t entry_index, uint32_t 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 3.829. mem\_tcam384\_set16\_entry\_value\_ptr32 parameters**

Type	Name	Description
__mem mem_tcam384_in_mem_t*	tcam	Pointer to the TCAM structure in MEM

Type	Name	Description
uint32_t	<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)
uint32_t	<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, uint32_t entry_index,
                                         uint32_t 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 3.830. mem\_tcam384\_set16\_entry\_value\_ptr40 parameters**

Type	Name	Description
mem_tcam384_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
uint32_t	<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)
uint32_t	<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, uint32_t entry_index,
                                         uint32_t 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 3.831. 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>uint32_t</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>uint32_t</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, uint32_t entry_index,
                                         uint32_t 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 3.832. 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>uint32_t</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>uint32_t</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, uint32_t entry_index,
uint32_t value, uint32_t 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 3.833. 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>uint32_t</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>uint32_t</code>	<code>value</code>	Value to set the TCAM entry value to
<code>uint32_t</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, uint32_t entry_index, uint32_t
value, uint32_t 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 3.834. 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>uint32_t</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>uint32_t</code>	<code>value</code>	Value to set the TCAM entry value to

Type	Name	Description
uint32_t	<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, uint32_t index,
uint32_t 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 3.835. 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
uint32_t	<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)
uint32_t	<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, uint32_t index, uint32_t
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 3.836. mem\_tcam512\_set16\_word\_ptr40 parameters**

Type	Name	Description
mem_tcam512_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
uint32_t	<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)
uint32_t	<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, uint32_t
entry_index, uint32_t 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 3.837. 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
uint32_t	<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)
uint32_t	<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, uint32_t entry_index,
uint32_t 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 3.838. mem\_tcam512\_set16\_entry\_value\_ptr40 parameters**

Type	Name	Description
mem_tcam512_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
uint32_t	<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)
uint32_t	<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, uint32_t
entry_index, uint32_t 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 3.839. 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
uint32_t	<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)
uint32_t	<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, uint32_t entry_index,
uint32_t 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 3.840. mem\_tcam512\_set16\_entry\_mask\_ptr40 parameters**

Type	Name	Description
mem_tcam512_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
uint32_t	<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)
uint32_t	<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, uint32_t entry_index,
uint32_t value, uint32_t 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 3.841. 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>uint32_t</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>uint32_t</code>	<code>value</code>	Value to set the TCAM entry value to
<code>uint32_t</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, uint32_t entry_index, uint32_t
value, uint32_t 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 3.842. 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>uint32_t</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>uint32_t</code>	<code>value</code>	Value to set the TCAM entry value to
<code>uint32_t</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, uint32_t index,
uint32_t 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 3.843. 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>uint32_t</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>uint32_t</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, uint32_t index, uint32_t
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 3.844. 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>uint32_t</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>uint32_t</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, uint32_t
entry_index, uint32_t 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 3.845. mem\_tcam128\_set32\_entry\_value\_ptr32 parameters**

Type	Name	Description
__mem mem_tcam128_in_mem_t*	tcam	Pointer to the TCAM structure in MEM
uint32_t	entry_index	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
uint32_t	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, uint32_t entry_index,
uint32_t 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 3.846. 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
uint32_t	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
uint32_t	<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, uint32_t
entry_index, uint32_t 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 3.847. 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
uint32_t	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
uint32_t	<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, uint32_t entry_index,
uint32_t 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 3.848. mem\_tcam128\_set32\_entry\_mask\_ptr40 parameters**

Type	Name	Description
mem_tcam128_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
uint32_t	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
uint32_t	<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, uint32_t entry_index,
uint32_t value, uint32_t 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 3.849. 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
uint32_t	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
uint32_t	<i>value</i>	Value to set the TCAM entry value to
uint32_t	<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, uint32_t entry_index, uint32_t value, uint32_t 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 3.850. mem\_tcam128\_set32\_entry\_ptr40 parameters**

Type	Name	Description
mem_tcam128_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
uint32_t	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
uint32_t	<i>value</i>	Value to set the TCAM entry value to
uint32_t	<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, uint32_t index,  
uint32_t 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 3.851. 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
uint32_t	<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)
uint32_t	<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, uint32_t index, uint32_t
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 3.852. mem\_tcam256\_set32\_word\_ptr40 parameters**

Type	Name	Description
mem_tcam256_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
uint32_t	<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)
uint32_t	<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, uint32_t
entry_index, uint32_t 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 3.853. 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>uint32_t</code>	<code>entry_index</code>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
<code>uint32_t</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, uint32_t entry_index,
                                          uint32_t 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 3.854. 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>uint32_t</code>	<code>entry_index</code>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
<code>uint32_t</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, uint32_t
                                         entry_index, uint32_t 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 3.855. 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>uint32_t</code>	<code>entry_index</code>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
<code>uint32_t</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, uint32_t entry_index,
                                         uint32_t 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 3.856. 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>uint32_t</code>	<code>entry_index</code>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
<code>uint32_t</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, uint32_t entry_index,
uint32_t value, uint32_t 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 3.857. 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>uint32_t</code>	<code>entry_index</code>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
<code>uint32_t</code>	<code>value</code>	Value to set the TCAM entry value to
<code>uint32_t</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, uint32_t entry_index, uint32_t
value, uint32_t 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 3.858. 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>uint32_t</code>	<code>entry_index</code>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
<code>uint32_t</code>	<code>value</code>	Value to set the TCAM entry value to
<code>uint32_t</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, uint32_t index,
uint32_t 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 3.859. 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
uint32_t	<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)
uint32_t	<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, uint32_t index, uint32_t
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 3.860. mem\_tcam384\_set32\_word\_ptr40 parameters**

Type	Name	Description
mem_tcam384_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
uint32_t	<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)
uint32_t	<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, uint32_t
entry_index, uint32_t 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 3.861. 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
uint32_t	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
uint32_t	<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, uint32_t entry_index,
uint32_t 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 3.862. mem\_tcam384\_set32\_entry\_value\_ptr40 parameters**

Type	Name	Description
mem_tcam384_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
uint32_t	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
uint32_t	<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, uint32_t
entry_index, uint32_t 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 3.863. 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
uint32_t	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
uint32_t	<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, uint32_t entry_index,
uint32_t 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 3.864. mem\_tcam384\_set32\_entry\_mask\_ptr40 parameters**

Type	Name	Description
mem_tcam384_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
uint32_t	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
uint32_t	<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

### 3.22.5.95 mem\_tcam384\_set32\_entry\_ptr32

**Prototype:**

```
void mem_tcam384_set32_entry_ptr32(__mem mem_tcam384_in_mem_t* tcam, uint32_t entry_index,
uint32_t value, uint32_t 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 3.865. mem\_tcam384\_set32\_entry\_ptr32 parameters**

Type	Name	Description
<u>mem</u> mem_tcam384_in_mem_t*	<i>tcam</i>	Pointer to the TCAM structure in MEM
uint32_t	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
uint32_t	<i>value</i>	Value to set the TCAM entry value to
uint32_t	<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

### 3.22.5.96 mem\_tcam384\_set32\_entry\_ptr40

**Prototype:**

```
void mem_tcam384_set32_entry_ptr40(mem_tcam384_ptr40_t tcam, uint32_t entry_index, uint32_t value, uint32_t 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 3.866. mem\_tcam384\_set32\_entry\_ptr40 parameters**

Type	Name	Description
mem_tcam384_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
uint32_t	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
uint32_t	<i>value</i>	Value to set the TCAM entry value to
uint32_t	<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, uint32_t index, uint32_t 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 3.867. 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>uint32_t</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>uint32_t</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, uint32_t index, uint32_t
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 3.868. 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>uint32_t</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>uint32_t</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, uint32_t
entry_index, uint32_t 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 3.869. 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
uint32_t	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
uint32_t	<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, uint32_t entry_index,
                                          uint32_t 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 3.870. mem\_tcam512\_set32\_entry\_value\_ptr40 parameters**

Type	Name	Description
mem_tcam512_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
uint32_t	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
uint32_t	<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, uint32_t
entry_index, uint32_t 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 3.871. 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>uint32_t</code>	<code>entry_index</code>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
<code>uint32_t</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, uint32_t entry_index,
uint32_t 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 3.872. 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>uint32_t</code>	<code>entry_index</code>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
<code>uint32_t</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, uint32_t entry_index,  
uint32_t value, uint32_t 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 3.873. mem\_tcam512\_set32\_entry\_ptr32 parameters**

Type	Name	Description
__mem mem_tcam512_in_mem_t*	tcam	Pointer to the TCAM structure in MEM
uint32_t	entry_index	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
uint32_t	value	Value to set the TCAM entry value to
uint32_t	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, uint32_t entry_index, uint32_t  
value, uint32_t 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 3.874. mem\_tcam512\_set32\_entry\_ptr40 parameters**

Type	Name	Description
mem_tcam512_ptr40_t	<i>tcam</i>	Pointer to the TCAM structure in MEM
uint32_t	<i>entry_index</i>	Number of TCAM entry to set (0 is the first entry, 1 is the second entry)
uint32_t	<i>value</i>	Value to set the TCAM entry value to
uint32_t	<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 3.875. 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 3.876. 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 3.877. 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 3.878. 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 3.879. 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 3.880. mem\_tcam384\_lookup8\_ptr40 parameters**

Type	Name	Description
mem_tcam384_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

### 3.22.5.111 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 3.881. 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 3.882. 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 3.883. 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 3.884. 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 3.885. 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 3.886. 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 3.887. 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 3.888. 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 3.889. mem\_tcam512\_lookup16\_ptr32 parameters**

Type	Name	Description
__mem mem_tcam512_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

### 3.22.5.120 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 3.890. 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 3.891. mem\_tcam128\_lookup24\_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_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 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 3.892. mem\_tcam128\_lookup24\_ptr40 parameters**

Type	Name	Description
<code>mem_tcam128_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 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\_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 3.893. 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 3.894. 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 3.895. 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 3.896. 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 3.897. 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 3.898. 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 3.899. 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 3.900. 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 3.901. 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 3.902. 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 3.903. 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 3.904. 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 3.905. 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 3.906. 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:**

```
uint32_t mem_tcam_word_to_entry_index8(uint32_t word_idx)
```

**Description:**

Convert a TCAM 8 bit word index to an entry index.



### Note

This function does not perform range checking.

**Table 3.907. mem\_tcam\_word\_to\_entry\_index8 parameters**

Type	Name	Description
uint32_t	<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:**

```
uint32_t mem_tcam_entry_value_to_word_index8(uint32_t entry_idx)
```

**Description:**

Convert a TCAM 8 bit entry value index to a word index.



### Note

This function does not perform range checking.

**Table 3.908. mem\_tcam\_entry\_value\_to\_word\_index8 parameters**

Type	Name	Description
uint32_t	<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:**

```
uint32_t mem_tcam_entry_mask_to_word_index8(uint32_t entry_idx)
```

**Description:**

Convert a TCAM 8 bit entry mask index to a word index.



### Note

This function does not perform range checking.

**Table 3.909. mem\_tcam\_entry\_mask\_to\_word\_index8 parameters**

Type	Name	Description
uint32_t	<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:**

```
uint32_t mem_tcam_word_to_entry_index16(uint32_t word_idx)
```

**Description:**

Convert a TCAM 16 bit word index to an entry index.



### Note

This function does not perform range checking.

**Table 3.910. mem\_tcam\_word\_to\_entry\_index16 parameters**

Type	Name	Description
uint32_t	<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:**

```
uint32_t mem_tcam_entry_value_to_word_index16(uint32_t entry_idx)
```

**Description:**

Convert a TCAM 16 bit entry value index to a word index.



**Note**

This function does not perform range checking.

**Table 3.911. mem\_tcam\_entry\_value\_to\_word\_index16 parameters**

Type	Name	Description
uint32_t	<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:**

```
uint32_t mem_tcam_entry_mask_to_word_index16(uint32_t entry_idx)
```

**Description:**

Convert a TCAM 16 bit entry mask index to a word index.



**Note**

This function does not perform range checking.

**Table 3.912. mem\_tcam\_entry\_mask\_to\_word\_index16 parameters**

Type	Name	Description
uint32_t	<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:**

```
uint32_t mem_tcam_word_to_entry_index32(uint32_t word_idx)
```

**Description:**

Convert a TCAM 32 bit word index to an entry index.



### Note

This function does not perform range checking.

**Table 3.913. mem\_tcam\_word\_to\_entry\_index32 parameters**

Type	Name	Description
uint32_t	<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:**

```
uint32_t mem_tcam_entry_value_to_word_index32(uint32_t entry_idx)
```

**Description:**

Convert a TCAM 32 bit entry value index to a word index.



### Note

This function does not perform range checking.

**Table 3.914. mem\_tcam\_entry\_value\_to\_word\_index32 parameters**

Type	Name	Description
uint32_t	<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:**

```
uint32_t mem_tcam_entry_mask_to_word_index32(uint32_t entry_idx)
```

**Description:**

Convert a TCAM 32 bit entry mask index to a word index.



**Note**

This function does not perform range checking.

**Table 3.915. mem\_tcamb\_entry\_mask\_to\_word\_index32 parameters**

Type	Name	Description
uint32_t	<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 3.916. 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 3.917. 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 3.918. 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 3.919. 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 3.920. struct mem\_packet\_header\_t**

Type	Name	Description
uint32_t	ctm_number:6	CTM that contains the packet.
uint32_t	packet_number:10	Packet number used to address the packet in CTM.
uint32_t	buffer_list_queue:2	Buffer list queue associated with the MU pointer.
uint32_t	packet_length:14	Packet length in bytes.
uint32_t	value	Accessor to word_1 structure. Accessor to word_2 structure.
union mem_packet_header_t::@122	word_1	First word.
uint32_t	split:1	Indicates if the packet is split between CTM and a memory buffer.
uint32_t	reserved:2	Reserved.
uint32_t	mu_pointer:29	MU pointer to the memory buffer at 2KB boundaries.
union mem_packet_header_t::@123	word_2	Second word.

### 3.23.3 MU PE Unions

#### 3.23.3.1 mem\_packet\_alloc\_response\_t

Type for response of cmd\_mem\_packet\_alloc.

**Table 3.921. union mem\_packet\_alloc\_response\_t**

Type	Name	Description
uint32_t	reserved:2	Reserved.
uint32_t	packet_number:10	Last segment received.
uint32_t	packet_credit:11	Number of packets to be returned to the master.
uint32_t	buffer_credit:9	Number of buffers to be returned to the master.
uint32_t	value	Accessor to entire structure.

#### 3.23.3.2 mem\_packet\_complete\_request\_t

Packet complete request.

**Table 3.922. union mem\_packet\_complete\_request\_t**

Type	Name	Description
uint32_t	reserved:4	Not used.
uint32_t	nbi_number:2	NBI number.
uint32_t	total_packet_length:14	Total packet length.
MEM_PACKET_REWRITE_SCRIPT_OFFSET	script_offset:14	Script offset.
uint32_t	sequencer_number:5	Sequencer number.
uint32_t	sequence_number:12	Sequence number.
uint32_t	tx_queue_number:10	Tx queue number.
uint32_t	retry_bit:1	Retry bit.
uint64_t	value	Accessor to entire structure.

### 3.23.3.3 mem\_packet\_read\_status\_response\_t

Response of cmd\_mem\_packet\_read\_packet\_status.

**Table 3.923. union mem\_packet\_read\_status\_response\_t**

Type	Name	Description
uint32_t	error:1	Error.
uint32_t	last_segment_received:1	Last segment received.
uint32_t	first_segment_received:1	First segment received.
uint32_t	sent_to_ME:1	Packet sent to ME.
uint32_t	packet_not_valid:1	Packet not valid, error.
uint32_t	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 .
uint32_t	reserved2:6	Reserved.
MEM_PACKET_LENGTH	packet_length:2	Packet size.
uint32_t	reserved:6	Reserved.
uint32_t	ctm_dcache_address_256B:10	CTM DCACHE address.  Multiply by 256 to get actual CTM address.
uint32_t	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 3.924. 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 3.925. 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 3.926. 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 3.927. 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 cmd\_mem\_packet\_read\_packet\_status.

**Table 3.928. `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 `cmd_mem_packet_read_packet_status` in read registers.

**Table 3.929. `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 `cmd_mem_packet_alloc`.

**Table 3.930. `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 `cmd_mem_packet_alloc` in read registers.

**Table 3.931. `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 3.932. `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 3.933. `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 `cmd_mem_packet_credit_get`

**Prototype:**

```
void cmd_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 3.934. `cmd_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"> <li>• 0 - ME</li> <li>• 1 - NBI0</li> <li>• 2 - NBI1</li> </ul>
<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 `cmd_mem_packet_alloc`

**Prototype:**

```
void cmd_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 3.935. cmd\_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"> <li>• 0 - ME</li> <li>• 1 - NBI0</li> <li>• 2 - NBI1</li> </ul>
<code>enum MEM_PACKET_LENGTH</code>	<code>packet_length</code>	<ul style="list-style-type: none"> <li>• 0 - 256B</li> <li>• 1 - 512B</li> <li>• 2 - 1KB</li> <li>• 3 - 2KB</li> <li>• 4 - 4KB This option is available for Kestrel Only</li> <li>• 5 - 8KB This option is available for Kestrel Only</li> <li>• 6 - 16KB This option is available for Kestrel Only</li> <li>• 7 - Reserved This option is available for Kestrel Only</li> <li>• 8 - 6KB This option is available for Kestrel Only</li> <li>• 9 - 10KB This option is available for Kestrel Only</li> <li>• 10 - 12KB This option is available for Kestrel Only</li> <li>• 11 - 14KB This option is available for Kestrel Only</li> </ul>
<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 cmd\_mem\_packet\_alloc\_poll

**Prototype:**

```
void cmd_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 `cmd_mem_packet_alloc_poll()` request with `cmd_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;
uint32_t                                pkt_number;
SIGNAL                                    sig;

// allocate two packets
cmd_mem_packet_alloc_poll(&alloc_response, bucket_master, PACKET_LENGTH_1KB, ctx_swap, &sig);
cmd_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;

    cmd_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 3.936. cmd\_mem\_packet\_alloc\_poll parameters**

Type	Name	Description
__xread mem_packet_alloc_response_t*	data	Transfer register for packet allocation response
enum MEM_PACKET_MASTER_BUCKET	credit_bucket	<ul style="list-style-type: none"> <li>• 0 - ME</li> <li>• 1 - NBI0</li> <li>• 2 - NBI1</li> </ul>
enum MEM_PACKET_LENGTH	packet_length	<ul style="list-style-type: none"> <li>• 0 - 256B</li> <li>• 1 - 512B</li> <li>• 2 - 1KB</li> <li>• 3 - 2KB</li> <li>• 4 - 4KB This option is available for Kestrel Only</li> </ul>

Type	Name	Description
		<ul style="list-style-type: none"> <li>• 5 - 8KB This option is available for Kestrel Only</li> <li>• 6 - 16KB This option is available for Kestrel Only</li> <li>• 7 - Reserved This option is available for Kestrel Only</li> <li>• 8 - 6KB This option is available for Kestrel Only</li> <li>• 9 - 10KB This option is available for Kestrel Only</li> <li>• 10 - 12KB This option is available for Kestrel Only</li> <li>• 11 - 14KB This option is available for Kestrel Only</li> </ul>
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

### 3.23.5.4 cmd\_mem\_packet\_free

**Prototype:**

```
void cmd_mem_packet_free(uint32_t packet_number)
```

**Description:**

Packet free request.

Free the packet

**Table 3.937. cmd\_mem\_packet\_free parameters**

Type	Name	Description
uint32_t	<i>packet_number</i>	Packet number to free

### 3.23.5.5 cmd\_mem\_packet\_free\_and\_return\_pointer

**Prototype:**

```
void cmd_mem_packet_free_and_return_pointer(__xread void* data, uint32_t 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 3.938. cmd\_mem\_packet\_free\_and\_return\_pointer parameters**

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Transfer register for packet free response
<code>uint32_t</code>	<code>packet_number</code>	Packet number to be freed
<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

### 3.23.5.6 cmd\_mem\_packet\_free\_and\_signal

**Prototype:**

```
void cmd_mem_packet_free_and_signal(uint32_t packet_number, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

Packet free and signal request.

Free Packet and signal when packet is freed.

**Table 3.939. cmd\_mem\_packet\_free\_and\_signal parameters**

Type	Name	Description
<code>uint32_t</code>	<code>packet_number</code>	Packet number to be freed
<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

### 3.23.5.7 cmd\_mem\_packet\_return\_pointer

**Prototype:**

```
void cmd_mem_packet_return_pointer(__xread void* data, uint32_t packet_number, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

Packet return pointer request.

Return the MU Pointer in the packet header

**Table 3.940. cmd\_mem\_packet\_return\_pointer parameters**

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Transfer register for packet MU pointer
<code>uint32_t</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)

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

### 3.23.5.8 cmd\_mem\_packet\_complete\_drop

**Prototype:**

```
void cmd_mem_packet_complete_drop(mem_packet_complete_request_t packet_complete, uint32_t
packet_number)
```

**Description:**

Packet processing complete,.

Used for sequence number only.

**Table 3.941. cmd\_mem\_packet\_complete\_drop parameters**

Type	Name	Description
mem_packet_complete_request_t	<i>packet_complete</i>	Data for the packet complete command.
uint32_t	<i>packet_number</i>	Packet number of packet complete.

**Availability:**

NFP-6xxx Indirect Reference Mode

### 3.23.5.9 cmd\_mem\_packet\_complete\_unicast

**Prototype:**

```
void cmd_mem_packet_complete_unicast(mem_packet_complete_request_t packet_complete,
uint32_t packet_number)
```

**Description:**

Packet processing complete for unicast packet.

Packet processing complete and free packet on last transfer.

**Table 3.942. cmd\_mem\_packet\_complete\_unicast parameters**

Type	Name	Description
mem_packet_complete_request_t	<i>packet_complete</i>	Data for the packet complete command.
uint32_t	<i>packet_number</i>	Packet number of packet complete.

**Availability:**

NFP-6xxx Indirect Reference Mode

### 3.23.5.10 cmd\_mem\_packet\_complete\_multicast

**Prototype:**

```
void cmd_mem_packet_complete_multicast(mem_packet_complete_request_t packet_complete,  
uint32_t packet_number)
```

**Description:**

Packet processing complete for multicast packet.

Packet processing complete but do not free packet on last transfer.

**Table 3.943. cmd\_mem\_packet\_complete\_multicast parameters**

Type	Name	Description
mem_packet_complete_request_t	<i>packet_complete</i>	Data for the packet complete command.
uint32_t	<i>packet_number</i>	Packet number of packet complete.

**Availability:**

NFP-6xxx Indirect Reference Mode

### 3.23.5.11 cmd\_mem\_packet\_complete\_multicast\_free

**Prototype:**

```
void cmd_mem_packet_complete_multicast_free(mem_packet_complete_request_t packet_complete,  
uint32_t packet_number)
```

**Description:**

Packet processing complete for multicast packet.

Packet processing complete and free packet on last transfer.

**Table 3.944. cmd\_mem\_packet\_complete\_multicast\_free parameters**

Type	Name	Description
mem_packet_complete_request_t	<i>packet_complete</i>	Data for the packet complete command.
uint32_t	<i>packet_number</i>	Packet number of packet complete.

**Availability:**

NFP-6xxx Indirect Reference Mode

### 3.23.5.12 cmd\_mem\_packet\_read\_packet\_status

**Prototype:**

```
void cmd_mem_packet_read_packet_status(__xread mem_packet_read_status_response_t* data,
uint32_t 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 3.945. cmd\_mem\_packet\_read\_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>uint32_t</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

### 3.23.5.13 cmd\_mem\_packet\_wait\_packet\_status

**Prototype:**

```
void cmd_mem_packet_wait_packet_status(__xread mem_packet_read_status_response_t* data,
uint32_t 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 3.946. cmd\_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>uint32_t</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 cmd\_mem\_packet\_add\_thread

**Prototype:**

```
void cmd_mem_packet_add_thread(__xread void* data, uint32_t packet_offset, uint32_t 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 3.947. cmd\_mem\_packet\_add\_thread parameters**

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Transfer register for packet allocation response
<code>uint32_t</code>	<code>packet_offset</code>	Offset to receive from after the first 6 32-bit words at 4 byte granularity.
<code>uint32_t</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 cmd\_mem\_pe\_dma\_to\_memory\_packet

**Prototype:**

```
void cmd_mem_pe_dma_to_memory_packet(volatile void __addr40 __mem* address, uint32_t
packet_offset, uint32_t packet_number, uint32_t 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 `cmd_mem_pe_dma_to_memory_packet()`:

```
__emem_n(1) __addr40 uint64_t           emem_address[6];
mem_packet_alloc_response_in_read_reg_t alloc_response;
uint32_t                                     pkt_number;
SIGNAL                                         sig;
uint32_t                                     packet_offset = 0;

// allocate a packet
cmd_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 uint32_t                  ctm_packet;
    __xwrite uint64_t      write_data[2];

    write_data[0] = 0x1122334455667788;
    write_data[1] = 0x9900aabbccddeeff;
    ctm_packet = (1 << 31) | (uint32_t)(pkt_number << 16) | (packet_offset);
    cmd_mem_write64_ptr32
    (
        (void*)write_data,
        (volatile void __ctm*)ctm_packet,
        2,
        ctx_swap,
        &sig
    );
}
cmd_mem_pe_dma_to_memory_packet(emem_address, packet_offset, pkt_number, 0, ctx_swap, &sig);

// verify data in emem1
{
    __xread uint64_t      read_data[2];

    cmd_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 3.948. cmd\_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.
uint32_t	packet_offset	8B aligned offset with packet data to start DMA from.
uint32_t	packet_number	Packet number of packet to write.
uint32_t	ctm_island	Address of the CTM to execute instruction. Can be 0 for local island. See 6xxx databook for recommended addressing mode.

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

**Availability:**

NFP-6xxx Indirect Reference Mode

### 3.23.5.16 cmd\_mem\_pe\_dma\_to\_memory\_packet\_free

**Prototype:**

```
void cmd_mem_pe_dma_to_memory_packet_free(volatile void __addr40 __mem* address, uint32_t
packet_offset, uint32_t packet_number, uint32_t 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 3.949. cmd\_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.
uint32_t	<i>packet_offset</i>	8B aligned offset with packet data to start DMA from.
uint32_t	<i>packet_number</i>	Packet number of packet to write.
uint32_t	<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 cmd\_mem\_pe\_dma\_to\_memory\_packet\_swap

**Prototype:**

```
void cmd_mem_pe_dma_to_memory_packet_swap(volatile void __addr40 __mem* address, uint32_t packet_offset, uint32_t packet_number, uint32_t 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 3.950. cmd\_mem\_pe\_dma\_to\_memory\_packet\_swap parameters**

Type	Name	Description
volatile void __addr40 __mem*	address	40 bit address of external memory including the MU island. Must be 8B aligned.
uint32_t	packet_offset	8B aligned offset with packet data to start DMA from.
uint32_t	packet_number	Packet number of packet to write.
uint32_t	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

### 3.23.5.18 cmd\_mem\_pe\_dma\_to\_memory\_packet\_free\_swap

**Prototype:**

```
void cmd_mem_pe_dma_to_memory_packet_free_swap(volatile void __addr40 __mem* address, uint32_t packet_offset, uint32_t packet_number, uint32_t 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 3.951. cmd\_mem\_pe\_dma\_to\_memory\_packet\_free\_swap parameters**

Type	Name	Description
<code>volatile void __addr40 __mem*</code>	<code>address</code>	40 bit address of external memory including the MU island. Must be 8B aligned.
<code>uint32_t</code>	<code>packet_offset</code>	8B aligned offset with packet data to start DMA from.
<code>uint32_t</code>	<code>packet_number</code>	Packet number of packet to write.
<code>uint32_t</code>	<code>ctm_island</code>	Address of the CTM to execute instruction. Can be 0 for local island. See 6xxx databook for 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

**Availability:**

NFP-6xxx Indirect Reference Mode

### 3.23.5.19 cmd\_mem\_pe\_dma\_to\_memory\_buffer

**Prototype:**

```
void cmd_mem_pe_dma_to_memory_buffer(volatile void __ctm32 *source_address, volatile void __addr40 __mem* destination_address, uint32_t ctm_island, uint32_t 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 3.952. cmd\_mem\_pe\_dma\_to\_memory\_buffer parameters**

Type	Name	Description
volatile void __ctm32*	<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.
uint32_t	<i>ctm_island</i>	Address of the CTM to execute instruction. Can be 0 for local island. See 6xxx databook for recommended addressing mode.
uint32_t	<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

### 3.23.5.20 cmd\_mem\_pe\_dma\_to\_memory\_buffer\_swap

**Prototype:**

```
void cmd_mem_pe_dma_to_memory_buffer_swap(volatile void __ctm32 *source_address, volatile
void __addr40 __mem* destination_address, uint32_t ctm_island, uint32_t 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 3.953. cmd\_mem\_pe\_dma\_to\_memory\_buffer\_swap parameters**

Type	Name	Description
volatile void __ctm32*	<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.
uint32_t	<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
uint32_t	<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

### 3.23.5.21 cmd\_mem\_pe\_dma\_to\_memory\_buffer\_le

#### Prototype:

```
void cmd_mem_pe_dma_to_memory_buffer_le(volatile void __ctm32 *source_address, volatile
void __addr40 __mem* destination_address, uint32_t ctm_island, uint32_t 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 3.954. cmd\_mem\_pe\_dma\_to\_memory\_buffer\_le parameters**

Type	Name	Description
volatile void __ctm32*	<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.
uint32_t	<i>ctm_island</i>	Address of the CTM to execute instruction. Can be 0 for local island. See 6xxx databook for recommended addressing mode.
uint32_t	<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

### 3.23.5.22 cmd\_mem\_pe\_dma\_to\_memory\_buffer\_le\_swap

**Prototype:**

```
void cmd_mem_pe_dma_to_memory_buffer_le_swap(volatile void __ctm32 *source_address,  
volatile void __addr40 __mem* destination_address, uint32_t ctm_island, uint32_t 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 3.955. cmd\_mem\_pe\_dma\_to\_memory\_buffer\_le\_swap parameters**

Type	Name	Description
volatile void __ctm32*	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.
uint32_t	ctm_island	Address of the CTM to execute instruction. Can be 0 for local island. See 6xxx databook for recommended addressing mode.
uint32_t	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

### 3.23.5.23 cmd\_mem\_pe\_dma\_from\_memory\_buffer

**Prototype:**

```
void cmd_mem_pe_dma_from_memory_buffer(volatile void __addr40 __mem* source_address,
volatile void __ctm32 *destination_address, uint32_t ctm_island, uint32_t 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**

The architecture of NFP38xx requires this function to be supplied with a a SIGNAL\_PAIR ptr.

LoadTimeConstant() can be used to populate ctm\_island address correctly, i.e.  
LoadTimeConstant("\_\_ADDR\_I34\_CTM") for CTM on i34.

**Table 3.956. cmd\_mem\_pe\_dma\_from\_memory\_buffer 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 __ctm32*	destination_address	32 bit destination address of CTM (must be 8B aligned)
uint32_t	ctm_island	Address of the CTM to execute instruction. Can be 0 for local island. See 6xxx databook for recommended addressing mode.
uint32_t	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, NFP-38xx uses a signal_pair ptr

**Availability:**

NFP-6xxx Indirect Reference Mode

### 3.23.5.24 cmd\_mem\_pe\_dma\_from\_memory\_buffer\_swap

**Prototype:**

```
void cmd_mem_pe_dma_from_memory_buffer_swap(volatile void __addr40 __mem* source_address,
volatile void __ctm32 *destination_address, uint32_t ctm_island, uint32_t 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 3.957. cmd\_mem\_pe\_dma\_from\_memory\_buffer\_swap 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 __ctm32*	destination_address	32 bit destination address of CTM (must be 8B aligned)
uint32_t	ctm_island	Address of the CTM to execute instruction. Can be 0 for local island. See 6xxx databook for recommended addressing mode.
uint32_t	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

### 3.23.5.25 cmd\_mem\_pe\_dma\_from\_memory\_buffer\_le

**Prototype:**

```
void cmd_mem_pe_dma_from_memory_buffer_le(volatile void __addr40 __mem* source_address,
volatile void __ctm32 *destination_address, uint32_t ctm_island, uint32_t 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 3.958. cmd\_mem\_pe\_dma\_from\_memory\_buffer\_le 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 __ctm32*	<i>destination_address</i>	32 bit destination address of CTM (must be 8B aligned)
uint32_t	<i>ctm_island</i>	Address of the CTM to execute instruction. Can be 0 for local island. See 6xxx databook for recommended addressing mode.
uint32_t	<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

### 3.23.5.26 cmd\_mem\_pe\_dma\_from\_memory\_buffer\_le\_swap

**Prototype:**

```
void cmd_mem_pe_dma_from_memory_buffer_le_swap(volatile void __addr40 __mem*
source_address, volatile void __ctm32 *destination_address, uint32_t ctm_island, uint32_t
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 3.959. cmd\_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 __ctm32*	<i>destination_address</i>	32 bit destination address of CTM (must be 8B aligned)

Type	Name	Description
uint32_t	<i>ctm_island</i>	Address of the CTM to execute instruction. Can be 0 for local island. See 6xxx databook for recommended addressing mode.
uint32_t	<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 cmd\_mem\_pe\_dma\_to\_memory\_indirect

#### Prototype:

```
void cmd_mem_pe_dma_to_memory_indirect(__xread uint32_t* data, uint32_t 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 3.960. cmd\_mem\_pe\_dma\_to\_memory\_indirect parameters**

Type	Name	Description
__xread uint32_t*	<i>data</i>	Transfer register with read result
uint32_t	<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

### 3.23.5.28 cmd\_mem\_pe\_dma\_to\_memory\_indirect\_swap

#### Prototype:

```
void cmd_mem_pe_dma_to_memory_indirect_swap(__xread uint32_t* data, uint32_t 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 3.961. cmd\_mem\_pe\_dma\_to\_memory\_indirect\_swap parameters**

Type	Name	Description
<code>__xread uint32_t*</code>	<code>data</code>	Transfer register with read result
<code>uint32_t</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 cmd\_mem\_pe\_dma\_to\_memory\_indirect\_free

**Prototype:**

```
void cmd_mem_pe_dma_to_memory_indirect_free(__xread uint32_t* data, uint32_t 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 3.962. cmd\_mem\_pe\_dma\_to\_memory\_indirect\_free parameters**

Type	Name	Description
<code>__xread uint32_t*</code>	<code>data</code>	Transfer register with read result
<code>uint32_t</code>	<code>packet_number</code>	Packet number of packet to write

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

### 3.23.5.30 cmd\_mem\_pe\_dma\_to\_memory\_indirect\_free\_swap

**Prototype:**

```
void cmd_mem_pe_dma_to_memory_indirect_free_swap(__xread uint32_t* data, uint32_t
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 3.963. cmd\_mem\_pe\_dma\_to\_memory\_indirect\_free\_swap parameters**

Type	Name	Description
__xread uint32_t*	<i>data</i>	Transfer register with read result
uint32_t	<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 3.964. 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 & 0xffffffff)
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

Defined	Definition
	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 & 0x3fffffffff)  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.
MEM_STATS_SATURATE_BYTE_COUNT_SATURATE_IND(value)	((value >> 34) & 0x01)  Helper macro to extract only the saturate byte count indicator (1 bit) from 64-bit value after reading statistics in saturating statistic format.
MEM_STATS_EXTRACT_SATURATE_PKT_AND_BYTE_COUNT	(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); \ }  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.

## 3.24.2 MU Statistics Enumerations

### 3.24.2.1 MEM\_STATS\_BASE\_ADDRESS\_SELECT

Base address CSR select (MUSEBaseAddr).

**Table 3.965. enum MEM\_STATS\_BASE\_ADDRESS\_SELECT**

Name	Description
BASE_ADDRESS_0	Base address 0.
BASE_ADDRESS_1	Base address 1.
BASE_ADDRESS_2	Base address 2.
BASE_ADDRESS_3	Base address 3.

### 3.24.2.2 MEM\_STATS\_ADDRESS\_PACK\_CONFIG

Address packing configuration.

**Table 3.966. 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 cmd\_mem\_stats\_log\_command\_address\_format\_t

Log statistic command address field.

**Table 3.967. union cmd\_mem\_stats\_log\_command\_address\_format\_t**

Type	Name	Description
uint32_t	reserved:14	Reserved.
MEM_STATS_ADDRESS_PACK_CONFIG	address_packed_config:2	How the pulled addresses are packed into the transfer registers.
uint32_t	add_byte_count_value:16	Value to add to the byte count statistics field.
uint32_t	value	Accessor to entire descriptor structure.

### **3.24.3.2 cmd\_mem\_stats\_read\_command\_address\_format\_t**

Read (push) statistic command address field.

**Table 3.968. union cmd\_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.
uint32_t	reserved_2:8	Reserved.
uint32_t	statistic_address:19	Statistic address (64b aligned).
uint32_t	reserved_1:3	Reserved.
uint32_t	value	Accessor to entire descriptor structure.

### **3.24.3.3 mem\_stats\_packed\_address\_detail\_t**

Statistic packed address detail (16 bit).

**Table 3.969. 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.
uint32_t	statistic_address:14	Statistic address (64b aligned).
uint16_t	value	Accessor to entire descriptor structure.

### **3.24.3.4 mem\_stats\_unpacked\_address\_detail\_t**

Statistic unpacked address detail (32 bit).

**Table 3.970. 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.
uint32_t	reserved:11	Reserved.
uint32_t	statistic_address:19	Statistic address (64b aligned).
uint32_t	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 3.971. `typedef MEM_STATS_BASE_ADDRESS_SELECT`**

Type	Definition
<code>MEM_STATS_BASE_ADDRESS_SELECT</code>	<code>enum MEM_STATS_BASE_ADDRESS_SELECT</code>

### 3.24.4.2 `MEM_STATS_ADDRESS_PACK_CONFIG`

Address packing configuration.

**Table 3.972. `typedef MEM_STATS_ADDRESS_PACK_CONFIG`**

Type	Definition
<code>MEM_STATS_ADDRESS_PACK_CONFIG</code>	<code>enum MEM_STATS_ADDRESS_PACK_CONFIG</code>

### 3.24.4.3 `cmd_mem_stats_log_command_address_format_t`

Log statistic command address field.

**Table 3.973. `typedef cmd_mem_stats_log_command_address_format_t`**

Type	Definition
<code>cmd_mem_stats_log_command_address_format_t</code>	<code>union cmd_mem_stats_log_command_address_format_t</code>

### 3.24.4.4 `cmd_mem_stats_read_command_address_format_t`

Read (push) statistic command address field.

**Table 3.974. `typedef cmd_mem_stats_read_command_address_format_t`**

Type	Definition
<code>cmd_mem_stats_read_command_address_format_t</code>	<code>union cmd_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 3.975. `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 3.976. `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 cmd\_mem\_stats\_read

**Prototype:**

```
void cmd_mem_stats_read(__xread void* xfer, volatile void __addr40 __mem* address, uint32_t
count, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

Retrieve the statistic data from the data cache.

**Table 3.977. cmd\_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>cmd_mem_stats_read_command_address_format_t</code> for the lower 32 bits.
<code>uint32_t</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 cmd\_mem\_stats\_read\_and\_clear

**Prototype:**

```
void cmd_mem_stats_read_and_clear(__xread void* xfer, volatile void __addr40 __mem*
address, uint32_t 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 3.978. cmd\_mem\_stats\_read\_and\_clear parameters**

Type	Name	Description
<code>__xread void*</code>	<code>xfer</code>	Transfer registers containing statistics data.

Type	Name	Description
volatile void __addr40 __mem*	address	40 bit pointer containing mu island in upper 8 bits (see 6xxx databook for recommended addressing mode) and cmd_mem_stats_read_command_address_format_t for the lower 32 bits.
uint32_t	count	Length in 64-bits 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.

### 3.24.5.3 cmd\_mem\_stats\_log

**Prototype:**

```
void cmd_mem_stats_log(__xwrite void* xfer, volatile void __addr40 __mem* data, uint32_t
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 3.979. cmd\_mem\_stats\_log parameters**

Type	Name	Description
__xwrite void*	xfer	Transfer registers containing statistic addresses. Use either mem_stats_unpacked_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 cmd_mem_stats_log_command_address_format_t for the lower 32 bits
uint32_t	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

### 3.24.5.4 cmd\_mem\_stats\_log\_saturate

**Prototype:**

```
void cmd_mem_stats_log_saturate(__xwrite void* xfer, volatile void __addr40 __mem* data,
uint32_t 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 3.980. cmd\_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>cmd_mem_stats_log_command_address_format_t</code> for the lower 32 bits
<code>uint32_t</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 cmd\_mem\_stats\_log\_event

**Prototype:**

```
void cmd_mem_stats_log_event(__xwrite void* xfer, volatile void __addr40 __mem* data,
                           uint32_t 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 3.981. cmd\_mem\_stats\_log\_event 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>cmd_mem_stats_log_command_address_format_t</code> for the lower 32 bits
<code>uint32_t</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:**

- cmd\_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 cmd\_mem\_stats\_log\_event\_saturate

**Prototype:**

```
void cmd_mem_stats_log_event_saturate(__xwrite void* xfer, volatile void __addr40 __mem* data, uint32_t 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 3.982. cmd\_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>cmd_mem_stats_log_command_address_format_t</code> for the lower 32 bits
<code>uint32_t</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:**

- cmd\_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 3.983. MU LB Defines**

Defined	Definition
MEM_LB_HASH_SEED_HIGH(seed)	((seed >> 13) & 0x7ff);  Helper macro to extract the hash_seed_high (higher 11 bits) from 24-bit hash seed as needed for load balance descriptor data (mem_lb_desc_t).
MEM_LB_HASH_SEED_LOW(seed)	(seed & 0xffff);  Helper macro to extract the hash_seed_low (lower 13 bits) from 24-bit hash seed as needed for load balance descriptor data (mem_lb_desc_t).
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 3.984. union mem\_lb\_bucket\_dcache\_address\_format\_t**

Type	Name	Description
uint32_t	reserved_2:10	Reserved.
uint32_t	bucket_address:18	Dcache hash bucket address in hash table array.
uint32_t	reserved_1:4	Reserved.
uint32_t	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 3.985. union mem\_lb\_bucket\_local\_address\_format\_t**

Type	Name	Description
uint32_t	reserved_2:19	Reserved.
uint32_t	bucket_address:9	Local hash bucket address in hash table array.
uint32_t	reserved_1:4	Reserved.
uint32_t	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 3.986. union mem\_lb\_dcache\_stats\_address\_format\_t**

Type	Name	Description
uint32_t	reserved_2:10	Reserved.
uint32_t	statistic_address:19	Statistic address (64b address).
uint32_t	reserved_1:3	Reserved.
uint32_t	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 3.987. union mem\_lb\_descriptor\_address\_format\_t**

Type	Name	Description
uint32_t	reserved:26	Reserved.
uint32_t	array_entry:6	Descriptor array entry location.
uint32_t	value	Accessor to entire descriptor structure.

### 3.25.2.5 mem\_lb\_desc\_t

Load balance descriptor detail.

**Table 3.988. union mem\_lb\_desc\_t**

Type	Name	Description
uint32_t	hash_base_addr_low:13	Upper 13 bits of 18 bit hash base address.
uint32_t	stat_addr_ref:1	Location reference for StatAddr (0-local mem,1-DCache).
uint32_t	stat_base_addr:18	Statistics base address.
uint32_t	hash_seed_low:13	Lower 13 bits of 24 bit hash seed value.
uint32_t	boundaries:3	Number of boundaries in the hash bucket.
uint32_t	index_bits:5	Number of bits of hash to use as address offset.
uint32_t	compare_bits:5	Number of bits of hash to use for compare.
uint32_t	hash_addr_ref:1	Location reference for HashAddr (0-local mem,1-DCache).
uint32_t	hash_base_addr_high:5	Upper 5 bits of 18 bit hash base address.
uint32_t	reserved_1:21	Reserved.
uint32_t	hash_seed_high:11	Upper 11 bits of 24 bit hash seed value.
uint32_t	reserved_2:32	Reserved.
uint32_t	user_data_1:32	User data first 32-bit word.
uint32_t	user_data_2:32	User data second 32-bit word.
uint32_t	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 3.989. union mem\_lb\_id\_table\_address\_format\_t**

Type	Name	Description
uint32_t	reserved_2:21	Reserved.
uint32_t	id_table_address:8	Local ID table array address.
uint32_t	reserved_1:3	Reserved.

Type	Name	Description
uint32_t	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 3.990. union mem\_lb\_local\_stats\_address\_format\_t**

Type	Name	Description
uint32_t	reserved_2:19	Reserved.
uint32_t	statistic_address:10	Statistic address (64b address).
uint32_t	reserved_1:3	Reserved.
uint32_t	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 3.991. union mem\_lb\_lookup\_bundle\_id\_result\_t**

Type	Name	Description
uint32_t	reserved_1:24	Reserved.
uint32_t	bundle_id:8	Bundle id.
uint32_t	reserved_2:32	Reserved.
uint32_t	user_data_1:32	User_data_1.
uint32_t	user_data_2:32	User data_2.
uint32_t	value[4]	Accessor to entire lookup detail structure.

### 3.25.2.9 mem\_lb\_lookup\_command\_format\_t

Lookup data detail.

**Table 3.992. union mem\_lb\_lookup\_command\_format\_t**

Type	Name	Description
uint32_t	reserved_1:8	Reserved.
uint32_t	hash_input:24	Initial hash input.
uint32_t	reserved_2:16	Reserved.
uint32_t	stat_add_value:16	Value to add to statistic.
uint32_t	padding[2]	Padding required for xfer to be used together with result lookup.

Type	Name	Description
uint32_t	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 3.993. union mem\_lb\_lookup\_direct\_result\_t**

Type	Name	Description
uint32_t	lo_result:32	Bottom 32 bits of id table lookup result.
uint32_t	hi_result:32	Top 32 bits of id table lookup result.
uint32_t	user_data_1:32	User_data_1.
uint32_t	user_data_2:32	User data_2.
uint32_t	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 3.994. 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 3.995. 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 3.996. `typedef mem_lb_desc_in_write_reg_t`**

Type	Definition
<code>mem_lb_desc_in_write_reg_t</code>	<code>__xwrite mem_lb_desc_t</code>

### 3.25.3.4 `mem_lb_desc_in_read_reg_t`

descriptor in write\_reg

**Table 3.997. `typedef mem_lb_desc_in_read_reg_t`**

Type	Definition
<code>mem_lb_desc_in_read_reg_t</code>	<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 3.998. `typedef mem_lb_local_stats_address_format_t`**

Type	Definition
<code>mem_lb_local_stats_address_format_t</code>	<code>union 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 3.999. `typedef mem_lb_dcache_stats_address_format_t`**

Type	Definition
<code>mem_lb_dcache_stats_address_format_t</code>	<code>union 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 3.1000. `typedef mem_lb_descriptor_address_format_t`**

Type	Definition
<code>mem_lb_descriptor_address_format_t</code>	<code>union 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 3.1001. `typedef mem_lb_id_table_address_format_t`**

Type	Definition
<code>mem_lb_id_table_address_format_t</code>	<code>union 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 3.1002. `typedef mem_lb_bucket_local_address_format_t`**

Type	Definition
<code>mem_lb_bucket_local_address_format_t</code>	<code>union 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 3.1003. `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 3.1004. `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 3.1005. `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 3.1006. `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 3.1007. `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 3.1008. `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 3.1009. `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 `cmd_mem_lb_write_desc`

**Prototype:**

```
void cmd_mem_lb_write_desc(mem_lb_desc_in_write_reg_t* xfer, volatile void __addr40 __mem*
address, uint32_t count, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

Write descriptor data into the Local Descriptor Array.

**Table 3.1010. cmd\_mem\_lb\_write\_desc parameters**

Type	Name	Description
mem_lb_desc_in_write_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.
uint32_t	count	Length in 64 bit words to write, 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. * 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 cmd\_mem\_lb\_read\_desc

**Prototype:**

```
void cmd_mem_lb_read_desc(mem_lb_desc_in_read_reg_t* xfer, volatile void __addr40 __mem*
address, uint32_t 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 3.1011. cmd\_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.
uint32_t	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 cmd\_mem\_lb\_write\_id\_table

**Prototype:**

```
void cmd_mem_lb_write_id_table(__xwrite void* xfer, volatile void __addr40 __mem* address,
uint32_t 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 3.1012. cmd\_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 databook for recommended addressing mode for higher 8 bits of address parameter.
uint32_t	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 cmd\_mem\_lb\_read\_id\_table

**Prototype:**

```
void cmd_mem_lb_read_id_table(__xread void* xfer, volatile void __addr40 __mem* address,
uint32_t 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 3.1013. cmd\_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.
uint32_t	count	Length in 64 bit words to read (valid values 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.

### 3.25.4.5 cmd\_mem\_lb\_bucket\_write\_local

**Prototype:**

```
void cmd_mem_lb_bucket_write_local(__xwrite void* xfer, volatile void __addr40 __mem*
address, uint32_t 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 3.1014. cmd\_mem\_lb\_bucket\_write\_local parameters**

Type	Name	Description
__xwrite void*	<i>xfer</i>	Transfer registers containing entries to write.
volatile void __addr40 __mem*	<i>address</i>	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.
uint32_t	<i>count</i>	Length in 64 bit words to write, calculated as count * 2 (valid values 1 - 8).
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.

### 3.25.4.6 cmd\_mem\_lb\_bucket\_read\_local

**Prototype:**

```
void cmd_mem_lb_bucket_read_local(__xread void* xfer, volatile void __addr40 __mem*
address, uint32_t 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 3.1015. cmd\_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>uint32_t</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 (sig_done or ctx_swap).
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion.

### 3.25.4.7 cmd\_mem\_lb\_bucket\_write\_dcache

**Prototype:**

```
void cmd_mem_lb_bucket_write_dcache(__xwrite void* xfer, volatile void __addr40 __mem*
address, uint32_t 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 3.1016. cmd\_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>uint32_t</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 (sig_done or ctx_swap).
<code>SIGNAL*</code>	<code>sig_ptr</code>	Signal to raise upon completion.

### 3.25.4.8 cmd\_mem\_lb\_bucket\_read\_dcache

**Prototype:**

```
void cmd_mem_lb_bucket_read_dcache(__xread void* xfer, volatile void __addr40 __mem*
address, uint32_t 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 3.1017. cmd\_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>uint32_t</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 cmd\_mem\_lb\_stats\_read\_and\_clear\_local

**Prototype:**

```
void cmd_mem_lb_stats_read_and_clear_local(__xread void* xfer, volatile void __addr40 __mem* address, uint32_t 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 3.1018. cmd\_mem\_lb\_stats\_read\_and\_clear\_local 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 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>uint32_t</code>	<code>count</code>	Length in 64 bit words 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

### 3.25.4.10 cmd\_mem\_lb\_stats\_read\_local

**Prototype:**

```
void cmd_mem_lb_stats_read_local(__xread void* xfer, volatile void __addr40 __mem* address,
uint32_t 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 3.1019. cmd\_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.
uint32_t	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 cmd\_mem\_lb\_stats\_read\_and\_clear\_dcache

**Prototype:**

```
void cmd_mem_lb_stats_read_and_clear_dcache(__xread void* xfer, volatile void __addr40 __mem* address,
uint32_t 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 3.1020. cmd\_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

Type	Name	Description
		databook for recommended addressing mode for higher 8 bits of address parameter.
uint32_t	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 cmd\_mem\_lb\_stats\_read\_dcache

**Prototype:**

```
void cmd_mem_lb_stats_read_dcache(__xread void* xfer, volatile void __addr40 __mem* address, uint32_t 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 3.1021. cmd\_mem\_lb\_stats\_read\_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.
uint32_t	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 cmd\_mem\_lb\_lookup\_bundle\_id

**Prototype:**

```
mem_lb_lookup_bundle_id_result_in_read_reg_t*
cmd_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;
uint64_t mu_island = (uint64_t)LoadTimeConstant( "__ADDR_I29_IMEM" );
uint32_t descriptor_array = 3;
uint32_t count = 1;
uint32_t write_stat_base_addr = 0x100;
uint32_t write_hash_base_addr = 0x00;
uint32_t write_user_data_1 = 0x9abcdeff;
uint32_t write_user_data_2 = 0x12345678;
uint32_t write_hash_seed = 0xaaaaaaa;

// Write 0x0f to load balancing hash multiply CSR
{
    cluster_target_xpb_address_format_t xpb_command;
    __xwrite uint32_t wr_xfer;
    xpb_command.value = 0;
    xpb_command.device = 0x20;
    xpb_command.target_island = 29;
    xpb_command.global_xpb = 1;

    wr_xfer = 0x0f;
    cmd_cluster_target_xpb_write((void *)&wr_xfer, &xpb_command, 1, ctx_swap, &sig);
}

// Initialise load balancing descriptor array 3
cmd_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,
    ((uint64_t)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 uint32_t write_xfer[4];
    __xread uint32_t read_xfer[4];
    uint32_t hb_entry = 0;
    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;

{
    uint64_t addr = mu_island |
        (uint64_t)(write_hash_base_addr | hb_addr.value);

    cmd_mem_lb_bucket_write_local(
        (void *)write_xfer,
        (__addr40 __mem void *) addr,
        1,
        ctx_swap,
        &sig
    );
    cmd_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;
    uint64_t                           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 = cmd_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 3.1022. cmd\_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 (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).

### 3.25.4.14 cmd\_mem\_lb\_lookup\_dcache

**Prototype:**

```
mem_lb_lookup_direct_result_in_read_reg_t*
cmd_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 3.1023. cmd\_mem\_lb\_lookup\_dcache parameters**

Type	Name	Description
mem_lb_lookup_command_format_in_write_reg_t* <i>xfer</i>		Transfer registers containing one 64 bit word of hash and statistics
volatile void __addr40 __mem* <i>address</i>		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 <i>sync</i>		Type of synchronization to use (sig_done or ctx_swap)
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 (two 64-bit words).

### 3.25.4.15 cmd\_mem\_lb\_lookup\_id\_table

**Prototype:**

```
mem_lb_lookup_direct_result_in_read_reg_t*
cmd_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 3.1024. cmd\_mem\_lb\_lookup\_id\_table parameters**

Type	Name	Description
mem_lb_lookup_command_format_in_write_reg_t* <i>xfer</i>		Transfer registers containing one 64 bit word of hash and statistics
volatile void __addr40 __mem* <i>address</i>		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 <i>sync</i>		Type of synchronization to use (sig_done or ctx_swap)
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 (two 64-bit words).

### 3.25.4.16 cmd\_mem\_lb\_init

**Prototype:**

```
void cmd_mem_lb_init(uint32_t descriptor_array_number, uint32_t mu_island, uint32_t
stat_base_addr, enum MEM_LB_LOCATION stat_addr_ref, uint32_t hash_base_addr, enum
MEM_LB_LOCATION hash_addr_ref, uint32_t boundaries, uint32_t index_bits, uint32_t
compare_bits, uint32_t hash_seed, uint64_t user_data, uint32_t 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). cmd\_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 cmd\_mem\_lb\_init() is given below. The descriptor is stored in array number 4. cmd\_mem\_lb\_read\_desc() is used to read back the descriptor and verify the stored values.

```
uint64_t island = (uint64_t)LoadTimeConstant("__ADDR_I28_IMEM");
mem_lb_descriptor_address_format_t descriptor;
uint32_t descriptor_array = 5;
uint32_t count = 1;

uint32_t write_stat_base_addr = 0x100;
uint32_t write_hash_base_addr = 0x00;
uint32_t write_user_data_1 = 0x9abcdeff;
uint32_t write_user_data_2 = 0x12345678;
uint32_t write_hash_seed = 0xaaaaaaaa;

descriptor.value = 0;
descriptor.array_entry = descriptor_array;

cmd_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,
    ((uint64_t)write_user_data_1 << 32) | write_user_data_2,
    count
);
```

```
// Verify by reading back the descriptor
{
    uint64_t address = ((uint64_t) island | descriptor.value);
    SIGNAL           sig;
    mem_lb_desc_in_read_reg_t      lb_read_xfer_desc;

    cmd_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
    {
        uint32_t 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
    {
        uint32_t read_stat_base_addr= lb_read_xfer_desc.stat_base_addr;
        uint32_t 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
    {
        uint32_t read_hash_addr_ref = lb_read_xfer_desc.hash_addr_ref;
        uint32_t 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 3.1025. cmd\_mem\_lb\_init parameters**

Type	Name	Description
uint32_t	<i>descriptor_array_number</i>	Descriptor array entry location (0 - 63).
uint32_t	<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.
uint32_t	<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)
uint32_t	<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)
uint32_t	<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 the descriptor value. Hash is compared to boundary value in order to determine region indirection index
uint32_t	<i>index_bits</i>	Number of bits of hash to use as address offset
uint32_t	<i>compare_bits</i>	Number of bits of hash to use for compare
uint32_t	<i>hash_seed</i>	Hash seed value
uint64_t	<i>user_data</i>	User specific data
uint32_t	<i>count</i>	Length in 64 bit words to write, calculated as count * 3 (valid values 1- 4)

#### See Also:

- [cmd\\_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 3.1026. struct mem\_lookup\_result\_t**

Type	Name	Description
uint32_t	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 3.1027. 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 3.1028. 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 3.1029. 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 3.1030. 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 3.1031. 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 3.1032. 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.
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.

Name	Description
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 3.1033. 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 3.1034. 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 3.1035. 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.
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.

Name	Description
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 3.1036. union mem\_lookup\_alut\_t**

Type	Name	Description
uint32_t	internal_memory:1	If internal memory is used.
uint32_t	direct_lookup:1	0 = not a direct lookup.
uint32_t	lookup_type:1	0 = algorithmic lookup.
MEM_LOOKUP_ALUT_SIZE	table_size:2	Specified table size.
uint32_t	table_number:3	Table number 0-7 (algorithmic table location CSR) contains address bits 32:28 and IMEM or EMEM select.
uint32_t	address_bits:24	Address bits 27:4.
uint32_t	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 3.1037. union mem\_lookup\_dlut\_large\_recursive\_t**

Type	Name	Description
uint32_t	further_lookup_required:1	If further lookup is required.
uint32_t	direct_lookup:1	1 = direct lookup.
uint32_t	base_address:15	Base address at which the first result is located.
uint32_t	reserved_1:2	Reserved.
MEM_LOOKUP_RESULT	result:1	0 = 24 bits, 1 = 32 bits.
uint32_t	mem_location:1	Location where 0 = IMEM and 1 = EMEM.
MEM_LOOKUP_DLUT_LARGE_SIZE	table_size:3	Specified table size.
uint32_t	table_type:1	1 = large table size.

Type	Name	Description
uint32_t	start_bit_position:7	Starting bit position, bit 0 to bit 127.
uint32_t	value	Accessor to entire descriptor structure.

### 3.26.3.3 mem\_lookup\_dlut\_large\_t

Command for direct lookup large table.

**Table 3.1038. union mem\_lookup\_dlut\_large\_t**

Type	Name	Description
uint32_t	internal_memory:1	If internal memory is used.
uint32_t	direct_lookup:1	1 = direct lookup.
uint32_t	base_address:15	Base address at which the first result is located.
uint32_t	reserved_2:2	Reserved.
MEM_LOOKUP_RESULT	result:1	0 = 24 bits, 1 = 32 bits.
uint32_t	reserved_1:1	Reserved.
MEM_LOOKUP_DLUT_LARGE_SIZE	table_size:3	Specified table size.
uint32_t	table_type:1	1 = large table size.
uint32_t	start_bit_position:7	Starting bit position, bit 0 to bit 127.
uint32_t	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 3.1039. union mem\_lookup\_dlut\_small\_recursive\_t**

Type	Name	Description
uint32_t	further_lookup_required:1	If further lookup is required.
uint32_t	direct_lookup:1	1 = direct lookup.
uint32_t	base_address:16	Base address at which the first result is located.
uint32_t	reserved_1:1	Reserved.
MEM_LOOKUP_RESULT	result:1	0 = 24 bits, 1 = 32 bits.
uint32_t	mem_location:1	Location where 0 = IMEM and 1 = EMEM.
MEM_LOOKUP_DLUT_SMALL_SIZE	table_size:3	Specified table size.
uint32_t	table_type:1	0 = small table type: 1K to 128K.
uint32_t	start_bit_position:7	Starting bit position, bit 0 to bit 127.
uint32_t	value	Accessor to entire descriptor structure.

### **3.26.3.5 mem\_lookup\_dlut\_small\_t**

Command for direct lookup small table.

**Table 3.1040. union mem\_lookup\_dlut\_small\_t**

Type	Name	Description
uint32_t	internal_memory:1	If internal memory is used.
uint32_t	direct_lookup:1	1 = direct lookup.
uint32_t	base_address:16	Base address at which the first result is located.
uint32_t	reserved_2:1	Reserved.
MEM_LOOKUP_RESULT	result:1	0 = 24 bits, 1 = 32 bit.
uint32_t	reserved_1:1	Reserved.
MEM_LOOKUP_DLUT_SMALL_SIZE	table_size:3	Specified table entry size.
uint32_t	table_type:1	0 = small table type: 1K to 128K.
uint32_t	start_bit_position:7	Starting bit position, bit 0 to bit 127.
uint32_t	value	Accessor to entire descriptor structure.

### **3.26.3.6 mem\_lookup\_hash\_table\_t**

Command for hash lookup table.

**Table 3.1041. union mem\_lookup\_hash\_table\_t**

Type	Name	Description
uint32_t	internal_memory:1	If internal memory is used: 1 = IMEM 0=EMEM.
uint32_t	direct_lookup:1	0 = not a direct lookup.
uint32_t	hash_lookup:1	1 = hash lookup.
uint32_t	base_address:17	17-bit base address.
uint32_t	reserved_1:1	Reserved.
MEM_LOOKUP_HASH_TABLE_SIZE	table_size:3	Specified table size.
uint32_t	hash_command:6	6-bit hash command.
MEM_LOOKUP_HASH_STARTING_BIT	start_bit_position:2	Starting bit position.
uint32_t	value	Accessor to entire descriptor structure.

### **3.26.3.7 mem\_lookup\_recursive\_hash\_table\_t**

Command for recursive hash lookup table.

**Table 3.1042. union mem\_lookup\_recursive\_hash\_table\_t**

Type	Name	Description
uint32_t	further_lookup_required:1	1 = further lookup required.
uint32_t	direct_lookup:1	0 = not a direct lookup.
uint32_t	hash_lookup:1	1 = hash lookup.
uint32_t	base_address:17	17-bit base address.
uint32_t	mem_location:1	Location where 0 = IMEM and 1 = EMEM.
MEM_LOOKUP_HASH_TABLE_SIZE	table_size:3	Specified table size.
uint32_t	hash_command:6	6-bit hash command.
MEM_LOOKUP_HASH_STARTING_BIT	start_bit_position:2	Starting bit position.
uint32_t	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 3.1043. typedef MEM\_LOOKUP\_DLUT\_SMALL\_SIZE**

Type	Definition
MEM_LOOKUP_DLUT_SMALL_SIZE	enum MEM_LOOKUP_DLUT_SMALL_SIZE

### 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 3.1044. `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 3.1045. `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 3.1046. `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 3.1047. `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 3.1048. `typedef mem_lookup_dlut_large_t`**

Type	Definition
<code>mem_lookup_dlut_large_t</code>	<code>union mem_lookup_dlut_large_t</code>

### 3.26.4.7 `mem_lookup_dlut_small_recursive_t`

Command for recursive 32-bit direct lookup small table.

**Table 3.1049. `typedef mem_lookup_dlut_small_recursive_t`**

Type	Definition
<code>mem_lookup_dlut_small_recursive_t</code>	<code>union mem_lookup_dlut_small_recursive_t</code>

### 3.26.4.8 `mem_lookup_dlut_large_recursive_t`

Command for recursive 32-bit direct lookup large table.

**Table 3.1050. `typedef mem_lookup_dlut_large_recursive_t`**

Type	Definition
<code>mem_lookup_dlut_large_recursive_t</code>	<code>union mem_lookup_dlut_large_recursive_t</code>

### 3.26.4.9 `mem_lookup_result_t`

**Table 3.1051. `typedef mem_lookup_result_t`**

Type	Definition
<code>mem_lookup_result_t</code>	<code>struct mem_lookup_result_t</code>

### 3.26.4.10 `mem_lookup_result_in_read_reg_t`

**Table 3.1052. `typedef mem_lookup_result_in_read_reg_t`**

Type	Definition
<code>mem_lookup_result_in_read_reg_t</code>	<code>__xread mem_lookup_result_t</code>

### 3.26.4.11 `MEM_LOOKUP_ALUT_SIZE`

Algorithmic lookup table sizes in cache lines where 16 bytes (128-bit) per cache line.

**Table 3.1053. `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 3.1054. `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 3.1055. `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 3.1056. `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 3.1057. `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 3.1058. `typedef MEM_LOOKUP_HASH_COMMANDS`**

Type	Definition
<code>MEM_LOOKUP_HASH_COMMANDS</code>	<code>enum MEM_LOOKUP_HASH_COMMANDS</code>

### 3.26.4.17 `mem_lookup_hash_table_t`

Command for hash lookup table.

**Table 3.1059. `typedef mem_lookup_hash_table_t`**

Type	Definition
<code>mem_lookup_hash_table_t</code>	<code>union mem_lookup_hash_table_t</code>

### 3.26.4.18 `mem_lookup_recursive_hash_table_t`

Command for recursive hash lookup table.

**Table 3.1060. `typedef mem_lookup_recursive_hash_table_t`**

Type	Definition
<code>mem_lookup_recursive_hash_table_t</code>	<code>union mem_lookup_recursive_hash_table_t</code>

## 3.26.5 MU Lookup Functions

### 3.26.5.1 `cmd_mem_lookup`

**Prototype:**

```
mem_lookup_result_in_read_reg_t* cmd_mem_lookup(__xwrite void* xfer, volatile void __addr40
__mem* address, uint32_t 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 3.1061. `cmd_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"> <li>• <code>mem_lookup_dlut_small_t</code> or <code>mem_lookup_dlut_large_t</code> for DLUT tables,</li> <li>• <code>mem_lookup_alut_t</code> for ALUT tables,</li> <li>• <code>mem_lookup_recursive_hash_table_t</code> for HASH tables.</li> </ul>
<code>uint32_t</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 cmd\_pcie\_read\_ptr32

**Prototype:**

```
void cmd_pcie_read_ptr32(__xread void* data, volatile void* address, uint32_t count,  
sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

32-bit read from PCIe into transfer registers.

**Table 3.1062. cmd\_pcie\_read\_ptr32 parameters**

Type	Name	Description
__xread void*	data	Address to read data into
volatile void*	address	Address to read data from (aligned on 4-byte boundary)
uint32_t	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

#### 3.27.1.2 cmd\_pcie\_read\_ind\_ptr32

**Prototype:**

```
void cmd_pcie_read_ind_ptr32(__xread void* data, volatile void* address, uint32_t max_nn,  
generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

Read data from PCIe into transfer register indirect mode.

**Table 3.1063. cmd\_pcie\_read\_ind\_ptr32 parameters**

Type	Name	Description
__xread void*	data	Address to read data into
volatile void*	address	Address to read data from (aligned on 4-byte boundary)

Type	Name	Description
uint32_t	<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)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

### 3.27.1.3 cmd\_pcie\_write\_ptr32

**Prototype:**

```
void cmd_pcie_write_ptr32(__xwrite void* data, volatile void* address, uint32_t count,
                           sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

32-bit write to PCIe from transfer register.

**Table 3.1064. cmd\_pcie\_write\_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)
uint32_t	<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

### 3.27.1.4 cmd\_pcie\_write\_ind\_ptr32

**Prototype:**

```
void cmd_pcie_write_ind_ptr32(__xwrite void* data, volatile void* address, uint32_t
                               max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

Write data to PCIe from transfer register indirect mode.

**Table 3.1065. cmd\_pcie\_write\_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)
uint32_t	<i>max_nn</i>	Maximum number of 32-bit words to write (1-32)
generic_ind_t	<i>ind</i>	Indirect word

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

### 3.27.1.5 cmd\_pcie\_read\_pci\_ptr32

**Prototype:**

```
void cmd_pcie_read_pci_ptr32(__xread void* data, volatile void* address, uint32_t count,
sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

Read 4-byte words from PCIe internal into transfer register.

**Table 3.1066. cmd\_pcie\_read\_pci\_ptr32 parameters**

Type	Name	Description
__xread void*	<i>data</i>	Address to read data into
volatile void*	<i>address</i>	Address to read from
uint32_t	<i>count</i>	Number of 32-bit words to read (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

### 3.27.1.6 cmd\_pcie\_read\_pci\_ind\_ptr32

**Prototype:**

```
void cmd_pcie_read_pci_ind_ptr32(__xread void* data, volatile void* address, uint32_t
max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

Read data from PCIe internal to transfer register indirect mode.

**Table 3.1067. cmd\_pcie\_read\_pci\_ind\_ptr32 parameters**

Type	Name	Description
__xread void*	<i>data</i>	Address to read data into
volatile void*	<i>address</i>	Address to read data from
uint32_t	<i>max_nn</i>	Maximum number of 32-bit words to read (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

### 3.27.1.7 cmd\_pcie\_write\_pci\_ptr32

**Prototype:**

```
void cmd_pcie_write_pci_ptr32(__xwrite void* data, volatile void* address, uint32_t count,
sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

Write 4-byte words to PCI internal from transfer register.

**Table 3.1068. cmd\_pcie\_write\_pci\_ptr32 parameters**

Type	Name	Description
__xwrite void*	data	Data to write to PCIe internal
volatile void*	address	Address to write data to (aligned on 4-byte boundary)
uint32_t	count	Number of 32-bit words to write (1-16).
sync_t	sync	Type of synchronization to use (sig_done or ctx_swap)
SIGNAL*	sig_ptr	Signal to raise upon completion

### 3.27.1.8 cmd\_pcie\_write\_pci\_ind\_ptr32

**Prototype:**

```
void cmd_pcie_write_pci_ind_ptr32(__xwrite void* data, volatile void* address, uint32_t
max_nn, generic_ind_t ind, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

Write data to PCI internal from transfer register indirect mode.

**Table 3.1069. cmd\_pcie\_write\_pci\_ind\_ptr32 parameters**

Type	Name	Description
__xwrite void*	data	Data to write to PCIe internal
volatile void*	address	Address to write data to
uint32_t	max_nn	Maximum number of 32-bit words to write (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

### 3.27.1.9 cmd\_pcie\_read\_rid\_ptr32

**Prototype:**

```
void cmd_pcie_read_rid_ptr32(__xread void* data, uint32_t requester_id, volatile void*
address, uint32_t 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.

**Table 3.1070. cmd\_pcie\_read\_rid\_ptr32 parameters**

Type	Name	Description
__xread void*	<i>data</i>	Address to read data into
uint32_t	<i>requester_id</i>	PCIe requester ID to override
volatile void*	<i>address</i>	Address to read data from (aligned on 4-byte boundary)
uint32_t	<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

### 3.27.1.10 cmd\_pcie\_read\_rid\_ind\_ptr32

**Prototype:**

```
void cmd_pcie_read_rid_ind_ptr32(__xread void* data, volatile void* address, uint32_t
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.

**Table 3.1071. cmd\_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)
uint32_t	<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)
SIGNAL*	<i>sig_ptr</i>	Signal to raise upon completion

### 3.27.1.11 cmd\_pcie\_write\_rid\_ptr32

**Prototype:**

```
void cmd_pcie_write_rid_ptr32(__xwrite void* data, uint32_t requester_id, volatile void*
address, uint32_t 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.

**Table 3.1072. cmd\_pcie\_write\_rid\_ptr32 parameters**

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Data to write to PCIe
<code>uint32_t</code>	<code>requester_id</code>	PCIe requester ID to override
<code>volatile void*</code>	<code>address</code>	Address to write data to (aligned on 4-byte boundary)
<code>uint32_t</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

### 3.27.1.12 cmd\_pcie\_write\_rid\_ind\_ptr32

**Prototype:**

```
void cmd_pcie_write_rid_ind_ptr32(__xwrite void* data, volatile void* address, uint32_t
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.

**Table 3.1073. cmd\_pcie\_write\_rid\_ind\_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>uint32_t</code>	<code>max_nn</code>	Maximum number of 32-bit words to write (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

### 3.27.1.13 cmd\_pcie\_read\_ptr40

**Prototype:**

```
void cmd_pcie_read_ptr40(__xread void* data, volatile void __addr40 *address, uint32_t
count, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

read from PCIe into transfer registers, 40-bit addr

**Table 3.1074. cmd\_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>uint32_t</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

### 3.27.1.14 cmd\_pcie\_write\_ptr40

**Prototype:**

```
void cmd_pcie_write_ptr40(__xwrite void* data, volatile void __addr40 *address, uint32_t
count, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

write to PCIe from transfer register, 40-bit addr

**Table 3.1075. cmd\_pcie\_write\_ptr40 parameters**

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Data to write to PCIe
<code>volatile void __addr40*</code>	<code>address</code>	Address to write data to (aligned on 4-byte boundary)
<code>uint32_t</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

### 3.27.1.15 cmd\_pcie\_read\_pci\_ptr40

**Prototype:**

```
void cmd_pcie_read_pci_ptr40(__xread void* data, volatile void __addr40 *address, uint32_t
count, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

read from PCIe internal target into transfer registers, 40-bit addr

**Table 3.1076. cmd\_pcie\_read\_pci\_ptr40 parameters**

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Xfer reg to read data into
<code>volatile void __addr40*</code>	<code>address</code>	Address, [9:16] target number, [15:0] target addr
<code>uint32_t</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

### 3.27.1.16 cmd\_pcie\_write\_pci\_ptr40

**Prototype:**

```
void cmd_pcie_write_pci_ptr40(__xwrite void* data, volatile void __addr40 *address,
                           uint32_t count, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

write to PCIe internal target from transfer register, 40-bit addr

**Table 3.1077. cmd\_pcie\_write\_pci\_ptr40 parameters**

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Xfer reg to write to PCIe
<code>volatile void __addr40*</code>	<code>address</code>	Address, [9:16] target number, [15:0] target addr
<code>uint32_t</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

### 3.27.1.17 cmd\_pcie\_read\_rid\_ptr40

**Prototype:**

```
void cmd_pcie_read_rid_ptr40(__xread void* data, uint32_t requester_id, volatile void
                           __addr40 *address, uint32_t 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 3.1078. cmd\_pcie\_read\_rid\_ptr40 parameters**

Type	Name	Description
<code>__xread void*</code>	<code>data</code>	Address to read data into
<code>uint32_t</code>	<code>requester_id</code>	PCIe requester ID to override
<code>volatile void __addr40*</code>	<code>address</code>	40-bit address
<code>uint32_t</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

### 3.27.1.18 cmd\_pcie\_write\_rid\_ptr40

**Prototype:**

```
void cmd_pcie_write_rid_ptr40(__xwrite void* data, uint32_t requester_id, volatile void __addr40 *address, uint32_t 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 3.1079. cmd\_pcie\_write\_rid\_ptr40 parameters**

Type	Name	Description
<code>__xwrite void*</code>	<code>data</code>	Data to write to PCIe
<code>uint32_t</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>uint32_t</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 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 3.1080. 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 3.1081. 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 3.1082. 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 cmd\_cluster\_target\_xpb\_write.

**Table 3.1083. union ila\_config\_control\_register\_format\_t**

Type	Name	Description
uint32_t	reserved_2:11	Reserved.
uint32_t	rx_fifo_pack:1	ILA RX FIFO packing (0=packing enabled / 1=packing disabled).
uint32_t	tx_idle_count:10	ILA tx idle count.
uint32_t	rx_fifo_size:1	ILA Rx FIFO size (0=4KByte / 1=8KByte).
uint32_t	rx_tx_credit_count:1	ILA Tx credit count enabled (0=disabled / 1=enabled).
uint32_t	tcam_vendor_select:1	TCAM vendor select (0=select NetLogic TCAM / 1=reserved).
uint32_t	reserved_1:1	Reserved.
uint32_t	error_poison_flag:1	Enables poisoning of data on receipt of ILA Rx error (0=disabled/1=enabled).
uint32_t	stats_channel_select:1	Select channel statistics (0=channel 0 / 1=channel 1).
uint32_t	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).
uint32_t	cpp_addr_mode:1	C PP address bit mode (0=40-bits / 1=32-bits).
uint32_t	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 3.1084. union ila\_cpp\_to\_ilabar\_register\_format\_t**

Type	Name	Description
uint32_t	reserved:10	Reserved.
ILA_CHANNEL	channel_select:1	Selects which ILA channel to send the command on.
uint32_t	address:21	In 32-bit mode, ILA address bits [47:27].  In 40-bit mode, ILA address bits [47:35].
uint32_t	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 3.1085. union ila\_dma\_command\_register\_format\_t**

Type	Name	Description
uint32_t	xfer_length:12	Length of transfer in bytes.  This should be length - 1.
uint32_t	reserved:4	Reserved.
uint32_t	ila_address_hi:16	Upper 16-bits of 48-bit ILA address.
uint32_t	ila_address_lo:30	Lower 30-bits of 48-bit ILA address.
uint32_t	complete_indication_hi:2	Action on completed command (no signal/generate event/send signal to CPP master).
uint32_t	complete_indication_lo:16	Action on completed command (no signal/generate event/send signal to CPP master).
uint32_t	token:2	CPP token bits for CPP command.
uint32_t	signal_both_on_error:1	On ILA Rx Error signal both odd or even (0=disabled / 1=enabled).
uint32_t	cpp_is_target_64:1	Is CPP target 64 bit (0=32-bit / 1=64-bit).
uint32_t	cpp_target:4	Target id for CPP transaction.
uint32_t	cpp_address_hi:8	CPP bus address - higher 8-bit.
uint32_t	cpp_address_lo:32	CPP bus address - lower 32-bit.
uint32_t	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 (cmd\_ilा\_read\_internal\_ptr40 and cmd\_ilा\_write\_internal\_ptr40).

**Table 3.1086. union ila\_internal\_address\_format\_t**

Type	Name	Description
uint32_t	reserved:12	
uint32_t	target:4	Internal target number.
uint32_t	address:16	Internal target address.
uint32_t	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 3.1087. union ila\_to\_cpp\_bar\_register\_format\_t**

Type	Name	Description
uint32_t	reserved_2:2	Reserved.
uint32_t	target_id:5	CPP target id.
uint32_t	token:2	CPP token field.
uint32_t	length_select:1	CPP length field (0=32-bit increments/1=64-bit increments.
uint32_t	reserved_1:22	Reserved.
uint32_t	value	Accessor to entire structure.

### 3.28.3 ILA Typedefs

#### 3.28.3.1 ILA\_OPERATION\_MODE

Enum for ILA operation mode.

**Table 3.1088. 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 3.1089. typedef ILA\_LOGIC\_RESET**

Type	Definition
ILA_LOGIC_RESET	enum ILA_LOGIC_RESET

#### 3.28.3.3 ILA\_CHANNEL

Enum for ILA channels.

**Table 3.1090. typedef ILA\_CHANNEL**

Type	Definition
ILA_CHANNEL	enum ILA_CHANNEL

#### 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 cmd\_cluster\_target\_xpb\_write.

**Table 3.1091. `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 (cmd\_ila\_read\_internal\_ptr40 and cmd\_ila\_write\_internal\_ptr40).

**Table 3.1092. `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 3.1093. `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 3.1094. `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 3.1095. `typedef ila_dma_command_register_format_t`**

Type	Definition
<code>ila_dma_command_register_format_t</code>	<code>union ila_dma_command_register_format_t</code>

## 3.28.4 ILA Functions

### 3.28.4.1 cmd\_ila\_write\_ptr40

**Prototype:**

```
void cmd_ila_write_ptr40(__xwrite void* data, volatile void __addr40 __mem* address,  
uint32_t count, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

Write data to ILA attached device.

**Table 3.1096. cmd\_ila\_write\_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.
uint32_t	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.

### 3.28.4.2 cmd\_ila\_read\_ptr40

**Prototype:**

```
void cmd_ila_read_ptr40(__xread void* data, volatile void __addr40 __mem* address, uint32_t  
count, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

Read data from ILA attached device.

**Table 3.1097. cmd\_ila\_read\_ptr40 parameters**

Type	Name	Description
__xread void*	data	Xfer register(s) containing data read from memory.
volatile void __addr40 __mem*	address	40 bit address in ILA SRAM to read data from.
uint32_t	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 cmd\_ila\_write\_check\_error\_ptr40

**Prototype:**

```
void cmd_ila_write_check_error_ptr40(__xwrite void* data, volatile void __addr40 __mem*
address, uint32_t 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 3.1098. cmd\_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.
uint32_t	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 cmd\_ila\_read\_check\_error\_ptr40

**Prototype:**

```
void cmd_ila_read_check_error_ptr40(__xread void* data, volatile void __addr40 __mem*
address, uint32_t 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 3.1099. cmd\_ila\_read\_check\_error\_ptr40 parameters**

Type	Name	Description
__xread void*	data	Xfer register(s) containing data read from memory.
volatile void __addr40 __mem*	address	40 bit address in ILA SRAM to read data from.
uint32_t	count	Number of 32-bit words to read (1 - 32).
sync_t	sync	Type of synchronization to use (can only be sig_done).

Type	Name	Description
SIGNAL_PAIR*	<i>sig_pair_ptr</i>	Signal pair to raise upon completion.

### 3.28.4.5 cmd\_ila\_write\_internal\_ptr40

**Prototype:**

```
void cmd_ila_write_internal_ptr40(__xwrite void* data, volatile void __addr40 __mem*
address, uint32_t count, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

Write data to ILA internal target.

**Table 3.1100. cmd\_ila\_write\_internal\_ptr40 parameters**

Type	Name	Description
__xwrite void*	<i>data</i>	Xfer register(s) containing data to write to memory.
volatile void __addr40 __mem*	<i>address</i>	40 bit address in ILA where the lower 32 bits are of format 32 bit address in ILA of type <i>ila_internal_address_format_t</i> . See 6xxx databook for recommended addressing mode for address bits.
uint32_t	<i>count</i>	Number of 32-bit words to read (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.

**See Also:**

- *ila\_internal\_address\_format\_t* for format of 32 bit of the address parameter.

### 3.28.4.6 cmd\_ila\_read\_internal\_ptr40

**Prototype:**

```
void cmd_ila_read_internal_ptr40(__xread void* data, volatile void __addr40 __mem*
address, uint32_t count, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

Read data from ILA internal target.

**Table 3.1101. cmd\_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 32 bit address in ILA of type <code>ila_internal_address_format_t</code> . See <code>ila_internal_address_format_t</code> . See 6xxx databook for recommended addressing mode for address bits.
<code>uint32_t</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 cmd\_nbi\_write

**Prototype:**

```
void cmd_nbi_write(__xwrite void* data, volatile void __addr40 __mem* address, uint32_t count, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

Write data to NBI.

**Table 3.1102. cmd\_nbi\_write 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 NBI. See 6xxx databook for recommended addressing mode for address bits.
<code>uint32_t</code>	<code>count</code>	Number of 64-bit words to read (1 - 16).

Type	Name	Description
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 cmd\_nbi\_read

**Prototype:**

```
void cmd_nbi_read(__xread void* data, volatile void __addr40 __mem* address, uint32_t
count, sync_t sync, SIGNAL* sig_ptr)
```

**Description:**

Read data from ILA internal target.

**Table 3.1103. cmd\_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.
uint32_t	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 3.1104. enum sync\_t**

Name	Description
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 3.1105. enum signal\_t**

Name	Description
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 3.1106. struct SIGNAL\_PAIR**

Type	Name	Description
int32_t	even	
int32_t	odd	

### 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 3.1107. typedef SIGNAL\_MASK**

Type	Definition
SIGNAL_MASK	int32_t

#### 3.31.3.2 SIGNAL

SIGNAL data type is used to declare signal variables.

**Table 3.1108. typedef SIGNAL**

Type	Definition
SIGNAL	<code>__declspec(signal) int32_t</code>

#### 3.31.3.3 SIGNAL\_PAIR

SIGNAL\_PAIR container type for even/odd signal pair.

**Table 3.1109. 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\_same\_ME

**Prototype:**

```
void signal_same_ME(uint32_t sig_no, uint32_t 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 3.1110. signal\_same\_ME parameters**

Type	Name	Description
uint32_t	<i>sig_no</i>	Signal number to raise
uint32_t	<i>ctx</i>	Context in which the signal is to be raised

### 3.31.4.2 signal\_same\_ME\_next\_ctx

**Prototype:**

```
void signal_same_ME_next_ctx(uint32_t sig_no)
```

**Description:**

Signal the same Microengine, next context.

Raise the specified signal number in the next context number of the same Microengine.

**Table 3.1111. signal\_same\_ME\_next\_ctx parameters**

Type	Name	Description
uint32_t	<i>sig_no</i>	Signal number to raise

### 3.31.4.3 signal\_prev\_ME

**Prototype:**

```
void signal_prev_ME(uint32_t sig_no, uint32_t ctx)
```

**Description:**

Signal the previous Microengine.

Raise the specified signal number in the specified context of the previous Microengine.

**Table 3.1112. signal\_prev\_ME parameters**

Type	Name	Description
uint32_t	<i>sig_no</i>	Signal number to raise
uint32_t	<i>ctx</i>	Context in which to raise the signal

### 3.31.4.4 signal\_prev\_ME\_this\_ctx

**Prototype:**

```
void signal_prev_ME_this_ctx(uint32_t sig_no)
```

**Description:**

Signal the previous Microengine, same context.

Raise the specified signal number in the same context number of the previous Microengine.

**Table 3.1113. signal\_prev\_ME\_this\_ctx parameters**

Type	Name	Description
uint32_t	<i>sig_no</i>	Signal number to raise

### 3.31.4.5 signal\_next\_ME

**Prototype:**

```
void signal_next_ME(uint32_t sig_no, uint32_t ctx)
```

**Description:**

Signal the next Microengine.

Raise the specified signal number in the specified context of the next Microengine.

**Table 3.1114. signal\_next\_ME parameters**

Type	Name	Description
uint32_t	<i>sig_no</i>	Signal number to raise
uint32_t	<i>ctx</i>	Context in which to raise the signal

### 3.31.4.6 signal\_next\_ME\_this\_ctx

**Prototype:**

```
void signal_next_ME_this_ctx(uint32_t sig_no)
```

**Description:**

Signal the next Microengine, same context.

Raise the specified signal number in the same context number of the next Microengine.

**Table 3.1115. signal\_next\_ME\_this\_ctx parameters**

Type	Name	Description
uint32_t	<i>sig_no</i>	Signal number to raise

## 3.32 Math Intrinsics

FIX ME doxy\_MATH\_functions.xml

## 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.

FIX ME doxy\_Unaligned\_functions.xml

## 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;           // Violates restr 4.  
    }  
    mem_write64(&rd, &y, 1, sig_done, &s);           // Violates restr 3.  
}
```

## 3.35 Miscellaneous Intrinsics

### 3.35.1 Miscellaneous Enumerations

#### 3.35.1.1 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 3.1116. enum swpack\_t**

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.

#### 3.35.1.2 inp\_state\_t

This enumeration defines the state values that can be tested with the `inp_state_test()` intrinsic.

**Table 3.1117. enum inp\_state\_t**

Name	Description
inp_state_nn_empty	NN_Ring in the neighbor microengine is empty.
inp_state_nn_full	NN_Ring in the neighbor microengine is full.
inp_state_ctm_ring0_status	Indicates if CTM Ring 0 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_0 register.
inp_state_ctm_ring1_status	Indicates if CTM Ring 1 is full or empty.  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.

Name	Description
	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.  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.

Name	Description
	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.  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.

### 3.35.2 Miscellaneous Defines

**Table 3.1118. Miscellaneous Defines**

Defined	Definition
i24_emem_base	"__ADDR_I24_EMEM"
i25_emem_base	"__ADDR_I25_EMEM"
i26_emem_base	"__ADDR_I26_EMEM"
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()
profile_count_stop(a)	_profile_count_stop(a)

### 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 3.1119. union cam\_lookup\_t**

Type	Name	Description
uint32_t	zeros1:20	All zeros.
uint32_t	state:4	CAM entry state.
uint32_t	hit:1	hit (1) or miss (0).
uint32_t	entry_num:4	CAM entry number.
uint32_t	zeros2:3	All zeros.
uint32_t	value	Accessor to all fields simultaneously.

**See Also:**

- `cam_lookup()`, `cam_read_state()`.

### 3.35.4 Miscellaneous Functions

#### 3.35.4.1 nn\_ring\_dequeue

**Prototype:**

`uint32_t 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

#### 3.35.4.2 nn\_ring\_dequeue\_incr

**Prototype:**

```
uint32_t 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

### 3.35.4.3 nn\_ring\_enqueue\_incr

**Prototype:**

```
void nn_ring_enqueue_incr(uint32_t 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 3.1120. nn\_ring\_enqueue\_incr parameters**

Type	Name	Description
uint32_t	val	Value to set next neighbor ring indexed by NN_PUT

### 3.35.4.4 byte\_align\_block\_be

**Prototype:**

```
void byte_align_block_be(uint32_t 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 3.1121. byte\_align\_block\_be parameters**

Type	Name	Description
uint32_t	n_byte_align_oper	Number of byte_align operations to perform on dest and src
void*	dest	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

### 3.35.4.5 byte\_align\_block\_le

**Prototype:**

```
void byte_align_block_le(uint32_t 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 3.1122. byte\_align\_block\_le parameters**

Type	Name	Description
uint32_t	<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

### 3.35.4.6 ffs

**Prototype:**

```
uint32_t ffs(uint32_t 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 3.1123. ffs parameters**

Type	Name	Description
uint32_t	<i>data</i>	Data to examine

### 3.35.4.7 halt

**Prototype:**

```
void halt()
```

**Description:**

Halt the ME.

### 3.35.4.8 bswap

**Prototype:**

```
uint32_t bswap(uint32_t lw)
```

**Description:**

Swap bytes in a long word.

**Table 3.1124. bswap parameters**

Type	Name	Description
uint32_t	lw	Long word to swap bytes in

### 3.35.4.9 bitswap

**Prototype:**

```
uint32_t bitswap(uint32_t lw)
```

**Description:**

Swap bits in each byte of a long word.

**Table 3.1125. bitswap parameters**

Type	Name	Description
uint32_t	lw	Long word where bits in bytes are swapped

### 3.35.4.10 cam\_write

**Prototype:**

```
void cam_write(uint32_t entry_num, uint32_t tag, uint32_t 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 3.1126. cam\_write parameters**

Type	Name	Description
uint32_t	<i>entry_num</i>	CAM entry number to write
uint32_t	<i>tag</i>	Value to set for this CAM entry
uint32_t	<i>cam_state</i>	State to set for this CAM entry

### 3.35.4.11 cam\_lookup

**Prototype:**

```
cam_lookup_t cam_lookup(uint32_t 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 3.1127. cam\_lookup parameters**

Type	Name	Description
uint32_t	<i>tag</i>	The value to lookup in the CAM

### 3.35.4.12 cam\_write\_state

**Prototype:**

```
void cam_write_state(uint32_t entry_num, uint32_t 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 3.1128. cam\_write\_state parameters**

Type	Name	Description
uint32_t	<i>entry_num</i>	CAM entry whose state is set
uint32_t	<i>cam_state</i>	State to set for the CAM entry

### 3.35.4.13 cam\_read\_tag

**Prototype:**

```
uint32_t cam_read_tag(uint32_t 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 3.1129. cam\_read\_tag parameters**

Type	Name	Description
uint32_t	<i>entry_num</i>	CAM entry whose tag is returned

### 3.35.4.14 cam\_read\_state

**Prototype:**

```
cam_lookup_t cam_read_state(uint32_t 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 3.1130. cam\_read\_state parameters**

Type	Name	Description
uint32_t	<i>entry_num</i>	CAM entry whose state is returned

### 3.35.4.15 cam\_clear

**Prototype:**

```
void cam_clear(void)
```

**Description:**

Clear all entries in the CAM.

### **3.35.4.16 multiply\_24x8**

**Prototype:**

```
uint32_t multiply_24x8(uint32_t x, uint32_t y)
```

**Description:**

Multiply 24-bit by 8-bit.

Returns the result of 24-bit x multiplied by 8-bit y.

**Table 3.1131. multiply\_24x8 parameters**

Type	Name	Description
uint32_t	x	24-bit int32_t to multiply
uint32_t	y	8-bit int32_t to multiply

### **3.35.4.17 multiply\_16x16**

**Prototype:**

```
uint32_t multiply_16x16(uint32_t x, uint32_t y)
```

**Description:**

Multiply 16-bit by 16-bit.

Returns the result of 16-bit x multiplied by 16-bit y.

**Table 3.1132. multiply\_16x16 parameters**

Type	Name	Description
uint32_t	x	16-bit int32_t to multiply
uint32_t	y	16-bit int32_t to multiply

### **3.35.4.18 multiply\_32x32\_lo**

**Prototype:**

```
uint32_t multiply_32x32_lo(uint32_t x, uint32_t y)
```

**Description:**

Multiply 32-bit by 32-bit.

Returns the lower 32-bit result of 32-bit x multiplied by 32-bit y.

**Table 3.1133. multiply\_32x32\_lo parameters**

Type	Name	Description
uint32_t	x	32-bit int32_t to multiply
uint32_t	y	32-bit int32_t to multiply

### 3.35.4.19 multiply\_32x32\_hi

**Prototype:**

```
uint32_t multiply_32x32_hi(uint32_t x, uint32_t y)
```

**Description:**

Multiply 32-bit by 32-bit.

Returns the higher 32-bit result of 32-bit x multiplied by 32-bit y.

**Table 3.1134. multiply\_32x32\_hi parameters**

Type	Name	Description
uint32_t	x	32-bit int32_t to multiply
uint32_t	y	32-bit int32_t to multiply

### 3.35.4.20 multiply\_32x32

**Prototype:**

```
uint64_t multiply_32x32(uint32_t x, uint32_t y)
```

**Description:**

Multiply 32-bit by 32-bit.

Returns the higher 32-bit result of 32-bit x multiplied by 32-bit y.

**Table 3.1135. multiply\_32x32 parameters**

Type	Name	Description
uint32_t	x	32-bit int32_t to multiply
uint32_t	y	32-bit int32_t to multiply

### 3.35.4.21 \_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 3.1136. `_set_timestamp` parameters**

Type	Name	Description
<code>__int64</code>	<code>timestamp</code>	Timestamp to set

### 3.35.4.22 `_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.

### 3.35.4.23 `_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 3.1137. `_timestamp_stop` parameters**

Type	Name	Description
<code>__int64</code>	<code>handle</code>	Timestamp handle returned by the <code>_timestamp_start()</code> function

### 3.35.4.24 \_sleep

**Prototype:**

```
void _sleep(uint32_t cycles)
```

**Description:**

Sleep for a number of cycles.



#### Note

The ME timers must be enabled before using the sleep function.

**Table 3.1138. \_sleep parameters**

Type	Name	Description
uint32_t	<i>cycles</i>	Approximate number of cycles to sleep, resolution is 16 cycles and should be less than (1 << 20).

### 3.35.4.25 \_ME

**Prototype:**

```
uint32_t _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

### 3.35.4.26 \_island

**Prototype:**

```
uint32_t _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 \_set\_profile\_count**

**Prototype:**

```
void _set_profile_count(uint32_t profile_count)
```

**Description:**

Sets the local\_csr\_profile\_count field of the local\_csr\_t enum.

**Table 3.1139. \_set\_profile\_count parameters**

Type	Name	Description
uint32_t	<i>profile_count</i>	Profile count to set.

### **3.35.4.28 \_profile\_count\_start**

**Prototype:**

```
uint32_t _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](#)

### **3.35.4.29 \_profile\_count\_stop**

**Prototype:**

```
uint32_t _profile_count_stop(uint32_t 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 3.1140. `_profile_count_stop` parameters**

Type	Name	Description
<code>uint32_t</code>	<code>handle</code>	Handle returned by the <code>_profile_count_start()</code> function that is passed to the <code>_profile_count_stop()</code> function.

**See Also:**

- `_profile_count_start`

### 3.35.4.30 assert\_range

**Prototype:**

```
void assert_range(uint32_t value, uint32_t min_value, uint32_t 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 3.1141. `assert_range` parameters**

Type	Name	Description
<code>uint32_t</code>	<code>value</code>	Value to verify if within range
<code>uint32_t</code>	<code>min_value</code>	The minimum value of the range
<code>uint32_t</code>	<code>max_value</code>	The maximum value of the range

### 3.35.4.31 bit\_test

**Prototype:**

```
int32_t bit_test(uint32_t data, uint32_t bit_pos)
```

**Description:**

Test the bit\_pos in data word and return a 1 if set or 0 if clear.

**Table 3.1142. `bit_test` parameters**

Type	Name	Description
<code>uint32_t</code>	<code>data</code>	Data word to test
<code>uint32_t</code>	<code>bit_pos</code>	Bit position in data to test

### 3.35.4.32 inp\_state\_test

**Prototype:**

```
int32_t 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 3.1143. inp\_state\_test parameters**

Type	Name	Description
inp_state_t	state	State to test

## 4. Technical Support

---

To obtain additional information, or to provide feedback, please email <[support@netronome.com](mailto:support@netronome.com)> or contact the nearest **Netronome** technical support representative.